Trường Đại học Bách Khoa Hà Nội Viện Công nghệ thông tin và truyền thông



-------------\*\*\*-------------

### **HỆ CƠ SỞ DỮ LIỆU ĐA PHƯƠNG TIỆN**

### ĐỀ TÀI :

**Hệ thống phát hiện và nhận dạng mặt người**

*Nhóm sinh viên thực hiện* :

|  |  |  |
| --- | --- | --- |
| Lưu Trung Hiếu | - 20141515 | - CNTT 2-4 |
| Vũ Trọng Hiệu | - 20141683 | - CNTT 2-3 |
| Roãn Văn Thụ | - 20144416 | - CNTT 2-4 |

*Giảng viên hướng dẫn :* **TS. Nguyễn Thị Oanh**

Hà Nội, 12/2017

**MỤC LỤC**

[1. Giới thiệu bài toán. 3](#_Toc502156936)

[2. Thuật toán Viola&Jones trong phát hiện khuôn mặt 3](#_Toc502156937)

[2.1 Các đặc trưng 3](#_Toc502156938)

[2.2 Integral Image 3](#_Toc502156939)

[2.3 Học hàm phân loại 4](#_Toc502156940)

[2.4 Mô hình Cascade (phân tầng) 6](#_Toc502156941)

[3. Nhận dạng khuôn mặt sử dụng Eigen Face. (Principal Component Analysis) 6](#_Toc502156942)

[4. Nhận dạng khuôn mặt sử dụng Fisherfaces. 8](#_Toc502156943)

[5. Kết quả 9](#_Toc502156944)

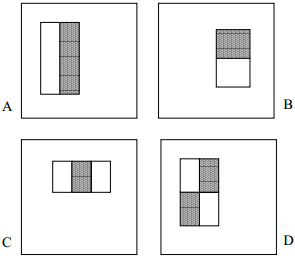
[6. Tài liệu tham khảo 11](#_Toc502156945)

1. Giới thiệu bài toán.

Phát hiện và nhận dạng mặt người là bài toán mới xuất hiện cách đây không lâu. Ngày nay, các hệ thống phát hiện và nhận dạng mặt người đang được ứng dụng trong nhiều lĩnh vực, đặc biệt là an ninh và bảo mật. Nhận dạng mặt người tức là đưa ra những thông tin về đối tượng được đưa vào từ một bức ảnh, hay từ camera quan sát. Bước đầu tiên của một hệ thống nhận dạng mặt người là phát hiện khuôn mặt, tức xác định vị trí khuôn mặt trên bức ảnh, sau đấy tách khuôn mặt ra khỏi ảnh để tiến hành nhận dạng. Để giải quyết bài toán này, nhóm chúng em đã chọn phương pháp phát hiện khuôn mặt bằng thuật toán Viola&Jones và nhận diện chúng trên 2 phương pháp EigenFace và FisherFace.

2. Thuật toán Viola&Jones trong phát hiện khuôn mặt

2.1 Các đặc trưng

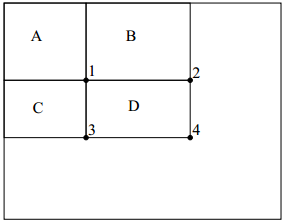


Hình 1: Các đặc trưng hình chữ nhật (Haar-like feature). Được tính với mọi kích thước và vị trí trong một cửa sổ chạy

* Giá trị của đặc trưng =
* Lý do sử dụng các đặc trưng thay vì giá trị pixel:
  + Mã hóa được các tri thức lĩnh vực (domain knowledge) đặc biệt mà khó có thể học được từ tập huấn luyện hữu hạn.
  + Hệ thống hoạt động dựa trên feature xử lý nhanh hơn hệ thống hoạt động dựa trên pixel

2.2 Integral Image

* Việc tính tổng pixel trong các hình chữ nhật rất mất thời gian và tăng theo kích thước hình, do đó Integral Image được đề xuất để cho phép tính các tổng với thời gian hằng số, không phụ thuộc vào kích thước hình.
* Giá trị Integral Image ở vị trí x, y thì bằng tổng các pixel phía trên và bên trái của điểm đó trong ảnh gốc bao gồm chính nó:
* Việc tính tổng các pixel theo một hình chữ nhật như sau:

Hình 2: Tính tổng pixel trong hình chữ nhật D thông qua 4 truy vấn giá trị trong Integral Image.

Việc tính cho toàn bộ ảnh có thể được tính bằng cách duyệt một lần qua ảnh theo công thức:

y)

Trong đó là tổng tích lũy theo hàng

2.3 Học hàm phân loại

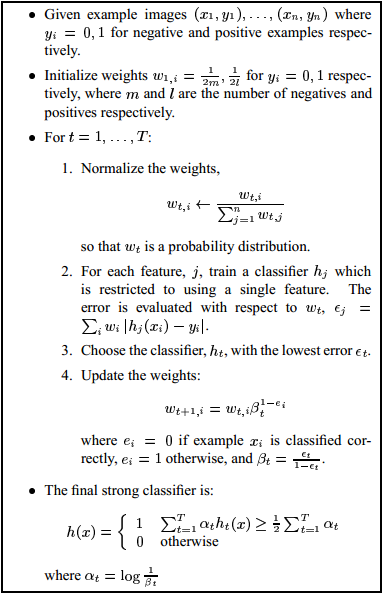
* Thuật toán AdaBoost được sử dụng vừa để trọn ra tập nhỏ các thuộc tính tốt có giá trị phân loại, vừa để huấn luyện bộ phân loại mạnh (strong classifier) từ các bộ phân loại yếu (weak classifier)
* Bộ phân loại yếu là bộ phân loại chỉ sử dụng một đặc trưng hình chữ nhật để thực hiện việc phân loại (với thừa nhận rằng nó sẽ phân loại tốt hơn 50%). Bộ phân loại yếu:

Với là giá trị của đặc trưng đối với ảnh đầu vào

là giá trị xác định chiều so sánh

là giá trị ngưỡng cho bộ phân loại

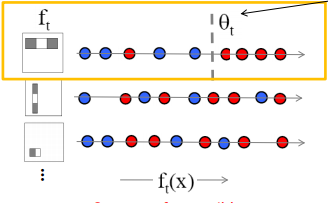
* Thuật toán AdaBoost:

Đầu vào gồm tập các ví dụ chứa mặt và không chứa mặt

Khởi tạo weight cho từng ví dụ, khởi đầu bằng nhau trong từng lớp

Thuật toán chạy T vòng, mỗi vòng chọn ra một weak classifier để dùng cho strong classifier

1. Chuẩn hóa trọng số tập trọng số tạo nên một phân phối xác suất
2. Huấn luyện các bộ phân loại , trả về lỗi



1. Chọn ra bộ phân loại nhỏ nhất để làm weak classifier
2. Cập nhật trọng số cho các ví dụ dựa trên trọng số hiện tại và lỗi của weak classifier. Tăng trọng số những ví dụ bị phân loại sai, giảm trọng số ví dụ được phân loại đúng

Strong classifier cuối cùng là tổ hợp tuyến tính của T các weak classifier, với trọng số tương ứng với lỗi của từng classifier, lỗi càng cao thì trọng số càng nhỏ và ngược lại

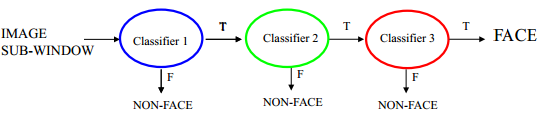
* Chọn threshold cho mỗi classifier, với mỗi example, tìm giá trị lỗi:

: tổng trọng số của các example; : tổng trọng số các ví dụ có giá trị feature nhỏ hơn giá trị hiện tại (+ cho positive example, - cho negative example)

* Chọn giá trị nhỏ nhất, giá trị của feature tương ứng với sample đó là threshold cần tìm

2.4 Mô hình Cascade (phân tầng)

* Sau khi tìm được bộ phân loại mạnh, với mỗi ví dụ (sub-window) đưa vào ta cần tìm giá trị theo T weak classifier để biết đó có phải khuôn mặt không, tuy nhiên việc tính toán như vậy là không được tối ưu vì phần lớn các ví dụ không phải khuôn mặt.
* Mô hình Cascade được xây dựng để có thể sớm bỏ qua những ví dụ không chứa khuôn mặt ở những tầng (stage) đầu tiên (sử dụng các bộ phân loại đơn giản), những ví dụ qua được tầng trước sẽ đến các tầng sau (dùng các bộ phân loại phức tạp hơn) và có càng nhiều khả năng là khuôn mặt.



Hình 3: Kết hợp các bộ phân loại theo mô hình cascade

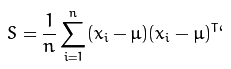
* Thông thường, bộ phân loại với nhiều đặc trưng hơn sẽ đạt được tỉ lệ phát hiện (detection rates = TP/(TP+FN)) cao hơn và tỉ lệ false positive (FP/(TN+FP)) thấp hơn. Mục tiêu huấn luyện mỗi stage là giảm tỉ lệ false positive, tăng tỉ lệ detection. Có 2 tham số cần lựa chọn đó là giá trị lớn nhất cho tỉ lệ false positive (maxFalseAlarmRate) và giá trị nhỏ nhất cho tỉ lệ detection (minHitRate).
* Huấn luyện một mô hình cascade:
  + Chọn minimum detection rate () và maximum false positive rate () cho mỗi stage
  + Chọn cho toàn mô hình,
  + - * Sử dụng để huấn luyện một bộ strong classifer với feature
      * Đánh giá trên tập validation để xác định
      * Giảm threshold của strong classifier hiện tại cho đến khi mô hình cascade hiện có đạt tỉ lệ detection
    - Nếu thì dùng mô hình cascade hiện tại để đánh giá trên tập , với các ví dụ phân loại sai, thêm vào tập

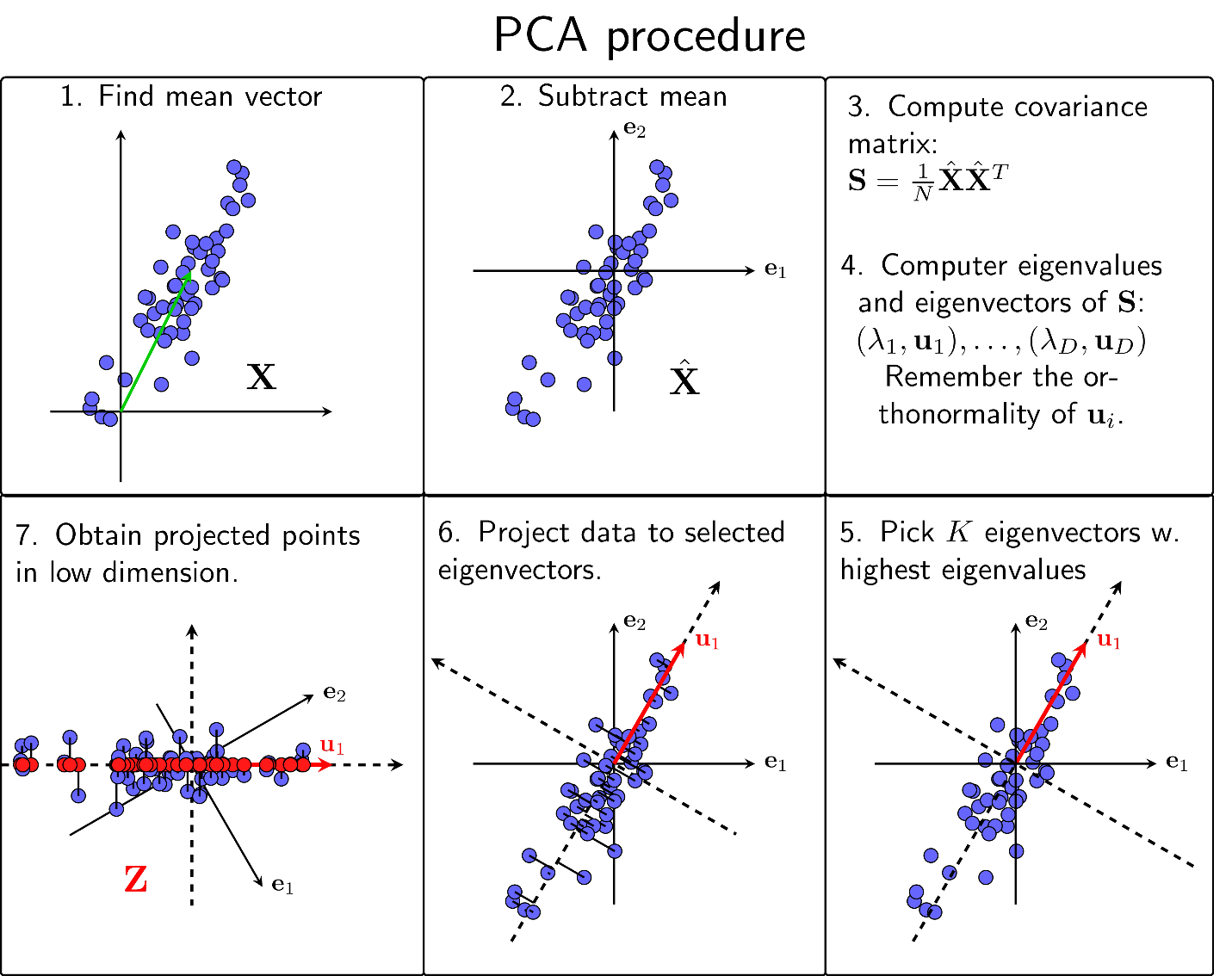
3. Nhận dạng khuôn mặt sử dụng Eigen Face. (Principal Component Analysis)

**Ý tưởng** của thuật toán: Giảm số chiều của các ảnh sao cho lượng thông tin giữ lại được nhiều nhất có thể. Từ đó dễ dàng so sánh giữa ảnh test và các ảnh train.

**Các bước thực hiện** thuật toán:

Với xi là một vector được tạo ra bằng cách sắp xếp các cột trong ma trận của 1 ảnh (1 điểm dữ liệu) thành 1 cột.

* Tính vector trung bình:
* 
* Tính ma trận hiệp phương sai:
* 
* Tính các trị riêng và vector riêng và sắp xếp các vector riêng theo giá trị giảm dần của trị riêng:
* 
* Chọn k vector riêng tương ứng với k trị riêng lớn nhất để xây dựng ma trận Uk có các cột là các vector riêng này. Các vector này còn được gọi là các thành phần chính (principal components), tạo thành một không gian con.
* Chiếu dữ liệu đã được chuẩn hóa ban đầu (trừ đi vector trung bình) xuống không gian con tìm được.
* Dữ liệu mới chính là tọa độ của các điểm dữ liệu trên không gian mới.
* Khi có một ảnh đầu vào cần nhận dạng, ta cũng chiếu ảnh đó xuống chiều không gian mới và áp dụng thuật toán KNN để tìm nhãn cho ảnh đầu vào.



4. Nhận dạng khuôn mặt sử dụng Fisherfaces.

**Ý tưởng cơ bản.**

Thuật toán sử dụng cho phương pháp Fisherface là Linear Discriminant Analysis (LDA), là một phương pháp giảm chiều dữ liệu cho bài toán phân lớp. LDA đi tìm một phép chiếu sao cho trong chiều dữ liệu mới: dữ liệu trong mỗi class có xu hướng giống nhau, trong khi dữ liệu thuôc các class khác nhau thì càng cách xa nhau càng tốt.

**Các bước thưc hiện thuật toán.**

Giả sử X là tập dữ liệu gồm ***c*** lớp, trong đó X­i là tập các vector biểu diễn dữ lệu ảnh trong class i.

https://docs.opencv.org/2.4/_images/math/18a6fcf31effcba883e2ab9edda8a41c69a55c1f.png

Tính vector trung bình của tất cả các điểm dữ liệu:

https://docs.opencv.org/2.4/_images/math/b8ccd3cb61a7a760b25fb8c11b7c134c2c393785.png

Tính vector trung bình của class i:

https://docs.opencv.org/2.4/_images/math/3c376f49b70e2c4e3046f13d7d5d8dfdd4773fd4.png

**Tính ma trận đo mức độ phân tán dữ liệu bên trong các class:**

**D:\hoc tap\ky 20181\47c5a42e0f497c2f6364c687f5d248f9a0c8bd78.png**

**Tính ma trận đo mức độ phân tán dữ liệu của các class với nhau:**

**D:\hoc tap\ky 20181\47c5a42e0f497c2f6364c687f5d248f9a0c8bd78.png**

Thuật toán của Fisherfaces là tìm một phép chiếu để tối đa hóa hàm mục tiêu:

W_{opt} = \operatorname{arg\,max}_{W} \frac{|W^T S_B W|}{|W^T S_W W|}

Giải bài toán tối ưu trên ta được phương trình như sau:

\begin{align*}
    S_{B} v_{i} & = & \lambda_{i} S_w v_{i} \nonumber \\
    S_{W}^{-1} S_{B} v_{i} & = & \lambda_{i} v_{i}
\end{align*}

Vậy lời giải của bài toán là các giá trị riêng của ma trận .

Trong trường hợp suy biến, dữ liệu ảnh được chiếu lên một không gian thấp hơn để p không còn suy biến. Để làm được điều này, ta dùng PCA để giảm kích thước của không gian đăc trưng thành N-c chiều, sau đó áp dụng giải thuât LDA trên dữ liệu vừa giảm.

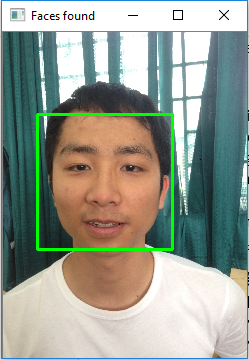
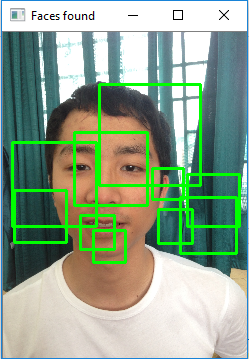
Sau khi tìm được các vector riêng , chọn k vector ứng với trị riêng lớn nhất để xây dựng ma trận Uk và thực hiện các bước còn lại tương tự như phương pháp Eigen Face.

5. Kết quả

**Phát hiện khuôn mặt với thuật toán Viola&Jones:**

Trường hợp 1: sử dụng model frontal-face có sẵn trong opencv.

Trường hợp 2: tự train 1 model với 3361 ảnh face, 5000 ảnh non-face, kích thước 19x19 pixel, với minHitRate = 0.995 và maxFalseAlarmRate = 0.5 cho mỗi stage ( được tổng hợp từ tập CBCL Face Database - <http://www.ai.mit.edu/projects/cbcl> , The UMIST Face Database và một số ảnh tự chụp)

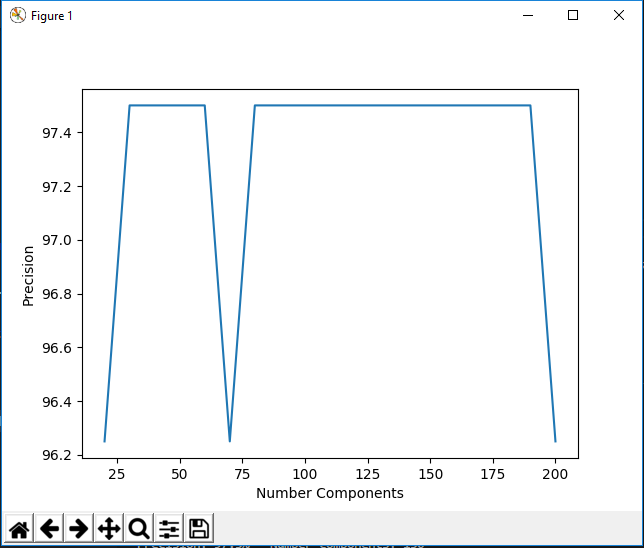
Hình : Trường hợp 1 (model có sẵn) Hình : Trường hợp 2 (model tự train)

**Kết luận**: Với model có sẵn trong OpenCV, việc phát hiện khuôn mặt khá chính xác. Còn với model tự train, kết quả chưa được như ý muốn.

Lý do: tập data train còn nhỏ, không bao quát được hết các đặc trưng khuôn mặt ở các điều kiện khác nhau

**Nhận dạng khuôn mặt với EigenFace:**

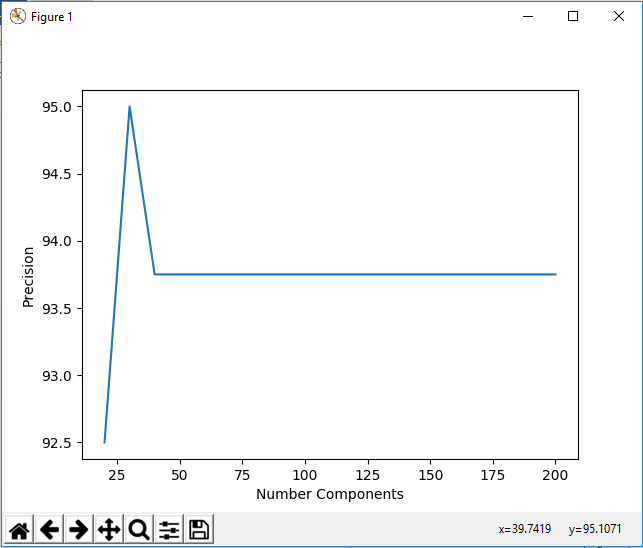
Kết quả được test trên tập AT&T gồm ảnh mặt của 40 người khác nhau, mỗi người gồm 10 ảnh, trong đó lấy 8 ảnh đầu để train và 2 ảnh sau để test. Tham số Number Components chạy từ 20 đến 200 bước nhảy 10 (mặc định là 80). Độ chính xác cao nhất đạt 97.5 % tại các num\_components = [30,60],[80,190]



**Kết luận**: Kết quả trên tập AT&T mẫu là khá chính xác. Tuy nhiên tập data set này mới chỉ là tập rất nhỏ, chưa bao quát được hết các trường hợp.

**Nhận dạng khuôn mặt với FisherFace:**

Kết quả được test trên tập AT&T như phương pháp EigenFace với bộ tham số tương tự. Độ chính xác cao nhất đạt 95% tại num\_components = 30



**Kết luận**: Kết quả trên tập AT&T mẫu khá chính xác, kết quả có hơi thấp hơn so với phương pháp EigenFace.

6. Tài liệu tham khảo

* Robust Real-Time Face Detection – Viola & Jones
* Recognition, Part II: Face Detection via AdaBoost - Linda Shapiro
* <http://note.sonots.com/SciSoftware/haartraining.html>
* Blog machine learning cơ bản: PCA, LDA

<https://machinelearningcoban.com/2017/06/15/pca/>

<https://machinelearningcoban.com/2017/06/30/lda/>

* Face Recognition OpenCV

<https://docs.opencv.org/3.3.1/da/d60/tutorial_face_main.html>