**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN KHOA HỌC MÁY TÍNH**

- - - - 🙞🙞🟔🙜🙜 - - - -

**NHÓM LOTUS**

**0712228 – TRẦN TRUNG KIÊN**

**0712263 – VẠN DUY THANH LONG**

**PHÁT HIỆN KHUÔN MẶT VỚI THUẬT TOÁN ADABOOST**

**ĐỒ ÁN MÔN HỌC KỸ THUẬT TRÍ TUỆ NHÂN TẠO**

**GIẢNG VIÊN LÝ THUYẾT**

**TS. LÊ HOÀNG THÁI**

**NĂM 2010**

# MỤC LỤC

MỤC LỤC 2

Chương 1 NỀN TẢNG LÝ THUYẾT 4

1.1 Đặc trưng Haar-like 4

1.1.1 Tổng quan về đặc trưng Haar-like 4

1.1.2 Hình dung ban đầu về việc áp dụng đặc trưng Haar-like để phát hiện khuôn mặt 6

1.2 Thuật toán Adaboost (Adaptive boost) 7

1.2.1 Sơ bộ về ý tưởng 7

1.2.2 Thuật toán 8

1.2.3 Hiểu thuật toán 8

1.2.3.1 Mức sơ bộ 8

1.2.3.2 Mức chi tiết 13

1.3 Chuỗi các bộ phân lớp (The cascade of classifiers) 17

1.3.1 Dẫn nhập 17

1.3.2 Ý tưởng 17

1.3.3 Huấn luyện một cascade 18

Chương 2 CÀI ĐẶT 21

2.1 Phát biểu bài toán 21

2.2 Giải pháp 21

2.2.1 Các bộ phân lớp yếu 21

2.2.2 Tiền xử lý 22

2.2.3 Xây dựng chuỗi các bộ phân lớp 22

2.2.4 Hậu xử lý 23

Chương 3 THỰC NGHIỆM 25

3.1 Dữ liệu 25

3.1.1 Dữ liệu huấn luyện 25

3.1.2 Dữ liệu kiểm thử 25

3.2 Kịch bản 26

3.2.1 Số bộ phân lớp trong cascade 26

3.2.2 Kết quả trên các bộ test 27

Chương 4 KẾT LUẬN 28

4.1 Nhận xét về mô hình 28

4.1.1 Ưu điểm 28

4.1.2 Khuyết điểm 29

4.2 Hướng phát triển 29

PHỤ LỤC: HƯỚNG DẪN SỬ DỤNG CHƯƠNG TRÌNH 30

TÀI LIỆU THAM KHẢO 32

# NỀN TẢNG LÝ THUYẾT

## Đặc trưng Haar-like

### Tổng quan về đặc trưng Haar-like

Mỗi đặc trưng Haar-like chứa 2, 3 hoặc 4 miền hình học có màu “trắng” hoặc “đen.” Để rõ hơn, ta hãy ngó qua các đặc trưng Haar-like dưới đây:

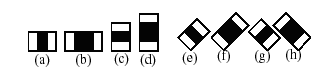
* Tập các đặc trưng Haar-like cơ sở:



* Tập các đặc trưng Haar-like mở rộng:
  + Đặc trưng cạnh:



* + Đặc trưng đường:



* + Đặc trưng tâm – xung quanh:



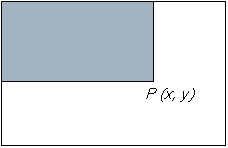
Giá trị của một đặc trưng Haar-like chỉ đơn giản là sự khác biệt giữa tổng các giá trị xám của các pixel trong vùng “đen” với tổng các giá trị xám của các pixel trong vùng “trắng”:

*f(x)=Sumblack rectangle (pixel gray level) – Sumwhite rectangle (pixel gray level)*

Nhận thấy ý nghĩa của đặc trưng Haar-like là nó thể hiện được tri thức về các đối tượng trong ảnh (thông qua việc thể hiện mối quan hệ giữa các bộ phận của đối tượng.)

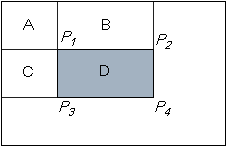
Ta tính giá trị của đặc trưng Haar-like rất nhanh thông qua một cách thức được gọi là “Integral Image.” Ở đây, Integral Image tại vị trí (x, y) có giá trị bằng tổng các giá trị pixel nằm trong hình chữ nhật được định bởi góc trái trên là (0, 0) và góc phải dưới là (x, y):





Từ đây, ta có thể tính được ngay tổng các giá trị pixel của một hình chữ nhật bất kỳ thông qua Integral Image tại 4 đỉnh của nó.

Ví dụ: Tính D.



Trong hình vẽ trên:

* A, B, C, D lần lượt là tổng các giá trị pixel của hình chữ nhật tương ứng.
* P1, P2, P3, P4 lần lượt là Integral Image tại 4 đỉnh của hình chữ nhật D.

Có ngay:

P1 = A, P2 = A + B, P3 = A + C, P4 = A + B + C + D

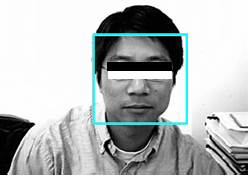
Suy ra:

P1+ P4–P2–P3= A + (A + B + C + D) – (A + B) – (A + C) = D

Vậy ta có: **D = P1 + P4 – P2 – P3**

### Hình dung ban đầu về việc áp dụng đặc trưng Haar-like để phát hiện khuôn mặt

Để phát hiện khuôn mặt, ảnh được quét bằng một cửa sổ con chứa một đặc trưng Haar-like.



Ứng với một đặc trưng Haar-like fj, một bộ phân lớp yếu yj(x) được định nghĩa như sau:

Trong đó:

* x là cửa sổ con.
* là ngưỡng.
* là hệ số với ý nghĩa là quyết định dấu của bất phương trình.

Ta hiểu công thức này đơn giản như sau: nếu giá trị của đặc trưng Haar-like tại cửa sổ con x vượt quá một ngưỡng nào đó thì đó là khuôn mặt; ngược lại thì không phải là khuôn mặt.

## Thuật toán Adaboost (Adaptive boost)

### Sơ bộ về ý tưởng

Kỹ thuật Boosting: nguyên tắc cơ bản của Boosting là kết hợp các bộ phân lớp yếu (hay các bộ phân lớp cơ sở) để tạo nên một bộ phân lớp mạnh. Các bộ phân lớp yếu này thậm chí chỉ cần nhỉnh hơn phương pháp random một chút. Bằng cách này, ta nói bộ phân lớp đã được “boost.”

Adaboost là một thuật toán sử dụng kỹ thuật Boosting được đưa ra bởi Freund và Schapire vào 1996. Thuật toán này được sử dụng rất phổ biến cho bài toán phát hiện khuôn mặt.

Điểm cải tiến của Adaboost là ta sẽ **gán cho mỗi mẫu một trọng số**. Ý nghĩa của việc gán trọng số như sau:

Ở mỗi vòng lặp của quá trình huấn luyện, khi một bộ phân lớp yếu yi đã được xây dựng, ta sẽ tiến hành cập nhật trọng số cho các mẫu. Việc cập nhật này được tiến hành như sau: ta sẽ tăng trọng số của các mẫu bị phân lớp sai bởi bộ phân lớp yếu yi và giảm trọng số của các mẫu được phân lớp đúng bởi yi. Bằng cách này, ở vòng lặp kế, ta sẽ xây dựng bộ phân lớp yếu yi+1 theo hướng: **tập trung vào các mẫu bị phân lớp sai bởi bộ phân lớp yếu trước đó**.

Cuối cùng, để có được bộ phân lớp mạnh, ta sẽ kết hợp tuyến tính các bộ phân lớp yếu đã tìm được lại với nhau. Mỗi bộ phân lớp yếu sẽ được đánh một trọng số tương ứng với độ tốt của bộ phân lớp yếu đó.

Đến đây, có lẽ ta đã hiểu được ý nghĩa của từ “Adaptive” (thích nghi) trong tên gọi của thuật toán này: các bộ phân lớp yếu tạo thành một chuỗi, trong đó bộ phân lớp yếu sau sẽ tập trung giải quyết các mẫu bị phân lớp sai bởi các bộ phân lớp yếu trước đó.

### Thuật toán

Cho tập ảnh huấn luyện (x1, t1), …, (xn, tn) với ti {-1, 1}

1. Khởi tạo hệ số trọng cho mỗi mẫu huấn luyện: với n = 1, 2, …, N.
2. For m = 1, …, M:

(a)Xây dựng bộ phân lớp yếu ym:

+ Với mỗi đặc trưng j, xây dựng một bộ phân lớp yjvới độ lỗi:

(1.0)

với nếu và = 0 nếu ngược lại.

+ Chọn bộ phân lớp yj với độ lỗi nhỏ nhất ta được ym.

(b) Cập nhật trọng số:

+ Tính:

(1.1)

và:

(1.2)

+ Cập nhật trọng số:

(1.3)

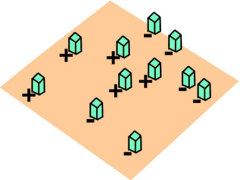
1. Bộ phân lớp mạnh cuối cùng:

(1.4)

### Hiểu thuật toán

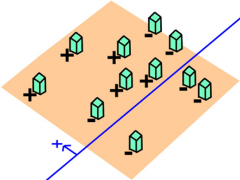
#### Mức sơ bộ

Thuật toán bắt đầu bằng việc khởi tạo trọng số cho các mẫu huấn luyện. Các trọng số này được khởi tạo bằng nhau. Các trọng số này cho thuật toán biết độ quan trọng của mẫu.



Ở mỗi vòng lặp, ta làm 2 việc:

* Thứ 1: tìm bộ phân lớp yếu ymdựa vào độ lỗi nhỏ nhất.

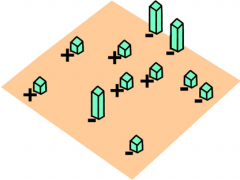
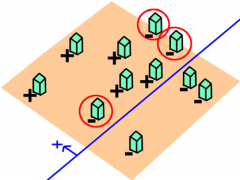


Công thức tính độ lỗi:

với nếu và = 0 nếu ngược lại.

Công thức này đơn giản chỉ là tính tổng các trọng số của các mẫu bị phân lớp sai.

* Thứ 2: cập nhật trọng số theo nguyên tắc: ta sẽ tăng trọng số cho các mẫu hiện đang bị phân lớp sai và giảm trọng số cho các mẫu hiện đang được phân lớp đúng. Bằng cách này, ở vòng lặp kế ta sẽ xây dựng bộ phân lớp yếu theo hướng tập trung giải quyết các mẫu bị phân lớp sai bởi bộ phân lớp yếu trước đó.



Trong đó:

với:

Ta nhận thấy nếu mẫu được phân lớp đúng thì trọng số không thay đổi; nếu mẫu bị phân lớp sai thì:

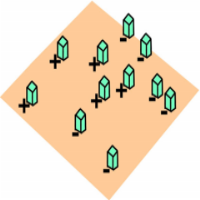
Thấy ngay

Thật vậy, giả sử , nghĩa là tỉ lệ phần sai lớn hơn phần đúng. Điều này mâu thuẫn với điều kiện bộ phân lớp yếu là tối ưu (có độ lỗi nhỏ nhất) vì rằng chỉ cần đổi chiều của mặt phẳng phân lớp lại là ta có ngay một bộ phân lớp yếu tốt hơn.

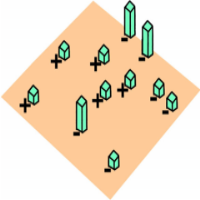
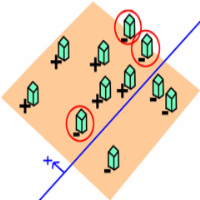
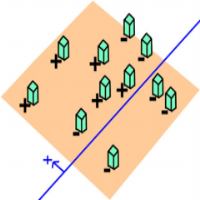
Và vì nên : trọng số của các mẫu bị phân lớp sai được tăng lên.

Để trực quan, ta hãy quan sát thuật toán thông qua chuỗi các hình vẽ dưới đây:

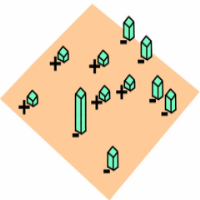
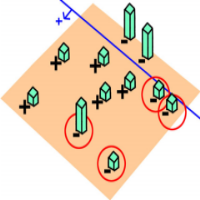
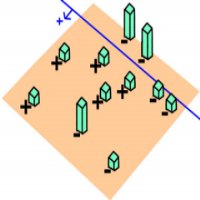
* Khởi tạo trọng số cho các mẫu:



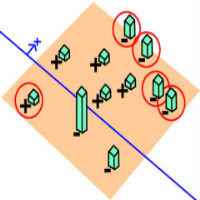
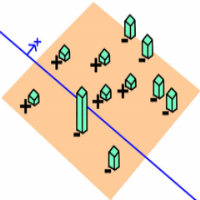
* Vòng lặp thứ 1:



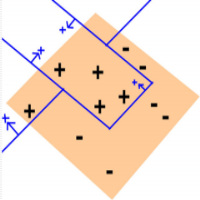
* Vòng lặp thứ 2:



* Vòng lặp thứ 3:



* Kết hợp các bộ phân lớp yếu lại:



Cuối cùng, kết hợp tuyến tính các bộ phân lớp yếu lại ta được một bộ phân lớp mạnh:

Nhận thấy mức độ đóng góp của mỗi bộ phân lớp yếu vào bộ phân lớp mạnh được quyết định bởi :

Dễ thấy tỉ lệ nghịch với mà thể hiện độ lỗi ứng với bộ phân lớp ym. Như vậy, độ lỗi càng thấp, tức bộ phân lớp càng tốt thì mức độ đóng góp của nó vào bộ phân lớp mạnh cuối cùng sẽ càng nhiều.

Nói thêm về điều kiện dừng: trong thực tế cài đặt, người ta sử dụng một giá trị ngưỡng là tỉ lệ nhận dạng sai tối đa (max false positive) để làm điều kiện dừng. Qua các vòng lặp, tỉ lệ nhận dạng sai của bộ phân lớp mạnh sẽ giảm dần. Đến một lúc nào đó, tỉ lệ này nhỏ hơn tỉ lệ nhận dạng sai tối đa và ta sẽ dừng thuật toán.

#### Mức chi tiết

Ở đây, ta sẽ chứng minh về toán học để thấy được vì sao lại có được các công thức trong thuật toán Adaboost.

Friedman (2000) đã đưa ra một cách hiểu rất đơn giản về thuật toán Adaboost. Ý tưởng sơ bộ là tối ưu hóa liên tiếp (sequential minimization) hàm lỗi mũ.

Ta có hàm lỗi mũ:

trong đó:

* là bộ phân lớp được tạo bởi sự kết hợp tuyến tính của các bộ phân lớp cơ sở:

với

Mục tiêu của ta là minimize E theocác biến: hệ số trọng và các tham số của bộ phân lớp cơ bản .

Ở đây, thay vì tối ưu hóa toàn cục, ta sẽ tiến hành tối ưu hóa dần dần (sequential minimization.)

Giả sử ta đã tìm được m-1 bộ phân lớp đầu. Như vậy, ở đây ta cần minimize E theo và .

Ta có:

với (1.5)

Đặt:

* Tm là tập các mẫu được phân lớp đúng bởi ym.
* Mm là tập các mẫu bị phân lớp sai bởi ym.

Có ngay:

Thêm bớt ta được:

(1) Minimize E theo : dẫn đến minimize:

Đây chính là công thức (1.0)

(2) Minimize E theo :

Cho đạo hàm = 0, ta có:

Đặt ta có:

Suy ra:

Suy ra tiếp:

Đến đây, ta có được công thức (1.2)

Bây giờ, ta sẽ chứng minh công thức cập nhật trong số (1.3)

Dựa vào (1.5) có ngay:

mà:

nên:

Vì với mọi mẫu, lượng là giống nhau nên ta có thể bỏ nó đi.

Vậy ta đã chứng minh xong công thức (1.3)

Và cuối cùng, dễ có công thức (1.4).

Như vậy, đến đây ta đã chứng minh xong thuật toán Adaboost.

## Chuỗi các bộ phân lớp (The cascade of classifiers)

### Dẫn nhập

Đến đây, suy nghĩ đầu tiên của ta là dùng thuật toán Adaboost để huấn luyện một bộ phân lớp mạnh. Sau đó, ta rải các cửa sổ con (sub window), chứa bộ phân lớp mạnh này, khắp một tấm ảnh đưa vào để phát hiện khuôn mặt.

*Tuy nhiên, cách làm này tốn chi phí*. Giả sử, bằng Adaboost ta có được một bộ phân lớp mạnh gồm 10 bộ phân lớp yếu. Nếu làm như trên, tại tất cả các cửa sổ con trên tấm ảnh ta đều phải dùng cả 10 bộ phân lớp yếu. Trong khi đó, ta thấy: những cửa sổ thật sự là khuôn mặt rất ít và tại những cửa sổ không phải là khuôn mặt ta có thể loại bỏ mà chỉ cần dùng một bộ phân lớp mạnh gồm ít hơn 10 bộ phân lớp yếu.

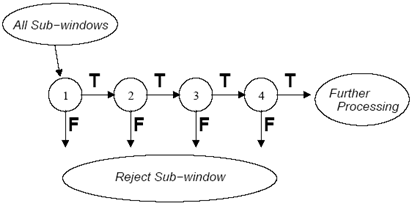
🡪Người ta giải quyết vấn đề này bằng chuỗi các bộ phân lớp (cascade).

### Ý tưởng

Ta sẽ có một chuỗi các bộ phân lớp, trong đó mỗi bộ phân lớp được xây dựng bằng thuật toán Adaboost.

Bây giờ, ta đưa tất cả các cửa sổ con đi qua chuỗi các bộ phân lớp này:

* Bộ phân lớp đầu tiên sẽ loại bỏ phần lớn các cửa sổ không phải khuôn mặt (nagative sub window) và cho đi qua các cửa sổ được cho là khuôn mặt (positive sub window). Ở đây, bộ phân lớp này rất đơn giản và do đó, độ phức tạp tính toán cũng rất thấp. Tất nhiên, vì rằng nó đơn giản nên trong số các cửa sổ được nhận dạng là khuôn mặt sẽ có một số lượng lớn cửa sổ bị nhận dạng sai (không phải là khuôn mặt.)
* Những cửa chỗ được cho đi qua bởi bộ phân lớp đầu sẽ được xem xét bởi bộ phân lớp sau đó: nếu bộ phân lớp cho rằng đó không phải là khuôn mặt thì ta loại bỏ; nếu bộ phân lớp cho rằng đó là khuôn mặt thì ta lại cho đi qua và chuyển đến bộ phân lớp phía sau.
* Những bộ phân lớp càng về sau thì càng phức tạp hơn, đòi hỏi sự tính toán nhiều hơn. Người ta gọi những cửa sổ con (mẫu) mà bộ phân lớp không loại bỏ được là những mẫu khó nhận dạng. Những mẫu này càng đi sâu vào trong chuỗi các bộ phân lớp thì càng khó nhận dạng. Chỉ những cửa sổ đi qua được tất cả các bộ phân lớp thì ta mới quyết định đó là khuôn mặt.



Tóm lại, chuỗi các bộ phân lớp sẽ xử lý các mẫu (cửa sổ con) đi vào theo nguyên tắc sau: nếu một bộ phân lớp nào đó cho rằng đó không phải là mặt người thì ta loại bỏ ngay; còn nếu bộ phân lớp cho rằng đó là khuôn mặt thì ta chuyển đến bộ phân lớp sau. Nếu một mẫu trót lọt hết tất cả các bộ phân lớp thì ta mới quyết định đó là khuôn mặt.

### Huấn luyện một cascade

Trước tiên, ta cần nói đến hai độ đo quan trọng là **detection rate** và **false positive rate**. Cụ thể như sau:

* Detection rate = Số lượng các vùng được nhận dạng là khuôn mặt và đúng / Số lượng khuôn mặt thật sự trong ảnh.
* False positive rate = Số lượng các vùng được nhận dạng là khuôn mặt và sai / Số lượng các vùng được nhận dạng là khuôn mặt.

Xét một cascade gồm K bộ phân lớp.

* Detection rate của cascade được tính như sau:

với di là detection rate của bộ phân lớp thứ i.

* False positive rate của cascade được tính như sau:

với di là detection rate của bộ phân lớp thứ i.

*Thuật toán huấn luyện:*

* Bước 1: Người dùng chọn các giá trị.
  + f: false positive rate lớn nhất chấp nhận được của mỗi bộ phân lớp trong cascade.
  + d: detection rate nhỏ nhất chấp nhận được của mỗi bộ phân lớp trong cascade.
  + Ftarget: false positive rate lớn nhất chấp nhận được của cascade.
* Bước 2: Khởi gán
  + P = tập các mẫu là khuôn mặt (positive.)
  + N = tập các mẫu không phải là khuôn mặt (negative.)
  + F0 = 1.0; D0 = 1.0
  + i = 0. //Chỉ số của bộ phân lớp trong cascade.
* Bước 3:Trong khi mà Fi> Ftarget:
  + i = i +1.

//Xây dưng bộ phân lớp thứ i trong cascade.

* + ni = 0. //Số bộ phân lớp yếu của bộ phân lớp thứ i
  + Fi = Fi-1.
  + Trong khi mà Fi> f \* Fi-1
    - ni = ni + 1.
    - Thuật toán Adaboost: Dùng bộ P và N để huấn luyện 1 bộ phân lớp với ni đặc trưng (ni bộ phân lớp yếu.)
    - Tính Fi và Di của cascade hiện tại thông qua validation set.
    - Giảm ngưỡng của bộ phân lớp thứ i cho tới khi cascade hiện tại đạt detection rate ít nhất là d\*Di-1(để ý là Ficũng sẽ thay đổi theo.)
  + N = 0.
  + Nếu Fi> Ftarget: Đưa những mẫu không phải là khuôn mặt vào cascade hiện tại và tiến hành detect. Những mẫu mà bị nhận dạng là khuôn mặt sẽ được đưa vô N.

*Nói thêm về thuật toán:*

Như đã nói nguyên tắc xử lý của cascade là nếu có một bộ phân lớp nào đó nói mẫu không phải là khuôn mặt thì loại bỏ ngay, còn nếu bộ phân lớp nói đó là khuôn mặt thì ta chuyển cho bộ phân lớp phía sau. Do đó, ta thấy trong thuật toán, khi xây các bộ phân lớp của cascade, tập P (ứng với tập ảnh khuôn mặt) luôn không đổi; còn tập N (ứng với tập ảnh không phải khuôn mặt) thì thay đổi: Giả sử ta đã xây dựng được cascade đến bộ phân lớp thứ i. Ta sẽ đưa các ảnh không phải khuôn mặt vô cascade hiện thời và tiến hành detect; các ảnh mà bị cho là khuôn mặt sẽ được đưa vào tập N để xây dựng tiếp bộ phân lớp thứ i+1.

# CÀI ĐẶT

## Phát biểu bài toán

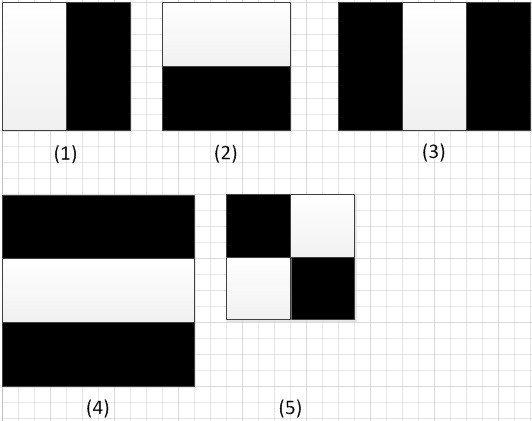
Từ một ảnh đầu vào, yêu cầu bài toán là xác định tập các vị trí và kích thước mà khuôn mặt người xuất hiện ở đó. Ở đây, ảnh đầu vào có thể ở dạng gray scale hoặc ảnh màu và tập các vị trí và kích thước kết xuất sẽ được thể hiện một cách trực quan thông qua các đường bao hình chữ nhật trên ảnh.

## Giải pháp

Dùng đặc trưng Haar-like để xây dựng nên các bộ phân lớp yếu để phát hiện khuôn mặt. Từ các bộ phân lớp này, ta sẽ dùng thuật toán Adaboost để xây dựng và huấn luyện nên bộ phân lớp mạnh.

### Các bộ phân lớp yếu

Để xây dựng nên các bộ phân lớp yếu, ta dùng 5 đặc trưng Haar-like cơ bản như sau.



Các dạng haar-like trên thể hiện các đặc trưng đường và cạnh cơ bản nhất. Tuy nhiên, theo báo cáo của [2], các đặc trưng này thể hiện đầy đủ đặc điểm một khuôn mặt có khung nhìn trực diện. Bên cạnh đó, để có thể mềm dẻo với các khuôn mặt hơi nghiêng hoặc có các góc quay khác nhau trong ảnh, đặc trưng Haar-like (5) đã được thêm vào.

### Tiền xử lý

Để phục vụ cho việc rút trích đặc trưng Haar-like, bước tiền xử lý khá đơn giản. Các ảnh đầu vào đòi hỏi phải ở dạng mức xám, do đó, tất cả ảnh màu đầu vào sẽ được chuyển hết về dạng mức xám theo công thức

**O(x,y)= IR(x,y) \* 0.287 + IG(x,y) \* 0.599 + IB(x,y) \* 0.114**

Với O(x,y) giá trị pixel đầu ra,

IR(x,y); IG(x,y); IB(x,y) lần lượt là các giá trị đỏ, lục và xanh dương của pixel đầu vào.

### Xây dựng chuỗi các bộ phân lớp

Dùng thuật toán Adaboost để xây dựng một chuỗi các bộ phân lớp như mô tả ở phần 1.3. Phần cài đặt dựa hoàn toàn trên những mô tả lý thuyết, tuy nhiên, có một vài điểm cần chú ý như sau:

* Trong quá trình huấn luyện cascade, phần 1.3.3 có đề cập đến 2 độ đo quan trọng là detection rate và false positive rate. Tuy nhiên, trong việc hiện thực, mô hình chỉ sử dụng false positive rate để huấn luyện và giá trị này được nhập vào từ người dùng. Giá trị này sẽ ảnh hưởng khá lớn đến hiệu quả của mô hình. Nếu quá nhỏ, mô hình có thể đạt được tỷ lệ nhận diện sai rất thấp, đồng thời cũng đối mặt với nguy cơ bỏ qua vị trị là khuôn mặt. Ngược lại mô hình sẽ đưa ra nhiều kết quả mà vị trí đó không phải là khuôn mặt. Từ các báo cáo gần đầy, trong đó [3] đã khẳng định sự hiệu quả khi chọn một tỉ lệ trung dung 0.5.
* Số bộ phân lớp trong một cascade sẽ được cài đặt để cho người dùng lựa chọn. Nếu giá trị này quá thấp hoặc quá cao, mô hình không có khả năng nhận diện được khuôn mặt hoặc tỷ lệ từ chối sai sẽ tăng lên rất cao.
* Trong việc sử dụng sub-window để phát hiện khuôn mặt, ý tưởng lý thuyết là dịch chuyển sub-window này đến hết ảnh sau đó tăng kích thước của nó lên. Điều này làm tăng độ phức tạp tính toán vì kéo theo đó là ta phải tăng kích thước cũng như điều chỉnh vị trí của các cửa sổ đặc trưng Haar-like. Với một bộ huấn luyện vài ngàn ảnh, cách tiếp cận này không khả thi. Vì vậy, thay vì tăng kích cỡ sub-window ta giảm dần kích thước của ảnh với tỷ lệ 0.8. Như vậy với sub-window 24x24, ảnh đầu vào 384x288 sẽ trải qua 12 cấp giảm tỷ lệ[2].

### Hậu xử lý

Với cách giảm tỷ lệ của ảnh thật như trên, một vấn đề gặp phải khi kết xuất là các hình vuông xác định khuôn mặt có khả năng chồng lấp lên nhau trên cùng một khuôn mặt. Có 2 dạng chồng lấp: 2 hình vuông giao nhau với tỷ lệ lớn và 2 hình vuông nằm lọt thỏm vào nhau.

* Hai hình vuông giao nhau

Đối với hai hình vuông chồng lên nhau, nếu giao trên 60% kích thước của hình vuông nhỏ hơn, ta lấy phần hội của 2 hình.

* Hai hình vuông lọt thỏm vào nhau

Trường hợp này, ta sẽ xóa đi hình vuông nhỏ bên trong, chỉ giữ lại hình vuông bên ngoài.

# THỰC NGHIỆM

## Dữ liệu

Dữ liệu huấn luyện và kiểm thử được lấy từ cơ sở dữ liệu của MIT – CMU, CMU/VASC image database.

### Dữ liệu huấn luyện

Bao gồm 3 bộ dữ liệu:

* Bộ huấn luyện có nhãn: gồm 10000 ảnh có kích thước 24x24 được đánh nhãn 0/1 (là khuôn mặt/không là khuôn mặt).
* Bộ xác thực có nhãn: gồm 4832 ảnh kích thước 24x24 được đánh nhãn 0/1.
* Bộ bootstrap: có 7323 ảnh có kích thước khác nhau. Những ảnh này không chứa mặt người.

### Dữ liệu kiểm thử

Thực hiện trên 4 bộ dữ liệu có những đặc trưng sau:

* Test set A: có 40 ảnh. Các ảnh mặt người với các kích thước to nhỏ khác nhau trong cùng một ảnh, phần nền và phần mặt người không phân biệt rõ ràng ở chế độ ảnh xám
* Test set B: có 24 ảnh. Cũng bao gồm đặc điểm như bộ A, nhưng mức độ có phần khó hơn, đặc biệt là sự gây nhiễu của nền. Ngoài ra, có một số ảnh khuôn mặt là hình vẽ.
* Test set C: 55 ảnh. Các khuôn mặt khá trực diện, không bị nhiễu bởi nền. Các phần của khuôn mặt rõ ràng và dễ phân biệt.
* Test set D: 50 ảnh. Các khuôn mặt xoay theo nhiều hướng khác nhau.

Số khuôn mặt trong mỗi bộ là 160 – 200.

## Kịch bản

Ở đây, các tham số cần phải khảo sát ngoài kết quả của quá trình phát hiện khuôn mặt là tỷ lệ false positive và số bộ phân lớp trong một cascade. Tuy nhiên, như đã nói, tỷ lệ false positive đã được khảo sát và làm rõ trong [3] nên nhóm sẽ tập trung vào việc tìm ra số bộ phân lớp tối ưu trong cascade (số node trong cascade) và kết quả của mô hình.

### Số bộ phân lớp trong cascade

Do thời gian chạy thuật toán là khá lâu, khoảng 7 tiếng để huấn luyện cho một cascade 30 node, và giới hạn trong phạm vi của đồ án môn học nên nhóm chỉ khảo sát dựa trên 4 giá trị, và thu được kết quả sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 | 20 | 30 | 40 |
| False positive rate | 12% | 39% | 32% | 30% |
| True positive rate | 2% | 52% | 43% | 37% |

Kết quả được kiểm thử trên bộ test set A, tỷ lệ nhận diện đúng cao nhất thuộc về số node là 20. Ta dễ dàng rút ra được một vài nhận xét như sau:

Ở số node là nhỏ, các bộ phân lớp không đủ để nhận diện hoàn chỉnh một khuôn mặt. Tỷ lệ từ chối là khuôn mặt rất cao, điều này dẫn đến tỷ lệ nhận diện sai cũng thấp. Tuy nhiên kéo theo đó là hiệu quả mô hình đạt được cũng thấp (2%).

Số node càng nhiều, cũng gặp phải tình trạng tương tự như vậy nhưng lý do là vì để chấp nhận một sub-window là khuôn mặt, nó phải trải qua đến 40 bộ phân lớp. Việc đi qua một số lượng lớn như vậy dẫn đến việc chấp nhận một mẫu là khuôn mặt sẽ khó khăn hơn, dẫn đến kết quả từ chối cũng cao hơn.

Ở số node là 20, ta đạt được tỷ lệ chính xác tối ưu 52%. Và khi thử điều chỉnh giá trị này dao động xung quanh 20, nhóm tìm được giá trị tối ưu là 22 node với độ chính xác 55%.

### Kết quả trên các bộ test

Với số node tìm được là 22, tỷ lệ false positive là 0.5, ta thực hiện kiểm thử trên 4 bộ dữ liệu đã mô tả. Kết quả thu được như sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Bộ test A** | **Bộ test B** | **Bộ test C** | **Bộ test D** | **Trung bình** |
| False positive | 40 % | 42 % | 12.4 % | 3.5% | 24.5% |
| True positive | 55 % | 49 % | 94 % | 28 % | 57 % |

Nhận thấy, kết quả phản ánh đặc trưng khác nhau của các bộ test. Trong đó, mức độ nhiễu của khuôn mặt và sự tương phản giữa khuôn mặt với nền ảnh hưởng đáng kể đến kết quả dự đoán trong bộ A và B. Bộ C cho kết quả cao nhất nhờ vào sự phân biệt rõ ràng giữa mặt với nền. Đối với bộ D, việc xoay khuôn mặt theo nhiều trục khác nhau và nhiều chiều khác nhau làm kết quả dự đoán thấp nhất trong 4 bộ dữ liệu.

# KẾT LUẬN

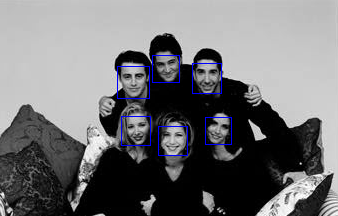
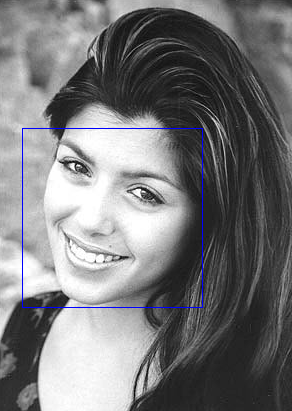
## Nhận xét về mô hình

### Ưu điểm

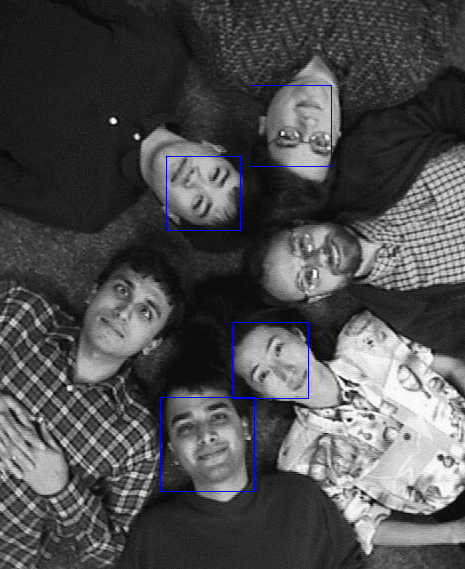
Với kết quả nhận diện đến 94% với bộ dữ liệu C, có thể nói, mô hình thực hiện tốt đối với các đặc trưng đã được học. Bộ đặc trưng Haar-like mô tả tốt đặc điểm của khuôn mặt chính diện.

Thông qua thực nghiệm, nhóm cũng nhận thấy mô hình có khả năng nhận dạng được các khuôn mặt có góc quay so với trục dọc là 45 độ và trục ngang là 15 độ.

Nhờ vào bộ Haar-like tương đối tốt, trong nhiều trường hợp mô hình vẫn nhận ra khuôn mặt bị dựng ngược.



Khuôn mặt trực diện và có góc quay trục dọc nhỏ hơn 45 độ



Khuôn mặt bị dựng ngược

### Khuyết điểm

Hiệu quả của bộ phân lớp phụ thuộc rất lớn vào bộ đặc trưng Haar-like đã chọn và việc thực nghiệm số bộ phân lớp như đã mô tả trong kịch bản kiểm thử.

Sự không rạch ròi về mức xám giữa nền và khuôn mặt cũng làm cho hiệu quả giảm đáng kể.

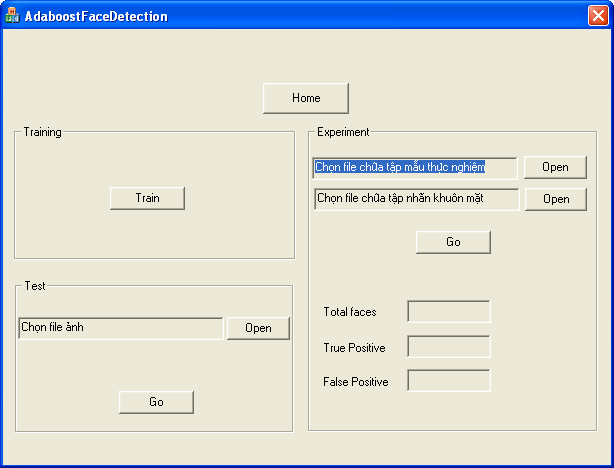
Nhược điểm lớn nhất của mô hình là không phản ánh được khuôn mặt nghiêng (chỉ thấy một bên mắt) và những khuôn mặt bị quay ngang.

## Hướng phát triển

Với sự không phản ánh được đặc trưng nghiêng của khuôn mặt, hướng cải tiến trong tương lai là áp dụng thêm một vài đặc trưng Haar-like để có thể giải quyết nhược điểm này. Bên cạnh đó, nhóm cũng sẽ xây dựng một cơ chế động, cho phép xác định số node của một cascade.

# PHỤ LỤC: HƯỚNG DẪN SỬ DỤNG CHƯƠNG TRÌNH

Thư mục release chứa đầy đủ các file cấu hình cũng như dữ liêu cần chạy.



Giao diện chương trình

* Huấn luyện

Để huấn luyện lại một cascade, đầu tiên ta vào file cascade.txt và thay toàn bộ nội dung trong file bằng số 0. (Ghi số 0 vào file và lưu lại).

Vào file options.txt thay đổi giá trị “số node tối đa trong cascade” theo mong muốn

mong muốn.

Mở ứng dụng, chọn “Home” để load file options.txt rồi bấm “Train” và … chờ đợi.

* Kiểm thử

Để đảm bảo chương trình chạy được, bộ cascade phải được huấn luyện. Ở đây, đã cung cấp sẵn file cascade.txt trong thư mục cascade. Việc cần làm là sao chép file này ra vị trí thư mục cùng cấp với ứng dụng. Thêm vào đó giá trị “số node tối đa trong cascade” của file options.txt phải lớn hơn hoặc bằng số node trong file cascade.txt (ở đây là 22 node).

Chọn file ảnh và “Go”. Xem kết quả.

* Thực nghiệm

Giống quá trình kiểm thử, nhưng thay vì chọn file ảnh ta chọn 2 file.

* File chứa tập mẫu thực nghiêm: testsetA\_filename.txt
* File chứa tập nhãn khuôn mặt: testsetA\_label.txt

# TÀI LIỆU THAM KHẢO

[1] Christopher M. Biship, *Pattern Recognition and Machine Learning,* Springer (2007)

[2] Paul Viola, Michael J. Jones, *Robust Real-Time Face Detection*, International Journal of Computer Vision 57(2), 137–154, 2004

[3] Jianxin Wu, James M.Rehg, Matthew D.Mullin, Learning a Rare Event Detection Cascade by Direct Feature Selection.