TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

ĐINH BÁ THẮNG – ĐẶNG BÁC VĂN

TÌM HIỂU CÁC KỸ THUẬT ÁP DỤNG CHO BÀI TOÁN NHẬN DẠNG KÝ HIỆU NGƯỜI CÂM

KHÓA LUẬN CỬ NHÂN TIN HỌC

TP.HCM, NĂM 2005

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

ĐINH BÁ THẮNG – 0112446
ĐẶNG BÁC VĂN – 0112454

TÌM HIỂU CÁC KỸ THUẬT ÁP DỤNG CHO BÀI TOÁN NHẬN DẠNG KÝ HIỆU NGƯỜI CÂM

KHÓA LUẬN CỬ NHÂN TIN HỌC

GIÁO VIÊN HƯỚNG DẪN
T.S DƯƠNG ANH ĐỨC
Th.S NGUYỄN TRI TUẤN

NIÊN KHÓA 2001 - 2005

LỜI NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

LỜI NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Lời cảm ơn

Chúng em xin chân thành cảm ơn Khoa Công nghệ Thông tin, trường Đại học Khoa học Tự nhiên TP.HCM đã tạo điều kiện cho chúng em thực hiện đề tài luận văn tốt nghiệp này.

Chúng em xin gửi lời cảm ơn sâu sắc đến Tiến sĩ Dương Anh Đức, Thạc sĩ Nguyễn Tri Tuấn và Thạc sĩ Lê Đình Duy đã tận tình hướng dẫn chúng em trong suốt thời gian thực hiện đề tài. Qua thời gian được các thầy hướng dẫn, chúng em đã biết cách làm việc khoa học hơn, biết cách "Khi em viết ra một cái gì không phải của em thì em phải cho người khác biết cái đó là của ai", cũng như "Khi em nói kết quả em đạt được là khá tốt thì phải nó rõ tốt là tốt thế nào".

Chúng em xin chân thành cảm ơn các Thầy Cô trong Khoa đã truyền đạt cho chúng em những kiến thức quý báu trong những năm học vừa qua, giúp chúng em có được một nền tảng lý thuyết vững chắc để có thể tiếp tục theo học hay đi tìm việc làm.

Chúng con xin được bày tỏ lòng biết ơn sâu sắc đối với Ông Bà, Cha Mẹ, người luôn luôn quan tâm chăm sóc cả về mặt vật chất lẫn tinh thần, luôn tạo điều kiện cho chúng con chuyên tâm học tập và nghiện cứu.

Cuối cùng, xin được nói lời cảm ơn chân thành đến các anh chị và các bạn đã giúp đỡ, khích lệ cũng như phê bình, góp ý, giúp chúng em hoàn thành công việc một cách tốt nhất.

Tuy chúng em đã nổ lực hết sức mình và hoàn thành luận văn, nhưng chắc chắn luận văn vẫn còn nhiều thiếu sót. Chúng em rất mong nhận được sự góp ý, chỉ bảo tận tình của các Thầy Cô và các bạn để chúng em có thể tiếp tục thực hiện những gì do chính chúng em viết ra trong mục "hướng phát triển" của khóa luận này.

Thành phố Hồ Chí Minh, Tháng 7/2005

Nhóm SV thực hiện

Đinh Bá Thắng – Đặng Bác Văn

Lời nói đầu

Sự ra đời của máy tính đã giúp ích rất nhiều cho công việc và cuộc sống của con người. Với máy tính, con người có thể soạn thảo văn bản, nghe nhạc, xem phim, thiết kế đồ họa, xử lý ảnh, biên tập phim ... Tuy nhiên, việc giao tiếp giữa con người và máy tính phụ thuộc chủ yếu vào bàn phím và chuột, và hầu như con người luôn phải ngồi trước máy tính. Dần dần, các nhà sản xuất thấy được sự bất tiện và đã tạo ra bàn phím và chuột không dây với mong muốn mang lại sự tự do hơn cho người dùng. Tuy nhiên, bàn phím không dây thì vẫn là bàn phím, con người cũng chỉ có thể tương tác với máy tính thông qua hệ thống 104 phím. Con người chỉ thật sự được "giải phóng" khi việc tương tác với máy tính được thực hiện thông qua các cử chỉ trong cuộc sống hàng ngày, tức là máy tính phải "hiểu" được các cử chỉ của con người. Đó chính là vấn đề đặt ra cho bài toán nhận dạng và phân loại cử chỉ. Cho đến thời điểm hiện nay, dù đã có nhiều cách tiếp cận khác nhau cho bài toán này, nhưng dường như vẫn chưa có một hệ thống nhận dạng cử chỉ nào thực sự hiệu quả.

Bên cạnh đó, bài toán nhận dạng mặt người đang đạt được một kết quả rất khả quan với mô hình *Cascade of Boosted Classifiers* do *Viola* và *Jones* [1] đề nghị. Mô hình này đạt hiệu quả cao cả về độ chính xác lẫn thời gian nhận dạng. *Eng Jon* [14] đã áp dụng mô hình này lên bài toán nhận dạng bàn tay và cũng đạt được kết quả tốt.

Mục tiêu của khóa luận này là thử áp dụng mô hình *Cascade of Boosted Classifiers* lên bài toán phân loại cử chỉ với hi vọng nó cũng sẽ đạt được kết quả tốt như trên bài toán nhận dạng mặt người và nhận dạng bàn tay. Luận văn được trình bày trong 6 chương với bố cục như sau:

- **Chương 1-Mở đầu**: Nêu lên tầm quan trọng của bài toán phân loại cử chỉ và mô tả sơ bộ phạm vi bài toán mà khóa luận này sẽ giải quyết. Đồng thời giới điểm sơ qua các cách tiếp cận hiện có với các ưu khuyết điểm của chúng và giới thiệu về mô hình sử dụng trong khóa luận này.

- **Chương 2-Giới thiệu bài toán phân loại cử chỉ**: Phát biểu cụ thể và mô tả chi tiết phạm vi bài toán sẽ giải quyết, giải thích tại sao lại chọn mô hình *Cascade of Boosted Classifiers*.
- **Chương 3-Cơ sở lý thuyết**: Trình bày về AdaBoost, Haar Feature, mô hình Cascade of Classifiers, khái niệm Integral Image, từ đó hình thành nên cấu trúc Cascade of Boosted Classifiers. Tiếp đó là phần giới thiệu các ứng dụng của mô hình và một số nhận xét, đánh giá.
- **Chương 4-Áp dụng mô hình Cascade of Boosted Classifiers**: Trình bày chi tiết cách áp dụng mô hình cascade lên bài toán phân loại cử chỉ.
- **Chương 5-Kết quả thử nghiệm**: Giới thiệu về tập huấn luyện, cách thu thập mẫu, cách tiến hành và kết quả huấn luyện, đồng thời so sánh đối chiếu với kết quả của người khác.
- **Chương 6-Tổng kết**: Tóm tắt các kết quả nghiên cứu, những gì đã đạt được, những gì còn hạn chế và nêu ra hướng phát triển trong tương lai.

Khoa CNTT - ĐHKHTN Tp.HCM

Mục Lục

Danh sách hình

Danh sách bảng

Chương 1 Mở đầu

Mặc dù nền công nghệ thông tin vẫn phát triển liên tục với tốc độ vũ bão nhưng chúng ta vẫn còn một chặng đường rất dài để có thể giao tiếp một cách hoàn toàn tự nhiên với máy tính như giao tiếp giữa con người với nhau. Cách giao tiếp tự nhiên nhất với máy tính chính là dùng giao tiếp thông qua tiếng nói và cử chỉ. Trong khi lĩnh vực nhận dạng tiếng nói đã đạt được những thành công đáng kể trong vòng 10 năm gần đây thì lĩnh vực nhận dạng cử chỉ vẫn còn tụt lại phía sau. Tuy nhiên, ngôn ngữ cử chỉ lại chính là ngôn ngữ chuyển tải thông tin giữa người và người một cách trọn vẹn nhất. Nếu giả sử chúng ta phát triển được một hệ thống nhận dạng tiếng nói kết hợp với nhận dạng cử chỉ thì chúng ta hoàn toàn có thể thay thế chuột hay bàn phím bằng một hệ thống sử dụng ngôn ngữ tự nhiên điều khiển máy tính thông qua bộ giao tiếp bằng lệnh dựa trên cử chỉ và giọng nói.

Trong cuộc sống hàng ngày, nhận dạng cử chỉ có thể giúp cho việc giao tiếp giữa người bình thường với người khiếm thính dễ dàng hơn, vì máy tính sẽ giúp người bình thường không cần hiểu hệ thống kí hiệu của người khiếm thính, một ngôn ngữ không phải dễ học. Trong công nghiệp và sản xuất, chỉ cần trang bị cho các robot hệ thống camera, việc điều khiển robot sẽ trở nên đơn giản hơn bao giờ hết. Trong lĩnh vực đồ họa 3 chiều, ta có thể dùng một số động tác yêu cầu máy tính xoay mô hình theo ý muốn của mình.Trong công việc văn phòng, nhận dạng cử chỉ giúp ta có thể yêu cầu máy tính thực thi một chương trình, mở một bài hát, gởi một lá thư ... chỉ với một vài cử chỉ ra hiệu từ xa. Trong lĩnh vực giải trí, các trò chơi thực tế ảo (Virtual Reallity) – các trò chơi mà người chơi điều khiển hành động nhân vật bằng chính hành động của mình – luôn có sức cuốn hút đặc biệt với người chơi.

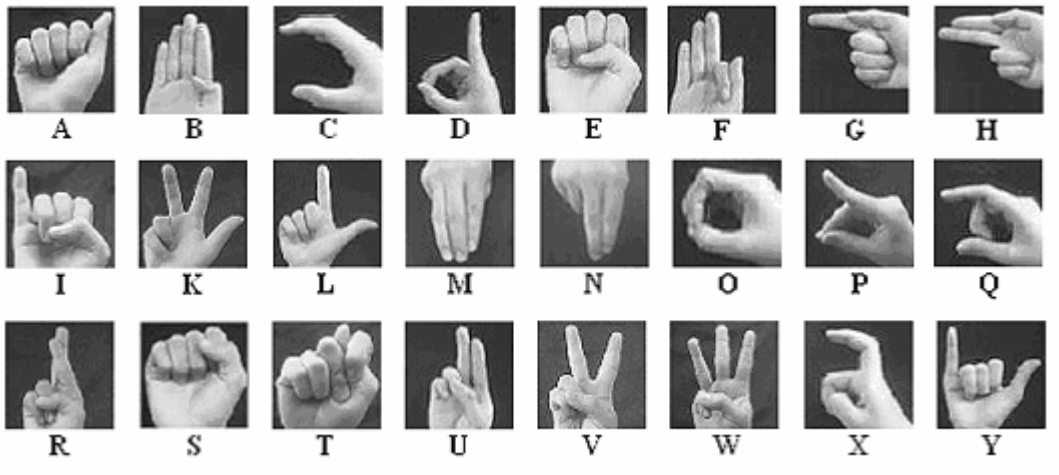
Cùng với nhận dạng âm thanh, tiếng nói và xử lý ngôn ngữ, nhận dạng cử chỉ giúp máy tính thực sự trở nên "người" hơn – điều mà các nhà khoa học đang miệt mài nghiên cứu với hi vọng sẽ đạt được trong một tương lai không xa.

Tuy nhiên, chính các ứng dụng to lớn trên khiến cho bài toán thực sự là một thách thức. Để có thể được đưa vào sử dụng, một hệ thống trước hết phải hiểu đúng các cử chỉ của con người, tức là nó phải nhận dạng được chính xác các cử chỉ đó. Một hệ thống xoay mô hình không đúng với ý của chuyên viên đồ họa, hay một con robot chuyên làm sai chỉ thị thì khó có thể được chấp nhận.

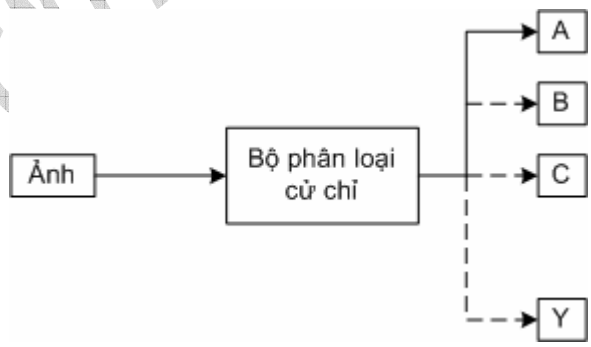
Bên cạnh đó, để có thể tương tác với người dùng, hệ thống nhận dạng xây dựng phải là hệ thống thời gian thực, phải xử lý nhanh. Một chuyên viên đồ họa sẽ không chấp nhận một hệ thống cần đến 30 giây để xoay mô hình của họ. Một con robot sẽ không được chấp nhận nếu nó cần đến 20 giây để hiểu ra rằng nó phải làm một việc gì đó ngay lập tức. Hay như trong một cuộc giao tiếp, nếu một hệ thống phải mất đến 10 giây cho mỗi cử chỉ mà người khiếm thính ra dấu thì nó cũng không thể được chấp nhận.

Bài toán nhận dạng cử chỉ có thể chia làm 2 loại chính: nhận dạng cử chỉ tĩnh và nhận dạng cử chỉ động. Cử chỉ tĩnh là các cử chỉ ứng với một tư thế cố định của một bàn tay, còn cử chỉ động là chuyển động theo một quỹ đạo nhất định của một hay hai bàn tay. Nhận dạng cử chỉ động bao hàm cả nhận dạng cử chỉ tĩnh và một số xử lý trên chuyển động nên hết sức phức tạp. Khóa luận này chỉ tập trung vào bài toán nhận dạng cử chỉ tĩnh.

Hệ thống phân loại cử chỉ xây dựng trong khóa luận này là một hệ thống có khả năng nhận dạng và phân loại 24 cử chỉ ứng với 24 kí tự trong bảng chữ cái (trừ chữ J và Z do đòi hỏi chuyển động của bàn tay)



Hình 1 - Hệ thống 24 cử chỉ



Hình 2 - Bộ phân loại cử chỉ

Bộ phân loại có thể được ứng dụng để xây dựng hệ thống hoạt động dựa trên 1 webcam dùng để theo dõi chuyển động của bàn tay. Khi người dùng ra dấu với 1 cử chỉ, hệ thống sẽ rút trích một khung hình chính trong số các khung hình mô tả toàn bộ quá trình ra dấu của người dùng được webcam ghi lại và đưa nó vào bộ phân loại cử chỉ. Bộ phân loại sẽ cho kết quả phân loại là nó thuộc về cử chỉ nào hay nó không nằm trong hệ thống 24 cử chỉ. Từ đó có thể phát triển thêm để hệ thống thực hiện một số chức năng cụ thể khi nhận được các cử chỉ tương ứng từ người dùng.

Bài toán đặt ra 2 khó khăn lớn: nhận dạng không những phải chính xác mà còn phải nhanh bởi vì hệ thống hoạt động theo thời gian thực. Yêu cầu về tính chính xác cũng là khó khăn của bất cứ một bài toán nhận dạng nào. Riêng đối với bài toán phân

loại cử chỉ thì việc phân loại chính xác lại càng khó khăn hơn bởi vì trong số 24 cử chỉ, có rất nhiều cử chỉ giống nhau (chẳng hạn như A, E, S và T).

Để có thể đạt độ chính xác cao, trước hết hệ thống phải có các đặc trưng (feature) tốt. Hệ thống phải biết chọn đặc trưng như thế nào để có thể biểu diễn tốt được thông tin đối tượng cần nhận dạng. Đồng thời, đặc trưng phải được tính toán nhanh, để không làm chậm công việc nhận dạng. Thêm vào đó, hệ thống phải có phương pháp học hiệu quả, có khả năng nhận dạng tốt các mẫu mới chứ không chỉ làm tốt trên các mẫu đã học (vấn đề về generalization).

Để có thể đạt được các mục tiêu trên, đã có nhiều cách tiếp cận được đưa ra. Freeman sử dụng đặc trưng *Orientation Histogram* [15] với thời gian tính toán nhanh, nhưng lại đòi hỏi mẫu nhận dạng phải là mẫu chụp cận cảnh của bàn tay. Đồng thời cách này không áp dụng được khi hệ thống có các cử chỉ tương tự nhau, vì các cử chỉ tương tự nhau cho *Orientation Histogram* giống nhau.

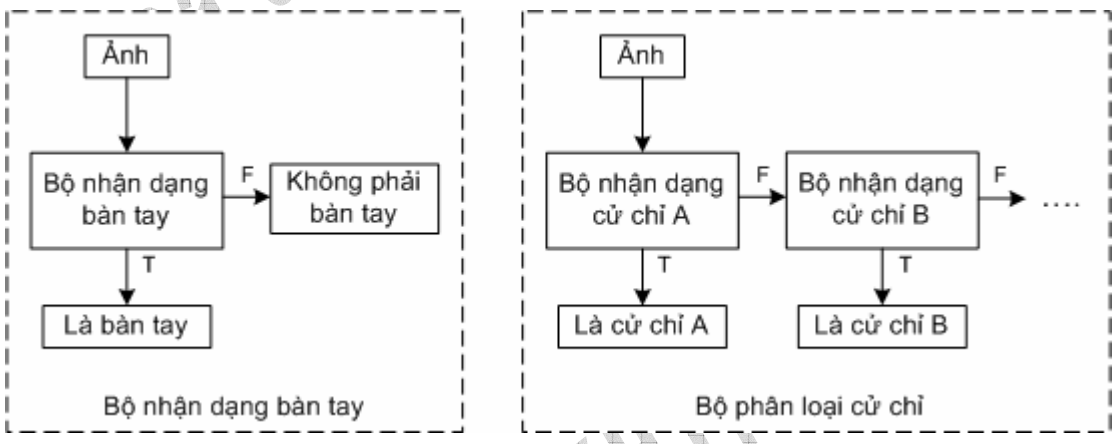
Bowden và Sarhadi [16] sử dụng mô hình *nonlinear PDM* (*nonlinear point distribution model*) để biểu diễn được nhiều thông tin về bàn tay. Nhưng việc huấn luyện bằng PDM nói chung khó đạt được sự vững chắc.

Ngoài ra, còn có cách tiếp cận dựa trên màu sắc của da trên bàn tay [17] bởi vì màu da tương đối thuần nhất. Đây cũng không phải một cách tiếp cận tốt vì các bộ nhận dạng xây dựng trên đặc trưng là màu da rất nhạy cảm với ánh sáng, và hệ thống sẽ nhận dạng sai khi mẫu đưa vào có chứa các đối tượng khác có màu giống với màu da..

Nhìn chung, trong các cách tiếp cận trên đều có chung một hạn chế là không thể đạt được sự cân đối giữa khả năng nhận dạng và thời gian xử lý. Trong khi đó, hiện có một mô hình đang đạt được sự cân đối giữa 2 mặt này và hiện đang được *Viola* và *Jones* áp dụng rất thành công trong lĩnh vực nhận dạng mặt người: mô hình *Cascade of Boosted Classifiers* với đặc trưng sử dụng là *Haar Feature* [1] (mô hình *cascade* này do chính *Viola* và *Jones* đề nghị).

Cascade of Boosted Classifiers là một cấu trúc cây mà ở mỗi tầng là một classifier (bộ phân loại). Classifier này được xây dựng bằng thuật toán *AdaBoost* trên nguyên tắc sự kết hợp của nhiều *weak classifiers* (các bộ phân loại đơn giản chỉ cần có độ chính xác trên 50%) sẽ tạo ra một strong classifier (bộ phân loại có độ chính xác cao). Đặc trưng sử dụng là *Haar Feature*, là một tập các hình chữ nhật thể hiện mối liên hệ giữa các vùng ảnh với nhau (mô hình này sẽ được trình bày chi tiết trong chương 3). Bên cạnh đó, mô hình này cũng đã được Eng Jon áp dụng lên bài toán nhận dạng bàn tay [14] với kết quả khả quan.

Bài toán phân loại cử chỉ khá giống với bài toán nhận dạng bàn tay của Eng Jon. Một *bộ phân loại cử chỉ* có thể được xây dựng từ nhiều *bộ nhận dạng cử chỉ*, trong đó, mỗi bộ nhận dạng ứng với 1 cử chỉ cụ thể.



Hình 3 - Bộ phân loại được tạo thành từ sự kết hợp các bộ nhận dạng

Khóa luận này xây dựng các bộ nhận dạng cho từng cử chỉ theo mô hình *Cascade of Boosted Classfiers* và dựa theo bộ nhận dạng cho từng nhóm bàn tay của Eng Jon [14]. Do vấn đề của bài toán nhận dạng bàn tay, các bàn tay có thể ở nhiều tư thế khác nhau, Eng Jon đã sử dụng thuật toán K-mediod để tiến hành phân cụm cho tập huấn luyện và mỗi cụm như vậy sẽ được xây dựng một bộ nhận dạng cho nó. Còn đối với từng bộ phân loại cử chỉ, khóa luận này giới hạn bàn tay phải ở tư thế cố định và phải được chụp chính diện, do đó sẽ không cần đến thuật toán K-mediod.

Sau khi xây dựng được các bộ nhận dạng cho từng cử chỉ, bộ phân loại cử chỉ sẽ được xây dựng từ các bộ nhận dạng này. Tuy nhiên, chúng sẽ không được kết hợp một cách tuần tự như trong hình 3 mà bản thân các cử chỉ cũng được phân thành nhóm do sự giống nhau giữa một số cử chỉ trong hệ thống 24 cử chỉ. Sự giống nhau này sẽ được xác định thông qua kết quả của quá trình thử nghiệm.

Các phần tiếp theo của luận văn sẽ lần lượt trình bày các vấn đề liên quan. Chương 2 giới thiệu cặn kẽ hơn về bài toán, các vấn đề đặt ra, các cách tiếp cận và lý do tại sao chọn mô hình *cascade*. Chương 3 trình bày cơ sở lý thuyết của cách tiếp cận cascade, thuật toán *AdaBoost* và *Haar Feature* cùng với các ứng dụng thành công của chúng. Chương 4 là phần áp dụng mô hình lên bài toán phân loại cử chỉ, bao gồm cách huấn luyện , hệ thống các *Haar Features* sử dụng cùng với cấu trúc cây *cascade* và cách kết hợp các bộ nhận dạng cử chỉ để hình thành bộ phân loại. Chương 5 là kết quả thử nghiệm và chương 6 là các đánh giá và hướng phát triển của luận văn.

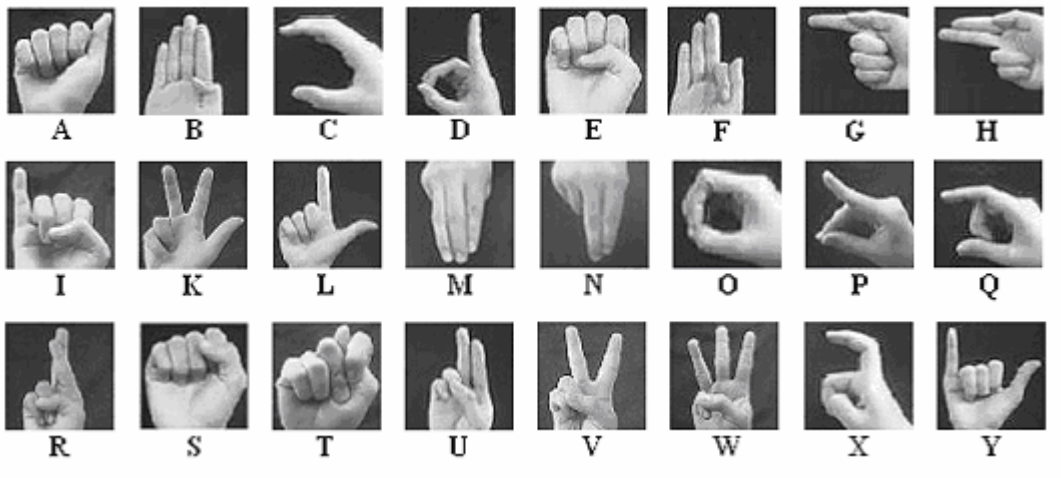
Chương 2 Giới thiệu về hệ thống phân loại cử chỉ

Như đã trình bày ở phần trên, những ứng dụng to lớn của bài toán phân loại cử chỉ mang lại nhiều khó khăn không nhỏ, việc giải quyết nó đòi hỏi kiến thức cả về nhận dạng lẫn máy học. Trong đó, 2 mục tiêu lớn sau cùng chính là độ chính xác và tốc độ. Hệ thống nhận dạng phải có được các đặc trưng tốt, chứa đựng được nhiều thông tin về đối tượng. Đối với các cử chỉ, tuy có thể sử dụng các đặc trưng về hình học như chiều của ngón tay hay đường bao, nhưng các đặc trưng này không phải lúc nào cũng rõ ràng và đủ tin cậy để phân biệt bàn tay với các đối tượng xung quanh, đặc biệt là dưới các điều kiện chiếu sáng khác nhau.

Đã có nhiều cách tiếp cận giải quyết vấn đề này. Freeman đã sử dụng đặc trưng *Orientation Histogram* [15] để nhận dạng cử chỉ. *Orientation Histogram* có ưu điểm là có thể tính toán nhanh và Freeman đã xây dựng được một hệ thống nhận dạng cử thời gian thực dựa trên đặc trưng này. Đồng thời, đặc trưng này giúp cho hệ thống hoạt động hiệu quả ở môi trường điều kiện chiếu sáng khác nhau vì *Orientation Histogram* tương đối độc lập với điều kiện chiếu sáng. Tuy nhiên, *Orientation Histogram* chỉ áp dụng được trên tập các cử chỉ hoàn toàn khác nhau, vì các cử chỉ giống nhau cũng sẽ cho *Orientation Histogram* tương tự nhau. Đồng thời, khi tiến hành nhận dạng, bàn tay phải chiếm gần trọn mẫu đưa vào, nếu mẫu đưa vào là một không gian lớn trong đó chứa bàn tay (kích thước nhỏ) thì histogram lấy được sẽ không phải đặc trưng của bàn tay, dẫn đến kết quả phân loại sai. Bowden và Sarhadi [16] thì sử dụng mô hình *nonlinear PDM* để biểu diễn được nhiều thông tin về bàn tay. Nhưng việc huấn luyện bằng PDM nói chung khó đảm bảo được tỉ lệ nhận dạng cao và không đủ tính tổng quát. Nolker và Nitter [18] sử dụng *Local Linear Mapping (LLN) Neural Network* để ánh xạ hình chụp 2 chiều về tọa độ 3 chiều theo *Parametric Self-Organizing Map* (*PSOM*). Từ đó lưu được nhiều thông tin về bàn tay, giúp có thể nhận dạng cử chỉ từ nhiều góc độ

khác nhau. Dù vậy, hệ thống này lại không đạt được hiêu quả về mặt thời gian. Nhìn chung các cách tiếp cận này không đạt được sự cân đối giữa 2 mục tiêu đề ra ban đầu.

Mục tiêu của khóa luận này là xây dựng được một hệ thống có khả năng phân loại nhanh và chính xác hệ thống 24 cử chỉ:



Hình 4 - Hệ thống 24 cử chỉ

Hệ thống phân loại gồm 3 phần chính:

1. Module huấn luyện: xây dựng bộ nhận dạng cho từng cử chỉ (Sign Detector). Bộ nhận dạng này được tổ chức theo cấu trúc *Cascade of Boosted Classfiers*, một *cascade tree* mà mỗi tầng là một *strong classifier* được xây dựng bằng *Gentle AdaBoost* [4] - một biến thể của thuật toán *AdaBoost*. Mỗi classifier này là một chuỗi các *weak classifiers* – các classifier đơn giản chỉ cần có độ chính xác trên 50% - mỗi weak classifier sẽ gồm 1 *haar feature* và 1 ngưỡng để phân loại. Kết quả trả ra của module này là các tập tin dữ liệu ứng với cấu trúc cascade tree tạo thành.
2. Module nhận dạng cử chỉ: tạo lại *cascade tree* từ tập tin của bộ nhận dạng đã xây dựng để tiến hành nhận dạng. Ví dụ sử dụng tập tin của bộ nhận dạng cử chỉ A thì ta sẽ được A-Detector, việc nhận dạng sẽ là nhận dạng một vùng ảnh có phải cử chỉ A hay không. Module này nhận vào một ảnh, trích ra tất cả các vùng ảnh với vị trí và kích thước khác nhau, đưa chúng vào cascade tree và trả ra các

hình chữ nhật ứng với các vùng ảnh được bộ nhận dạng đánh giá là cử chỉ. Việc nhận dạng ở khâu này được áp dụng thêm một số phương pháp heuristic nhằm tăng *detection rate* và giảm *false alarm* cho hệ thống.

3. Bộ phân loại cử chỉ: kết hợp các bộ nhận dạng cử chỉ để thực hiện phân loại. Bộ phân loại sẽ nhận vào một ảnh và cho biết trong ảnh đó có chứa những cử chỉ nào (cho biết vị trí của vùng ảnh tương ứng với các cử chỉ).

Lý do mà khóa luận này chọn *Cascaded of Boosted Classifiers* với *Haar Feature* là:

1. *Haar Feature* [2] phản ánh được tốt thông tin về đối tượng, đồng thời có thể tính toán nhanh nhờ khái niệm Integral Image [1] do Viola đưa ra.
2. *AdaBoost* chạy nhanh và giúp nâng cao tỉ lệ nhận dạng.
3. Cấu trúc *cascade* cho phép loại nhanh các mẫu background đơn giản ngay từ những stages đầu tiên, giúp rút ngắn thời gian nhận dạng, đồng thời đáp ứng tốt nhất với độ phức tạp gia tăng của các mẫu cần nhận dạng, loại nhanh các mẫu background có độ phức tạp thấp bằng các bộ phân loại đơn giản trước khi gọi đến các bộ phân loại phức tạp giúp giảm thiểu false alarm..

Tính hiệu quả của mô hình này đã được thực tế chứng minh thông qua thành công của Viola & Jones và Eng Jon. Ngoài ra, Mathias Kolsch và Matthew Turk [6,7] đã áp dụng kết hợp *AdaBoost* với biến đổi Fourier lên bài toán nhận dạng cử chỉ tĩnh, tuy chỉ tiến hành thử nghiệm trên tập cử chỉ hạn chế (chỉ có 8 cử chỉ) nhưng đã đạt được kết quả đáng khích lệ: *detection rate* đạt 95% với *false alarm* là 10^{-4}.

Trong chương tiếp theo, chúng ta sẽ tìm hiểu cụ thể về *AdaBoost*, *Haar Feature* và mô hình *Cascade of Classifiers*, từ đó tìm hiểu về *Cascade of Boosted Classifiers*.

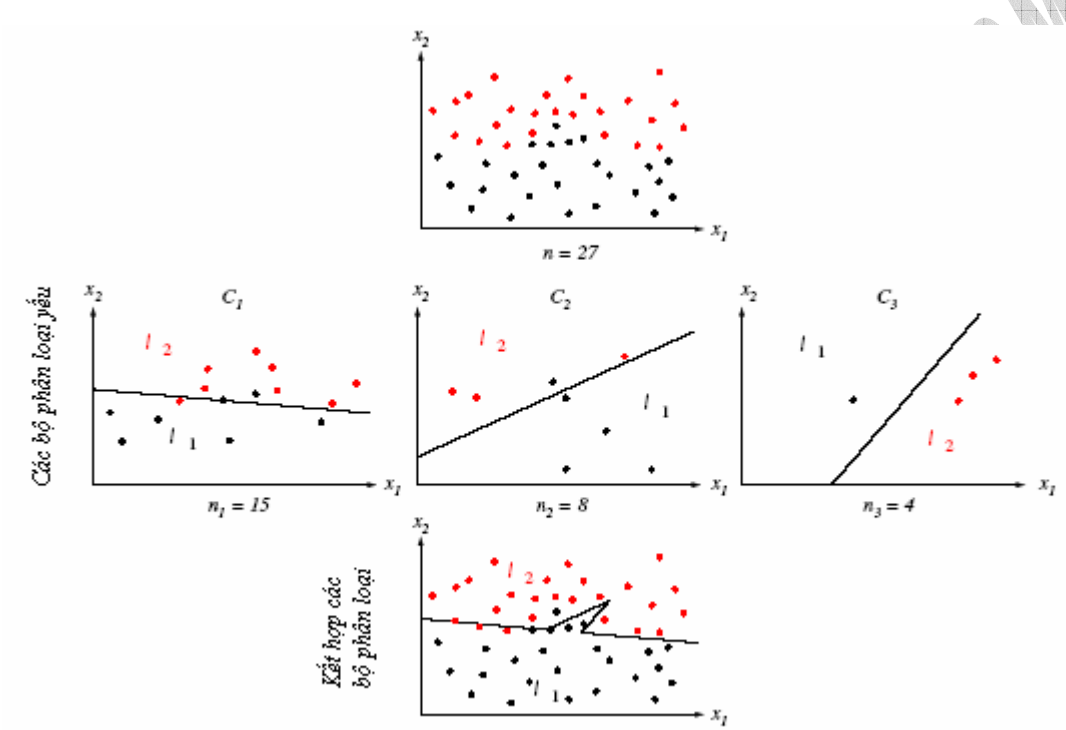
Chương 3 Các cơ sở lý thuyết

3.1 Tiếp cận Boosting

Boosting là kỹ thuật dùng để tăng độ chính xác cho các thuật toán học (Learning algorithm). Nguyên lý cơ bản của nó là 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".
Xét một bài toán phân loại 2 lớp (mẫu cần nhận dạng sẽ được phân vào 1 trong 2 lớp) với D là tập huấn luyện gồm có n mẫu. Trước tiên, chúng ta sẽ chọn ngẫu nhiên ra n_1 mẫu từ tập D ($n_1<n$) để tạo tập D_1. Sau đó, chúng ta sẽ xây dựng weak classifier đầu tiên C_1 từ tập D_1. Tiếp theo, chúng ta xây dựng tập D_2 để huấn luyện bộ phân loại C_2. D_2 sẽ được xây dựng sao cho một nửa số mẫu của nó được phân loại đúng bởi C_1 và nửa còn lại bị phân loại sai bởi C_1. Bằng cách này, D_2 chứa đựng những thông tin bổ sung cho C_1. Bây giờ chúng ta sẽ xây huấn luyện C_2 từ D_2.

Tiếp theo, chúng ta sẽ xây dựng tập D_3 từ những mẫu không được phân loại tốt bởi sự kết hợp giữa C_1 và C_2: những mẫu còn lại trong D mà C_1 và C_2 cho kết quả khác nhau. Như vậy, D_3 sẽ gồm những mẫu mà C_1 *và* C_2 hoạt động không hiệu quả. Sau cùng, chúng ta sẽ huấn luyện bộ phân loại C_3 từ D_3.

Bây giờ chúng ta đã có một strong classifier: sự kết hợp C_1, C_2 và C_3. Khi tiến hành nhận dạng một mẫu X, kết quả sẽ được quyết định bởi sự thỏa thuận của 3 bộ C_1, C_2 và C_3: Nếu cả C_1 và C_2 đều phân X vào cùng một lớp thì lớp này chính là kết quả phân loại của X; ngược lại, nếu C_1 và C_2 phân X vào 2 lớp khác nhau, C3 sẽ quyết định X thuộc về lớp nào.



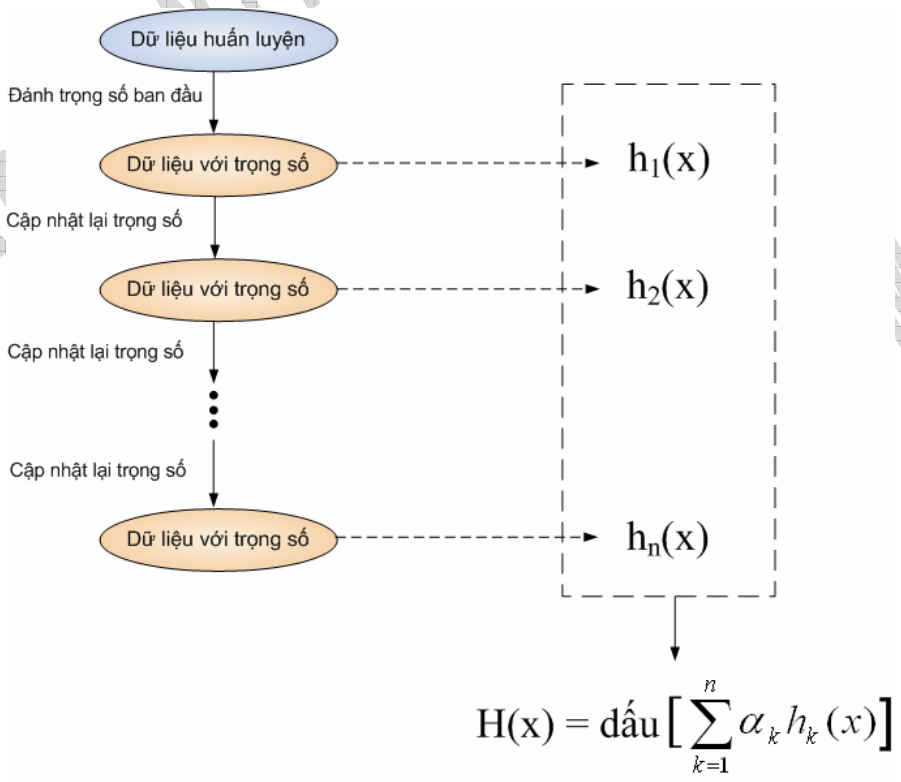
Hình 5 - Boosting

3.2 AdaBoost

Adaboost (Adaptive Boost) [5] là một tiếp cận boosting được *Freund* và *Schapire* đưa ra vào năm 1995. Adaboost cũng hoạt động trên nguyên tắc kết hợp tuyến tính các weak classifiers để có một strong classifier.

Là một cải tiến của tiếp cận boosting, Adaboost sử dụng thêm khái niệm 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 kế tiếp: *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 classifiers trước nó chưa làm tốt. Sau cùng, các weak classifiers sẽ được kết hợp tùy theo mức độ "tốt" của chúng để tạo dựng nên strong classifier.

Có thể hình dung một cách trực quan như sau: để biết một ảnh có phải là bàn tay hay không, ta hỏi T người (tương đương với T weak classifiers xây dựng từ T vòng lặp của boosting), đánh giá của mỗi người (tương đương với một weak classifier) chỉ cần tốt hơn ngẫu nhiên một chút (tỉ lệ sai dưới 50%). Sau đó, ta sẽ đánh trọng số cho đánh giá của từng người (thể hiện qua hệ số α), người nào có khả năng đánh giá tốt các mẫu khó thì mức độ quan trọng của người đó trong kết luận cuối cùng sẽ cao hơn những người chỉ đánh giá tốt được các mẫu dễ. Việc cập nhật lại trọng số của các mẫu sau mỗi vòng boosting chính là để đánh giá độ khó của các mẫu (mẫu càng có nhiều người đánh giá sai là mẫu càng khó).



Hình 6 - Strong classifier H(x) xây dựng bằng AdaBoost

Các weak classifiers $h_k(x)$ được biểu diễn như sau:

$$h_k(x) = \begin{cases} 1, p_k f_k(x) < p_k \theta_k \\ 0, otherwise \end{cases}$$

Trong đó:

- $x = (x_1, x_2, ..., x_n)$: vector đặc trưng của mẫu.
- θ: ngưỡng
- f_k: hàm lượng giá vector đặc trưng của mẫu
- p_k: hệ số quyết định chiều của bất phương trình

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

Thuật toán AdaBoost:

1. Cho một tập huấn luyện gồm n mẫu có đánh dấu $(x_1, y_1), (x_1, y_1),...., (x_n, y_n)$ với $x_k \in X = (x_{k1}, x_{k2}, ..., x_{km})$ là vector đặc trưng và $y_k \in \{-1, 1\}$ là nhãn của mẫu (1 ứng với object, -1 ứng với background).
2. Khởi tạo trọng số ban đầu cho tất cả các mẫu:

$$w_{1,k} = \frac{1}{n}$$

3. Xây dựng T weak classifiers

   Lặp $t = 1, ..., T$

   - Với mỗi đặc trưng trong vector đặc trưng, xây dựng một weak classifier h_j với ngưỡng θ_j và lỗi ε_j

$$\varepsilon_j = \sum_k^n w_{t,k} \left| h_j(x_k) - y_k \right|$$

   - Chọn ra h_j với ε_j nhỏ nhất, ta được h_t.

$$h_t : X \to \{1, -1\}$$

   - Cập nhật lại trọng số

$$w_{t+1,k} = \frac{w_{t,k}}{Z_t} \times \begin{cases} e^{-\alpha_t}, h_t(x_k) = y_k \\ e^{\alpha_t}, h_t(x_k) \neq y_k \end{cases}$$

   Trong đó:

- $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$
- Z_t: hệ số dùng để đưa $w_{t+1,k}$ về đoạn [0,1] (normalization factor)

4. Strong classifier xây dựng được

$$H(x) = \text{dấu}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

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 h_t thực hiện việc phân loại với lỗi ε_t 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 h_t, Adaboost sẽ tính giá trị α_t theo công thức ở trên. α_t cũng được chọn trên nguyên tắc làm giảm thiểu giá trị lỗi ε_t.

Hệ số α_t nói lên mức độ quan trọng của h_t:

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

$$H(x) = \text{dấu}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

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

- Trong công thức tính α_t:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$$

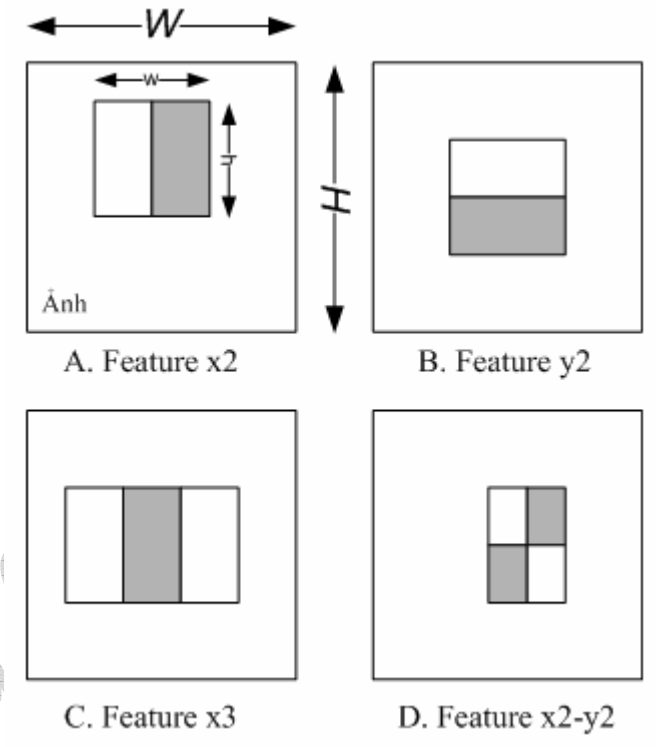
Dễ thấy giá trị α_t tỉ lệ nghịch với ε_j. Bởi vì h_t được chọn với tiêu chí đạt ε_j nhỏ nhất, do đó nó sẽ đảm bảo giá trị α_t lớn nhất. Công thức này do Freund và Schapire đưa ra [5].

Sau khi tính được giá trị α_t, Adaboost tiến hành cập nhật lại trọng số của các mẫu: tăng trọng số các mẫu mà h_t phân loại sai, giảm trọng số các mẫu mà h_t phân loại đúng. Bằng cách này, trọng số của mẫu phản ánh được mức độ khó nhận dạng của mẫu đó và h_{t+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 $H_t(x)$ 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*.

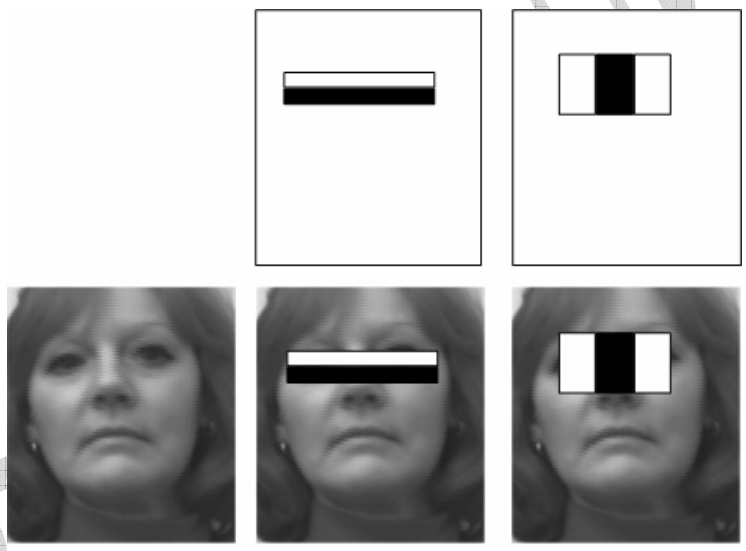
3.3 Haar Feature

Haar Feature [2] là một loại đặc trưng thường được dùng cho bài toán nhận dạng trên ảnh. Haar Feature được xây dựng từ các hình chữ nhật có kích thước bằng nhau, dùng để tính độ chênh lệch giữa các giá trị điểm ảnh trong các vùng kề nhau. Trong hình 7a và 7b, giá trị của feature cho bởi 1 ảnh bằng hiệu số giữa tổng các điểm ảnh thuộc 2 vùng hình chữ nhật sáng và tối. Trong hình 7c thì giá trị feature bằng tổng các điểm ảnh trong 2 vùng hình chữ nhật bên ngoài trừ cho tổng các điểm ảnh trong hình chữ nhật ở giữa. Trong hình 7d, giá trị feature bằng tổng các điểm ảnh nằm trong vùng 2 hình chữ nhật màu tối trừ cho tổng các điểm ảnh nằm trong 2 hình chữ nhật màu sáng.



Hình 7 - Haar Feature cơ bản

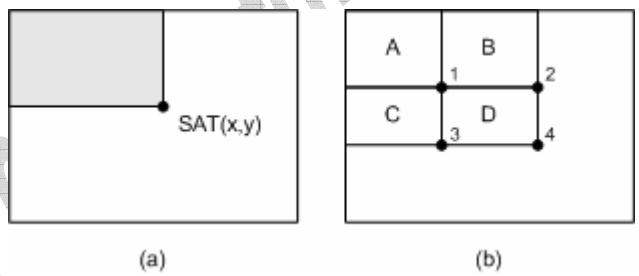
Lợi ích của Haar feature là nó diễn đạt được tri thức về các đối tượng trong ảnh (bởi vì nó biểu diễn mối liên hệ giữa các bộ phận của đối tượng), điều mà bản thân từng điểm ảnh không diễn đạt được.



Hình 8 - Haar Feature cho mặt người

Trong quá trình huấn luyện, số lượng xử lý trên các Haar Feature là rất lớn, việc tính tổng các điểm ảnh cho bởi từng feature làm cho thời gian xử lý tăng đáng kể. Để khắc phục điều này, *Viola* và *Jones* đã đưa ra khái niệm *Integral Image* [3] để tính toán nhanh cho khác feature cơ bản, *Lienhart* [4] kế thừa (gọi *Integral Image* là *SAT – Summed Area Table*) và đưa ra thêm khái niệm *RSAT – Rotated Summed Area Table* để tính toán nhanh cho các feature xoay 1 góc 45º.

Integral Image (hay SAT)



Hình 9 - SAT(x,y) và cách tính tổng các điểm ảnh trong một hình chữ nhật bất kì

Với định nghĩa integral image tại điểm (x,y) là:

$$SAT(x,y)=\sum_{x'\le x,y'\le y} I(x',y').$$

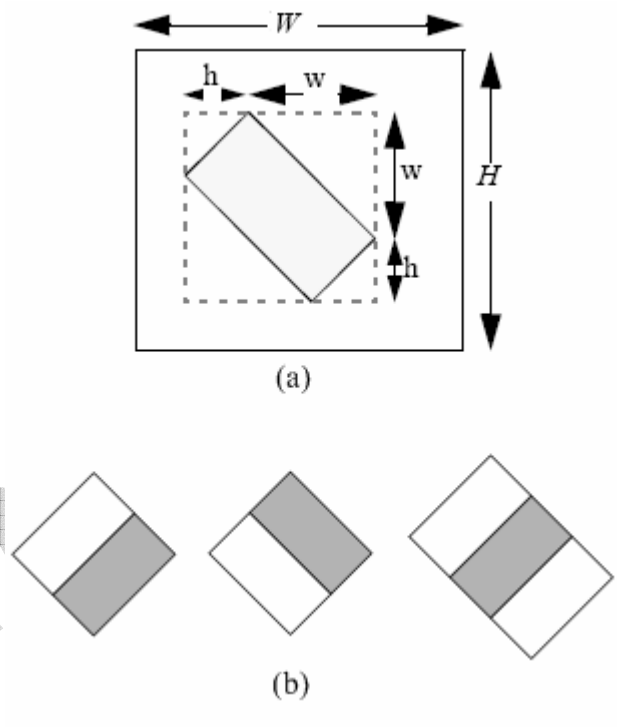
mỗi phân tử của bảng SAT có thể được tính như sau:

$$SAT(x,y)=SAT(x,y-1)+SAT(x-1,y)+I(x,y)-SAT(x-1,y-1)$$

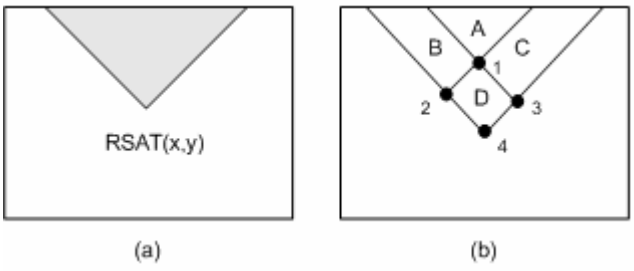
với $SAT(-1,y)=SAT(x,-1)=SAT(-1,-1)=0$

Khi đó, tổng các điểm ảnh trong môt hình chữ nhật bất kì có thể tính nhanh dựa trên integral image tại 4 đỉnh của nó $Sum(D)=4+1-(2+3)$.

Rotated Summed Area Table (RSAT)



Hình 10 - Haar Feature xoay 45º do Lienhart đề nghị



Hình 11 - RSAT(x,y) và cách tính tổng các điểm ảnh trong một hình chữ nhật xoay 1 góc 45º

Integral image tại một điểm (x,y) được định nghĩa:

$$RSAT(x,y)=\sum_{y'\le y, y'\le y-|x-x'|} I(x',y').$$

Mỗi phần tử của bảng RSAT được tính như sau:

$$RSAT(x,y)=RSAT(x-1,y-1)+RSAT(x+1,y-1)-RSAT(x,y-2)+I(x,y)+I(x,y-1)$$

Trong đó

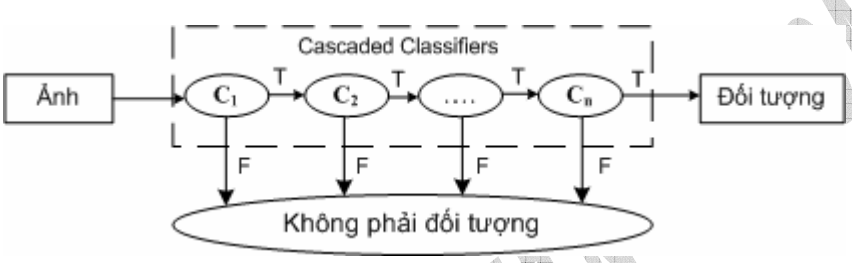
$$RSAT(-1,y)=RSAT(x,-1)=RSAT(x,-2)=0$$
$$RSAT(-1,-1)=RSAT(-1,-2)=0$$

Khi đó, tổng các điểm ảnh trong một hình chữ nhật xoay 45º bất kì có thể được tính nhanh dựa trên integral image tại 4 đỉnh của nó $Sum(D) = 4 + 1 - (2 + 3)$.

3.4 Cascade of Classifiers

Các bộ phân loại tốt thường tốn rất nhiều thời gian để cho ra kết quả phân loại bởi vì nó phải xét *rất nhiều* đặc trưng của mẫu. 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.

Cascade of Classifiers [3] 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.



Hình 12 - Cascade of Classifiers

3.5 Cascade of Boosted Classifiers

Cascade of Boosted Classifiers là mô hình Cascade of Classifers với mỗi classifier được xây dựng bằng AdaBoost sử dụng Haar Feature. Mô hình này đã được *Viola* và *Jones* sử dụng rất thành công trong bài toán nhận dạng mặt người (Face Detection) [1]. Với tập huấn luyện gồm 4196 hình mặt người được đưa về ảnh grayscale kích thước 24x24 và 9500 hình background, *Viola* và *Jones* đã xây dựng cấu trúc cascade tree gồm 38 stage với tổng cộng 6060 haar features. Thực nghiệm đã cho thấy classifier ở stage đầu tiên sử dụng 2 features và loại được khoảng 50% mẫu background (non-face) và có detection rate là 100%. Classifier ở stage thứ 2 sử dụng 10 features loại được 80% mẫu background vẫn với 100% detection rate. Hệ thống này được so sánh với hệ thống của *Rowley-Baluja-Kanade* [11] (sử dụng neural network), *Schneiderman-Kanade* [12] (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*.

Bên cạnh đó, mô hình này cũng được *Eng-Jon Ong* và *Richard Bowden* áp dụng thành công trong bài toán nhận dạng bàn tay (Hand Detection) [14]. Do bàn tay có nhiều biến động hơn so với mặt người, *Ong* và *Bowden* đã sử dụng phương pháp học không giám sát (unsupervised learning): tiến hành phân cụm cho tất cả các mẫu trong tập huấn luyện chứa 2504 hình bàn tay chụp ở nhiều tư thế khác nhau bằng thuật toán *K-mediod clustering*. Cấu trúc bộ nhận dạng của *Ong* và *Bowden* gồm 2 lớp: lớp ở trên

là 1 Cascade of Boosted Classifiers để nhận dạng sơ bộ bàn tay, lớp bên dưới là từng Cascade of Boosted Classifiers ứng với từng cụm được chia bằng K-mediod. Kết quả thu được rất khả quan, cấu trúc cascade của bộ nhận dạng ở lớp trên gồm 11 stage với tổng cộng 634 weak classifiers đã đạt tỉ lệ nhận dạng là 99.8% trên tập test, còn các bộ cascade ở lớp dưới có tỉ lệ nhận dạng trung bình là 97.4%.

Bài toán phân loại cử chỉ tương tự như bài toán hand detection của *Ong* và *Bowden* nhưng phức tạp hơn. Trong bài toán của họ, một mẫu chỉ cần thỏa một trong các bộ nhận dạng ứng với các cluster thì được xem là bàn tay. Trong thực tế, có những mẫu vốn thuộc về cluster 1, nhưng bộ nhận dạng cluster 1 cho kết quả nó không thuộc cluster 1, trong khi bộ nhận dạng cluster 2 lại cho rằng nó thuộc về cluster 2. Khi đó, rõ ràng cả 2 bộ nhận dạng đều cho kết quả sai nhưng mẫu này vẫn được xem là một test thành công của bộ nhận dạng bàn tay, vì mẫu được đưa vào vẫn được đánh giá là bàn tay. Còn đối với bài toán nhận dạng cử chỉ, nếu xem các bộ nhận dạng từng cử chỉ (A, B, C ...) ứng với các bộ phân loại cluster ở trên thì trường hợp tương tự sẽ là một test thất bại (bởi vì cho kết quả phân loại sai).

Bên cạnh đó, thuật toán K-mediod cũng không áp dụng được vì bài toán phân loại cử chỉ là bài toán học giám sát (các lớp đã được qui định trước). Ngoại trừ những khác biệt trên, nhìn chung thì bài toán phân loại cử chỉ với bài toán hand detection là giống nhau. Do đó, một mô hình cho kết quả tốt trên bài toán hand detection cũng có thể cho kết quả tốt trên bài toán phân loại cử chỉ.

3.6 Đánh giá

Trước khi có hệ thống của *Viola* và *Jones*, hệ thống của *Rowley* [11] được đánh giá là bộ face detecton có tốc độ nhanh nhất. Hệ thống của *Rowley* thực chất cũng là môt cấu trúc cascade với 2 neural networks: neural network thứ nhất khá đơn giản nhằm mục đích chính là loại bỏ các hình background có độ khó thấp, neural network thứ 2 phức tạp hơn, đảm nhiệm việc nhận dạng các mẫu đi qua neural network thứ 1.

Điều này chứng tỏ cách tổ chức cascade nhằm loại nhanh các mẫu có độ phức tạp thấp thực sự đẩy nhanh tốc độ của hệ thống.

Ý tưởng của Viola và Jones khi đưa ra Cascade of Boosted Classifiers thật ra cũng tương tự vậy, nhưng nó mở rộng 2 stages thành 38 stages của Cascade Tree. Hệ thống của *Viola* và *Jones* càng chứng tỏ khả năng tăng tốc của mô hình cascade khi đạt tốc độ nhanh hơn hệ thống của *Rowley* và hệ thống của *Schneiderman-Kanade* (vốn không hề sử dụng cascade) lần lượt 15 lần và 600 lần.

Trong bài viết [1], *Viola* và *Jones* cũng đã tiến hành so sánh hệ thống sử dụng Cascade of Boosted Classifiers với một hệ thống chỉ có một bộ phân loại duy nhất xây dựng bằng AdaBoost với tổng số Haar Features sử dụng là 200. Kết quả là hệ thống theo mô hình Cascade of Boosted Classifiers nhanh hơn đến 10 lần.

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 thực sự cần tiến hành nhận dạng).

Bên cạnh đó, hệ thống của *Viola* và *Jones* cũng đạt được độ chính xác cao tương đương các hệ thống khác là nhờ thuật toán cấu trúc cascade các bộ nhận dạng được huấn luyện bằng AdaBoost với đặc trưng Haar Feature mô tả tốt thông tin đối tượng, cùng với cách tính Integral Image tính nhanh các features, không làm giảm tốc độ nhận dạng của hệ thống.

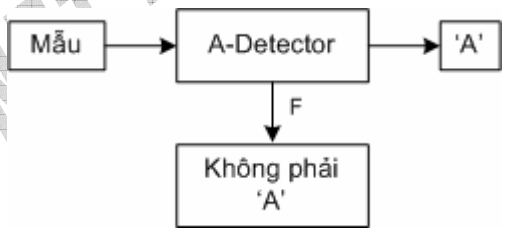
Như vậy, mô hình Cascade of Boosted Classifiers thật sự là một cách tiếp cận tốt cả về tốc độ lẫn khả năng nhận dạng, rất phù hợp với bài toán nhận dạng cử chỉ.
Tuy nhiên, bài toán cũng đặt ra một số khó khăn về số lượng mẫu và thời gian huấn luyện. AdaBoost đòi hỏi phải có số lượng mẫu rất lớn (tối thiểu phải lên đến hàng nghìn) để huấn luyện được bộ phân loại hiệu quả. Hệ thống nhận dạng mặt người của *Viola* và *Jones* cần đến 4916 ảnh mặt người. Với bài toán phân loại 24 cử chỉ, nếu mỗi cử chỉ cần đến 4000 mẫu thì tổng số mẫu sẽ là 96000 mẫu. Việc thu thập đủ số lượng mẫu này là một trở ngại rất lớn. Bên cạnh đó, do số lượng mẫu nhiều, đồng thời số

lượng Haar Feature xử lý lớn nên thời gian huấn luyện rất lâu. Chỉ huấn luyện 1 bộ nhận dạng cử chỉ A với 1000 mẫu positive và 1000 mẫu negative đã mất đến 14 giờ (trên máy P4 2.0Ghz, 512 MB RAM). Do đó, việc xây dựng toàn bộ hệ thống phân loại 24 cử chỉ sẽ tốn rất nhiều thời gian (phải kể cả việc trong quá trình thử nghiệm sẽ có những lần huấn luyện thất bại, buộc phải tiến hành huấn luyện lại). Chương tiếp theo sẽ là phần trình bài chi tiết việc áp dụng mô hình cascade lên bài toán phân loại cử chỉ.

Chương 4 Phân loại cử chỉ với Cascade of Boosted Classifiers

4.1 Bộ nhận dạng 1 cử chỉ

Bộ nhận dạng cho 1 cử chỉ có chức năng nhận dạng một mẫu có thuộc về một cử chỉ đó không. Bộ nhận dạng này được xây dựng theo cấu trúc Cascade of Boosted Classifiers.

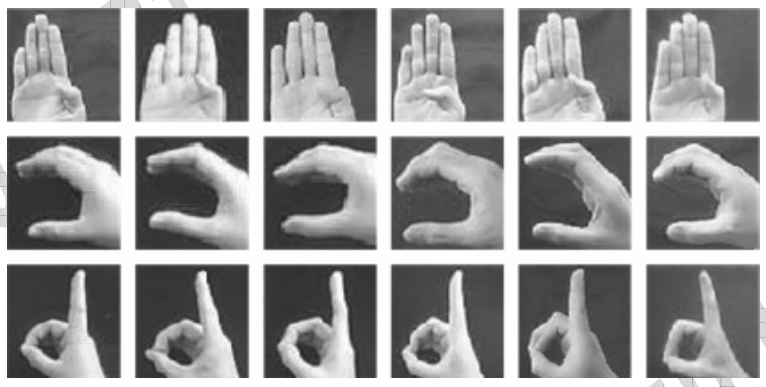


Hình 13 - Bộ nhận dạng cử chỉ A

4.1.1 Tập huấn luyện

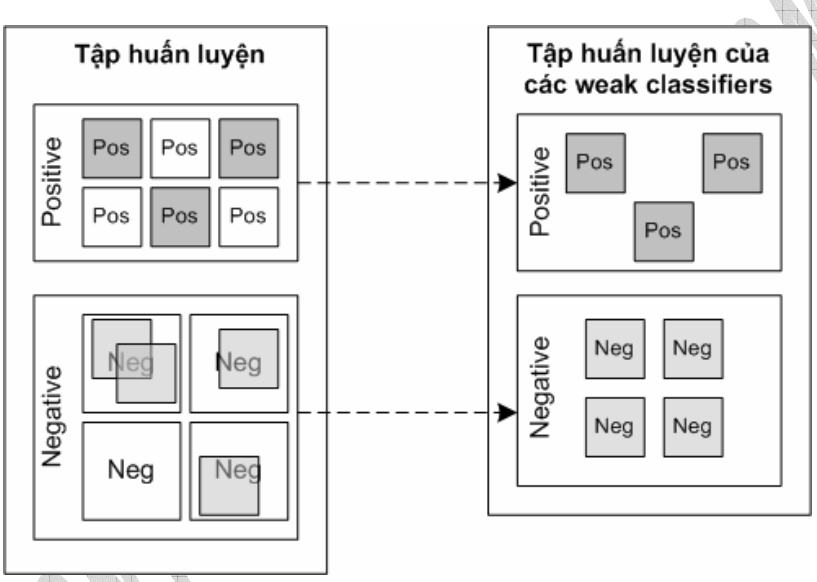
Tập huấn luyện bao gồm các mẫu positive (đối tượng cần nhận dạng) và negative (mẫu không chứa đối tượng). Trong bộ nhận dạng 1 cử chỉ, các mẫu positive là các hình chụp của cử chỉ đó đã qua chuẩn hóa: kích thước 32x32 và được chuyển về ảnh grayscale. Các mẫu negative bao gồm tất cả các ảnh không liên quan đến cử chỉ cần nhận dạng: các hình background như nhà cửa, xe cộ, cây cối, mặt người... và ảnh của *các cử chỉ khác*. Hình 14 là 18 mẫu chữ A chụp từ 18 người đã qua chuẩn hóa.

Hình 14 - Các mẫu positive cho bộ nhận dạng chữ A



Hình 15 - Các mẫu negative (B, C, D) cho bộ nhận dạng chữ A

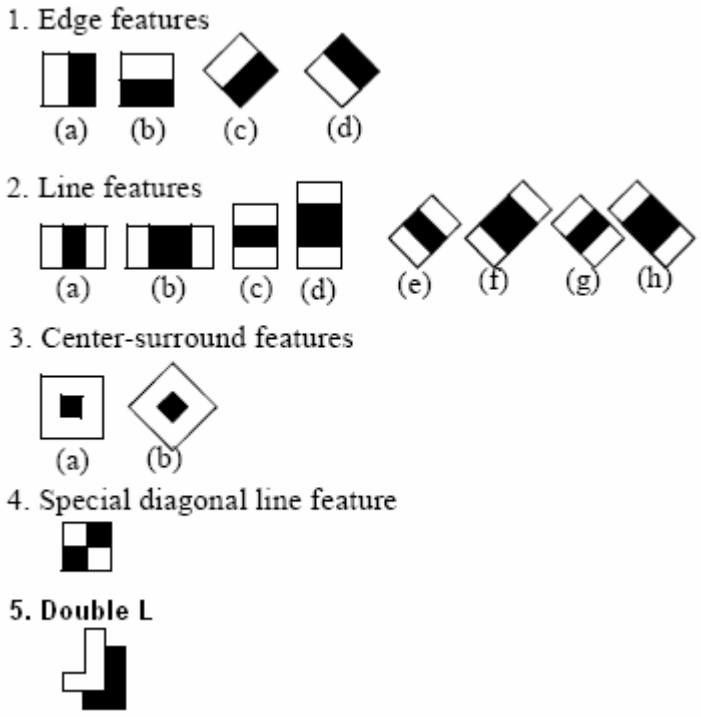
Không như các mẫu positive có kích thước cố định, các mẫu negative trong tập huấn luyện có thể có kích thước tùy ý nhưng phải lớn hơn kích thước mẫu positive. Trong quá trình huấn luyện, các weak classifiers sẽ học từ các mẫu positive trong tập huấn luyện và các mẫu negative là các vùng ảnh (sub window) trích ra từ các mẫu negative trong tập huấn luyện.



Hình 16 - Tập huấn luyện của các weak classifiers

4.1.2 Đặc trưng

Khóa luận sử dụng các loại Haar Feature sau:



Hình 17 - Các Haar Feature sử dụng trong bộ nhận dạng 1 cử chỉ

Ngoại trừ feature 5 do khóa luận đưa ra, các feature còn lại đều được lấy từ [4]. Hệ thống đặc trưng trên bao gồm cả Haar Feature cơ bản (1a, 1b, 2a, 2b, 2c, 2d, 3a, 1, 5) và các Haar Feature mở rộng (xoay 45^o - 1c, 1d, 2e, 2f, 2g, 2h, 3b). Các feature này đều được tính toán nhanh nhờ vào khái niệm Integral Image.

4.1.3 Xây dựng bộ nhận dạng với AdaBoost

Với 16 loại feature sử dụng và với kích thước mẫu positive là 24x24, tập feature (Feature Pool) của chúng ta có khoảng 160000 feature. Tuy nhiên, không phải feature nào cũng thực hiện tốt việc phân loại mà chỉ có một số lượng nhỏ trong số 160000 feature này là thực sự hữu dụng. Nhiệm vụ của bộ phân loại là phải tìm ra được các feature này.

Mỗi weak classifier gồm có 1 feature và 1 ngưỡng, ngưỡng này chính là giá trị của một mẫu cụ thể cho bởi feature này (vấn đề của thuật toán là phải tìm được mẫu nào dùng làm ngưỡng). Như vậy, với N mẫu và M features thì số weak classifier có thể có là $M \times N$, tức là với 2000 mẫu cho một cử chỉ và 160000 features thì hệ thống sẽ phải chọn được 1 weak classifier trong số $160000 \times 2000 = 320000000$ weak classifiers trong mỗi vòng lặp boosting.

AdaBoost được thiết kế để có thể chọn nhanh các features, cũng là chọn nhanh các weak classifiers. Thuật toán sử dụng ở đây là Gentle AdaBoost [4], một biến thể của AdaBoost. Gentle AdaBoost để xây dựng bộ nhận dạng cho 1 cử chỉ như sau:

Thủ tục ***CreateClassifier:***

1. Cho một tập huấn luyện gồm n mẫu (x, y) với x ∈ X, y ∈ {-1,+1}
2. Chọn trước *min detection rate* và *max false alarm*
3. Xây dựng tập feature và tính toán các bảng *SAT* và *RSAT* cho tất cả các mẫu trong tập huấn luyện.
4. Khởi tạo trọng số ban đầu cho tất cả các mẫu

$$w_{1,k} = \frac{1}{n}$$

5. Lặp

- Với mỗi đặc trưng trong tập đặc trưng, xây dựng một weak classifier h_j với hàm phân loại f_t, ngưỡng θ_j và tính giá trị lỗi ε_j của bộ phân loại này

$$\varepsilon_j = \sum_k^n wt,k\left|h_j(x_k) - y_k\right|$$

- Từ các h_j đã có, chọn ra h_j có ε_j nhỏ nhất, ta được h_t:

$$h_t(x) = g_t(f_t(x), \theta_t) : X \to R$$

- Cập nhật lại trọng số:

$$w_{t+1,k} = \frac{w_{t,k}}{Z_t} \times e^{-y_k h_t(x_k)}$$

với $Z_t = \sum_k w_{t+1,k}$ là normalization factor.

- $F_t(x) = \sum_{i=1}^{t} h_i(x)$, tính ngưỡng θ

$$H_t(x) = \begin{cases} 1, F(x) \geq \theta \\ -1, F(x) < \theta \end{cases}$$

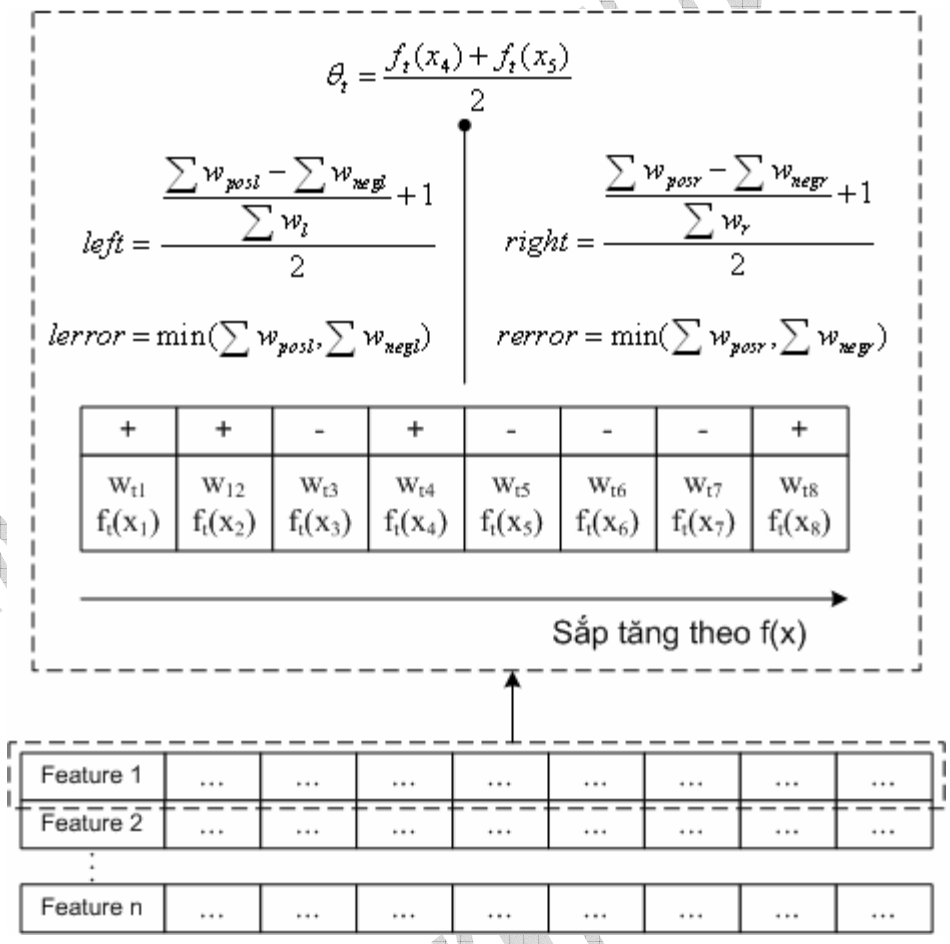
Cho đến khi $\sum_{k=1}^{n} 1_{y_k=-1, H_t(x_k)=1} \leq max\_false\_alarm$

6. Bộ nhận dạng *H(x)*

$$H(x) = \begin{cases} 1, F(x) \geq \theta \\ 0, F(x) < \theta \end{cases}$$

với $F(x) = F_t(x)$

Mỗi weak classifier $h_t(x)$ sẽ được chọn bằng cách tính giá trị của từng feature cho tất cả các mẫu trong tập huấn luyện $f(x_k)$ và sắp xếp các mẫu theo thứ tự tăng dần của giá trị này:



Hình 18 - Cách chọn weak classifier của AdaBoost

Feature được chọn là feature thực hiện phân đôi tập huấn luyện với lỗi $\varepsilon_t = lerror + rerror$ nhỏ nhất.

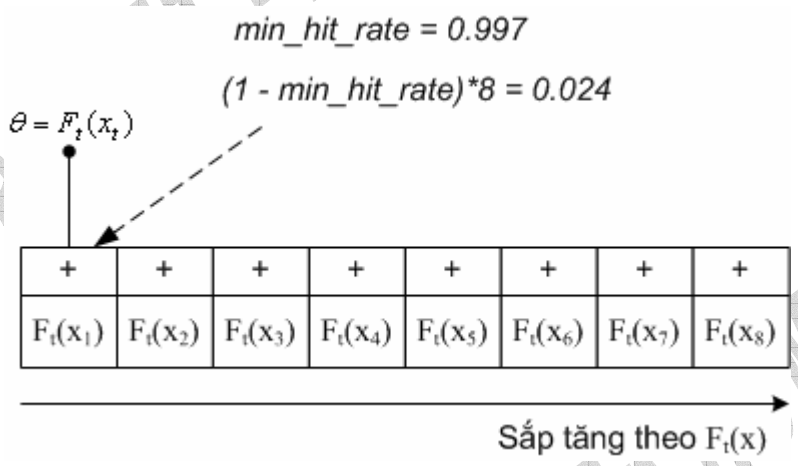
$$h_t(x) = \begin{cases} left, f_t(x) < \theta_t \\ right, f_t(x) >= \theta_t \end{cases}$$

Sau mỗi vòng lặp của boosting, ta xây dựng thêm được một weak classifier, ta có thể xây dựng được một strong classifier mới từ sự kết hợp của các weak classifiers có được cho tới lần boost hiện tại:

$$F_t(x) = \sum_{i=1}^{t} h_t(x)$$

$$H_t(x) = \begin{cases} 1, F_t(x) >= \theta \\ -1, F_t(x) < \theta \end{cases}$$

Giá trị ngưỡng θ được chọn nhờ vào giá trị *min detection rate (hay min hit rate)*. *min detection rate* là *tỉ lệ nhận dạng đúng tối thiểu các mẫu positive* mà bộ phân loại phải đạt, giá trị này do chúng ta xác lập khi bắt đầu quá trình huấn luyện. Bằng cách sắp xếp các mẫu *positive* x_k theo $F_t(x_k)$ tăng dần:



Hình 19 - Chọn ngưỡng θ dựa vào min detection rate

Trong hình 19, *min detection rate* được chọn là *0,997*, nghĩa là tỉ lệ nhận dạng đúng các mẫu positive không được thấp hơn 99.7%. Do đó, phần tử được chọn làm ngưỡng là mẫu đầu tiên trong danh sách sắp tăng. Giá trị ngưỡng θ này đảm bảo có ít nhất 99.7% các mẫu positive sẽ có $F_t(x) >= \theta$. Nếu ta chọn *min detection rate* là *0.5*, khi đó (1-min detection rate)*số_mẫu = 4, phần tử thứ 4 của danh sách các mẫu sẽ được chọn làm ngưỡng, như vậy sẽ chỉ có 4 mẫu trong số 8 mẫu positive là được nhận dạng đúng.

Ngưỡng có ý nghĩa rất quan trọng, ngưỡng càng nhỏ thì *hit rate* sẽ càng cao nhưng *false alarm* cũng tăng theo và ngược lại.

Sau khi có được $H_t(x)$, vấn đề cần xem xét là liệu bộ phân loại này có đủ tốt chưa, nghĩa là giá trị lỗi của nó đã thấp hơn *max false alarm* hay chưa. *Max false alarm* là một giá trị được xác lập trước khi tiến hành huấn luyện. F*alse alarm* (hay còn gọi là *false positive*) là tỉ lệ nhận dạng sai các mẫu negative. Ví dụ, nếu *max false alarm* = 0.5 thì trên 100 mẫu negative, nó phải nhận đúng ít nhất là 50 mẫu (50 mẫu còn lại bị phân loại nhầm thành positive). Vòng lặp xây dựng strong classifier sẽ kết thúc khi giá trị false alarm của strong classifier này thấp hơn *max false alarm*.

4.1.4 Cascade of Boosted Classifiers

Bộ nhận dạng 1 cử chỉ là 1 cấu trúc cascade gồm K stages, f_i, d_i lần lượt là *max false alarm* và *min detection rate* của stage classifier ở tầng thứ i, *max false alarm* và *min detection rate* của cascade tree sẽ lần lượt là:

$$F = \sum_{i=1}^{K} f_i$$

$$D = \sum_{i=1}^{K} d_i$$

Trên lý thuyết, các stage classfiers sẽ có *max false alarm* và *min detection rate* khác nhau, các classifier ở các stage càng sâu thì *max false alarm* sẽ càng lớn và *min detection rate* càng nhỏ do nó học trên các mẫu khó hơn. Tuy nhiên, trong thực tế cài đặt, chúng ta không thể biết chính xác số stage mà cascade tree sẽ có trước khi tiến hành huấn luyện, dẫn đến khó khăn trong việc chọn giá trị *max false alarm* và *min detection rate* cho mỗi stage classifiers. Do đó, trong phần cài đặt của bài toán nhận dạng cử chỉ, *max false alarm* và *min detection rate* được xác lập bằng nhau ở tất cả các stages. Khi đó, *max false alarm* và *min detection rate* của bộ nhận dạng lần lượt là $F = f^K$ và $D = d^K$.

Thuật toán xây dựng cascade tree:

- Xác lập *max false alarm f* , *min detection rate d* cho các bộ phân loại ở mỗi tầng và số stage tối đa cascade tree sẽ có
- Tính *false alarm F* cho bộ phân loại chính (cây phân lớp)
- $F_0 = 0$, i = 0
- *p*, *n* là số lượng mẫu positive và negative
- P_0, N_0 là tập positive và negative cho bộ phân lớp ở tầng đầu tiên
- Trong khi $F_i > F$
  - Huấn luyện bộ phân loại H_i từ tập P_i và N_i với *detection rate d* và *max false alarm f*(thủ tục ***CreateClassifier***)
  - Thêm H_i vào cây phân lớp
  - Dùng cây phân lớp hiện có để tính F_{i+1}: Duyệt qua *N* mẫu negative cho đến khi nào tìm đủ *n* mẫu mà cây phân lớp hiện có phân loại sai

$$F_{i+1} = \frac{n}{N}$$

  - Nếu $F_{i+1} > F$, đưa *n* mẫu negative trên vào N_{i+1}; xây dựng P_{i+1} với tối đa *p* mẫu positive mà cây phân lớp hiện có phân loại dúng.
  - $i \leftarrow i+1$

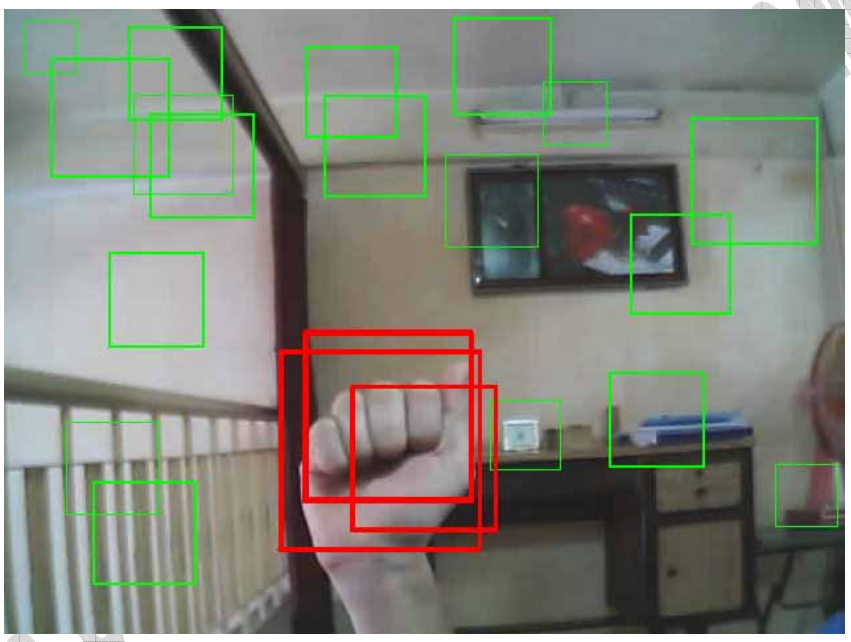
Qua thuật toán trên ta thấy, các mẫu negative trong tập huấn luyện của stage classifier sau sẽ là những mẫu negative mà stage classifier trước nó nhận dạng sai. Như vậy, nó có điều kiện tập trung học những mẫu background khó. Trong bộ nhận dạng cử chỉ A xây dựng được, stage classifier thứ 2 phải tìm trong 5055 vùng ảnh background để có thể đưa vào tập huấn luyện của mình 1791 mẫu background. Điều này có nghĩa là trong số 5505 mẫu background thì stage classifier thứ 1 đã để bị sai 1791 mẫu. Tuy nhiên, stage classifier thứ 3 phải tìm trong 17023 mẫu background mới lấy được 1791 mẫu background đưa vào tập huấn luyện – tức là sự kết hợp stage classifier 1 và 2 chỉ nhận dạng sai 1971 mẫu

trong số 17024 mẫu background (xử lý background tốt hơn khi chỉ sử dụng stage classifier 1). Đến stage classifier cuối cùng thì cần phải tìm trong số 42818355 mẫu background để có thể chọn ra 1791 mẫu background cho mình, tức là đạt được giá trị false alarm khoảng 0.000041. Như vậy ta thấy các stage classifier được xây dựng sao cho sự kết hợp của chúng xử lý rất tốt các mẫu background. Đây chính là lý do mà cấu trúc cascade giúp cho bộ nhận dạng giảm thiểu false alarm, và cũng là tăng tỉ lệ nhận dạng cho hệ thống. Đồng thời, cấu trúc này cũng đủ thông minh để biết phải loại một mẫu background ở stage nào tùy thuộc vào độ phức tạp của chúng, chứ không nhất thiết phải gọi đến tất cả các stage classifiers đối với mọi mẫu background – chính điều này đã giúp tăng tốc độ nhận dạng.

4.1.5 Hoạt động của bộ nhận dạng cử chỉ

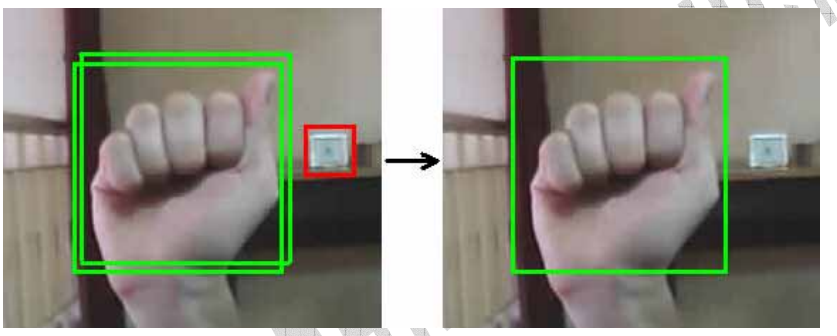
Khi đưa một ảnh vào nhận dạng, bộ nhận dạng sẽ phải xét tất cả các vùng ảnh với kích thước khác nhau trích ra được từ ảnh này để có thể đưa ra kết quả. Kích thước khởi đầu của vùng ảnh sẽ là một window bằng với kích thước của mẫu positive trong quá trình huấn luyện, tức là 24x24. Các window này sẽ được dịch theo chiều ngang và dọc 1 lượng từ 1 đến 2 pixel cho đến khi phủ kín ảnh cần nhận dạng. Sau đó, window sẽ được mở ra với tỉ lệ 1.1 (giá trị này người dụng được phép thay đổi khi tiến hành nhận dạng) và tiếp tục quá trình duyệt ảnh như trên cho đến khi window được mở ra bằng kích thước ảnh.

Nhờ có cấu trúc cascade, các vùng ảnh không liên quan bị loại nhanh từ những stages đầu tiên.



Hình 20 - Các vùng ảnh không liên quan (nét mảnh) sẽ bị loại ngay từ những stages đầu tiên

Trong quá trình trích vùng ảnh, sẽ có các vùng có vị trí và kích thước tương tự nhau. Các vùng này có thể đều được bộ nhận dạng trả về true, khiến cho 1 cử chỉ được nhận dạng nhiều hơn 1 lần.



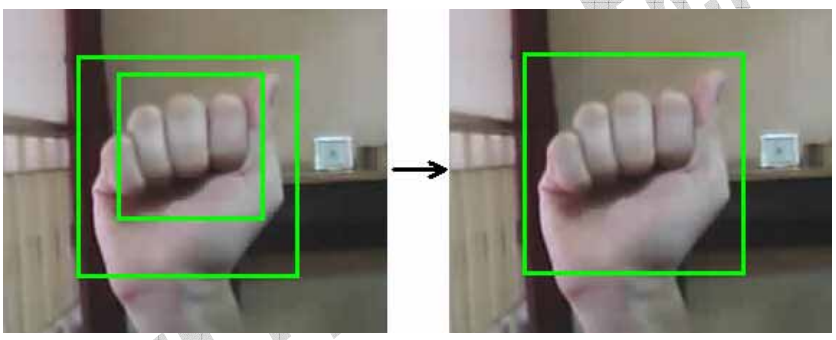
Hình 21 - Khắc phục trường hợp nhiều vùng ảnh kế cận nhau bằng cách lấy vùng ảnh trung bình

Để khắc phục điều này, trên các vùng ảnh được bộ nhận dạng đánh giá là cử chỉ, ứng dụng nhận dạng sẽ tìm và nhóm các vùng ảnh tương tự nhau bằng cách thay chúng bằng 1 vùng ảnh duy nhất có được bằng cách lấy trung bình của các vùng ảnh này.

Qua thực nghiệm, ta thấy số lượng vùng ảnh tương tự nhau cũng có tác dụng trong việc điều chỉnh giữa *hit rate* và *false alarm*. Do tỉ lệ nhận dạng không phải

100% nên khi tiến hành nhận dạng, có thể có vài mẫu background sẽ bị nhầm là cử chỉ (như chiếc đồng hồ trong hình bên trái của hình 21). Trong khi bàn tay trong hình đưa vào luôn có nhiều hơn 1 vùng ảnh chứa nó được bộ nhận dạng đánh giá là cử chỉ, các mẫu background bị nhận dạng sai thường nằm tách biệt (chiếc đồng hồ chỉ có 1 vùng ảnh dung nhất là được đánh giá là cử chỉ). Do đó, ứng dụng nhận dạng sử dụng khái niệm *min neighbor*, tức là số vùng ảnh tương tự nhau tối thiểu phải có để một vùng ảnh có thể được phân loại là cử chỉ. Giá trị này được người dùng xác lập khi tiến hành nhận dạng. Trong hình 21, với *min neighbor* là 2, chiếc đồng hồ sẽ bị loại, tức là giảm *false alarm*. Tuy nhiên, cần phải thận trọng trong việc sử dụng *min neighbor*, vì nếu chỉnh *min neighbor* quá lớn, có thể làm giảm luôn cả *hit rate*. Rõ ràng với hình 21 nếu xác lập *min neighbor* là 3 thì không chỉ có chiếc đồng hồ mà cả bàn tay cũng bị loại vì nó chỉ có 2 vùng ảnh. Với các bộ nhận dạng cử chỉ xây dựng thì giá trị *min neighbor* là 5 cho kết quả tốt nhất.

Bên cạnh đó, có thể trong việc nhận dạng còn tồn tại vấn đề các vùng ảnh lồng vào nhau.



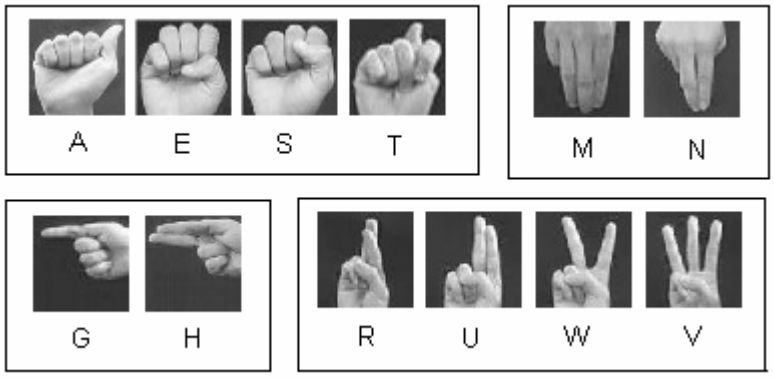
Hình 22 - Đối với các vùng ảnh lồng nhau, các vùng ảnh bên trong sẽ bị loại bỏ

Đối với các trường hợp nay, ứng dụng chỉ cần đơn giản là loại bỏ tất cả các vùng ảnh ở bên trong, chỉ giữ lại vùng ảnh ngoài.

4.2 Bộ phân loại cử chỉ

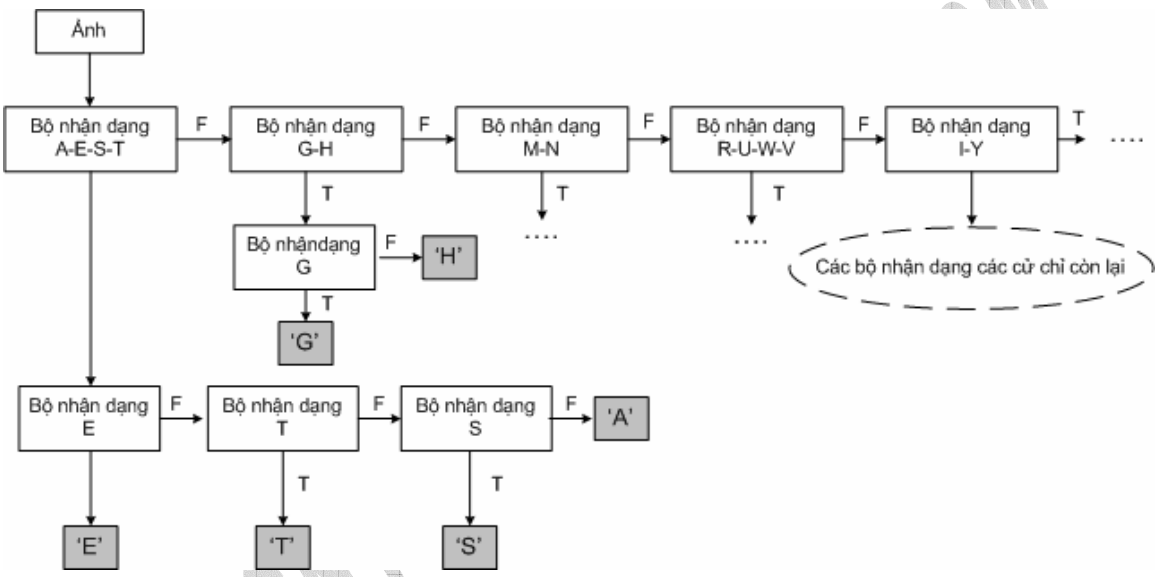
Bộ phân loại cử chỉ là một bộ phân loại được xây dựng dựa trên sự kết hợp các bộ nhận dạng một cử chỉ đã trình bày ở trên. Đầu vào của bộ phân loại này là một ảnh mẫu, đầu ra là lớp của ảnh này (A, B, C ... hay Y).

Các bộ nhận dạng một cử chỉ (ví dụ bộ nhận dạng chữ A) có thể đươc kết hợp theo cách tiếp cận one against all truyền thống. Tuy nhiên, trong hệ thống 24 cử chỉ, có rất nhiều cử chỉ khá giống nhau, những cử chỉ này có thể được gom chung lại thành 1 nhóm.



Hình 23 - Các cử chỉ giống nhau trong hệ thống 24 cử chỉ

Mỗi nhóm cử chỉ sẽ được xây dựng một bộ nhận dạng nhóm (nhận dạng một cử chỉ có thuộc về nhóm này hay không). Dưới các bộ nhận dạng nhóm này sẽ là các bộ nhận dạng của từng cử chỉ thuộc về nhóm đó. Một mẫu đưa vào sẽ đi qua tuần tự các bộ nhận dạng từ trên xuống dưới cho đến khi nào nó được phân vào một lớp cụ thể hay được cho là không phải một cử chỉ.



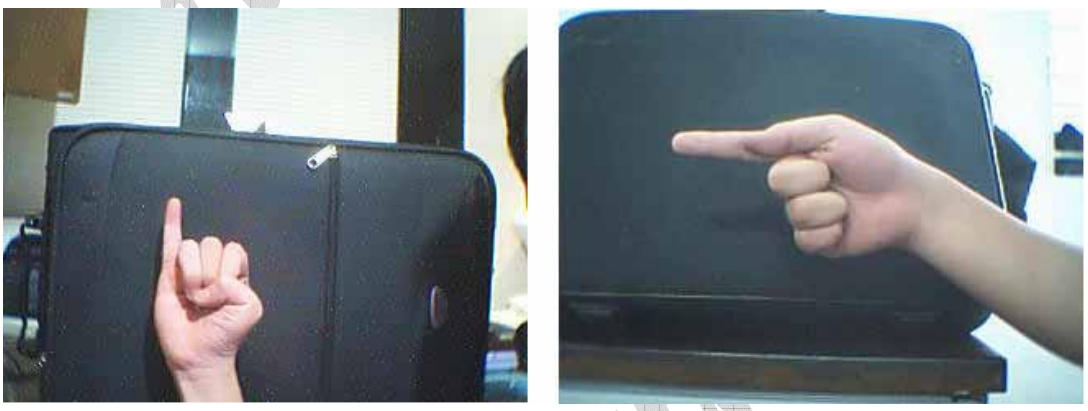
Hình 24 – Cấu trúc bộ phân loại cử chỉ

Chương 5 Kết quả thử nghiệm

Chương này sẽ trình bày về phương pháp thực hiện các thí nghiệm, bao gồm cách xây dựng bộ dữ liệu huấn luyện, bộ dữ liệu test và các kết quả đạt được, đồng thời so sánh kết quả chúng em đạt được với các paper có liên quan.

5.1 Tập huấn luyện

Chúng em đã thực hiện thu thập khoảng trên 1000 mẫu của 50 người khác nhau bằng webcam Colorvis với độ phân giải 320x240 trong nhiều điều kiện môi trường khác nhau: dưới ánh sáng đèn neon, ánh sáng tự nhiên vào các thời điểm khác nhau... sau đó loại trừ các mẫu không đạt chất lượng để còn lại khoảng 750 mẫu.

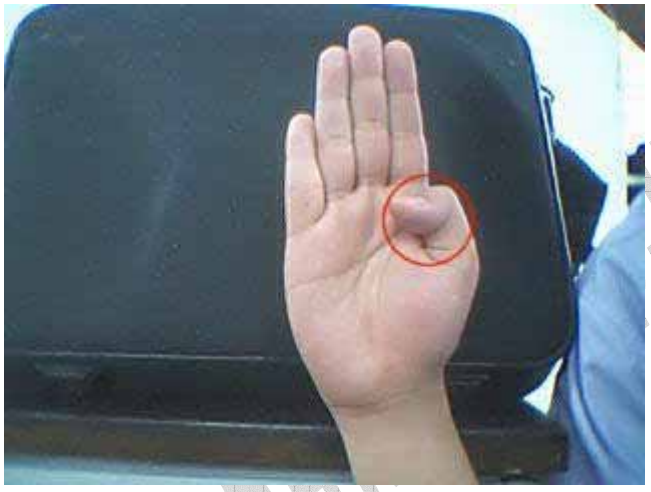


Hình 25 - Hình chụp bằng Webcam

Những người được chọn làm mẫu sẽ thực hiện lần lượt 24 động tác tương ứng với 24 mẫu chữ cái trên một phông nền màu đen. Sau đó các hình chụp được sẽ được cắt lại sao cho mẫu bàn tay sẽ nằm trong một hình vuông dựa trên một tiêu điểm của từng ký tự để đảm bảo các tiêu điểm đó sẽ giúp phân biệt chữ này với các chữ còn lại. Ví dụ: Giả sử chúng ta cần cắt mẫu chữ B, chúng ta sẽ lần lượt làm như sau:

Hình 26 - Hình chụp chữ B

Chú ý rằng, chỗ ngón cái gập vào của chữ B chính là tiêu điểm. Chúng ta sẽ phân biệt được chữ B so với các chữ cái còn lại là nhờ vào vị trí đó. Do đó chúng ta sẽ tiến hành chọn vị trí đó làm vị trí chuẩn để cắt hình



Hình 27 - Tiêu điểm của cử chỉ B

Sau đó, cắt sao cho bàn tay nằm trong một hình vuông, rồi làm tương tự với các hình mẫu khác sao cho vị trí tiêu điểm của các hình của cùng 1 cử chỉ phải có vị trí tương đối giống nhau trong hình vuông cần cắt.

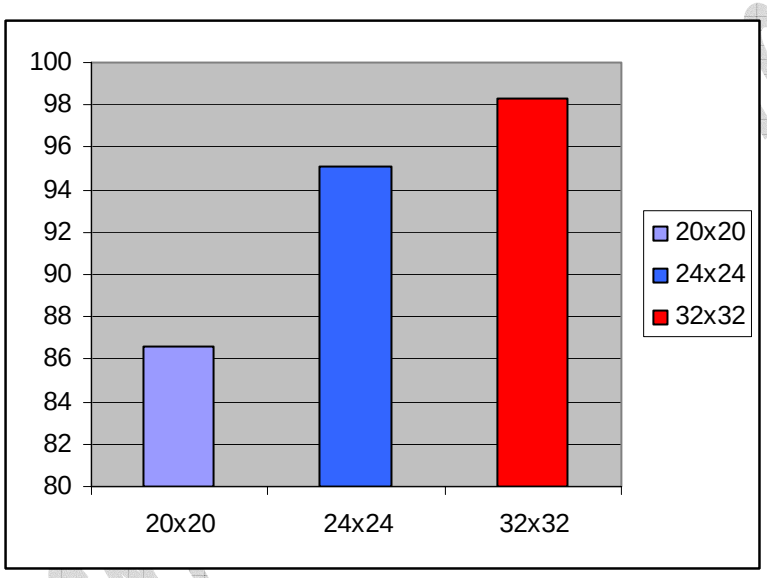
Hình 28 - Hình chữ B sau khi cắt

Tiếp đến, trong tất cả các mẫu chụp được, chúng em chọn mẫu có kích thước nhỏ nhất, rồi giảm kích thước của tất cả các hình còn lại về kích thước này. Sau đó từ mỗi mẫu này, chúng em sẽ cho phát sinh ra 20 mẫu tương ứng bằng các phép quay từ -5 đến +5 độ, dịch chuyển trong khoảng từ -1 đến 1 pixel, giảm hoặc tăng độ sáng tối, phóng to, thu nhỏ bàn tay đi 0.1... Cuối cùng, chuyển tất cả các hình gốc và hình phát sinh về ảnh grayscale có cùng kích thước để tiến hành huấn luyện..

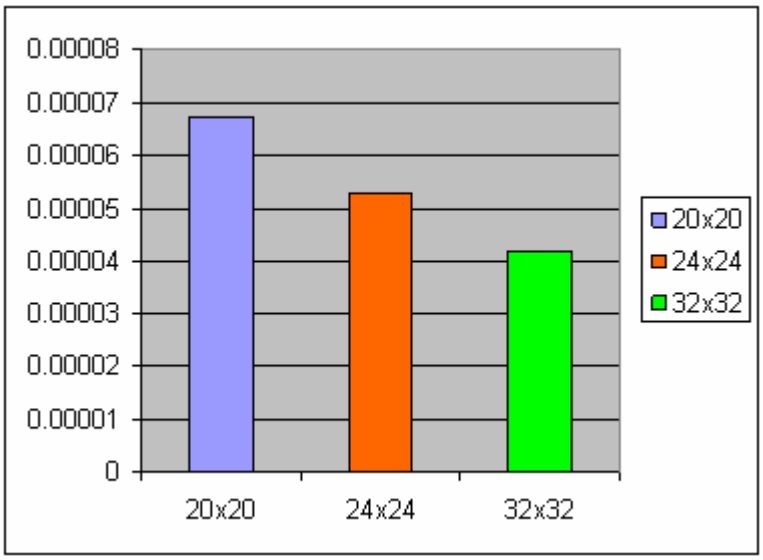
Trong quá trình huấn luyện chúng em đã tiến hành thử nghiệm trên các mẫu với các kích thước 20x20, 24x24, 32x32 và thấy bộ huấn luyện với kích thước mẫu 32x32 cho kết quả tốt nhất. Tuy nhiên chúng ta không nên tăng kích thước mẫu lên nữa vì sẽ làm cho quá trình huấn luyện chậm đi rất nhiều. Dưới đây là kết quả huấn luyện bộ nhận dạng cử chỉ ‘B’ với kích thước mẫu positive khác nhau..

Kích thước	Positive	Negative	Số stage	Hit rate	False Alarm
20x20	630	7980	8	86.6%	0.000067
24x24	630	7980	11	95.1%	0.000053
32x32	630	7980	15	98.3%	0.000042

Bảng 1 - Kết quả huấn luyện với 3 kích thước của mẫu positive



Hình 29 - Biểu đồ Hit Rate



Hình 30 - Biểu đồ False Alarm

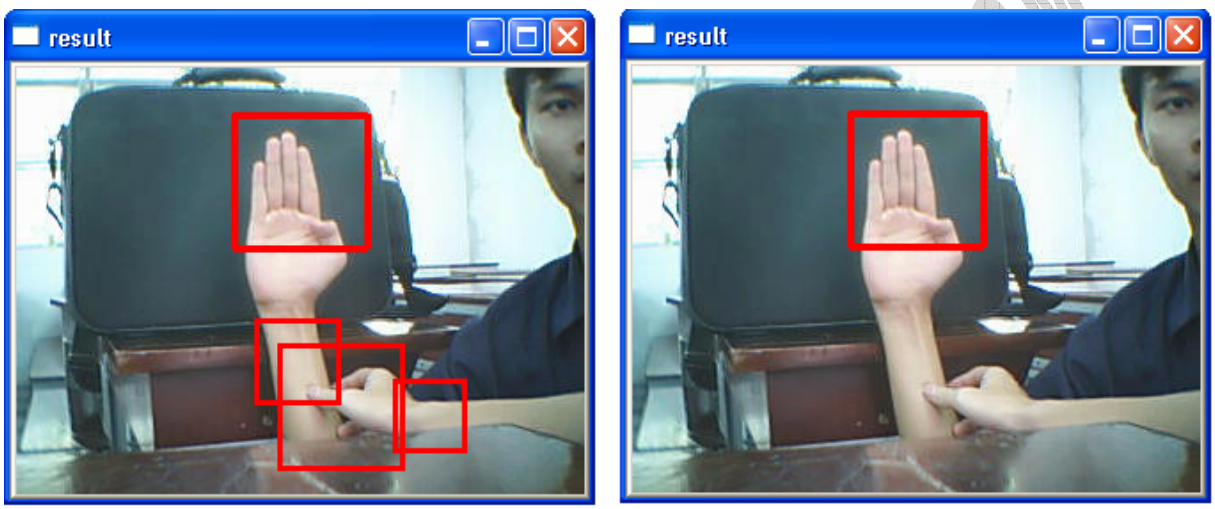
Từ kết quả thực nghiệm ở trên, chúng em đã chọn kích thước 32x32 là kích thước cho mẫu chuẩn. Bên cạnh đó chúng em tiến hành thu thập thêm 192 hình các ký tự mới từ 7 người khác dùng làm mẫu kiểm thử (mẫu test).

Đối với các mẫu negative, chúng em đưa vào các hình phong cảnh, cây cối, nhà cửa ... và các hình có chứa mặt người hoặc các bộ phận cơ thể như: tay, chân... Bộ dữ liệu có chứa mặt người chúng em lấy từ bộ dữ liệu được dùng trong bài toán nhận dạng

mặt người của CMU, những hình phong cảnh là những hình do chúng em chụp và tải xuống từ kết quả tìm kiếm trên google. Từ đó chúng em đã thu thập 3476 hình để làm nền cho phần huấn luyện.

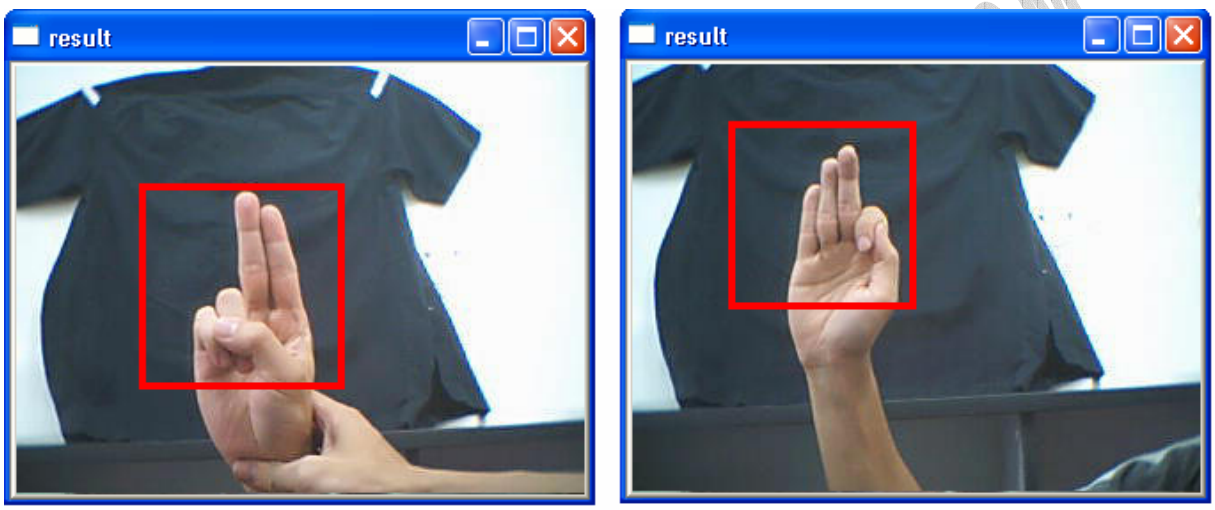
5.2 Cách tiến hành huấn luyện

Mỗi bộ nhận dạng cho từng cử chỉ sẽ được huấn luyện 3 lần. Lần đầu tiên là huấn luyện bộ nhận dạng tách từng ký tự ra khỏi background. Mục đích là để loại bỏ các false alarm trên các cảnh vật thường gặp các bộ phận cơ thể như khuôn mặt, cánh tay... Qua bước này, bộ nhận dạng đã giảm được false alarm trên các ảnh background không phải cử chỉ. Khâu này cho kết quả rất tốt với detection rate là 100% với false alarm là 3 trên 100 ảnh background (chưa xét ảnh của các cử chỉ khác), 3 vùng ảnh này đều là hình cánh tay và khuôn mặt. Lưu ý là các hình về các bộ phận trên cơ thể người có vai trò rất quan trọng, những vùng ảnh này gây ra nhiều false alarm hơn các ảnh background thuần túy.



Hình 31 - Sự khác biệt giữa bộ nhận dạng huấn luyện trên ảnh background có và không có các bộ phận cơ thể (bên trái là không và bên phải là có)

Tuy nhiên bộ nhận dạng có được chưa tách được cử chỉ cần nhận dạng ra khỏi cử chỉ khác. Ta thấy trong hình 32, khi đưa cử chỉ 'U' và 'F' vào thì bộ nhận dạng cử chỉ 'B' vẫn cho rằng đó là cử chỉ 'B'.



Hình 32 - Kết quả có được khi đưa cử chỉ 'U' và 'F' vào bộ nhận dạng cử chỉ 'B'

Do đó bộ phân loại có được qua bước này được tiếp tục huấn luyện để có thể khắc phục được điểm này. Tập huấn luyện mới sẽ gồm các mẫu postive như cũ và các mẫu negative là tất cả cử chỉ còn lại, sau khi đã phát sinh mỗi cử chỉ thêm 10 hình nữa với kích thước đúng bằng gấp đôi so với các mẫu positive. Việc thực hiện bước này nhằm tạo ra một bộ nhận dạng tập trung phân biệt với các ký tự còn lại. Như chúng ta thấy, việc đưa chính các ký tự còn lại vào bộ các mẫu negative là hoàn toàn có cơ sở, vì như vậy sẽ lọc bớt được các đặc trưng không tốt vốn sẽ được chọn nếu không có bộ lọc này. Kết quả có được sau lần huấn luyện thứ 2 này khá tốt với detection rate là 100% và false alarm là 40 trên 100 mẫu các cử chỉ khác. 40 mẫu này chủ yếu là của các cử chỉ F, U, R và một số tí của chữ A, E, M bởi vì các cử chỉ này khá giống với ‘B’, riêng ‘F’ thì rất giống.

Trên cơ sở này, chúng em tiến hành huấn luyện thêm lớp thứ 3. Lần huấn luyện này là tùy chọn, tùy thuộc vào kết quả lần huấn luyện thứ 2. Có một số ký tự sẽ không cần bộ nhận dạng thứ 3 mà chỉ cần bộ nhận dạng thứ 1 và thứ 2 là đủ. Thông thường, khi chúng ta nhận dạng một cử chỉ thì sẽ có một số cử chỉ khác hay bị nhận nhầm. Do đó, sau khi test qua bộ nhận dạng huấn luyện qua 2 lớp, chúng em sẽ thống kê lại tất cả các chữ bị nhận dạng sai để đưa vào mẫu negative cho bộ nhận dạng thứ 3. Đối với bộ

nhận dạng chữ B như trên thì sẽ được huấn luyện lần 3 với các mẫu negative là các cử chỉ 'F', 'U', 'R', 'A', 'E'. Qua 3 lần huấn luyện liên tiếp như vậy có thể sẽ giảm được nhiều trường hợp bị false alarm. Tuy nhiên, qua thử nghiệm thì thấy lần huấn luyện thứ 3 này không thực sự hiệu quả. Dưới đây là kết quả có được trên tập huấn luyện của bộ nhận dạng cử chỉ 'B':

#	Số mẫu Positive	Số mẫu Negative	Số stages	Hit Rate	FA Rate
1	600	3000	12	100%	0.000039
2	600	900	14	98.3%	0.000042
3	600	150	15	97.8%	0.000043

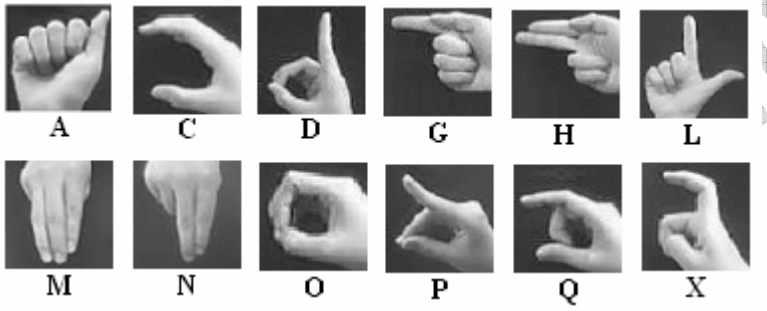
Bảng 2 - Kết quả huấn luyện qua 3 lớp của bộ nhận dạng cử chỉ B

Lưu ý rằng false alarm đạt được trong lần huấn luyện thứ 1 thấp hơn false alarm đạt được ở lần thứ 2 không có nghĩa là lần thứ 2 không tốt. False alarm lần 1 tốt hơn là vì tập negative của nó chỉ chứa các ảnh background (phong cảnh và các bộ phận khác của cơ thể), trong khi lần thứ 2 phải học từ các cử chỉ khác – khó hơn nhiều so với các ảnh background thuần túy – đây cũng là lý do khiến cho detection rate ở lần thứ 2 thấp hơn lần thứ 1. Như đã trình bày ở trên, rõ ràng lần huấn luyện thứ 3 không mang lại hiệu quả đáng kể.

5.3 Kết quả thử nghiệm

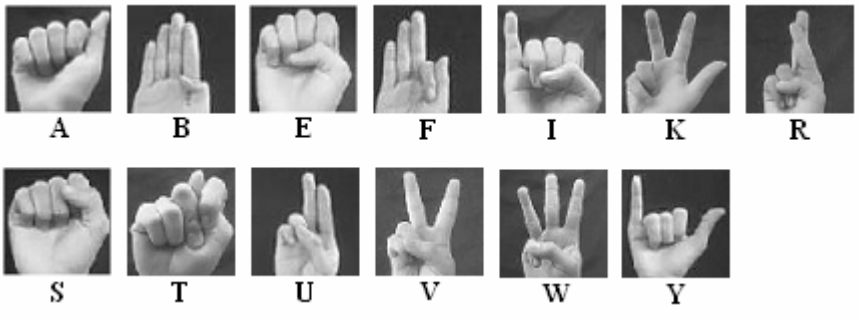
24 bộ nhận dạng cho 24 cử chỉ xây dựng được có trung bình là 14 stages với tổng cộng 123 features. Các bộ nhận dạng đạt detection rate khoảng 96% với false alarm rate là 10^{-5} trên tập huấn luyện.

Để kiểm tra hiệu quả hoạt động của bộ nhận dạng trên các mẫu chưa học, chúng em tiến hành xây dựng 2 tập test, 1 tập đơn giản và 1 tập phức tạp. Tập đơn giản sẽ gồm các cử chỉ khác nhiều với cử chỉ cần nhận dạng và tập phức tạp sẽ gồm những cử chỉ tương đối giống. Với bộ nhận dạng chữ A, tập test thứ nhất sẽ gồm các cử chỉ sau:



Hình 33 - Các cử chỉ trong tập test thứ nhất

Và tập test thứ 2 gồm các cử chỉ:



Hình 34 - Các cử chỉ trong tập test thứ hai

Trong cả 2 tập test, mỗi cử chỉ sẽ gồm 7 ảnh chụp từ 7 người khác nhau. Kết quả thu được như sau:

#	Số Pos	Số Neg	Số vùng ảnh Neg	HR	Số FA
Test1	7	7 x 11 = 77	$77 \times 320 \times 240 \times 1.5 \approx 8870400$	100%	0
Test2	7	7 x 12 = 84	$84 \times 320 \times 240 \times 1.5 \approx 9676800$	100%	213

Bảng 3 - Kết quả thu được của bộ nhận dạng cử chỉ A trên 2 tập test

Đối với tập test thứ 1, bộ nhận dạng đã cho kết quả rất tốt, detection rate đạt 100%, không bỏ sót một chữ A nào, đồng thời false alarm bằng 0, bộ nhận dạng đã loại bỏ được 100% các mẫu negative. Tuy nhiên, khi bước sang tập test thứ với các mẫu khó hơn (ví dụ như cử chỉ E, S, T rất giống với A), bộ nhận dạng bắt đầu bị nhầm đối với các các mẫu negative. Trong số 213 false alarm, có đến 201 vùng ảnh là của các cử chỉ khác, còn lại 12 vùng là của các bức tường, rèm cửa, ... Tuy nhiên, với số lượng vùng

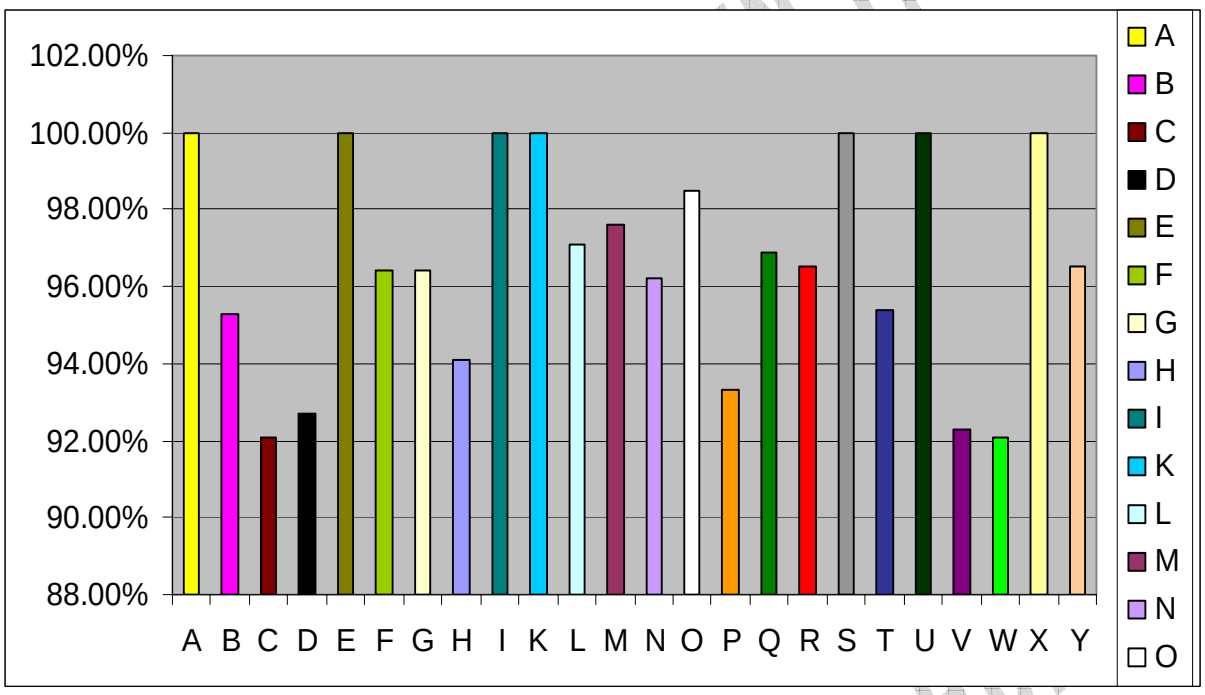
ảnh negative là 9676800 thì để sai 213 vùng ảnh cũng không phải là một kết quả không tốt.

Sau đây là kết quả cụ thể của 24 bộ nhậndạng.

Cử chỉ	Số Positive	Số Negative	Tổng số mẫu	Hitrate	FA
A	1323	7710	592	100%	76
B	630	7980	592	95.3%	148
C	546	7980	592	92.1%	132
D	630	7990	592	92.7%	149
E	1218	7760	592	100%	71
F	613	7960	592	96.4%	98
G	483	8110	592	96.4%	102
H	525	8090	592	94.1%	130
I	462	8110	592	100%	95
K	567	7980	592	100%	67
L	420	8130	592	97.1%	89
M	504	8040	592	98.6%	121
N	483	8110	592	96.2%	118
O	546	8080	592	98.5%	62
P	399	8150	592	93.3%	134
Q	504	8040	592	96.9%	121
R	2163	7300	592	96.5%	156
S	1302	7720	592	100%	82
T	783	7980	592	95.4%	76
U	588	8060	592	100%	134
V	546	8080	592	92.3%	127
W	567	8070	592	92.1%	141

X	462	8080	592	100%	37
Y	441	8040	592	96.5%	119

Bảng 4 - Kết quả thử nghiệm của 24 bộ nhận dạng trên tập test gồm 592 hình



Hình 35 - Biểu đồ thống kê Hit Rate của 24 bộ nhận dạng trên tập test gồm 592 hình

Dựa vào biểu đồ trên chúng ta có thể thấy được rằng tỉ lệ hitrate của các ký tự phụ thuộc khá nhiều vào số lượng samples dùng để huấn luyện. Ngoài ra các ký tự có đặc trưng ít thì thường hitrate sẽ không cao.

Bên cạnh đó, ở các ký tự A, E, S, T ta thấy tỉ lệ false alarm lớn là do các ký tự này khá giống nhau về hình dạng nên rất thường xuyên bị nhận dạng nhầm lẫn nhau. Trường hợp này cũng xảy ra tương tự cho các chữ R, U, V, W hay G và H hoặc M và N.

5.4 So sánh và đánh giá

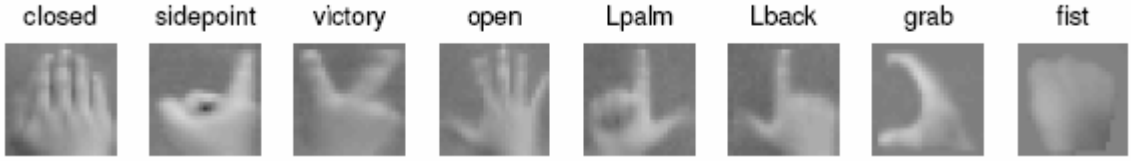
Hệ thống nhận dạng mặt người của *Viola* và *Jones* [1] được đánh giá là hệ thống nhận dạng mặt người nhanh và chính xác nhất hiện nay. *Viola* và *Jones* xây dựng hệ thống này trên tập huấn luyện gồm 4916 ảnh mặt người và 9500 ảnh background. Hình 37 là kết quả thử nghiệm của *Viola* và *Jones*:

	False detections							
Detector	10	31	50	65	78	95	167	422
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%	94.1%

Hình 36 - Kết quả thử nghiệm của Viola và Jones

Kết quả trên có được trên tập test gồm 503 ảnh mặt người và 130 ảnh background. Ta thấy với detection rate là 94.1% thì false alarm của họ là 422, tức là bộ nhận dạng nhận nhầm 422 vùng ảnh thành mặt người trong số 75081800 vùng ảnh background đưa vào. Trong khi đó, do hạn chế về số lượng mẫu, mỗi cử chỉ chỉ có được khoảng 30 mẫu chụp, cộng với số lượng phát sinh do các phép biến đổi thì cũng chỉ có 630 mẫu/cử chỉ thì việc kết quả bộ nhận dạng từng cử chỉ của chúng em khi đạt false alarm là 213 trên 84 ảnh negative cũng có thể xem là một kết quả khả quan.

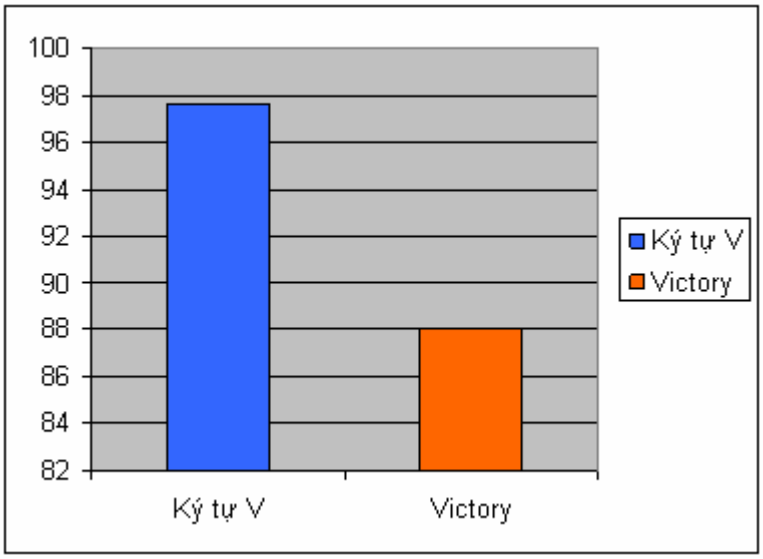
Kolsch [6] cũng áp dụng mô hình Cascade of Boosted Classifiers lên bài toán phân loại cử chỉ với hệ thống gồm 8 cử chỉ:



Hình 37 - Hệ thống 8 cử chỉ trong bài toán của Kolsch

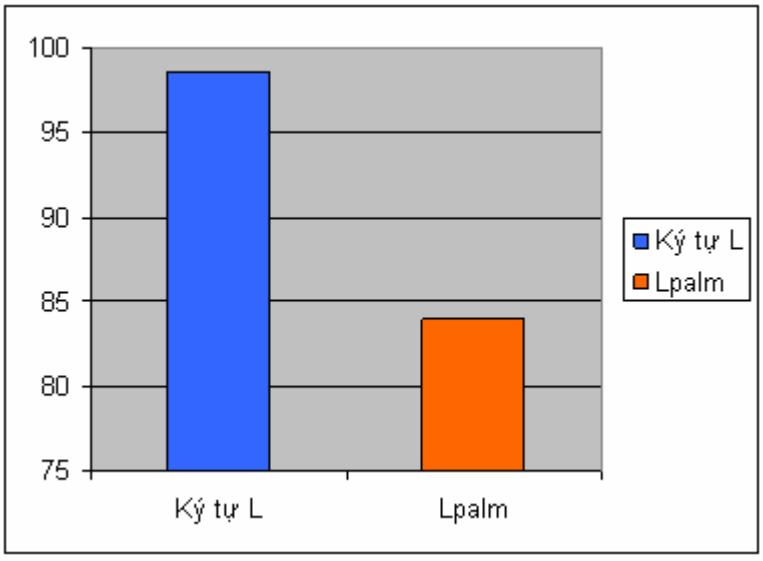
Chúng ta có thể thấy ký hiệu Victory được đề cập có hình dạng giống như cử chỉ V, Lpalm giống với cử chỉ L trong hệ thống cử chỉ mà chúng ta đang sử dụng. Theo kết quả quá trình huấn luyện, với false alarm ở mức 10^{-5} thì tỉ lệ hitrate giữa ký tự V chúng

em đã huấn luyện và ký hiệu Victory trong bài viết của Kolsch (chọn false alarm ở mức 10^{-3}) có thể hiện qua biểu đồ sau:



Hình 38 - Biểu đồ so sánh Hit Rate giữa ký hiệu Victory với cử chỉ V

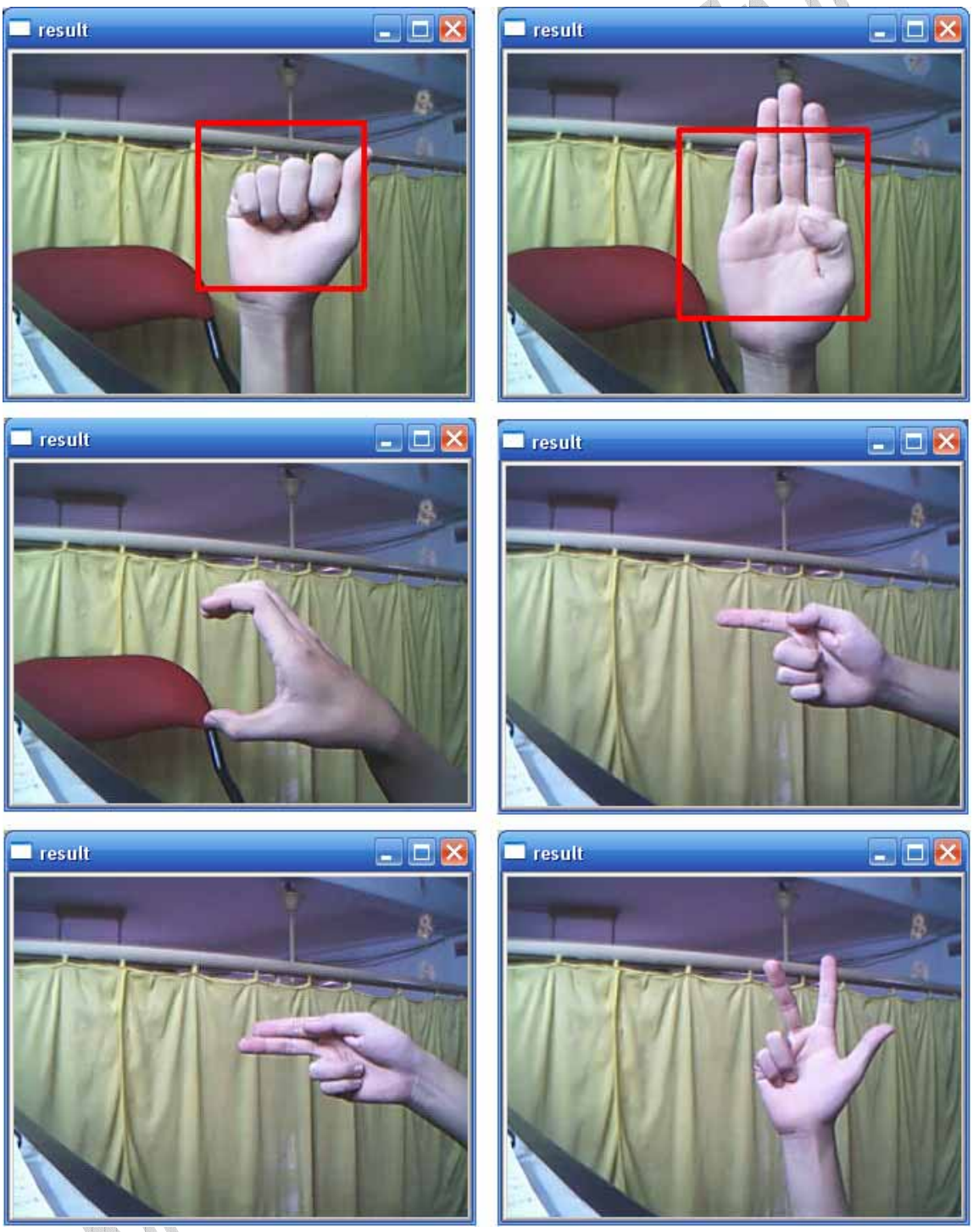
Còn với cùng tỉ lệ false alarm như vậy cho ký tự L và ký hiệu Lpalm, ta sẽ có kết quả như sau:



Hình 39 - Biểu đồ so sánh Hit Rate giữa ký hiệu LPalm với cử chỉ L

Những so sánh trên đây chỉ mang tính tương đối, để cho thấy hiệu quả của bộ nhận dạng cho từng cử chỉ xây dựng trong khóa luận này chứ không thực sự nói lên được

tương quan giữa 2 hệ thống, bởi vì các số liệu trên lấy từ các tập huấn luyện và tập test khác nhau.



Hình 40 - Vài kết quả test của bộ nhận dạng cử chỉ A (cử chỉ B là một trường hợp false alarm)

Chương 6 Tổng kết

6.1 Kết luận

Trên cơ sở tìm hiểu về thuật toán AdaBoost, Haar Feature và mô hình Cascade of Classifiers, chúng em đã áp dụng được mô hình Cascade of Boosted Classifers – vốn được áp dụng trong lĩnh vực nhận dạng mặt người – lên bài toán nhận dạng cử chỉ và đã đạt những những kết quả bước đầu. Chúng em đã tiến hành đủ tất cả các khâu bắt đầu từ việc lấy mẫu bằng cách chụp hình các bạn trên lớp, các Thầy Cô cán bộ giảng dạy trẻ và những người thân trong gia đình chúng em; tiếp đến là chuẩn hóa các hình này; sau đó bắt tay vào chỉnh sửa chương trình huấn luyện được thư viện OpenCV cung cấp; và cuối cùng là tiến hành huấn luyện thử nghiệm rất nhiều lần để có thể chọn ra được kích thước mẫu phù hợp, để có thể chọn được những tham số về *max false alarm*, *min hit rate* ... sao cho hệ thống đạt hiệu quả tốt nhất. Bên cạnh đó, chúng em cũng đã xây dựng một số chương trình tiện ích phục vụ cho việc chuẩn hóa các mẫu huấn luyện (xem phần *Phụ lục C* để biết thêm chi tiết), qua đó biết cách sử dụng các hàm trong thư viện *OpenCV*.

24 bộ nhận dạng cho 24 cử chỉ mà chúng em xây dựng được đã cho kết quả tốt với detection rate khoảng 96%. Tuy false alarm còn tương đối cao, nhưng đó cũng là điều chấp nhận được với số lượng mẫu giới hạn và chất lượng không cao (do được chụp từ webcam với độ phân giải chỉ là 320x240) của mỗi cử chỉ mà chúng em đã thu thập.

Tuy nhiên, khóa luận này chỉ dừng lại ở mức xây dựng được 24 bộ nhận dạng cử chỉ mà chưa thể kết hợp chúng để tạo thành một bộ phân loại cử chỉ như mục tiêu đề ra

ban đầu do hạn chế về thời gian, một phần là vì thời gian thực hiện không đủ, một phần là vì khâu huấn luyện chiếm quá nhiều thời gian.

6.2 Hướng phát triển

Hướng phát triển trước mắt là xây dựng một bộ phân loại cử chỉ dựa trên sự kết hợp các bộ nhận dạng cử chỉ đã có. Từ đó xây dựng một hệ thống phân loại thời gian thực dựa trên một camera để ghi nhận các cử chỉ mà người dùng ra dấu. Một khi có được hệ thống có khả năng "hiểu" được cử chỉ của con người, ta bắt đầu có thể gán cho hệ thống một số tính năng và yêu cầu chúng thực hiện khi nhận được một cử chỉ nào đó – một trong những ứng dụng dạng này chính là điều khiển robot.

Hệ thống hiện tại chỉ hoạt động đối với hình chụp chính diện, nó còn khá nhạy cảm và góc quay của bàn tay. Trong tương lai, ta có thể sử dụng thêm các đặc trưng khác ngoài Haar Feature, có thể áp dụng thêm biến đổi Fourier (như tiếp cận của Kolsch [7]) để có thể nhận dạng được cử chỉ ở mọi góc quay. Đồng thời có thể tiến hành xây dựng nhiều bộ nhận dạng cho một cử chỉ, mỗi bộ sẽ đảm nhiệm một góc nhìn của cử chỉ đó để có thể nhận dạng cử chỉ từ mọi góc độ.

Đối với bài toán nhận dạng cử chỉ động thì còn nhiều hạn chế và hạn chế lớn nhất vẫn là số lượng từ vựng. Mô hình Cascade of Boosted Classifiers cho kết quả rất tốt trên bài toán nhận dạng cử chỉ tĩnh, ta có thể áp dụng nó lên bài toán nhận dạng cử chỉ động bằng cách nhận dạng từng khung hình của quá trình chuyển động, sự chuyển tiếp giữa các khung hình đó có thể được thực xử lý bằng mô hình Markov ẩn.

Với những kết quả đạt được hiện này, để có được một hệ thống thực sự có thể tương tác với con người thông qua cử chỉ thì các chúng ta vẫn còn một chặng đường dài phải đi. Tuy nhiên, điều đó cũng không có nghĩa là không thể. Nếu chúng ta có thể vượt qua được những trở ngại trước mắt, nếu có thể nhận dạng được các cử chỉ động với số lượng tự vựng nhiều hơn thì một thế giới mới, một thế giới mà máy chỉ cần có 1

hệ thống camera là có thể tương tác với con người, là một đích mà chúng ta có thể nhắm đến được.

Phụ lục A: Các thuật ngữ liên quan

Pattern (Mẫu)

Được chia là 2 loại:

- Training Samples (Mẫu huấn luyện): Mẫu huấn luyện là các mẫu dùng cho việc học của một hệ nhận dạng. Trong quá trình huấn luyện, bộ nhận dạng sẽ học từ các mẫu này, thông qua các đặc trưng để nhận ra đối tượng cần nhận dạng. Mẫu huấn luyện gồm 2 loại:
    - *Positive samples*: các mẫu chứa đối tượng cần nhận dạng.
    - *Negative samples*: các mẫu không chứa đối tượng cần nhận dạng.

  Trong quá trình học, bộ nhận dạng sẽ cố gắng tìm các đặc trưng của các mẫu positive mà mẫu negative *không có*, từ đó rút ra được các đặc trưng của đối tượng cần nhận dạng. Khi đưa một mẫu mới vào nhận dạng, bộ nhận dạng sẽ áp các đặc trưng này lên mẫu, nếu thỏa thì mẫu này là đối tượng cần nhận dạng, ngược lại thì không phải.
- Test Samples (Mẫu kiểm thử): Mẫu kiểm thử là các mẫu dùng cho việc kiểm tra tỉ lệ nhận dạng của một hệ nhận dạng. Các mẫu này phải không nằm trong tập huấn luyện. Tỉ lệ nhận dạng đúng các mẫu trong tập kiểm thử nói lên tính hiệu quả của hệ nhận dạng, bởi vì các mẫu này hệ nhận dạng chưa hề được học, nếu nó vẫn nhận dạng đúng chứng tỏ các đặc trưng mà nó rút ra thật sự là các đặc trưng của đối tượng.

Feature (Đặc trưng)

Là các thông tin giúp đối tượng tự định danh mình. Để hệ nhận dạng có thể biết được trong một mẫu đưa vào có chứa đối tượng cần nhận dạng hay không, bộ nhận dạng phải biết được đặc trưng của đối tượng đó. Ví dụ như trong bài toán phân loại cá hồi và cá mú, đặc trưng của cá hồi có thể là "độ sáng không dưới 0.5 và chiều dài vảy không

quá 3 cm". Khi đó, nếu mẫu đưa vào thỏa 2 đặc trưng này thì nó là cá hồi, ngược lại là cá mú.

Trong bất cứ bài toán nhận dạng nào, thông thường thì không tồn tại một đặc trưng đơn nào có thể giúp nhận dạng tốt đối tượng ("độ sáng không dưới 0.5" chưa đủ để kết luận một mẫu đưa vào là "cá hồi"), mà nó đòi hỏi phải kết hợp nhiều đặc trưng với nhau, khi đó chúng được gọi là *không gian đặc trưng*.

Threshold (Ngưỡng)

Ngưỡng là giá trị "ranh giới" giữa các lớp. Trong ví dụ ở trên, thì giá trị "0.5" và "3" được gọi là ngưỡng. Ngưỡng có thể hiệu chỉnh được, và thông thường được chọn ra bằng thực nghiệm (Người ta thử bộ nhận dạng với các giá trị ngưỡng khác nhau để chọn ra ngưỡng cho tỉ lệ nhận dạng đúng tốt nhất).

Classifier (Bộ phân loại)

Một hệ nhận dạng gồm nhiều bộ phân loại, mỗi bộ phân loại ứng với một lớp, gồm một hay nhiều luật dạng If…then… với các đặc trưng và các ngưỡng tương ứng. Khi một mẫu được đưa vào bộ phân loại, bộ phân loại sẽ kiểm tra xem nó có thỏa các luật này hay không, nếu thỏa thì nó được xếp vào lớp tương ứng với bộ phân loại này, nếu không thì sẽ được đưa qua các bộ phân loại kế tiếp (nếu có).

Detector (Bộ nhận dạng)

Chức năng tương tự như bộ nhận dạng nhưng nó có phạm vi hẹp hơn, nó chỉ có nhiệm vụ cho biết một mẫu có thuộc về một lớp cụ thể nào đó hay không. Detector và classifier đôi khi được dùng thay thế cho nhau, vì detector có thể được xem như một classifier cho 2 lớp là "có" và "không".

Hit Rate (Detection Rate)

Là tỉ lệ nhận dạng *đúng* các *object* (các đối tượng cần nhận dạng). Ví dụ trong bài toán nhận dạng mặt người, *Hit Rate* = 0.95 có nghĩa là trong số 100 mẫu mặt người, bộ phân loại chỉ nhận ra được 95 mẫu (5 mẫu còn lại được bộ phân loại cho là *background*).

False Positive (False Alarm)

Là tỉ lệ nhận dạng *sai* các *background* (các mẫu không phải đối tượng cần nhận dạng). Ví dụ như False Alarm = 0.01 có nghĩa là cứ 100 mẫu *background* thì có 1 mẫu bị bộ phân loại lầm tưởng là *object*.

Weak classifier

Là các classifier đơn giản chỉ cần có độ chính xác trên 50%.

Strong classifier

Là classifier có độ chính xác cao , được xây dựng từ sự kết hợp các weak classifier.

Phụ lục B: Các chương trình dùng cho huấn luyện

Tạo sample

Việc tạo samples được thực hiện bằng chương trình *CreateSamples*, một chương trình đi kèm trong thư viện *Intel OpenCV* có chức năng tạo file dữ liệu cho các mẫu (positive lẫn negative) từ các file ảnh (bmp, jpg ...), đồng thời cho phép phát sinh thêm các ảnh từ một hay nhiều ảnh ban đầu bằng cách áp dụng các phép xử lý ảnh như rotate, dilate, erode ... lên các ảnh này. Chương trình này đã được chúng em sửa đổi để cho đáp ứng được nhu cầu xây dựng bộ nhận dạng cử chỉ. Các sửa đổi bao gồm cách tạo samples, thêm phép tịnh tiến cho việc phát sinh các ảnh. Chương trình sau khi chỉnh sửa có tên là *MyCreateSamples*.

Chương trình này có 4 chức năng chính sau:

1. Phát sinh ra các hình tương tự từ 1 hay nhiều ảnh ban đầu

   Xác lập giá trị cho các tham số sau:

   - *-info*: tập tin chứa tên các tập tin hình cần phát sinh.
   - *-dir*: tên thư mục sẽ chứa các tập tin hình được phát sinh.
   - *-w -h*: kích thước của hình gốc (mặc định 24x24).
   - *-num*: số lượng hình sẽ được phát sinh từ mỗi hình đưa vào (mặc định là 10).

2. Tạo dữ liệu huấn luyện

   Xác lập giá trị cho các tham số sau:

   - *-info*: tập tin liệt kê danh sách các ảnh positive (ảnh chụp của cử chỉ đã qua chuẩn hóa).
   - *-vec*: tập tin dữ liệu sẽ phát sinh để dùng cho quá trình huấn luyện.
   - *-nongen*: chương trình sẽ ghi dữ liệu của các ảnh vào thẳng file vec.
   - *-gen*: với mỗi ảnh trong file info, chương trình sẽ phát sinh ra *num* ảnh tương tự, sau đó ghi dữ liệu của tất cả các ảnh này vào file vec.

- *-num*: số ảnh sẽ phát sinh cho từng ảnh positive (chỉ sử dụng kèm với tham số -gen).
- *-w -h*: kích thước ảnh positive.

3. Phủ kín các ảnh nền bằng một ảnh cho trước

   Xác lập giá trị cho các tham số sau:

   - *-info*: tập tin ảnh sẽ phát sinh.
   - *-img*: tập tin ảnh dùng để phủ.
   - *-bg*: tập tin chứa danh sách các ảnh nền cần được phủ.
   - *-w -h*: kích thước ảnh –img
   - *-ran*: áp dụng các phép biến đổi ngẫu nhiên lên ảnh trước khi phủ lên ảnh nền (ảnh nền sẽ được phủ bằng nhiều ảnh khác nhau – những ảnh được phát sinh từ -img).
   - *-nonran*: dùng ảnh –img phủ kín tất cả các ảnh nền.

4. Phủ kín nhiều ảnh nền bằng nhiều ảnh cho trước

   Xác lập giá trị cho các tham số sau:

   - *-info*: tập tin liệt kê danh sách các ảnh dùng để phủ.
   - *-bg*: tập tin chứa danh sách các ảnh nền cần được phủ.
   - *-w -h*: kích thước ảnh –img.
   - *-ran*: áp dụng các phép biến đổi ngẫu nhiên lên ảnh trước khi phủ lên ảnh nền (ảnh nền sẽ được phủ bằng nhiều ảnh khác nhau – những ảnh được phát sinh từ các ảnh trong -info).
   - *-nonran*: dùng lần lượt các ảnh trong –info phủ kín tất cả các ảnh nền.

Huấn luyện

Haartraining là ứng dụng dùng để xây dựng một hệ nhận dạng theo mô hình *Cascade of Boosted Classifiers*. Đây cũng là một chương trình mã nguồn mở được cung cấp như một bộ phận của thư viện *Intel OpenCV*. Chương trình này cũng được chúng em sửa

đổi, sửa cấu trúc dữ liệu biểu diễn feature và thêm vào đó các feature sử dụng trong bài toán nhận dạng cử chỉ, đồng thời thêm các chức năng khác như phát sinh file log, phát sinh file mô tả các feature được chọn, ... Chương trình sau khi sửa đổi có tên là *Parsehaartraining*.

Chương trình có những tham số sau:

- *-data*: thư mục sẽ chứa cấu trúc cây cascade.
- *-vec*: tập tin dữ liệu positive phát sinh bằng *CreateSamples*.
- *-bg*: tập tin liệt kê danh sách các ảnh negative (các ảnh background và ảnh của các cử chỉ khác với cử chỉ cần nhận dạng)
- *-subwnd*: tên file log lưu tọa độ các feature được chọn. Nếu không xác lập tham số này thì chức năng này sẽ không được thực hiện.
- *-log*: tên file log các thông số của quá trình huấn luyện. Nếu không nhập vào thì chương trình sẽ lấy tên mặc định là logxxx.txt bắt đầu từ 000 và tăng dần trong khi file logxxx.txt đã tồn tại.
- *-npos*: số lượng positive samples dùng để huấn luyện.
- *-nneg*: số lượng negative samples dùng để huấn luyện.
- *-nstages*: số stage dự kiến sẽ train.
- *-nsplits*: số feature sử dụng trong mỗi weak classifier. Bài toán nhận dạng cử chỉ sử dụng –nsplits là 1.
- *-sym*: dùng khi mẫu nhận dạng có tính chất đối xứng nhằm giảm false alarm.
- *-minhitrate*: tỉ lệ nhận dạng đúng tối thiểu các mẫu positve mà mỗi stage classifier phải đạt. *minhitrate* của cả cấu trúc cascade sẽ là –nstages$^{\text{-minhitrate}}$.
- *-maxfalsealarm*: tỉ lệ nhận dang sai tối đa các mẫu negative mà mỗi stage classifier được phép mắc phải. maxfalsealarm của cả cấu trúc cascade sẽ là –nstages$^{\text{-maxfalsealarm}}$.

- *-eqw*
- *-mode* <BASIC (mặc định) | CORE | ALL>: các loại haar features sẽ sử dụng. BASIC gồm các kiểu feature không xoay, CORE bao gồm cả các feature xoay, ALL bao gồm thêm feature 2 hình vuống lồng nhau. (Xem ***4.1.2*** để biết thêm chi tiết).
- *-w -h*: kích thước của mẫu positive.
- *-bt*<DAB | RAB | LB | GAB>: thuật toán boost muốn sử dụng: Discreate AdaBoost, Real AdaBoost, LogitBoost và Gentle AdaBoost. Bài toán nhận dạng cử chỉ trong khóa luận này sử dụng GAB.

Nhận dạng

Chúng em tự xây dựng chương trình H*aardetecting* (phỏng theo chương trình *facedetect* dựa trên sự hỗ trợ của *OpenCV)* dùng để tiến hành nhận dạng. Chương trình này sẽ sử dụng cấu trúc cây cascade phát sinh được từ ứng dụng *Haartraining* để nhận dang một mẫu đưa vào. Chương trình có 2 tham số đơn giản như sau:

- -data: thư mục chứa cấu trúc cây cascade phát sinh từ ứng dụng Haartraining.
- -img: ảnh cần nhận dạng.
- -cam: webcam dùng để chụp ảnh và nhận dạng trực tuyến, không sử dụng tham số này nếu nhận dạng trên ảnh tĩnh.
- -w -h: kích thước mẫu positive đã tiến hành huấn luyện.

Phụ lục C: Các chương trình tiện ích

Để có thể tiến hành huấn luyện đòi hỏi các ảnh chụp phải được chuẩn hóa. Việc chuẩn hóa các ảnh chụp nếu thực hiện bằng tay hay bằng các chương trình xử lý đồ họa sẵn có sẽ mất rất nhiều thời gian vì các chương trình này không sát với nhu cầu. Do đó, chúng em đã xây dựng một tập các ứng dụng hỗ trợ cho việc chuẩn hóa này. Hầu hết các chương trình này đều được viết với sự hỗ trợ của thư viện *Open CV*.

1. *PathToTxt*: với input là một thư mục, chương trình sẽ phát sinh một tập tin chứa đường dẫn đến tất cả các tập tin có trong thư mục này (các tập tin ở cấp độ bất kỳ). Output của chương trình này dùng cho chương trình *MyCreateSamples*.
2. *ImageCropper:* chương trình giúp cắt nhanh các vùng ảnh (vùng chứa bàn tay) và tự động lưu nó vào một file mới với cùng tên với ảnh mà nó xử lý.
3. *MakeGrayScaleImg*: chương trình giúp chuyển cùng lúc nhiều hình về ảnh grayscale, đồng thời cho phép chọn có thay đổi kích thước ảnh hay không (thay đổi thành một kích thước cố định hay đổi theo tỉ lệ).
4. *MyDemHist*: Dùng để threshold, tô đen vùng nền cho các ảnh dựa vào histogram.

Tài liệu tham khảo

[1] P. Viola and M. J. Jones. *Robust real-time face detection*. International Journal of Computer Vision, 57(2):137--154, May 2004.

[2] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for Object Detection. In *International Conference on Computer Vision*, 1998.

[3] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.

[4] Rainer Lienhart and Jochen Maydt. An Extended Set of Haarlike Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[5] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[6] M. Kolsch and M. Turk. Robust Hand Detection. In *Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition*, May 2004.

[7] M. Kolsch and M. Turk. Analysis of Rotational Robustness of Hand Detection with a Viola-Jones Detector. In *Proc. IEEE Intl. Conference on Pattern Recognition, 2004.*

[8] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[9] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

[10] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[11] Rowley, H., Baluja, S., and Kanade, T. 1998. Neural network-based face detection. *IEEE Patt. Anal. Mach. Intell.*, 20:22–38.

[12] Schneiderman, H. and Kanade, T. 2000. A statistical method for 3D object detection applied to faces and cars. In *InternationalConference on Computer Vision*.

[13] Roth, D., Yang, M., and Ahuja, N. 2000. A snowbased face detector. In *Neural Information Processing* 12.

[14] Eng-Jon Ong and Bowden, R. A Boosted Classifier Tree for Hand Shape Detection. In *Proc. IEEE Intl. Conference on* A*utomatic Face and Gesture Recognition*, 2004.

[15] William T. Freeman, Michal Roth. Orientation Histograms for Hand Gesture Recognition. In *Proc. IEEE Intl. Wkshp. on Automatic Face and Gesture Recognition*, Zurich, June, 1995.

[16] R. Bowden and M. Sarhadi. Building temporal models for gesture recognition. In *Proc. BMVC.*, volume 1, pages 32–41, 2000.

[17] X. Zhu, J. Yang, and A. Waibel. Segmenting Hands of Arbitrary Color. In *Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition*, 2000.

[18] Nolker, C. Ritter, H.: Illumination Independent Recognition on Deictic Arm Postures, *Proc. 24th Annual Conf. of the IEEE Industrial Electronics Society*, Germany, pp. 2006- 2011. (1998).

[19] Duy-Dinh Le, Shin'ichi Satoh. Feature Selection By AdaBoost For Efficient SVM-Based Face Detection, In *Information Technology Letters*, Vol.3, pp. 183-186, Kyoto, Japan, Sep 2004.