Một mô hình tìm kiếm ảnh dựa trên cấu trúc R-Tree kết hợp KD-Tree Random Forest

Lê Mạnh Thạnh[1], Lê Thị Vĩnh Thanh[1,4], Lương Thị Thanh Xuân[1,5], Nguyễn Thị Định[1,3], Văn Thế Thành[2]
[1] Trường Đại học Khoa học, Đại học Huế
[2] Trường Đại học Sư phạm TP. HCM
[3] Khoa Công nghệ Thông tin, Trường ĐH Công nghiệp Thực phẩm TP.HCM
[4] Trường Đại học Bà Rịa Vũng Tàu
[5] Trường THCS-THPT Nguyễn Văn Khải, Đồng Tháp

Tác giả liên hệ: Văn Thế Thành, thanhvt@hcmue.edu.vn
Ngày nhận bài: 15/04/2022, ngày sửa chữa: 01/06/2022, ngày duyệt đăng: 30/06/2022
Định danh DOI: 10.32913/mic-ict-research-vn.v2022.n1.1053

Tóm tắt: Nâng cao hiệu suất truy vấn ảnh là vấn đề được quan tâm trong lĩnh vực thị giác máy tính. Trong bài báo này, một phương pháp kết hợp cấu trúc R-Tree và KD-Tree Random Forest được thực hiện nhằm nâng cao hiệu suất phân lớp và tìm kiếm ảnh. Với mỗi ảnh đầu vào được phân lớp bằng KD-Tree Random Forest; sau đó, kỹ thuật gom cụm trên R-Tree được thực hiện và tìm kiếm tập ảnh tương tự. Để thực hiện bài toán này, phương pháp xây dựng KD-Tree Random Forest kết hợp với cấu trúc R-Tree cùng với các thuật toán xây dựng rừng ngẫu nhiên, phân lớp và tìm kiếm ảnh được đề xuất. Trên cơ sở đó, một mô hình tìm kiếm ảnh dựa trên sự kết hợp KD-Tree Random Forest và R-Tree được đề xuất. Thực nghiệm được thực hiện trên các bộ ảnh COREL và Caltech101 nhằm để đánh giá tính khả thi, hiệu quả của mô hình; đồng thời so sánh với các công trình khác thực nghiệm trên cùng bộ dữ liệu và so sánh với kết quả truy vấn ảnh khi sử dụng một cấu trúc riêng lẻ. Theo kết quả thực nghiệm cho thấy phương pháp đề xuất của chúng tôi là hiệu quả và có thể áp dụng được cho các hệ truy vấn ảnh với nhiều bộ dữ liệu khác nhau.

Từ khóa: *KD-Tree, Random Forest, R-Tree, gom cụm, phân lớp, tìm kiếm ảnh*

Title: **A Model of Combining R-Tree and KD-Tree Random Forest for Content-based Image Retrieval**

Abstract: Improving image query performance is a matter of interest in the field of computer vision. In this paper, a method combining R-Tree and KD-Tree Random Forest structures is implemented to enhance the performance of im-age classification and retrieval. For each input image is classified by KD-Tree Random Forest; then, the clustering technique on R-Tree is performed and a similar image set is retrieved. To implement this problem, the method of construction KD-Tree Random Forest combined with the R-Tree structure with the algorithms for building a random forest, classification, and image retrieval. On that basis, an image retrieval model based on the combination of KD-Tree Random Forest and R-Tree is proposed. Experiments were performed on the COREL and Caltech101 image sets to evaluate the feasibility and effectiveness of the model; at the same time experiment is compared with other works on the same data set and compared with image query results when using an individual structure. The experimental results show that our proposed method is effective and can be applied to image retrieval systems for many sets of images in many different fields

Keywords: *KD-Tree, R-Tree, Random Forest, Clustering, Image Classification, Image Retrieval*

I. GIỚI THIỆU

Phương pháp kết hợp nhiều kỹ thuật học máy nhằm nâng cao hiệu suất tìm kiếm ảnh là vấn đề cần được quan tâm trong lĩnh vực truy vấn ảnh và thị giác máy tính. Bên cạnh đó, dữ liệu đa phương tiện tăng nhanh theo thời gian về số lượng và thể loại là thách thức cho vấn đề lưu trữ, hiệu suất và thời gian tìm kiếm. Vì vậy, việc tích hợp một số kỹ thuật và phương pháp cho một hệ truy vấn ảnh nhằm nâng cao hiệu suất, ổn định thời gian tìm kiếm cũng như tối ưu hóa không gian lưu trữ là cần thiết [1].

Không gian lưu trữ ảnh số là vấn đề cần được quan tâm và tối ưu nhằm đáp ứng thực trạng gia tăng ảnh số trong bối cảnh hiện nay. Để giải quyết vấn đề này, một số cấu trúc dữ liệu như S-Tree [2], R-Tree [3], v.v. nhằm tối ưu không gian lưu trữ ảnh số đã được thực hiện đã mang lại những kết quả khả quan. Trong bài báo này, một cấu trúc

R-Tree được ứng dụng cho quá trình gom cụm dữ liệu thành các cụm tương đồng và áp dụng cho bài toán tìm kiếm ảnh được đề xuất. Bên cạnh đó, phân lớp dữ liệu hình ảnh là giai đoạn đầu, đồng thời góp phần nâng cao hiệu suất cho bài toán tìm kiếm ảnh tương tự. Hiện nay, có nhiều kỹ thuật phân lớp hình ảnh được đánh giá với hiệu suất phân lớp khá cao như CNN, DNN, k-NN, v.v. Tuy nhiên, hướng tiếp cận mô hình phân lớp bằng KD-Tree là một cải tiến trên KD-Tree đã mang lại hiệu suất phân lớp ổn định, đồng thời áp dụng cho bài toán tìm kiếm ảnh hiệu quả [16, 26]. Do đó, trong bài báo này một phương pháp phân lớp ảnh số bằng KD-Tree Random Forest được thực hiện dựa trên cơ sở xây dựng nhiều cấu trúc KD-Tree đồng thời để hình thành quá trình phân lớp ảnh bằng rừng ngẫu nhiên. Bên cạnh đó, gom cụm trên R-Tree là một phương pháp nâng cao hiệu quả tìm kiếm ảnh [15]. Vì vậy, mục tiêu của bài báo này là thực hiện sự kết hợp những điểm mạnh của cấu trúc KD-Tree và R-Tree nhằm nâng cao hiệu suất tìm kiếm ảnh cần được thực hiện.

Đóng góp của bài báo gồm: (1) Đề xuất phương pháp kết hợp cấu trúc R-Tree và KD-Tree Random Forest cho bài toán tìm kiếm ảnh; (2) Đề xuất mô hình tìm kiếm ảnh dựa trên kỹ thuật phân lớp KD-Tree Random Forest và gom cụm R-Tree; (3) Đánh giá, so sánh hiệu suất truy vấn ảnh trên các phương pháp khác cùng bộ ảnh thực nghiệm COREL [4], Clatech101 [5].

Cấu trúc của bài báo gồm: phần II trình bày các công trình liên quan về gom cụm trên R-Tree, phân lớp ảnh bằng KD-Tree và rừng ngẫu nhiên; phần III đề xuất mô hình tìm kiếm ảnh dựa trên sự kết hợp của cấu trúc KD-Tree Random Forest và R-Tree; phần IV trình bày cấu trúc và phương pháp xây dựng KD-Tree Random Forest, xây dựng và tìm kiếm trên R-Tree cho bài toán tìm kiếm ảnh; phần V thực nghiệm và đánh giá kết quả trên bộ ảnh COREL, Calterch101; kết luận và hướng phát triển được trình bày ở phần VI.

II. CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN

Phân lớp và gom cụm luôn được sử dụng cho bài toán tìm kiếm ảnh vì sự ảnh hưởng kết quả của các quá trình này đến kết quả tìm kiếm ảnh là khá lớn. Hiện nay, có nhiều công trình tìm kiếm ảnh đã sử dụng các kỹ thuật phân lớp và gom cụm như CNN, DNN, k-NN, k-Means, v.v kết hợp với cấu trúc R-Tree, KD-Tree đã mang lại hiệu suất cao tiêu biểu là:

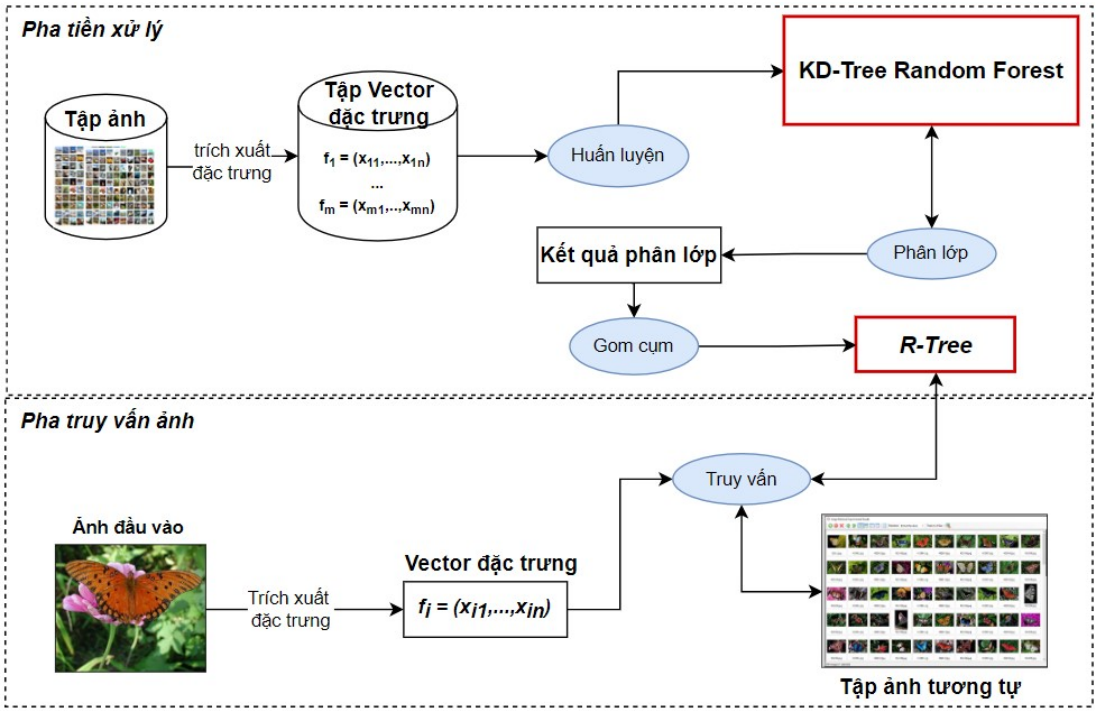
Yuqian Zhang và cộng sự (2016) [6] đã sử dụng phương pháp phân chia vùng ảnh để nhận diện khuôn mặt sử dụng cấu trúc KD-Tree. Tại pha huấn luyện, mỗi ảnh được chia thành nhiều vùng; cấu trúc KD-Tree được xây dựng dựa trên tập ảnh phân vùng nhằm phân lớp và lưu trữ tập ảnh huấn luyện. Tại pha kiểm thử, mỗi ảnh được chia thành nhiều vùng và cấu trúc KD-Tree được sử dụng trong quá trình tìm kiếm theo k láng giềng gần nhất bằng cách đối sánh các vùng ảnh tìm kiếm với ảnh gốc bằng thuật toán k-NN. Công trình này được đánh giá là sự kết hợp giữa cấu trúc KD-Tree và thuật toán k-NN đã mang lại hiệu suất cao hơn so với chỉ dùng một kỹ thuật đơn lẻ.

Fatin Zaklouta và cộng sự (2011) [17], đã thực hiện đánh giá hiệu suất phân lớp ảnh đèn giao thông bằng KD-Tree và rừng ngẫu nhiên để phân loại biển báo giao thông bằng cách sử dụng đặc trưng biểu đồ kích thước khác nhau bằng đặc trưng HOG (Histogram of Oriented Gradients) và phép biến đổi khoảng cách (Distance Transforms). Các tác giả sử dụng tập dữ liệu ảnh biển báo giao thông của Đức gồm 43 lớp và hơn 50.000 hình ảnh. Trong bài báo này, sự kết hợp của bộ phân loại KD-Tree với đặc trưng HOG và Distance Transforms thì tỷ lệ phân loại lên tới 0.97 và 0.8180. Kết quả này được đánh giá là khá cao và có khả năng mở rộng cho các hệ phân loại ảnh khác.

Weishi Man và cộng sự (2018) [18] đã thực hiện một cải tiến trong thuật toán tách nút rừng ngẫu nhiên để nâng cao hiệu suất phân lớp hình ảnh bằng cách kết hợp mô hình phân tách thuộc tính rừng ngẫu nhiên ID3 và CART. Thứ nhất, tính hợp lệ của mô hình Kim tự tháp không gian được xác minh dựa trên mô hình túi từ (Bag of Words). Thứ hai, thuật toán tách nút của rừng ngẫu nhiên được cải tiến. Cuối cùng, hiệu quả và độ chính xác của thuật toán đề xuất được minh chứng thông qua so sánh thực nghiệm trên bộ dữ liệu hình ảnh Caltech101 với hiệu suất cao, điều này cho thấy phân lớp ảnh bằng rừng ngẫu nhiên là một phương pháp đúng đắn, hiệu quả.

Vanitha J., Senthilmurugan M. (2018) [7] đã đề xuất một cấu trúc cây chỉ mục SR-Tree (Sphere Rectangle Tree) là sự kết hợp giữa cây SS-Tree và cây R-Tree để lưu trữ dữ liệu hình ảnh dạng chỉ mục. Một hệ thống truy vấn ảnh dựa trên nội dung sử dụng cấu trúc cây SR-Tree được thực nghiệm trên các bộ ảnh với kết quả được đánh giá là khả quan. Cùng thời điểm này, Arpitha Y.C và cộng sự (2018) [8] đã đề xuất một cách tiếp cận bài toán xử lý hình ảnh đa chiều bằng cách sử dụng cấu trúc R-Tree, trong đó các đối tượng hình ảnh đa chiều được lưu trữ theo phương pháp chỉ mục. Trên cơ sở này, một phương pháp tìm kiếm các đối tượng trong một vùng cho trước và tìm kiếm láng giềng gần nhất dựa trên cây R-Tree được thực hiện đã mang lại kết quả tốt.

Bên cạnh đó, Xinlu Wang và cộng sự (2019) [9] đã đề xuất một phương pháp truy vấn động trên cấu trúc R-Tree, các tâm cụm động được tối ưu hóa dựa trên cấu trúc R-Tree. Bằng cách chọn một tâm cụm tối ưu hóa, phương pháp này tổ chức lưu dữ liệu cùng một không gian con thành cùng cây con và xây dựng một cây chỉ mục R-Tree. Kết quả thực nghiệm cho thấy phương pháp được đề xuất cải thiện tính



Hình 1. Mô hình truy vấn ảnh dựa trên R-Tree kết hợp KD-Tree Random Forest

ổn định của hệ thống và hiệu quả cao, đồng thời ứng dụng cho hệ thống tra cứu thông tin bệnh viện.

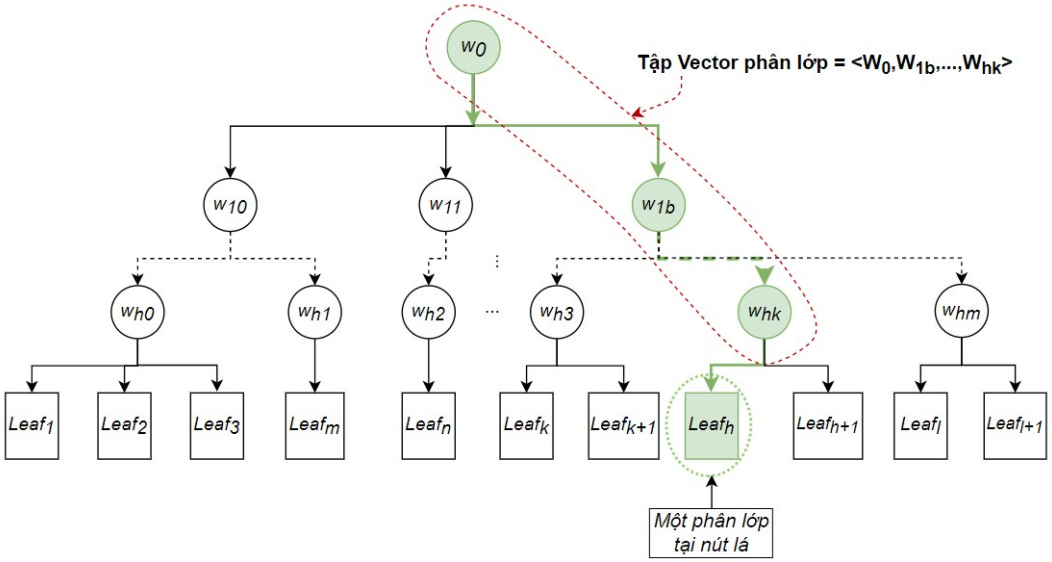
Trên cơ sở các công trình nghiên cứu liên quan, nhóm tác giả Nguyễn Thị Định và cộng sự [16, 26] đã thực hiện một phương pháp phân lớp dữ liệu bằng KD-Tree và áp dụng cho bài toán tìm kiếm ảnh tương tự. Trong những công trình này, nhóm tác giả đã thực nghiệm trên các bộ ảnh COREL, Wang, image CLEF với hiệu suất khá cao. Quá trình xây dựng KD-Tree theo hướng tiếp cận phân lớp dữ liệu để hình thành các phân lớp tại nút lá. Đồng thời nhóm tác giả Lê Thị Vĩnh Thanh và cộng sự [15] đã thực hiện một phương pháp gom cụm trên R-Tree cho bài toán tìm kiếm ảnh đã mang lại hiệu suất cao khi so sánh với các công trình khác cùng lĩnh vực. Nhóm tác giả đã thực hiện cải tiến cấu trúc R-Tree nhằm nâng cao khả năng mở rộng không gian lưu trữ, tối ưu thời gian tìm kiếm.

Từ những công trình đã khảo sát cho thấy các loại đặc trưng cơ bản như màu sắc, hình dạng, kết cấu, đường viền được sử dụng khá phổ biến và mang lại hiệu suất cao. Tuy nhiên các công trình này chưa quan tâm đến vấn đề tối ưu khả lưu trữ. Vì vậy, trong bài báo này một số đặc trưng sau khi trích xuất véc-tơ được phân lớp bằng KD-Tree Random Forest, sau đó gom cụm trên R-Tree để ổn định thời gian tìm kiếm và tăng khả năng lưu trữ đáp ứng nhu cầu ngày càng gia tăng dữ liệu ảnh số. Phương pháp kết hợp giữa kỹ thuật phân lớp ảnh bằng rừng ngẫu nhiên trên cơ sở xây dựng nhiều KD-Tree đồng thời; sau đó tiếp tục gom cụm trên R-Tree đã mang lại hiệu suất tìm kiếm cao, thời gian tìm kiếm ổn định và áp dụng cho các bộ ảnh có tính chất khác nhau với hiệu quả cao.

III. MÔ HÌNH TÌM KIẾM ẢNH DỰA TRÊN R-TREE KẾT HỢP KD-TREE RANDOM FOREST

Để minh chứng cho cơ sở lý thuyết đề xuất, một mô hình tìm kiếm ảnh kết hợp cấu trúc R-Tree và KD-Tree minh họa bởi Hình 1. Trong đó, cấu trúc KD-Tree được sử dụng cho quá trình phân lớp hình ảnh, từ phân lớp để thực hiện gom cụm trên R-Tree để trích xuất tập ảnh tương tự tại nút lá và được kế thừa từ công trình [15, 16]. Để thực hiện mô hình này, các thuật toán đề xuất gồm: (1) thuật toán huấn luyện KD-Tree; (2) thuật toán xây dựng KD-Tree Random Forest; (3) thuật toán phân lớp ảnh bằng KD-Tree Random Forest; (4) thuật toán gom cụm bằng R-Tree; (5) thuật toán tìm kiếm ảnh tương tự trên R-Tree.

Mô hình truy vấn ảnh dựa trên cấu trúc R-Tree kết hợp KD-Tree Random Forest gồm pha tiền xử lý và pha truy vấn ảnh với các bước như sau: (1) Trích xuất đặc trưng cho tập dữ liệu thực nghiệm thành tập véc-tơ đặc trưng; (2) Xây dựng và huấn luyện cấu trúc KD-Tree Random Forest từ tập véc-tơ đặc trưng theo phương pháp Bagging; (3) Phân lớp ảnh bằng cấu trúc KD-Tree Random Forest theo cơ chế phiếu bầu để trích xuất tên phân lớp ảnh; (4) Gom cụm dữ liệu trên R-Tree đã xây dựng sau khi thực hiện phân lớp;



Hình 2. Cấu trúc KD-Tree phân lớp

(5) Trích xuất véc-tơ đặc trưng cho ảnh đầu vào; (6) Truy vấn trên R-Tree để trích xuất tập ảnh tương tự với ảnh đầu vào.

IV. CẤU TRÚC KD-TREE RANDOM FOREST VÀ R-TREE

Thuật toán 1: Huấn luyện KD-Tree

Input:Bộ Véc-tơ InitWeight, chiều cao h, b nhánh
Output: Trained-KD-Tree
Function: **TKDT** (InitWeight, h, b)
begin
    Init-KD-Tree = ∅;
    while $P_j < P_i$ **do**
        Xây dựng Init-KD-Tree với bộ Véc-tơ InitWeight, chiều cao h, b nhánh;
        Gán nhãn cho các nút lá trên Init-KD-Tree;
        Pi = Tính hiệu suất phân lớp trên Init-KD-Tree;
        Xác định đường đi sai của những véc-tơ f_k trên Init-KD-Tree;
        Xác định đường đi đúng của những véc-tơ f_k bị sai đường đi;
        Tìm vị trí $Node_i$ làm sai đường đi của vectơ f_k;
        TrainVéc-tơ = Điều chỉnh véc-tơ lưu tại $Node_i$ để f_k đúng đường đi;
        Tạo lại Trained KD-Tree theo bộ TrainVéc-tơ;
        Gán nhãn cho các nút lá trên Trained-KD-Tree;
        P_j= Tính hiệu suất phân lớp sau khi điều chỉnh véc-tơ tại Nodei;
    end
    Véc-tơ = TrainVéc-tơ; **return** Trained-KD-Tree;
end

Cấu trúc dữ liệu lưu trữ là đối tượng giúp tối ưu hóa không gian lưu trữ, giảm thời gian tìm kiếm và ảnh hưởng trực tiếp đến hiệu suất truy vấn ảnh. Trong bài báo này, một phương pháp kết hợp giữa hai cấu trúc R-Tree và KD-Tree Random Forest để thực hiện gom cụm và phân lớp cho bài toán tìm kiếm ảnh được thực hiện.

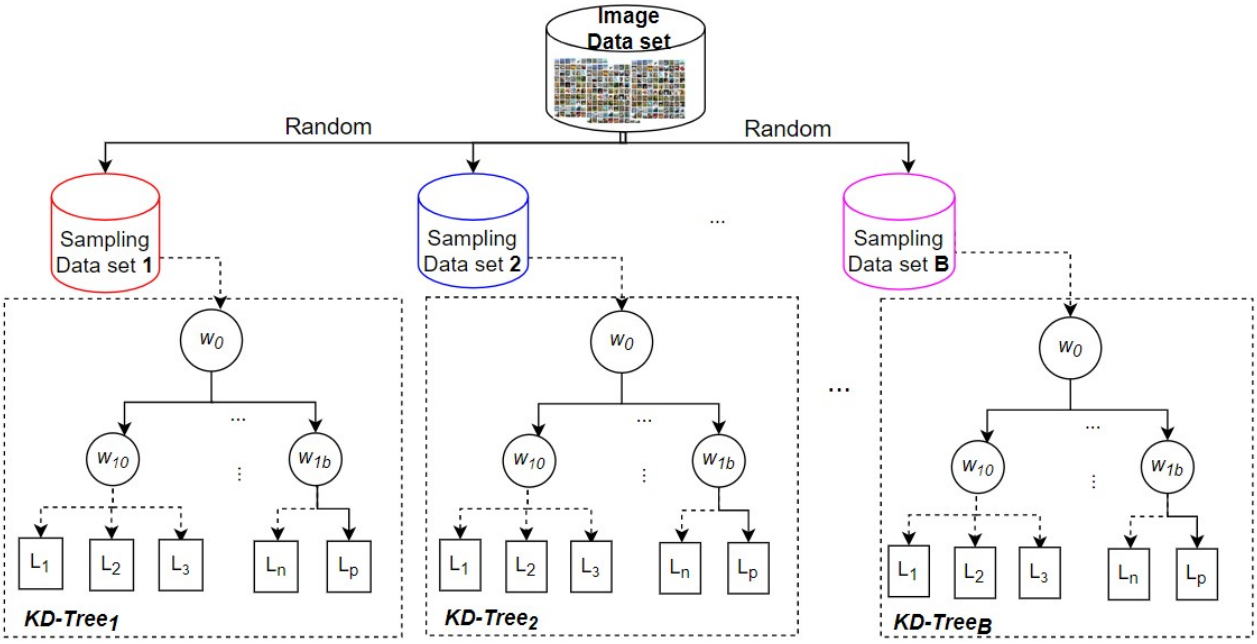
1. Cấu trúc KD-Tree cho phân lớp hình ảnh

Dựa trên tính chất cây KD-Tree nguyên thủy [10] và kế thừa từ công trình [16, 26], trong bài báo này cấu trúc KD-Tree phân lớp được xây dựng là một cây đa nhánh cân bằng, gồm một nút gốc lưu trữ véc-tơ phân lớp (w_0); tập nút trong Node, mỗi nút trong lưu trữ một véc-tơ phân lớp ($w_l k$) và tập nút lá Leaf, mỗi nút lá chứa nhiều véc-tơ hình ảnh $F = f_1, f_2, \ldots, f_k$. Sau khi xây dựng cấu trúc KD-Tree phân lớp tại mỗi nút lá là một phân lớp chứa tập véc-tơ hình ảnh thuộc nhiều phân lớp khác nhau. Cấu trúc KD-Tree đóng vai trò phân lớp cho ảnh đầu vào từ nút gốc đến nút lá và quá trình phân lớp trên KD-Tree được minh họa như Hình 2.

Độ phức tạp của thuật toán TKDT (Huấn luyện KD-Tree) là $O(p*h*n)$. Trong đó p là số lần điều chỉnh véc-tơ trọng số; h là chiều cao cây (hằng số); n là số phần tử tham gia vào quá trình xây dựng cây. Quá trình huấn luyện cấu trúc KD-Tree được thực hiện thông qua số lần cập nhật trọng số để tạo cây. Do đó độ phức tạp của thuật toán TKDT là độ phức tạp của số lần tạo cây. Vì vậy, độ phức tạp của thuật toán TKDT là $O(p*n)$.

2. Xây dựng cấu trúc KD-Tree Random Forest bằng kỹ thuật Bagging

Để nâng cao hiệu suất phân lớp cho ảnh đầu vào, trong bài báo này một cấu trúc KD-Tree Random Forest được



Hình 3. Minh họa KD-Tree Random Forest

xây dựng với kỹ thuật Bagging được đề xuất. Trong đó, KD-Tree Random Forest là cấu trúc gồm nhiều KD-Tree được xây dựng độc lập và huấn luyện để hình thành mô hình phân lớp bằng rừng ngẫu nhiên.

Rừng ngẫu nhiên là một trong những kỹ thuật phân lớp hình ảnh hiệu quả bởi sự kết hợp các kết quả đánh giá khách quan từ nhiều mô hình phân lớp độc lập. Quá trình xây dựng rừng ngẫu nhiên kết hợp với kỹ thuật Bagging được nhiều công trình đã công bố với kết quả phân lớp ảnh khả quan [19, 20, 21]. Trong bài báo này, một phương pháp kết hợp rừng ngẫu nhiên trong đó mỗi bộ phân lớp là một cấu trúc KD-Tree được xây dựng với kỹ thuật Bagging nhằm nâng cao hiệu suất phân lớp hình ảnh đầu vào. Rừng ngẫu nhiên được ứng dụng khá nhiều vào các công trình phân loại dữ liệu, phân lớp hình ảnh [18, 23] bởi hiệu suất phân loại đối tượng cao hơn một số kỹ thuật học máy đơn lẻ như k-NN, CNN, ANN, v.v. Tuy nhiên chi phí thời gian xây dựng và huấn luyện mô hình phân loại bằng rừng ngẫu nhiên là khá lớn. Trong bài báo này, trên cơ sở xây dựng KD-Tree Random Forest nhằm phân loại đối tượng hình ảnh bằng một mô hình kết hợp để thực hiện phân loại hình ảnh qua nhiều cấu trúc KD-Tree độc lập, rừng ngẫu nhiên cần được xây dựng. Kết quả phân lớp hình ảnh bằng mô hình rừng ngẫu nhiên dựa trên KD-Tree Random Forest bằng kỹ thuật Bagging áp dụng cho bài toán tìm kiếm ảnh được đánh giá là hiệu quả.

Trong công trình [16, 26] việc phân lớp ảnh bằng cấu trúc KD-Tree là thực hiện phân lớp cho ảnh đầu vào bằng một cấu trúc KD-Tree nên hiệu suất phân lớp chưa thực sự khách quan. Do đó, kỹ thuật phân lớp ảnh bằng nhiều cây KD-Tree để đảm bảo tính khách quan cho người dùng và nhằm nâng cao hiệu suất phân lớp là thực sự cần thiết. Để thực hiện điều này, rừng ngẫu nhiên được xây dựng bởi nhiều cây KD-Tree để thực hiện phân lớp đồng thời cho ảnh đầu vào. Sau khi thực hiện phân lớp trên từng cây KD-Tree, phương pháp bỏ phiếu bầu để lấy số đông cho kết quả phân lớp bằng rừng ngẫu nhiên.

Thuật toán 2: Xây dựng KD-Tree Random Forest

Input:Traning set S, Number of KD-Tree B;
Output: KD-Tree Forest
Function: **RF-KDT** (S,B)
begin
  foreach $x \in \mathcal{X}$ **do**
    S_i = a bootstrap sample from S;
    h_i = Create KD-Tree(S_i, W_{kt}, h, b);
    KD-TreeForest = KD-TreeForest;
    $\cup\ h_i$;
  end
  return KD-Tree Forest;
end

Trong bài báo này, rừng ngẫu nhiên được xây dựng là tập hợp nhiều cây KD-Tree nhằm thực hiện phân lớp nhiều lần cho một ảnh đầu vào (I) dựa trên các mô hình phân lớp riêng biệt và thực hiện song song. Minh họa rừng ngẫu nhiên KD-Tree RandomForest và kỹ thuật Bagging được minh họa trong hình 3 gồm các bước: (1) Xây dựng bộ Data Radom (Data$_i$); (2) Xây dựng KD-Tree cho từng bộ Data$_i$; (3) Tập hợp B cây KD-Tree hình thành KD-Tree Random Forest.

Độ phức tạp của thuật toán RF-KDT (Xây dựng KD-Tree Random Fores) là *B* O(k*h)*; trong đó *B* là số cây KD-Tree cần xây dựng và *O(P)* (*P = k*h*) là độ phức tạp cho xây dựng một cây KD-Tree có *k* phần tử và chiều cao *h*. Vậy độ phức tạp của thuật toán RF-KDT là *O(B*P)*, vì *B* là hằng số nên độ phức tạp của thuật toán RF-KDT là *O(P)*.

3. Phân lớp ảnh bằng KD-Tree Random Forest theo cơ chế phiếu bầu

Sự kết hợp nhiều cấu trúc cây phân lớp KD-Tree để xây dựng rừng ngẫu nhiên nhằm thực hiện phân loại đối tượng qua nhiều cấu trúc KD-Tree khác nhau. Mỗi hình ảnh được phân lớp bằng nhiều cây KD-Tree sẽ có nhiều hơn một kết quả nhãn lớp. Do đó, kết quả phân loại được thực hiện thông qua cơ chế phiếu bầu để mang lại hiệu quả phân lớp tốt hơn cho tập ảnh đơn đối tượng cũng như ảnh đa đối tượng [21, 22]. Trong bài báo này, một phương pháp phân lớp bằng KD-Tree Random Forest theo cơ chế phiếu bầu được đề xuất và thực hiện. Một ảnh đầu vào được thực hiện phân lớp bởi nhiều cây KD-Tree, kết quả phân lớp trên mỗi cây là một phân lớp cho ảnh đầu vào. Do đó, cần thực hiện bỏ phiếu bầu để tìm số phân lớp giống nhau là nhiều nhất cho ảnh I bởi rừng ngẫu nhiên. Sau khi thực hiện phiếu bầu thì phân lớp cho ảnh I là phân lớp có phiếu bầu nhiều nhất. Công thức phiếu bầu để chọn phân lớp cuối cùng (Final Class) cho ảnh đầu vào I là:

FinalClass(I) = max(number of voting I in Random Forest)

Trong trường hợp ảnh đầu vào thuộc *m* phân lớp khác nhau, kỹ thuật phiếu bầu sẽ được thực hiện lấy kết hợp *m* phân lớp có số phiếu bầu cao nhất.

Thuật toán 3: Phân lớp ảnh bằng KD-Tree Random Forest theo cơ chế phiếu bầu

```
Input:vetor f_i of image I
Output: Final class name of image I
Function: CKDFV (f_i, KD-Tree Forest)
begin
    Khởi tạo N KD-Tree cho rừng ngẫu nhiên (KD-Tree Forest);
    foreach KD-Tree[i] in KD-Tree Forest do
        Phân lớp ảnh I theo từng KD-Tree[i];
    end
    FinalClass(I) = Phân lớp được chọn theo phiếu bầu nhiều nhất;
    return FinalClass(I);
end
```

Độ phức tạp của thuật toán CKDFV là *O(N*h*k)*. Trong đó *B* là số cây KD-Tree trong KD-Tree Forest; *h* là chiều cao cây KD-Tree và k là số nhánh tối đa trên KD-Tree. Vì *B, h, k* là những hằng số nên đặt *T = B*h*k*. Vậy độ phức tạp của thuật toán CKDFV là *O(T)*, *T* là hằng số.

4. Cấu trúc R-Tree gom cụm dữ liệu

Quá trình gom cụm các véc-tơ đặc trưng dựa trên cấu trúc R^S-Tree [13]. Trong công trình này, tác giả đề xuất một cấu trúc cây để gom cụm và lưu trữ các véc-tơ đa chiều: R^S-Tree, một cấu trúc cải tiến của R-Tree, là cây đa nhánh cân bằng ứng dụng cho bài toán tìm kiếm ảnh tương tự. Cây R^S-Tree là cây phân hoạch dữ liệu không gian bao gồm: một nút gốc, một tập nút trong và một tập nút lá. R^S-Tree được đề xuất nhằm lưu trữ các phần tử hình ảnh trong không gian d chiều, mỗi phần tử $spED$ là một khối cầu có tâm $\overrightarrow{c}_{sp}$ và bán kính r_{sp} chứa véc-tơ đặc trưng ảnh $\overrightarrow{f_I} = (v_{I1}, v_{I2}, ..., v_{Id})$, với v_{Ii} là đặc trưng cấp thấp thứ i của hình ảnh I. Các nút trên R^S bao gồm: Nút trong S_{node}: là một bộ ⟨MBS,p⟩, trong đó MBS là một khối cầu có tâm $\overrightarrow{c}_{node}$ và bán kính r_{node}, p là con trỏ liên kết đến các nút con. Khối cầu này bao phủ các khối cầu của các nút thuộc nhánh cây con. Mỗi nút trong S_{node} có số phần tử tối thiểu là 2 và tối đa là N. Nút lá S_{leaf}: là bộ $<MBS, element>$, trong đó MBS là một khối cầu có tâm $\overrightarrow{c}_{leaf}$ và bán kính r_{leaf} chứa một tập thực thể element với mỗi thực thể $spED$ là một bộ $<MBS, oid>$ trong đó MBS là khối cầu có tâm $\overrightarrow{c}_{sp}$ và bán kính r_{sp} chứa không gian đối tượng, oid là định danh đối tượng $\overrightarrow{f_I} = (v_{I1}, v_{I2}, ..., v_{Id})$. Mỗi nút lá S_{leaf} có số phần tử tối đa là M và số phần tử tối thiểu là $m(1 < m < M/2)$.

5. Xây dựng cấu trúc R^S-Tree

Khi một phần tử $spED = <MBS(\overrightarrow{c}_{sp}, r_{sp}); f>$ được thêm vào cây R^S-Tree, việc chèn được thực hiện bắt đầu từ nút gốc, lần lượt duyệt qua tất cả các nút con của nút gốc và đi theo nhánh có khoảng cách $d_{\min}\{d_E(S_{node}.\overrightarrow{c}_{node}, spED.\overrightarrow{c}_{sp}), j = 1..N\}$, cho đến khi tìm được nút lá. Sau đó, phần tử $spED_i$ được phân phối vào nút lá hiện hành thỏa điều kiện $d_E(spED.\overrightarrow{c}_{sp}, S_{leaf}.\overrightarrow{c}_{leaf})<\theta$. Ngược lại, một nút lá mới $S_{newleaf}$ được tạo cùng cha với nút lá hiện hành để lưu phần tử $spED$. Quá trình xây dựng cấu trúc R^S-Tree theo các khối cầu nhằm giảm không gian lưu trữ so với phương pháp xây dựng theo khối hình chữ nhật, đây là một cải tiến của bài báo sử dụng cấu trúc R^S-Tree cho tìm kiếm ảnh. Thuật toán xây dựng R^S-Tree được trình bày như trong **Thuật toán 4**.

Gọi n là số phần tử của tập dữ liệu, M là số phần tử tối đa trong một nút của R^S-Tree. Thuật toán CRST lần lượt thực hiện duyệt từ nút gốc đến nút lá, mỗi lần duyệt qua

Thuật toán 4: Xây dựng R^S-Tree

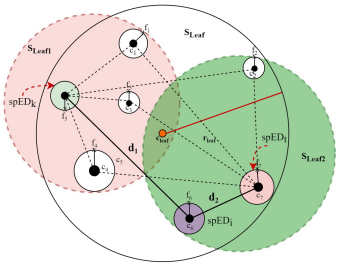
Input: Nút S_N, ngưỡng M và phần tử dữ liệu S_{spED}
Output: Cây R^S-Tree sau khi thêm phần tử
Function: **CRST** (S_N,$spED$)
begin
  if (S_N *không phải nút lá*) **then**
    Chọn nhánh gần với phần tử được thêm vào theo độ do Euclidean cho đến khi gặp được nút lá hiện hành S_{Lcrt};
  else
    if (S_{Lcrt} *có số phần tử nhỏ hơn* M) **then**
      Tính khoảng cách Euclidean $d = Euclidean(S_{spED}.\vec{c}, S_{Lcrt}.\vec{c}) + S_{spED}.r$;
      if ($d < \theta$) **then**
        Thêm phần tử S_{spED} vào nút lá S_{Lcrt};
        Cập nhật tâm cụm;
      else
        Tạo một nút lá mới S_{Lnew} cùng cha với nút lá hiện hành;
        Thêm phần tử S_{spED} vào nút lá mới S_{Lnew};
        Cập nhật tâm cụm;
      end
    else
      Thực hiện tách nút lá S_{leafk};
    end
  end
  return R^S-Tree;
end

M phần tử, mỗi lần duyệt thuật toán CRST thực hiện phép cập nhật tâm và tách nút từ nút lá đến nút gốc. M là hằng số. Do đó, thuật toán CRST có độ phức tạp là $O(logn)^3$.

6. Cải tiến tách nút trên R^S-Tree

Thuật toán tách nút trong công trình [13], việc phân hoạch tập các phần tử về hai nút con được thực hiện bằng cách lựa chọn tuần tự ngẫu nhiên trong tập $M-1$ phần tử để phân phối lần lượt vào hai nút con. Trong bài báo này, thuật toán tách nút được cải tiến như sau: tập $M-1$ phần tử sẽ được tính độ đo sai biệt so với hai phần tử được chọn làm hai tâm của hai nút mới. Trong quá trình tách nút lá thành hai nút con, việc lựa chọn phần tử $spED_i$ phân phối vào hai nút lá mới cho phù hợp được thực hiện dựa theo độ sai biệt (ký hiệu là d_{Dif}) được minh họa như trong Hình 4, phần tử có độ đo sai biệt lớn sẽ được chọn để phân phối trước.

Trong Hình 4, độ đo sai biệt (d_{Dif}) của phần tử $spED_i$ so với hai phần tử $spED_k$ và $spED_t$ là hai phần tử được chọn làm tâm của hai nút lá mới S_{Leaf1} và S_{Leaf2} được tính như sau: $d_{Dif} = |d_1 - d_2|$, trong đó $d_1 = d_E(spED_i.\vec{c}_{spi}, spED_k.\vec{c}_{spk}), d_2 = d_E(spED_i.\vec{c}_{spi}, spED_t.\vec{c}_{spt})$. Độ sai biệt càng lớn thì khả năng xác định phần tử thuộc về một trong hai nút càng cao.



Hình 4. Mô tả thuật toán tách nút dựa vào độ đo sai biệt

Để thực hiện tách nút lá thành hai nút con, hai khối cầu S_{spEDk}, S_{spEDt} xa nhau nhất trong nút tách sẽ được chọn để làm tâm khởi đầu hai nút con. Việc chọn hai tâm được thực hiện theo **Thuật toán 5**.

Thuật toán 5: Lựa chọn hai phần tử làm tâm hai nút con

Input: Nút lá S_{Leaf}
Output: Hai khối cầu phần tử $spED_k$, $spED_t$
Function: **SelectedSpEDs** (S_{Leaf})
begin
  Khởi tạo $d_{crmax} = 0$;
  Khởi tạo $spED_k$ = null;
  Khởi tạo $spED_t$ = null;
  for *Duyệt các phần tử trong nút lá* S_{Leaf} **do**
    Tính khoảng cách Euclidean;
    Chọn phần từ xa tâm nút lá S_{Leaf} nhất gán giá trị cho phần tử $spED_k$;
  end
  $S_{Leaf} = S_{Leaf}$-$spED_k$;
  for *Duyệt các phần tử trong nút lá* S_{Leaf} **do**
    Chọn phần tử xa phần tử $spED_k$ gán cho $spED_t$;
  end
  return {$spED_k$,$spED_t$};
end

Để phân phối các phần tử vào hai nút lá được tối ưu, các phần tử trong danh sách nút lá cần tách được sắp xếp theo độ đo theo độ sai biệt, thuật toán được trình bày như trong **Thuật toán 6**.

Sau khi lựa chọn được hai phần tử khối cầu làm tâm của hai nút lá được tách ra, các phần tử còn lại lần lượt được phân phối vào hai nút đã tách. Quá trình phân phối được thực hiện quy tắc chọn nhánh tốt nhất. Thuật toán phân phối các phần tử được trình bày như trong **Thuật toán 7**.

Nút S_{Leaf} bị tràn được xử lý bằng cách tạo hai nút mới S_{Leaf1}, S_{Leaf2}, cùng mức với S_{Leaf}, phân vùng $M+1$ phần tử vào hai nút. Khi quá trình phân tách xảy ra có thể lan truyền đến nút trong, thực hiện tách nút trong. Quá trình này có thể lan truyền đến nút gốc, khi nút gốc đầy một nút gốc mới sẽ được tạo ra. Thuật toán tách nút lá được trình

Thuật toán 6: Sắp xếp các phần tử theo độ đo sai biệt

Input: Ba nút lá S_{Leaf}, S_{Leaf1}, S_{Leaf2}
Output: Danh sách các phần tử sau khi sắp xếp S_{Lsort}
Function: **SortListSpED** (S_{Leaf},S_{Leaf1},S_{Leaf2})
begin
  for *Duyệt các phần tử trong nút lá* S_{Leaf} **do**
    Tính các khoảng cách Euclidean
    d_1=$Euclidean(S_{Leaf1}.\overrightarrow{c}, S_{Leaf}.SpED[i].\overrightarrow{c})$;
    d_2=$Euclidean(S_{Leaf2}.\overrightarrow{c}, S_{Leaf}.SpED[i].\overrightarrow{c})$;
    Tính độ đo sai biệt; $d_i = |d_1 - d_2|$;
    Đưa các phần tử vào danh sách $List_{SL}$;
    $List_{SL}.Add(S_{Leaf}.SpED[i], d_i)$;
  end
  Sắp xếp các phần tử theo độ đo sai biệt; $S_{Lsort} = List_{SL}.Sort$;
  return S_{Lsort};
end

Thuật toán 7: Phân phối các phần tử vào hai nút lá

Input: Ba nút lá cần tách S_{Leaf}, S_{Leaf1}, S_{Leaf2}
Output: Các nút sau khi phân phối xong
Function: **DistributeSpED** (S_{Leaf},S_{Leaf1},S_{Leaf2})
begin
  Khởi tạo
  $S_{temp} = SortListSpED(S_{Leaf}, S_{Leaf1}, S_{Leaf2})$;
  while $(S_{temp} == \emptyset || S_{Leaf1}.size! = M - m + 1 || S_{Leaf2}.size! = M - m + 1)$ **do**
    Tính khoảng cách Euclidean đến tâm của hai nút lá S_{Leaf1}, S_{Leaf2};
    if $(Euclidean(S_{Leaf1}.\overrightarrow{c}, spED.\overrightarrow{c} < Euclidean(S_{Leaf2}.\overrightarrow{c}, spED.\overrightarrow{c}))$ **then**
      Đưa phần tử spED vào nút lá S_{Leaf1};
      Cập nhật tâm;
    else
      Đưa phần tử spED vào nút lá S_{Leaf1};
      Cập nhật tâm;
      $S_{temp} = S_{temp} - \{spED\}$;
    end
  end
  if $(S_{Leaf1}.size == M - m + 1)$ **then**
    $S_{Leaf2} = S_{Leaf2} \cup S_{temp}$;
  else
    $S_{Leaf1} = S_{Leaf1} \cup S_{temp}$;
  end
  return $\{S_{Leaf1}, S_{Leaf2}\}$;
end

bày như **Thuật toán 8**.

Gọi n là số phần tử của tập dữ liệu, M là số phần tử tối đa trong một nút R^S-Tree. Khi thực hiện tách nút, trong trường hợp xấu nhất, Thuật toán SplitLeafRST phải tách từ nút lá đến nút gốc. Mỗi lần tách nút, Thuật toán SplitLeafRST phải thực hiện M phép so sánh để phân bố về k-cụm. Mặt khác, mỗi lần thực hiện tách nút, trong trường hợp xấu nhất Thuật toán SplitLeafRST phải gọi đệ

Thuật toán 8: Thuật toán tách nút lá

Input: Nút lá cần tách S_{Leaf}
Output: Cây R^S-Tree sau khi tách
Function: **SplitLeafRST** (S_{Leaf})
begin
  Tạo nút lá S_{Leaf1}; Tạo nút lá S_{Leaf2};
  Chọn hai phần tử xa nhau nhất trong nút lá S_{Leaf};
  SelectedSpEDs(S_{Leaf});
  Đưa $spED_k$ vào nút lá S_{Leaf1};
  Cập nhật tâm và bán kính;
  Đưa $spED_t$ vào nút lá S_{Leaf2};
  Cập nhật tâm và bán kính;
  Xóa hai phần tử này ra khỏi S_{Leaf};
  Gọi hàm sắp xếp các phần tử theo độ đo sai biệt;
  S_{Lsort} = SortListSpED(S_{Leaf},S_{Leaf1},S_{Leaf2});
  Gọi hàm phân phối các phần tử vào hai nút;
  DistributeSpED(S_{Lsort},S_{Leaf1},S_{Leaf2});
  S_{Lpr} = S_{Leaf}.parent;
  if (*Số phần tử nút cha* S_{Lpr} *nhỏ hơn* N) **then**
    Cập nhật tâm nút cha;
  else
    Tách nút cha;
  end
end

quy từ nút lá đến nút gốc. M là hằng số. Do đó, độ phức tạp của Thuật toán SplitLeafRST là $O(logn)^2$.

7. Tìm kiếm ảnh tương tự dựa trên R-KD-Tree

Từ cây phân cụm dữ liệu không gian R^S-Tree đã tạo, một thuật toán tra cứu ảnh tương tự theo nội dung dựa trên cây R^S-Tree được đề xuất. Quá trình tìm kiếm ảnh tương tự được thực hiện trên cây R^S-Tree và được mô tả như **Thuật toán 9**.

Thuật toán 9: Thuật toán tìm kiếm ảnh trên R-KD-Tree

Input: Đặc trưng ảnh truy vấn $spED$, R^S-Tree
Output: Tập ảnh tương tự SI
Function: **RSIR** (S_{Nr}, $spED$)
begin
  $S_{Node} = S_{Nr}$;
  if (S_{Node} *trỏ tới phần tử null*) **then**
    return *null*;
  else
    if (S_{Nk} *không phải lá nút lá*) **then**
      Đi theo nhánh gần nhất với phần tử truy vấn ký hiệu S_{Nk};
      RSIR (S_{Nk}, $spED$);
    else
      SI = {Các phần tử ảnh tương tự trong nút lá hiện hành S_{Lcrt}};
    end
  end
  return SI;
end

Bảng I
MÔ TẢ CÁC THAM SỐ THỰC NGHIỆM

Tham số	COREL	Caltech-101
M	120	100
m	1	1
N	20	25
θ	0.078	0.085
$Topk$	80	50-100
Thời gian xây dựng cây R^S-Tree (giờ)	0.41	2.79

Gọi n là số phần tử của tập dữ liệu, M là số phần tử tối đa trong một nút của R^S-Tree. Thuật toán RSIR lần lượt duyệt qua các nút từ gốc đến lá, hơn nữa vì cây R^S-Tree là cây cân bằng nên thuật toán RSIR duyệt qua chiều cao h của cây. Mỗi lần duyệt, thuật toán RSIR phải so sánh với M phần tử của mỗi nút. M là hằng số. Do đó, độ phức tạp của thuật toán RSIR là $O(logn)$.

V. THỰC NGHIỆM

1. Mô tả dữ liệu thực nghiệm

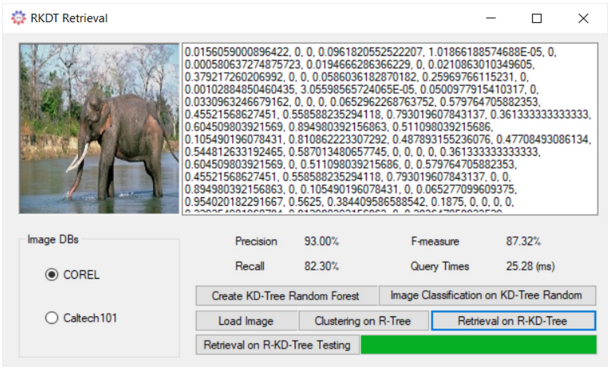
Để đánh giá hiệu năng của các hệ thống truy vấn ảnh, một số bộ dữ liệu ảnh tiêu chuẩn được sử dụng cho quá trình thực nghiệm gồm COREL, Caltech101. Bộ ảnh COREL gồm 1,000 ảnh JPEG chia thành 10 chủ đề khác nhau. Mỗi chủ đề gồm 100 ảnh. Bộ ảnh Caltech-101 gồm 9,144 ảnh JPEG chia thành 102 chủ đề. Mỗi chủ đề chứa từ 40 đến 800 ảnh. Mỗi ảnh có kích thước từ 200 – 300 pixels. Mỗi hình ảnh được trích xuất thành một véc-tơ đặc trưng có 81 thành phần. Quá trình trích xuất véc-tơ đặc trưng được kế thừa từ công trình [16]. Dữ liệu thực nghiệm trong bài báo này không có ràng buộc thêm so với các công trình khác thực nghiệm trên cùng bộ ảnh.

2. Thực nghiệm và đánh giá

Trong bài báo này, các tham số M, m, N, θ được tiến hành thực nghiệm để lựa chọn giá trị nhằm nâng cao độ chính xác truy vấn. Gọi $nMax_{inclass}$ là số phần tử tối đa trong một phân lớp. Chúng tôi thực nghiệm chọn $M \in [nMax_{inclass} - \alpha, nMax_{inclass} + \alpha]$, α là hằng số; $N \in [2, M]$. Ngưỡng θ để đánh giá độ tương tự các phần tử thuộc một cụm. Số phần tử của mỗi phân lớp xấp xỉ 10% của bộ dữ liệu. Do đó, chúng tôi thực nghiệm giá trị $\theta \in [0.1 - \mu, 0.1 + \mu], \mu \in [0, 0.05]$ là hệ số học. Thông qua quá trình thực nghiệm, các tham số tối ưu được lựa chọn sao cho hệ thống đạt được độ chính xác tốt nhất được trình bày như Bảng I.

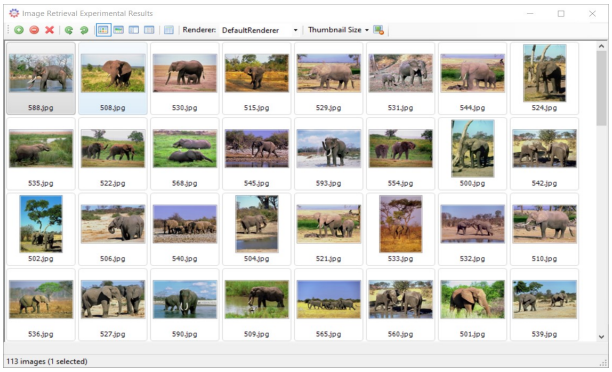
Việc lựa chọn tham số trong quá trình thực nghiệm phụ thuộc vào tập dữ liệu bao gồm: (1) số các phần tử dữ liệu mẫu trong một phân lớp; (2) số lượng phần tử trong tập dữ liệu ảnh. Số phần tử trong một phân lớp càng lớn thì tham số M càng lớn. Số lượng phần tử trong tập dữ liệu lớn thì N càng lớn nhằm hạn chế chiều cao của cây giúp giảm thời gian tìm kiếm. Ngưỡng θ phụ thuộc vào độ tương tự của các phần tử trong một phân lớp, nếu các phần tử trong một phân lớp càng gần nhau thì ngưỡng θ càng bé. $TopK$ càng lớn độ chính xác càng thấp, độ phủ càng cao. Tham số $TopK$ được lựa chọn sao cho độ đo dung hòa đạt tốt nhất.

Thực nghiệm tìm kiếm ảnh tương tự dựa trên cấu trúc R-Tree kết hợp KD-Tree Random Forest được minh họa bởi hình 5. Hệ truy vấn ảnh RKDT dựa trên kết quả phân lớp ảnh đầu vào (Load Image) bằng KD-Tree Random Forest (Image Classification on KD-Tree Random Forest), sau khi đã xây dựng và huấn luyện mô hình phân lớp bằng rừng ngẫu nhiên là nhiều cấu trúc KD-Tree (Create KD-Tree Randm Forest). Sau khi thực hiện phân lớp cho ảnh đầu vào là quá trình gom cụm bằng R-Tree (Clustering on R-Tree). Dựa trên kết quả gom cụm từ R-Tree để tìm kiếm và trích xuất tập ảnh tương tự với ảnh đầu vào sau khi trích xuất véc-tơ đặc trưng.

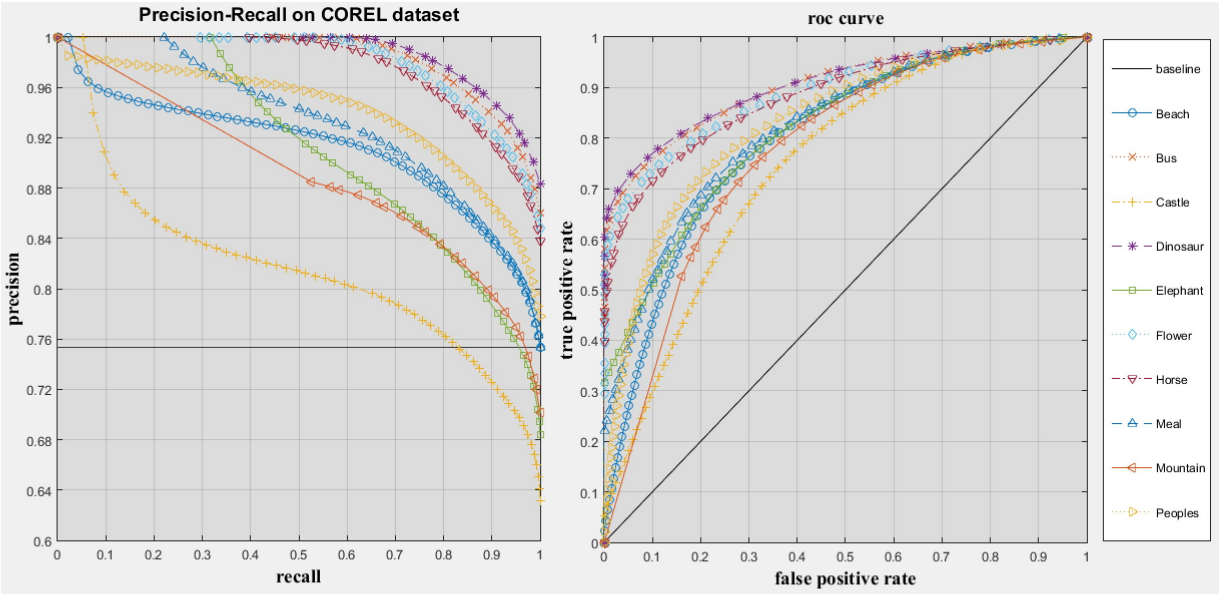


Hình 5. Hệ truy vấn ảnh RKDT

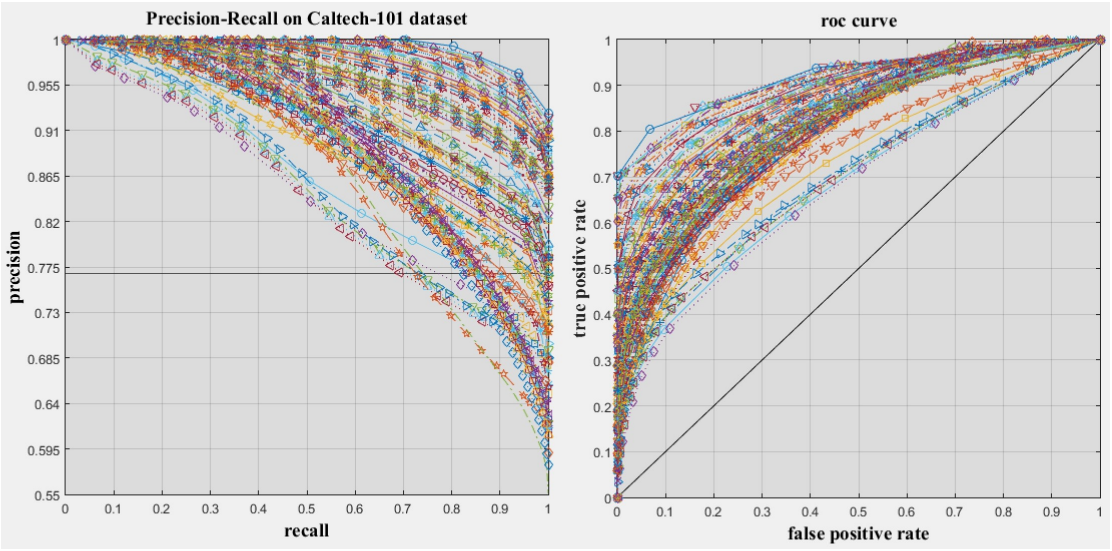
Kết quả tìm kiếm tập ảnh tương tự với ảnh đầu vào (ảnh 588.jpg bộ COREL) được minh họa bởi Hình 6.



Hình 6. Kết quả truy vấn ảnh 588.jpg ($COREL$)



Hình 7. Biểu đồ Precision, Recall và ROC bộ ảnh COREL



Hình 8. Biểu đồ Precision, Recall và ROC bộ ảnh Caltech101

Bảng II
HIỆU SUẤT PHÂN LỚP ẢNH BẰNG
KD-TREE RANDOM FOREST

Bộ dữ liệu	Số ảnh	Hiệu suất phân lớp
COREL	1,000	0.8972
Caltech101	9,144	0.7705

Sau khi xây dựng và huấn luyện mô hình phân lớp ảnh bằng KD-Tree Random Forest kết quả thực nghiệm phân lớp bằng rừng ngẫu nhiên được trình bày trong Bảng II.

Kết quả thực nghiệm tìm kiếm ảnh trên các bộ ảnh COREL, Caltech101 với độ chính xác (*Precision*), độ phủ (*Recall*) và độ dung hòa (*F-measure*), thời gian truy vấn trung bình của các bộ ảnh (*Time query*) tính theo mili giây (*ms*) được trình bày trong Bảng III. Kết quả này cho thấy hiệu suất tìm kiếm ảnh trên các bộ ảnh khi kết hợp hai cấu trúc này là cao hơn so với các công trình sử dụng riêng lẻ từng cấu trúc trong công trình [15, 16].

Kết quả thực nghiệm trên các bộ ảnh COREL, Caltech101 được minh họa bằng biểu đồ ROC thực hiện trên Matlab 2015 được trình bày như Hình 7 – Hình 8.

Bảng IV, trình bày so sánh thực nghiệm tìm kiếm ảnh hệ truy vấn RKDT trên từng bộ dữ liệu với các công trình

Bảng III
HIỆU SUẤT TRUY VẤN ẢNH TRÊN HỆ **RKDT**

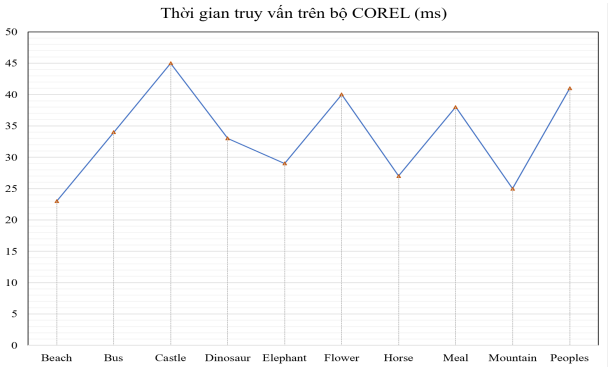
Bộ ảnh	Precision	Recall	F-measure	Query time
COREL	**0.8101**	0.7344	0.7703	33.16
Caltech101	**0.7354**	0.7020	0.7183	98.12

Bảng IV
SO SÁNH HIỆU SUẤT TRUY VẤN VỚI CÁC PHƯƠNG PHÁP KHÁC TRÊN CÙNG BỘ ẢNH

Phương pháp	Tập ảnh	Precision
B-SHIFT (2016), [11]	COREL	0.7200
k-NN KD-Tree (2019), [12]	COREL	0.6430
ORB 8-D with MPL (2020), [24]	COREL	0.6656
RKDT	**COREL**	**0.8101**
BDIPs, GLCM - SVM (2020), [13]	Caltech101	0.6314
MTSD, (2020), [14]	Caltech101	0.6942
Feature Detector (2021), [25]	Caltech101	0.6200
RKDT	**Caltech101**	**0.7354**

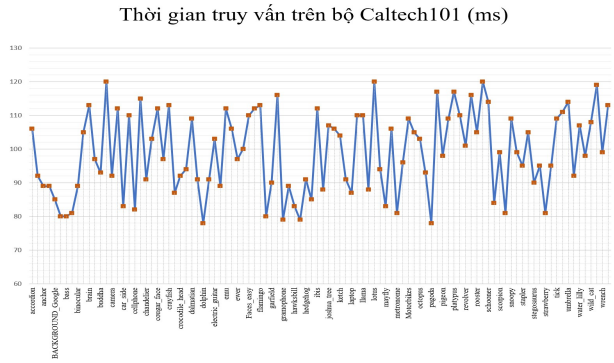
khác là cao hơn bởi các yếu tố: (1) hệ truy vấn RKDT đã kết hợp được cấu trúc R-Tree với KD-Tree nên phát triển những ưu điểm từ hai cấu trúc này; (2) Quá trình phân lớp trên KD-Tree Random For-est trước khi gom cụm trên R-Tree đã góp phần nâng cao hiệu suất truy vấn ảnh. Đồng thời, kết quả thực nghiệm bộ ảnh COREL cho thấy sự kết hợp cấu trúc KD-Tree và R-Tree đã mang lại hiệu suất tìm kiếm ảnh cao hơn so với các công trình đã công bố bởi chúng tôi đã công bố trước đây khi chỉ dùng một cấu trúc KD-Tree hoặc R-Tree riêng lẻ [15, 16].

Hình 9 và Hình 10 biểu diễn hiệu suất thời gian truy vấn trung bình cho từng phân lớp trên các bộ ảnh COREL và Caltech101.



Hình 9. Thời gian truy vấn trung bình của bộ ảnh COREL

Trong bài báo này, kết quả hiệu suất truy vấn ảnh được thực hiện so sánh với một số công trình khác trên cùng bộ ảnh thực nghiệm nhưng không so sánh chi phí thời gian huấn luyện mô hình phân lớp vì cấu hình máy tính không đồng nhất. Đồng thời, chi phí huấn luyện mô hình phân lớp KD-Tree Random Forest chỉ thực hiện một lần và sau đó được thực thi nhiều lần với bộ trọng số đã huấn



Hình 10. Thời gian truy vấn trung bình của bộ ảnh Caltech101

luyện. Kết quả thời gian truy vấn ảnh trung bình trên các bộ ảnh COREL, Cal-tech101 cho thấy phương pháp phân lớp ảnh bằng KD-Tree Random Forest là hoàn toàn khả thi và hiệu quả. Từ kết quả phân lớp này thực hiện gom cụm trên R-Tree đã mang lại hiệu suất truy vấn ảnh được đánh giá là khá cao.

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong bài báo này, một mô hình truy vấn ảnh có sự kết hợp cấu trúc R-Tree với KD-Tree Random Forest được đề xuất và thực nghiệm trên các bộ ảnh COREL, Caltech101. Sự kết hợp hai cấu trúc được xây dựng theo hướng tiếp cận gom cụm và phân lớp đã góp phần nâng cao hiệu suất tìm kiếm ảnh vì phát huy được những ưu điểm của từng cấu trúc và bổ trợ cho nhau. Thực nghiệm truy vấn ảnh trên các bộ ảnh COREL, Caltech101 với độ chính xác trung bình tương ứng là 0.8101, 0.7354. Trong công trình này đã cải tiến nâng cao hiệu suất cho quá trình phân lớp bằng KD-Tree Random Forest trước khi gom cụm và tìm kiếm trên R-Tree. Định hướng phát triển tiếp theo của chúng tôi là từ kết quả phân lớp bằng KD-Tree Random Forest kết hợp cải tiến R-Tree và đồ thị làng giềng để nhằm thực hiện truy vấn trên ontology nhằm nâng cao hiệu suất cho bài toán tìm kiếm ảnh theo ngữ nghĩa và áp dụng cho nhiều bộ dữ liệu nhằm đa dạng tập ảnh thực nghiệm.

LỜI CẢM ƠN

Chúng tôi xin trân trọng cảm ơn Khoa Công nghệ thông tin – Trường Đại học Khoa học - Đại học Huế, nhóm nghiên cứu SBIR-HCM đã góp ý chuyên môn cho nghiên cứu này. Chúng tôi xin trân trọng cảm ơn Trường Đại học Sư phạm TP.HCM đã tạo điều kiện về cơ sở vất chất giúp chúng tôi hoàn thành bài nghiên cứu này.

TÀI LIỆU THAM KHẢO

[1] A Patrizio, "Data center explorer", *Network World*, https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html, (31/3/2022).

[2] Le, Thanh Manh, et al, "Image retrieval system based on EMD similarity measure and S-tree", *In: Intelligent Technologies and Engineering Systems, Springer, New York, NY*, pp. 139-146, 2013.

[3] Chandresh Pratap Singh, "R-Tree implementation of image databases", *Sig-nal and Image Processing: An International Journal (SIPIJ)*, vol. 18.4, 2011.

[4] Corel 1k Database: http://wang.ist.psu.edu/docs/related/, (21/2/2022)

[5] Caltech101: http://www.vision.caltech.edu/Images Dataset/-Caltech101, (21/2/2022).

[6] Zhang, Yuqian, et al, "Fast face sketch synthesis via kd-tree search", *European Conference on Computer Vision. Springer, Cham*, 2016.

[7] Vanitha J., Senthilmurugan M, "Multidimensional Image Indexing Using SR-Tree Algorithm for Content-Based Image Retrieval System", *In: Shetty N., Patnaik L., Prasad N., Nalini N. (eds) Emerging Research in Computing, In-formation, Communication and Applications*, pp. 81-88, 2018.

[8] Arpitha Y.C, Bhargavi A.G, Mahima K.T, Nagavi K.K, Meenakshi H.N, "A Navigation Supporting System Using R-Tree", *Published by AIJR Publisher in Proceedings of the 3rd National Conference on Image Processing, Compu-ting, Communication, Networking and Data Analytics*, 2018.

[9] Xinlu Wang, Weiming Meng, Mingchuan Zhang, "A novel information re-trieval method based on R-tree index for smart hospital information system", *International Journal of Advanced Computer Research*, vol. 9.41, 2019.

[10] Bentley, Jon Louis, "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, vol. 18.9, pp. 509-517, 1975.

[11] Douik, Ali, Mehrez Abdellaoui, and Leila Kabbai, "Content based image re-trieval using local and global features descriptor", *2016 2nd international conference on advanced Technologies for Signal and Image Processing (ATSIP) IEEE*, pp. 151-154, 2016.

[12] Abdulkadhem, Abdulkadhem Abdulkareem, and Tawfiq A. Al-Assadi, "Pro-posed a Content-Based Image Retrieval System Based on the Shape and Tex-ture Features", *Int. J. Innovat. Technol. Explor*, vol. 8, pp. 2189, 2019.

[13] Rajesh, M. B., Sathiamoorthy, S., "Co-occurrence of Edges and Valleys with Support Vector Machine for Content Based Image retrieval", *In 2020 Fourth International Conference on Computing Methodologies and Communication*, pp. 465-470, 2020.

[14] Sathiamoorthy, S., Natarajan, M., "An efficient content based image retrieval using enhanced multi-trend structure descriptor", *SN Applied Sciences*, vol. 2.2, pp. 1-20, 2020.

[15] Lê Thị Vĩnh Thanh, Văn Thế Thành, Lê Mạnh Thạnh, "Một phương pháp tìm kiếm ảnh hiệu quả dựa trên cấu trúc R-Tree", *Kỷ yếu Hội thảo Quốc gia về Công nghệ thông tin và ứng dụng trong các lĩnh vực (CITA2021), Đại học Đà Nẵng, Nhà xuất bản Đà Nẵng, ISBN: 978-604-84-5998-7*, trang 259-271, 2021.

[16] Nguyễn Thị Định, Văn Thế Thành, Lê Mạnh Thạnh, "Phân lớp ảnh bằng cây KD-Tree cho bài toán tìm kiếm ảnh tương tự", *Chuyên san Các công trình nghiên cứu, phát triển và ứng dụng Công nghệ thông tin và Truyền thông*, Tập 2021, Số 1, 2021.

[17] Zaklouta, F., Stanciulescu, B., and Hamdoun, O., "Traffic sign classification us-ing kd trees and random forests", *In The 2011 international joint conference on neural networks, IEEE*, pp. 2151-2155, 2011.

[18] Man, W., Ji, Y., and Zhang, Z., "Image classification based on improved random forest algorithm", *In 2018 IEEE 3rd International Conference on Cloud Com-puting and Big Data Analysis (ICCCBDA)*, pp. 346-350, 2018.

[19] Amini, S., Homayouni, S., Safari, A., and Darvishsefat, A, "Object-based classi-fication of hyperspectral data using Random Forest algorithm", *Geo-spatial in-formation science*, vol. 21.2, pp. 127-138, 2018.

[20] Jain, V., and Phophalia, A, "M-ary Random Forest-A new multidimensional par-titioning approach to Random Forest", *Multimedia Tools and Applications*, vol. 80.28, pp. 35217-35238, 2021.

[21] Kabari, L. G., and Onwuka, U. C, "Comparison of Bagging and Voting Ensemble Machine Learning Algorithm as a Classifier", *International Journals of Ad-vanced Research in Computer Science and Software Engineering*, vol. 9.3, pp. 19-23, 2019.

[22] Zhang, J., and Shi, H, "Kd-tree based efficient ensemble classification algorithm for imbalanced learning", *In 2019 international conference on machine learn-ing, big data and business intelligence (MLBDBI), IEEE*, pp. 203-207, 2019.

[23] Wang, A., Wang, Y., and Chen, Y, "Hyperspectral image classification based on convolutional neural network and random forest", *Remote sensing letters*, vol. 10.11, pp. 1086-1094, 2019.

[24] Chhabra, P., Garg, N. K., and Kumar, M, "Content-based image retrieval system using ORB and SIFT features", *Neural Computing and Applications*, vol. 32.7, pp. 2725-2733, 2020.

[25] Ahmed, K. T., Aslam, S., Afzal, H., Iqbal, S., Mehmood, A., and Choi, G. S, "Symmetric Image Contents Analysis and Retrieval Using Decimation", *Pattern Analysis, Orientation, and Features Fusion. IEEE Access*, vol. 9, pp. 57215-57242, 2021.

[26] Nguyễn Thị Định, Thế Thành Văn, Mạnh Thạnh Lê, "Một phương pháp phân lớp dựa trên cấu trúc KD-Tree cho bài toán tìm kiếm ảnh theo ngữ nghĩa", *Kỹ yếu hội nghị quốc gia lần thứ 14 về nghiên cứu cơ bản và ứng dụng công nghệ thông tin, Fair2021, TP. Hồ Chí Minh, 23/12/2021*. DOI: 10.15625/vap.2021.0075. (2021).

[27] Dinh, N. T., and Le, T. M, "An Improvement Method of Kd-Tree Using k-Means and k-NN for Semantic-Based Image Retrieval System", *In World Conference on Information Systems and Technologies, Springer, Cham*, pp. 177-187, 2022.

[28] Thanh, L. T. V., and Thanh, L. M, "Semantic-Based Image Retrieval Using-Tree and Neighbor Graph", *In World Conference on Information Systems and Technologies . Springer, Cham*, pp. 165-176, 2022.

SƠ LƯỢC VỀ TÁC GIẢ

Lê Mạnh Thạnh

Sinh năm 1953, nhận bằng Tiến sĩ ngành khoa học máy tính tại Đại học Budapest (ELTE), Hungary vào năm 1994. Nhận hàm Phó giáo sư tại trường Đại học Khoa học, Đại học Huế, Việt Nam vào năm 2004. Lĩnh vực nghiên cứu quan tâm bao gồm cơ sở dữ liệu, cơ sở tri thức và lập trình logic. Email: lmthanh@hueuni.edu.vn

Lê Thị Vĩnh Thanh

Sinh năm 1983, tốt nghiệp ngành Sư phạm tin học Trường Đại học Sư phạm TP.HCM vào năm 2006, nhận bằng Thạc sĩ ngành Hệ thống thông tin tại Trường Đại học Khoa học Tự nhiên TP.HCM vào năm 2014. Hiện là nghiên cứu sinh ngành Khoa học máy tính Trường Đại học Khoa học, Đại học Huế. Lĩnh vực nghiên cứu: xử lý ảnh, tìm kiếm ảnh và cơ sở dữ liệu. Email: thanhltv@bvu.edu.vn

Lương Thị Thanh Xuân

Sinh năm 1987, tốt nghiệp ngành Sư phạm Tin học Trường Đại học Đồng Tháp năm 2008. Hiện đang học Thạc sĩ ngành Khoa học máy tính khóa 2020 – 2022 tại trường Đại học Khoa học – Đại học Huế. Lĩnh vực nghiên cứu quan tâm là tìm kiếm ảnh. Email: lttxuan.c23nvk.dtp@moet.edu.vn

Nguyễn Thị Định

Sinh năm 1983, tốt nghiệp ngành Sư phạm tin học Trường Đại học Sư phạm TP.HCM vào năm 2006, nhận bằng Thạc sĩ ngành Truyền số liệu và mạng máy tính tại Học viện Công nghệ Bưu chính viễn thông TP.HCM vào năm 2011. Hiện là nghiên cứu sinh ngành Khoa học máy tính Trường Đại học Khoa học, Đại học Huế. Lĩnh vực nghiên cứu quan tâm gồm xử lý ảnh, tìm kiếm ảnh và cơ sở dữ liệu. Email: dinhnt@hufi.edu.vn

Văn Thế Thành

Sinh năm 1979, tốt nghiệp chuyên ngành Toán tin tại Đại học Khoa học Tự nhiên - Đại học Quốc gia TP.HCM vào năm 2001, nhận bằng Thạc sĩ Khoa học Máy tính tại Đại học Quốc gia TP.HCM vào năm 2008. Năm 2016, nhận bằng Tiến sĩ Khoa học Máy tính tại Đại học Khoa học, Đại học Huế. Lĩnh vực nghiên cứu quan tâm bao gồm xử lý ảnh, khai thác dữ liệu ảnh và tìm kiếm ảnh. Email: thanhvt@hcmue.edu.vn