TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH CÔNG NGHỆ THÔNG TIN

**TÊN ĐỀ TÀI**

NGHIÊN CỨU VÀ XÂY DỰNG ỨNG DỤNG CHẤM CÔNG BẰNG NHẬN DIỆN KHUÔN MẶT

Sinh viên thực hiện : **Nguyễn Dũng Tuấn**

Lớp CNTT&TT2.2 – K56

Giáo viên hướng dẫn: ThS. **Hoàng Anh Việt**

HÀ NỘI 12-2015

# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

**1. Thông tin về sinh viên**

Họ và tên sinh viên: Nguyễn Dũng Tuấn

Điện thoại liên lạc: 01667653393 Email: tuanbk210@gmail.com

Lớp: CNTT&TT2.2 K56 Hệ đào tạo: Chính quy

Đồ án tốt nghiệp được thực hiện tại: **Bộ môn Công nghệ phần mềm – Viện CNTT&TT**

Thời gian làm ĐATN: Từ ngày 14/09/2015 đến 16/12/2015

**2. Mục đích nội dung của ĐATN**

Nghiên cứu và xây dựng ứng dụng chấm công bằng khuôn mặt

**3. Các nhiệm vụ cụ thể của ĐATN**

* Ở phía client: Nghiên cứu giải pháp dò tìm và nhận diện khuôn mặt dựa trên thuật toán ADABOOST để phát hiện khuôn mặt và thuật toán Local Binary Patterns Histograms để nhận diện khuôn mặt. Và để thực hiện tác giả sử dụng thư viện OpenCV kết hợp với ngôn ngữ Python.
* Ở phía server: xây dựng API giúp nhận thông tin từ phía client, từ đó xác nhận và chấm công cho nhân viên đó
* Client sau khi nhận diện được khuôn mặt thì đưa ra thông tin của người được nhận diện rồi gửi lên server để phục vụ việc chấm công cho công ty.
* Xây dựng mô hình ứng dụng thử nghiệm(dùng phần mềm đã xây dựng để chấm công).
* Cài đặt thử nghiệm mô hình đã được xây dựng trước đó và đánh giá kết quả.

**4. Lời cam đoan của sinh viên:**

Tôi – ***Nguyễn Dũng Tuấn*** - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của ***ThS. Hoàng Anh Việt***.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 16 tháng 12 năm 2015*  Tác giả ĐATN  *Nguyễn Dũng Tuấn* |

**5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:**

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Giáo viên hướng dẫn  *ThS. Hoàng Anh Việt* |

# TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Ngày nay, việc chấm công cho nhân viên được thực hiện chủ yếu bằng cách dùng sinh trắc học (ví dụ: quét vân tay), quẹt thẻ nhân viên hoặc đăng nhập vào website chấm công của công ty bằng mạng của công ty, đăng nhập vào phần mềm chấm công của công ty...

Với mong muốn có 1 cách chấm công khác cho nhân viên trong công ty, với tiềm năng rộng mở trong tương lai, trong đồ án này sẽ nghiên cứu và xây dựng mô hình dò tìm và nhận diện khuôn mặt thông qua camera của máy tính, từ đó phát hiện được đó là nhân viên nào, phục vụ cho mục đích chấm công.

Với cách chấm công như vậy thì ta cần tìm hiểu các thuật toán dò tìm và nhận diện khuôn mặt, từ đó mới có thể trích xuất ra thông tin hữu ích phục vụ việc chấm công.

Trong đồ án này sẽ trình bày giải pháp dò tìm và nhận diện khuôn mặt dựa trên thuật toán ADABOOST để phát hiện khuôn mặt và thuật toán Local Binary Patterns Histograms để nhận diện khuôn mặt. và để thực hiện tác giả sử dụng thư viện OpenCV kết hợp với ngôn ngữ Python.

Đồ án chia thành 5 chương:

* Chương 1: Tổng quan về bài toán nhận dạng khuôn mặt.
* Chương 2: Cơ sở lý thuyết về nhận dạng và phát hiện khuôn mặt.
* Chương 3: Thiết kế và xây dựng ứng dụng thử nghiệm chấm công bằng nhận diện khuôn mặt.
* Chương 4: Cài đặt mô hình và đánh giá kết quả.
* Chương 5: Kết luận và hướng phát triển.

# LỜI CẢM ƠN

Lời đầu tiên, em xin được gửi lời cảm ơn chân thành đến các thầy cô trong trường Đại Học Bách Khoa Hà Nội cũng như các thầy cô giáo trong Viện Công Nghệ Thông Tin và Truyền Thông đã truyền dạy cho em những kiến thức và kinh nghiệm quý báu trong suốt quá trình học tập và tu dưỡng trong 5 năm qua.

Em xin được gửi lời cảm ơn chân thành tới thầy Ths. Hoàng Anh Việt - Giảng viên bộ môn công nghệ phần mềm, Viện Công Nghệ Thông tin và Truyền Thông đã giúp đỡ, hướng dẫn và chỉ dạy em trong quá trình học tập nghiên cứu với những lời khuyên, bài học bổ ích.

Em cũng muốn gửi lời cảm ơn chân thành đến tập thể lớp CNTT&TT2.2 K56, là một tập thể đoàn kết, cùng nhau phát triển, tạo điều kiện cho các thành viên trong lớp học hỏi, cùng nhau trau dồi thêm những kiến thức mới.

Cuối cùng, em xin được gửi lời cảm ơn chân thành tới gia đình, bạn bè đã quan tâm, động viên, đóng góp ý kiến giúp đỡ em trong quá trình học tập, nghiên cứu và hoàn thành đồ án tốt nghiệp.

Hà Nội, ngày 16 tháng 12 năm 2015

Nguyễn Dũng Tuấn

CNTT&TT2.2 K56

# Mục lục

[PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP 2](#_Toc438048718)

[TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP 3](#_Toc438048719)

[LỜI CẢM ƠN 4](#_Toc438048720)

[Mục lục 5](#_Toc438048721)

[Danh mục hình ảnh 7](#_Toc438048722)

[Danh mục bảng biểu 8](#_Toc438048723)

[Lời nói đầu 9](#_Toc438048724)

[Chương 1: Tổng quan về bài toán nhận dạng khuôn mặt 10](#_Toc438048725)

[1.1. Giới thiệu bài toán nhận dạng khuôn mặt. 10](#_Toc438048726)

[1.2. Các kết quả nghiên cứu của bài toán nhận dạng khuôn mặt. 11](#_Toc438048727)

[1.3. Phân tích các đặc tính sinh trắc học 12](#_Toc438048728)

[1.4. Ứng dụng của bài toán nhận dạng khuôn mặt. 12](#_Toc438048729)

[1.5. Ưu và nhược điểm của các ứng dụng nhận diện khuôn mặt. 13](#_Toc438048730)

[1.6. Mục tiêu của đề tài 14](#_Toc438048731)

[1.7. Kết chương 14](#_Toc438048732)

[Chương 2: Cơ sở lý thuyết về nhận dạng và phát hiện khuôn mặt 15](#_Toc438048733)

[2.1. Bài toán phát hiện khuôn mặt. 15](#_Toc438048734)

[2.1.1. Các phương pháp phát hiện khuôn mặt. 15](#_Toc438048735)

[2.1.2. Phát hiện khuôn mặt bằng thuật toán ADABOOST. 15](#_Toc438048736)

[2.1.3. Các đặc trưng Haar-Like. 20](#_Toc438048737)

[2.1.4. Cascade of Classifiers. 24](#_Toc438048738)

[2.1.5. Nhận xét. 27](#_Toc438048739)

[2.2. Bài toán nhận diện khuôn mặt. 27](#_Toc438048740)

[2.2.1. Một số phương pháp nhận dạng khuôn mặt. 27](#_Toc438048741)

[2.2.2. Nhận dạng bằng phương pháp Local Binary Patterns Histograms. 28](#_Toc438048742)

[2.3. Kết chương. 32](#_Toc438048743)

[Chương 3: Thiết kế và xây dựng ứng dụng thử nghiệm chấm công bằng nhận diện khuôn mặt 33](#_Toc438048744)

[3.1. Phát biểu bài toán chấm công bằng nhận diện khuôn mặt. 33](#_Toc438048745)

[3.2. Giới thiệu về OpenCV. 34](#_Toc438048746)

[3.3. Giới thiệu về Python và thư viện OpenCV cho Python. 35](#_Toc438048747)

[3.3.1. Giới thiệu về Python 35](#_Toc438048748)

[3.3.2 Điểm mạnh 36](#_Toc438048749)

[3.3.3. Điểm yếu 36](#_Toc438048750)

[3.3.4. Thư viện OpenCV cho Python 36](#_Toc438048751)

[3.4. Phân tích thiết kế hệ thống. 36](#_Toc438048752)

[3.4.1. Biểu đồ Usercase của hệ thống 37](#_Toc438048753)

[3.4.2. Lưu đồ thuật toán chấm công bằng khuôn mặt. 40](#_Toc438048754)

[3.4.3. Phát hiện khuôn mặt. 41](#_Toc438048755)

[3.4.4. Huấn luyện nhận dạng khuôn mặt 42](#_Toc438048756)

[3.4.5. Nhận dạng khuôn mặt 44](#_Toc438048757)

[3.5. Kết chương. 46](#_Toc438048758)

[Chương 4: Cài đặt mô hình và đánh giá kết quả 47](#_Toc438048759)

[4.1. Chương trình nhận diện khuôn mặt. 47](#_Toc438048760)

[4.2. Kết quả phát hiện khuôn mặt. 48](#_Toc438048761)

[4.3. Nhận diện khuôn mặt 50](#_Toc438048762)

[4.4. Chấm công bằng nhận diện khuôn mặt. 50](#_Toc438048763)

[4.5. Đánh giá độ tin cậy của chương trình. 50](#_Toc438048764)

[4.5.1. Mô hình đánh giá. 50](#_Toc438048765)

[4.5.2. Kết quả đánh giá về mặt thời gian. 51](#_Toc438048766)

[4.5.3. Kết quả đánh giá về tính chính xác nhận dạng khuôn mặt. 51](#_Toc438048767)

[4.6. Kết luận. 52](#_Toc438048768)

[4.7. Kết chương. 52](#_Toc438048769)

[Chương 5: Kết luận và hướng phát triển. 54](#_Toc438048770)

[TÀI LIỆU THAM KHẢO 56](#_Toc438048771)

# Danh mục hình ảnh

[Hình 1.1: Hệ thống nhận dạng khuôn mặt 11](#_Toc438048772)

[Hình 2.1: Boosting. 17](#_Toc438048773)

[Hình 2.2 : Sơ đồ mô tả thuật toán Adaboost. 19](#_Toc438048774)

[Hình 2.3: Đặt trưng Haar-like cơ bản. 21](#_Toc438048775)

[Hình 2.4: Đặc trưng cạnh 21](#_Toc438048776)

[Hình 2.5: Đặc trưng đường 21](#_Toc438048777)

[Hình 2.6: Đặc trưng xung quanh tâm 21](#_Toc438048778)

[Hình 2.7: Cách tính Integral Image của ảnh. 22](#_Toc438048779)

[Hình 2.8: Ví dụ cách tính nhanh tổng các điểm ảnh của vùng D. 22](#_Toc438048780)

[Hình 2.9: Tính tổng điểm ảnh của vùng D với các đặc trưng xoay 450. 23](#_Toc438048781)

[Hình 2.10: Hệ thống phát hiện khuôn mặt. 24](#_Toc438048782)

[Hình 2.11: Cascade of classifiers. 25](#_Toc438048783)

[Hình 2.12: Cascade of boosting classifiers. 26](#_Toc438048784)

[Hình 2.13: Ví dụ về LBP và độ tương phản cục bộ C. 29](#_Toc438048785)

[Hình 2.14: Ví dụ tính toán LBP. 29](#_Toc438048786)

[Hình 2.15: Minh họa toán tử LBP mở rộng với các giá trị P và R khác nhau. 30](#_Toc438048787)

[Hình 2.16: Điểm chấm, điểm chấm nhạt, điểm cuối đường thẳng, biên cạnh, góc được phát hiện bởi LBPu2. 31](#_Toc438048788)

[Hình 2.17: Biến đổi ảnh bằng LBP 31](#_Toc438048789)

[Hình 2.18: Ảnh khuôn mặt được chia ra thành các miền con 31](#_Toc438048790)

[Hình 3.1 : Hệ thống chấm công bằng khuôn mặt 33](#_Toc438048791)

[Hình 3.2: Cấu trúc cơ bản của OpenCV 35](#_Toc438048792)

[Hình 3.3: Biểu đồ Usercase tổng quát 37](#_Toc438048793)

[Hình 3.4: Biểu đồ Usecase phân rã mức 2 Capture Face 38](#_Toc438048794)

[Hình 3.5: Biểu đồ Usecase phân rã mức 2 Training Face 39](#_Toc438048795)

[Hình 3.6: Lưu đồ thuật toán chấm công bằng khuôn mặt 40](#_Toc438048796)

[Hình 3.7: Chức năng phát hiện khuôn mặt 41](#_Toc438048797)

[Hình 3.8: Chụp ảnh khuôn mặt 42](#_Toc438048798)

[Hình 3.9: Huấn luyện nhận dạng khuôn mặt 43](#_Toc438048799)

[Hình 3.10 : Thiết lập LBP Histogram cho các mẫu 44](#_Toc438048800)

[Hình 3.11: So sánh mô hình ảnh nhận dạng (Model) với ảnh mẫu (Sample) 45](#_Toc438048801)

[Hình 3.12: nhận dạng đối tượng bằng so sánh khoảng cách 45](#_Toc438048802)

[Hình 3.13: Nhận dạng khuôn mặt 46](#_Toc438048803)

[Hình 4.1 : Giao diện hệ thống 47](#_Toc438048804)

[Hình 4.2 : Phát hiện khuôn mặt đúng 49](#_Toc438048805)

[Hình 4.3 : Phát hiện khuôn mặt sai 49](#_Toc438048806)

# Danh mục bảng biểu

[Bảng 4.1: Thời gian thực thi 51](#_Toc438048807)

[Bảng 4.2: Kết quả nhận dạng khuôn mặt. 52](#_Toc438048808)

# Lời nói đầu

Nhận diện khuôn mặt – kỹ thuật dùng để xác định khuôn mặt trong khuôn hình là ai – là một kỹ thuật luôn được quan tâm bởi tính ứng dụng của nó, nếu giải thuật tốt, kết quả đáng tin cậy, việc nhận dạng khuôn mặt có thể được sử dụng trong các hệ thống thông minh như tự động điều khiển, cảnh báo, lấy thông tin thông qua phân tích khuôn mặt. Tuy nhiên, do tính phổ biến của đặc trưng khuôn mặt, cùng với sự phong phú phức tạp trong sự xuất hiện của con người nên nhận diện khuôn mặt vẫn chỉ được ứng dụng trong các hệ thống không yêu cầu cao về bảo mật như những hệ thống đưa ra thông tin hay cảnh báo.

Chấm công cho nhân viên, 1 công việc đã xuất hiện từ khá lâu, xuất hiện từ trước khi máy tính ra đời. Khi máy tính xuất hiện, nó đã làm thay đổi hoàn toàn cách chấm công của các công ty, giúp công ty quản lý việc chấm công 1 cách hiệu quả hơn. Tuy nhiên, việc chấm công cho công ty hiện nay mới chỉ dừng lại ở việc dùng sinh trắc học (ví dụ: quét vân tay), quét thẻ nhân viên hoặc đăng nhập website chấm công của công ty bằng mạng nội bộ của công ty, đăng nhập vào phần mềm chấm công của công ty... Với mong muốn có 1 cách chấm công khác cho nhân viên trong công ty, với tiềm năng rộng mở trong tương lai, trong đồ án này sẽ nghiên cứu và xây dựng mô hình dò tìm và nhận diện khuôn mặt thông qua camera của máy tính, từ đó phát hiện được đó là nhân viên nào, phục vụ cho mục đích chấm công.

# Chương 1: Tổng quan về bài toán nhận dạng khuôn mặt

|  |
| --- |
| Nội dung chương này trình bày những vấn đề sau:   * Giới thiệu bài toán nhận dạng khuôn mặt. * Các kết quả bài toán nhận dạng khuôn mặt. * Phân tích các đặc tính sinh trắc học. * Ứng dụng của bài toán nhận dạng khuôn mặt. * Ưu và nhược điểm của các ứng dụng nhận dạng khuôn mặt. * Mục tiêu của đề tài. * Kết chương |

## Giới thiệu bài toán nhận dạng khuôn mặt.

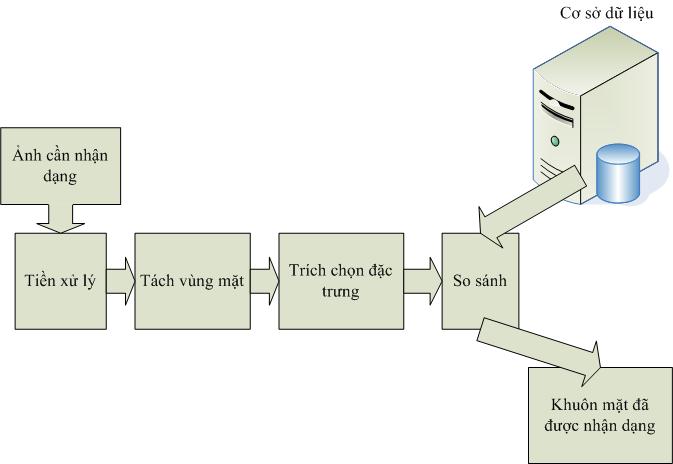
Nhận diện khuôn mặt là bài toán đã được tìm hiểu, nghiên cứu từ khoảng những năm 70 của thế kỷ trước, đã có rất nhiều giải thuật ra đời như LBP, PCA,… để cho phép có thể nhận diện khuôn mặt được tốt nhất trong các điều kiện khác nhau như trong tình trạng ảnh có ánh sáng yếu, biểu cảm khuôn mặt, góc nhìn thay đổi… Và vẫn còn tiếp tục phát triển để đạt được hiệu quả cao hơn trong việc nhận dạng khuôn mặt do yêu cầu tính chính xác ngày càng cao trong các hệ thống mong muốn sử dụng kỹ thuật này và để khắc phục các khó khăn vẫn còn tiềm tàng khi nhận diện khuôn mặt:

* Hướng của khuôn mặt đối với máy ảnh, như: nhìn thẳng, nhìn nghiêng hay nhìn từ trên xuống. Cùng trong một ảnh có thể có nhiều khuôn mặt ở những tư thế khác nhau.
* Sự có mặt của các chi tiết không phải là đặc trưng riêng của khuôn mặt, như: râu quai nón, mắt kính, ….
* Các nét mặt khác nhau trên khuôn mặt, như: vui, buồn, ngạc nhiên, ….
* Khuôn mặt bị che khuất bởi các đối tượng khác có trong ảnh.
* Có những người có đặc trưng khá tương đồng nhau. Không có tính duy nhất như dấu vân tay hay đồng tử mắt.
* Ảnh không rõ nét hoặc bị mất mát thông tin quan trọng gây khó khăn trong quá trình nhận dạng.
* Trong video có thể dùng ảnh, hình nộm,… để thay thế cho khuôn mặt thật nhằm đánh lừa hệ thống.

Vì vậy, trong phần này báo cáo sẽ trình bày tổng quan về phương pháp chung để có thể nhận diện được khuôn mặt từ đó làm nền tảng để nghiên cứu về các kỹ thuật cụ thể và cài đặt ứng dụng.

## Các kết quả nghiên cứu của bài toán nhận dạng khuôn mặt.

Để có thể phát hiện được khuôn mặt, đầu tiên cần phải xác định được các đặc điểm của khuôn mặt con người dựa trên các đặc tính sinh trắc học, sau đó các khuôn mặt trong hình ảnh sẽ được xác định phân biệt với các thực thể khác trong ảnh và cuối cùng là được gán nhãn. Cấu trúc của một hệ thống nhận dạng được thể hiện như hình sau:



Hình 1.1: Hệ thống nhận dạng khuôn mặt

* Tiền xử lý: bước này cho phép hệ thống xử lý ảnh trước khi đưa vào huấn luyện để phát hiện khuôn mặt và gán nhãn ảnh. Ảnh càng có chất lượng tốt sẽ càng đem lại hiệu quả cao và giảm thời gian tính toán cho các bước sau. Công việc của bước này có thể là tăng giảm độ sáng, khôi phục ảnh,… Ngày nay có hai phương hướng nghiên cứu là cố gắng tiền xử lý tốt hoặc huấn luyện hệ thống cả theo những đặc điểm sẵn có.
* Tách khuôn mặt: bước này làm nhiệm vụ xác định vị trí, kích cỡ của một hoặc nhiều khuôn mặt trên ảnh chụp từ đó tách ra phần mặt. Phần ảnh mặt đươc tách ra thường nhỏ hơn nhiều so với ảnh chụp ban đầu, nó sẽ là các khuôn mặt cần tìm và chức năng trích chọn đặc trưng sẽ sử dụng các ảnh được tách ra này.
* Trích chọn đặc trưng: Tìm ra các đặc trưng chính của ảnh mặt, từ các đặc trưng này hình thành các vector đặc trưng, các vector này sẽ được sử dụng để đối sánh sự giống nhau giữa ảnh mặt cần tìm và ảnh mặt trong cơ sở dữ liệu.
* Đối sánh: Thực hiện việc so sánh giữa các vector đặc trưng để chọn ra độ tương tự giữa ảnh cần tìm và ảnh trong cơ sở dữ liệu.

Hệ thống nhận dạng khuôn mặt cần đảm bảo các yêu cầu:

* Độ chính xác nhận dạng có thể chấp nhận được đối với yêu cầu của bài toán nhận dạng;
* Tốc độ vận hành cao đối với các cơ sở dữ liệu lớn và số lượng các yêu cầu có thể giải quyết được;
* Phù hợp với môi trường cài đặt, dễ sử dụng.

## Phân tích các đặc tính sinh trắc học

Đặc tính sinh trắc của người là đặc tính đo được các nét hành vi riêng của con người. Nhận diện người được thực hiện trong quá trình kiểm tra tính đồng nhất của đặc tính sinh trắc giữa người cần kiểm tra với đặc tính tương tự của người đã được lưu sẵn trong cơ sở dữ liệu. Có thể nhận diện người dựa trên các đặc điểm sinh trắc tĩnh trên người như khuôn mặt, mắt, vân tay, bàn tay, gen…hay các đặc trưng hành vi như dáng đi, chữ viết, giọng nói…

Dữ liệu sinh trắc lý tưởng của người cần phải có những đặc tính cơ bản như:

* Tính tổng quát: Mỗi người đều được thể hiện bởi đặc tính này.
* Tính duy nhất: Không thể tồn tại hai người có đặc tính giống nhau.
* Tính thường xuyên: Là sự độc lập của đặc tính đối với thời gian.
* Tính thu thập được: Là đặc tính được thu thập một cách tương đối đơn giản và nhanh chóng từ mỗi cá nhân và có thể được chi tiết hóa.

## Ứng dụng của bài toán nhận dạng khuôn mặt.

Các ứng cơ bản của xác định khuôn mặt có thể kể đến là:

**Xác minh tội phạm:** Dựa vào ảnh của một người, nhận dạng xem người đấy có phải là tội phạm hay không bằng cách so sách với các ảnh tội phạm đang được lưu trữ. Hoặc có thể sử dụng camera để phát hiện tội phạm trong đám đông. Ứng dụng này giúp cơ quan an ninh quản lý con người tốt hơn.

**Camera chống trộm:** Các hệ thống camera sẽ xác định đâu là con người và theo dõi xem con người đó có làm gì phạm pháp không, ví dụ như lấy trộm đồ, xâm nhập bất hợp pháp vào một khu vực nào đó.

**Bảo mật:** Các ứng dụng về bảo mật rất đa dạng, một trong số đó là công nghệ nhận dạng khuôn mặt của laptop, công nghệ này cho phép chủ nhân của máy tính chỉ cần ngồi trước máy là có thể đăng nhập được. Để sử dụng công nghệ này, người dùng phải sử dụng một webcam để chụp ảnh khuôn mặt của mình và cho máy “học” thuộc các đặc điểm của khuôn mặt giúp cho quá trình đăng nhập sau này.

**Lưu trữ khuôn mặt:** Xác định khuôn mặt có thể được ứng dụng trong các trạm rút tiền tự động (ATM) để lưu trữ khuôn mặt của người rút tiền. Hiện nay có những người bị người khác lấy trộm thẻ ATM và mã PIN, và bị rút tiền trộm, hoặc có những chủ tài khoản đi rút tiền nhưng lại báo với ngân hàng là bị mất thẻ và bị rút tiền trộm. Nếu lưu trữ được khuôn mặt của người rút tiền, ngân hàng có thể đối chứng và xử lý dễ dàng hơn.

**Các ứng dụng khác:**

* Vào/ra tự động: văn phòng, công ty, trụ sở, máy tính,… Kết hợp thêm vân tay và mống mắt. Cho phép nhân viên được ra vào nơi cần thiết.
* An ninh sân bay, xuất nhập cảnh: Dùng camera quan sát để xác thực người nhập cảnh và kiểm tra xem người đấy có phải là tội phạm hay phần tử khủng bố không.
* Tìm kiếm và tổ chức dữ liệu liên quan đến con người thông qua khuôn mặt trên nhiều hệ cơ sở dữ liệu lưu trữ thật lớn, như internet, các hãng truyền hình,…
* Tương lai sẽ phát triển các loại thẻ thông minh có tích hợp sẵn đặc trưng của người dùng trên đó, bất cứ người dùng nào dùng để truy cập hay xử lý tại các hệ thống sẽ được yêu cầu kiểm tra các đặc trưng khuôn mặt so với thẻ để biết nay có phải là chủ thẻ hay không.
* …

## Ưu và nhược điểm của các ứng dụng nhận diện khuôn mặt.

**Ưu điểm:**

* Là công nghệ tiên tiến đang ngày một được ứng dụng rông trong đời sống.
* Được áp dụng trong nhiều lĩnh vực: Giám sát an ninh, chống tội phạm, tìm kiếm thông tin,…

**Nhược điểm:**

* Độ chính xác của các ứng dụng này chưa cao.
* Tính chính xác phụ thuộc nhiều vào ánh sáng, góc nhìn của máy, biểu cảm của khuôn mặt muốn nhận diện …
* Máy khó phân biệt giữa khuôn mặt với con rối hay tranh ảnh.

## Mục tiêu của đề tài

Trong đồ án này, mục tiêu cần đạt được:

* Nghiên cứu thuật toán Adaboost.
* Nghiên cứu và cài đặt thuật toán Adaboost để phát hiện vùng khuôn mặt.
* Nghiên cứu và sử dụng tập thuộc tính Haar-like vào bài toán phát hiện khuôn mặt.
* Nghiên cứu và sử dụng thuật toán Cascade of Classifier vào việc tăng tốc độ thực thi và giảm khả năng phát hiện sai của ứng dụng.
* Nghiên cứu thuât toán Local Binary Patterns Histograms.
* Nghiên cứu và cài đặt thuật toán LBP để nhận diện khuôn mặt.
* Đánh giá về tính chính xác của thuật toán.
* Xây dựng ứng dụng chấm công bằng khuôn mặt
* Đề xuất mô hình.
* Cài đặt ứng dụng cụ thể.

## Kết chương

Như vậy ở Chương 1 người viết đồ án đã trình bày các vấn đề: Giới thiệu về bài toán nhận dạng khuôn mặt, các kết quả nghiên cứu của bài toán nhận dạng khuôn mặt, phân tích các đặc tính sinh trắc học, ứng dụng của bài toán nhận dạng khuôn mặt, ưu và nhược điểm của các ứng dụng nhận dạng khuôn mặt và mục tiêu của đề tài. Sang Chương 2 người viết báo cáo sẽ trình bày về cơ sở lý thuyết của bài toán nhận dạng khuôn mặt.

# Chương 2: Cơ sở lý thuyết về nhận dạng và phát hiện khuôn mặt

|  |
| --- |
| Nội dung chương này người viết báo cáo sẽ trình bày 2 vấn đề lớn là:   * Bài toán phát hiện khuôn mặt. * Bài toán nhận diện khuôn mặt. * Kết chương. |

## 2.1. Bài toán phát hiện khuôn mặt.

### 2.1.1. Các phương pháp phát hiện khuôn mặt.

Có nhiều nghiên cứu tìm phương pháp phát hiện khuôn mặt, từ ảnh xám đến ngày nay là ảnh màu. Dựa vào tính chất của các phương pháp phát hiện khuôn mặt trên ảnh, chúng ta có thể phân chia các phương pháp này thành bốn hướng tiếp cận chính:

* Hướng tiếp cận dựa trên tri thức: Mã hóa các hiểu biết của con người về các loại khuôn mặt thành các luật. Thông thường các luật mô tả quan hệ của các đặc trưng.
* Hướng tiếp cận dựa trên đặc trưng không thay đổi: Mục tiêu các thuật toán đi tìm các đặc trưng mô tả cấu trúc khuôn mặt mà các đặc trưng này sẽ không thay đổi khi tư thế khuôn mặt, vị trí đặt thiết bị thu hình hoặc điều kiện ánh sáng thay đổi.
* Hướng tiếp cận dựa trên so khớp mẫu: Dùng các mẫu chuẩn của khuôn mặt (các mẫu này được chọn lựa và lưu trữ) để mô tả cho khuôn mặt hay các đặc trưng khuôn mặt (các mẫu này phải chọn làm sao cho tách biệt nhau theo tiêu chuẩn mà các tác giả định ra để so sánh).
* Hướng tiếp cận dựa trên diện mạo: Trái ngược hẳn với so khớp mẫu, các mô hình học ở đây được học từ một tập ảnh huấn luyện cho trước. Sau đó hệ thống sẽ xác định khuôn mặt. Một số tác giả còn gọi hướng tiếp cận này là hướng tiếp cận theo phương pháp học.

### 2.1.2. Phát hiện khuôn mặt bằng thuật toán ADABOOST.

Học máy Adaboost là một cách trong hướng tiếp cận dựa trên diện mạo, Viola và Jones dùng AdaBoost kết hợp cascade để xác định khuôn mặt với các đặc trưng dạng Haar wavelet-like. Tốc độ xử lý khá nhanh và tỷ lệ chính xác hơn 80% trên ảnh xám.

Thuật toán học máy Adaboost được phát triển dựa trên thuật toán boosting, do đó báo cáo sẽ trình bầy một chút về thuật toán boosting trước khi trình bầy về adaboost [1][2].

**Thuật toán boosting** [1]**:**

Về lịch sử, boosting bắt nguồn từ câu hỏi nổi tiếng được đưa ra bời Kearns vào năm 1989 : “*Liệu có thể tạo ra một strong classifier (bộ phân loại mạnh) từ một tập các weak classifiers(bộ phân loại yếu)?”*. Năm 1990, Robert Schapire đưa ra thuật toán boosting đầu tiên, tiếp đến năm 1993 thì nó được Drucker, Schapire và Simard kiểm nghiệm trong trong các chương trình nhận dạng (*OCR application*). Freund đã tiếp tục các nghiên cứu của Schaprire, và đến năm 1995 thì ông cùng với Schapire phát triển boosting thành adaboost [2].

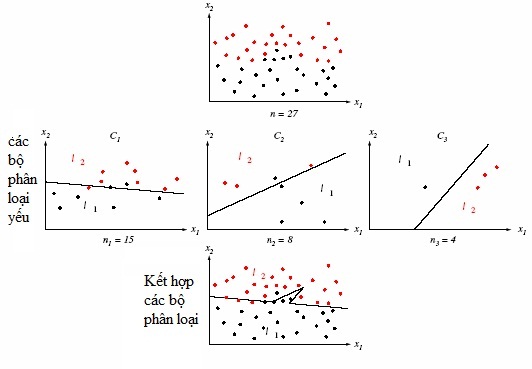
Như vậy, nguyên lý cơ bản của boosting là sự kết hợp các *weak classifiers* thành một *strong classifier*. Trong đó, *weak classifier* là các bộ phân loại đơn giản chỉ cần có độ chính xác trên 50%. Bằng cách này, chúng ta nói bộ phân loại đã được “boost” [2].

Để hiểu cách hoạt động của thuật toán boosting, ta xét một bài toán phân loại 2 lớp (mẫu cần nhận dạng chỉ thuộc một trong hai lớp) với *D* là tập huấn luyện gồm có *n* mẫu. Trước tiên, chúng ta sẽ chọn ngẫu nhiên ra *n1* mẫu từ tập *D* (*n1<n*) để tạo tập *D1*. Sau đó, chúng ta sẽ xây dựng weak classifier đầu tiên *C1* từ tập *D1*. Tiếp theo, chúng ta xây dựng tập *D2* để huấn luyện bộ phân loại *C2*. *D2* sẽ được xây dựng sao cho một nửa số mẫu của nó được phân loại đúng bởi *C1* và nửa còn lại bị phân loại sai bởi *C1*. Bằng cách này, *D2* chứa đựng những thông tin bổ sung cho *C1*. Bây giờ chúng ta sẽ xây dựng weak classifier *C2* từ *D2*.

Tiếp theo, chúng ta sẽ xây dựng tập *D3* từ những mẫu không được phân loại tốt bởi sự kết hợp giữa *C1* và *C2*: những mẫu còn lại trong *D* mà *C1* và *C2* cho kết quả khác nhau. Như vậy, *D3* sẽ gồm những mẫu mà *C1 và C2* hoạt động không hiệu quả. Sau cùng, chúng ta sẽ huấn luyện bộ phân loại *C3* từ *D3*.

Bây giờ chúng ta đã có một strong classifier: sự kết hợp *C1*, *C2* và *C3*. Khi tiến hành nhận dạng một mẫu X, kết quả sẽ được quyết định bởi sự thỏa thuận của 3 bộ *C1*, *C2* và *C3*: Nếu cả *C1* và *C2* đều phân *X* vào cùng một lớp thì lớp này chính là kết quả phân loại của *X*; ngược lại, nếu *C1* và *C2* phân X vào 2 lớp khác nhau, C3 sẽ quyết định *X* thuộc về lớp nào.

Như đã biết, AdaBoost (*Adaptive Boost*) là một bộ phân loại mạnh phi tuyến phức dựa trên hướng tiếp cận boosting được Freund và Schapire đưa ra vào năm 1995. Adaboost hoạt động trên nguyên tắc kết hợp tuyến tính các weak classifier dựa trên các đặc trưng Haar- Like để hình thành một strong classifier.



Hình 2.1: Boosting.

Để có thể kết hợp các bộ phân loại yếu, adaboost sử dụng một trọng số (weight) để đánh dấu các mẫu khó nhận dạng. Trong quá trình huấn luyện, cứ mỗi weak classifier được xây dựng, thuật toán sẽ tiến hành cập nhật lại trọng số để chuẩn bị cho việc xây dựng weak classifier tiếp theo: tăng trọng số của các mẫu bị nhận dạng sai và giảm trọng số của các mẫu được nhận dạng đúng bởi weak classifier vừa xây dựng. Bằng cách này, các weak classifier sau có thể tập trung vào các mẫu mà các weak classifier trước đó chưa thực hiện tốt. Sau cùng các weak classifier sẽ được kết hợp tùy theo mức độ ‘tốt’ của chúng để tạo nên một strong classifier [4].

Các weak classifiers hk*(x)* là các bộ phân loại yếu, được biểu diễn như sau:

Trong đó:

* x : mẫu hay cửa sổ con cần xét ( x = (x1,x2,…,xn) là vector đặc trưng của mẫu)
* : ngưỡng
* fk: giá trị của đặc trưng Haar-like
* pk: hệ số quyết định chiều của bất phương trình

Công thức trên có thể diễn giải như sau: Nếu giá trị đặc trưng của mẫu cho bởi hàm đánh giá của bộ phân loại vượt qua một ngưỡng cho trước thì mẫu đấy là khuôn mặt (gọi là *object*: đối tượng cần nhận dạng), ngược lại thì mẫu là *background* (không phải là đối tượng).

**Thuật toán AdaBoost:** [1]

Bước 1: Cho một tập gồm n mẫu có đánh dấu (x1,y1), (x2,y2),… (xn,yn) với xk∈ (xk1, xk2, … , xkm­) là vector đặc trưng và yk∈ (-1, 1) là nhãn của mẫu (1 ứng với object, -1 ứng với background).

Bước 2: Khởi tạo trọng số ban đầu cho tất cả các mẫu: với m là số mẫu đúng (ứng với object và y = 1) và l là số mẫu sai (ứng với background và y = -1).



Bước 3: Xây dựng T weak classifiers

Lặp t = 1, …, T

* Với mỗi đặc trưng trong vector đặc trưng, xây dựng một weak classifier hj với ngưỡng θj và lỗi εj.



* Chọn ra hj với εj nhỏ nhất, ta được ht:



* Cập nhật lại trọng số:



* Trong đó:



: Hệ số dùng để đưa về đoạn [0,1] (normalization factor)



Bước 4: Strong classifier được xây dựng :



**Sơ đồ khối:**

Bắt đầu huấn luyện

Khởi tạo tập đặc trưng ban đầu

Xác định các đặc trưng trong từng mẫu, xây dựng các bộ phân loại yếu tương ứng

Đặc trưng haar-like

Tính toán giá trị lỗi cho mỗi đặc trưng (*false alarm*)

Xác định ngưỡng

Chọn *weak classifier* có giá trị lỗi bé nhất

Lưu weak classifier được chọn

Tập các mẫu và trọng số

Tập các mẫu

Mẫu sai

Mẫu đúng

Cập nhật lại trọng số

Sai

Kết thúc, strong classifier đuợc xây dựng

Đúng

False alarm ≤max false alarm

Hình 2.2 : Sơ đồ mô tả thuật toán Adaboost.

Quá trình huấn luyện bộ phân loại được thực hiện bằng một vòng lặp mà ở mỗi bước lặp, thuật toán sẽ chọn ra weak classifier ht thực hiện việc phân loại với lỗi εt nhỏ nhất (do đó sẽ là bộ phân loại tốt nhất) để bổ sung vào strong classifier. Mỗi khi chọn được 1 bộ phân loại ht, Adaboost sẽ tính giá trị αt theo công thức ở trên. αt cũng được chọn trên nguyên tắc làm giảm thiểu giá trị lỗi εt [6].

Hệ số αt nói lên mức độ quan trọng của ht:

* Trong công thức của bộ phân loại H(x):

Ta thấy tất cả các bộ phân loại ht đều có đóng góp vào kết quả của bộ phân loại H(x), và mức độ đóng góp của chúng phụ thuộc vào giá trị αt tương ứng: ht với αt càng lớn thì nó càng có vài trò quan trọng trong H(x).

* Trong công thức tính αt:

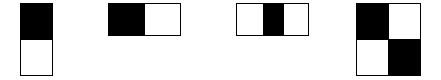
Dễ thấy giá trị αt tỉ lệ nghịch với εt. Bởi vì ht được chọn với tiêu chí đạt εt nhỏ nhất, do đó nó sẽ đảm bảo giá trị αt lớn nhất.

Sau khi tính được giá trị αt, Adaboost tiến hành cập nhật lại trọng số của các mẫu: tăng trọng số các mẫu mà ht phân loại sai, giảm trọng số các mẫu mà ht phân loại đúng. Bằng cách này, trọng số của mẫu phản ánh được mức độ khó nhận dạng của mẫu đó và ht­+1 sẽ ưu tiên học cách phân loại những mẫu này.

Vòng lặp xây dựng strong classifier sẽ dừng lại sau T lần lặp. Trong thực tế cài đặt (thư viện OpenCV của Intel), người ta ít sử dụng giá trị T vì không có công thức nào đảm bảo tính được giá trị T tối ưu cho quá trình huấn luyện. Thay vào đó, người ta sử dụng giá trị *max false positive* hay *max false alarm* (tỉ lệ nhận dạng sai tối đa các mẫu background). Tỉ lệ này của bộ phân loại cần xây dựng không được phép vượt quá giá trị này. Khi đó, qua các lần lặp, *false alarm* của strong classifier Ht(x) xây dựng được (tại lần lặp thứ t) sẽ giảm dần, và vòng lặp kết thúc khi tỉ lệ này thấp hơn *max false alarm* [1][2]*.*

### 2.1.3. Các đặc trưng Haar-Like.

Viola và Jones dùng 4 đặc trưng cơ bản để xác định khuôn mặt. Mỗi đặc trưng Haar–like là sự kết hợp của hai hay ba hình chữ nhật "trắng" hay "đen" như trong hình sau [1]:



Hình 2.3: Đặt trưng Haar-like cơ bản.

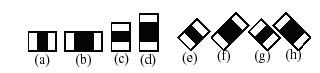
Để áp dụng các đặt trưng này vào việc bài toán xác định khuôn mặt, 4 đặt trưng Haar-like cơ bản được mở rộng ra, và được chia làm 3 tập đặc trưng như sau [1][6]:

1. *Đặc trưng cạnh (edge features):*

*D:\Hoc Tap\Course VIII\OpenCV\face detection\search_files\ap_20090116031945921.jpg*

Hình 2.4: Đặc trưng cạnh

*2. Đặc trưng đường (line features):*

**

Hình 2.5: Đặc trưng đường

*3. Đặc trưng xung quanh tâm (center-surround features):*

D:\Hoc Tap\Course VIII\OpenCV\face detection\search_files\ap_20090116032402237.jpg

Hình 2.6: Đặc trưng xung quanh tâm

Lợi ích của các đặc trưng Haar-like là nó diễn đạt được tri thức về các đối tượng trong ảnh (bởi vì nó biểu diễn mối liên hệ giữa các bộ phận của đối tượng), điều mà bản thân từng điểm ảnh không diễn đạt được. Để tính giá trị các đặc trưng haar-like, ta tính sự chênh lệch giữa tổng của các pixel của các vùng đen và các vùng trắng như trong công thức sau [1][4]:

*f(x) = Tổngvùng đen(pixel) - Tổngvùng trắng(pixel)*

Như vậy ta có thể thấy rằng, để tính các giá trị của đặc trưng Haar-like, ta phải tính tổng của các vùng pixel trên ảnh. Nhưng để tính toán các giá trị của các đặc trưng Haar-like cho tất cả các vị trí trên ảnh đòi hỏi chi phí tính toán khá lớn, không đáp ứng được cho các ứng dụng đòi hỏi tính run-time. Do đó Viola và Jones đưa ra một khái niệm gọi là *Integral Image* để tính toán nhanh cho khác feature cơ bản. Sau này, Lienhart kế thừa gọi *Integral Image* là SAT – *Summed Area Table* và đưa ra thêm khái niệm RSAT – *Rotated Summed Area Table* dùng để tính toán nhanh cho các đặc trưng xoay 1 góc 45o*. Integral Image* là một mảng 2 chiều với kích thước bằng với kích thước của ảnh cần tính các đặc trưng Haar-like, với mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên (dòng-1) và bên trái (cột-1) của nó. Bắt đầu từ vị trí trên, bên trái đến vị trí dưới, phải của ảnh, việc tính toán này đơn thuần chỉ đựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh.

P(x,y)

Hình 2.7: Cách tính Integral Image của ảnh.

Sau khi đã tính được Integral Image, việc tính tổng điểm ảnh của một vùng bất kỳ nào đó trên ảnh thực hiện rất đơn giản theo cách sau: (Giả sử ta cần tính tổng điểm ảnh của vùng D):

*D = A + B + C + D – (A+B) – (A+C) + A*

Với A + B + C + D chính là giá trị tại điểm P4 trên Integral Image, tương tự như vậy A+B là giá trị tại điểm P2, A+C là giá trị tại điểm P3, và A là giá trị tại điểm P1. Vậy ta có thể viết lại biểu thức tính D ở trên như sau:

A

C

B

D

P1

P2

P3

P4

Hình 2.8: Ví dụ cách tính nhanh tổng các điểm ảnh của vùng D.

RSAT(x,y)

B

A

C

2

4

1

3

D

(a)

(b)

Hình 2.9: Tính tổng điểm ảnh của vùng D với các đặc trưng xoay 450.

Với các đặc trưng Haar-like xoay 45o *Integral Image* tại một điểm *(x, y)* được tính theo công thức:

Tổng pixel của một vùng bất kỳ trên ảnh vẫn được tính theo cách sau:

*D = A + B + C + D – (A+B) – (A+C) + A*

Như vậy tổng các điểm ảnh trong một hình chữ nhật (kể cả trường hợp xoay 45º) bất kì đều có thể được tính nhanh dựa trên *integral image* tại 4 đỉnh của nó:

*Sum (D) = .4 - 2 – 3 + 1*

***Mô hình phát hiện khuôn mặt***

Hình dưới đây là mô hình phát hiện khuôn mặt của thuật toán do Viola và Jones đưa ra [1][4]:

Ảnh

gốc

Tính các đặc trưng nhanh từ nhiều tỉ lệ

Integral Image

Các hàm Haar cơ bản

Ước lượng các đặc trưng

Điều chỉnh thông số Adaboost

Chọn đặc trưng

Số lượng lớn các đặc trưng

Cấu trúc phân tầng (*Cascade* *of classifier*)

Một tập nhỏ các đặc trưng

**Quyết định kết quả**

C1

C2

Cn

I

Y

Y

N

N

N

C

L

A

S

S

NON-CLASS

Hình 2.10: Hệ thống phát hiện khuôn mặt.

Từ ảnh gốc ban đầu ta sẽ tính được Integral Image là mảng hai chiều với phần tử (x, y) sẽ được tính bằng tổng của các phần tử (x’, y’) với x’<x và y’<y. Mục đích là để tính nhanh tổng của các giá trị mức xám của một vùng hình chữ nhật bất kỳ trên ảnh gốc. Các vùng ảnh con này sẽ được đưa qua các hàm Haar cơ bản để ước lượng đặc trưng. Kết quả ước lượng sẽ được đưa qua bộ điều chỉnh AdaBoost để loại bỏ nhanh các đặc trưng không có khả năng là đặc trưng của khuôn mặt. Chỉ có một tập nhỏ các đặc trưng mà bộ điều chỉnh AdaBoost cho là có khả năng là đặc trưng của khuôn mặt. Tập các đặc trưng này sẽ được chuyển sang cho bộ quyết định kết quả. Bộ quyết định kết quả là tập các bộ phân loại yếu. Bộ này sẽ tổng hợp kết quả là khuôn mặt nếu kết quả của các bộ phân loại yếu trả về là khuôn mặt. Mỗi bộ phân loại yếu sẽ quyết định kết quả cho một đặc trưng Haar-like, được xác định ngưỡng đủ nhỏ sao cho có thể vượt được tất cả các bộ dữ liệu mẫu trong tập dữ liệu tập huấn. Trong quá trình xác định khuôn mặt, mỗi vùng ảnh con sẽ được kiểm tra với các đặc trưng trong chuỗi các đặc trưng Haar-like. Nếu có một đặc trưng Haar-like nào cho ra kết quả là khuôn mặt thì các đặc trưng khác không cần xét nữa [1][3].

### 2.1.4. Cascade of Classifiers.

Ta thấy quá trình huấn luyện, bộ phân loại phải duyệt qua tất cả các đặc trưng của các mẫu trong tập *training*. Việc này tốn rất nhiều thời gian, tuy nhiên trong các mẫu đưa vào, không phải mẫu nào cũng thuộc loại khó nhận dạng, có những mẫu *background* rất dễ nhận ra (ta gọi đây là những mẫu *background* đơn giản). Đối với những mẫu này, ta chỉ cần xét một hay vài đặc trưng đơn giản là có thể nhận diện được chứ không cần xét tất cả các đặc trưng. Nhưng đối với các bộ phân loại thông thường thì cho dù mẫu cần nhận dạng là dễ hay khó thì nó vẫn sẽ xét tất cả các đặc trưng mà nó rút ra được trong quá trình học. Do đó, chúng tốn thời gian xử lý một cách không cần thiết [2][4].

*Cascade of Classifiers* được xây dựng chính là nhằm rút ngắn thời gian xử lý, giảm thiểu *false alarm* cho bộ phân loại. *Cascade tree* gồm nhiều *stage* (hay còn gọi là *layer*), mỗi stage của cây sẽ là một *stage classifier*. Một mẫu để được phân loại là đối tượng thì nó cần phải đi qua hết tất cả các *stages* của cây. Các *stage classifiers* ở *stage* sau được huấn luyện bằng những mẫu *negative* mà *stage classifier* trước nó nhận dạng sai, tức là nó sẽ tập trung học từ các mẫu *background* khó hơn, do đó sự kết hợp các *stage classifiers* này lại sẽ giúp bộ phân loại có *false alarm* thấp. Với cấu trúc này, những mẫu *background* dễ nhận diện sẽ bị loại ngay từ những *stages* đầu tiên, giúp đáp ứng tốt nhất đối với độ phức tạp gia tăng của các mẫu đưa vào, đồng thời giúp rút ngắn thời gian xử lý [4].

**Thuật toán *Cascade training*:**

1. Gọi:

* **F** là giá trị false alarm và **d** là độ chính xác của weak classifier ở mỗi stage
* **Ftarget**: Giá trị max false alarm.
* **P, N** là số lượng mẫu positive và negative.
* **Pi, Ni** là tập positive và negative cho bộ phân lớp ở tầng thứ **i**.
* **Fi**, **Di**: Giá trị false alarm và độ chính xác của cascade trước khi đến tầng thứ i.

1. Khởi tạo i=0; F0=1.0; D0 = 1.0
2. Lặp : While **Fi**>**Ftarget**

* i=i+1;
* Huấn luyện bộ phân loại **hi** từ tập **Pi** và **Ni** với detection rate **d** và max false alarm **f**. Thêm **hi** vào cây phân lớp.
* Dùng cây phân lớp hiện có để tính **Fi**: Duyệt qua **N** mẫu negative cho đến khi nào tìm đủ **n** mẫu mà cây phân lớp hiện có phân loại sai. (



* **N** :=



* Nếu Fi >Ftarget

**N =** { số mẫu sai ở stage hiện tại phân loại sai }.

**P** = { số mẫu positive mà stage hiện tại phân loại dúng }.

**Minh họa thuật toán *Cascade training*:**

h

h

h

1-f

1-f

1-f

1-f

stage 1

2

N-1

N

……

hitrate = hN

False alarms = fN

Input pattern classified as a non-object

Hình 2.11: Cascade of classifiers.

Minh họa sự huấn luyện của một cascade gồm ***N*** *stages*. Ở mỗi stage, weak classifier tương ứng sẽ được huấn luyện sao cho độ chính xác của nó là *h* và false alarm bằng *f* [1].

**Cascade of boosting classifiers**

*Cascade of boosting classifiers* là mô hình *cascade of classifiers* với mỗi *classifier* được xây dựng bằng *Adaboost* sử dụng *haar-like.*



I

Y

Y

Y

N

N

N

N

C

L

A

S

S

NON-CLASS

Hình 2.12: Cascade of boosting classifiers.

Viola và Jones đã sử dụng rất thành công cascade of boosting classifiers cho bài toán nhận dạng khuôn mặt. Với tập huấn luyện gồm 4196 hình khuôn mặt được đưa về ảnh grayscale kích thước 24x24 và 9500 hình background, hai ông đã xây dựng cấu trúc cascade tree gồm 38 stage với tổng cộng 6060 đặc trưng haar-like. Thực nghiệm đã cho thấy classifier ở stage đầu tiên sử dụng 2 đặc trưng và loại được khoảng 50% mẫu background (không phải khuôn mặt) và có độ chính xác là d = 100%. Classifier ở stage thứ 2 sử dụng 10 đặc trưng loại được 80% mẫu background với độ chính xác vẫn là 100%. Hệ thống này được so sánh với hệ thống của Rowley-Baluja-Kanade (sử dụng mạng neural), Schneiderman-Kanade (sử dụng phương pháp thống kê), và cho thấy tỉ lệ nhận dạng là ngang nhau, trong khi hệ thống của Viola và Jones chạy nhanh hơn đến 15 lần so với hệ thống của Rowley-Baluja-Kanade và nhanh hơn 600 lần hệ thống của Schneiderman-Kanade [1].

Lý do mà cấu trúc *cascade* đạt tốc độ nhận dạng nhanh chính là nhờ nó sớm loại bỏ được các mẫu *background* đơn giản (thường có số lượng lớn hơn nhiều so với các mẫu chứa *object* – các mẫu chưa khuôn mặt cần tiến hành nhận dạng). Bên cạnh đó, hệ thống của Viola và Jones cũng đạt được độ chính xác khá cao nhờ vào thuật toán *cascade training*, các bộ nhận dạng được huấn luyện bằng *AdaBoost* với đặc trưng Haar-like mô tả tốt thông tin đối tượng, cùng với cách *Integral Image* tính nhanh các đặc trưng, không làm giảm tốc độ nhận dạng của hệ thống. Như vậy, mô hình *Cascade of Boosted Classifiers* thật sự là một cách tiếp cận tốt cả về tốc độ lẫn khả năng nhận dạng, rất phù hợp với bài toán nhận dạng khuôn mặt [1].

### 2.1.5. Nhận xét.

**Ưu điểm:**

* Phương pháp cho độ chính xác tương đối cao, tốc độ dò tìm khuôn mặt khác nhanh, thích hợp để dò tìm khuôn mặt trong thời gian thực, trong video.
* Thích hợp huấn luyện dữ liệu bị nhiễu.
* Phương pháp trích chọn đặc trưng khá nhanh.

**Khuyết điểm:**

* Thuật toán huấn luyện khuôn mặt thực hiện chậm do có nhiều bộ phân loại yếu.
* Chỉ dò tìm được các khuôn mặt nhìn thằng và góc quay nhỏ.
* Không ít các tính chất của AdaBoost mang tính chất nhận định và chưa được chứng minh chặt chẽ.

## 2.2. Bài toán nhận diện khuôn mặt.

### 2.2.1. Một số phương pháp nhận dạng khuôn mặt.

Phương pháp nhận dạng hiện nay có 2 loại:

* Nhận dạng dựa trên các đặc trưng của các phần tử trên khuôn mặt (Feature based face recognition).
* Nhận dạng dựa trên xét tổng thể khuôn mặt (Appearance based face recognition).

Ngoài ra còn có một số loại nhận dạng sử dụng mô hình về khuôn mặt, một số phương pháp được dùng cho loại này:

* Nhận dạng 2D: Elastic Bunch Graph, Active Appearance Model.
* Nhận dạng 3D: 3D Morphable Model.
* Nhận dạng sử dụng mạng noron.
* Nhận dạng bằng phương pháp máy vecto hỗ trợ.
* Nhận dạng bằng mô hình Makov ẩn.
* Nhận dạng bằng phương pháp PCAEigenFaces.
* Nhận dạng bằng phương pháp FisherFaces.
* Nhận dạng bằng phương pháp Local Binary Patterns Histograms.

**Nhận dạng dựa trên mối quan hệ giữa các phần tử:**

Đây là phương pháp nhận dạng khuôn mặt dựa trên việc xác định các đặc trưng hình học của các chi tiết trên một khuôn mặt (như vị trí, diện tích, hình dạng của mắt, mũi, miệng,…), và mối quan hệ giữa chúng (như khoảng cách của hai mắt, khoảng cách của hai lông mày,…).

Ưu điểm của phương pháp này là nó gần với cách mà con người sử dụng để nhận biết khuôn mặt. Hơn nữa với việc xác định đặc tính và các mối quan hệ, phương pháp này có thể cho kết quả tốt trong các trường hợp ảnh có nhiều nhiễu như bị nghiêng, bị xoay hoặc ánh sáng thay đổi.

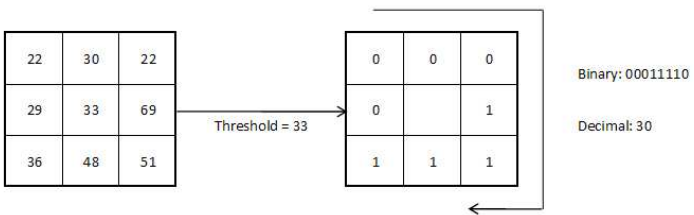
Nhược điểm của phương pháp này là cài đặt thuật toán phức tạp do việc xác định mối quan hệ giữa các đặc tính đòi hỏi các thuật toán phức tạp. Mặt khác, với các ảnh kích thước bé thì cácđặc tính sẽ khó phân biệt.

**Nhận dạng dựa trên xét toàn diện khuôn mặt:**

Nội dung chính của hướng tiếp cận này là xem mỗi ảnh có kích thước RxC là một vector trong không gian RxC chiều. Ta sẽ xây dựng một không gian mới có chiều nhỏ hơn sao cho khi biểu diễn trong không gian đó các đặc điểm chính của một khuôn mặt không bị mất đi. Trong không gian đó, các ảnh của cùng một người sẽ được tập trung lại thành một nhóm gần nhau và cách xa các nhóm khác.

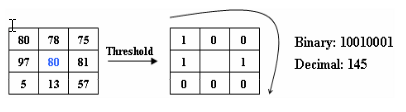
### 2.2.2. Nhận dạng bằng phương pháp Local Binary Patterns Histograms.

**LBP (Local Binary Pattern)** là mẫu nhị phân địa phương được Ojala trình bày vào năm 1996 như là một đơn vị đo độ tương phản cục bộ của ảnh. Giá trị LBP được xác định bằng cách nhân các giá trị ngưỡng với trọng số ứng với mỗi điểm ảnh sau đó cộng tổng lại. Hình dưới minh họa cách tính độ tương phản trực giao (C) là hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng. Phân phối hai chiều của mã LBP và độ tương phản cục bộ được lấy làm đặc trưng gọi là LBP/C [2].



Hình 2.13: Ví dụ về LBP và độ tương phản cục bộ C.

LBP là một toán tử kernel 3×3, nó tổng quát hóa cấu trúc không gian cục bộ của một ảnh. Ojala và các đồng nghiệp đã giới thiệu phương pháp LBP và chỉ ra khả năng phân tách cao của chúng cho sự phân lớp vân. Tại một vị trí pixel (xc, yc) cho trước, LBP được định nghĩa như một chuỗi nhị phân có trật tự dựa trên sự so sánh giá trị độ xám của pixel trung tâm (xc, yc) và 8 pixel lân cận của nó. Như vậy mỗi pixel sẽ được biểu diễn bởi một chuỗi nhị phân, giá trị thập phân của chuỗi nhị phân này chính là giá trị của pixel trung tâm trong sự biểu diễn bởi toán tử LBP [2].



Hình 2.14: Ví dụ tính toán LBP.

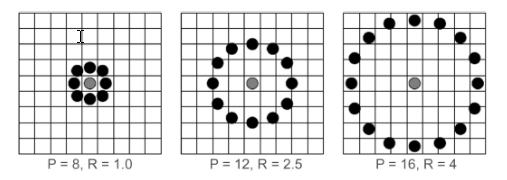
Giá trị thập phân của của chuỗi LBP có thể được biểu diễn như sau:

LBP(xc,yc)=

Với gc là giá trị độ xám của pixel trung tâm (xc, yc), gn là giá trị độ xám của pixel thứ n trong lân cận 8 của pixel trung tâm. Hàm sn được định nghĩa như sau:

s(x)=

Bằng cách định nghĩa này, toán tử LBP bất biến đối với bất kỳ sự biến đổi độ sáng đều và bảo toàn trật tự mật độ các pixel trong một lân cận cục bộ.Sau đó, Ojala và các đồng nghiệp đã mở rộng toán tử LBP cũ đến một lân cận tròn với các bán kính khác nhau. LBPP,R kí hiệu sự xem xét đến P pixels lân cận trên một vòng tròn có bán kính R. Hình 2.12 minh họa toán tử LBPP,R [1][2].



Hình 2.15: Minh họa toán tử LBP mở rộng với các giá trị P và R khác nhau.

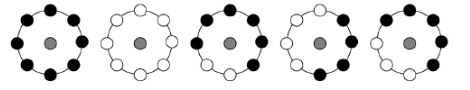
Nếu tọa độ của pixel tâm là (xc, yc) thì tọa độ của P pixel lân cận trên đường tròn tâm (xc, yc) bán kính R (tính theo đường tròn lượng giác) là:

xp= xc + Rcos(2πp/P)

yp= yc + Rsin(2πp/P), p={0,1,….,P-1}

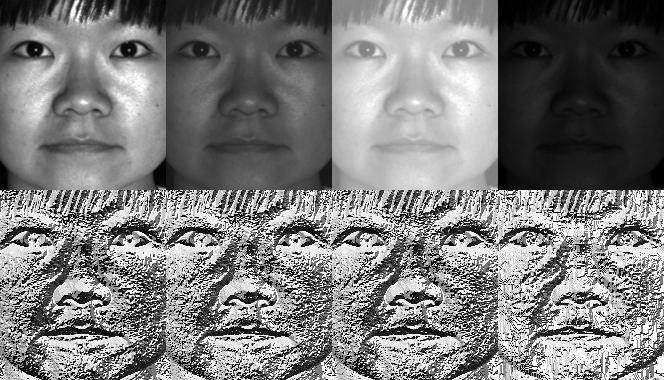
Để xác định giá trị giá trị của các pixel lân cận không nằm trong tâm của một pixel, sẽ sử dụng phép nội suy song tuyến tính [3].

Các tác giả đã kết luận rằng: hầu hết các thông tin vân (texture) trong ảnh được chứa trong một tập con của các mẫu LBP. Các mẫu này được gọi là các mẫu đồng nhất (uniform patterns). Các mẫu đồng nhất chứa tối đa hai sự chuyển đổi bit từ 0 đến 1 hoặc từ 1 đến 0. 11111111, 00000110 hoặc 10000111 là ví dụ cho các mẫu đồng nhất. Toán tử LBPP,R đồng nhất được kí hiệu là . Với chuỗi LBP có chiều dài P thì số mẫu có tối đa hai sự chuyển đổi (mẫu đồng nhất là) P(P-1). Có hai mẫu không có sự chuyển đổi nào là mẫu toàn 0 hoặc 1. Việc sử dụng mẫu LBP đồng nhất có hai lợi điểm quan trọng. Thứ nhất là tiết kiệm bộ nhớ, vì trường hợp LBP tổng quát chúng ta có 2P mẫu có thể, nhưng nếu chỉ xét mẫu đồng nhất thì chúng ta có tối đa là P(P+1) + 2 nếu chúng ta sử dụng toán tử . Lợi điểm thứ hai là LBPu2 chỉ phát hiện những mẫu vân cục bộ quan trọng như các điểm, các điểm cuối đường thẳng, biên cạnh và các góc. Hình 2.13 minh họa một số mẫu vân quan trọng có thể được phát hiện bởi LBPu2 [1].



Hình 2.16: Điểm chấm, điểm chấm nhạt, điểm cuối đường thẳng, biên cạnh, góc được phát hiện bởi LBPu2.

Bởi vì khả năng phân tách và chi phí tính toán thấp, LBP trở nên rất phổ biến trong nhận dạng mẫu LBP đã được áp dụng cho phát hiện khuôn mặt, nhận dạng khuôn mặt, xác thực khuôn mặt, truy vấn ảnh.

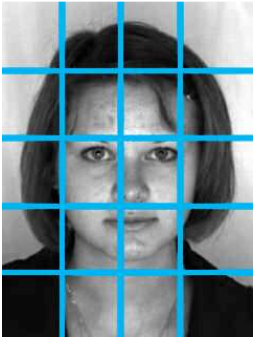


Hình 2.17: Biến đổi ảnh bằng LBP

**Mô tả khuôn mặt với LBP [2][3]:**

Biểu đồ LBP có thể được xây dựng bằng cách tính toán mã LBP cho mỗi điểm ảnh. Mỗi mã LBP đồng dạng sẽ tương ứng để tách cột của biểu đồ. Mỗi một cột trong biểu đồ tương ứng với một tập mã LBP, những mã đồng dạng.

Các ảnh khuôn mặt có thể được xem như là một cấu thành của siêu thành phần được mô tả một cách hiệu quả bởi các biểu đồ LBP. Một biểu đồ LBP tính toán mã trên toàn bộ khuôn mặt chỉ xuất hiện ở trong siêu thành phần mà không tính đến vị trí của chúng. Để xem xét thông tin về hình dạng khuôn mặt, ảnh khuôn mặt được chia thành các tiểu vùng nhỏ để trích suất các biểu đồ LBP.



Hình 2.18: Ảnh khuôn mặt được chia ra thành các miền con

Các đặc tính LBP được rút ra từ các tiểu vùng được đưa vào một biểu đồ duy nhất – biểu đồ đặc trưng tăng cường.

Các biểu đồ đặc trưng mở rộng đại diện cho kết cấu địa phương và toàn cục của hình dạng khuôn mặt. Một số vùng khuôn mặt bị mất thông tin có thể là các vùng quan trọng, nó có thể được gán trọng số cho từng vùng, tùy thuộc vào tầm quan trọng của nó trong việc nhận diện khuôn mặt.

Một vài thuộc tính có thể được tối ưu hóa để khai thác các tính năng đặc biệt. Như số lượng tiểu vùng của ảnh, toán tử LBP.

**Nhận xét phương pháp Local Binary Patterns Histograms:**

* Khả năng xử lí tương đối chậm.
* Có độ chính xác cao.
* Nhận diện tốt đối với ảnh đầu vào có độ sáng kém.

LBPH tuy tốc độ có thể chậm hơn các thuật toán khác, nhưng vì có thể nhận dạng tốt với ảnh có độ sáng kém, trong khi camera trên máy tính không thể tốt bằng camera giám sát.

## 2.3. Kết chương.

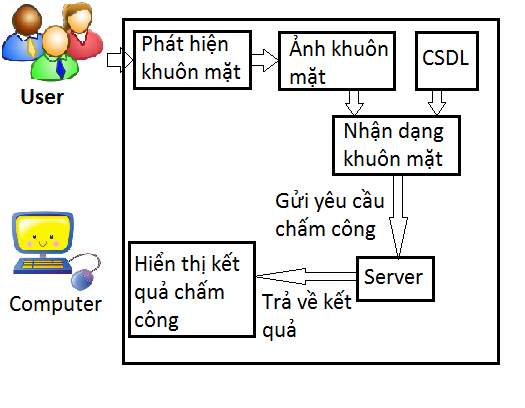
Trong chương này, người viết đồ án đã trình bày về cơ sở lý thuyết của bài toán phát hiện khuôn mặt sử dụng thuật toán ADABOOST với các đặc trưng Haar-Like và sử dụng Cascade of classifier để tăng tốc độ và giảm khả năng nhận dạng sai.Bên cạnh đó người viết đồ án cũng trình bày về việc dùng thuật toán Local Binary Patterns Histogramsđể nhận diện khuôn mặt trong một cơ sở dữ liệu.

Đến chương tiếp theo, người viết đồ án sẽ trình bày về giải pháp thiết kế và xây dựng ứng dụng thử nghiệm chấm công bằng nhận diện khuôn mặt đã đưa ra ở trên.

# Chương 3: Thiết kế và xây dựng ứng dụng thử nghiệm chấm công bằng nhận diện khuôn mặt

|  |
| --- |
| Nội dung chính được trình bày ở chương này:   * Phát biểu bài toán chấm công bằng nhận diện khuôn mặt. * Giới thiệu về OpenCV. * Giới thiệu về Python và thư viện OpenCV cho Python. * Phân tích thiết kế hệ thống. * Kết chương. |

## 3.1. Phát biểu bài toán chấm công bằng nhận diện khuôn mặt.



Hình 3.1 : Hệ thống chấm công bằng khuôn mặt

Chương trình được xây dựng với hai phần :

* Ở phía Client : Gồm các chức năng :
  + Start Capture : Bắt đầu bật camera, phục vụ việc thêm người dùng vào cơ sở dữ liệu hoặc nhận diện người dùng
  + Capture Face : Chụp ảnh người dùng, để thêm vào cơ sở dữ liệu
  + Training Face : Huấn luyện ảnh người dùng
  + Recognize Face : Nhận diện người dùng và gửi yêu cầu chấm công lên Server
  + End Capture : Tắt camera
  + Quit : Tắt ứng dụng
* Ở phía Server : Gồm các chức năng :
  + API nhận yêu cầu chấm công
  + API truy vấn tới CSDL

Lựa chọn công cụ, nền tảng cài đặt phần mềm :

* Tác giả sử dụng bộ thư viện mã nguồn mở OpenCV để cài đặt phần mềm. OpenCV có những thư viện cấp cao cho phép phát hiện và nhận dạng khuôn mặt. Tác giả sử dụng các API của OpenCV đồng thời tìm hiểu phương thức lưu trữ, xử lý thông tin của OpenCV. Qua quá trình thực nghiệm, tác giả sẽ đánh giá được hiệu quả nhận dạng khuôn mặt với các thuật toàn Adaboost, PBPH.
* Tác giả chọn ngôn ngữ Python để demo ứng dụng. Hiện tại OpenCV – bộ thư viện thị giác mã nguồn mở cũng hỗ trợ Python nên việc sử dụng OpenCV cho Python là 1 lựa chọn khá tốt.

## 3.2. Giới thiệu về OpenCV.

OpenCV là thư viện thị giác máy tính mã nguồn mở, được cấp phép từ năm 2000 bởi BSD. OpenCV được xây dựng nhằm mục đích cung cấp những công cụ cần thiết để giải quyết các vấn đề thị giác máy tính. Bao gồm các chức năng xử lý hình ảnh ở mức độ thấp và các thuật toán cao cấp như nhận diện khuôn mặt, nhận diện người đi bộ. Thư viện đã được tải về hơn 5 triệu lần. Các trường đại học hàng đầu và hơn 100 công ty Foutine và nhiều thành viên khác đã phát triển và duy trì hơn 2500 thuật toán có trong thư viện. OpenCV được viết bằng C, C++, nhưng giao diện ứng dụng cũng bao gồm wrapper cho Java, MATLAB và Python. OpenCV hiện đang hỗ trợ cho Linux, Mac OS X, Android và các nền tảng iOS.

Cấu trúc tổng quan của OpenCV bao gồm 5 phần chính. 4 trong 5 phần đó được chỉ ra trong hình vẽ dưới.

**CV**

Các hàm về xử lý ảnh và giải thuật về thị giác máy tính

**CXCORE**

Các cấu trúc dữ liệu cơ bản, cấu trúc XML, các hàm về đồ họa …

**MLL**

Các thuật toán học máy, bao gồm các bộ phân cụm, phân loại thống kê

**HighGUI**

Các hàm và thủ tục làm việc với file ảnh và file video

Hình 3.2: Cấu trúc cơ bản của OpenCV

Phần CV bao gồm các thư viện cơ bản về xử lý ảnh và các giải thuật về thị giác máy tính. ML là bộ thư viện về các thuật toán học máy, bao gồm rất nhiều bộ phân cụm và phân loại thống kê. HighGUI chứa đựng những thủ tục vào ra, các chức năng về lưu trữ cũng như đọc các file ảnh và video. Phần thứ 4, Cxcore chứa đựng các cấu trúc dữ liệu cơ bản ( ví dụ như cấu trúc XML, các cây dữ liệu …). Phần cuối cùng là CvAux, phần này bao gồm các thư viện cho việc phát hiện, theo dõi và nhận dạng đối tượng (khuôn mặt, mắt …).

## 3.3. Giới thiệu về Python và thư viện OpenCV cho Python.

### 3.3.1. Giới thiệu về Python

Python là một ngôn ngữ mục đích chung. Nó có nghĩa là chúng ta có thể sử dụng nó trong bất kỳ lĩnh vực như phát triển web, phát triển phần mềm… Python là một ngôn ngữ cấp cao, giải thích, ngôn ngữ tương tác và hướng đối tượng.

Vì Python là một ngôn ngữ mục đích chung nên bạn có thể làm tất cả mọi thứ từ phát triển trang web và chơi game để robot và điều khiển tàu vũ trụ. Một số ví dụ là lập trình hệ thống, ảnh minh hoạ, kịch bản internet, hội nhập thành phần, lập trình cơ sở dữ liệu, nguyên mẫu nhanh chóng, số và lập trình khoa học, Gaming, hình ảnh, khai thác dữ liệu, Robots... Và trong đồ án này, tác giả sử dụng Python để xử lý nhận diện khuôn mặt.

Python là một ngôn ngữ lập trình thông dịch do Guido van Rossum tạo ra năm 1990. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

### 3.3.2 Điểm mạnh

* Dễ dàng để tìm hiểu, đọc và duy trì: Cú pháp của Python là đơn giản. Nó rất dễ dàng để tìm hiểu và đọc. Mã này là có thể nhìn thấy rất nhiều vì vậy nó rất dễ dàng để duy trì.
* Một tiêu chuẩn thư viện mạnh mẽ là và ngôn ngữ phát triển hệ thống (chẳng hạn như C, C ++, và Java).
* Chế độ tương tác: Python đã hỗ trợ cho một chế độ tương tác cho phép thử nghiệm tương tác và gỡ lỗi các đoạn mã.
* Portable: Việc thực hiện tiêu chuẩn của Python được viết trong di ANSI C, và biên dịch và chạy trên hầu như tất cả các nền tảng lớn hiện đang sử dụng.
* Extendable: Bạn có thể thêm các module ở mức độ thấp để các thông dịch Python. Những module này cho phép lập trình để thêm vào hoặc tùy chỉnh các công cụ của họ có hiệu quả hơn.
* Cơ sở dữ liệu: Python cung cấp giao diện cho tất cả các cơ sở dữ liệu thương mại lớn.
* Khả năng mở rộng: chương trình rất phức tạp có thể được xây dựng trong python.
* Tốc độ: nhanh hơn so với PHP.

### 3.3.3. Điểm yếu

* Không nhanh hơn so với C, C ++ và Java.
* Python không có “switch” và do…while
* Không có private, protected và public

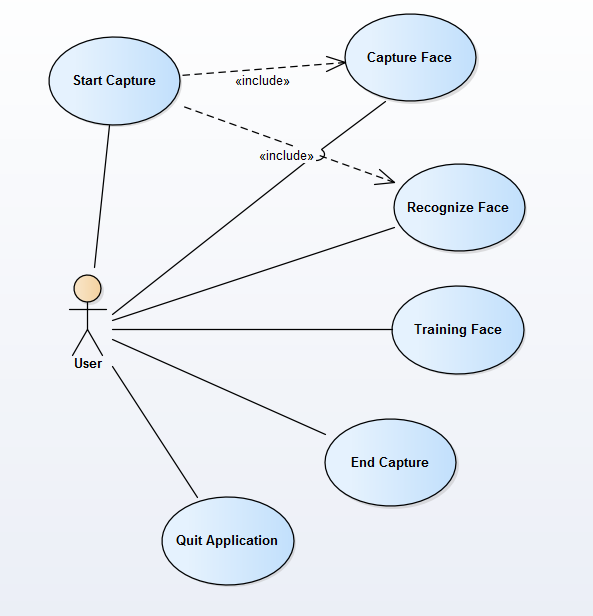
### 3.3.4. Thư viện OpenCV cho Python

OpenCV được viết bằng C, C++, nhưng giao diện ứng dụng cũng bao gồm wrapper cho Java, MATLAB và Python. Vì có wrapper cho Python nên Python có thể tận dụng rất tốt điều này để sử dụng OpenCV cho ứng dụng của mình. Với các điểm mạnh được nêu lên ở trên, cùng với thư viện hỗ trợ cho Python (có thể tìm bằng cách download OpenCV tại trang chủ, rồi vào phần opencv/build/python để lấy được thư viện OpenCV cho Python), tác giả đã quyết định chọn Python làm ngôn ngữ kết hợp với OpenCV để xây dựng đồ án này.

## 3.4. Phân tích thiết kế hệ thống.

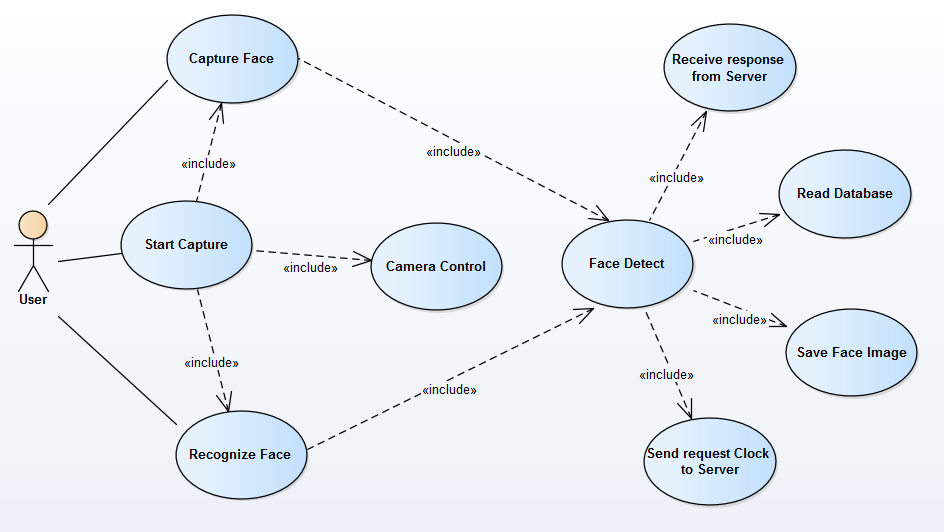
Dưới đây, báo cáo trình bày về quá trình phân tích thiết kế phần mềm chấm công bằng khuôn mặt. Trong đó trình bày các ý chính về huấn luyện hệ thống nhận dạng khuôn mặt, nhận dạng khuôn mặt.

### 3.4.1. Biểu đồ Usercase của hệ thống



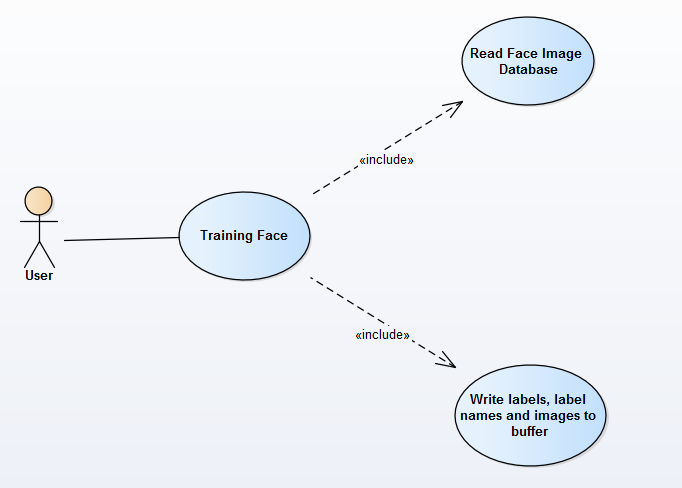
Hình 3.3: Biểu đồ Usercase tổng quát

Trong biểu đồ Usercase tổng quát này người dùng thấy được các usercase chính của hệ thống là: Start Capture (Bắt đầu bật camera), Capture Face (Chụp ảnh người dùng để lưu vào CSDL), Recognize Face (Nhận diện khuôn mặt của người dùng đứng trước camera), Training Face (Huấn luyện ảnh khuôn mặt của người dùng), End Capture(Tắt Camera), Quit Application(Tắt ứng dụng). Hai Usecase Capture Face và Recognize Face được include trong Usecase Start Capture vì chỉ khi bật camera thì 2 Usecase này mới được hoạt động (do 2 Usecase này cần sử dụng camera). Chi tiết được thể hiện ở các Usecase phân rã mức 2.



Hình 3.4: Biểu đồ Usecase phân rã mức 2 Capture Face

Biểu đồ Usecase phân rã mức 2 này thể hiện rõ hơn 3 Usecase Start Capture, Capture Face và Recognize Face. Usecase Start Capture bao gồm 2 Usecase Capture Face và Recognize Face, lý do là 2 Usecase này cần sử dụng Camera, mà Camera được quản lý bởi Usecase Camera Control, thuộc Usecase Start Capture. Vì thế Usecase Start Capture cần được chạy trước khi 2 Usecase Capture Face và Recognize Face chạy, nếu không sẽ báo lỗi. 2 Usecase Capture Face và Recognize Face đều bao gồm Usecase Face Detect (Phát hiện khuôn mặt). Ở Usecase Capture Face, ta cần chụp ảnh người dùng để lưu vào cơ sở dữ liệu, vì thế cần phát hiện xem trước camera có khuôn mặt nào không, nếu có thì mới lưu ảnh đó vào cơ sở dữ liệu, còn nếu không thì đưa ra thông báo: “Không phát hiện ra khuôn mặt nào trước camera”. Ở Usecase Recognize Face cũng vậy, trước khi nhận diện xem khuôn mặt người dùng có ở trong cơ sở dữ liệu không thì cần phát hiện xem trước camera có khuôn mặt nào không, nếu có thì mới so sánh với các khuôn mặt có trong cơ sở dữ liệu, còn nếu không thì đưa ra thông báo: “Không phát hiện ra khuôn mặt nào trước camera”. Sau khi phát hiện khuôn mặt trước camera có trong cơ sở dữ liệu thì gửi yêu cầu lên server để chấm công cho người dùng đó. Nếu Server hoạt động ổn định, nó sẽ trả về kết quả chấm công thành công cho người dùng đó.

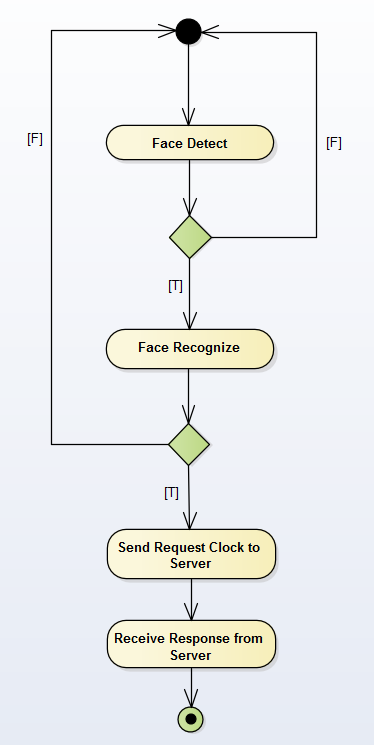


Hình 3.5: Biểu đồ Usecase phân rã mức 2 Training Face

Biểu đồ Usecase phân rã mức 2 Training Face bao gồm 2 Usecase Read Face Image Database(Đọc cơ sở dữ liệu ảnh khuôn mặt của người dùng) và Usecase Write labels, label names and images to buffer(Ghi nhãn, tên của nhãn và ảnh vào buffer, phục vụ việc nhận diện khuôn mặt(Recognize Face)). Để huấn luyện việc nhận diện khuôn mặt thì trước hết ứng dụng cần đọc cơ sở dữ liệu khuôn mặt của những người dùng đã chụp ảnh và lưu vào cơ sở dữ liệu. Sau khi đọc cơ sở dữ liệu thì ứng dụng sẽ phân loại ảnh và lấy được nhãn của ảnh, tên của ảnh và lưu vào buffer, phục vụ việc nhận diện khuôn mặt sau này.

### 3.4.2. Lưu đồ thuật toán chấm công bằng khuôn mặt.

Khi bắt đầu chạy ứng dụng, người dùng sẽ nhấn vào nút Start Capture. Sau đó nếu muốn chấm công bằng khuôn mặt mình, người dùng sẽ tiếp tục nhấn vào nút Recognize Face. Ứng dụng sẽ gửi 1 request lên server để yêu cấu chấm công cho người dùng này, nếu thành công server sẽ trả về thông báo chấm công thành công, còn nếu không server sẽ trả về thông báo chấm công thất bại. Sau đây là lưu đồ thuật toán mô tả quy trình chấm công bằng khuôn mặt:



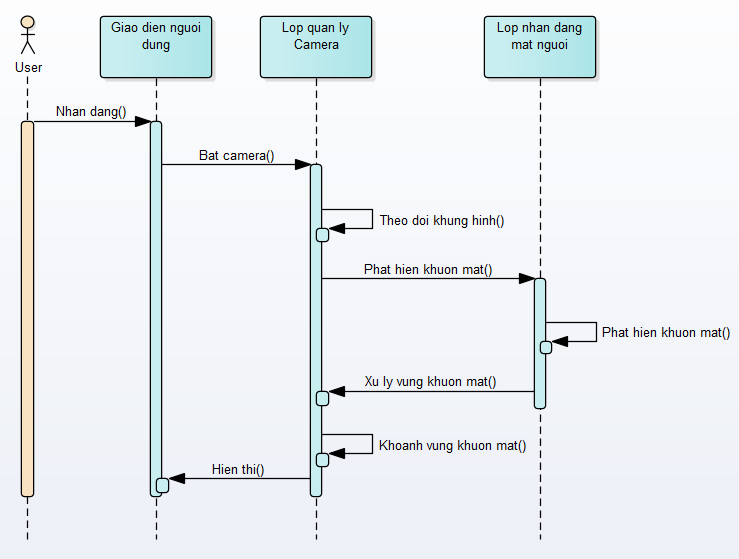
Hình 3.6: Lưu đồ thuật toán chấm công bằng khuôn mặt

Đầu vào của chức năng này là kết quả của huấn luyện khuôn mặt người dùng. Nếu khuôn mặt đã được gán nhãn thì ứng dụng sẽ gửi yêu cầu chấm công lên server và nhận lại kết quả trả về từ server.

### 3.4.3. Phát hiện khuôn mặt.

Chương trình sử dụng thuật toán adaboost, với tập huấn luyện đã được trainning trong OpenCV. OpenCV có tập huấn luyện tốt để huấn luyện cho phát hiện mặt, mắt,... Tập huấn luyện được lưu trong các file XML trong opencv/data/haarcascades/. Các file XML này là kết quả huấn luyện nhận dạng từ vài trăm ảnh khẳng định và phủ định của OpenCV, khi cần phát hiện khuôn mặt chỉ cần lấy các kết quả này, không cần trainning lại. Đề tài sử dụng file haarcascade\_frontalface\_default.xml là kết quả huấn luyện nhận dạng khuôn mặt mặc định của OpenCV.

Đầu vào của chức năng này là một Frame của camera. Sau khi có được frame, frame sẽ được xám hóa, sau đó cho các đặc trưng haar-like chạy khắp bức ảnh, những khu vực so sánh được cho là giống với nhiều đặc trưng haar-like nhất sẽ được đánh dấu lại, và chính là vùng khuôn mặt.



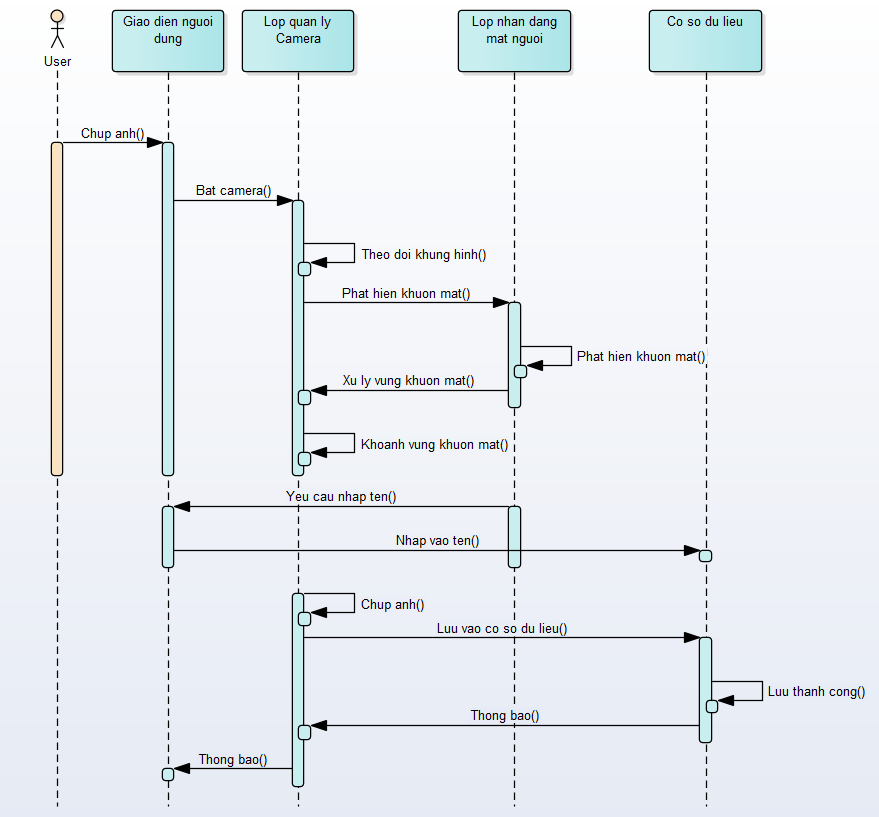
Hình 3.7: Chức năng phát hiện khuôn mặt

### 3.4.4. Huấn luyện nhận dạng khuôn mặt

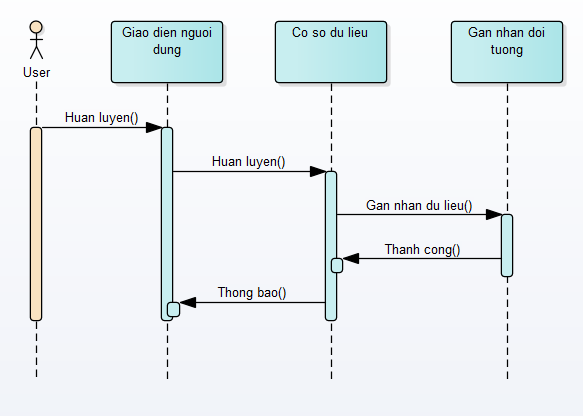
Để có thể huấn luyện hệ thống nhận ra một người nào đó, phần mềm yêu cầu người dùng phải chụp một vài bức ảnh để lưu vào cơ sở dữ liệu và được gán nhãn là email mà người dùng nhập vào. Tuy nhiên, hệ thống không cho phép chụp ảnh bất cứ khi nào mà chỉ cho phép chụp ảnh khi hệ thống đã phát hiện ra có khuôn mặt trong khung hình. Điều này sẽ giúp tiết kiệm thời gian để phân tích trích chọn khuôn mặt từ một bức ảnh đã chụp ngẫu nhiên.

Theo kết quả thực nghiệm các thuật toán nhận diện khuôn mặt với OpenCV [5] thì để có thể nhận dạng được khuôn mặt cho kết quả chấp nhận được cần có khoảng 8 ảnh/mẫu. Vì vậy, hệ thống cho phép người dùng chụp nhiều bức ảnh.

Sau khi đã có đủ thông tin về cơ sở dữ liệu, người dùng sẽ yêu cầu huấn luyện. Hệ thống sẽ trích chọn đặc trưng trong các bức ảnh, lưu trữ lại thông tin để dùng cho việc phát hiện khuôn mặt về sau.



Hình 3.8: Chụp ảnh khuôn mặt

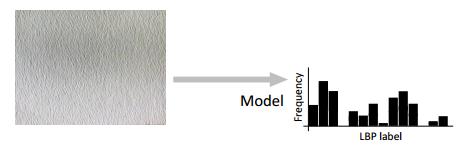


Hình 3.9: Huấn luyện nhận dạng khuôn mặt

Trong phần mềm, các ảnh huấn luyện được lưu trong thư mục /trainning\_images/ cùng mã nguồn dự án. Các ảnh được lưu với quy tắc như sau : nhãn của ảnh – tên nhãn của ảnh – số thứ tự, do một nhãn có nhiều ảnh dùng để huấn luyện. Tên nhãn của ảnh sẽ là email của người dùng, dùng để gửi request lên server. Ảnh được lưu dưới định dạng png.

Để huấn luyện nhận dạng khuôn mặt, mỗi pixel của khuôn mặt được lưu trong cơ sở dữ liệu sẽ được tính LBP. Sau đó biểu đồ Histogram sẽ được xây dựng cho từng mẫu.

Các biểu đồ - model này sẽ được dùng cho việc nhận dạng khuôn mặt, biểu đồ của ảnh cần nhận dạng sẽ được so sánh với các biểu đồ của mẫu. Vì vậy, sau khi người dùng đã chụp ảnh vùng khuôn mặt và gán nhãn nó, chương trình sẽ tính LBP và LBP Histogram cho từng ảnh.



Hình 3.10 : Thiết lập LBP Histogram cho các mẫu

### 3.4.5. Nhận dạng khuôn mặt

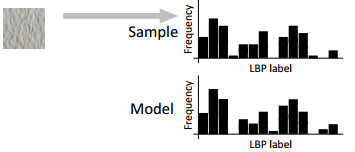
Ứng dụng sẽ bắt đầu chạy bằng cách hiện thị 1 Control Panel, nơi người dùng có thể lựa chọn giữa các tính năng : Start Capture, Capture Face, Recognize Face, Trainning Face, End Capture, Quit Application. Với 2 tính năng Capture Face và Recognize Face thì đều yêu cầu nhấn nút Start Capture trước. Với trường hợp đã có người dùng lưu hình khuôn mặt mình vào hệ thống, giờ muốn nhận dạng khuôn mặt mình để chấm công thì người dùng sẽ phải nhấn nút Start Capture, sau đó nhấn nút Recognize Face, nếu ứng dụng nhận diện thành công khuôn mặt của người dùng, ứng dụng sẽ gửi 1 request lên Server để chấm công cho người dùng đó. Sau đó Server sẽ check xem request này có phải từ ứng dụng gửi lên hay không, nếu đúng thì trả về thông báo chấm công thành công cho người dùng đó. Nếu ứng dụng nhận diện thất bại (do người dùng chưa được thêm vào trong cơ sở dữ liệu, hoặc ứng dụng nhận diện không chính xác), sẽ không có request nào được gửi lên Server. Người dùng có thể nhấn lại nút Recognize Face để ứng dụng nhận diện lại.

OpenCV 2.4 trở lên đi kèm với một lớp FaceRecognizer cho phép nhận dạng khuôn mặt, với các thuật toán có sẵn :

* Eigenfaces (createEigenFaceRecognizer ())
* Fisherfaces (createFisherFaceRecognizer ())
* Local Patterns Binary Histograms (createLBPHFaceRecognizer ())

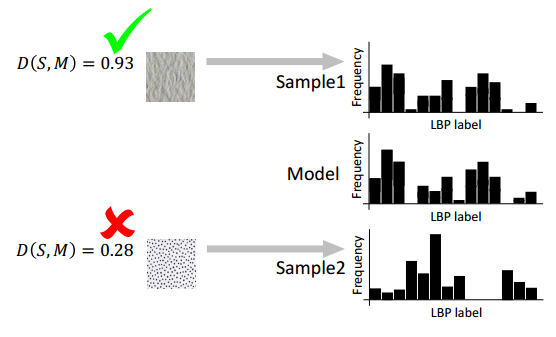
Để nhận dạng khuôn mặt với LBPH, ảnh cần nhận dạng được tính LBP và dựng mô hình LBPH, sau đó mô hình này được dùng để tính khoảng cách với mô hình của các mẫu trong tập huấn luyện. Công thức để tính khoảng cách được sử dụng là:





Hình 3.11: So sánh mô hình ảnh nhận dạng (Model) với ảnh mẫu (Sample)

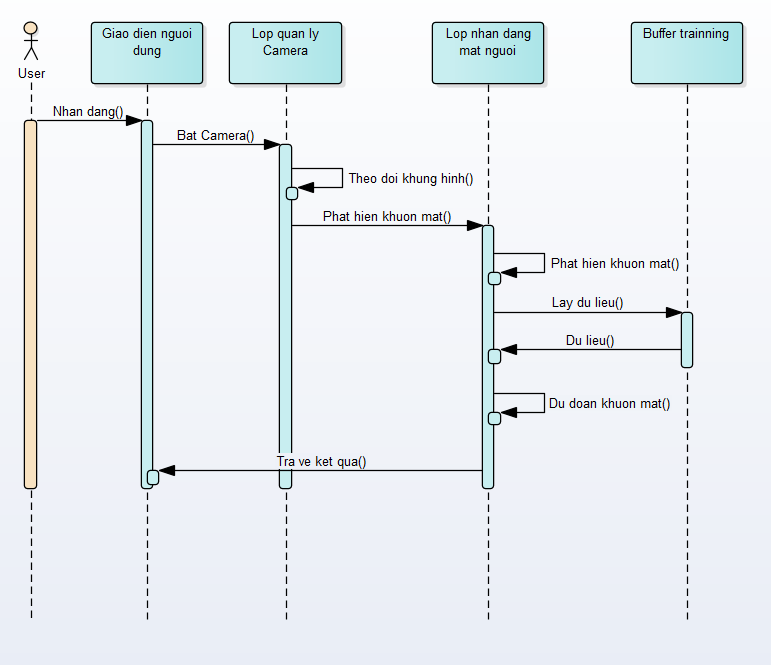
Mẫu nào có khoảng cách lớn nhất tới mô hình ảnh nhận dạng thì nhãn của mẫu đó được chọn để gán cho khuôn mặt cần nhận dạng:



Hình 3.12: nhận dạng đối tượng bằng so sánh khoảng cách

Do đó đầu vào của chức năng này là mô hình LBPH của các ảnh mẫu, và mô hình LBPH của ảnh cần nhận dạng sau khi đã được tính toán. Kết quả là ảnh đã được phân lớp.

Trong chương trình, khi người dùng chọn chức năng nhận dạng khuôn mặt, hệ thống sẽ tiến hành dò khuôn mặt (phát hiện khuôn mặt trước), sau đó, gọi chức năng predict() – dự đoán khuôn mặt, chức năng này sẽ lấy kết quả của tập huấn luyện so khớp với kết quả mô hình của ảnh cần gán nhãn, lấy maximum và đưa ra nhãn cho ảnh.



Hình 3.13: Nhận dạng khuôn mặt

## 3.5. Kết chương.

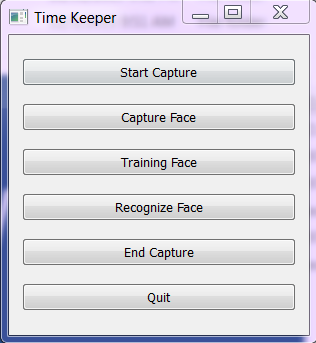
Trong chương này, người viết đồ án đã trình bày về phương hướng thiết kế và xây dựng ứng dụng thử nghiệm chấm công bằng nhận diện khuôn mặt sử dụng thuật toán được trình bày ở chương 2.

Trong chương tiếp theo, người viết đồ án trình bày về cách cài đặt hệ thống đồng thời đánh giá độ chính xác của hệ thống đó.

# Chương 4: Cài đặt mô hình và đánh giá kết quả

|  |
| --- |
| Nội dung chính được trình bày trong chương này :   * Chương trình nhận diện khuôn mặt. * Kết quả phát hiện khuôn mặt. * Nhận diện khuôn mặt. * Chấm công bằng nhận diện khuôn mặt. * Đánh giá độ tin cậy của chương trình. * Kết luận. * Kết chương. |

## 4.1. Chương trình nhận diện khuôn mặt.



Hình 4.1 : Giao diện hệ thống

Chương trình chấm công bằng khuôn mặt gồm có ba chức năng chính như sau:

* Chức năng Capture Face cho phép thêm người dùng vào hệ thống: Người dùng dùng chức năng này để thêm người được phép chấm công từ hệ thống.
* Chức năng Training Face cho phép huấn luyện các khuôn mặt: Người dùng dùng chức năng này để huấn luyện các khuôn mặt đã được thêm vào hệ thống từ trước.
* Chức năng Recognize Face cho phép nhận diện khuôn mặt : Người dùng dùng chức năng này để hệ thống nhận diện khuôn mặt của mình, từ đó hệ thống sẽ gửi yêu cầu chấm công lên server.

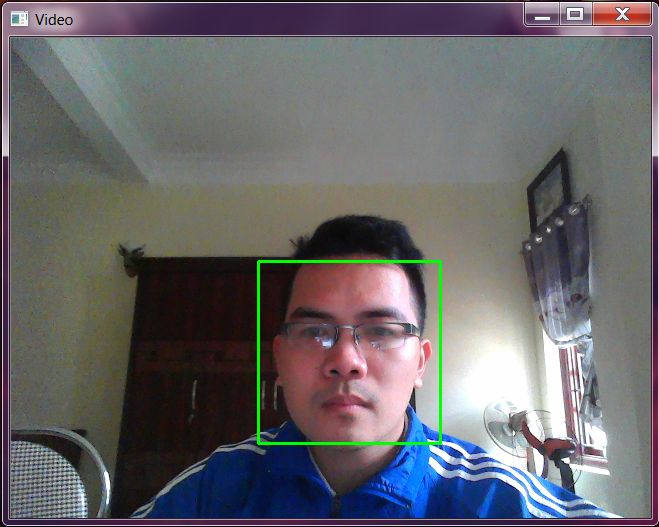
Ngoài ra chương trình còn thiết kế một số chức năng phụ:

* Start Capture : để sử dụng được 2 chức năng chính Capture Face và Training Face thì trước hết cần phải bật camera lên. Chức năng Start Capture dùng để bật camera phục vụ cho 2 chức năng chính đó.
* End Capture : dùng để tắt camera khi không còn nhu cầu dùng
* Quit : Dùng để thoát khỏi ứng dụng.

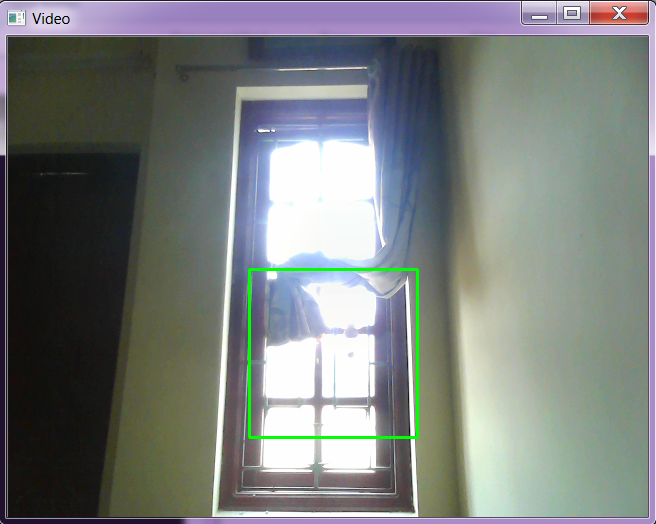
## 4.2. Kết quả phát hiện khuôn mặt.

Trong OpenCV tập huấn luyện phát hiện khuôn mặt có vài trăm mẫu ảnh khẳng định, phủ định cùng kích cỡ 20X20. Với số lượng như vậy, việc nhận diện khuôn mặt tương đối chính xác. Tuy nhiên còn phụ thuộc vào chất lượng và tầm nhìn của camera, với laptop thông thường, camera chỉ rơi vào 1.2-2.0 Mpx, nên việc phát hiện khuôn mặt chỉ phù hợp với khoảng cách ngắn giữa người dùng và camera laptop, qua thực nghiệm với các loại laptop tầm trung thì khoảng cách chỉ trong phạm vi nhỏ hơn 3 mét với ánh sáng yếu, 5 – 10 mét với ánh sáng tốt, với khoảng cách xa việc phát hiện khuôn mặt gặp nhiều khó khăn. Với ứng dụng chấm công bằng khuôn mặt thì nhược điểm này có thể chấp nhận được, vì người dùng sẽ đứng trước camera của laptop 1 khoảng cách đủ gần nên vấn đề về khoảng cách không phải là vấn đề đáng lo.

Trong điều kiện ánh sáng tốt, không có nhiều nhiễu thì khả năng phát hiện chính xác đạt 87% với 150 lần thử nghiệm với các khuôn mặt khác nhau của báo cáo. Trong điều kiện ánh sáng yếu, với những đối tượng ở xa việc nhận diện giảm xuống còn 80% với 150 lần thử nghiệm. Với các đối tượng tương tự khuôn mặt như ô cửa sổ, ngăn bàn,… có các vùng sáng tối giống các vùng của khuôn mặt thường bị nhận nhầm.



Hình 4.2 : Phát hiện khuôn mặt đúng



Hình 4.3 : Phát hiện khuôn mặt sai

Để tăng tính chính xác, trong thời gian gần tác giả sẽ thay đổi tập huấn luyện, tăng số lượng ảnh phủ định và khẳng định sao cho phù hợp, đồng thời thử nghiệm với các máy tính có camera tốt hơn.

## 4.3. Nhận diện khuôn mặt

LBPH là thuật toán tốt có độ tin cậy cao, tuy nhiên, chất lượng nhận diện cũng phụ thuộc nhiều vào tập huấn luyện. Chương trình chạy tốt với số lượng ảnh có từ 2 con số trở nên. Với số lượng ít hơn các ảnh có các thuộc tính gần giống nhau rất dễ bị nhận nhầm.

## 4.4. Chấm công bằng nhận diện khuôn mặt.

Với kết quả thực nghiệm về nhận dạng khuôn mặt và phát hiện khuôn mặt đã nêu ở trên. Thì chương trình có tốc độ phát hiện đủ nhanh và khá chính xác với khuôn mặt, không gây khó khăn cho người sử dụng. Tuy nhiên, chương trình dễ bị nhầm lẫn các khuôn mặt có thuộc tính tương tự nhau và không được thiết kế để phân biệt người và ảnh, do đó chương trình chỉ thích hợp để tạo thêm một tính năng chấm công mới phục vụ cho việc chấm công của công ty, không thích hợp với những công ty có yêu cầu bảo mật dữ liệu cao.

## 4.5. Đánh giá độ tin cậy của chương trình.

### 4.5.1. Mô hình đánh giá.

Chương trình được xây dựng trên các bộ huấn luyện và thử nghiệm như sau :

* Bộ huấn luyện phát hiện khuôn mặt gồm các mẫu của OpenCV.
* Bộ huấn luyện nhận dạng khuôn mặt gồm 10 người với mỗi người một bộ từ 10-15 ảnh huấn luyện được chụp từ camera của laptop tác giả hoặc từ các nguồn khác. Với bộ huấn luyện nhận dạng khuôn mặt này ta sẽ thử nghiệm nhận dạng trong nhà 100 lần với 10 người và thử nghiệm nhận dạng ngoài trời 100 lần với 10 người này.

Các tham số tham gia đánh giá độ tin cậy có thể hiểu như sau:

* a là số trường hợp dự đoán là khuôn mặt và sau đó thực nghiệm chứng minh đúng là khuôn mặt được gán nhãn.
* b là số trường hợp dự đoán không là khuôn mặt và sau đó thực nghiệm chứng minh không là khuôn mặt.
* c là số trường hợp dự đoán là khuôn mặt và sau đó thực nghiệm chứng minh không là khuôn mặt.
* d là số trường hợp dự đoán không là khuôn mặt và sau đó thực nghiệm chứng minh là khuôn mặt.

Độ chính xác (Precesion) thể hiện các văn bản được đưa vào hệ thống đánh giá có đúng với bản chất thực tế của nó không:

Precision =

Độ hồi quy (Recall) thể hiện các văn bản được đưa vào hệ thống có phù hợp với truy vấn đầu vào của người dùng không:

Recall =

Độ F (F-measure) thể hiện giá trị trung hòa của độ quy hồi và độ chính xác:

F =2

### 4.5.2. Kết quả đánh giá về mặt thời gian.

|  |  |  |
| --- | --- | --- |
| Loại | Số thực thể | Thời gian trung bình/lần (giây) |
| Phát hiện khuôn mặt | 100 | 1 |
| Nhận dạng khuôn mặt | 100 | 1 |
| All | 200 | 2 |

Bảng 4.1: Thời gian thực thi

Với khoảng 200 lần thử nghiệm trên 10 người khác nhau, thời gian thực thi của phần mềm được tổng hợp trong bảng 4.1. Do thuật toán AdaBoost có tốc độ thực hiện nhanh và tập huấn luyện nhỏ nên thời gian thực thi của phần mềm là chấp nhận được với yêu cầu bài toán và với người sử dụng.

### 4.5.3. Kết quả đánh giá về tính chính xác nhận dạng khuôn mặt.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Loại | Số  thực thể | Số thực thể được nhận dạng | Số thực thể được nhận đúng | Precision | Recall | F-measure |
| Ánh sáng phòng | 100 | 85 | 72 | 84.71% | 72% | 77.84% |
| Ánh sáng ngoài trời | 100 | 93 | 84 | 90.32% | 84% | 87.05% |
| All | 200 | 178 | 156 | 86.64% | 78% | 82.01 % |

Bảng 4.2: Kết quả nhận dạng khuôn mặt.

Với các thuật toán và tập huấn luyện được sử dụng, phần mềm cho kết quả độ chính xác đạt xấp xỉ 87% và độ F ở khoảng trên 82%. Kết quả tương đối tốt khi thực nghiệm ở trong phòng và ngoài trời do khoảng cách gần và hướng khuôn mặt là trực diện với camera. Tác giả sẽ tiếp tục nghiên cứu để cải thiện phần mềm cho phép nhận dạng với nhiều góc nhìn khác nhau và nhiều chi tiết có thêm của khuôn mặt.

## 4.6. Kết luận.

Việc sử dụng nhận diện khuôn mặt để chấm công có một số ưu điểm như sau:

* Người dùng có thêm 1 cách mới để áp dụng vào việc chấm công.
* Chi phí cài đặt hệ thống chấm công bằng nhận diện khuôn mặt sẽ ít hơn so với các hệ thống cùng loại như là nhận diện bằng vân tay, nhận diện bằng tròng mắt,...
* Thời gian chấm công khá nhanh.

Các vấn đề còn tồn tại :

* Với khoảng cách gần với camera và không dùng thêm các kỹ thuật phụ trợ, người dùng có thể dễ dàng chấm công bằng một tấm ảnh, làm giảm tính bảo mật của chương trình.
* Không thể mở khóa trong bóng tối, vì hệ thống nhận dạng không chính xác khi ánh sáng quá kém.

Để cải tiến thì những ứng dụng sử dụng nhận dạng khuôn mặt cho các hoạt động liên quan đến đóng/ mở hệ thống, an minh,… cần phải dùng thêm một số kỹ thuật phụ trợ để có thể phân biệt con người với các thực thể khác như hình nộm, tranh ảnh,… nhằm tăng tính chính xác cho ứng dụng.

## 4.7. Kết chương.

Trong chương này,người viết báo cáo đã trình bày về ứng dụng chấm công bằng nhận diện khuôn mặt và một số giao diện của hệ thống.

Và phần cuối chương là mô tả các đánh giá về độ chính xác của hệ thống bao gồm độ chính xác, chỉ số hồi quy và độ F.

# Chương 5: Kết luận và hướng phát triển.

Nhận dạng khuôn mặt là một bài toán có tính ứng dụng cao và có thể khai thác trên nhiều hệ thống, nhất là những hệ thống có camera. Qua quá trình thực hiện báo cáo tác giả đã nghiên cứu tổng quan về các phương pháp nhận dạng khuôn mặt và lựa chọn được các công cụ, phương pháp phù hợp để có thể cài đặt ứng dụng nhận dạng khuôn mặt, và ứng dụng vào chấm công.

Qua thực nghiệm chương trình hoạt động tốt với thời gian xử lý nhanh chóng và độ chính xác chấp nhận được, với các thuật toán áp dụng là adaboost và LBPH. Tác giả đồng thời cũng nhận ra những vấn đề về nhận dạng đối tượng bằng camera máy tính đó là: chất lượng camera thấp do đó chỉ có thể nhận diện khuôn mặt tốt trong một khoảng cách nhất định; Do khuôn mặt có tính phổ biến cao nên chương trình dễ bị nhầm lẫn khi gán nhãn; Khuôn mặt có thể được làm giả bằng ảnh người nên sẽ không mang lại hiệu quả bảo mật cao. Đồng thời tác giả cũng chưa xử lý được tốt việc phát hiện các đối tượng tương tự khuôn mặt do tập huấn luyện chưa đủ lớn.

Trong thời gian tới, tác giả sẽ tiếp tục nghiên cứu và thực nghiệm để tìm ra con số cụ thể hợp lý cho tập huấn luyện phát hiện và nhận dạng khuôn mặt, cũng như cách lưu trữ cơ sở dữ liệu và thông tin huấn luyện sao cho tốc độ truy xuất và tính toán được giảm xuống. Đồng thời, tác giả cũng sẽ tìm hiểu thêm cách sử dụng các thuật toán và công nghệ khác để có thể nhận dạng ra khuôn mặt trong video phân biệt với việc nhận dạng một bức ảnh.

**Đánh giá kết quả và hướng phát triển.**

1. **Mục tiêu hoàn thành.**

* Phát triển ứng dụng phát hiện và nhận dạng khuôn mặt và áp dụng vào việc chấm công.
* Nghiên cứu và áp dụng thuật toán ADABOOST vào xây dựng ứng dụng.
* Nghiên cứu và áp dụng thuật toán Cascade of Classifier vào xây dựng ứng dụng.
* Nghiên cứu và áp dụng các đặc tính Haar-like vào ứng dụng.
* Nghiên cứu và ứng dụng thuật toán LBP.
* Nghiên cứu và áp dụng bộ thư viện OpenCV vào xây dựng ứng dụng.
* Nghiên cứu và áp dụng ngôn ngữ Python vào xây dựng ứng dụng.

1. **Tồn tại**.

* Chưa xây dựng được tập các đặc tính Haar tối ưu cho việc phát hiện khuôn mặt.
* Chưa tối ưu code.
* Độ chính xác chưa cao.

1. **Hướng phát triển của đề tài.**

* Bổ xung các thuộc tính Haar nhằm nâng cao độ chính xác trong việc phát hiện khuôn mặt.
* Cải thiện thuật toán LBP và áp dụng thêm các thuật toán khác nhằm nâng cao tính chính xác của bài toán nhận dạng.
* Tối ưu hóa code của ứng dụng.

# TÀI LIỆU THAM KHẢO

1. Paul Viola, Michael Jones. Rapid object detection using a boosted cascade of simple features. In Proceeding of Conference on Computer Vision and Pattern Recognition, pages 511–518, 2014.

2. M. Turk, A. Pentland, “Face Recognition Using EigenFaces,”Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pages 586-591, 1991.

3. P.N. Belhumeur, J.P. Hespanha and D.J. Kriegman, “EigenFacesvs. Fisherfaces: Recognition Using Class Specific Linear Projection,”European Conf. Computer Vision, pages 45-58, 1996.

4. X. Feng, A. Hadid , M. Pietikainen, Facial Expression Recognition with Local Bi-nary Patternsand Linear Programming", Pattern Recognition and ImageAnalysis, Vol. 15(2) , pages 546-549, 2005.

5. Daniel Lélis Baggio, Shervin Emami, Mastering OpenCV with Practical Computer Vision Projects, Packt Publishing Ltd., December 3, 2012

6. Gary Bradski, Adrian Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O’Reilly Media Inc., October 1, 2008

7. Jan Erik Solem, Programming Computer Vision with Python: Tools and algorithms for analyzing images, O’Reilly Media Inc., June 29, 2012.

8. Oscar Deniz Suarez, Mª del Milagro Fernandez Carrobles, OpenCV Essentials, Packt Publishing Ltd., August 25, 2014.

9. Simon J. D. Prince, Computer Vision: Models, Learning, and Inference Hardcover, Cambridge University Press, June 18, 2012.

10. Joseph Howse, OpenCV Computer Vision with Python, Packt Publishing Ltd., January 7, 2015.

11. Kurt Demaagd, Anthony Oliver, Practical Computer Vision with SimpleCV: The Simple Way to Make Technology See, O’Reilly Media Inc., August 9, 2012.

12. Robert Laganière, OpenCV2 Computer Vision Application Programming Cookbook, Packt Publishing Ltd., May 23, 2011.

13. Reinhard Klette, Concise Computer Vision: An Introduction into Theory and Algorithms (Undergraduate Topics in Computer Science), Springer Publishing, January 31, 2014.

14. Richard Hartley, Andrew Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, April 19, 2004.

15. Samarth Brahmbhatt, Practical OpenCV (Technology in Action), November 19, 2013.

16. OpenCV documentation index, http://docs.opencv.org, last visited December, 2015.

17. Python homepage, https://www.python.org/, last visited December, 2015.

18. ZetCode, tutorials for programmers, http://zetcode.com/, last visited December, 2015.

19. PyQt Class Reference, http://pyqt.sourceforge.net/Docs/PyQt4/classes.html, last visited December, 2015.

20. Requests: HTTP for Humans &mdash; Requests 2.9.0 documentation, http://docs.python-requests.org/en/latest/, last visited December, 2015.

21. NumPy, http://www.numpy.org/, last visited December, 2015.