Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX "Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR'9)"; Cần Thơ, ngày 4-5/8/2016
DOI: 10.15625/vap.2016.00056

MỘT SỐ CẢI TIẾN CHO HỆ TRUY VẤN ẢNH DỰA TRÊN CÂY S-TREE

Văn Thế Thành[1,2], Lê Mạnh Thạnh[2]

[1]Trung tâm Công nghệ Thông tin, Trường Đại học Công nghiệp Thực phẩm Tp.HCM
[2]Khoa Công nghệ Thông tin, Trường Đại học Khoa học, Đại học Huế

vanthethanh@gmail.com, lmthanh@hueuni.edu.vn

TÓM TẮT— *Ảnh số đã trở nên thân thuộc với cuộc sống hàng ngày, nên bài toán truy vấn ảnh phù hợp với nhu cầu xã hội hiện nay. Bài báo tiếp cận xây dựng hệ truy vấn ảnh theo nội dung CBIR (Content-Based Image Retrieval) dựa trên chữ ký nhị phân (binary signature) và cây S-Tree. Để tạo chữ ký nhị phân, chúng tôi ứng dụng phương pháp gom cụm K-mean để tạo dải màu từ tập hình ảnh gồm 36,986 ảnh. Tiếp đến, bài báo thiết kế cấu trúc dữ liệu cây Sig-Tree dựa trên cấu trúc dữ liệu S-Tree, từ đó mô tả các các thao tác trên cây Sig-Tree. Nhằm đánh giá độ tương tự giữa các hình ảnh, bài báo ứng dụng độ đo Hamming, EMD (Earth Mover Distance) trên không gian màu CIE-Lab. Nhằm minh chứng cho lý thuyết đã đề nghị, chúng tôi xây dựng thực nghiệm và đánh giá kết quả trên các tập dữ liệu ảnh gồm: COREL (1,000 ảnh), Wang (10,800 ảnh), Bộ sưu tập ảnh ImgCollect (36,986 ảnh).*

Từ khóa— *CBIR, S-tree, Sig-tree, image retrieval, binary signature.*

I. GIỚI THIỆU

Ngày nay, dữ liệu đa phương tiện (văn bản, hình ảnh, âm thanh, video) được lưu trữ và ứng dụng rộng rãi trong nhiều hệ thống như: hệ thống thông tin WWW, hệ thống thư viện số, hệ thống tra cứu video, hệ thống thông tin địa lý, các nghiên cứu thiên văn học, hệ thống quan sát vệ tinh, hệ thống điều tra hình sự, ứng dụng y sinh, giáo dục đào tạo, giải trí,… [4, 14, 16, 19, 23, 28].

Lyman và cộng sự đã ước tính dung lượng thông tin trên toàn cầu có hơn 4 exabyte (1 exabyte = 1 tỷ gigabyte) trong năm 2000. Theo Hilbert và López ước tính dung lượng thông tin toàn cầu năm 2007 khoảng 1.15 zettabyte (1 zettabyte = 1000 exabyte) [11]. Theo ước tính của Bohn và Short, năm 2008 dung lượng thông tin toàn cầu khoảng 3.6 zettabyte và kích thước gia tăng trong năm 2011 khoảng 1,800 exabyte gấp 700 lần so với dung lượng gia tăng năm 2002 (khoảng 2-3 exabyte) [19]. Theo như số liệu của hiệp hội ACI (*Airports Council International*), trong năm 2014, trung bình mỗi phút có 2.5 triệu nội dung được chia sẻ trên Facebook, có gần 300,000 tin nhắn trên Twitter, có khoảng 220,000 hình ảnh mới trên Instagram, có khoảng 72 giờ nội dung video được đăng tải mới trên YouTube, có gần 50,000 ứng dụng được tải từ Apple, có trên 200 triệu Email mới, có trên $80,000 được mua từ Amazon [2]. Theo như tập đoàn dữ liệu thế giới IDC (*International Data Corporation*), dung lượng dữ liệu gia tăng trong năm 2012 là 2,800 exabyte và ước tính dung lượng gia tăng của năm 2020 là 40 zettabyte [12].

Dữ liệu đa phương tiện, đặc biệt là ảnh số đã trở nên thân thuộc với cuộc sống hàng ngày và được sử dụng trên nhiều thiết bị khác nhau như camera, mobile, smartphone, tablet,… Theo như báo cáo của IDC, trong năm 2015 trên thế giới đã tạo và chia sẻ hơn 1.6 nghìn tỷ hình ảnh, trong đó 70% hình ảnh được tạo ra từ thiết bị mobile [7]. Việc số hóa dữ liệu đa phương tiện đã tạo ra các cơ sở dữ liệu khổng lồ làm cho bài toán tìm kiếm đối tượng trở nên phức tạp và có nhiều thách thức như: phân lớp tự động và truy xuất theo nội dung đối tượng, tạo chỉ mục và truy vấn nhanh các đối tượng liên quan, giảm không gian tìm kiếm,...Hơn nữa, truy vấn hình ảnh tương tự từ tập dữ liệu ảnh lớn là bài toán quan trọng trong lĩnh vực thị giác máy tính [1, 9]. Theo như kết quả khảo sát và dự báo của các nghiên cứu gần đây cho thấy việc tìm kiếm các hình ảnh liên quan với yêu cầu người dùng là bài toán phù hợp với nhu cầu xã hội hiện đại [2].

Phần cơ bản của hệ truy vấn ảnh là tạo chỉ mục (*indexing*) và truy hồi (*retrieval*) nhằm đưa ra các thông tin đáp ứng yêu cầu người dùng tại một thời điểm trong một lĩnh vực cụ thể [18, 19]. Việc thiết kế chỉ mục, xây dựng cấu trúc dữ liệu và đưa ra thuật toán truy vấn chính xác (hoặc gần đúng) với đối tượng truy vấn là trọng tâm của bài toán truy vấn dữ liệu ảnh [22, 28]. Vấn đề đặt ra là xây dựng phương pháp truy vấn ảnh hiệu quả, nghĩa là tìm kiếm nhanh các hình ảnh tương tự trong một tập dữ liệu ảnh lớn. Hơn nữa, hình ảnh là dạng dữ liệu không có cấu trúc vì nội dung của các đối tượng này có tính chất trực quan [1], nên bài toán khai phá dữ liệu ảnh (*image mining*) có nhiều thách thức và là động lực để truy tìm các thông tin hữu ích từ các tập dữ liệu ảnh lớn.

Bài báo này xây dựng hệ truy vấn ảnh theo nội dung CBIR (*Content-Based Image Retrieval*) dựa trên chỉ mục nhị phân gọi là chữ ký nhị phân (*binary signature*). Nhằm thực hiện truy hồi thông tin, bài báo tiếp cận cấu trúc cây *Sig*-Tree nhằm lưu trữ chữ ký nhị phân để thực hiện truy tìm hình ảnh tương tự. Các đóng góp của bài báo gồm: (1) tạo dải màu bằng phương pháp gom cụm K-mean để từ đó tạo chữ ký nhị phân cho hình ảnh; (2) Thiết kế cấu trúc dữ liệu cây *Sig*-Tree và đưa ra các thao tác trên cây *Sig*-Tree gồm: chèn nút, tách nút, xóa phần tử trên nút và tìm kiếm trên cây; (3) Ứng dụng độ đo Hamming và độ đo EMD trên không gian màu CIE-Lab để đánh giá độ tương tự giữa hai hình ảnh qua chữ ký nhị phân; (4) Xây dựng hệ truy vấn ảnh CBIR, từ đó đánh giá và so sánh với các phương pháp khác.

Phần còn lại của bài báo gồm những nội dung như sau: **Phần 2**: Khảo sát các công trình liên quan nhằm đánh giá tính khả dĩ của phương pháp truy vấn ảnh dựa trên chữ ký nhị phân; **Phần 3**: Tạo chữ ký nhị phân dựa trên dải màu và đánh giá độ đo tương tự giữa hai hình ảnh bằng độ đo Hamming và độ đo EMD trên không gian màu CIE-Lab; **Phần 4**: Trình bày cấu trúc dữ liệu cây *Sig*-Tree và các thao tác trên cây để từ đó làm cơ sở thực hiện truy vấn ảnh tương tự; **Phần 5**: Xây dựng ứng dụng và đánh giá kết quả thực nghiệm dựa trên cơ sở lý thuyết đã đề nghị; **Phần 6**: Đưa ra kết luận và hướng phát triển trong tương lai.

II. CÁC CÔNG TRÌNH LIÊN QUAN

Năm 1986, Uwe Deppisch đã giới thiệu cây S-Tree nhằm nâng cao hiệu quả tìm kiếm chữ ký nhị phân và ứng dụng trên các tập dữ liệu lớn [10]. Cây S-Tree là cây cân bằng về chiều cao, tức là các nút lá có cùng một mức. Trong cây S-Tree có thể chứa các chữ ký trùng nhau tương ứng với các đối tượng dữ liệu khác nhau. Đến năm 2003, Yannis Manolopoulos và cộng sự đã phát triển cây S-Tree và ứng dụng truy vấn cho dữ liệu đa phương tiện [15, 27].

Elizabeth Shanthi và cộng sự [24] đã tiếp cận phương pháp truy vấn dữ liệu dựa trên chữ ký nhị phân bằng các cấu trúc dữ liệu khác nhau như SSF (*Sequential Signature File*), BSSF (*Bit-Sliced Signature File*), CBSSF (*Compressed Bit-Sliced Signature File*), S-Tree, SD-Tree,… Trong bài báo này đã thực hiện các thao tác chèn, tìm kiếm và xóa chữ ký trên cây SD-Tree. Trong thực nghiệm của bài báo này đã cho thấy phương pháp truy vấn trên cây SD-Tree cải thiện đáng kể tốc độ truy vấn. Trong phương pháp này, các nút trên cây đều có kích thước cố định để liên kết đến các nút con theo chuỗi bit tiền tố. Các nút lá vẫn có chiều dài cố định và liên kết đến các nút ở mức chữ ký theo giá trị khóa được đánh dấu theo vị trí bit 1. Tuy nhiên, trên cây SD-Tree chỉ tìm kiếm các chữ ký nhị phân theo vị trí của các bit trên chuỗi và không thực hiện truy tìm các chữ ký tương tự theo một độ đo cho trước.

Theo tài liệu [21], J. Platos và cộng sự đã tiếp cận phương pháp tìm kiếm ảnh dựa trên cây S-Tree kết hợp với chữ ký mờ [26]. Hình ảnh được chia thành các khối bằng nhau, chỉ mục của mỗi khối được tạo ra bằng cách xấp xỉ trên một bảng tra cứu; các thành phần của bảng tra cứu này là các chuỗi bit mô tả các điểm ảnh đơn sắc trong không gian màu RGB. Vector đặc tính được tạo thành bằng cách ghép nối tất cả các chỉ mục của các khối trên hình ảnh. Chữ ký mờ là một vector được xây dựng từ giá trị tần suất của các chỉ mục. Việc tra cứu hình ảnh được thực hiện trên cây S-Tree qua phép toán mờ và độ đo Euclidean. Trong phương pháp này phụ thuộc vào kích thước của bảng tra cứu, nếu kích thước bảng tra cứu nhỏ thì có độ chính xác thấp vì chữ ký nhị phân của mỗi khối có thể không tương tự với tất cả các thành phần trong bảng; nếu bảng tra cứu có kích thước lớn thì dẫn đến kích thước chữ ký mờ trở lớn hơn (vì phải bằng kích thước của bảng tra cứu).

Theo tài liệu [26], Vaclav Snasel và cộng sự đã tiếp cận cấu trúc dữ liệu cây S-Tree để lưu trữ chữ ký mờ nhằm truy vấn ảnh tương tự dựa trên độ đo Hamming. Tuy nhiên, phép toán xóa một phần tử trên cây quá phức tạp và có thể tái cấu trúc lại toàn bộ cây S-Tree. Do đó, cần có một phương pháp đơn giản hơn nhưng vẫn không ảnh hưởng đến kết quả truy vấn.

Theo tài liệu [15, 27], Yannis Manolopoulos và cộng sự đã phát triển cấu trúc cây S-Tree và ứng dụng truy vấn ảnh tương tự theo nội dung dựa trên độ đo Hamming. Trong công trình này đã mô tả thực nghiệm và đánh giá kết quả trên bộ dữ liệu ảnh COREL và cho thấy tính hiệu quả về thời gian truy vấn. Trong cấu trúc cây này đưa ra phương pháp liên kết đơn giản từ nút cha đến nút con, do đó khi truy ngược từ nút lá trở về nút gốc sẽ tốn kém nhiều chi phí, đặc biệt là phép chèn vào nút, tách nút và xóa nút sẽ làm thay đổi cấu trúc cây theo hướng gốc. Ngoài ra, trong công trình này không đề cập đến thao tác xóa nút trên cây, phép tách nút dựa trên phương pháp bậc hai và bậc ba từ đó làm cho quá trình tạo cây tốn kém nhiều chi phí.

Trên cơ sở cấu trúc cây S-Tree và SD-Tree, bài báo trình bày cấu trúc cây *Sig*-Tree nhằm đơn giản hóa các thao tác trên cây S-Tree cũng như tăng tính hiệu quả truy vấn các hình ảnh tương tự theo các cụm chữ ký tại các nút lá. Giữa nút cha và nút con được liên kết với nhau nhằm đơn giản hóa thao tác truy ngược từ nút lá đến nút gốc. Dựa trên chữ ký nhị phân đã tạo ra từ dải màu đã có, bài báo trình bày hệ truy vấn ảnh theo nội dung bằng độ đo EMD để từ đó đánh giá phương pháp trên các bộ dữ liệu ảnh mẫu thực nghiệm.

III. CHỮ KÝ NHỊ PHÂN VÀ ĐỘ ĐO TƯƠNG TỰ

A. Tạo dải màu cơ sở

Theo công trình [31], Christian Wengert và cộng sự đã tiếp cận tạo chữ ký ảnh và chữ ký nhị phân dựa trên màu sắc. Trong phương pháp này tạo ra dải màu (*palette*) dựa trên không gian màu CIE-Lab và gom cụm K-mean để từ đó tạo ra chữ ký màu sắc. Trên cơ sở phương pháp này, chúng tôi thực hiện phương pháp gom cụm các điểm ảnh trong không gian màu CIE-Lab nhằm xây dựng các dải màu để làm tiền đề tạo chữ ký nhị phân. Trong thực nghiệm, chúng tôi tạo các dải màu gồm: 32 màu, 64 màu, 128 màu và 256 màu.

Hình 1. Các dải màu để lượng tử hình ảnh, gồm: 32 màu, 64 màu, 128 màu và 256 màu

B. Tạo chữ ký nhị phân dựa trên đặc trưng màu sắc

Nhằm tạo chỉ mục nhị phân cho hình ảnh và lưu trữ chữ ký trên cây *Sig*-Tree, thì hình ảnh được chuyển đổi trở thành chữ ký nhị phân. Với mỗi màu c_j của hình ảnh được mô tả bằng một dãy bit $b_1^j b_2^j ... b_t^j$. Vì vậy mỗi hình ảnh được mô tả thành một dãy bit $S = b_1^1 b_2^1 ... b_t^1 \dots b_1^2 b_2^2 ... b_t^2 \dots b_1^n b_2^n ... b_t^n$. Trên cơ sở tham khảo tài liệu [20], quá trình tạo chữ ký nhị phân của hình ảnh được mô tả theo các bước như sau:

Bước 1. Lượng tử hoá màu sắc của ảnh I trên dải màu $C = \{c_1, c_2, ..., c_n\}$, vector histogram màu của ảnh I là:

$$H_I = (h_1^I, h_2^I, ..., h_{n_I}^I) \tag{3.1}$$

Bước 2. Thực hiện chuẩn hoá histogram của ảnh I trên dải màu C được vector histogram chuẩn hoá $H = (h_1, h_2, ..., h_n)$, với mỗi giá trị $h_i \in [0,1]$ được chuẩn hoá theo công thức:

$$h_i = \frac{h_j^I}{\sum_j h_J^I} \tag{3.2}$$

Bước 3. Mỗi màu c_j^I mô tả thành dãy bit có chiều dài m là $b_1^j b_2^j, ..., b_m^j$, do đó chữ ký nhị phân của ảnh I là:

$$Sig(I) = b_1^1 b_2^1, ..., b_m^1 b_1^2 b_2^2, ..., b_m^2 ... b_1^n b_2^n, ..., b_m^n \tag{3.3}$$

trong đó: $b_i^j = \begin{cases} 1 & i = \lceil h_i \times m \rceil \\ 0 & i \neq \lceil h_i \times m \rceil \end{cases}$

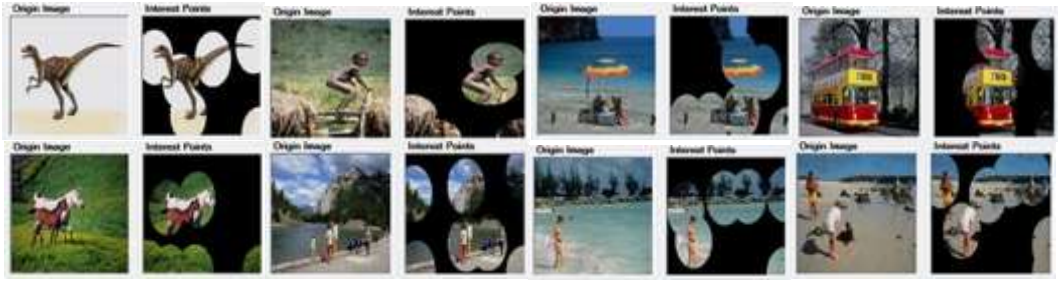
Đặt $B^j = b_1^j b_2^j ... b_m^j$, chữ ký nhị phân của ảnh I là:

$$SIG = B^1 B^2 ... B^n \tag{3.4}$$

C. Tạo chữ ký nhị phân dựa trên đặc trưng SIFT và màu sắc

Bài toán đặt ra là cần nhận diện được tập các điểm đặc trưng trên hình ảnh để từ đó chọn vùng đặc trưng tương ứng. Dựa trên các vùng đặc trưng đã được nhận diện này, thực hiện mô tả chữ ký nhị phân để làm cơ sở cho việc đối sánh hình ảnh. Có nhiều phương pháp dò tìm đặc trưng thông dụng đã được giới thiệu [30], 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 Laplacian 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 Laplacian dựa trên toán tử Laplacian của hàm Gauss được giới thiệu năm 2001 bởi Mikolajczyk và C.Schmid,… Phương pháp tìm điểm đặc trưng Harris Laplacian có thể áp dụng cho ảnh màu và bất biến đối với sự biến đổi cường độ ảnh cũng như bất biến đối với các phép biến đổi tỉ lệ, phép quay, phép biến đổi affine. Vì vậy, bài báo tiếp cận phương pháp tìm đặc trưng SIFT dựa trên phương pháp tìm Harris Laplacian và áp dụng cho ảnh màu. Từ đó làm cơ sở tạo ra chữ ký nhị phân mô tả các vùng đặc trưng tương ứng với các điểm đặc trưng đã có.

Sau khi đã có đặc trưng SIFT của hình ảnh, chúng tôi thực hiện tạo chữ ký nhị phân màu sắc của vùng đặc trưng. Việc tạo chữ ký nhị phân này tương tự như phần **III.*B*** đã trình bày như trên.



Hình 2. Minh họa về trích xuất đặc trưng SIFT theo phương pháp Harris Laplacian

D. Độ đo Hamming áp dụng cho chữ ký nhị phân

Gọi SIG_I và SIG_J lần lượt là hai chữ ký nhị phân của hai hình ảnh I và J. Độ trùng khớp d_i được đối sánh trên mỗi phần tử của hai chữ ký dựa trên độ đo Hamming được mô tả như sau:

$$d_i = \begin{cases} 1 & if\ (sig_i^I = sig_i^J) \\ 0 & if\ (sig_i^I \neq sig_i^J) \end{cases} \tag{3.5}$$

Độ đo tương tự của hai hình ảnh I và J được định nghĩa là:

$$\phi = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{3.6}$$

E. Độ đo EMD áp dụng cho chữ ký nhị phân

Vì khoảng cách EMD được dùng để đánh giá sự khoảng cách giữa hai phân bố, nên phù hợp cho việc đánh giá sự phân bố màu sắc giữa các thành phần của hai hình ảnh [32]. Hơn nữa, khoảng cách Euclidean của không gian màu CIE-Lab đồng nhất với nhận thức của con người [1, 17]. Vì vậy, chúng tôi ứng dụng khoảng cách EMD và không gian màu CIE-Lab để đánh giá độ tương tự giữa hai hình ảnh.

Xét hình ảnh I có chữ ký nhị phân mô tả màu sắc là $SIG_I = B_I^1 B_I^2 ... B_I^n$, trọng số của thành phần B_I^j là:

$$w_I^j = w(B_I^j) = \sum_{i=1}^{m} (b_i^j \times \frac{i}{m} \times 100) \text{, với } B_I^j = b_1^j b_2^j ... b_m^j \tag{3.7}$$

Do đó, vector trọng số của hình ảnh I là:

$$W_I = (w_I^1, w_I^2, ..., w_I^n) \tag{3.8}$$

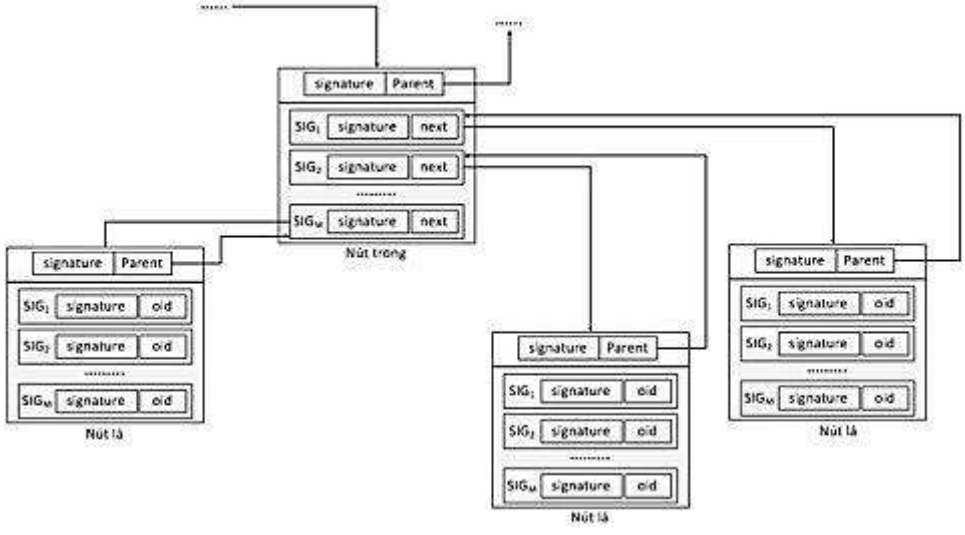
Gọi J là hình ảnh cần tính độ tương tự so với ảnh I, do đó cần cực tiểu hoá chi phí chuyển đổi phân bố màu sắc là: $\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} f_{ij}$, với $F = \left(f_{ij}\right)$ là ma trận phân phối luồng màu sắc từ màu c_I^i đến màu c_J^j và $D = \left(d_{ij}\right)$ là ma trận khoảng cách Euclidean trong không gian màu CIE-Lab từ màu c_I^i đến màu c_J^j. Khi đó, độ đo tương tự giữa hai hình ảnh I và J dựa trên khoảng cách EMD là cực tiểu hoá giá trị sau:

$$EMD(I,J) = \min_{F=(f_{ij})} \frac{(\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} f_{ij})}{\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}} \text{, với } \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} = \min(\sum_{i=1}^{n} w_I^j, \sum_{j=1}^{n} w_J^j) \tag{3.9}$$

IV. THIẾT KẾ CẤU TRÚC DỮ LIỆU CÂY *Sig*-Tree

A. Cấu trúc dữ liệu cây Sig-Tree

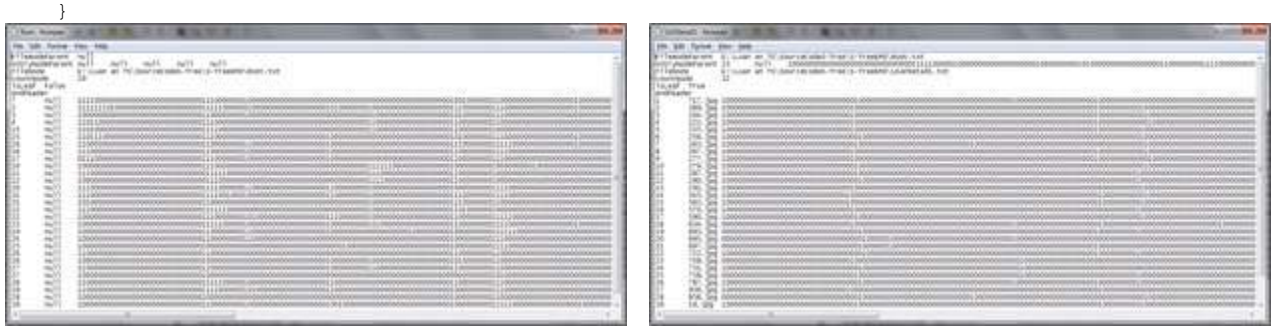
Tương tự như cây S-Tree, trong **Hình 3** mô tả cấu trúc cây *Sig*-Tree cũng gồm một nút gốc, các nút trong và các nút lá. Để dễ dàng đối sánh với các thành phần trong một nút, mỗi nút trong của cây *Sig*-Tree được chia thành hai phần gồm: (1) phần liên kết đến nút cha được ký hiệu $\langle signature, parent \rangle$ mô tả chữ ký tổ hợp trong nút và liên kết ngược về nút cha; (2) phần nội dung là tập các chữ ký $\{SIG_i = \langle signature_i, next_i \rangle \mid i = 1,...,k\}$ mô tả các chữ ký của mỗi thành phần và liên kết đến các nút con. Mỗi nút lá cũng gồm hai phần là: phần liên kết nút cha $\langle signature, parent \rangle$ và phần nội dung $\{SIG_i = \langle signature_i, oid_i \rangle \mid i = 1,...,k\}$ mô tả chữ ký nhị phân của hình ảnh thứ i và mã số tương ứng.



Hình 3. Minh họa cấu trúc dữ liệu cây *Sig*-Tree

Trong thực nghiệm, mỗi nút trong cây *Sig*-Tree được lưu trữ dưới dạng tập tin văn bản và được liên kết nhau để tạo thành một cấu trúc cây (được mô tả ở **Hình 3** và **Hình 4**). Cấu trúc dữ liệu cây *Sig*-Tree được cài đặt như sau:

```
struct FileEntryNode
 {   int Pos;              //Vị trí của phần tử trong nút
     string oid;           //Mã số của hình ảnh, nếu là nút trong thì oid = null
     string signature;     //Chữ ký của một thành phần trong nút
     string FileNext ;     //Liên kết đến tập tin chứa nút con
     string Filename;      //Mô tả tập tin chứa nút hiện hành
 }
 struct StreeNodeHeader
 {   string FileNodeParent;          //Liên kết đến tập tin chứa nút cha
     FileEntryNode EntryNodeParent;  //Mô tả liên kết đến nút cha
     string FileNode;                //Mô tả tập tin chứa nút hiện hành
     int CountNode;                  //Số lượng chữ ký trong nút
     bool isLeaf;                   // Dùng để kiểm tra nút lá
 }
 struct FileStreeNode
 {   StreeNodeHeader Header;         // Phần liên kết đến nút cha
     List<FileEntryNode> ListEntry; // Phần mô tả nội dung của nút
 }
```



Hình 4. Minh họa một nút gốc và một nút lá trong cây *Sig*-Tree

B. Phép tổ hợp các chữ ký trên cây Sig-Tree

Theo cấu trúc cây *Sig*-Tree, mỗi phần tử của một nút cha đều là tổ hợp của các chữ ký của nút con. Do đó, nếu các thành phần của nút con thay đổi thì phải tổ hợp lại các chữ ký của nút cha và thực hiện cho đến nút gốc. Thao tác này tương đối đơn giản nhưng thường xuyên thực hiện vì các thao tác trên cây đều ảnh hưởng đến các thành phần trong một nút. Thuật toán tổ hợp các nút trên cây được mô tả như sau:

Thuật toán 1: Tổ hợp chữ ký
***Đầu vào*:** Nút *v* cần tổ hợp
***Đầu ra*:** Cây *Sig*-Tree sau khi tổ hợp
Begin
  If v_parent ≠ null **then**
    Signature = ∨(SIG$_i$→sig), với SIG$_i$ ∈ v;
    v_parent → (SIG$_i$→ sig) = Signature;
    Gọi đệ quy **Thuật toán 1** ứng với v_parent;
  EndIf
End

C. Phép tách một nút trên cây Sig-Tree

Việc tạo cây *Sig*-Tree thực hiện bằng cách chèn từng chữ ký vào cây, nếu một nút trong cây bị đầy thì phải tách thành hai nút. Theo như tài liệu [5] và [6], phép tách một nút trong cây S-Tree có độ phức tạp là $O(l^2)$ và $O(l^3)$ tương ứng với thuật toán tách bậc hai (*quadratic split*) và bậc ba (*cubic algorithm*), với l là số chữ ký của nút cần tách. Yannis Manolopoulos và cộng sự đã đề xuất phương pháp tách bậc hai có độ phức tạp $O(l^2)$, tách bậc ba có độ phức tạp $O(l^3)$, tách nút dựa trên phân cụm phân cấp có độ phức tạp là $O(l^2)$ [15, 27]. Trong phương pháp tách bậc hai thực hiện phân phối từng chữ ký vào hai nút ứng với hai chữ ký α và β đã chọn trước; mỗi lần thực hiện phân phối, chọn chữ ký có độ sai biệt trọng số ε lớn nhất. Việc chọn lựa các chữ ký để phân phối tạo ra sự phân biệt giữa các chữ ký trong hai nút đã được tách. Đối với phương pháp tách phân cụm phân cấp được thực hiện dựa trên thao tác trộn hai cụm chữ ký có độ sai biệt gần nhất để kết quả cuối cùng có được hai cụm phân biệt tương ứng với hai nút đã được tách.

Tuy nhiên, thứ tự chọn lựa các chữ ký để chèn vào các nút tách không ảnh hưởng đến việc tìm kiếm chữ ký trên cây vì trong quá trình tìm kiếm vẫn phải kiểm tra tất cả các chữ ký tại mỗi nút trong cây *Sig*-Tree. Hơn nữa, nếu sử dụng độ đo d giữa hai chữ ký và ứng dụng thuật toán phân cụm không phân cấp K-mean với số cụm là $k = 2$ thì vẫn đảm bảo tách thành hai nút riêng biệt. Kết quả của quá trình tách nút này là tạo thành hai cụm chữ ký rời nhau tương ứng với hai tâm là α và β, mỗi cụm tương ứng với một nút trong cây *Sig*-Tree. Khi đó, độ phức tạp của thuật toán chỉ còn $O(l)$ và được thực hiện như sau:

Thuật toán 2: Tách một nút
***Đầu vào*:** Tập các chữ ký *S* của nút cần tách
***Đầu ra*:** Hai nút đã được tách gồm *NodeA* và *NodeB*

```
Begin
        Chọn hai chữ ký α và β theo phương pháp tuyến tính [15];
        NodeA = {α}; NodeB = {β};
        Foreach s_i ∈ S do
                Tính độ đo d(α,s_i) và d(β,s_i);
                If d(α,s_i) < d(β,s_i) then
                        Chèn chữ ký s_i vào NodeA;
                ElseIf
                        Chèn chữ ký s_i vào NodeB;
                EndIf
        EndFor
End
```

Khi tách một nút trở thành hai nút cần bổ sung hai chữ ký tổ hợp của hai nút này vào nút cha. Nếu việc bổ sung này làm cho nút cha trở nên đầy thì phải tiếp tục thực hiện tách nút, do đó cây *Sig*-Tree tăng trưởng theo hướng gốc và chiều cao của tất cả nút lá luôn bằng nhau. Thuật toán tách một nút trong cây *Sig*-Tree được đề xuất như sau:

```
Thuật toán 3: Tách nút trong cây Sig-Tree
Đầu vào: Nút cần tách v
Đầu ra: Cây Sig-Tree sau khi được tách
Begin
        Tạo tập chữ ký S của nút v;
        Tạo hai nút tách NodeA và NodeB bằng Thuật toán 2;
        Tạo hai chữ ký tổ hợp SigA, SigB của hai nút NodeA, NodeB;
        Gán v_parent là nút cha của nút v;
        If v_parent == Null then
                Tạo mới nút gốc Root = {SigA, SigB};
        ElseIf
                Bổ sung chữ ký SigA, SigB vào nút v_parent;
                Tổ hợp các chữ ký từ nút v_parent đến nút gốc;
                If v_parent đầy then
                        Gọi đệ quy Thuật toán 3 ứng với nút v_parent;
                EndIf
        EndIf
End
```

Gọi n là số chữ ký trên cây; M và m lần lượt là số chữ ký nhiều nhất và ít nhất tại mỗi nút; theo tài liệu [15], chiều cao tối đa của cây có thể đạt được là $h = \lceil \log_m n - 1 \rceil$. Trong trường hợp xấu nhất là tách một nút lá của một nhánh cây có tất cả các nút đều bị đầy, tức là mỗi nút còn lại đều chứa M chữ ký. Mỗi lần tách nút có độ phức tạp $O(M)$, nên độ phức tạp trong trường hợp này là $O(M \times \lceil \log_k n - 1 \rceil) \approx O(M \log n)$.

D. Phép xóa chữ ký trên cây Sig-Tree

Phương pháp xóa chữ ký trong cây S-Tree thực hiện tại nút lá và đã được tiếp cận trong tài liệu [10, 26]. Có hai trường hợp như sau: (1) Nếu nút hiện hành có nhiều hơn số phần tử tối thiểu thì chỉ cần loại bỏ phần tử này và tổ hợp lại các chữ ký từ nút hiện hành đến nút gốc; (2) Nếu số lượng phần tử tại nút này sau khi xóa ít hơn số lượng tối thiểu thì phải xóa luôn cả nút hiện hành, sau đó chèn các phần tử còn lại vào cây. Việc xóa một nút trong cây dẫn đến phải xóa đi một phần tử tương ứng của nút cha. Nếu nút cha sau khi xóa có số phần tử ít hơn số lượng tối thiểu thì phải xóa luôn nút cha và thu gom các chữ ký tại nút lá theo nút cha này và thực hiện thao tác chèn trở lại.

Trong trường hợp xấu nhất, thao tác xóa có thể tái tạo lại toàn bộ cây *Sig*-Tree và trở nên không hiệu quả. Một giải pháp đề xuất để giải quyết trong trường hợp này đó là thực hiện gán mã đối tượng $oid = null$ nếu như một nút lá sau khi xóa có số lượng phần tử nhỏ hơn số lượng tối thiểu, tức là không thực hiện xóa phần tử này nhưng không đưa vào kết quả truy vấn. Khi thực hiện thao tác chèn chữ ký tại một nút lá, nếu số lượng phần tử có $oid \neq null$ nhiều hơn số lượng tối thiểu thì thực hiện xóa các phần tử có $oid = null$. Do đó, thuật toán xóa chữ ký tại một nút lá của cây được thực hiện đơn giản như sau:

```
Thuật toán 4: Xóa chữ ký
Đầu vào: Chữ ký Sig vào nút lá v, số chữ ký tối thiểu m
Đầu ra: Cây Sig-Tree sau khi xóa chữ ký
Begin
        Khởi tạo SIG = v.SIG sao cho v.SIG→sig = Sig;
        If (v.count > m) then
                Xóa phần tử v.SIG;
                Thực hiện tổ hợp các chữ ký từ nút v trở về nút gốc;
        ElseIf
                v.SIG → oid = null;
        EndIf
End
```

E. Phép chèn chữ ký trên cây Sig-Tree

Theo như cách tiếp cận của Uwe Deppisch và Vaclav Snasel [8, 26], thao tác chèn chữ ký nhị phân được thực hiện tại nút lá của cây. Bước đầu tiên của thao tác chèn này là tìm ra nút lá phù hợp dựa trên một độ đo cho trước. Nếu thực hiện phép chèn làm cho nút lá bị đầy thì thực hiện phép tách nút. Trong phương pháp tiếp cận của bài báo, tại các

nút lá có thể chứa các phần tử trống (tức là có $oid = null$). Vì vậy, mỗi lần chèn chữ ký vào nút lá thực hiện đếm số phần tử khác trống, nếu vượt qua số lượng tối thiểu thì thực hiện xóa các phần tử trống này, sau đó cập nhật các chữ ký tổ hợp từ nút lá đến nút gốc. Nếu một nút lá bị đầy thì thực hiện phép tách nút theo ***Thuật toán 3*** đã đề nghị như trên. Thuật toán chèn chữ ký vào cây *Sig*-Tree được mô tả như sau:

```
Thuật toán 5: Chèn chữ ký
Đầu vào: Chữ ký Sig vào nút v
Đầu ra: Cây Sig-Tree sau khi chèn chữ ký
Begin
        If (v là nút lá) then
                Chèn chữ ký Sig vào nút v;
                If số lượng phần tử nút v > m then
                        Xóa các phần tử trống trong nút v;
                EndIf
                Thực hiện tổ hợp các chữ ký từ nút v trở về nút gốc;
                If nút v đầy then
                        Tách nút v theo Thuật toán 3;
                EndIf
        ElseIf
                Chọn phần tử SIG ∈ v sao cho độ đo d(SIG.Sig,Sig) nhỏ nhất;
                v = SIG→next;
                Gọi đệ quy Thuật toán 5 ứng với nút v đã cập nhật;
        EndIf
End
```

Giả sử ban đầu xuất phát từ nút gốc của cây *Sig*-Tree. Bước đầu tiên của thao tác chèn là duyệt qua các nút từ gốc đến lá. Trong trường hợp xấu nhất là phải duyệt qua M phần tử tại mỗi nút của cây, tức có độ phức tạp là $O(M \times \lceil \log_k n - 1 \rceil)$. Sau khi thực hiện thao tác chèn thì có thể thực hiện tách nút, do đó độ phức tạp của toàn bộ thao tác chèn một nút trên cây là $O(M \times \lceil \log_k n - 1 \rceil) \times O(M \times \lceil \log_k n - 1 \rceil) \approx O((M \log n)^2)$.

F. Truy vấn trên cây Sig-Tree

Theo như cách tiếp cận của Uwe Deppisch [10] và Yannis Manolopoulos [15], phép tìm kiếm được duyệt theo mức của cây và thực hiện trên cơ sở gọi đệ quy tại mỗi mức. Nhằm đơn giản hóa quá trình tìm kiếm trên cây *Sig*-Tree, bài báo mô tả việc thực hiện tìm kiếm trên cơ sở duyệt các thành phần phù hợp của mỗi nút và đưa vào cấu trúc STACK. Gọi n là số chữ ký trên cây *Sig*-Tree, trong trường hợp xấu nhất có thể đi qua tất cả các hướng trên cây. Độ phức tạp trong trường hợp này là: $O(n \times \lceil \log_k n - 1 \rceil)$. Thuật toán truy tìm các chữ ký tương tự trên cây *Sig*-Tree được mô tả như sau:

```
Thuật toán 6: Truy vấn ảnh trên cây Sig-Tree
Đầu vào: Chữ ký Sig truy vấn và Cây Sig-Tree
Đầu ra: Tập chữ ký và định danh SigOid = {⟨sig_i, oid_i⟩| i = 1,…,p}
Begin
        Khởi tạo:       v = Root; SigOid = ∅; STACK = ∅;
        While(not Empty(STACK)) do
                v = Pop(STACK);
                If (v Không phải nút lá) then
                        Chọn SIG0∈ v sao cho độ đo d(SIG0→sig, Sig) nhỏ nhất;
                        Push(STACK,Sig0→next);
                Else
                        SigOid = SigOid ∪ {⟨v→sig_i, v→oid_i⟩| i = 1,…,|v|};
                EndIf
        EndWhile
End
```

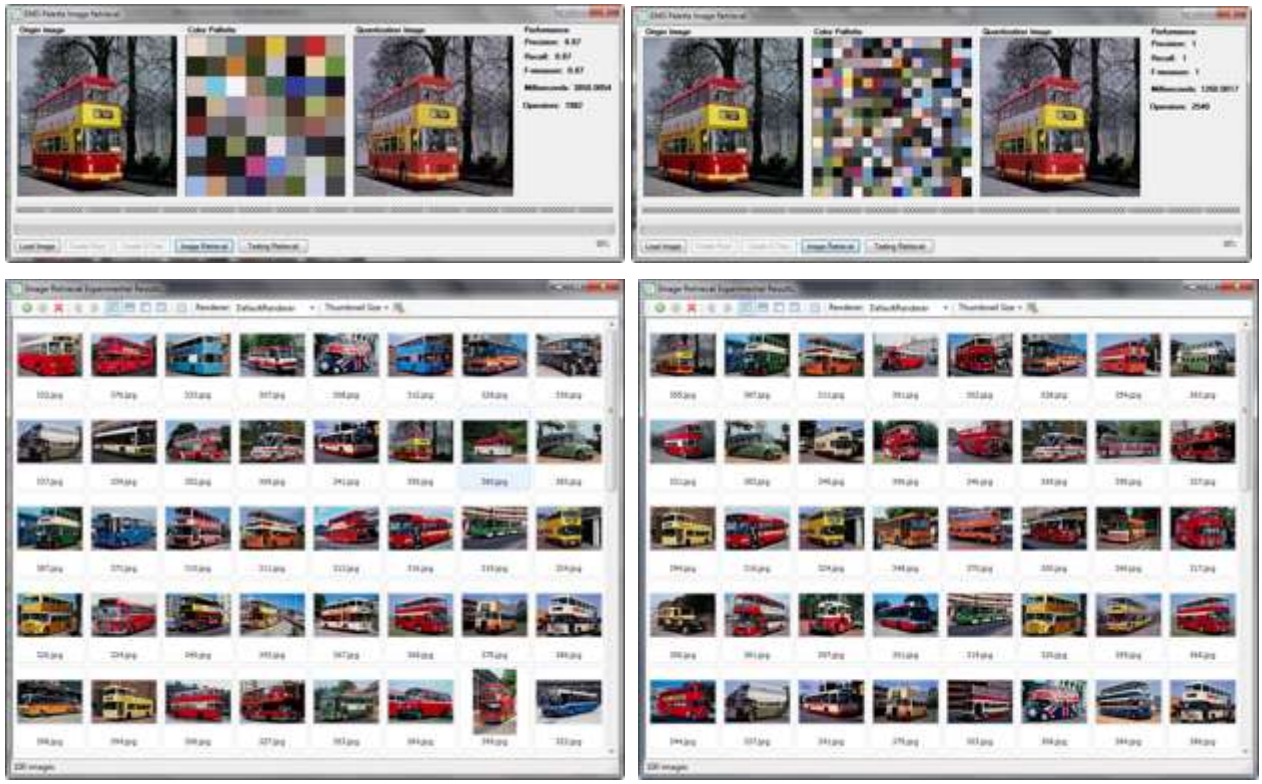
Sau khi tìm được các chữ ký tương tự tại các nút lá của các hình ảnh, dựa trên các mã số oid tương ứng với các chữ ký nhị phân, ta thực hiện truy tìm các hình ảnh tương tự trong tập dữ liệu ảnh tương ứng để có được bộ ảnh tương tự mong muốn.

V. THỰC NGHIỆM

A. Xây dựng ứng dụng thực nghiệm

Ứng dụng thực nghiệm mô tả trong **Hình 5** được chia làm hai giai đoạn gồm: (1) Tạo chỉ mục nhị phân cho tập dữ liệu hình ảnh và lưu trữ lên cây *Sig*-Tree; (2) Thực hiện truy tìm các hình ảnh tương tự sau khi đã có danh sách mã số các hình ảnh tương tự theo nội dung với ảnh truy vấn.

Giai đoạn tiền xử lý được thực nghiệm trên máy tính có bộ xử lý Intel(R) Xeon(R) X3440 @ 2.53GHz x 2, hệ điều hành Windows Server 2008 R2 Enterprise 64-bit, RAM 8.00GB; Giai đoạn truy vấn ảnh được thực thi trên máy tính có bộ xử lý Intel(R) CoreTM i7-2620M, CPU 2.70GHz, RAM 4GB và hệ điều hành Windows 7 Professional. Thực nghiệm được kiểm tra trên các bộ ảnh gồm COREL (1,000 ảnh), Wang (10,800 ảnh), ImgCollect (36,986 ảnh).



Hình 5. Mô tả ứng dụng truy vấn ảnh trên tập ImgCollect (36,896 ảnh) theo các dải màu: 64 màu và 128 màu

B. Kết quả thực nghiệm

Để đánh giá tính hiệu quả của hệ truy vấn CBIR, phần thực nghiệm của bài báo 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 đường cong *recall-precision* và được cong đặc tính ROC (*Receiver Operating Characteristic*). Theo Alzu'bi và Jenni [3, 13], các giá trị đánh giá hiệu suất được mô tả theo công thức như sau:

$$precision = \frac{(relevant\ images \cap retrieved\ images)}{retrieved\ images} \tag{5.1}$$

$$recall = \frac{(relevant\ images \cap retrieved\ images)}{relevant\ images} \tag{5.2}$$

$$F - measure = 2 \times \frac{(precision \times recall)}{(precision + recall)} \tag{5.3}$$

Trong đó, *relevant images* là số lượng ảnh tương tự với ảnh truy vấn có trong tập dữ liệu ảnh; *retrieved images* là số lượng ảnh đã truy vấn được.

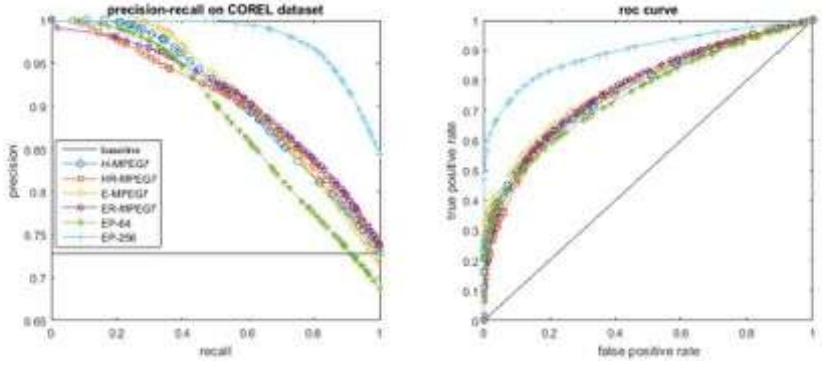
Bảng 1. Đánh giá hiệu suất trung bình của các phương pháp trên các bộ dữ liệu

Tên CSDL	Phương pháp	Thời gian tạo cây (milli giây)	Tổng số phép toán thực hiện tạo cây	Số nút lá của cây	Số lượng mẫu kiểm thử	Tỉ lệ kiểm thử	Độ chính xác trung bình	Độ phủ trung bình	Độ đo F-measure trung bình	Thời gian truy vấn trung bình (milli giây)	Số phép toán thực hiện truy vấn trung bình
COREL	H-MPEG7	10,374.0180	78,926	13	691	69.10%	0.727945007	0.728254451	0.728099696	4.470049638	42.51664255
	HR-MPEG7	5,974.8081	84,305	14	654	65.40%	0.73675841	0.737549361	0.737153673	4.86606208	47.49235474
	E-MPEG7	20,217.6365	8,425,717	16	578	57.80%	0.717266436	0.717538139	0.717402262	3.805543253	43.79238754
	ER-MPEG7	15,958.8275	9,184,228	17	626	62.60%	0.739472843	0.739783073	0.739627926	4.535472364	48.64217252
	EP-64	67,840.0957	25,123,957	22	512	51.20%	0.687480469	0.687487535	0.687484002	8.257046875	59.5859375
	EP-256	**587,575.1129**	**64,311,879**	**14**	**544**	**54.40%**	**0.843602941**	**0.843331798**	**0.843467348**	**26.81254688**	**60.05882353**
WANG	H-MPEG7	56,788.2505	1,630,451	153	2,890	26.76%	0.756657439	0.577469804	0.655030209	39.57224948	337.7806228
	HR-MPEG7	59,261.3933	1,553,154	155	2,611	24.18%	0.738521639	0.578869932	0.649021871	34.62357687	359.2830333
	E-MPEG7	136,801.3689	107,776,364	177	2,500	23.15%	0.736752	0.578271586	0.647962138	36.33558544	345.4544
	ER-MPEG7	148,198.4766	113,271,910	185	2,325	21.53%	0.724	0.566786346	0.635819112	28.65708374	355.2197849
	EP-64	860,989.2461	269,556,572	191	2,736	25.33%	0.802697368	0.658330355	0.723381268	91.95805691	553.3815789
	EP-256	**12,202,465.4815**	**1,123,060,059**	**184**	**4,276**	**39.59%**	**0.908543031**	**0.834190753**	**0.869780803**	**298.5712446**	**692.9471469**
ImgCollect	H-MPEG7	534,160.5293	6,014,781	549	5,473	14.80%	0.889420793	0.542777417	0.674149035	164.9390066	1376.397771
	HR-MPEG7	499,185.2694	6,893,898	715	5,289	14.30%	0.951796181	0.511041921	0.66501925	153.9208601	1518.078843
	E-MPEG7	1,105,698.7514	399,628,572	627	3,286	8.88%	0.857577602	0.532540704	0.657059156	137.257453	1284.471394
	ER-MPEG7	1,002,223.7651	398,839,670	623	4,581	12.39%	0.871731063	0.59175723	0.704963834	130.9673552	1317.882558
	EP-64	4,207,389.7730	931,274,257	647	6,348	17.16%	0.899774732	0.676759198	0.772493144	353.9520398	2045.435885
	EP-256	**48,544,118.4751**	**3,817,477,152**	**596**	**12,220**	**33.04%**	**0.955059738**	**0.859059351**	**0.904519448**	**1128.856025**	**2423.78036**

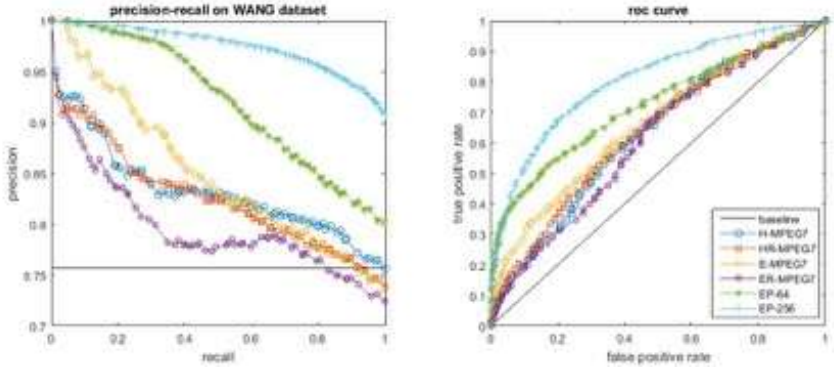
Trong thực nghiệm, bài báo thực hiện truy vấn trên các tập ảnh gồm COREL, Wang, ImgCollect ứng với 6 phương pháp truy vấn ảnh dựa trên chữ ký nhị phân gồm: (1) Sử dụng độ đo Hamming kết hợp dải màu MPEG7 (H-MPEG7); (2) Trích xuất vùng cục bộ và sử dụng độ đo Hamming kết hợp dải màu MPEG7 (HR-MPEG7); (3) Sử dụng độ đo EMD kết hợp dải màu MPEG7 (E-MPEG7); (4) Trích xuất vùng đặc trưng và sử dụng độ đo EMD kết hợp với dải màu MPEG7 (ER-MPEG7); (5) Sử dụng độ đo EMD kết hợp với 64 màu (EP-64); (6) Sử dụng độ đo EMD kết hợp với 256 màu (EP-256).

Trong **Bảng 1** đã cho thấy phương pháp EP-256 có độ chính xác cao hơn các phương pháp còn lại. Từ đó cho thấy sự hiệu quả của phương pháp truy vấn ảnh dựa trên cây *Sig*-Tree đã được đề xuất. Thời gian truy vấn của các phương pháp được đo trên milli giây, từ đó tính giá trị trung bình. Từ **Bảng 1** cho thấy, thời gian truy vấn trung bình của phương pháp EP-256 trên tập ảnh COREL là 26.81254688 milli giây, trên tập ảnh Wang là 692.9471469 milli giây và trên tập ảnh ImgCollect là 2423.78036 milli giây.

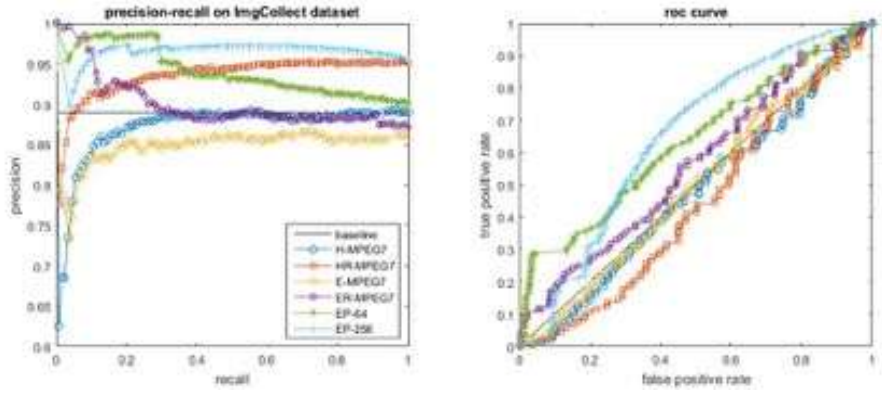
Thời gian tạo cây *Sig*-Tree trong phương pháp EP-64 và EP-256 lâu hơn các phương pháp khác. Đối với tập dữ liệu COREL, thời gian tạo cây của hai phương pháp này lần lượt là 67,840.0957 milli giây và 587,575.1129 milli giây (tức là khoảng 67.840 giây và 587.575 giây). Đối với tập dữ liệu Wang, thời gian tạo cây của hai phương pháp này lần lượt là 860,989.2461 milli giây và 12,202,465.4815 milli giây (tức là khoảng 14.3498 phút và 203.3744 phút). Đối với tập dữ liệu ImgCollect, thời gian tạo cây của hai phương pháp này lần lượt là 4,207,389.7730 milli giây và 48,544,118.4751 milli giây (tức là khoảng 1.168719 giờ và 13.48447735 giờ). Tuy nhiên, việc tạo cây *Sig*-Tree là pha chuẩn bị dữ liệu, không ảnh hưởng đến thời gian truy vấn ảnh.



Hình 6. Đồ thị *recall-precision* và đường cong ROC trên bộ dữ liệu COREL của các phương pháp



Hình 7. Đồ thị *recall-precision* và đường ROC trên bộ dữ liệu Wang của các phương pháp



Hình 8. Đồ thị *recall-precision* và đường ROC trên bộ dữ liệu ImgCollect của các phương pháp

Mỗi đường cong trong **Hình 6** mô tả chi tiết về độ chính xác truy vấn ảnh từng phương pháp trên tập ảnh COREL (1,000 ảnh). Theo như kết quả thực nghiệm tại **Hình 6**, phương pháp EP-256 có độ chính xác cao hơn tất cả các phương pháp còn lại. Điều này chứng tỏ sự hiệu quả của phương pháp truy vấn ảnh sử dụng chữ ký nhị phân và cây *Sig*-Tree, nghĩa là thời gian thực thi nhanh và có độ chính xác cao. Trong **Hình 7**, mỗi đường cong mô tả độ chính xác của từng phương pháp đã mô tả như trên theo bộ dữ liệu ảnh Wang (10,800 ảnh). Theo như kết quả thực nghiệm trên bộ dữ liệu ảnh Wang cho thấy độ chính xác của phương pháp EP-265 vẫn vượt trội so với các phương pháp khác. Trong **Hình 8**, biểu diễn đường cong mô tả độ chính xác- độ phủ và đường cong ROC của các phương pháp như đã mô tả trong Bảng 1 trên bộ dữ liệu ImgCollect (36,986 ảnh). Theo như kết quả cho thấy, độ chính xác của phương pháp EP-256 cao hơn so với các phương pháp khác cùng thể loại. Qua thực nghiệm trên ba bộ dữ liệu khác nhau cho thấy tính hiệu quả của các phương pháp truy vấn ảnh theo nội dung dựa trên chữ ký nhị phân và cây *Sig*-Tree. Vì vậy, các cải tiến trên cây Sig-Tree thể hiện tính đúng đắn và phù hợp cho bài toán truy vấn ảnh.

C. So sánh kết quả thực nghiệm

Chúng tôi sử dụng hai phương pháp EP-64 và EP-256 để so sánh với các phương pháp khác đã công bố trên bộ dữ liệu ảnh COREL, gồm: (1) Phương pháp KD-Tree [32]; (2) Phương pháp S-Tree [15]; (3) Phương pháp QBIC, Fuzzy Signature [25]; (4) Phương pháp color histogram, fuzzy-color histogram, bit-planes [30]; (5) Phương pháp đối sánh histogram vùng đặc trưng [29].

Bảng 2. So sánh độ chính xác truy vấn giữa các phương pháp

Phương pháp	Độ chính xác trung bình	Độ phủ trung bình	Độ đo F-measure trung bình	Thời gian truy vấn trung bình
KD-Tree [32]	0.876031667	N/A	N/A	93; 63; 46 (msec)
S-Tree [15]	0.42	0.55	0.476289	186.25 I/Os
QBIC [25]	N/A	N/A	N/A	2-40 sec
Fuzzy Signatures [25]	N/A	N/A	N/A	20-50 I/Os
Color histogram [30]	0.29125	0.06400	0.104940	4.43750 sec
Fuzzy-color histogram [30]	0.37438	0.08100	0.133184	4.25625 sec
Bit-planes[30]	0.52938	0.12125	0.197308	4.18516 sec
Interest point [29]	0.65688	0.70500	0.68009	4.70938 sec
Sub-color histogram [29]	0.49125	0.61063	0.54447	4.43638 sec
Fuzzy color histogram [29]	0.50688	0.61625	0.55624	4.41863 sec
Interest region [29]	0.85200	0.78375	0.81645	4.78516 sec
EP-64	0.687480469	0.687487535	0.687484002	8.257 msec
EP-256	0.843602941	0.843331798	0.843467348	26.813 msec

Theo số liệu so sánh từ **Bảng 2** cho thấy phương pháp truy vấn ảnh theo nội dung dựa trên chữ ký nhị phân và cây *Sig*-Tree có thời gian truy vấn nhanh hơn các phương pháp khác đồng thời vẫn đảm bảo độ chính xác cao. Từ đó cho thấy phương pháp đã đề xuất hiệu quả đối với bài toán truy vấn ảnh tương tự.

VI. KẾT LUẬN

Trong bài báo đã tiếp cập xây dựng hệ truy vấn ảnh theo nội dung dựa trên chữ ký nhị phân và cây *Sig*-Tree. Bài báo đã mô tả cách tạo chữ ký nhị phân để tạo thành chỉ mục cho hình ảnh, đồng thời bài báo đã thiết kế cấu trúc cây *Sig*-Tree cũng như các thao tác tương ứng. Trên cơ sở lý thuyết đã đề xuất, bài báo mô tả ứng dụng thực nghiệm qua nhiều phương pháp biến thể khác nhau nhằm minh chứng rằng phương pháp đề xuất là một giải pháp đúng đắn. Kết quả thực nghiệm của hai phương pháp chính đó là EP-64 và EP-256 cũng được so sánh với các phương pháp khác. Theo như kết quả thực nghiệm cho thấy phương pháp đã đề xuất là giải pháp tốt để giải quyết bài toán truy vấn ảnh theo nội dung, hơn nữa đây cũng là một giải pháp hiệu quả. Trong phần phát triển tiếp theo, nhóm tác giả kết hợp phương pháp gom cụm không phân cấp với cấu trúc *Sig*-Tree để thực hiện các thao tác tách/ghép cụm tại các nút của cây, hơn nữa các cụm tại nút trong cây liên kết láng giềng với nhau để từ đó tăng tốc độ truy vấn ảnh.

VII.LỜI CẢM ƠN

Nhóm tác giả xin chân thành cám ơn Khoa Công nghệ Thông tin, Trường ĐH Khoa Học – Đại học Huế là nơi cố vấn chuyên môn cho nghiên cứu này. Nhóm tác giả xin gửi lời cám ơn đến Trung tâm Công nghệ Thông tin, Trường ĐH Công nghiệp Thực phẩm Tp.HCM là nơi bảo trợ cho nghiên cứu này.

TÀI LIỆU THAM KHẢO

[1] T. Acharya, A. K. Ray, *Image Processing: Principles and Applications*, John Wiley & Sons Inc. Publishers, Hoboken, New Jersey, 2005.

[2] Aci, *http://www.aci.aero/*, 2015.

[3] A. Alzu'bi, A. Amira, N. Ramzan, *Semantic content-based image retrieval: A comprehensive study*, Journal of Visual Communication and Image Representation, 32, pp. 20-54, 2015.

[4] A. Bhagyalaksluni, V. V. Ayachamundeeswari, *A Survey on Content Based Image Retrieval Using Various Operators*, International Conference on Computer Communication and Systems, IEEE, pp. 018-023, Chennai, 2014.

[5] Y. Chen, *On the signature trees and balanced signature trees*, 21st International Conference on Data Engineering, ICDE'05, IEEE, pp. 742-753, 2005.

[6] Y. Chen, Y. Chen, *On the Signature Tree Construction and Analysis*, IEEE Transactions On Knowledge And Data Engineering, 18(**9**), pp. 1-18, 2006.

[7] C. Chute, *Worldwide Digital Image 2015–2019 Forecast: The Image Capture and Share Bible*, International Data Corporation, pp. 13 pages, (February 2015 # 254256).

[8] W. Dejonge, P. Scheuermann, A. Schijf, *S^+-Trees: An Efficient Structure for the Representation of Large Pictures*, CVGIP: Image Understanding, 59(**3**), pp. 265-280, 1994.

[9] L. Deligiannidis, H. R. Arabnia, *Emerging Trends in Image Processing, Computer Vision, and Pattern Recognition*, Morgan Kaufmann, Elsevier, Waltham, MA 02451, USA, 2015.

[10] U. Deppisch, *S-tree: a dynamic balanced signature index for office retrieval*, Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp. 77-87, New York, NY, USA, 1986.

[11] M. Hilbert, *A review of large-scale 'How much information?' inventories: variations, achievements and challenges*, Information Research, 20(**4**), pp. paper 688, 2015.

[12] Idc, *https://www.idc.com*, 2016.

[13] K. Jenni, S. Mandala, M. S. Sunar, *Content Based Image Retrieval Using Colour Strings Comparison*, Procedia Computer Science, 50, pp. 374-379 2015.

[14] R. Ji, Y. Gao, L.-Y. Duan, H. Yao, Q. Dai, *Learning-Based Local Visual Representation and Indexing*, Elsevier, Radarweg 29, PO Box 211, 1000 AE Amsterdam, Netherlands, 2015.

[15] Y. Manolopoulos, A. Nanopoulos, E. Tousidou, *Advanced Signature Indexing for Multimedia and Web Applications*, Kluwer Academic Publishers, Springer Science+Business Media New York, 2003.

[16] O. Marques, B. Furht, *Content-Based Image and Video Retrieval*, Kluwer Academic Publishers, Springer Science+Business Media New York, 2002.

[17] V. Mezaris, I. Kompatsiaris, M. G. Strintzis, *Still Image Segmentation Tools for Object-based Multimedia Applications*, Int. Journal Of Pattern Recognition And Artificial Intelligence, 18(**4**), pp. 701-725, 2004.

[18] P. Muneesawang, L. Guan, *Multimedia Database Retrieval: A Human-Centered Approach*, Springer Scicnce+Businccss Media, New York, NY 10013, USA, 2006.

[19] P. Muneesawang, N. Zhang, L. Guan, *Multimedia Database Retrieval: Technology and Applications*, Springer International Publishing Switzerland, Springer Cham Heidelberg New York Dordrecht London, 2014.

[20] M. A. Nascimento, E. Tousidou, V. Chitkara, Y. Manolopoulos, *Image indexing and retrieval using signature trees*, Data & Knowledge Engineering, 43, pp. 57-77, 2002.

[21] J. Platos, P. Kromer, V. Snasel, A. Abraham, *Searching similar images - Vector Quantization with S-tree*, Fourth International Conference on Computational Aspects of Social Networks, CASoN 2012, IEEE, pp. 384-388, Sao Carlos, 2012.

[22] R. Priya, T. N. Shanmugam, *A comprehensive review of significant researches on content based indexing and retrieval of visual information*, Front. Comput. Sci., 7(**5**), pp. 782-799, 2013.

[23] R. Raieli, *Multimedia Information Retrieval: Theory and techniques*, Chandos Publishing, Oxford OX28 4BN UK, 2013.

[24] I. E. Shanthi, Y. Izaaz, R. Nadarajan, *On the SD-tree construction for optimal signature operations*, Proceedings of the 1st Bangalore Annual Compute Conference, COMPUTE '08, ACM, Article No. 14, New York, NY, USA, 2008.

[25] V. Snášel, *Fuzzy Signatures for Multimedia Databases*, Advances in Information Systems, ADVIS 2000, Springer Berlin Heidelberg, 1909, pp. 257-264, Izmir, Turkey, 2000.

[26] V. Snasel, Z. Horak, M. Kudelka, A. Abraham, *Fuzzy Signatures Organized Using S-Tree*, IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, pp. 633 - 637, Anchorage, AK, 2011.

[27] E. Tousidou, A. Nanopoulos, Y. Manolopoulos, *Improved methods for signature-tree construction*, The Computer Journal, 43(**4**), pp. 300-313, 2000.

[28] J. Z. Wang, *Integrated Region-Based Image Retrieval*, Kluwer Academic Publishers, Springer Science Business Media New York, 2001.

[29] X.-Y. Wang, J.-F. Wu, H.-Y. Yang, *Robust Image Retrieval Based on Color Histogram of Local Feature Regions*, Springer Science, Multimed Tools Appl, 49, pp. 323-345, 2010.

[30] X.-Y. Wang, H.-Y. Yang, Y.-W. Li, F.-Y. Yang, *Robust color image retrieval using visual interest point feature of significant bit-planes*, Digital Signal Processing, 23(**4**), pp. 1136-1153, 2013.

[31] C. Wengert, M. Douze, H. Jégou, *Bag-of-colors for Improved Image Search*, Proceedings of the 19th ACM international conference on Multimedia, ACM, pp. 1437-1440, Scottsdale, Arizona, USA, 2011.

[32] H. Zouaki, B. Abdelkhalak, *Indexing and content-based image retrieval*, International Conference on Multimedia Computing and Systems (ICMCS), IEEE, pp. 1-5, Ouarzazate, 2011.

IMPROVING FOR CONTENT-BASED IMAGE RETRIEVAL USING S-TREE

Van The Thanh, Le Manh Thanh

ABSTRACT*— Digital image has been becoming the popular things in the life of people, so the image retrieval problem is a need of modern society. This paper approaches the CBIR (Content-Based Image Retrieval) based on binary signature and S-Tree. To create the binary signature, we apply K-mean method to create the color palette from a set including 36,986 images. Next, the paper gives the Sig-Tree data structure rely on S-Tree, then we describle the operators on this tree. The paper uses Hamming, EMD distance (Earth Mover Distance) and CIE-Lab color space to assess the similarity measure between images. In order to show the proposed theory, we build the applications to assess the experimental results on the image databases including COREL (1,000 images), Wang (10,800 images), ImgCollect (36,986 images).*