BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP HCM

****KHOA CÔNG NGHỆ THÔNG TIN

**BÁO CÁO ĐỀ TÀI NGHIÊN CỨU KHOA HỌC**

**SINH VIÊN CẤP TRƯỜNG**

***TÊN ĐỀ TÀI:***

**TÌM KIẾM VÀ ĐỊNH DANH SINH VIÊN QUA HÌNH ẢNH**

|  |  |
| --- | --- |
| Chủ nhiệm đề tài: Nguyễn Thanh Quang – 06DHTH1 | |
| Thành viên tham gia: | |
| Trần Lê Văn Đức – 06DHTH1 |  |
| Đặng Xuân Dương – 08DHTH5  Đặng Hoàng Cẩm My – 09DHTH5  GVHD: Nguyễn Văn Thịnh |  |
|  |  |

*Tp.HCM, tháng 06 năm 2019*

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP HCM

****KHOA CÔNG NGHỆ THÔNG TIN

**BÁO CÁO ĐỀ TÀI NGHIÊN CỨU KHOA HỌC**

**SINH VIÊN CẤP TRƯỜNG**

***TÊN ĐỀ TÀI:***

**TÌM KIẾM VÀ ĐỊNH DANH SINH VIÊN QUA HÌNH ẢNH**

|  |  |
| --- | --- |
| Chủ nhiệm đề tài: Nguyễn Thanh Quang – 06DHTH1 | |
| Thành viên tham gia: | |
| Trần Lê Văn Đức – 06DHTH1 |  |
| Đặng Xuân Dương – 08DHTH5  Đặng Hoàng Cẩm My – 09DHTH5  GVHD: Nguyễn Văn Thịnh |  |
|  |  |

*Tp.HCM, tháng 06 năm 2019*

# LỜI CAM ĐOAN

Chúng em xin cam đoan rằng báo cáo đề tài nghiên cứu khoa học này là do chính nhóm thực hiện dưới sự hướng dẫn của thầy Nguyễn Văn Thịnh các kết quả nghiên cứu và các kết luận trong bài là trung thực, không sao chép từ bất cứ công trình nghiên cứu khoa học nào. Việc tham khảo các nguồn tài liệu đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo theo đúng yêu cầu.

|  |  |  |  |
| --- | --- | --- | --- |
| *Tp.HCM, tháng 06 năm 2019* | | | |
|  |  | **Chủ nhiệm đề tài**  *(Ký và ghi rõ họ tên)* |

# LỜI CẢM ƠN

Trước hết, chúng em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Văn Thịnh, người đã giúp đỡ và hướng dẫn chúng em rất nhiều về định hướng nghiên cứu trong suốt thời gian thực hiện đề tài này.

Chúng em xin cảm ơn trường Đại học Công nghiệp Thực Phẩm TP.Hồ Chí Minh đã tận tình giảng dạy và cung cấp kiến thức cần thiết cho em để hoàn thành quá trình thực hiện đề tài này.

Chúng em xin cảm ơn các thầy cô khoa Công nghệ thông tin đã cung cấp kiến thức, kinh nghiệm chuyên ngành cho em để có thể hoàn thành quá trình thực hiện đề tài này.

Chúng em xin chân thành cảm ơn!

**MỤC LỤC**

[LỜI CAM ĐOAN 1](#_Toc11271981)

[LỜI CẢM ƠN 2](#_Toc11271982)

[CHƯƠNG 1 TỔNG QUAN ĐỂ TÀI 8](#_Toc11271983)

[1.1 Giới thiệu 8](#_Toc11271984)

[1.2 Mục tiêu đề tài 9](#_Toc11271985)

[1.3 Nội dung nghiên cứu 10](#_Toc11271986)

[1.4 Phạm vi đề tài 10](#_Toc11271987)

[1.5 Phương pháp nghiên cứu 10](#_Toc11271988)

[1.6 Công nghệ sử dụng 10](#_Toc11271989)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 12](#_Toc11271990)

[2.1 Giới thiệu về học máy (Machine Learning) 12](#_Toc11271991)

[2.1.1 Khái niệm 12](#_Toc11271992)

[2.1.2 Học có giám sát (Supervised Learning): 12](#_Toc11271993)

[2.1.3 Học không giám sát (Unsupervised Learning): 13](#_Toc11271994)

[2.1.4 Học bán giám sát (Semi-supervised Learning): 14](#_Toc11271995)

[2.2 Các vấn đề cơ bản trong xử lý ảnh 15](#_Toc11271996)

[2.2.1 Xử lý ảnh (XLA) 15](#_Toc11271997)

[2.2.2 Các vấn đề cơ bản trong xử lý ảnh. 17](#_Toc11271998)

[2.3 Tổng quan về OpenCV 18](#_Toc11271999)

[2.3.1 Giới thiệu về OpenCV 18](#_Toc11272000)

[2.3.2 Sơ lược về cấu trúc thư viện OpenCV 19](#_Toc11272001)

[2.4 Tổng quan về dlib 23](#_Toc11272002)

[2.4.1 Giới thiệu về Dlib 23](#_Toc11272003)

[2.4.2 Cách sử dụng dlib từ Python 24](#_Toc11272004)

[2.5 Các vấn đề liên quan trong việc xử lý khuôn mặt người 24](#_Toc11272005)

[2.5.1 Khoảng cách Euclidean 24](#_Toc11272006)

[2.5.2 Xác định khuôn mặt (Face detection) 25](#_Toc11272007)

[2.5.3 Phương pháp trích chọn đặc trưng HOG 25](#_Toc11272008)

[2.5.4 SVM – Support Vector Machine 26](#_Toc11272009)

[2.5.5 Các dặc trưng về khuôn mặt (Facial landmarks) 27](#_Toc11272010)

[2.5.6 Định nghĩa bài toán xác định khuôn mặt 28](#_Toc11272011)

[2.5.7 Tìm kiếm và nhận diện khuôn mặt dựa trên các đặc trưng 34](#_Toc11272012)

[2.6 Cây S-Tree (Signature Tree) 35](#_Toc11272013)

[CHƯƠNG 3 THUẬT TOÁN VÀ MÔ HÌNH TÌM KIẾM ẢNH 36](#_Toc11272014)

[3.1 Tạo chữ ký của hình ảnh (trích xuất vector đặc trưng) 36](#_Toc11272015)

[3.2 Tạo cây S-Tree 39](#_Toc11272016)

[3.2.1 Cấu trúc cây 39](#_Toc11272017)

[3.2.2 Thuật toán tạo cây S-Tree 40](#_Toc11272018)

[3.2.3 Thuật toán tìm kiếm ảnh trên cây 43](#_Toc11272019)

[CHƯƠNG 4 THỰC NGHIỆM 45](#_Toc11272020)

[4.1 Mô hình thực nghiệm 45](#_Toc11272021)

[4.2 Dữ liệu thực nghiệm 46](#_Toc11272022)

[4.3 Ứng dụng thực nghiệm 47](#_Toc11272023)

[4.3.1 Giao diện ứng dụng 47](#_Toc11272024)

[4.3.2 Thống kê thực nghiệm 49](#_Toc11272025)

[CHƯƠNG 5 KẾT LUẬN 50](#_Toc11272026)

[TÀI LIỆU THAM KHẢO 51](#_Toc11272027)

**DANH MỤC HÌNH ẢNH**

[Hình 2-1 Các bước cơ bản trong một hệ thống xử lý ảnh 16](#_Toc11270766)

[Hình 2-2 Hình ảnh 68 điểm đặc trưng của khuôn mặt 28](#_Toc11270767)

[Hình 2-3 Hệ thống nhận dạng mặt người 29](#_Toc11270768)

[Hình 2-4 Hình ảnh đưa về hình ảnh đen trắng 30](#_Toc11270769)

[Hình 2-5 Xét độ đậm nhạt của pixel với các pixel xung quanh 30](#_Toc11270769)

[Hình 2-6 Lặp lại quá trình xét độ đậm nhạt cho mỗi pixel đơn trong hình 30](#_Toc11270769)

[Hình 2-7 Hình ảnh biểu diễn đơn giản cấu trúc cơ bản của khuôn mặt 31](#_Toc11270769)

[Hình 2-8 Sự thay đổi về sự sáng tối trong bức ảnh 32](#_Toc11270769)

[Hình 2-9 Hai hình ảnh thằng mặt và nghiêng mặt của Will Ferrell 32](#_Toc11270769)

[Hình 2-10 Hình ảnh các điểm đặc trưng trên khuôn mặt (facial landmarks) 33](#_Toc11270769)

[Hình 2-11 Hình ảnh đã phân biệt được các đặc điểm mắt, mũi, miệng 34](#_Toc11270769)

[Hình 2-12 Hình ảnh căn chỉnh thẳng của khuôn mặt 34](#_Toc11270769)

[Hình 2-13 Tóm tắt quy trình nhận dạng khuôn mặt 35](#_Toc11270769)

[Hình 3-1 Phương pháp phát hiện khuôn mặt 36](#_Toc11270769)

[Hình 4-1 Mô hình thực nghiệm của đề tài 41](#_Toc11270769)

[Hình 4-2 Tập ảnh thẻ sinh viên 46](#_Toc11270769)

[Hình 4-3 Tập thông tin sinh viên 47](#_Toc11270769)

[Hình 4-4 Giao diện chọn ảnh của ứng dụng 47](#_Toc11270769)

[Hình 4-5 Giao diện kết quả tìm kiếm của ứng dụng 48](#_Toc11270769)

[Hình 4-6 Giao diện hiển thị thông tin sinh viên của ứng dụng 48](#_Toc11270769)

[Hình 4-7 Biểu đồ thống kê thời gian so sánh giữa tuyến tính và cây 49](#_Toc11270769)

**DANH MỤC BẢNG BIỂU**

[Bảng 2‑1 Các thành phần chính của thư viện OpenCV 20](#_Toc11270762)

[Bảng 2‑2 Các thành phần mở rộng của thư viện OpenCV 22](#_Toc11270763)

# TỔNG QUAN ĐỂ TÀI

## 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, …

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,… Năm 2015, tổng số hình ảnh toàn cầu đạt 3.2 nghìn tỷ hình ảnh; năm 2016, có 3.5 triệu hình ảnh được chia sẻ trong mỗi phút và có 2.nghìn tỷ hình ảnh được chia sẻ và lưu trữ trực tuyến. Trong năm 2017, thế giới đã tạo ra 1.2 nghìn tỷ hình ảnh và tổng số ảnh toàn cầu đến năm 2017 là 4.7 2 nghìn tỷ hình ảnh; trong đó, các hình ảnh được tạo ra từ điện thoại thông minh và thiết bị di động là 90% [[1](#Del16)] [[2](#Del18)]. Vì vậy, bài toán khai phá dữ liệu và tra cứu các thông tin liên quan đến hình ảnh phải được quan tâm giải quyết; việc tìm kiếm ảnh tương tự là một trong những bài toán quan trọng của nhiều hệ thống đa phương tiện [[3](#LDe15)] [[4](#MHL14)]. 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. 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.

Trường Đại học Công nghiệp Thực phẩm cũng không nằm ngoài vấn đề đấy. Trường cũng đang đi lên song hành cùng sự phổ biến của công nghệ. Quy mô của trường ngày càng tăng, hệ thống sinh viên ngày càng lớn dẫn tới phần nào sự quá tải trong việc truy xuất dữ liệu sinh viên. Vì vậy việc truy xuất thông tin theo tìm kiếm so khớp thông thường sẽ không còn được nhanh chóng nữa, ảnh hưởng tới hiệu suất của một hệ thống lớn. Trong các trường hợp phải xử lý nhiều tác vụ tìm kiếm liên tục, phần nào đó sẽ dẫn tới quá tải hệ thống. Vì vậy cần có các phương pháp tối ưu hơn trong việc truy xuất thông tin nhanh chóng nhưng vẫn giữ được sự chính xác và một cấu trúc dữ liệu hợp lý bổ trợ cho việc truy xuất.

Việc tìm kiếm hình ảnh trong một cơ sở dữ liệu lớn gồm các ảnh số là công việc khó khăn. Bài toán đặt ra là cần truy vấn hình ảnh trong một hệ thống lớn các dữ liệu hình ảnh một cách hiệu quả. Để giải quyết vấn đề này, một số hệ thống truy vấn ảnh số đã xây dựng như: QBIC, ADL, DBLP, Virage, Alta Vista, SIMPLYcity,… Có hai cách tiếp cận thường dùng để truy vấn hình ảnh là: Truy vấn ảnh dựa trên từ khoá TBIR (Text-Based Image Retrieval) và truy vấn ảnh dựa trên nội dung CBIR (Content-Based Image Retrieval). Hệ thống TBIR có thể đưa ra các hình ảnh với nội dung không liên quan đến yêu cầu truy vấn bởi vì bản chất của từ khoá độc lập với nội dung hình ảnh. Để khắc phục vấn đề này, hệ thống CBIR sẽ rút trích thuộc tính thị giác của hình ảnh cần truy vấn, sau đó sẽ so sánh với các thuộc tính thị giác của các hình ảnh khác đã được lưu trữ trong dữ liệu ảnh. Tuy nhiên, nếu phương pháp so sánh độ tương tự về nội dung không hiệu quả thì kết quả truy vấn sẽ đưa ra các hình ảnh có nội dung không liên quan đến yêu cầu truy vấn. Trong đề tài sẽ xây dựng phương pháp đánh giá độ tương tự của hình ảnh dựa trên khoảng cách Euclide và phương pháp truy vấn hình ảnh dựa trên cây chữ ký S-tree. Các công trình về truy vấn hình ảnh dựa trên nội dung như: hệ truy vấn ảnh dựa trên histogram, lượng tử hoá và so sánh độ tương tự của hình ảnh dựa trên histogram, truy vấn hình ảnh dựa trên độ tương tự của hình ảnh, truy vấn ảnh dựa trên histogram và cấu trúc hình ảnh, … Đề tài sẽ tiếp cận việc mô tả ngữ nghĩa về mặt nội dung của hình ảnh thông qua chữ ký của ảnh đồng thời xây dựng lưu trữ chữ ký này lên cây S-tree. Cấu trúc dữ liệu S-tree sẽ mô tả mối quan hệ giữa các chữ ký, từ đó mô tả mối quan hệ giữa các nội dung của hình ảnh. Dựa trên việc mô tả mối quan hệ ngữ nghĩa nội dung hình ảnh của cấu trúc dữ liệu S-tree, chúng em sẽ tiến hành tìm ra các hình ảnh sinh viên tương tự theo nội dung trên bộ ảnh thẻ sinh viên khoa công nghệ thông tin trường đại học Công Nghiệp Thực Phẩm.

## Mục tiêu đề tài

* Nghiên cứu về phương pháp phát hiện khuôn mặt trong hình ảnh, trích xuất vector đặc trưng của khuôn mặt.
* Nghiên cứu về hệ thống tra cứu hình ảnh để áp dụng vào việc định danh sinh viên.
* Đề tài thực hiện nghiên cứu độ đo Euclide và cây S-Tree để phân cụm các vector theo phương pháp phân cấp. Chuyển đổi các hình ảnh khuôn mặt thành vector nhị phân dựa trên các đặc trưng của khuôn mặt.
* Xây dựng ứng dụng tìm kiếm và định danh sinh viên qua hình ảnh.

## Nội dung nghiên cứu

* Các thành phần của ảnh và các độ đo tương tự.
* Nghiên cứu ảnh thẻ sinh viên của trường Đại học Công nghiệp Thực phẩm.
* Các thuật toán sử dụng cho việc trích xuất đặc trưng ảnh:
* Thuật toán trích chọn đặc trưng HOG, SVM cho bài toán phát hiện khuôn mặt.
* Thuật toán phát hiện các điểm đặc trưng trên khuôn mặt (facial landmarks detection).
* Mô hình phân cụm dữ liệu – Cây S-Tree.
* Hệ thống truy vấn hình ảnh tương tự.

## Phạm vi đề tài

* Đề tài nghiên cứu về bộ dữ liệu ảnh thẻ và thông tin của bộ ảnh thẻ sinh viên khoa công nghệ thông tin trường đại học Công nghiệp Thực phẩm
* Xây dựng ứng dụng tìm kiếm và định danh sinh viên khoa công nghệ thông tin trường Đại học Công nghiệp Thực phẩm TPHCM qua ảnh thẻ.

## Phương pháp nghiên cứu

Nghiên cứu lý thuyết: nghiên cứu cở sở lý thuyết về các đối tượng trong đề tài từ đó áp dụng vào việc xây dưng mô hình thực nghiệm.

Nghiên cứu thực nghiệm: Tiến hành nghiên cứu trên các bộ dữ liệu chuẩn như LFW, qua đó rút ra được phương pháp tối ưu từ dó áp dụng cài đặt thực nghiệm trên bộ dữ liệu ảnh thẻ sinh viên.

## Công nghệ sử dụng

Ngôn ngữ sử dụng: Python 2.7.

Trình biên dịch: Visual studio code.

Các thư viện sử dụng:

* openCV: thư viện xử lý ảnh mã nguồn mở.
* Flask: thư viện xây dựng web của Python.
* Một số thư viện khác: math, types, …

# CƠ SỞ LÝ THUYẾT

## Giới thiệu về học máy (Machine Learning)

### Khái niệm

Machine learning (Học máy) – là kỹ thuật để tạo ra kiến ​​thức vào máy hay nói một cách khác là  một ứng dụng trí tuệ nhân tạo cung cấp cho hệ thống khả năng tự động học hỏi và cải thiện từ kinh nghiệm mà không được lập trình rõ ràng. Thay vì viết mã, thì sẽ đưa dữ liệu vào thuật toán chung chung và nó sẽ xây dựng logic riêng dựa trên dữ liệu.**Machine Learning** tập trung vào việc phát triển các chương trình máy tính có thể truy cập dữ liệu và sử dụng nó để tự học.

Ví dụ như thuật toán phân loại dữ liệu, nó có thể đưa dữ liệu vào các nhóm khác nhau. Thuật toán phân loại được sử dụng để nhận dạng số viết tay, cũng có thể được sử dụng để phân loại email rác và email thường mà không thay đổi một dòng mã nào. Cùng một thuật toán nhưng được đào tạo trên tập dữ liệu khác nhau với mục đích khác nhau.

### Học có giám sát (Supervised Learning):

Supervised **learning** là một hướng tiếp cận của machine learning để làm cho máy tính có khả năng "học". Trong hướng tiếp cận này, người ta "huấn luyện" máy tính dựa trên những **quan sát có dán nhãn**(labeled data). Có thể hình dung những quan sát (observation) này như là những câu hỏi, và nhãn (label) của chúng là những câu trả lời.

Ý tưởng của supervised learning là: bằng việc ghi nhớ và tổng quát hóa một số quy tắc từ một tập câu hỏi có đáp án trước, máy tính sẽ có thể trả lời được những câu hỏi dù chưa từng gặp phải, nhưng có mối liên quan. Ví dụ dạy máy tính "1 + 1 = 2" và hy vọng nó sẽ học được phép tính cộng x + 1 và trả lời được là "2 + 1 = 3". Supervised learning mô phỏng việc con người học bằng cách đưa ra dự đoán của mình cho một câu hỏi, sau đó đối chiếu với đáp án. Sau đó con người rút ra phương pháp để trả lời đúng không chỉ câu hỏi đó, mà cho những câu hỏi có dạng tương tự.

Trong supervised learning, các quan sát **bắt buộc phải được dán nhãn trước**. Đây chính là một trong những nhược điểm của phương pháp này, bởi vì không phải lúc nào việc dán nhãn chính xác cho quan sát cũng dễ dàng. Ví dụ như trong dịch thuật (machine translation), từ một câu của ngôn ngữ gốc có thể dịch thành rất nhiều phiên bản khác nhau trong ngôn ngữ cần dịch sang. Tuy nhiên, việc quan sát được dán nhãn cũng lại chính là ưu điểm của supervised learning bởi vì một khi đã thu thập được một bộ dữ liệu lớn được dán nhãn chuẩn xác, thì việc huấn luyện trở nên dễ dàng hơn rất nhiều so với khi dữ liệu không được dán nhãn

Tóm lại thuật toán này dự đoán đầu ra của một dữ liệu mới dựa trên các cặp (input, outcome) hoặc (data,label) đã biết từ trước. Nghĩa là, một mô hình được chuẩn bị thông qua một quá trình đào tạo để yêu cầu các thuật toán này đưa ra dự đoán. Dự đoán sai, ngay lập tức được thông báo để chỉnh sửa, tiếp tục cho đến khi mô hình đạt đến mức hoàn thiện mong muốn.

Vídụ**:** Điển hình là trong facebook có chức năng nhận diện khuôn mặt sau đó yêu cầu người dùng tag(gắn nhãn) bạn bè  => Cặp dữ liệu (khuôn mặt, tên người) => số lượng cặp dữ liệu càng lớn, độ chính xác ở những lần tự động tag tiếp theo sẽ càng lớn.

Thuật toán Supervised Learning tiếp tục chia nhỏ ra thành 2 loại chính: *Classification* (Phân loại) và *Regression* (Hồi quy).

§ *Classification* (Phân lớp):  Các label của dữ liệu đầu vào được chia thành một số hữu hạn nhóm, giải giá trị của đầu ra là rời rạc.

Ví dụ: Gmail xác định xem một email có phải là spam hay không, các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không.

§  *Regression* (Hồi quy): label không chia thành các nhóm, giải giá trị của đầu ra là liên tục, một giá trị thực cụ thể.

Ví dụ: Một căn nhà x m², có y phòng ngủ và cách trung tâm thành phố z km là có giá là bao nhiêu? x,y,z là một giá trị thực cụ thể.

### Học không giám sát (Unsupervised Learning):

Dữ liệu đầu vào đều không được dán nhãn và không có kết quả rõ ràng. Vì vậy, chuẩn bị mô hình bằng cách suy đoán các cấu trúc có trong dữ liệu đầu vào.

Ví dụ:Dễ thấy nhất chính là gợi ý kết bạn của facebook, để đưa ra 1 gợi ý phù hợp cho người dùng, facebook tập hợp những người bạn quen biết nhau nhiều nhất lại thành một nhóm, từ đó đưa ra gợi ý phù hợp cho người dùng.

*§ Clustering (Phân nhóm):* Một bài toán phân nhóm toàn bộ dữ liệu XX thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm.

Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc ta đưa cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, ví dụ tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

*§ Association:* Là bài toán khi chúng ta muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước.

Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng; những khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

### Học bán giám sát (Semi-supervised Learning):

Trong khoa học máy tính, học bán giám sát là một phương thức của ngành học máy sử dụng cả dữ liệu gán nhãn và chưa gán nhãn, nhiều nghiên cứu của ngành học máy có thể tìm ra được dữ liệu chưa gán nhãn khi sử dụng với một số lượng nhỏ dữ liệu gán nhãn. Công việc thu được kết quả của dữ liệu gán nhãn thường đòi hỏi ở trình độ tư duy và khả năng của con người, công việc này tốn nhiều thời gian và chi phí, do vậy dữ liệu gán nhãn thường rất hiếm và đắt, trong khi dữ liệu chưa gán nhãn thì lại rất phong phú. Trong trường hợp đó, chúng ta có thể sử dụng học bán giám sát để thi hành các công việc ở quy mô lớn.

Học bán giám sát bao gồm dữ liệu gán nhãn và chưa gán nhãn. Học bán giám sát có thể được áp dụng vào việc phân lớp và phân cụm. Mục tiêu của học bán giám sát là huấn luyện tập phân lớp tốt hơn học có giám sát từ dữ liệu gán nhãn và chưa gán nhãn.

Như vậy, có thể nói học bán giám sát là phương pháp học có giám sát kết hợp với việc tận dụng các dữ liệu chưa gán nhãn. Trong phần bổ sung thêm vào cho dữ liệu gán nhãn, thuật toán cung cấp một vài thông tin giám sát, việc này không cần thiết cho tất cả các mẫu huấn luyện. Thông thường thông tin này sẽ được kết hợp với một vài mẫu cho trước.

Học bán giám sát là một nhánh của ngành học máy (machine learning). Các dữ liệu gán nhãn thường hiếm, đắt và rất mất thời gian, đòi hỏi sự nỗ lực của con người, trong khi đó dữ liệu chưa gán nhãn thì vô vàn nhưng để sử dụng vào mục đích cụ thể của chúng ta thì rất khó, vì vậy ý tưởng kết hợp giữa dữ liệu chưa gán nhãn và dữ liệu đã gán nhãn để xây dựng một tập phân lớp tốt hơn là nội dung chính của học bán giám sát. Bởi vậy học bán giám sát là một ý tưởng tốt để giảm bớt công việc của con người và cải thiện độ chính xác lên mức cao hơn.

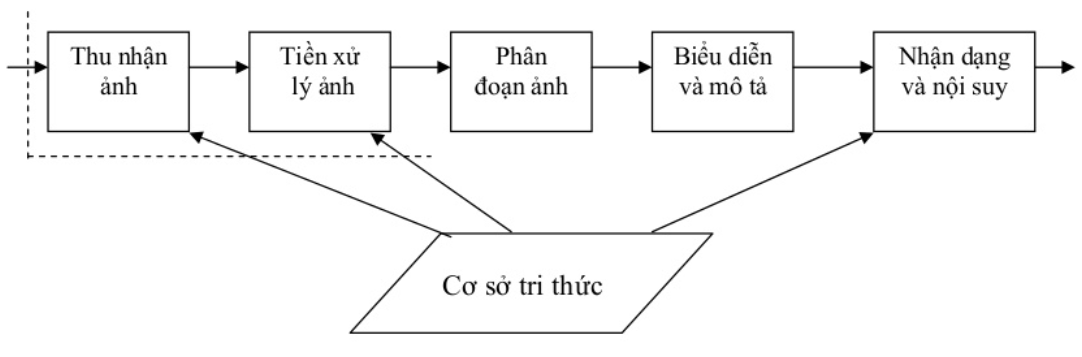
## Các vấn đề cơ bản trong xử lý ảnh

### Xử lý ảnh (XLA)

Con người thu nhận thông tin qua giác quan, trong đó thị giác đóng vai trò quan trọng. Những năm trở lại đây với sự phát triển của phần cứng máy tính, xử lý ảnh và đồ họa phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ họa đóng vai trò quan trọng trong tương tác người máy. [1]

Quá trình xử lý ảnh được xem như quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của quá trình xử lý ảnh có thể là một ảnh “tốt hơn” hoặc một kết luận.

Sơ đồ tổng quát của hệ thống xử lý ảnh:



Hình 2‑1 Các bước cơ bản trong một hệ thống xử lý ảnh

#### Thu nhận ảnh (Image Acquisition)

Ảnh có thể nhận qua camera màu hoặc trắng đen. Thường thì ảnh nhận qua camera là ảnh tương tự (loại camera ống chuẩn CCIR với tần số 1/25, mỗi ảnh 25 dòng), cũng có loại camera đã số hóa (như loại CCD – Change Coupled Device) là loại photodiot tạo cường độ sáng tại mỗi điểm ảnh.

Camera thường dùng là loại quét dòng ảnh tạo có dạng hai chiều. Chất lượng mỗi ảnh có thu nhận được phụ thuộc vào thiết bị thu, môi trường (ánh sáng, thời tiết).

#### Tiền xử lý ảnh (Image processing)

Sau bộ thu nhận, ảnh có thể bị nhiễu, độ tương phản thấp nên cần đưa vào bộ tiền xử lý để nâng cao chất lượng. Chức năng chính của bộ tiền xử lý là lọc nhiễu nâng độ tương phản để làm rõ, nét hơn.

#### Phân đoạn (Segmentation) hay phân vùng ảnh

Phân vùng ảnh là tách một ảnh đầu vào thành các vùng thành phần để biểu diễn phân tích, nhận dạng ảnh. Ví dụ: để nhận dạng chữ (hoặc mã vạch) trên phong bì thư cho mục đích phân loại bưu phẩm, cần chia các câu, chữ, về địa chỉ hoặc tên người thành từ, các chữ, các số (hoặc các vạch) riêng biệt để nhận dạng. Đây là phần phức tạp khó khăn nhất trong xử lý ảnh và cũng dễ gây lỗi, làm mất độ chính xác của ảnh. Kết quả nhận dạng ảnh phụ thuộc rất nhiều vào công đoạn này.

#### Biểu diễn ảnh (Image Representation)

Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của vùng ảnh (ảnh đã phân đoạn) cộng với mã liên kết với các vùng lân cận. Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng (Feature Selection) gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được. Ví dụ: trong nhận dạng ký tự trên phong bì thư, chúng ta miêu tả các đặc trung của từng ký tự giúp phân biệt ký tự này với ký tự khác.

#### Nhận dạng ảnh và nội suy ảnh (Image Recognition and Interpretation)

Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được học (hoặc lưu từ trước). Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng. Ví dụ: một loạt chữ số và nét gạch ngang trên phong bì thư có thể được nội suy thành mã điện thoại. Có nhiều cách phân loại ảnh khác nhau về ảnh. Theo lý thuyết về nhận dạng, các mô hình toán học về ảnh được phân theo hai loại nhận dạng ảnh cơ bản:

* Nhận dạng theo tham số.
* Nhận dạng theo cấu trúc.

Một số đối tượng nhận dạng khá phổ biến hiện nay đang được áp dụng trong khoa học và công nghệ là: nhận dạng ký tự (chữ in, chữ viết tay, chữ ký điện tử), nhận dạng văn bản (Text), nhận dạng vân tay, nhận dạng mã vạch, nhận dạng mặt người,…

Trên đây là các phần cơ bản trong khâu xử lý ảnh. Trong thực tế, các quá trình sử dụng ảnh số không nhất hiết phải qua hết các khâu đó, tùy theo đặc điểm ứng dụng.

### Các vấn đề cơ bản trong xử lý ảnh.

#### Điểm ảnh (Picture Element).

Gốc của ảnh (ảnh tự nhiên) là ảnh liên tục về không gian và độ sáng. Để xử lý bằng máy tính (số), ảnh cần phải được số hóa. Số hóa ảnh là sự biến đổi gần đúng một ảnh liên tục thành một tập các điểm phù hợp với ảnh thật về vị trí (không gian) và độ sáng (mức xám). Khoảng cách giữa các điểm ảnh đó được thiết lập sao cho mắt người không phân biệt được ranh giới giữa chúng. Mỗi một điểm như vậy gọi là điểm ảnh (PEL Picture Element) hay gọi tắt là Pixel. Trong khuôn khổ ảnh hai chiều, mỗi pixel ứng với cặp tọa độ

***Định nghĩa:*** Điểm ảnh (Pixel) là một phần tử của ảnh số tại tọa độ với độ xám hoặc màu nhất định. Kích thước và khoảng cách giữa các điểm ảnh đó được chọn thích hợp sao cho mắt người cảm nhận được sự liên tục về không gian và mức xám (hoặc màu) của ảnh số gần như ảnh thật. Mỗi phần tử trong ma trận được gọi là một phần tử ảnh.

#### Độ phân giải của ảnh

***Định nghĩa:*** Độ phân giải (Resolution) của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiện thị.

Theo định nghĩa, khoảng cách giữa các điểm ảnh phải được chọn sao cho mắt người vẫn thấy được sự liên tục của ảnh. Việc lựa chọn khoảng cách thích hợp tạo nên một mật độ phân bổ, đó chính là độ phân giải và được phân bố theo trục và trong không gian hai chiều.

Ví dụ: Độ phân giải của ảnh trên màn hình CGA (Color Graphic Adaptor) là một lưới điểm theo chiều ngang màn hình: 320 điểm chiều dọc \*200 điểm ảnh (320\*20). Rõ ràng, cùng màn hình CGA 12” ta nhận thấy mịn hơn màn hình CGA 17” độ phân giải 320\*200. Lý do: cùng một mật độ (độ phân giải) nhưng diện tích màn hình rộng hơn thì độ mịn (liên tục của các điểm) kém hơn.

#### Mức xám của ảnh

Một điểm ảnh (Pixel) có hai đặc trưng cơ bản là vị trí của điểm ảnh và độ xám của nó.

***Định nghĩa:*** Mức xám của điểm ảnh là cường độ xám của nó gắn bằng giá trị số tại điểm đó.

Các thang giá trị mức xám thông thường: 16, 32, 64, 128, 256 (Mức 256 là mức phổ dụng. Lý do: từ kỹ thuật máy dùng 1 byte (8 bit) để biểu diễn mức xám: Mức xám dùng 1 byte biểu diễn: 28=256 mức, tức là 0 đến 255).

*Ảnh đen trắng:* là ảnh có hai màu đen, trắng (không chứa màu khác) với mức xám ở các điểm ảnh có thể khác nhau.

*Ảnh màu:* trong khuôn khổ lý thuyết ba màu (Red, Blue, Green) để tạo nên thế giới màu, người ta thường dùng 3 byte để mô tả mức màu, khi đó các giá trị màu: triệu màu.

#### Ảnh số

Ảnh số là tập hợp các điểm ảnh với mức xám phù hợp dùng để mô tả ảnh gần với ảnh thật.

## Tổng quan về OpenCV

### Giới thiệu về OpenCV

OpenCV (Open Source Computer Vision Library) [2]: là một thư viện mã nguồn mở về thị giác máy tính và học máy. Thư viện được xây dựng để cung cấp nền tảng cho các ứng dụng thị giác máy tính nhằm đẩy mạnh sự phát triển về hàm lượng tri thức máy tính trong các sản phẩm thương mại. Nhờ giấy phép bản quyền BSD và được nhiều công ty lớn hàng đầu thế giới như Google, Yahoo, Microsoft, Intel, IBM,… cùng đóng góp xây dựng thư viện, OpenCV là một trong những công cụ mạnh và được sử dụng rộng rãi trong trường học cũng như các công ty khởi nghiệp.

Thư viện OpenCV bao gồm nhiều giao diện dành cho C++, C, Python, Java, MATLAB và hỗ trợ cho các hệ điều hành khác nhau như Windows, Linux, Android, MacOS. Trong phiên bản OpenCV 3.1, giao diện sử dụng cho CUDA và OpenCL cũng đã được phát triển hoàn thiện. OpenCV được viết nguyên bản bằng ngôn ngữ C++.

Trang chính thức: <http://opencv.org>

Nhóm phát triển OpenCV: <http://itseez.com>

### Sơ lược về cấu trúc thư viện OpenCV

Thư viện OpenCV có thể được chia thành 2 phần (module) chính. Phần căn bản (basic) là mã nguồn được nhóm phát triển xây dựng và kiểm định toàn diện, gồm các thuật toán đã được thế giới công nhận và đánh giá dựa trên cơ sở lý thuyết chắc chắn. Phần mở rộng (contribution) được nhiều tổ chức khoa học khác nhau trên thế giới đóng góp, gồm nhiều thuật toán cập nhật được xây dựng dựa trên các công trình nghiên cứu, bài báo mới đăng trong thời gian gần đây. Do vậy các thuật toán trong phần mở rộng có độ ổn định và tối ưu không cao. Từ phiên bản 3.0, phần mở rộng được tách riêng không còn được gộp chung với thư viện mặc định.

OpenCV là thư viện mã nguồn mở được đóng gói thành tập tin nén. Tùy vào phiên bản dành cho các hệ điều hành khác nhau mà tập tin nén này có định dạng tương ứng. Thư viện OpenCV cung cấp cho người dùng các cấu trúc dữ liệu, đối tượng và hàm bằng cách khai báo nguyên mẫu (prototype) của chúng trong các tập tin thư viện C/C++ (\*.h, \*.hpp,…) và định nghĩa chi tiết trong các tập tin mã nguồn (\*.c, \*.cpp). Với mức độ sử dụng OpenCV, ta chỉ cần giải nén các tập tin đã được biên dịch sẵn rồi thực hiện các thao tác cài đặt đường dẫn cho thích hợp để hệ điều hành tìm đến đúng vị trí của các tập tin thư viện. Ở mức độ cao hơn, nếu muốn hiệu chỉnh sửa đổi thuật toán hay sử dụng phần mở rộng của OpenCV ta cần phải biên dịch mã nguồn trực tiếp trên máy trước khi cài đặt.

Để cài đặt OpenCV cơ bản ta có thể sử dụng tập tin nhị phân mà nhóm phát triển đã biên dịch sẵn. Ta truy cập vào địa chỉ http://opencv.org/downloads.html và chọn phiên bản dành cho hệ điều hành thích hợp. Để cài đặt phần mở rộng (dành cho OpenCV 3.0 trở lên) ta cần có cả mã nguồn của cả phần cơ bản và phần mở rộng. Chúng được công khai trên tài khoản Github của nhóm phát triển theo đường dẫn http://github.com/opencv. Ngoài ra ta cũng có thể đóng góp cho thư viện này bằng cách sử dụng tài khoản cá nhân trên Github và làm theo hướng dẫn của nhóm phát triển.

Thư viện mã nguồn mở OpenCV gồm các thành phần nhỏ sau đây (tính đến phiên bản 3.1.0):

Các thành phần chính của thư viện OpenCV gồm:

Bảng 2‑1 Các thành phần chính của thư viện OpenCV

|  |  |
| --- | --- |
| Core | Các hàm cơ bản |
| Imgproc | Xử lý ảnh |
| Imgcodecs | Đọc và ghi tập tin ảnh |
| Videoio | Đọc và ghi đa phương tiện |
| Highgui | Giao diện người dùng bậc cao |
| Video | Phân tích video |
| calib3d | Hiệu chuẩn thiết bị ghi hình và tái cấu trúc 3D |
| features2d | Bộ khung các đặc trưng 2D |
| Objdetect | Nhận dạng đối tượng |
| Ml | Học máy |
| Flann | Gom nhóm và tìm kiếm trong nhiều chiều |
| Photo | Các thuật toán chụp ảnh |
| Stitching | Vá ảnh |
| Cudaarithm | Các toán tử trên ma trận |
| Cudabgsegm | Phân đoạn nền ảnh |
| Cudacodec | Mã hóa/giải mã video |
| cudafeatures2d | Phát hiện và mô tả đặc trưng |
| Cudafilters | Lọc ảnh |
| Cudaimgproc | Xử lý ảnh với CUDA |
| Cudalegacy | Hỗ trợ Legacy |
| Cudaobjdetect | Phát hiện đối tượng |
| Cudaoptflow | Dòng quang học |
| Cudastereo | Thư viện âm thanh |
| Cudawarping | Bẻ cong ảnh |
| Cudev | Tầng thiết bị |
| Shape | Khác biệt hình học và so sánh |
| Superres | Siêu phân giải |
| Videostab | Ổn định video |
| Viz | Hiển thị 3D |

Các thành phần mở rộng của thư viện OpenCV gồm:

Bảng 2‑2 Các thành phần mở rộng của thư viện OpenCV

|  |  |
| --- | --- |
| Aruco | Phát hiện ArUco Marker |
| Bgsegm | Các phương pháp phân đoạn đối tượng mới 6 |
| Bioinspired | Các mô hình thị giác sinh học và các công cụ phái sinh |
| Ccalib | Tùy chọn hiệu chuẩn mẫu cho tái cấu trúc 3D |
| Cvv | GUI cho thao tác bắt lỗi trực quan của các chương trình thị giác máy tính |
| Datasets | Thư viện hỗ trợ làm việc với các bộ dữ liệu khác nhau |
| Dnn | Bộ thư viện mạng neural học sâu |
| Dpm | Các mô hình biến dạng từng phần |
| Face | Nhận dạng mặt người |
| Fuzzy | Xử lý ảnh dựa vào lý thuyết mờ |
| Hdf | Nhập xuất dữ liệu cho định dạng HDF (Hierarchical Data Format) |
| line\_descriptor | Mô tả nhị phân cho các đường thẳng trích chọn từ một ảnh |
| Matlab | Liên kết với MATLAB |
| Optflow | Các thuật toán dòng quan học |
| Plot | Hàm vẽ cho dữ liệu ma trận |
| Reg | Đăng kí ảnh |
| Rgbd | Xử lý chiều sâu trong hệ màu RGB |
| Saliency | API xử lý vùng lồi trên ảnh |
| Sfm | Cấu trúc từ chuyển động |
| Stereo | Các thuật toán tín hiệu nổi tương tự |
| structured\_light | API cho tính cấu trúc của ánh sáng |
| surface\_matching | Khớp bề mặt |
| Text | Phát hiện và nhận dạng chữ trong cảnh tự nhiên |
| Tracking | Các phương pháp theo vết |
| xfeatures2d | Tính năng đặc trưng 2D mở rộng |
| Ximgproc | Xử lý ảnh mở rộng |
| Xobjdetect | Phát hiện đối tượng mở rộng |
| Xphoto | Các phương pháp xử lý hình ảnh nâng cao |

## Tổng quan về dlib

### Giới thiệu về Dlib

Dlib là bộ công cụ C ++ hiện đại chứa các thuật toán và công cụ học máy để tạo phần mềm phức tạp trong C ++ để giải quyết các vấn đề trong thế giới thực. Nó được sử dụng trong cả ngành công nghiệp và học thuật trong nhiều lĩnh vực bao gồm robot, thiết bị nhúng, điện thoại di động và môi trường điện toán hiệu năng cao lớn. [Cấp phép nguồn mở](http://dlib.net/license.html) của Dlibcho phép sử dụng nó trong bất kỳ ứng dụng nào, miễn phí. [3]

Thư viện Dlib được viết bằng ngôn ngữ C++ do **Davis King** tạo ra vào năm 2002. Được sử dụng nhiều trong lĩnh vực computer vision đặc biệt là nhận dạng object và face.

### Cách sử dụng dlib từ Python

Hoặc chạy: pip install dlib --verbose hoặc lấy các nguồn mới nhất từ github, đi vào thư mục gốc của kho dlib, và chạy python setup.py install . Khi một trong hai lệnh này kết thúc chạy, đã sẵn sàng sử dụng dlib từ Python. Lưu ý rằng cần cài đặt CMake và trình biên dịch C ++ để hoạt động.

Cũng lưu ý rằng các tính năng tùy chọn khác nhau như hỗ trợ GUI (ví dụ: dlib.image\_window) và khả năng tăng tốc CUDA sẽ tự động được bật hoặc tắt dựa trên những gì có sẵn trên máy tính. Khi chạy lệnh cài đặt, nó sẽ in các thông báo cho biết nó đang sử dụng cái gì. Đọc những tin nhắn đó và có hành động phù hợp nếu không thích kết quả. Ví dụ: người dùng Linux và OSX có thể phải cài đặt libX11 để sử dụng các công cụ GUI. Nếu quan tâm đến điều này thì hãy đọc các tin nhắn vì chúng cho biết cách cài đặt các tính năng tùy chọn này.

Ngoài ra, nếu muốn thêm nhiều ràng buộc python vào giao diện python của dlib thì có lẽ muốn tránh tệp setup.py và làm việc trực tiếp bằng CMake. Cụ thể, API python của dlib được xây dựng bởi dự án CMake trong thư mục tools / python. Xây dựng dự án này bằng các lệnh CMake thông thường và khi được biên dịch, nó xuất ra thư viện chia sẻ dlib xác định API python cho dlib.

## Các vấn đề liên quan trong việc xử lý khuôn mặt người

### Khoảng cách Euclidean

Khoảng cách Euclidean (Euclidean Distance) [4]

Khoảng cách giữa hai điểm là chiều dài của đường thẳng nối chúng. Trong mặt phẳng, khoảng cách giữa hai điểm (­x1, y1­)  và ( ­x2­­­, y2) được định lý Pythagorean như sau:

Trong không gian ba chiều Euclidean, khoảng cách giữa hai điểm (­x1, y1­, z1­­­) và (­x2, y2, z2­­­) là:

Một cách tổng quát, khoảng cách giữa hai điểm x, y trong không gian Euclidean R với n chiều được tính như sau:

### Xác định khuôn mặt (Face detection)

Xác định khuôn mặt người (Face detection) [5]: Là một kỹ thuật trong máy tính dùng để xác định các kích cỡ, vị trí, diện mạo của khuôn mặt trong các ảnh bất kỳ (ảnh kỹ thuật số). Vì kỹ thuật này nhận biết các đặc trưng trên của khuôn mặt nên nó sẽ bỏ qua các vật thể khác như: cơ thể, tòa nhà, cây cối,…

Face detection có nhiều ứng dụng trong thực tế: tương tác giữa người và máy tính, trong các hệ thống giám sát an ninh, phân tích cảm xúc

Cũng giống như **Opencv**, Dlib cũng hỗ trợ nhận dạng khuôn mặt. Dlib sử dụng HOG (Histogram of Oriented Gradients) và SVM (Support Vector Machine) để tiến hành phát hiện khuôn mặt.

### Phương pháp trích chọn đặc trưng HOG

Histogram of Oriented Gradients (HOG): Thể hiện sự thay đổi về sự sáng tối trong bức ảnh. Tại các nét trên khuôn mặt có sự thay đổi sáng tối rõ rệt. HOG thể hiện được đường nét của khuôn mặt.

Về cơ bản, một vector đặc trưng HOG được thành lập từ việc tính các histogram về các hướng của đạo hàm của một ảnh tại các vùng cục bộ gọi là “tế bào” (cell). Cơ sở cho phương pháp này là thông tin của ảnh có thể được biểu diễn bằng cách sử dụng sự phân bố của các giá trị đạo hàm của ảnh hoặc hướng của các đạo hàm cục bộ tại các điểm ảnh. Ảnh được chia thành nhiều tế bào là các vùng con để tính các histogram của các hướng đạo hàm. Biểu diễn histogram của các tế bào khi hợp lại sẽ tạo thành biểu diễn HOG cho ảnh ban đầu. Chuỗi histogram có thể được chuẩn hóa để tăng hiệu quả nhận dạng vì chúng có tính bất biến cao hơn đối với các thay đổi về ánh sáng.

Để tính một vector HOG từ một ảnh đầu vào chúng ta cần thực hiện qua 4 bước như mô tả sau đây.

Bước 1: tính đạo hàm của ảnh. Bước này được thực hiện bằng cách nhân chập ảnh đầu vào với hai nhân 1 chiều tương ứng cho việc lấy đạo hàm theo hai hướng Ox và Oy, cụ thể giá trị của hai nhân là: Dx = [-1 0 1] và Dy = [1 0 -1]T, (1) Trong đó T là ký hiệu của phép toán lấy ma trận chuyển vị. Với một ảnh đầu vào I sẽ có 2 đạo hàm được tính là Ix = I \* Gx, Iy = I \* Gy. Tiếp đến hai thành phần cường độ (magnitude) và hướng (orientation) sẽ được tính theo các công thức: |G| = sqrt(Ix 2 + Iy 2 ), T = arctan(Iy, Ix) (2)

Bước 2: gán hướng. Sau khi tính xong hướng của các đạo hàm tại các điểm ảnh, ta sẽ nhóm các giá trị hướng khác nhau trong mỗi vùng của khoảng [0o , 360o ] thành một nhãn duy nhất đại diện cho nhóm đó. Cụ thể sẽ có 9 nhãn từ 0 tới 8 tương ứng với các vùng giá trị của các hướng từ [0o , 360o /9), [360o /9, 2\*360o /9),… Sau đó histogram của các cell sẽ được tính dựa trên số lần xuất hiện của các nhãn được gán.

Bước 3: Tính histogram của các khối. Các cell nhỏ sẽ được ghép với nhau tạo thành các khối (block) không tách rời nhau (các cell sẽ xuất hiện nhiều hơn 1 lần trong các khối). Vector HOG được tạo thành từ histogram đã được chuẩn hóa của các khối.

Bước 4: chuẩn hóa khối. Ở bước này các vector HOG sẽ được chuẩn hóa bằng cách sử dụng các hàm chuẩn hóa thông dụng như L1-norm, L2-norm để có thể đáp ứng tốt hơn trong các điều kiện ánh sáng thay đổi.

### SVM – Support Vector Machine

SVM sử dụng thuật toán học nhằm xây dựng một siêu phẳng làm cực tiểu hoá độ phân lớp sai của một đối tượng dữ liệu mới. Độ phân lớp sai của một siêu phẳng được đặc trưng bởi khoảng cách bé nhất tới siêu phẳng đấy. SVM có khả năng rất lớn cho các ứng dụng được thành công trong bài toán phân cụm hình ảnh.

Tập phân cụm SVM là mặt siêu phẳng phân tách các mẫu dương khỏi các mẫu âm với độ chênh lệch cực đại, trong đó độ chênh lệch – còn gọi là Lề (margin) xác định bằng khoảng cách giữa các mẫu dương và các mẫu âm gần mặt siêu phẳng nhất. Mặt siêu phẳng này được gọi là mặt siêu phẳng lề tối ưu.

SVM xác định khuôn mặt bằng cách quét toàn bộ hình ảnh cho các mẫu giống như khuôn mặt ở nhiều tỷ lệ khác nhau. Trước khi lưu trữ hình ảnh, một số bước xử lý trước như cân bằng sáng, loại bỏ đối tượng nhiễu (tập trung vào đối tượng quan trọng) được thực hiện. Sau đó ta chọn một nhóm các hình ảnh không phải mặt người để tạo ra bộ phân loại sai (không phải mặt người). SVM sẽ dựa vào bộ dữ liệu đúng (có khuôn mặt) và bộ dữ liệu sai (không có khuôn mặt) để tạo ra một vector ngăn cách 2 bộ dữ liệu này. Với mỗi hình ảnh, SVM sẽ xác định ảnh đó thuộc về phân lớp nào (có khuôn mặt hay không có khuôn mặt).

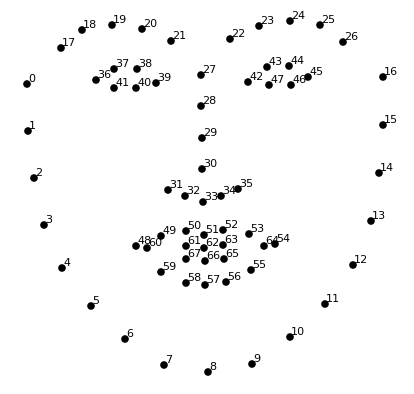
### Các dặc trưng về khuôn mặt (Facial landmarks)

Facial landmarks được dùng để khoanh vùng và thể hiện các vùng nhô ra, nổi bật của khuôn mặt, như là: Mắt, lông mày, mũi, miệng, đường viền hàm.

Facial landmarks được áp dụng thành công để căn chỉnh khuôn mặt, ước tính tư thế đầu, hoán đổi khuôn mặt, phát hiện chớp mắt và nhiều hơn thế nữa [5]

Bộ xác định facial landmark của dlib sẽ xác định 68 điểm chính theo tọa độ (x,y) cấu tạo nên khuôn mặt người như hình bên dưới:

* Mouth point = 48-67
* Right brow point = 17-21
* Left brow point = 22-26
* Right eye point = 36-41
* Left eye point = 42-47
* Nose point = 27-35
* Jaw point = 0-16



Hình 2‑2 Hình ảnh 68 điểm đặc trưng của khuôn mặt

68 mốc sẽ xác định vị trí trên mọi khuôn mặt. Hình ảnh này được tạo bởi [Brandon Amos](http://bamos.github.io/) của CMU, người làm việc trên [OpenFace](https://github.com/cmusatyalab/openface) .

Để có thể xác định được 68 điểm này trên khuôn mặt người, bộ xác định facial landmark của dlib được huấn luyện với bộ dữ liệu [iBUG 300-W](https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/)

Xác định được càng nhiều điểm landmark thì khuôn mặt người càng cụ thể, rõ ràng và chính xác hơn. Ví dụ FaceID đã sử dụng bộ cảm biến hồng ngoại với 30 triệu điểm để dựng lên cấu trúc khuôn mặt 3D của người sử dụng điện thoại.

### Định nghĩa bài toán xác định khuôn mặt

Hiện nay có rất nhiều kỹ thuật để thực hiện việc nhận dạng khuôn mặt, tuy nhiên điểm chung của các kỹ thuật này là đều sẽ phải thực hiện qua các bước sau:

* Xác định và lấy ra (các) khuôn mặt có trong hình ảnh
* Tiền xử lý, căn chỉnh khuôn mặt
* Từ hình ảnh các khuôn mặt lấy ra từ bước trên, thực hiện việc phân tích, trích xuất các đặt trưng của khuôn mặt
* Từ các thông tin có được sau khi phân tích, kết luận và xác minh danh tính người dùng

Hình 2‑3 Hệ thống nhận dạng mặt người

#### Find all face

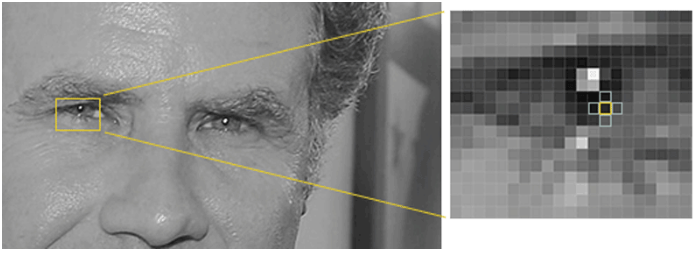
Đầu tiên trong pipeline là face detection, để tìm kiếm khuôn mặt trong bức hình

Làm cho bức hình thành đen trắng



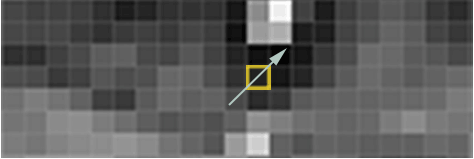
Hình 2‑4 Hình ảnh đưa về hình ảnh đen trắng

Đối với mỗi pixel đơn, xem xét các pixel xung quanh nó



Hình 2‑5 Xét độ đậm nhạt của pixel với các pixel xung quanh

Mục tiêu là để tìm ra độ đậm của điểm ảnh hiện tại so với các điểm ảnh xung quanh



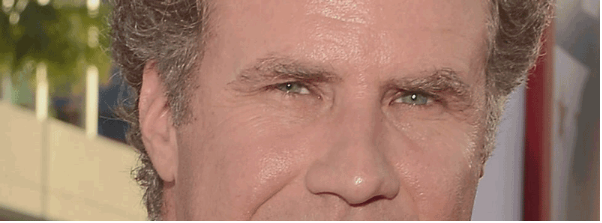
Hình 2‑6 Lặp lại quá trình xét độ đậm nhạt cho mỗi pixel đơn trong hình

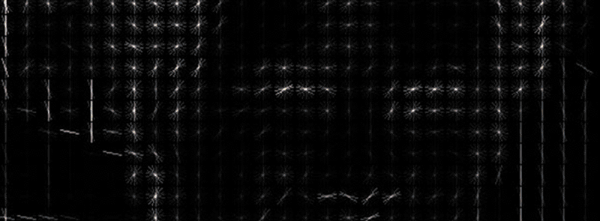
Lặp lại quá trình cho mỗi pixel đơn trong hình ảnh, và kết thúc khi toàn bộ các pixel được thay thế bởi một mũi tên. Những mũi tên này được gọi là gradients và chúng cho thấy flow từ các điểm sáng tới vùng bóng tối trên toàn bộ hình ảnh.

Nếu phân tích pixel trực tiếp, hình ảnh tối và hình ảnh sáng của cùng một người sẽ có các giá trị pixel hoàn toàn khác nhau. Nhưng nếu chỉ xem xét hướng thay đổi độ sáng tối. Thì như vậy với một bức ảnh thuần tối, hoặc thuần sáng sẽ kết thúc với cùng một mô tả vector. Do đó vấn đề trên đã có thể dễ dàng giải quyết hơn nhiều.

Thực hiện việc này bằng cách chia nhỏ hình ảnh thành những ô vuông nhỏ 16x16. Trong mỗi ô vuông, đếm số lượng điểm gradient trong mỗi hướng chính (bao nhiêu điểm lên, điểm lên-phải, điểm sang phải, vv ...). Sau đó, thay thế ô vuông đó trong hình ảnh bằng hướng mũi tên mạnh nhất.

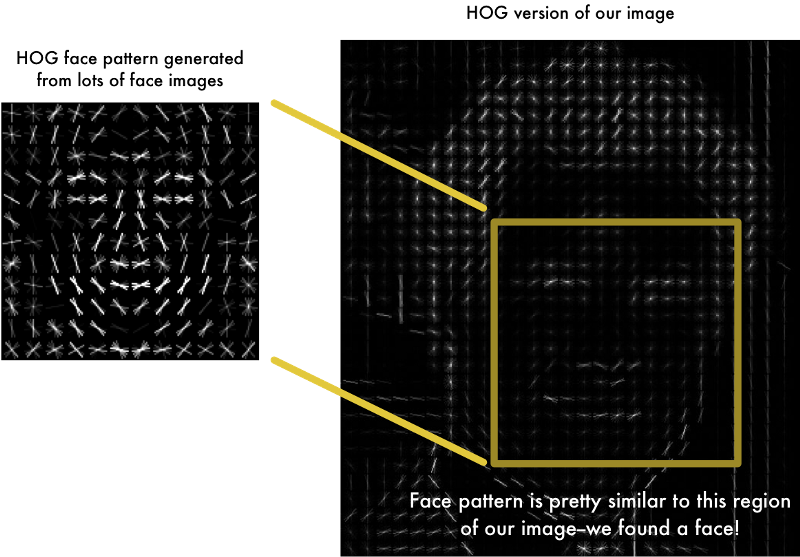
Kết quả cuối cùng là biến hình ảnh ban đầu thành một biểu diễn rất đơn giản để nắm bắt cấu trúc cơ bản của khuôn mặt một cách đơn giản:





Hình 2‑7 Hình ảnh biểu diễn đơn giản cấu trúc cơ bản của khuôn mặt

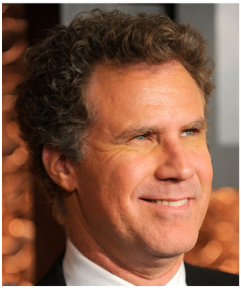
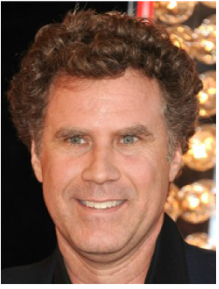
Để tìm khuôn mặt trong hình ảnh này, tất cả những gì phải làm là tìm một phần hình ảnh trông giống nhất với các hình ảnh mẫu đã biết được trích ra từ một loạt các khuôn mặt đã được training khác:



Hình 2‑8 Sự thay đổi về sự sáng tối trong bức ảnh

#### Posing and Projecting Faces

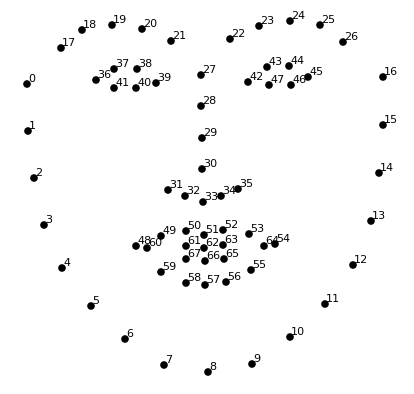
Con người có thể dễ dàng nhận ra rằng cả hai hình ảnh đều là của Will Ferrell, nhưng máy tính sẽ thấy những bức ảnh này là hai người hoàn toàn khác nhau.



Hình 2‑9 Hai hình ảnh thằng mặt và nghiêng mặt của Will Ferrell

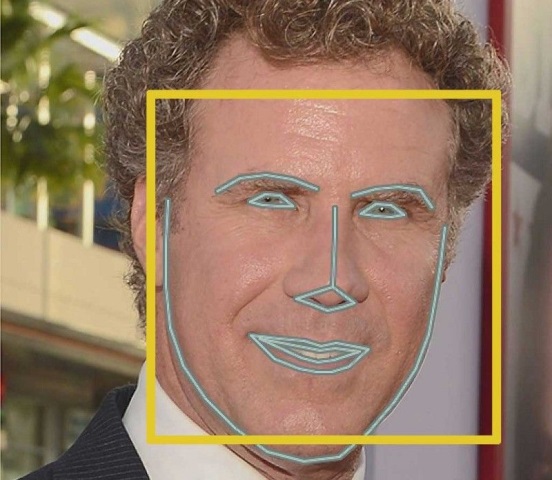
Thực tế, có những khuôn mặt thì nghiêng sang trái, nghiêng sang phải, chỉ nhìn thấy một phần khuôn mặt, hay có những bức hình thì hơi nghiêng đầu, khiến cho bức ảnh bị lệch khỏi khung nhìn. Do đó, cần phải cố gắng làm cong bức hình để đôi mắt, miệng luôn ở vị trí mẫu trong ảnh. Điều này sẽ làm cho việc so sánh khuôn mặt ở các bước tiếp theo dễ dàng hơn nhiều.

Để làm điều này, sử dụng một thuật toán được gọi là **face landmark estimation**. Ý tưởng cơ bản của thuật toán là tìm ra 68 điểm cụ thể (được gọi là mốc) tồn tại trên mỗi khuôn mặt - đầu cằm, cạnh bên ngoài của mỗi mắt, cạnh bên trong của mỗi lông mày ... Sau đó, chúng ta sẽ dùng một thuật toán machine learning để training để có thể tìm thấy 68 điểm cụ thể trên từng khuôn mặt.



Hình 2‑1 Hình ảnh các điểm đặc trưng trên khuôn mặt (facial landmarks)

Kết quả sau khi tìm ra 68 điểm đặc trưng trên khuôn mặt của bức ảnh test:



Hình 2‑2 Hình ảnh đã phân biệt được các đặc điểm mắt, mũi, miệng

Đã phân biệt được các đặc điểm mắt, miệng, như vậy chỉ cần xoay và cắt ảnh để mắt và miệng được căn giữa là xong. Ở bước này chỉ cần sử dụng các phép biến đổi hình ảnh cơ bản như xoay, scale (affine trainformations) là đủ.



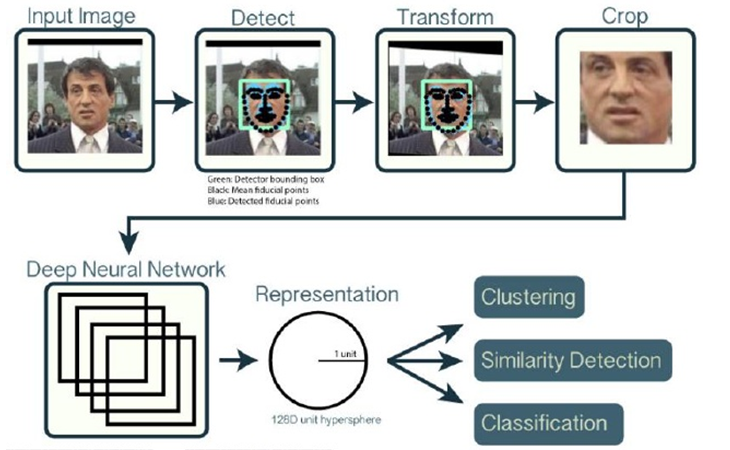
Hình 2‑3 Hình ảnh căn chỉnh thẳng của khuôn mặt

Có thể thấy, mắt và miệng trong bức hình gần như đã được xoay về vị trí trung tâm. việc này sẽ giúp cho các bước xử lý tiếp theo đạt kết quả chinh xác cao hơn nhiều.

### Tìm kiếm và nhận diện khuôn mặt dựa trên các đặc trưng

Từ hình ảnh các khuôn mặt lấy ra được ở trên, thực hiện phân tích, trích xuất các khoảng cách đặc trưng của khuôn mặt*.*

Từ thông tin phân tích, kết luận và xác định khuôn mặt



Hình 2‑4 Tóm tắt quy trình nhận dạng khuôn mặt

## Cây S-Tree (Signature Tree)

S-tree là cây nhiều nhánh cân bằng [1, 4], mỗi một nút của cây S-Tree gồm nhiều cặp phần tử <s, p>, với s là một chữ ký nhị phân, p là con trỏ tham chiếu đến nút con. Nút gốc của cây chứa ít nhất là hai cặp phần tử và nhiều nhất là M cặp phần tử <s, p>. Mỗi nút trong của cây chứa ít nhất là m cặp phần tử <s, p> và nhiều nhất là M cặp phần tử <s, p>, với điều kiện . Mỗi một nút lá của cây S-tree chứa tập các phần tử <s, oid>, với oid là định danh của đối tượng, s là chữ ký của đối tượng tương ứng. Mỗi chữ ký tại một nút cha là tổ hợp tất cả các chữ ký của nút con. Chiều cao tối đa của cây S-tree có n chữ ký sẽ là .

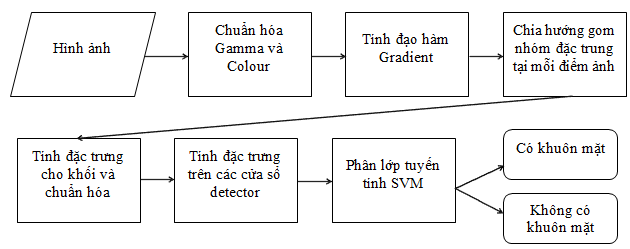
Với mỗi chữ ký truy vấn sẽ được thực hiện truy vấn top-down trên cây S-tree và có thể duyệt qua nhiều đường đi vì chữ ký truy vấn có thể phù hợp với nhiều chữ ký tại một nút trong cây S-tree. Quá trình xây dựng cây S-tree được thực hiện dựa trên thao tác chèn và tách nút. Tại thời điểm bắt đầu, cây S-tree chỉ chứa một nút lá rỗng, sau đó từng chữ ký trong tập tin sẽ được chèn vào trong cây S-tree. Khi nút lá v trở nên đầy sẽ được tách thành hai nút, đồng thời nút cha v sẽ được tạo ra (nếu chưa tồn tại) và hai chữ ký mới sẽ được đặt vào trong nút cha v.

# THUẬT TOÁN VÀ MÔ HÌNH TÌM KIẾM ẢNH

## Tạo chữ ký của hình ảnh (trích xuất vector đặc trưng)

Chữ ký của hình ảnh là sự mô tả ngắn gọn về nội dung của hình ảnh. Phương pháp chữ ký sẽ tiết kiệm trên 75% bộ nhớ. Mỗi hình ảnh trong cơ sở dữ liệu ảnh sẽ được trích xuất ra thành một vector đặc trưng của hình ảnh nhằm giảm kích thước của ảnh.

**Tạo lớp training dữ liệu**



Hình 3‑1 Phương pháp phát hiện khuôn mặt

Đầu vào: File các hình ảnh mặt người

Đầu ra: File text các vector đặc trưng khuôn mặt

B1: Load hình ảnh

Sử dụng thư viên OpenCV để đọc hình ảnh trong file

img = cv2.imread(image)

B2: Đưa hình ảnh thành hình ảnh đen trắng

Chuyển các hình ảnh đã load thành hình ảnh đen trắng với hàm của OpenCV cv2.cvtColor(input\_image, flag)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

B3: Face detection với dlib

Dữ liệu đầu vào là những image có object muốn detection và đầu ra là những bouding box sẽ có dạng (x, y, w, h). Trong đó x, y là tọa độ leftop của bounding box, (w,h) là width và height.

Để detectoion face trong dlib trước tiên ta cần tạo một object:

detector = dlib.get\_frontal\_face\_detector().

detector = dlib.get\_frontal\_face\_detector()

    predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")

Object detector có 2 tham số là chúng ta quan tâm

Một là image chúng ta muốn detect

Hai là tham số upsample . Là kiểu int khi mà chúng ta muốn detection các object nhỏ hơn. Thì tham số này có tác dụng phóng to ảnh.

Sau đó detection các hình ảnh trắng đen đã chuyển ở bước trên

Sau khi detection là 1 list các face detection. Mỗi face detection là 1 retangle tuple theo cấu trúc (left, top), (right, bottom)

    rects = detector(gray, 0)

B4: Facial landmark với dlib, OpenCV và Python

Ở bước này sử dụng một số chức năng tiên lợi của thư viện imutils, đặc biệt trong đó là [inside face\_utils.py](https://github.com/jrosebr1/imutils/tree/master/imutils/face_utils).

Từ bước Face detection ở trên, bộ phát hiện dlib (tức bộ phát hiện khuôn mặt) xác định được đường giới hạn khuôn mặt hình hộp chữ nhật (bouding box)

Tạo đường dẫn đến file training trích xuất điểm đặc trưng trên khuôn mặt được đào tạo trước của dlib

ap = argparse.ArgumentParser()

ap.add\_argument("-p", "--shape-predictor", required=True,

help="path to facial landmark predictor")

Tạo đường dẫn đến tập ảnh đầu vào muốn phát hiện các đặc trưng trên khuôn mặt.

ap.add\_argument("-f", "--folder-train", required=True,

help="path folder training")

Tiếp theo là  khởi tạo trình phát hiện khuôn mặt được đào tạo trước của dlib dựa trên việc sửa đổi [Histogram of Oriented Gradients + Linear SVM method](https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/) để phát hiện đối tượng.

Sau đó tải các yếu tố dự báo mang tính bước ngoặt trên khuôn mặt bằng cách sử dụng đường dẫn đến cung cấp shape\_predictor

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(args["shape\_predictor"])

Face landmark detector của dlib sẽ trả về một đối tượng rect có chứa các tọa độ (x, y) của 68 điểm mốc mặt của cấu trúc trên lý thuyết đã nhắc

crop\_img = cv2.resize(crop\_img, (250, int(scaleH)), interpolation = cv2.INTER\_AREA)

Xử lý trước hình ảnh bằng cách thay đổi kích thước hình ảnh

shape\_crop = predictor(gray\_crop, rect\_crop)

                shape\_crop = face\_utils.shape\_to\_np(shape\_crop)

Đối với mỗi lần phát hiện khuôn mặt, áp dụng facial landmark detection, cho tọa độ (x, y) của 68 điểm mốc mặt

Sau đó chuyển đổi đối tượng shape dlib thành mảng NumPy với shape(68, 2)

B5: Tính khoảng cách các vector đặc trưng khuôn mặt

Từ các tọa độ(x, y) của 68 điểm mốc mặt ở bước trên, tính các vector đặc trưng khuôn mặt dựa trên khoảng cách euclide của các điểm mốc mặt

Gồm 42 vector đặc trưng được lưu vào file text

Đầu ra: file text lưu 42 vector đặc trưng khuôn mặt

## Tạo cây S-Tree

Nhằm giảm không gian và tăng tốc độ truy vấn, đề tài nghiên cứu tiến hành xây dựng cây chữ ký S-tree lưu trữ các chữ ký nhị phân của hình ảnh.

### Cấu trúc cây

#### Class STree: cấu trúc của cây.

* Các thành phần:
* m: số lượng phần tử tối đa sẽ tách nút.
* countId: được sử dụng để đặt id cho các nút. (id sẽ được đặt theo thứ tự được sinh ra khi tạo cây)
* root: nút gốc của cây.
* lstLeaf: danh sách các nút lá trong cây.
* lstNode: danh sách các nút không phải là nút lá trong cây.
* Các phương thức:
* addRoot: thêm nút root.
* addImage: thêm 1 hình ảnh vào cây.
* cutRoot: cắt nút gốc.
* cutNode: cắt nút không phải nút lá trong cây.
* cutLeaf: cắt nút lá trong cây.

#### Class Image: lưu trữ cấu trúc của 1 hình ảnh.

* Các thành phần:
* id: id của hình ảnh. (tên file ảnh)
* vector: chứa vector đặc trưng của ảnh.
* Các phương thức:
* getId: lấy id của ảnh.
* getVector: lấy vector của ảnh.

#### Class Vector: lưu trữ cấu trúc của 1 vector được sinh ra khi tách nút.

* Các thành phần:
* link: id của node sẽ link tới trong cây.
* vector: vector trung bình của node sẽ link tới trong cây.
* Các phương thức:
* getLink: lấy link của vector.
* getVector: lấy vector.
* setVector: gán giá trị cho vector

#### Class Leaf: cấu trúc của một nút là nút lá trong cây.

* Các thành phần:
* id: id của node trong cây.
* n: số lượng phần tử của nút.
* lstImage: danh sách các hình ảnh trong nút. (Ở đây là danh sách các đối tượng có kiểu dữ liệu là Image được định nghĩa ở trên)
* Các phương thức:
* add: thêm 1 hình ảnh vào nút.

#### Class Node: cấu trúc của 1 nút không phải nút lá trong cây.

* Các thành phần:
* id: id của node trong cây.
* n: số lượng phần tử của nút.
* lstVector: danh sách các vector link trong cây (Ở đây là danh sách các đối tượng có kiểu dữ liệu là Vector được định nghĩa ở trên)
* Các phương thức:
* add: thêm 1 vector vào nút.

### Thuật toán tạo cây S-Tree

* B1: Tạo nút root của cây. Thêm hình ảnh đầu tiên vào cây.
* B2: Cho sức chứa của 1 nút trong cây (ở đây sử dụng 20: quá 20 hình ảnh trong 1 nút sẽ tiến hành tách nút).
* B3: Thêm lần lượt các hình tiếp theo vào cây. Khi vượt quá 20 sẽ tách nút gốc.
* Cách tách: chọn 2 vector xa nhau nhất tạo thành 2 nút (ở đây là nút lá). Đưa các hình ảnh trong nút gốc còn lại vào 2 nút vừa tạo theo quy tắc gần nút nào sẽ đưa vào nút đó. Sau đó tính vector trung bình của mỗi nút.
* Cách tính: cộng tuyến tính các thành phần của các vector của hình ảnh trong nút sau đó chia trung bình.
* B4: Tạo nút gốc mới với các thành phần là 2 vector trung bình của 2 cụm vừa tạo ra. Link 2 vector với các nút con tương ứng.
* B5: Tiếp tục thêm hình vào cây.
* Cách thêm: Mỗi hình đưa vào sẽ so sánh với các vector trong nút gốc. Gần vector nào ta sẽ lấy link của vector đó để đi xuống node mà vector link tới. Sau khi thêm vào nút lá sẽ cập nhật lại vector trung bình.
* B6: Tiếp tục thêm hình vào cây. Nếu nút lá đầy sẽ tiếp tục tách nút và thêm 2 vector trung bình vào root.
* B7: Nếu nút root đầy sẽ tách nút root.
* Cách tách: chọn 2 vector xa nhau nhất tạo thành 2 nút (ở đây sẽ là nút không phải nút lá trong cây). Đưa các vector còn lại vào 2 nút vừa tạo theo quy tắc gần nút nào sẽ đưa vào nút đó. Sau đó tính vector trung bình của mỗi nút và tạo nút gốc mới với 2 thành phần là 2 vector trung bình vừa tạo.
* B8: Tiếp tục thêm ảnh vào cây.1
* Cách thêm: Chúng ta sẽ đệ quy để tìm được nút lá sẽ thêm ảnh. Đệ quy sẽ đi từ nút root của cây.
* Lấy vector của ảnh so sánh với các vector trong nút root. Sau đó sẽ đi xuống nút được vector link tới. Nếu chưa phải lá thì tiếp tục so sánh với các vector trong nút đó và tìm đường link đi xuống. Tới khi nào là nút lá thì sẽ thêm vào. Khi thêm vào sẽ đệ quy tính lại các vector trung bình ở trên.
* **Mã giả hàm thêm 1 ảnh vào cây**

**Hàm addImage()**

**If** len(root) > 20:

**cutNode()**

**If** root is **Leaf**:

Root.**addImage**(image)

**Else:**

Temp = root

**While** true:

Location = findMinDistance(list vector in temp, vector of image)

Link = temp[location].getLink()

Node = findNodeById(link)

**If** node is **Leaf**:

Break

Node.**addImage**(image)

Compute average vector in tree

**If** len(node) > 20:

**cutNode()**

**Hàm cutNode():**

vectorA, vectorB = find max distance between two vector of list vector in root

add vectorA to NodeA

add vectorB to NodeB

for vector in list vector in root:

if distance euclide of vectorA and vector < distance euclide of vectorB and vector:

add vector to NodeA

else:

add vector to NodeB

averageVectorA = compute average of list vector in NodeA

averageVectorB = compute average of list vector in NodeB

create a new root of tree with two average vector: averageVectorA and averageVectorB

add NodeA and NodeB to tree

### Thuật toán tìm kiếm ảnh trên cây

Sau khi lưu trữ chữ ký và định danh của hình ảnh tương ứng trên cây S-tree, quá trình truy vấn sẽ tìm ra các chữ ký tương tự của hình ảnh dựa trên việc duyệt cây S-tree. Sau khi tìm ra các chữ ký hình ảnh, dựa vào định danh của các hình ảnh sẽ tìm ra cụ thể các hình ảnh tương tự với hình ảnh truy vấn. Do đó, bài toán cần thực hiện là tìm ra chữ ký của hình ảnh và định danh tương ứng.

* **Mã giả thuật toán tìm kiếm ảnh trên cây:**

Node = root

Route =

**While** (Node != Leaf) {

**For** (SIG in range(0, len(list SIG of Node))) {

**FindMinDistance**(SIG, sig of query image))

}

Route.**add**(Node)

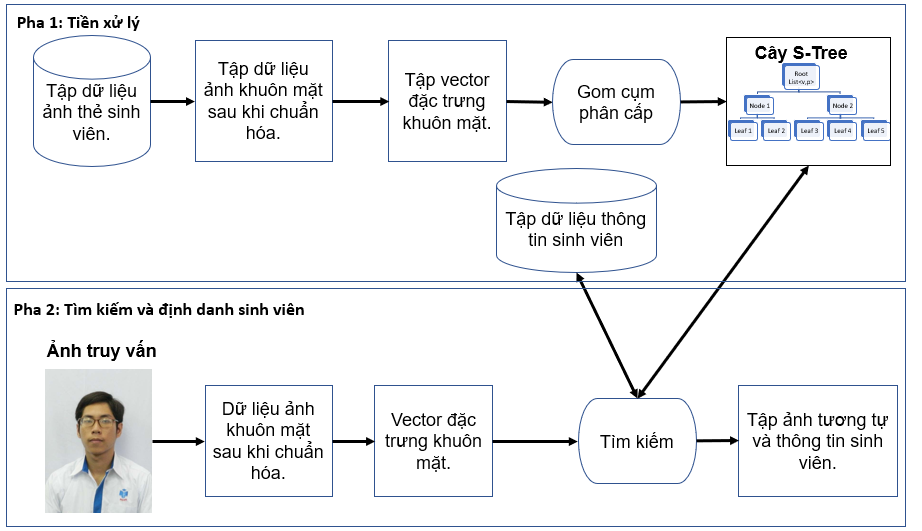
Node = SIG -> next

}

**Return** list SIG of Node

# THỰC NGHIỆM

## Mô hình thực nghiệm



Hình 4.1 Mô hình thực nghiệm của đề tài

Mô hình tổng quát của hệ thống tìm kiếm và định danh sinh viên qua hình ảnh. Mô hình bao gồm 2 pha là tiền xử lý, tìm kiếm và định danh sinh viên.

**Tiền xử lý:**

Bước 1: Từ tập dữ liệu ảnh thẻ của sinh viên: xác định khuôn mặt và tiến hành chuẩn hóa tập dữ liệu hình ảnh.

Bước 2: Trích xuất các điểm đặc trưng trên tập dữ liệu hình ảnh đã được chuẩn hóa ở bước 1. Từ các điểm đặc trưng của khuôn mặt: trích xuất ra vector đặc trưng của khuôn mặt.

Bước 3: Tiến hành gom cụm tập vector đã trích xuất ở bước 2 để tạo ra cây S-Tree.

**Tìm kiếm và định danh sinh viên:**

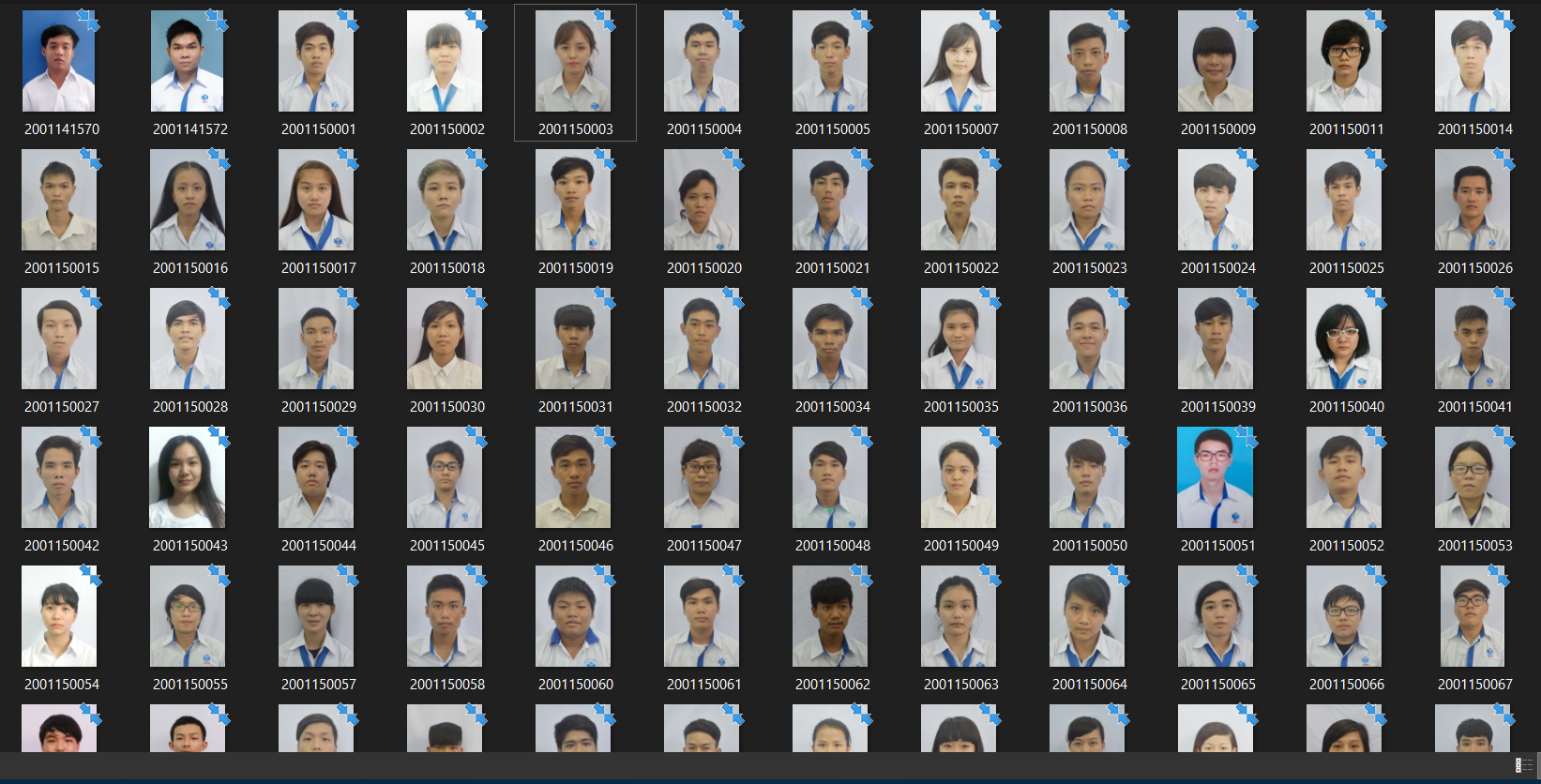
Bước 1: Từ ảnh truy vấn: xác định khuôn mặt và tiến hành chuẩn hóa hình ảnh.

Bước 2: Trích xuất các điểm đặc trưng trên ảnh truy vấn đã được chuẩn hóa ở bước 1. Từ các điểm đặc trưng của khuôn mặt: trích xuất ra vector đặc trưng của khuôn mặt.

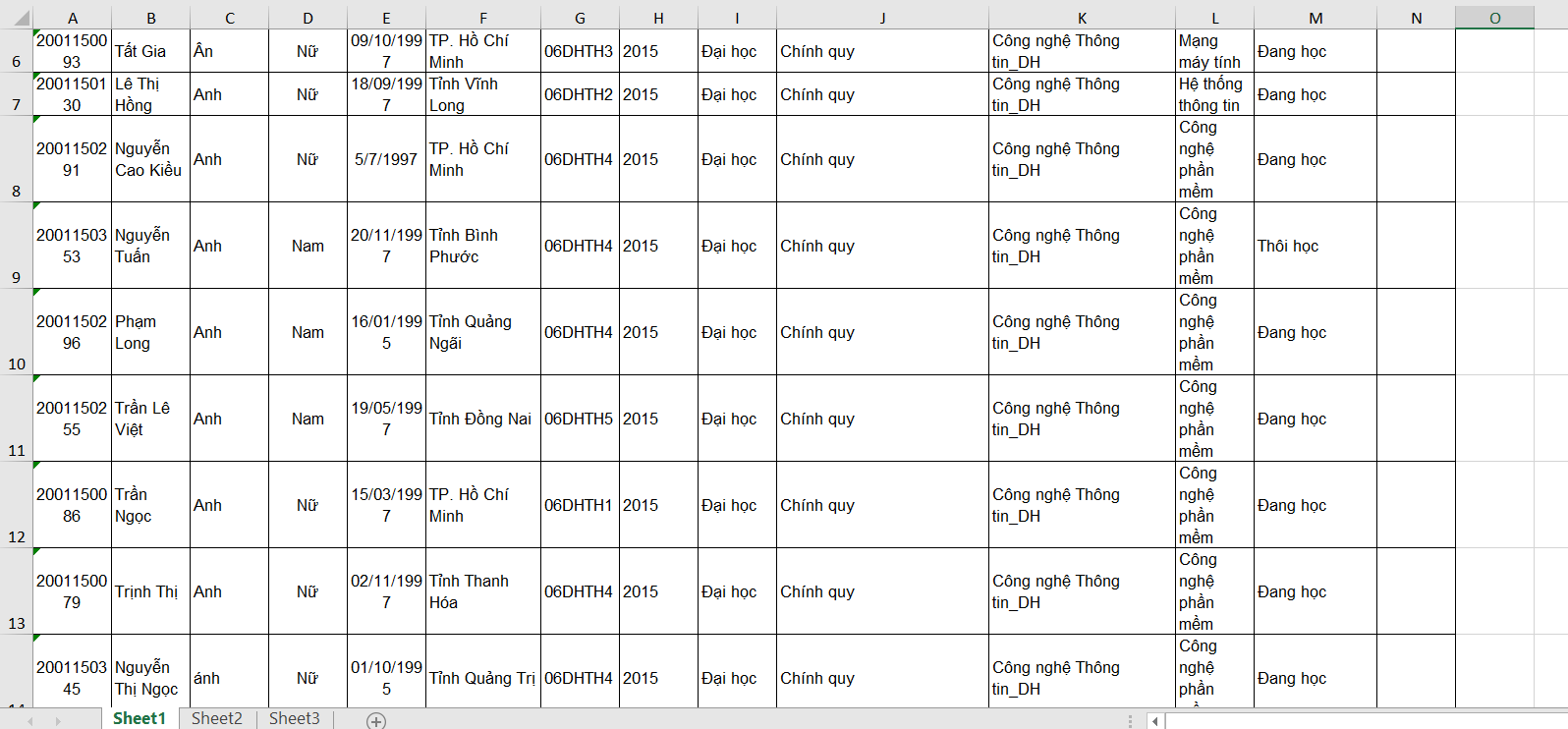
Bước 3: Tiến hành tìm kiếm trên cây S-Tree để lấy ra các ảnh tương tự. Từ đó truy xuất thông tin của sinh viên trên bộ dữ liệu sinh viên.

## Dữ liệu thực nghiệm

Đề tài thực nghiệm trên bộ dữ liệu sinh viên khoa công nghệ thông tin trường Đại học Công nghiệp Thực phẩm từ năm 2011-2018. Bộ dữ liệu sinh viên bao gồm ảnh thẻ với id là mã số của sinh viên và thông tin của sinh viên được lưu trữ trong file excel. Các thông tin của sinh viên gồm: mã số sinh viên, họ tên, giới tính, ngày sinh, lớp,…



Hình 4‑2 Tập ảnh thẻ sinh viên



Hình 4‑3 Tập thông tin sinh viên

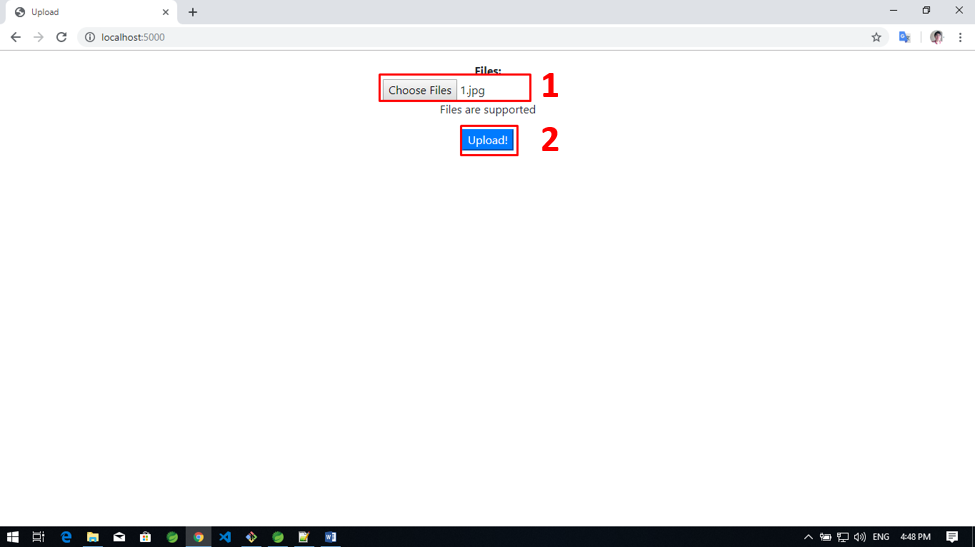
## Ứng dụng thực nghiệm

### Giao diện ứng dụng

**Giao diện chọn ảnh:**

Số 1: nút chọn ảnh tìm kiếm.

Số 2: nút tìm kiếm.



Hình 4‑4 Giao diện chọn ảnh của ứng dụng

**Giao diện kết quả tìm kiếm:**

Số 1: hình ảnh tìm kiếm.

Số 2: vùng các ảnh tìm kiếm được.

Số 3: vùng chọn ảnh để di chuyển sang màn hình thông tin ảnh.

Hình 4‑5 Giao diện kết quả tìm kiếm của ứng dụng

**3**

**2**

**1**

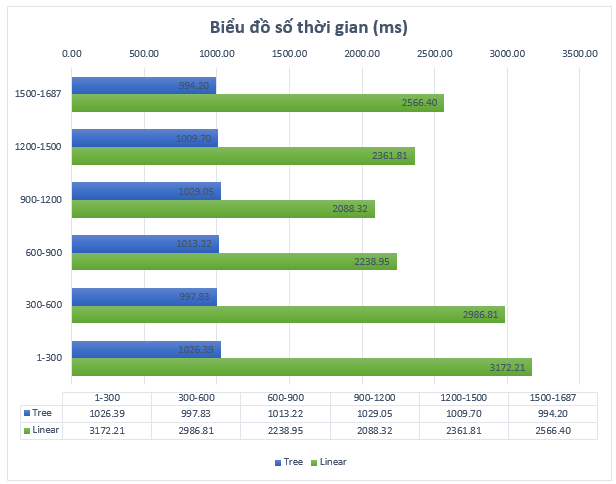
**Giao diện thông tin ảnh được chọn:**



Hình 4‑6 Giao diện hiển thị thông tin sinh viên của ứng dụng

### Thống kê thực nghiệm

Qua quá trình thực nghiệm cho thấy tốc độ tìm kiếm trên cây S-Tree nhanh hơn trên 50% so với phương pháp tìm kiếm tuyến tính thông thường. Độ chính xác của cây cũng đạt tỉ lệ rất cao. Đối với các ảnh thẻ có trong bộ dữ liệu thì đều có thể tìm được chính xác.



Hình 4‑7 Biểu đồ thống kê thời gian so sánh giữa tuyến tính và cây

# KẾT LUẬN

Đề tài đã xây dựng được một phương pháp tìm kiếm ảnh tương tự theo nội dung dựa trên cây S-Tree. Xây dựng được một ứng dụng tìm kiếm và định danh sinh viên khoa công nghệ thông tin trường đại học Công Nghiệp Thực Phẩm qua hình ảnh.

Theo kết quả thực nghiệm cho thấy, quá trình tạo cây S-tree từ vector đặc trưng của ảnh mất nhiều thời gian, nhưng việc truy vấn ảnh dựa trên cây S-tree diễn ra nhanh hơn rất nhiều so với phương pháp tìm kiếm tuyến tính thông thường.

Tuy nhiên, đề tài còn một số hạn chế như: chưa xử lý được trường hợp ảnh có nhiều khuôn mặt, ảnh chụp khuôn mặt có độ nghiêng lớn hơn 30o.

Hướng phát triển tiếp theo chúng em sẽ tiếp tục nghiên cứu các phương pháp chuẩn hóa, trích chọn vector đặc trưng của ảnh nhằm tăng độ chính xác của ứng dụng. Từ đó, mở rộng thêm các chức năng xoay quanh mô hình trích xuất ảnh như điểm danh sinh viên, giúp giảng viên truy cập và nắm được thông tin của một sinh viên chỉ thông qua bức ảnh của sinh viên đó. Qua đó dễ dàng hơn trong việc quản lý sinh viên của nhà trường.

Nhóm mong muốn áp dụng được các phương pháp trên vào hệ thống quản lý của nhà trường. Đặt nền móng cho việc sử dụng các công nghệ 4.0 vào hệ thống quản lý bộ máy của trường cũng như hướng tới một hệ thống tối ưu, đơn giản và hiểu quả cho cả sinh viên và các nhân viên khi làm việc với hệ thống.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Deloitte, *Photo sharing: trillions and rising*.: Deloitte Touche Tohmatsu Limited, Deloitte Global, 2016. |
| [2] | Deloitte. (2018) https://www2.deloitte.com. |
| [3] | H.R. Arabnia L. Deligiannidis, *Emerging Trends in Image Processing, Computer Vision, and Pattern Recognition*. ed S. Elliot, Elsevier, USA: Morgan Kaufmann, Waltham, MA 02451, 2015. |
| [4] | S. Rho, E.-I. Choi M.-H. Lee, *Ontology based user query interpretation for semantic multimedia contents retrieval*.: Multimedia Tools and Applications, vol. 73, no. 2, pp.901–915, 2014. |
| [5] | Introduction to OpenCV. [Online]. <https://docs.opencv.org/3.1.0/df/d65/tutorial_table_of_content_introduction.html> |
| [6] | IEEV. [Online]. <https://www.ieev.org/2013/04/khoang-cach-distance.html> |
| [7] | A. Amira, N. Ramzan A. Alzu’bi, *Semantic content-based image retrieval: A comprehensive study*.: Journal of Visual Communication and Image Representation, vol. 32, no. pp.20-54, 2015. |
| [8] | Carlo Tomasi, and Leonidas J. Guibas Yossi Rubner, *A Metric for Distributions with Applications to Image Databases*.: Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India, pp. 59-66, 1998. |