LỜI CẢM ƠN

Đầu tiên, em xin chân thành gửi lời tri ân đến Thầy Bùi Trọng Tú. Luận văn này sẽ không thể hoàn thành nếu không có sự hướng dẫn, tin tưởng và tạo cơ hội của thầy. Em xin chân thành cảm ơn sự chỉ bảo của thầy.

Em cũng xin bày tỏ sự biết ơn sâu sắc đến Thầy Nguyễn Xuân Thuận. Thầy đã tận tình hướng dẫn, luôn theo sát tiến độ thực hiện đề tài để giúp đỡ và góp ý cho em trong thời gian qua, cũng như tiếp thêm động lực, ý chí giúp em hoàn thành được luận văn.

Em xin chân thành cảm ơn quý Thầy Cô trong khoa Điện tử - Viễn thông trường Đại học Khoa Học Tự Nhiên Tp.Hồ Chí Minh đã tận tình giảng dạy, truyền đạt những kiến thức quý báu trong những năm học vừa qua, giúp em có được một nền tảng kiến thức vững chắc để hoàn thành luận văn này.

Nhân đây, con xin được bày tỏ lòng cảm ơn sâu sắc đến Mẹ, cảm ơn những tình cảm, sự vất vả, và hy sinh của Mẹ đã giúp con có thể theo đuổi con đường học tập cho đến ngày hôm nay.

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 em/mình hoàn thành công việc một cách tốt nhất.

Đề tài này thuộc một lĩnh vực nghiên cứu mới trên thế giới và lại rất mới ở Việt Nam cộng thêm năng lực hạn chế của người thực hiện cho nên đề tài chắc chắn là chưa hoàn thiện và có nhiều thiếu sót. Hy vọng sẽ nhận được nhiều ý kiến đóng góp để đề tài này hoàn thiện hơn.

Thành phố Hồ Chí Minh, Tháng 7/2012

Nguyễn Lâm Hoài Phong

LỜI NÓI ĐẦU

Kể từ khi ra đời máy tính, việc tương tác với chúng như thế nào luôn là một thử thách đối với các nhà phát triển. Chuột và bàn phím được xem là những thiết bị tương tác đầu vào cổ điển và ngày nay chúng vẫn là những thiết bị chính trong tương tác với máy tính. Theo thời gian thì việc tương tác với máy tính cũng trở nên đa dạng hơn, tất cả đều hướng đến mục tiêu đơn giản và hiệu quả, để đạt được mục tiêu này thì phương pháp tương tác cần hướng đến con người hơn nữa, nói cách khác chúng ta cần nhân cách hóa cách thức tương tác với máy tính.

Nhân cách hóa cách thức tương tác với máy tính là một chủ đề phức tạp và rất hấp dẫn các nhà nghiên cứu. Các hệ thống nhận dạng giọng nói, nhận dạng cử chỉ hay kết hợp cả hai phương pháp nhân dạng trên đang được nghiên cứu và dần áp dụng trong thực tế.

Ngày nay, những công nghệ mới cho phép hệ thống máy tính nhận biết cử chỉ của người dùng sẽ nhanh chóng thay thế các loại chuột, bàn phím truyền thống và trong một số trường hợp, thay thế cả các màn hình cảm ứng. Những công nghệ mới này sẽ không chỉ đơn giản hoá việc thực hiện "hàng đống" các thao tác truyền thống với PC mà còn cả với các nhiệm vụ phức tạp như điều khiển robot, điều khiển mô hình 3 chiều, làm việc với các hình ảnh y tế... Đó chính là vấn đề đặt ra cho bài toán phát hiện và nhận dạng 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.

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 bàn tay, còn cử chỉ động là chuyển động theo một quỹ đạo nhất định của một hay hai bàn tay. Nhận dang cử chỉ động bao hàm cả nhận dạng cử chỉ tĩnh và một số xử

lý trên chuyển động nên hết sức phức tạp. Đề tài này chỉ tập trung vào bài toán nhận dạng cử chỉ tĩnh.

Mục tiêu của đề tài là xây dựng một hệ thống nhận dạng cử chỉ có khả năng hoạt động đáp ứng thời gian thực. Do thời gian thực hiện đề tài ngắn cùng với mục tiêu là tập trung vào giải thuật, nên đề tài chọn thực hiện trên nền tảng bộ xử lý tín hiệu số (*Digital Signal Processor - DSP*). Bộ xử lý tín hiệu số DSP được lựa chọn bởi khả năng xử lý tín hiệu số rất mạnh cùng khả năng tái lập trình đơn giản.

Luận văn được trình bày gồm 6 chương:

- **Chương 1**: Tổng quan về hệ thống nhận dạng cử chỉ
- **Chương 2**: Phân vùng màu da
- **Chương 3**: Theo dõi đối tượng
- **Chương 4**: Tổng quan về Kit DSP TMS320DM642
- **Chương 5**: Xây dựng hệ thống nhận dạng cử chỉ trên DSP
- **Chương 6**: Kết luận và hướng phát triển

MỤC LỤC

DANH SÁCH CÁC CHỮ VIẾT TẮT

DANH SÁCH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1. TỔNG QUAN VỀ HỆ THỐNG NHẬN DẠNG CỬ CHỈ

Sự ra đời của máy tính đã giúp ích rất nhiều cho công việc và cuộc sống của con người. Con người có thể sử dụng máy tính cho nhiều mục đích khác nhau như soạn thảo, in ấn, lưu trữ văn phòng, hổ trợ việc quản lý, điều khiển tự động hóa, hay dùng cho mục đích giải trí như nghe nhạc, xem phim, chơi game…Tuy nhiên, cho đến thời điểm hiện tại việc giao tiếp giữa con người và máy tính còn khá nhiều bất tiện do phụ thuộc vào các công cụ hổ trợ như chuột, bàn phím, remote…Thực tế, con người chỉ cảm thấy thuận tiện và thoải mái khi giao tiếp với máy tính thông qua cách mà con người giao tiếp với nhau, tức là máy tính phải hiểu được các cử chỉ của con người. Đó chính là vấn đề đặt ra cho bài toán phát hiện và nhận dạng 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.

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 bàn tay, còn cử chỉ động là chuyển động theo một quỹ đạo nhất định của một hay hai bàn tay. Nhận dang cử chỉ động bao hàm cả nhận dạng cử chỉ tĩnh và một số xử lý trên chuyển động nên hết sức phức tạp. Đề tài này chỉ tập trung vào bài toán nhận dạng cử chỉ tĩnh.

Hệ thống nhận dạng cử chỉ trong đề tài này là một hệ thống có khả năng nhận dạng và phân loại 4 cử chỉ ứng với 4 mệnh lệnh.

Bài toán đặt ra hai khó khăn lớn: hệ thống nhận dạng cử chỉ không những phải chính xác mà còn phải đáp ứng được thời gian thực. Yêu cầu về tính chính xác luôn là khó khăn của bất cứ một bài toán nhận dạng nào. Riêng với bài toán nhận dạng cử chỉ thì việc nhận dạng chính xác lại càng khó khăn hơn bởi vì……

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 việc nhận dạng.

Để có thể đạt được mục tiêu trên, đã có nhiều cách tiếp cận được đưa ra. Feeman sử dụng đặc trưng *Orientation Histogram* [4] với thời gian tính toán nhanh, nhưng lại đòi hỏi hình ảnh nhận dạng phải là hình ảnh chụp cận cảnh của bàn tay. Đồng thời, các 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 [5] 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 khó đạt được sự vững chắc.

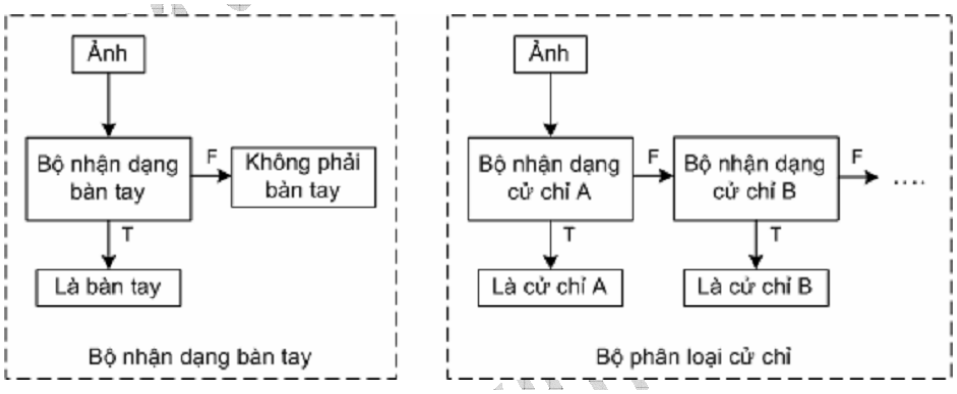
Ngoài ra, còn có cách tiếp cận dựa trên màu sắc của da trên bàn tay [6] bởi vì màu da tương đối thuần nhất. Đây cũng không phải một cách tiếp cận tốt vì các bộ nhận dạng xây dựng trên đặc trưng là màu da rất nhạy cảm với ánh sáng, và hệ thống sẽ nhận dạng sai khi mẫu đưa vào có chứa các đối tượng khác 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 khối lượng tính toán. Trong khi đó, có một phương pháp đạt được sự cân đối giữa 2 mặt này và đang được sử dụng khá thành công trong các hệ thống nhận dạng như: nhận dạng gương mặt, nhận dạng mắt, nhận dạng con người…Phương pháp này do *Viola* và *Jones* [1] đề nghị. Nó sử dụng một mô hình phân loại nhiều tầng (*Cascade of Boosted Classifiers*) với đặc trưng được sử dụng là *Haar Feature* [1].

Cascade of Boosted Classifiers là một cấu trúc cây mà ở mỗi tầng là một bộ phân loại (*classifier*). Bộ phân loại này được xây dựng bằng thuật toán *AdaBoost*

trên nguyên tắc kết hợp của nhiều bộ phân loại yếu (*weak classifier* - các bộ phân loại đơn giản chỉ cần có độ chính xác trên 50%) để tạo ra mộ bộ phân loại mạnh (*strong classfier* - 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 quan hệ giữa các vùng ảnh với nhau.

Bộ phân loại cử chỉ được xây dựng từ nhiều bộ nhận dạng cử chỉ, trong đó mỗi bộ nhận dạng ứng với 1 cử chỉ cụ thể.



Hình 1.1. Bộ phân loại được tạo thành từ sự kết hợp các bộ nhận dạng

Đề tài này xây dựng xây dựng các bộ nhận dạng cho từng cử chỉ theo mô hình *Cascade of Boosted Classifiers*. Sau khi đã xây dựng các bộ nhận dạng cho từng cử chỉ, bộ phân loại cử chỉ sẽ được xây dựng từ các bộ nhận dạng này.

1.1. AdaBoost

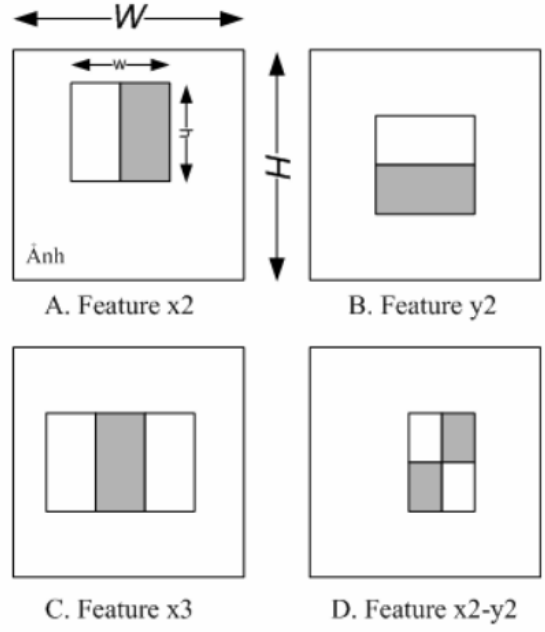
AdaBoost [3] là một thuật toán Boosting, ý tưởng cở bản của AdaBoost là kết hợp các bộ phân loại yếu để xây dựng bộ phân lớp mạnh.

Là một cải tiến của thuật toán 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 bộ phân loại yếu được xây dựng, thuật toán sẽ tiến hành cập nhập lại trọng số để chuẩn bị cho việc xây dựng bộ phân loại yếu kế tiếp: tăng trọng số của các mẫu bị nhận dạng sai và giảm trọng số của các mẫu được nhận dạng đúng bởi bộ

phân loại yếu vừa xây dựng. Bằng cách này, các bộ phân loại yếu sau có thể tập trung vào các mẫu mà các bộ phân loại yếu trước nó chưa làm tốt. Sau cũng, các bộ phân loại yếu sẽ được kết hợp để tạo nên một bộ phân loại mạnh.

1.2. **Đặc trưng Haar**

Đặc trưng Haar là một loại đặc trưng thường được dùng cho bài toán nhận dạng trên ảnh. Đặc trúng Haar 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 A và B bên dưới, giá trị của đặc trưng Haar cho bởi một ảnh bằng hiểu số giữa tổng các điểm ảnh thuộc hai vùng hình chữ nhật sáng và tối. Trong hình C thì giá trị đặc trưng Haar bằng tổng các điểm ảnh trong hai vùng chữ nhật bên ngoài trừ tổng các điểm anh trong hình chữ nhật ở giữa. Trong hình D, giá trị đặc trưng Haar bằng tổng các điểm ảnh nằm trong hai hình chữ nhật màu tối trừ tổng các điểm ảnh nằm trong hai hình chữ nhật màu sáng.



Hình 1.2. Các đặc trưng Haar cơ bản

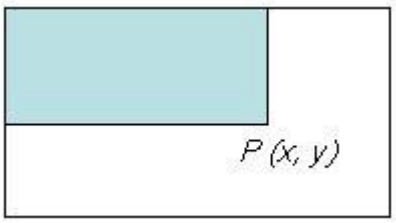
Tác dụng của đặc trưng Haar là nó thể hiện được thông tin về các đối tượng trong ảnh (vì nó biểu diễn mối liên hệ giữa các bộ phận của đối tượng), điều mà bản thân từng điểm ảnh không thể hiện được.

Hình 1.3. Đặc trưng Haar cho mặt người

Như vậy có thể thấy rằng, để tính các giá trị đặc trưng Haar, ta phải tính tổng các điểm ảnh nằm trong các vùng trên ảnh. Nhưng để tính toán các giá trị của các đặc trưng Haar cho tất cả các vị trí trên ảnh đòi hỏi chi phí tính toán khá lớn, làm cho thời gian xử lý tang đáng kể. Để khắc phục điều này, *Viola* và *Jones* đưa ra một khái niệm gọi là *Integral Image*.

1.3. **Integral Image**

Integral Image [2] là một mảng hai chiều với kích thước bằng với kích thước của ảnh cần tính các đặc trưng Haar, với mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên và bên trái của nó. Bắt đầu từ vị trí phía trên - bên trái đến vị trí phía dưới - bên phải của ảnh, việc tính toán này đơn thuần chỉ đựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh.



Hình 1.4. Điểm P(x,y)

$$P(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

$$D = P4 - P3 - P2 + P1$$

$$= P(x_4, y_4) - P(x_3, y_3) - P(x_2, y_2) + P(x_1, y_1)$$

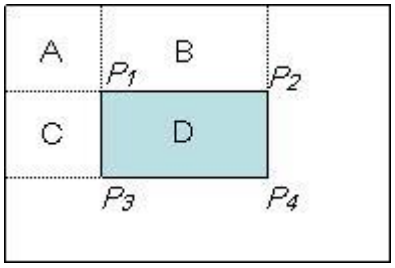
Sau khi đã tính được *Integral Image*, việc tính tổng các giá trị điểm ảnh của một vùng bất kỳ nào đó trên ảnh thực hiện rất đơn giản theo cách sau:

- Giả sử ta cần tính tổng các giá trị mức xám của vùng D như trong hình dưới, ta có thể tính như sau:

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

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

$$D = \underbrace{(x4,y4)}_{A+B+C+D} - \underbrace{(x2,y2)}_{(A+B)} - \underbrace{(x3,y3)}_{(A+C)} + \underbrace{(x1,y1)}_{A}$$

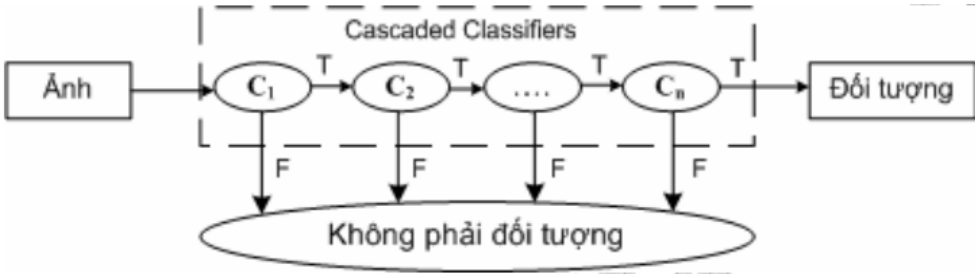


Hình 1.5. Các tính tổng điểm ảnh trong 1 hình chữ nhật bất kỳ

1.4. **Bộ phân loại nhiều tầng (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 rất dễ nhận ra. Đối với những mẫu này, ta chỉ cần xét một hay vài đặc trưng đơn giản là có thể nhận diện mà không cần xết 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 huấn luyện. Việc này làm tốn thời gian xử lý một cách không cần thiết.

Cascade of Classifiers được xây dựng chính là để rút ngắn thời gian xử lý giảm thiểu false alarm cho bộ phân loại. *Cascade* gồm nhiều *stage*, mỗi *stage* của cây sẽ là một *stage classifier*. Một mẫu để được phân loại là đối tượng thì nó cần phải qua hết tất cả các *stage* của cây. Các *stage classifier 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 classifier* 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 *stage* đầu tiên, giúp đáp ứng tốt nhất với sự gia tăng độ phức tạp của các mẫu đưa vào, đồng thời rút ngắn thời gian xử lý.



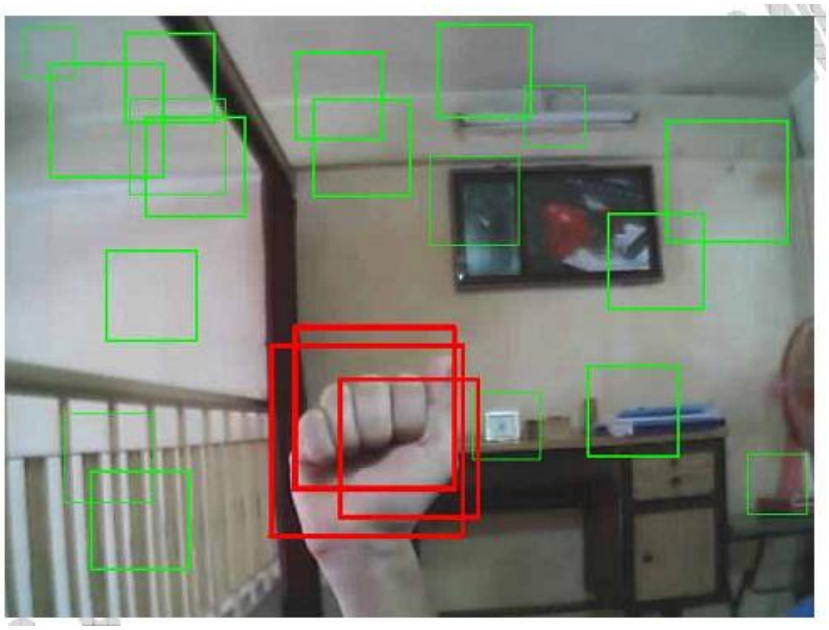
Hình 1.6. Bộ phân loại nhiều tầng

Ngoài ra, trong phương pháp của Viola và Jones, người ta còn nhắc đến thuật ngữ *Cascade of Booosted Classifiers. Cascade of Booosted Classifiers* là mô hình phân loại nhiều tầng (*Cascade of Classifiers*) với mỗi bộ phân loại được xây dựng bằng AdaBoost sử dụng các đặc trưng Haar.

1.5. **Hoạt động của bộ nhận dạng cử chỉ**

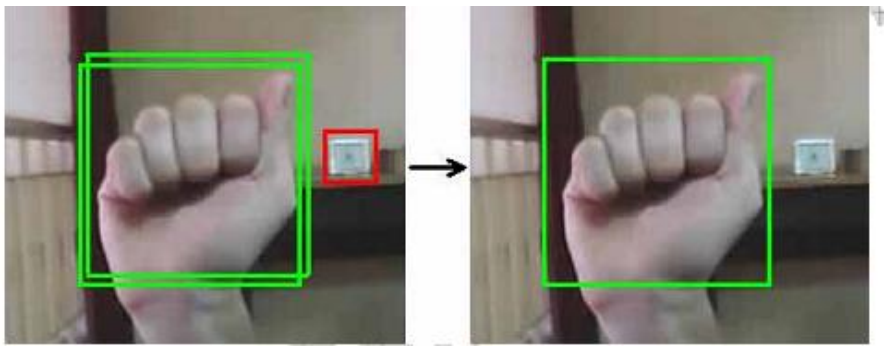
Khi đưa một ảnh vào nhận dạng, bộ nhận dạng sẽ phải xét tất cả các vùng ảnh với kích thước khác nhau trích ra được từ ảnh này để có thể đưa ra kết quả. Kích thước khởi đầu của vùng ảnh sẽ là một cửa sổ bằng với kích thước của mẫu *positive* trong quá trình huấn luyện. Các cửa sổ này sẽ được dịch theo chiều ngang và dọc với khoảng cách mỗi lần dịch phụ thuộc vào tỉ lệ kích thước giữa ảnh gốc và cửa sổ hiện tại cho đến khi phủ kín ảnh cần nhận dạng. Sau đó, cửa sổ sẽ được mở rộng ra với tỉ lệ 1.2 và tiếp tục quá trình duyệt ảnh như trên đến khi cửa sổ được mở rộng bằng kích thước ảnh.

Nhờ có cấu trúc phân tầng (*cascade*), các vùng ảnh không liên quan bị loại nhanh từ những *stage* đầu tiên.



Hình 1.7. 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 gần như là giống nhau. Các vùng này có thể đề đều được bộ nhận dạng xác nhận là cử chỉ, khiến cho một cử chỉ được nhận dạng nhiều hơn 1 lần.



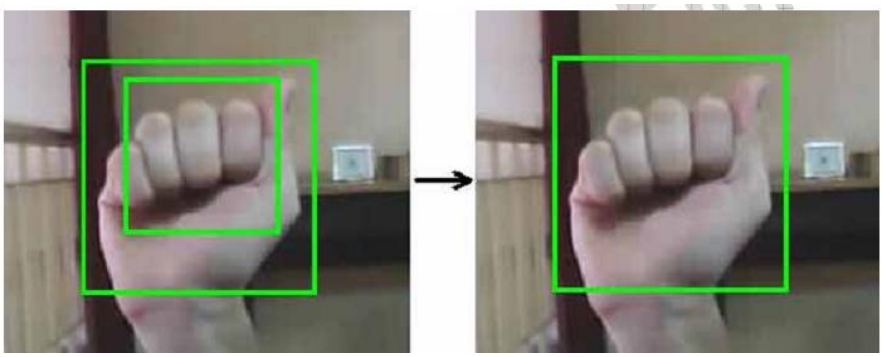
Hình 1.8. Khắc phục trường hợp nhiều vùng ảnh kế cận nhau bằng cách lấy vùng ảnh trung bình

Để khắc phục đều này, sau khi đã có được vị trí các vùng mà bộ nhận dạng xác nhận là cử chỉ, ứng dụng nhận dạng sẽ tìm và nhóm các vùng có vị trí gần nhau

bằng cách thay chúng bằng một vùng ảnh duy nhất có được cách lấy trung bình các vùng ảnh này.

Ngoài ra, do tỉ lệ nhận dạng không phải là 100% nên khi tiến hành nhận dạng, có thể có vài mẫu background sẽ bị nhầm là cử chỉ. Trong khi bàn tay trong hình đưa vào luôn có nhiều hơn 1 vùng ảnh chứa nó được bộ nhân dạng đánh giá là cử chỉ, thì các mẫu background bị nhận dạng sai thường nằm tách biệt. Do đó, ứng dụng nhận dạng sử dụng thêm khái niệm *min neighbor*, tức là số vùng ảnh gần nhau tối thiểu phải có để một vùng ảnh có thể được phân loại là cử chỉ. Giá trị này thường được thiết lập bằng 2. Tuy nhiên, cần thận trọng trong việc thiết lập giá trị cho min neighbor, vì nếu giá trị này quá lớn, có thể làm giảm tỉ lệ nhận dạng đúng của bộ nhận dạng.

Bên cạnh đó, có thể khi nhận dạng còn tồn tại vấn đề các vùng ảnh lồng vào nhau. Đối với các trường hợp này, đơn giản là chỉ cần loại bỏ tất cả các vùng ảnh bên trong, chỉ giữ lại vùng ảnh bên ngoài.



Hình 1.9. Đối với các vùng ảnh lồng nhau, các vùng ảnh bên trong sẽ bị loại bỏ

Trên thực tế, nếu chỉ sử dụng riêng phương pháp của Viola và Jones cho hệ thống nhận dạng cử chỉ thì hệ thống không đáp ứng được yêu cầu hoạt động thời gian thực. Vì thời gian tính toán quá lớn do phải tiến hành tìm kiếm bàn tay trên toàn bộ frame kể cả các vùng mà tại đó chắc chắn không thể chứa bàn tay như vùng back-ground…

Thay vì phải tiến hành tìm kiếm trên toàn bộ frame, ta có thể giới hạn vùng tìm kiếm tại các vùng có chứa màu da người. Để làm được việc đó, hệ thống cần phải có một khối chức năng thực hiện tách chọn vùng màu da trên frame ảnh đưa vào.

Ngoài ra, đề tài còn xây dựng thêm một khối có chức năng theo dõi bàn tay. Khối này thực hiện việc theo dõi vùng có chứa bàn tay sau khi đã được xác định ở frame trước. Ở những frame ảnh kế tiếp, hệ thống chỉ cần tiến hành nhận dạng cử chỉ bên trong vùng đang được theo dõi, việc này làm giảm đáng kể chi phí tính toán giúp cho hệ thống hoạt động đáp ứng thời gian thực.

CHƯƠNG 2. PHÂN VÙNG MÀU DA

2.1. **Giới thiệu**

Như đã trình bày trong phần trước, dựa vào màu sắc của da người cũng là một trong những phương pháp được dùng để phát hiện cử chỉ. Tuy nhiên nếu chỉ đơn thuần sử dùng màu sắc không thôi thì rất khó có thể đạt được hiệu quả cao trong phát hiện cử chỉ. Vì trong các khung cảnh thì có rất nhiều vật có màu sắc tương tự như màu của da. Tuy nhiên nếu kết hợp phương pháp này với các phương pháp khác lại có thể mang lại hiệu quả cao. Vì kinh nghiệm cho thấy màu da người có đặc tính màu riêng biệt, và đặc tính này cho phép dễ dàng nhận ra đâu là da người. Và thông thường trong hướng tiếp cận phát hiện cử chỉ dựa trên thông tin xuất hiện trong ảnh, thì màu da được sử dụng như một bước phân vùng các vùng ảnh có màu sắc giống màu da, điều đó cho phép giảm không gian tìm kiếm cử chỉ, cải thiện hiệu năng của hệ thống tìm kiếm. Do đó nhiều mô hình đã được xây dựng để có thể phát hiện được da người.

Khi xây dựng hay mô hình hóa một hệ thống phát hiện hay phân tách vùng màu da với mục đích sử dụng cho việc phát hiện cử chỉ, người ta thường đặt ra ba vấn đề chính [9]. Thứ nhất là mô hình đó được xây dựng trong không gian màu nào, thứ hai là hàm phân phối của màu da được mô hình hóa chính xác đến mức độ nào và cuối cùng là sẽ xử lý vùng màu da được phân vùng cho nhận biết cử chỉ như thế nào. Trong phần này, sẽ chỉ đề cập đến hai câu hỏi trên, còn việc xử lý vùng da như thế nào cho việc phát hiện cử chỉ, sẽ đề cập đến trong chương sau, với một phương pháp cụ thể được chọn để sử dụng cho đề tài này.

Phương pháp được được đề cập trong phần này là phương pháp phát hiện da người dựa trên đặc tính điểm ảnh, nghĩa là sẽ phân lớp điểm ảnh thành hai lớp, một là lớp điểm anh có thuộc màu da và lớp kia không phải là màu da. Các điểm ảnh là hoàn toàn độc lập với nhau. Ngược lại với phương pháp này là phương pháp dựa trên đặc tính vùng ảnh.

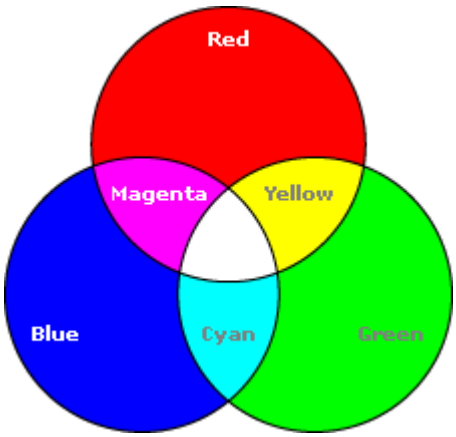
Phát hiện màu da dựa trên đặc tính điểm ảnh có một lịch sử phát triển khá dài, tuy nhiên trong khuôn khổ phần tổng quan này, chỉ đề cập và so sánh những kỹ thuật đã được công bố và được đánh giá hiệu quả.

Mục đích cuối cùng của phần tổng quan này là thu tập các kỹ thuật đã được công bố, mô tả những ý tưởng chính của kỹ thuật đó, tổng hợp và đưa ra những ưu điểm, nhược điểm và những đặc trưng của từng kỹ thuật. Từ đó sẽ đưa ra quyết định lựa chọn kỹ thuật phù hợp dùng để phân vùng màu da áp dụng cho đề tài này.

2.2. **Không gian màu sử dụng cho mô hình hóa màu da**

Trong lĩnh vực đo màu, cũng như các lĩnh vực trong truyền tín hiệu hình ảnh và video sử dụng rất nhiều không gian màu với các tính chất khác nhau. Và trong số đó nhiều không gian màu được áp dụng cho vấn đề mô hình hóa màu da. Sau đây là tóm lược nhóm các không gian màu được sử dụng rộng rãi nhất cũng như các tính chất của chúng.

2.2.1. *Không gian màu RGB*



Hình 2.1. Không gian màu RGB

RGB là không gian màu cơ bản được áp dụng từ lâu cho màn hình CRT. Trong không gian màu này, mỗi điểm màu là sự kết hợp của ba thành phần đơn màu (đỏ - Red, xanh lá cây- Green và xanh da trời : Blue). Đây là một trong những không gian màu được sử dụng phổ biến nhất cho việc xử lý và lưu trữ dữ liệu ảnh

số. Tuy nhiên do tính tương quan cao giữa các kênh, giá trị cảm nhận không đồng nhất, sự pha trỗn giữa dữ liệu thành phần màu và dữ liệu về độ sáng mà không gian RGB không được ưa thích sử dụng cho việc phân tích màu cũng như trong các thuật toán nhận dạng dựa trên màu sắc.

2.2.2. *Không gian RGB chuẩn hóa*

Không gian RGB chuẩn hóa là không gian màu nhận được từ không gian RGB cơ bản theo công thức chuẩn hóa đơn giản sau đây:

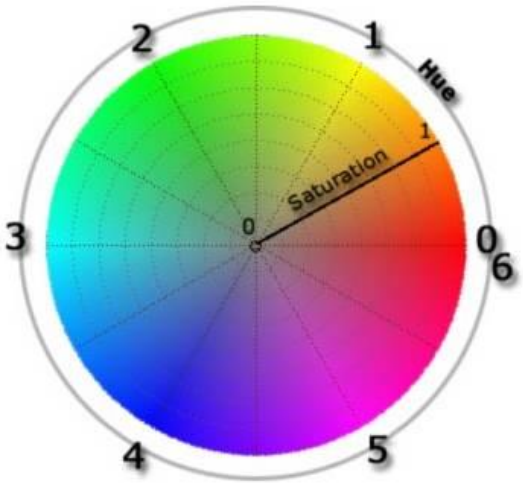
$$r = \frac{R}{R+G+B}; g = \frac{G}{R+G+B}; b = \frac{B}{R+G+B};$$

Có thể dễ dàng thấy rằng, trong không gian này, r+g +b = 1. Do đó chỉ cần hai trong ba thành phần trên là đủ để biểu diễn không gian màu này, thành phần thứ ba sẽ không còn giá trị và có thể được bỏ qua, để rút ngắn được số chiều của không gian này. Hai thành phần còn lại thường được gọi là các thành phần "màu tinh khiết" (*pure colors*). Thông thường, hai thành phần *r* và *b* thường được giữ lại, còn *b* bị rút bỏ đi. Tính chất cần chú ý của không gian màu này đó là tính bất biến đối với bề mặt. Nghĩa là, nếu như không quan tâm đến ánh sáng xung quanh, thì không gian chuẩn hóa RGB là bất biến đối với sự thay đổi về hướng bề mặt liên quan đến nguồn chiếu (tất nhiên là dưới một vài giả thiết nhất định). Kết hợp với phép chuyển đổi đơn giản từ không gian màu RGB cơ bản mà không gian RGB chuẩn hóa này ngày càng được sử dụng rộng rãi trong nhiều lĩnh vực, trong đó có lĩnh vực nhận dạng.

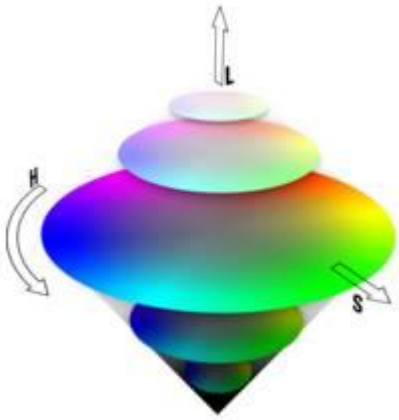
2.2.3. *HSI, HSV, HSL - Độ bão hòa của màu*

Không gian màu dựa trên tính bão hòa màu được giới thiệu khi có những nhu cầu trong việc xác định số lượng tính chất màu. Chúng miêu tả màu sắc với những giá trị thuộc về trực giác, dựa trên ý kiến của các họa sỹ về những trạng thái khác nhau của màu sắc, trạng thái bão hòa cũng như từng tông màu khác nhau. *Hue* biểu thị cho màu trỗi (như màu đỏ, màu xanh lá cây, màu đỏ tía và màu vàng) của một

vùng ảnh, *saturation* (độ bão hòa) là thước đo cho giới mức ngưỡng màu của một vùng ảnh. Các khái niệm như "*intensity*" (cường độ), "*lightness*" (tính dịu) hay "*value*" (giá trị) liên quan đến độ sáng của màu. Giá trị trực giác của các thành phần trong không gian màu này và sự phân biệt rõ ràng giữa độ sáng với cá thành phần màu của không gian màu là ưu điểm mà giúp cho không gian này được sử dụng phổ biến trong vấn đề phần vùng màu da.



Hình 2.2. Mô hình biểu diễn 2 thành phần màu H,S trong HSV



Hình 2.3. Mô hình hệ màu HSL

Công thức chuyển từ không gian RGB sang không gian này như sau:

$$H = arccos \frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}$$

$$S = 1 - 3\frac{\min(R,G,B)}{R+G+B}$$

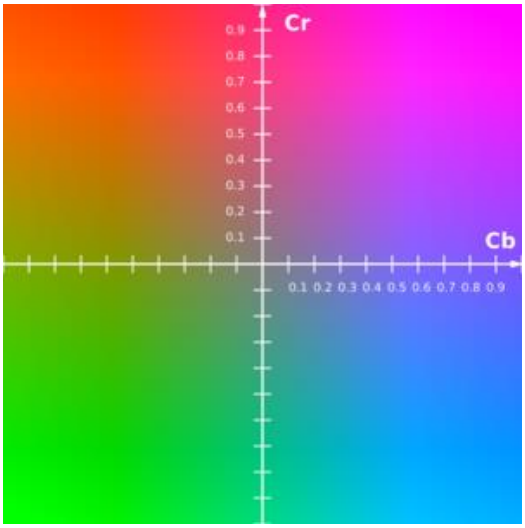
$$V = \frac{1}{3}(R+G+B)$$

Ngoài ra còn có thể tính *Hue* và *Saturation* bằng cách sử dụng hàm *log* cho các thành phần màu của không gian màu RGB. Phương pháp này có thể làm giảm sự độc lập của các thành phần màu theo mức sáng.

Hệ tọa độ cực giữa *Hue* và *Saturation* có thể gây ra nhiều khó khăn trong mô hình màu da, chính vì vậy người ta còn chuyển nó sang hệ tọa độ Đề các theo công thức sau:

$$X = S\cos H\,, Y = S\sin H$$

2.2.4. *YCrCb*



Hình 2.4. Không gian màu YcbCr

YCrCb là không gian màu được sử dụng nhiều trong vấn đề nén ảnh. Màu sắc được biểu diễn bởi luma (đó là giá trị độ sáng được tính toán từ không gian RGB), gồm ba thành phần, một thành phần là tổng các trọng số từ RGB, hai thành phần màu khác nhau Cr và Cb được tạo ra bằng cách từ hai thành phần *Red* và *Blue* trong không gian màu RGB. Công thức để chuyển đổi như sau:

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cr = R - Y$$

$$Cb = B - Y$$

Việc chuyển đổi đơn giản, tính phân chia rõ ràng của độ sáng và các thành phần màu là những đặc tính giúp cho không gian này lôi cuốn các nhà nghiên cứu sử dụng cho việc mô hình hóa màu da.

2.2.5. *Các hệ tọa độ không gian màu khác*

Bên cạnh YcrCb, một vài không gian màu khác được tạo ra từ chuyển đổi tuyến tính không gian RGB được sử dụng trong vấn đề phát hiện màu da. Như là TSL, YES, YUV hay YIQ. Tuy nhiên chúng ít được sử dụng hơn.

2.3. **Mô hình hóa màu da**

Mục đích cuối cùng của phát hiện màu da là xây dựng một quy tắc có tính quyết định. Đây là quy tắc sẽ giúp phân biệt một điểm ảnh là da hay không phải là da người. Thông thường, quy tắc này sẽ thiết lập một giá trị đo cho phép tính toán mức độ tương đồng giữa một điểm ảnh màu với đặc trưng màu da. Giá trị đo này được thiết lập như thế nào, công thức ra sao tùy thuộc vào từng phương pháp mô hình hóa màu da.

2.3.1. *Xác định ngưỡng cụ thể một điểm ảnh là màu da*

Trong một số không gian màu, phương pháp xây dựng và xếp lớp màu da bằng cách xác định rõ ràng (thông qua một số quy tắc) biên giới các giá trị của điểm ảnh là màu da hay không. Ví dụ như:

Trong không gian RGB:

(R,G,B) được xếp thuộc lớp màu da nếu như:

R>95 và G>40 và B>20 và

Max(R,G,B) – Min(R,G,B) >15 và

|R-B| > 15 và R>g và R>B

Trong không gian màu YCbCr:

370 < 2Cr + Cb < 400 và

5 < Cr – Cb < 60 và

7Y + 5(Cr - Cb) < 1750 và

10Y – 3(Cr - Cb) > 100

Tính đơn giản của phương pháp này cũng thu hút nhiều sự tập trung nghiên cứu. Ưu điểm dễ thấy của phươương pháp này đó là tính đơn giản của quy tắc nhận biết màu da. Điều này cho phép phân lớp một cách nhanh chóng và dễ dàng. Tuy nhiên kết quả đạt được khi phân lớp là không cao trong trường hợp tổng quát. Vì vậy khó khăn chính của phương pháp này nếu muốn có được hệ số nhận dạng cao đó là phải tìm ra được một không gian màu thích hợp cũng như các quy tắc tốt để nhận biết màu da trong không gian màu này.

Hiện nay người ta đang đề xuất sử dụng thuật toán máy học để tìm ra một không gian màu thích hợp cũng như các quy tắc phân lớp màu da với mong muốn có được hệ số nhận dạng cao. Tuy nhiên đó cũng chỉ mới là đề xuất và chưa có một kết quả cụ thể của một nghiên cứu nào được công bố.

Tuy nhiên, dựa vào kết quả đạt được, chúng ta vẫn có thể tìm ra được những quy tắc cho phép nhận biết chắc chắn một điểm ảnh không phải là màu da. Những quy tắc này có thể được sử dụng làm bước lọc khởi tạo cho các phương phân lớp pháp phức tạp hơn giữa vùng màu da và vùng không phải màu da. Nó giúp cho quá trình phân lớp được thực hiện nhanh chóng hơn và đỡ tốn công hơn.

2.3.2. *Mô hình hóa màu da sử dụng phân phối không tham số*

Ý tưởng chính của phương pháp mô hình hóa màu da không tham số đó là ước lượng phân phối màu da từ dữ liệu huấn luyện mà không xuất phát từ một mô hình rõ ràng nào của màu da. Kết quả của phương pháp này thường được biểu diễn dưới dạng một bản đồ phân bố màu da (SPM – *Skin Probability Map*). Mỗi một giá trị phân bố được gán cho mỗi điểm trong không gian màu.

2.3.2.1. Bảng tra cứu chuẩn hóa (LUT – *Lookup Table*)

Một số thuật toán phát hiện mặt người và bám sát mặt người sử dụng một lược đồ mức xám dựa trên hướng tiếp cận phân vùng các điểm ảnh là màu da. Không gian màu được lượng tự hóa thành từng nhóm, mỗi một nhóm đáp ứng cho một khoảng các thành phần màu. Các nhóm lược đồ này được tham chiếu tới một bảng gọi là bảng tra cứu. Mỗi một nhóm lưu trữ một số lượng lần xuất hiện của một màu khi tiến hành huấn luyện ảnh da người. Sau quá trình huấn luyện, biểu đồ sẽ tính toán và chuẩn hóa, chuyển sang giá trị biểu đô trong phân phối xác suất miền rời rạc:

$$P_{skin}(C) = \frac{skin[c]}{Norm}$$

Trong đó, skin[c] nhận giá trị của nhóm lược đồ, đáp ứng cho véc tơ màu c, Norm là một hệ số chuẩn hóa (tổng tất cả các giá trị của các nhóm biểu đồ) hay là giá trị lớn nhất của một nhóm biểu đồ. Giá trị chuẩn hóa của của bảng tra cứu các nhóm biểu đồ là căn cứ để cho phép quyết định một màu có là màu da hay không.

2.3.2.2. Phân lớp Bayes (Bayes Classifier)

Giá trị của $P_{skin}(c)$ trong công thức trên là một điều kiện xác suất – P(c|skin) – xác suất một màu quan sát c là một pixel màu da. Và xác suất thích hợp được dùng để phát hiện màu da đó là P(skin|c) – xác suất quan sát màu được màu da khi xuất hiện một giá trị màu c rời rạc. Để tính giá trị này, ta sử dụng công thức Bayes quen thuộc:

$$P(skin|c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin) + P(c|-skin)P(-skin)}$$

Trong đó P(c|skin) và P(c|-skin) đượoc tính trực tiếp từ biểu đồ màu da và không màu da. Xác suất toàn phần P(skin) và P(-skin) thì được ước lượng từ một số lượng các mẫu là màu da và không màu da trong tập mẫu huấn luyện. Bất đẳng thức P(skin|c) > Θ, trong đó Θ là một giá trị ngưỡng, có thể được sử dụng để trở thành quy tắc trong phát hiện màu da.

Công thức trên đôi khi hơi phức tạp, và để có thể tránh điều này, nếu như thực sự không cần phải biết một cách chính xác suất P(skin|c) và P(-skin|c) mà chỉ cần biết tỉ số giữa chúng thì người ta thường đưa về công thức như sau:

$$\frac{P(skin|c)}{P(-skin|c)} = \frac{P(c|skin)P(skin)}{P(c|-skin)P(-skin)}$$

So sánh công thức này với một ngưỡng có thể tạo ra một quy tắc cho phép phát hiện tỉ số *màu da/không phải màu da*. Sau một vài phép biến đổi, chúng tanhận được công thức

$$\frac{P(skin|c)}{P(-skin|c)} > \Theta$$

$$\Theta = \mathrm{K} \times \frac{1 - P(skin)}{P(skin)}$$

Công thức trên có thể thấy rằng, việc chọn lựa giá trị của xác suất toàn phần không ảnh hưởng đến chất lượng của bộ phát hiện, vì với bất kì một xác suất toàn phần P(skin) đều có thể chọn được một giá trị K phù hợp sao cho giá trị của ngưỡng là Θ.

Hai ưu điểm dễ thấy của phương pháp mô hình hóa phân phối không tham số đó là: thứ nhất, chúng có thể huấn luyện và sử dụng được một cách nhanh chóng. Thứ hai, chúng độc lập với lý thuyết về hình dạng của phân phối màu da (điều này không đúng trong mô hình hóa màu da có tham số). Tuy nhiên nhược điểm của phương pháp này đó là chúng yêu cầu nhiều bộ nhớ để lưu trữ và không có khả năng nội suy hay tạo ra dữ liệu huấn luyện. Lấy ví dụ như, chúng ta lượng tử hóa điểm ảnh trong không gian RGB về 8bit cho mỗi màu, khi đó chúng ta phải cần một mảng có tới 2^{24} phần tử để lưu trữ tập tất cả các xác suất của mô hình. Để có thể giảm bớt kích thước này bằng cách loại bỏ những dữ liệu huấn luyện nhỏ lẻ, không gian màu thường sử dụng kích thước 128×128×128, 64×64×64, 32×32×32. Theo như nghiên cứu thì kích thước 32×32×32 là kích thước không gian mang lại hiệu quả cao nhất.

2.3.3. *Mô hình hóa phân phố màu da có tham số*

Hầu hết các mô hình màu da không tham số dựa trên biểu đồ xám đều yêu cầu rất nhiều bộ nhớ và hiệu năng của chúng phụ thuộc hoàn toàn của tập ảnh huấn luyện cố định. Vì vậy cần có một mô hình màu da có thê tự thêm hoặc tự tạo ra dự liệu huấn luyện điều đó dẫn đến sự ra đời của mô hình phân phối có tham số.

2.3.3.1. Mô hình dựa trên phấn phối Gaussian đơn.

Phân phối màu da có thể được mô hình hóa bởi phân phối Gaussian thêm vào hàm mật độ xác suất. Định nghĩa như sau:

$$P(c|skin) = \frac{1}{2\pi|\Sigma_s|^{1/2}} e^{-\frac{1}{2}(c-\mu_s)^T \Sigma_s^{-1}(c-\mu_s)}$$

Ở đây, c là một véc tờ màu, μ_s và Σ_s là hai tham số phân phối (véc tơ trùng bình và ma trận hiệp phương sai). Các tham số của mô hình được ước lượng thông qua quá trình huấn luyện bởi công thức sau:

$$\mu_s = \frac{1}{n}\sum_{j=1}^{n} c_j$$

$$\Sigma_s = \frac{1}{n-1}\sum_{j=1}^{n}(c_j - \mu_s)(c_j - \mu_s)^T$$

Trong đó, n là tổng số các mẫu màu da . Xác suất p(c|skin) có thể được tính trực tiếp mức độ tương tự màu da (*likehood skin color*) hoặc có thể tính bằng khoảng cách Mahalanobis từ vé tơ màu c, véc tơ trung bình μ_s ma trận hiệp phương sai Σ_s. Công thức tính khoảng cách Mahalanonbis:

$$\lambda_s(c) = (c - \mu_s)^T \sum_{s}^{-1}(c_j - \mu_s)$$

Phương pháp mô hình hóa dựa trên phân phối đơn Gaussian đã được triển khai và nghiên cứu.

2.3.3.2. Mô hình kết hợp dựa trên phân phối Gaussian

Một mô hình công phu, phức tạp hơn, có khả năng biểu diễn được phân phối phức tạp đó là mô hình két hợp dựa trên phân phối Gaussian. Đây là mô hình mở rộng từ mô hình đơn Gaussian trên, trong trường hợp này, hàm phân phối mật độ xác suất là:

$$P(c|skin) = \sum_{i=1}^{k} \pi_i p_i(c|skin)$$

Trong đó, k là số lượng các thành phần được kết hợp, π_i là tham số kết hợp, thỏa mãn ràng buộc $\sum_{i-1}^{k} \pi_i = 1$, và $p_i(c|skin)$ thỏa mãn hàm phối mật độ xác suất Gaussian, với mỗi véc tơ trung bình và ma trận hiệp phương sai của nó. Huấn luyện mô hình được thực hiện với một kĩ thuật được biết đến nhiều gọi là thuật toán

kì vọng tối đa (EM - *Expectation Maximization*), trong đó giả sử rằng số lượng các thành phần k là đã biết trước. Chi tiết việc huấn luyện mô hình kết hợp Gaussian với thuật toán EM này có thể được tìm thấy trong nhiều nghiên cứu. Việc phân lớp trong mô hình kết hợp Gaussian được thực hiện nhờ việc so sánh xác suất p(c|skin) với một vài giá trị ngưỡng. Việc chọn lựa số lượng thành phần k ở đây là quan trọng. Vì nó ảnh hưởng đến độ chính xác của việc huấn luyện cho mô hình. Theo như những nghiên cứu hiện nay, k = 8 là sự lựa chọn mang hiệu năng cao nhất cho mô hình kết hợp phân phối Gausian.

2.3.3.3. Đa phân phối Gausian

Mức độ gần đúng của các nhóm màu da với phân phối Gaussian 3D trong không gian YCbCr đã được miêu tả trong nhiều bài báo. Một số lượng khác nhau các thuật toán phân nhóm K-trung bình được sử dụng cho nhóm Gaussian thực hiện việc huấn luyện mô hình. Các điểm ảnh được phân lớp thành lớp màu da nếu như khoảng cách Mahalanobis từ véctơ màu c đến trung tâm của cụm gần nhất trong mô hình nhỏ lớn hơn một ngưỡng cho trước.

Tất cả các phương pháp mô hình hóa theo tham số được miêu tả như trên(ngoại trừ phương pháp 2.3.3.3) đều tính toán trên mặt phẳng các thành phần màu của không gian màu mà bỏ qua thông tin về độ sáng.

Dĩ nhiên, khi một mô hình phân phối cụ thể được sử dụng, sẽ có câu hỏi đặt ra về sự xác thực về giá trị của mô hình đó. Hiển nhiên, mô hình độc lập với hình dạng của phân phối trong không gian màu thì càng tốt hơn, do đó mô hình không tham số xét về mặt này hiển nhiên sẽ tốt hơn mô hình có tham số. Tuy nhiên do yêu

cầu quá cao về bộ nhớ mà khi đánh giá hiệu năng thì mô hình có tham số lại có hiệu năng cao hơn.

2.3.4. *Mô hình hóa thích ứng với màu da của người dùng*

Ưu điểm chính của các phương pháp sử dụng các ngưỡng để phân lớp điểm ảnh là màu da hay không đó là tính đơn giản và tính trực giác cao trong các quy tắc phân lớp. Tuy nhiên, điểm khó khăn đó là cần phải tìm được cả một không gian màu tốt và các quy tắc xứng đáng trong không gian đó. Phương pháp được đề xuất hiện này sử dụng thuật toán máy học để có thể tìm được không gian và các quy tắc thích hợp, tuy nhiên đề xuất này vẫn đang là một vấn đề mở trong tương lai.

Các phương pháp sử dụng mô hình hóa không tham số thật sự nhanh trong cả việc huấn luyện và phân lớp, độc lập với phân bố hình dạng của màu da và cả không gian màu. Tuy nhiên, phương pháp này lại yêu cầu quá nhiều bộ nhớ lưu trữ và phụ thuộc cố dịnh vào tập dữ liệu huấn luyện.

Các phương pháp mô hình hóa có tham số cũng xử lý khá nhanh. Hơn nữa chúng lại có khả năng tự tạo ra các dữ liệu huấn luyện phù hợp, chúng được miêu tả bằng một số lượng không nhiều các tham số và đặc biệt chúng cần không đáng kể bộ nhớ lưu trữ. Tuy nhiên, hiệu năng của chúng phụ thuộc nhiều vào hình dạng của phân phối màu da. Bên cạnh đó, cả hai phương pháp mô hình hóa màu da có tham số và không có tham số đều sử dụng các phép tính khá phức tạp không phù hợp với nền tảng DSP mà đề tài sử dụng.

Với mục đích là sử dụng bộ nhận biết màu da để tiến hành phân vùng màu da, giảm không gian tìm kiếm cử chỉ trên ảnh và có thể hoạt động tốt trên nền tảng DSP. Vì vậy, phương pháp cần thiết cho đề tài phải có hiệu năng cao, thời gian thực hiện nhanh, yêu cầu bộ nhớ không lớn và sử dụng các phép tính toán đơn giản. Đồng thời, để bộ nhận biết màu da có thể hoạt động tốt trong các điều kiện ánh sáng thay đổi và có thể nhận biết được nhiều loại màu da khác nhau. Đề tài này sử dụng phương pháp mô hình hóa có khả năng thích nghi với điều kiện ánh sáng và màu da

của người dùng bằng cách trích xuất thông tin màu da trên gương mặt của chính người dùng đó. Phương pháp này được tiến hành qua các bước:

- Xác định vị trí gương mặt trên ảnh (sử dụng phương pháp phát hiện gương mặt cửa Viola và Jones).
- Tính *histogram* vùng chứa gương mặt trong mô hình màu RGB. Kết quả *histogram* này được dùng làm bảng tra (*Lookup Table*) trong đó mỗi giá trị RGB sẽ tương ứng với một giá trị là số lần xuất hiện điểm ảnh có giá trị RGB đó trong vùng chứa gương mặt hay có thể coi là giá trị xác suất điểm ảnh đó là màu da.
- Xác định 1 điểm ảnh có phải màu da hay không bằng cách lấy giá trị RGB của điểm ảnh đó rồi dựa vào *histogram* để lấy ra xác suất điểm ảnh đó là màu da. Nếu giá trị xác suất này nhỏ hơn một ngưỡng cho trước thì điểm ảnh này không phải màu da, ngược lại là màu da.

2.4. **Xử lý hình thái học Erosion và Dilation**

Hiểu một cách đầy đủ thì "Morphology" là hình thái học và cấu trúc của đối tượng, hay nó mô tả những phạm vi và các mối quan hệ giữa các thành phần trong một đối tượng. Hình thái học quá quen thuộc trong các lĩnh vực ngôn ngữ học và sinh học. Trong ngôn ngữ học, hình thái học là sự nghiên cứu về cấu trúc của từ, tập hợp từ, câu… và đó cũng là một lĩnh vực nghiên cứu nhiều năm nay. Còn trong sinh học, hình thái học lại chú trọng tới hình dạng của một cá thể hơn, chẳng hạn có thể phân tích hình dạng của một chiếc lá để từ đó có thể nhận dạng được đó là loại cây gì; nghiên cứu hình dạng của một nhóm vi khuẩn, dựa trên các đặc điểm nhận dạng để phân biệt chúng thuộc nhóm vi khuẩn nào… Tùy theo trường hợp cụ thể mà có cách phân lớp phù hợp với nó: Có thể phân lớp dựa trên những hình dạng của mặt cắt như (Elip, tròn…), kiểu và mức độ của những hình dạng bất quy tắc (lồi, lõm, …), những cấu trúc trong (lỗ, đường thẳng, đường cong, …) mà đã được tích lũy qua nhiều năm quan sát.

Tính khoa học của Hình thái học số mới chỉ thực sự phát huy khả năng của nó kể từ khi máy tính điện tử số ra đời và làm cho hình thái học trở lên thông dụng, có nhiều tính năng mới. Những đối tượng trong Hình thái học ta có thể coi như là tập hợp của các điểm ảnh, nhóm lại theo cấu trúc ma trận hai triều. Những thao tác toán học rời rạc trên tập hợp điểm đó được sử dụng để làm rõ những nét đặc trưng riêng

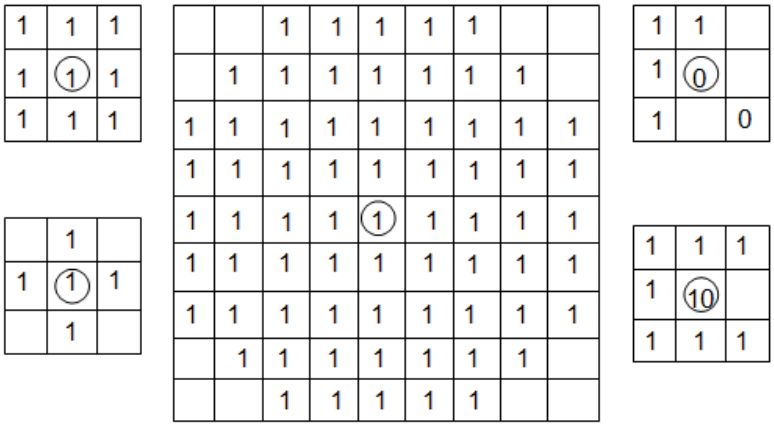
của hình dạng đối tượng, do vậy có thể tính toán được hay nhận biết được chúng một cách dễ dàng. Phần lớn các phép toán hình thái học được định nghĩa từ hai phép toán cơ bản là phép toán co nhị phân (Erosion) và phép toán giãn nhị phân (Dilation) .

Phép toán cơ sở được kết hợp với một đối tượng là tiêu chuẩn của các phép toán tập hợp như phép hợp (Union), Phép giao (InterSection), và phép bù (Complement) cộng với phép tịnh tiến nào đó. Vì vậy trong phần tiếp theo sẽ trình bày các khái niệm về tập hợp thường được sử dụng trong phép toán hình thái.

Phép toán hình thái cung cấp hai phép toán cơ bản là phép giãn nhị phân và phép co nhị phân, hai phép toán này là cơ sở ban đầu cho nhiều thuật toán về hình thái khác.

Để đi vào tìm hiểu các phép toán hình thái, đầu tiên ta làm quen với một số khái niệm quan trọng thường được sử dụng trong bài viết.

Phần tử cấu trúc (Constructing elements): Đôi khi được gọi là một nhân (Kernel). Nhưng thuật ngữ này thường được sử dụng cho các đối tượng trong tích chập (Convolutions). Các phần tử cấu trúc thường được quy định theo một mẫu riêng dựa trên tọa độ của một số điểm có liên quan tới đối tượng nào đó. Ta có thể quan sát ví dụ từ hình minh họa phía dưới, đó là các mẫu đặc trưng cho các phần tử cấu trúc có kích thước khác nhau.



Hình 2.5. Một vài thí dụ về các phần tử cấu trúc

Ở Hình 2.5 có đưa ra hai kiểu phần tử cấu trúc đặc trưng có kích thước là 3x3 và 9x9, các vòng tròn đánh dấu gốc của phần tử cấu trúc ở hai kiểu này khác nhau, nhưng đều có đặc điểm chung là vòng đánh dấu tâm điểm luôn được đặt ở điểm có tọa độ ở trung tâm của phần tử cấu trúc.

Lưu ý: Ở mỗi tọa độ điểm trên phần tử cấu trúc đều có thể có một giá trị riêng. Một thí dụ đơn giản nhất về ứng dụng của phần tử cấu trúc, như sử dụng vào phép co với ảnh nhị phân. Ngoài ra còn có nhiều các phần tử cấu trúc phức tạp phục vụ cho các mục đích khác, chẳng hạn như sử dụng để tìm xương hoặc ứng dụng cho các phép toán hình thái trên ảnh xám.

Một điểm quan trọng nữa cần lưu ý là các phần tử cấu trúc cũng có thể có dạng hình chữ nhật, hình tròn… Trên ví dụ minh họa thì trong các phần tử cấu trúc có một số điểm bị bỏ trống, về bản chất thì những chỗ bỏ trống có thể coi như có giá trị là 0 hoặc một giá trị nào đó không thuộc phân tử cấu trúc, nếu đưa vào sẽ gây khó hiểu với nội dung cần đề cập, nên ở đây các phần tử đó được để trống.

Khi một phép toán hình thái được thực hiện thì các gốc của phần tử cấu trúc thường dịch chuyển lần lượt trên các điểm ảnh. Và khi đó giá trị của các điểm ảnh vừa được quét qua sẽ được so sánh với nhau, các kết quả thu được sau khi so sánh phụ thuộc vào phép toán hình thái đang được sử dụng.

2.4.1. *Dilation (Giản nở)*

Tập hợp B thường thì được coi như là một phần tử cấu trúc(structuring element) trong giãn nhị phân, cũng như trong các phép toán hình thái khác, tập hợp A là tập hợp các phần tử của hình ảnh gốc.

Với A và B là các tập hợp trong $\mathbb{Z}^2$, thì phép giãn nhị phân của A theo B ($A \oplus B$) được định nghĩa qua công thức sau:

$$A \oplus B = \left\{ z \mid \left(\hat{B} \right)_z \cap A \neq \varnothing \right\}$$

Như vậy phép giãn nhị phân của tập hợp A bởi phần tử cấu trúc B là tập hợp của tất cả các điểm z (z là tâm điểm của phần tử cấu trúc B trên tập hợp A) sao cho phản xạ của B_z giao với tập A tại ít nhất một điểm. Hay nói cách khác, phép giãn nhị phân là sự chồng chéo từ ít nhất một phần tử từ phản xạ của phần tử cấu trúc B với tập hợp A. Đồng thời các phần tử này phải là tập con của tập hợp A.

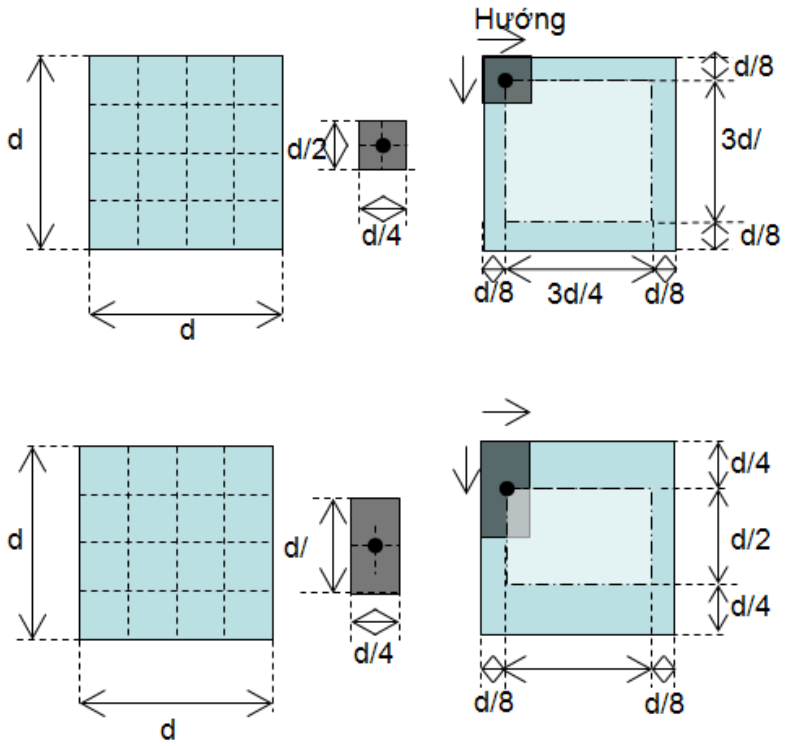
Công thức (1.12) có thể được viết lại :

$$A \oplus B = \left\{ z \mid \left[\left(\hat{B} \right)_z \cap A \right] \subseteq A \right\}$$

Phép giãn nhị phân của tập hợp A bởi tập hợp B là tồn tại các điểm w thuộc $\mathbb{Z}^2$ sao cho w là tổng của hai điểm tương ứng bất kỳ thuộc tập hợp A và tập hợp B, định nghĩa này được mô tả qua công thức :

$$A \oplus B = \left\{ w \in Z^2 \mid w = a + b, a \in A, b \in B \right\}$$

Tổng quát hơn, nếu A là một hình ảnh và B là phần tử cấu trúc có tâm điểm nằm trên hình ảnh A, khi đó phép giãn của hình ảnh A bởi phần tử cấu trúc B có thể được hiểu như quỹ tích của các điểm được phủ bởi phần tử cấu trúc B khi tâm điểm của B di chuyển trên cạnh của hình ảnh A.



Hình 2.6. Phép giãn nhị phân

Hình 2.6a gồm:

+ Tập hợp A có hai cạnh bên kích thước là d.

+ Phần tử cấu trúc vuông *B* kích thước *d/4*, trường hợp này thì phần tử cấu trúc *B* và tương phản của nó bằng nhau vì *B* có tập hợp các phần tử đối xứng nhau qua tâm điểm (dấu chấm đen ở giữa).

+ Cuối cùng là kết quả của phép giãn nhị phân giữa tập hợp *A* và phần tử cấu trúc *B*.

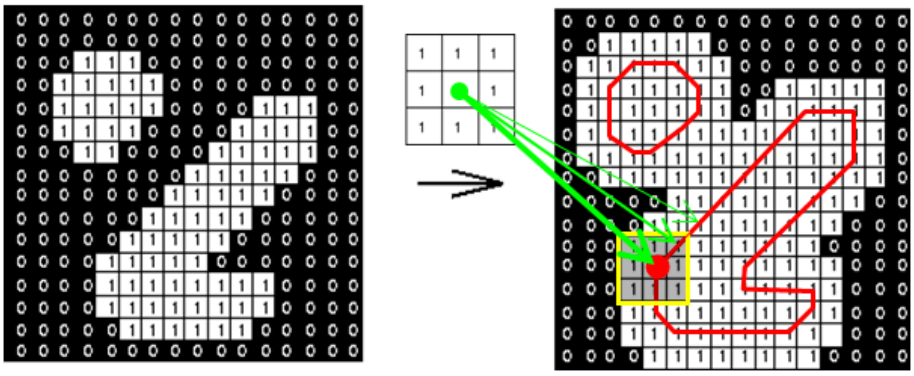
Hình 2.6b cũng gồm những thành phần tương tự nhưng với phần tử cấu trúc *B* là hình chữ nhật, nhưng cho ta một kết khác. Như vậy, mỗi kiểu phần tử cấu trúc khác nhau sẽ cho ta một kết quả khác nhau, sau khi thuật toán được thi hành.

Từ đó ta có công thức:

$$A \oplus B = \bigcup_{b \in B} A_b \,, \qquad (0.1)$$

Để thực tế hóa những lý thuyết đã nêu trên, ta coi những phần tử cấu trúc như một mẫu sẵn và dịch chuyển tịnh tiến trên bề mặt hình ảnh. Khi gốc của phần tử cấu trúc (chấm đen ở tâm điểm) khớp với điểm ảnh tại cạnh của hình ảnh thì các điểm ảnh tương ứng với với gốc này sẽ được đánh dấu và thay thế. Sau khi toàn bộ điểm ảnh đã được quét qua bởi phần tử cấu trúc thì, thao tác giãn hình ảnh được hoàn tất.

Một ví dụ về phép giãn trên một hình ảnh nhị phân sử dụng phần tử cấu trúc dạng ma trận vuông 3x3 như sau:



Hình 2.7. Quá trình quét của phần tử cấu trúc trên hình ảnh nhị phân

Ở ví dụ trên ta các điểm ảnh màu trắng mang giá trị là 1 là các điểm thuộc đối tượng đang cần quan tâm trên ảnh, và phần màu đen mang giá trị 0 là phần nằm ngoài đối tượng. Khi thuật toán được thi hành thì phần tử cấu trúc sẽ lần lượt quét qua các điểm ảnh ngoài cùng(Đi theo đường kẻ màu đỏ trên hình vẽ) của đối tượng

sau đó thay thế các điểm ảnh trên đối tượng này theo mẫu phần tử cấu trúc. Từ đó ta ứng dụng để nối các nét bi đứt gẫy của đoạn văn do quá trình xuống cấp, với khoảng cách lớn nhất của các nét bị đứt gãy tầm hai điểm ảnh.

2.4.2. *Erosion (Xói mòn)*

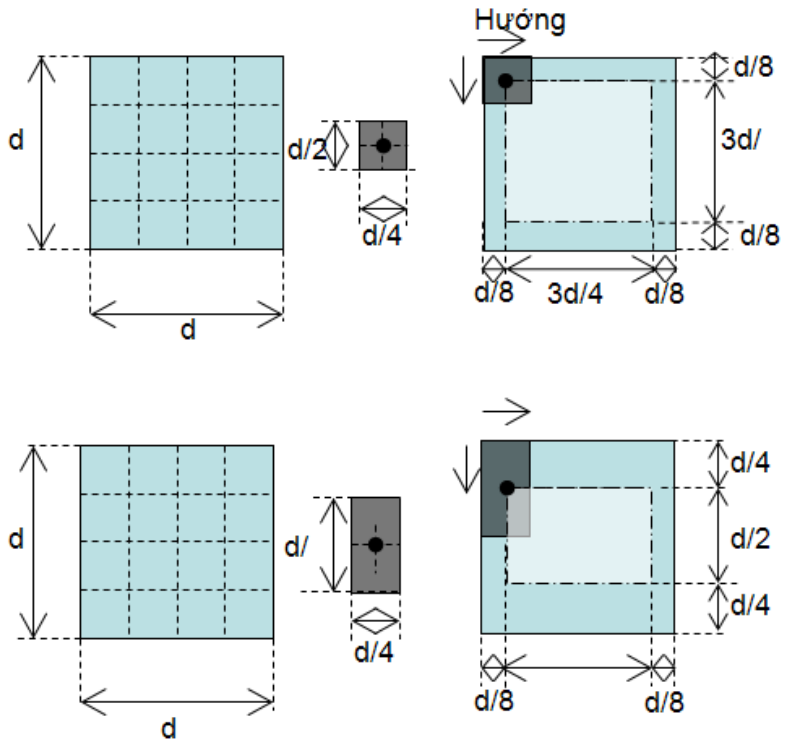
Ta cũng xét tập hợp A và tập hợp B (Phần tử cấu trúc) trong $\mathbb{Z}^2$, thì phép co nhị phân của tập hợp A bởi phần tử cấu trúc B được kí hiệu $A \ominus B$ và viết dưới dạng công thức như sau:

$$A \ominus B = \left\{ z \mid (B)_z \subseteq A \right\}, \qquad (0.2)$$

Với $$B_z = \left\{ b + z, b \in B \right\}, \qquad (0.3)$$

Phép co nhị phân của tập hợp A bởi phần tử cấu trúc B là tập hợp các điểm z (z nằm ở tâm điểm của phần tử cấu trúc B) sao cho B_z là tập con của A.

Xét hình vẽ sau:



Hình 2.8. Phép co nhị phân trên hai đối tượng

Hình 2.8a gồm:

+ Tập hợp *A* có hai cạnh bên kích thước là *d*.

+ Phần tử cấu trúc vuông *B* kích thước *d/4*(Dấu chấm đen ở giữa là tâm điểm).

+ Cuối cùng là kết quả của phép co nhị phân giữa tập hợp *A* và phần tử cấu trúc *B*.

Phần có màu nhạt hơn là kết quả sau khi thực hiện co hình ảnh bởi phần tử cấu trúc *B*. Hình 2.8b gồm những thành phần tương tự nhưng với phần tử cấu trúc *B* là hình chữ nhật, và cho ta một kết quả khác.

Vậy phép co nhị phân của ảnh *A* với phần tử cấu trúc *B* là quỹ tích các điểm được tạo ra bởi tâm điểm của phần tử cấu trúc *B* khi tịnh tiến trên hình ảnh *A*.

Từ đó ta có công thức:

$$A \ominus B = \bigcap_{b \in B} A_{-B}, \qquad (0.4)$$

Phép co nhị phân và giãn nhị phân có thể được với nhau qua phép bù và phép phản xạ của tập hợp, luận lý này sẽ được minh họa qua công thức sau:

$$(A \ominus B)^c = A^c \oplus \hat{B}, \qquad (0.5)$$

Ta chứng minh công thức trên là đúng:

Từ công thức co nhị phân ta có:

$$(A \ominus B)^c = \left\{ z \mid (B)_z \subseteq A \right\}^c, \qquad (0.6)$$

Nếu tập hợp B_z là tập con tập hợp *A* thì ta có, $B_z \cap A^c = \varnothing$, cho nên, trong trường hợp này ta sẽ có:

$$(A \ominus B)^c = \left\{ z \mid (B)_z \cap A^c = \varnothing \right\}^c, \qquad (0.7)$$

Vì phần bù của phép giãn nhị phân giữa tập hợp *A* và tập hợp *B* luôn thỏa mãn:

$(B_z \cap A^c = \varnothing)^c = B_z \cap A^c \neq \varnothing$

cho nên:

$(A \ominus B)^c = \left\{ z \mid (B)_z \cap A^c \neq \varnothing \right\}$

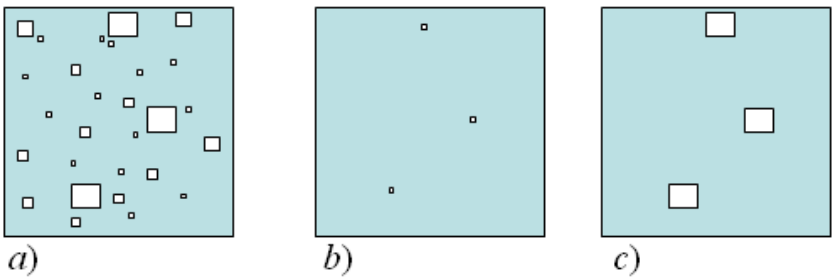
Mặt khác theo công thức 1.12 ta có:

$$A \oplus B = \left\{ z \mid \left(\hat{B} \right)_z \cap A \neq \varnothing \right\}$$

Suy ra: $(A \ominus B)^c = A^c \oplus \hat{B}$, □.+

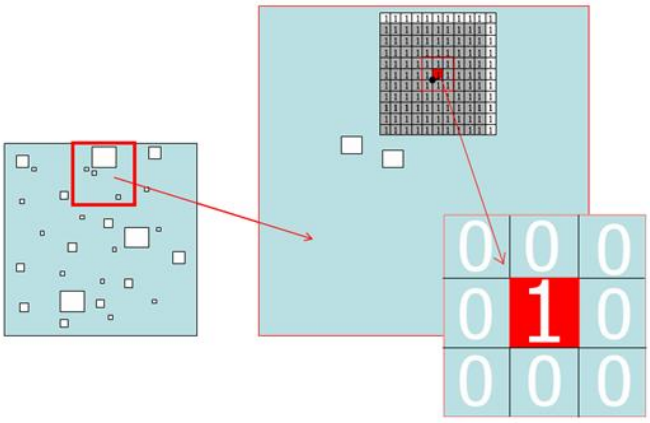
Như vậy, phần bù của phép co nhị phân giữa tập hợp *A* và Phần tử cấu trúc *B* là phép giãn nhị phân giữa phần bù của tập hợp *A* với phản xạ của phần tử cấu trúc *B*.

Một ứng dụng quan trọng của phép co nhị phân là dùng để loại trừ các chi tiết không cần thiết trên hình ảnh. Thí dụ, Trên một hình ảnh , ta có các đối tượng có cỡ tương ứng 1,4,6 và 11 điểm ảnh, Bây giờ nếu muốn loại trừ các đối tượng nhỏ không cần thiết trên ảnh, chỉ để lại các đối tượng có kích thước lớn, như trong hình vẽ đối tượng ta cần giữ lại là những đối tượng có kích thước 11 điểm ảnh. Ta sẽ sử dung phần tử cấu trúc có kích thước 10x10 điểm ảnh để thực hiện phép co nhị phân (Erosion). Kết quả sẽ chỉ còn lại 3 đối tượng có kích thước 1 điểm ảnh(Hình 2.9b). Sau đó để các đối tượng trở lại kích thước ban đầu ta sử dụng phép giãn nhị phân(Dilation) với phần tử cấu trúc có kích cỡ tương ứng (Hình 2.9c).



Hình 2.9. Quá trình lọc đối tượng sử dụng phép co nhị phân và phép giãn nhị phân

Quá trình thực hiện có thể được minh họa rõ ràng qua hình vẽ sau:



Hình 2.10. Ứng dụng của phép co ảnh dưới dạng số nhị phân

a) Hình ảnh ban đầu; b) Hình ảnh quá trình co nhị phân trên đối tượng với phần tử cấu trúc 10x10, phần tử được tô dậm màu sẽ có giá trị 1 sau quá trình co nhị phân; c) Phóng to đối tượng và giá trị của đối tượng sau quá trình co nhị phân với phần tử cấu trúc 10x10.

2.5. **Connected Component Labeling**

Một phương pháp đơn giản và hiệu quả của việc phân vùng ảnh nhị phân là việc kiểm tra sự kết nối của các điểm ảnh với các điểm lân cận và gán nhãn cho các kết nối đã được thiết lập.

Giả sử có một ảnh nhị phân được quét từ trái sáng phải từ trên xuống dưới. Điểm ảnh hiện tại là X, được gán nhãn nhận 1 trong 2 giá trị: 1 đối tượng (các bit 1) hay một hố (các bit 0) bởi việc kiểm tra sự kết nối của nó với các điểm lân cận A, B, C và D. Ví dụ như: nếu X – 1, thì nó được ấn định cho 1 (hoặc một số) đối tượng mà nó kết nối. Nếu có 2 hay nhiều đối tượng được kiểm tra, thì các đối tượng này được biểu thị cho các vùng tương đương và được kết hợp lại. Một đối tượng mới lại được ấn định khi có một sự chuyển từ các bit 0 sang một bit 1 riêng lẻ. Khi mà một điểm ảnh được gán nhãn, các đặc điểm của đối tượng được cập nhật. Ở cuối quá trình scan, các đặc điểm như trọng tâm, diện tích, chu vi được lưu lại cho mỗi vùng

của bit 1 được kết nối. Cụ thể thuật toán gồm hai bước duyệt qua từng phần tử của ảnh và một bước trung gian gọi là quá trình phân lớp tương thích. Nhiệm vụ của lần duyệt (*scan*) ảnh đầu tiên là gán nhãn tạm thời cho các điểm ảnh thuộc về đối tượng. Chúng ta sẽ minh họa cho tiến trình này bằng một ví dụ dưới đây.

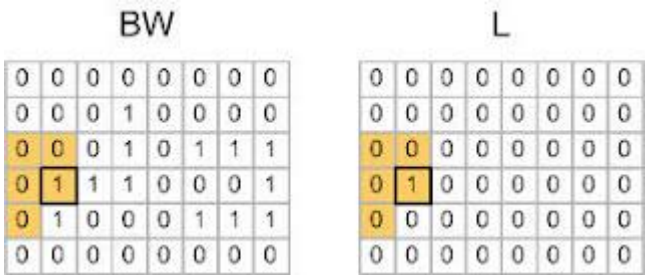
Dữ liệu đầu vào của chúng ta là một ảnh nhị phân có dạng ma trận (BW) như sau:

0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	1
0	1	1	1	0	0	0	1
0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0

Hình 2.11. Ảnh nhị phân BW

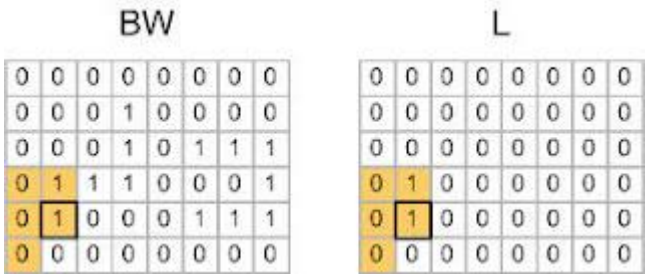
Và kết quả cần tìm là một ma trận nhãn (L) tương ứng cho từng điểm ảnh thuộc về foreground.

Chúng ta sẽ duyệt từng pixel của ảnh theo cột từ trái sang phải và từ trên xuống dưới (tiến trình này có thể được tiến hành theo hàng). Trong quá trình duyệt nếu gặp một điểm ảnh là foreground (giá trị = 1) thì ta sẽ tiến hành xem xét các điểm lân cận liền kề đã duyệt qua của nó. Hai hình dưới đây minh họa cho điểm ảnh foreground đầu tiên ta sẽ gặp trong quá trình duyệt ảnh và các pixel lân cận được xem xét.



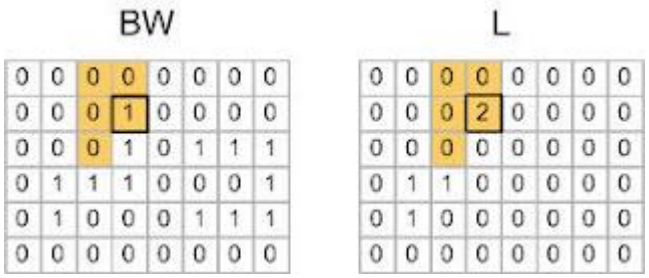
Hình 2.12. Duyệt đến pixel đầu tiên có giá trị 1

Do các điểm ảnh xem xét chưa được gán nhãn và giá trị nhãn hiện tại là 0 nên điểm ảnh foreground hiện tại sẽ được gán nhãn bằng một nhãn mới (1). Dưới đây sẽ là điểm ảnh foregournd kế tiếp chúng ta gặp trong trình tự duyệt điểm ảnh.



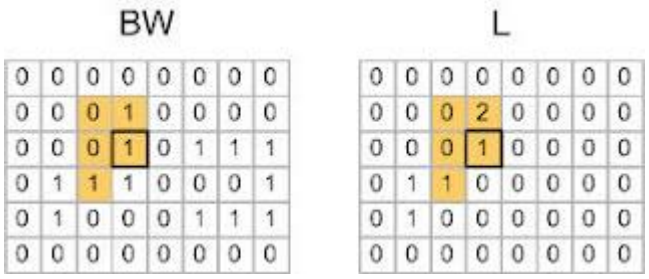
Hình 2.13. Duyệt đến pixel kế tiếp có giá trị 1

Để ý rằng một trong những điểm lân cận xem xét của nó đã được gán nhãn 1 vì vậy nó cũng sẽ có nhãn là 1. Cứ tiếp tục như thế cho tới điểm ảnh ở dòng 2 cột 4, không có điểm ảnh nào lân cận xem xét đã được gán nhãn vì vậy nó được gán bằng một nhãn mới (2).



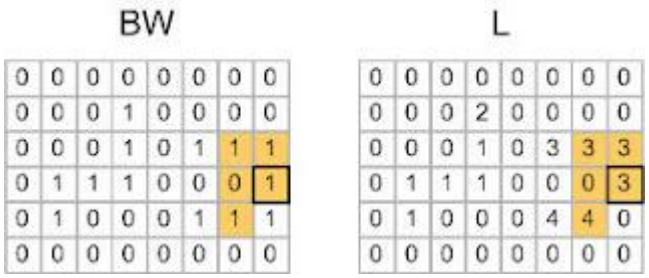
Hình 2.14. Gán nhãn 2

Tiếp tục quá trình scan đến điểm ảnh kế tiếp ở dòng 3 cột 4 chúng ta thấy một điều thú vị đã xãy ra.



Hình 2.15. Gán nhãn dòng 3 cột 4

Một trong những điểm lân cận thuộc diện xem xét của nó đã được gán nhãn là 1 tuy nhiên một điểm khác cũng được gán nhãn là 2, trong trường hợp này thuật toán thực hiện việc gán nhãn tùy ý (1 hoặc 2) nhưng đánh dấu lại rằng hai nhãn 1 và 2 cùng tham chiếu tới 1 đối tượng. Trường hợp này xãy ra một lần nữa khi chúng ta tới dòng 4 cột 8.



Hình 2.16. Gán nhãn dòng 4 cột 8

Vì vậy cặp nhãn 3 và 4 cũng là cặp tương thích được đánh dấu.

Khi lần scan thứ nhất được kết thúc chúng ta thu được một ma trận nhãn như sau

L

0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0
0	0	0	1	0	3	3	3
0	1	1	1	0	0	0	3
0	1	0	0	0	4	4	3
0	0	0	0	0	0	0	0

Hình 2.17. Kết quả lần quét thứ nhất

Kế đó quá trình phân lớp tương thích là quá trình quyết định những nhãn nào thực sự đánh dấu chung đối tượng nào. Kết quả nhận được từ xử lý này là 1--2 sẽ ánh xạ đến 1 và 3--4 sẽ ánh xạ tới 2. Lần duyệt ảnh thứ hai sẽ tái gán nhãn lại cho các pixel dựa trên bảng ánh xạ đã có từ quá trình phân lớp tương thích.

L

0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	2	2	2
0	1	1	1	0	0	0	2
0	1	0	0	0	2	2	2
0	0	0	0	0	0	0	0

Hình 2.18. Kết quả cuối cùng

CHƯƠNG 3. THEO DÕI ĐỐI TƯỢNG

3.1. **Giới thiệu**

Theo dõi đối tượng theo thời gian thực (*real-time object tracking*) là một công đoạn quan trọng trong rất nhiều ứng dụng thị giác máy tính (*computer vision applications*). Những hệ thống thuộc loại này có thể kể ra như là: hệ thống quan sát – theo dõi đối tượng, hệ thống giao tiếp với người dùng dựa vào tri giác (*perceptual user interface*), ngôi nhà thông minh, hệ thống nén video dựa vào đối tượng (*object-based video compression*) và những hệ thống thông minh hỗ trợ lái xe tự động...

Một trong những mục tiêu quan trong của theo dõi đối tượng là để "hiểu" được những chuyển động của đối tượng. "Hiểu" những thông tin về đối tượng như vị trí trong không gian, vận tốc chuyển động và những đặc trưng vật lý khác. Một hệ thống thông minh có thể thực hiện các bước xử lý khác ở cấp cao hơn như nhận dạng đối tượng, nhận dạng chuyển động, lý luận (*reasoning*) và tính toán trên những đặc trưng vật lý này.

Mặc dù đã được nghiên cứu nhiều năm, bài toán theo dõi đối tượng vẫn là một vấn đề nghiên cứu mở cho đến ngày nay. Mức khó khăn của vấn đề này phụ thuộc nhiều vào đối tượng được phát hiện và theo dõi như thế nào. Nếu như chỉ có một vài đặc trưng thị giác, chẳng hạn như màu sắc ... được dùng để biểu diễn đối tượng, thì khá dễ dàng để xác định tất cả các điểm ảnh cùng màu với đối tượng. Nhưng thực tế hoàn toàn khác, ví dụ như gương mặt của một người cụ thể sẽ có đầy đủ các chi tiết tri giác và thông tin nhiễu chảng hạn như các tư thế và sự chiếu sáng khác nhau, rất khó để phát hiện, nhận dạng và theo dõi. Hầu hết các khó khăn này nảy sinh từ khả năng biến động của các ảnh từ video bởi các đối tượng trong video thường là các đối tượng chuyển động. Khi một đối tượng chuyển dộng qua vùng quan sát của camera, hình ảnh về đối tượng có thể thay đổi rất nhiều. Sự thay đổi này đến từ 3 nguồn chính: sự thay đổi tư thế đối tượng hay sự biến dạng của đối

tượng, sự thay đổi về độ chiếu sáng, và sự che khuất (*occlusion*) một phần hay toàn bộ đối tượng.

Có rất nhiều phương pháp giải quyết bài toán này, ta có thể phân loại thành bốn các tiếp cận chính: tiếp cận dựa trên mô hình, tiếp cận dựa trên miền, tiếp cận dựa trên đường viền và tiếp cận dựa trên đặc trưng.

- **Tiếp cận dựa trên mô hình**

  Cách tiếp cận dựa trên mô hình bao gồm việc tạo mô hình học cấu trúc của đối tượng. Các ràng buộc trên việc mô hình được cho phép biến dạng như thế nào có thể kết hợp vào quá trình so khớp. Vấn đề với cách tiếp cận này là quá trình khởi tạo tự động gặp nhiều khó khăn và chi phí tính toán cao do độ phức tạp của mô hình.

- **Tiếp cận dựa trên miền**

  Cách tiếp cận dựa trên miền bao gồm việc kết hợp một miền với mỗi đối tượng đang theo dõi. Miền được theo dõi qua thời gian bằng phép đo độ tương tự. Lợi ích của cách tiếp cận này là khởi tạo khá dễ dàng, chỉ có vị trí và kích thước của cửa sổ cần được định nghĩa. Các tham số của thuật toán cũng có ý nghĩa vật lý, dễ dạng khái niệm hóa.

- **Tiếp cận dựa trên đường viền**

  Cách tiếp cận dựa trên đường viền bao gồm tìm được viền bao quanh của một đối tượng và sau đó cố gắng làm khớp đường viền với các đối tượng trong các frame sau. Nơi khớp nhất sẽ là đường viền hiện tại, mô hình sẽ cập nhật đường viền hiện tại để phản ánh hình dáng của đối tượng trong frame hiện tại. Quá trình này được lặp lại với mô hình đường viền được cập nhật. Ưu điểm của các tiếp cận này là khả năng xử lý hiệu quả sự che khuất một phần (*partial occlusion*). Tuy nhiên vấn đề với cách tiếp cận này là mô hình yêu cầu sự khởi tạo chính xác và điều này thì khó để thực hiện tự động.

- **Tiếp cận dựa trên đặc trưng**

Khác với các cách tiếp cận trên đều theo dõi toàn bộ đối tượng, cách tiếp cận dựa trên đặc trưng chỉ theo vết một tập các đặc trưng của đối tượng. Ví dụ như chỉ theo dõi các điểm ở góc của đối tượng, vị trí của đối tượng trong frame sau sẽ được tìm thấy bằng cách tìm các điểm góc mà khớp với các điểm của mô hình nhất. Ưu điểm của cách tiếp cận này là xử lý được sự che khuất một phần. Khi đối tượng bị che khuất, một số các đặc trưng vẫn còn thấy được và có thể được dùng trong quá trình theo dõi. Khuyết điểm của phương pháp này là chất lượng theo dõi phụ thuộc nhiều vào việc chọn các đặc trưng. Các đặc trưng phải được chọn sao cho chúng cung cấp sự nhận diện duy nhất cho đối tượng, việc này không phải là một nhiệm vụ dễ.

3.2. **Các phương pháp theo dõi đối tượng**

Theo dõi đối tượng là giám sát các thay đổi theo không gian và thời gian của đối tựng trong suốt chuỗi video, bao gồm sự hiện diện, vị trí, kích thước, hình dáng...của đối tượng.

Một số phương pháp theo dõi đối tượng thông thường:

- So khớp mẫu (*Template matching*)
- Tiếp cận Bayesian (lọc *Kalman*, lọc *Particle*)
- CAMshift (***C**ontinuously **A**daptive **M**ean **shift***)

3.2.1. *So khớp mẫu*

Thủ tục trong so khớp mẫu như sau: một miền nhỏ chung quanh điểm cần được theo dõi sẽ được dùng làm mẫu. Mẫu này sau đó dùng để tìm ra frame ảnh kế tiếp bằng cách sử dụng các kỹ thuật tương quan [11]. Vị trí với kết quả cao nhất sẽ là so khớp tốt nhất giữa mẫu và ảnh.

Bằng cách cập nhật các mẫu theo chuỗi ảnh, các biến dạng lớn cũng có thể được theo dõi [12]. Sử dụng một trong 3 luật cập nhật cơ bản như sau:

1. Nếu có một sự thay đổi lớn giữa vị trí mẫu ban đầu và vị trí mới thì vị trí mẫu mới được chọn. Trong trường hợp này, các mẫu được hoán đổi hoàn toàn bởi hình dạng mới của chúng.
2. Nếu có các thay đổi nhỏ giữa vị trí ban đầu và mới của mẫu, một phiên bản trung bình giữa mẫu mới và cũ sẽ được tính và được cập nhật như mẫu mới. Bằng cách này, các đạo hàm nhỏ liên quan đến nhiễu sẽ là trung bình, do đó gia tăng được khả năng theo dõi tránh nhiễu.
3. Nếu chỉ có các thay đổi quá nhỏ giữa các vị trí ban đầu và mới, thì mẫu cũ sẽ được sử dụng. Điều này rất quan trọng cho các đối tượng tịnh tiến bởi các lượng nhỏ hơn một pixel: nếu như ta cập nhật lại thì sẽ bị mất các thông tin dịch pixel nhỏ.

Ưu điểm: không chịu ảnh hưởng bởi nhiễu và hiệu ứng chiếu sáng, theo vết được các đối tượng biến dạng.

Khuyết điểm: độ phức tạp tính toán cao, chất lượng so khớp phụ thuộc vào chi tiết và độ chính xác của mẫu đối tượng.

3.2.2. *Tiếp cận Bayesian*

Tiếp cận theo phương pháp *Bayesian* [13] là phương pháp dựa trên xác suất, sử dụng các phương trình dự đoán (*prediction*) để dự đoán trạng thái của đối tượng và phương trình cập nhật (*updation*) - hay còn gọi là hàm trơn (*smoothing*) - để hiệu chỉnh lại các dự đoán trước đó về trạng thái của đối tượng dựa trên những tri thức thu thập được từ các quan sát (*observation*) trên đối tượng.

Mục tiêu của phương pháp Bayesian là ước lượng trạng thái X_k dựa trên $Z_{1:k}$. Ta ký hiệu, giá trị ước lượng X_k dựa trên $Z_{1:k}$ là $\hat{X}_{k|k}$. Với X_k là trạng thái tại thời điểm k, $Z_{1:k}$ là chuỗi các quan sát từ thời điểm 1 đến k.

Dễ thấy, giá trị ước lượng trên là một hàm phụ thuộc các quan sát:

$$\hat{X}_{k|k} = g(Z_{1:k})$$

Lỗi ước lượng: $\tilde{X}_{k|k} = X_k - \hat{X}_{k|k}$.

Hàm chi phí như là bình phương cho lỗi ước lượng:

$$C(\hat{X}_{k|k}, X_k) = (X_k - \hat{X}_{k|k})^T (X_k - \hat{X}_{k|k})$$

Mục tiêu của ước lượng Bayesian là tìm dạng hàm g(.) sao cho giảm thiểu chi phí kỳ vọng:

$$E_{X_{0:k}, Z_{1:k}}[C(\hat{X}_{k|k}, X_k)] = E_{X_{0:k}, Z_{1:k}}[C(g(Z_{1:k}), X_k)]$$

Ta có thể giảm thiểu chi phí kỳ vọng toàn thể bằng cách chọn g(.) sao cho giảm thiểu kỳ vọng điều kiện cho mỗi giá trị $Z_{1:k}$. Chi phí kỳ vọng này có thể được viết như sau:

$$E_{Z_{1:k}}\left[E_{X_{0:k}|Z_{1:k}=Z_{1:k}}[C(g(Z_{1:k}), X_k)]\right]$$

Lúc này, hàm g sẽ tối ưu khi $g(Z_{1:k}) = E_{X_k|Z_{1:k}}[X_k]$.

Đó là nguồn gốc cho việc tính mật độ xác suất sau (*posterior density*) $p(x_k|z_{1:k})$ trong phương pháp lọc Bayesian.

Phương pháp lọc Bayesian được thực hiện đệ quy đối với $p(x_k|z_{1:k})$ qua hai bước:

- Dự đoán:

$$p(x_k|z_{1:k-1}) = \int p(x_k|z_{k-1}) p(x_{k-1}|z_{1:k-1}) dx_{k-1}$$

- Cập nhật:

$$p(x_k|z_{1:k-1}) = \frac{p(z_k|x_k) p(x_k|z_{1:k-1})}{\int p(z_k|x_k) p(x_k|z_{1:k-1}) dx_k}$$

Một số phương pháp theo vết thông dụng dựa trên ước lượng Bayesian là:

- Lọc Kalman

- Lọc HMM
- Lọc Particle

Lọc Kalman đã được biết như là một phương pháp cổ điển, nổi tiếng được phát minh từ năm 1960 bởi *R.E.Kalman*. Nó là một thuật toán theo vết tối ưu nhất trong trường hợp hệ là tuyến tính và nhiễu có phân phối *Gauss*. *Extended Kalman* và *Unscented Kalman* tuy giải quyết được trường hợp phi tuyến và không phải nhiễu *Gauss* nhưng cũng chỉ giải quyết tốt bài toán trong trường hợp phương trình biến đổi có bậc 2.

Lọc HMM (*Hidden Markov Model*) là thuật toán lọc tối ưu rời rạc. Nó không đòi hỏi những yêu cầu về tuyến tính cũng như phân phối của hệ. Tuy nhiên, lọc HMM chỉ có thể được áp dụng trong trường hợp hệ có trạng thái hữu hạn và xác định.

Lọc Particle được phát minh nhằm giải quyết tốt hơn bài toán lọc, đặc biệt là nó có thể khắc phục được mọi nhược điểm của lọc Kalman và cũng không yêu cầu hệ phải có tập trạng thái hữu hạn.

3.2.3. *Theo dõi đối tượng dùng thuật toán CAMshift*

CAMshift (***C**ontinuously **A**daptive **M**ean **shift***) là thuật toán theo dõi đối tượng được phát triển từ thuật toán *Mean shift* để đạt được mục đích theo dõi các đối tượng có phân phối màu thay đổi.

3.2.3.1. Mean shift

Dorin Comaniciu đã giới thiệu phương pháp theo dõi đối tượng màu Mean shift [Comaniciu2003]. Đây là một phương pháp theo dõi tối ưu hóa tối thiểu cục bộ. Mỗi vị trí x_i trong vùng của một đối tượng tiềm năng sẽ tương ứng với một trọng số w_i.

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \delta(b(x_i) - u)$$

Với $b(x_i)$ là giá trị màu tại x_i và $\hat{q}_u$ là giá trị tại màu u của mô hình đích, và $\hat{p}_u(\hat{y}_0)$ là giá trị tại màu u của mô hình đối tượng tiềm năng.

Vị trí mới của đối tượng là vị trí mà khoảng cách nhỏ nhất giữa mô hình đích và mô hình ứng viên và được tính bởi:

$$\hat{y}_1 = \frac{\sum_i x_i w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}{\sum_i w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}$$

Với $g(x) = -k'(x)$ và $k(x)$ là một hàm nhân (*kernel function*).

Quá trình được lặp lại cho đến khi không có sự thay đổi trong vị trí mới.

Thuật toán Mean shift được sử dụng với mục tiêu nhanh chóng tìm ra trọng tâm của khối trong của sổ tìm kiếm dựa vào tính toán moment của các điểm ảnh trong hệ tọa độ gốc với trình tự sau:

- Bước 1: Khởi tạo kích thước cửa sổ tìm kiếm.
- Bước 2: Khởi tạo vị trí của cửa sổ tìm kiếm.
- Bước 3: Tính trọng tâm của khối nằm trong cửa sổ tìm kiếm.
- Bước 4: Đưa điểm giữa của cửa sổ tìm kiếm đến vị trí trọng tâm của khối vừa tính ở bước trên.
- Bước 5: Lặp lại bước 3 và 4 cho đến khi hội tụ (hay cho đến khi trọng tâm của khối nhỏ hơn 1 giá trị ngưỡng đã xác định trước).

Công thức tính các giá trị moment như sau:

Zeroth moment:

$$M_{00} = \sum_x \sum_y I(x, y)$$

First moment cho x và y:

$$M_{10} = \sum_x \sum_y xI(x,y)$$

$$M_{01} = \sum_x \sum_y yI(x,y)$$

Sau đó tính trọng tâm (*mean*) của khối trong cửa sổ tìm kiếm.

$$x_c = \frac{M_{10}}{M_{00}}; y_c = \frac{M_{01}}{M_{00}};$$

Với $I(x,y)$ là giá trị xác suất phân bố màu của điểm ảnh ở tọa độ (x,y).

3.2.3.2. CAMshift (Continuously Adaptive Mean shift)

Sự cải tiến của CAMshift so với Mean shift được minh họa ở Bảng dưới.

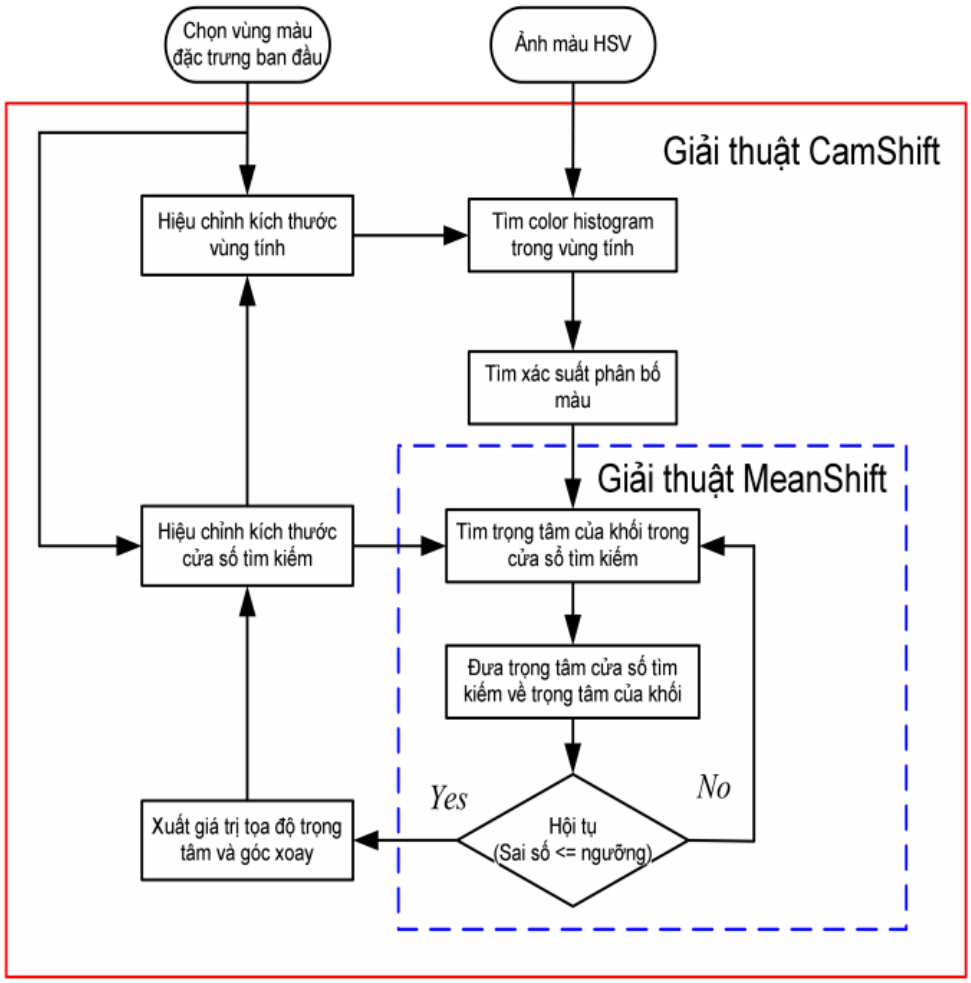
Bảng 3-1. So sánh CAMshift và Mean shift

Thông số	MeanShift	Camshift
Trạng thái đối tượng	Tĩnh, chỉ tịnh tiến	Động, có thể xoay
Kích thước đôi tượng	Cố định	Thay đổi
Dữ liệu phân tích	Phân bố màu tĩnh, chỉ cập nhật khi có sự thay đổi lớn	Phân bố màu liên tục cập nhật liên tục frame by frame
Kích thước cửa sổ tìm kiếm	Cố định	Thay đổi
Kích thước vùng tính	Cố định	Thay đổi

Tìm vector di chuyển	Dựa trên khoảng cách trọng tâm thay đổi	Dựa trên moment cấp 2 để tìm và dự đoán vector dị chuyển

Không giống như thuật toán Mean shift, chỉ hoạt động với điều kiện các phân phối màu không thay đổi. Thuật toán CAMshift được thiết kế để có thể theo dõi các đối tượng có phân phối màu thay đổi. CAMshift được sử dụng với mục tiêu hiệu chỉnh giá trị cửa sổ tìm kiếm và vùng tính 1 cách hợp lí nhất để đảm bảo thời gian tính toán nhanh và theo dõi chính xác.

Trình tự thực hiện thuật toán theo dõi đối tượng động được minh họa như Hình.



Hình 3.1. Thuật toán CAMshift theo dõi đối tượng động

- Bước 1: Khởi tạo vị trí của cửa sổ tìm kiếm.
- Bước 2: Thực hiện Mean shift như đã trình bày ở phần trên (một lần hoặc nhiều lần). Lưu lại giá trị zeroth moment.
- Bước 3: Điều chỉnh lại kích thước cửa sổ tìm kiếm dựa vào zeroth moment tìm được ở bước 2.
- Bước 4: Lặp lại bước 2 và 3 cho đến khi hội tụ (vị trí trọng tâm nhỏ hơn một giá trị ngưỡng đã xác đinh trước).

Khi theo dõi một đối tượng có màu sắc, CAMshift hoạt động trên hình ảnh phân phối xác suất thu được từ *histogram* của đối tượng. CAMshift tính trọng tâm của phân phối màu 2 chiều bên trong cửa sổ tìm kiếm, điều chỉnh lại điểm giữa của cửa sổ, sau đó tính kích thước cho cửa sổ tìm kiếm tiếp theo. Do đó, chúng ta không

cần tính phân phối màu toàn bộ hình ảnh mà chỉ cần tính bên trong 1 cửa sổ hơi lớn hơn kích thước của cửa sổ CAMshift hiện tại.

3.3. **Kết luận**

Mỗi phương pháp theo dõi đối tượng có điểm mạnh và điểm yếu riêng của nó. Tuy nhiên, đối với đề tài này đối tượng cần theo dõi là bàn tay, một đối tượng có màu đồng nhất. Và việc nhận dạng đối tượng được thực hiện bằng bộ phát hiện cử chỉ, vì thế, trong các phương pháp thì phương pháp theo dõi đối tượng bằng thuật toán CAMshift tỏ ra có ưu thế vì sự đơn giản và tính hiệu quả của nó.

CHƯƠNG 4. TỔNG QUAN VỀ KIT DSP TMS320DM642

Hiện nay có nhiều nền tảng để lập trình viên xây dựng hệ thống xử lý ảnh, mà nổi bật nhất là sử dụng bộ xử lý tín hiệu số - *Digital Signal Processor* (DSP) hoặc dùng các vi mạch mảng phần tử logic khả trình (FPGA), FPGA là vi mạch thuộc họ vi mạch tích hợp chuyên dụng (ASIC) lập trình được, sử dụng các ngôn ngữ đặc tả phần cứng để thiết kế những cấu trúc được tối ưu hóa cho những ứng dụng cụ thể. Ưu việt của FPGA là không thể bàn cãi, đặc biệt là khả năng xử lý nhiều tập lệnh cùng lúc cho tốc độ cao, và khả năng tiêu thụ ít điện năng hơn chip DSP. Tuy nhiên việc thực thi thuật toán trên kit DSP cũng có những đặc điểm nổi bật riêng:

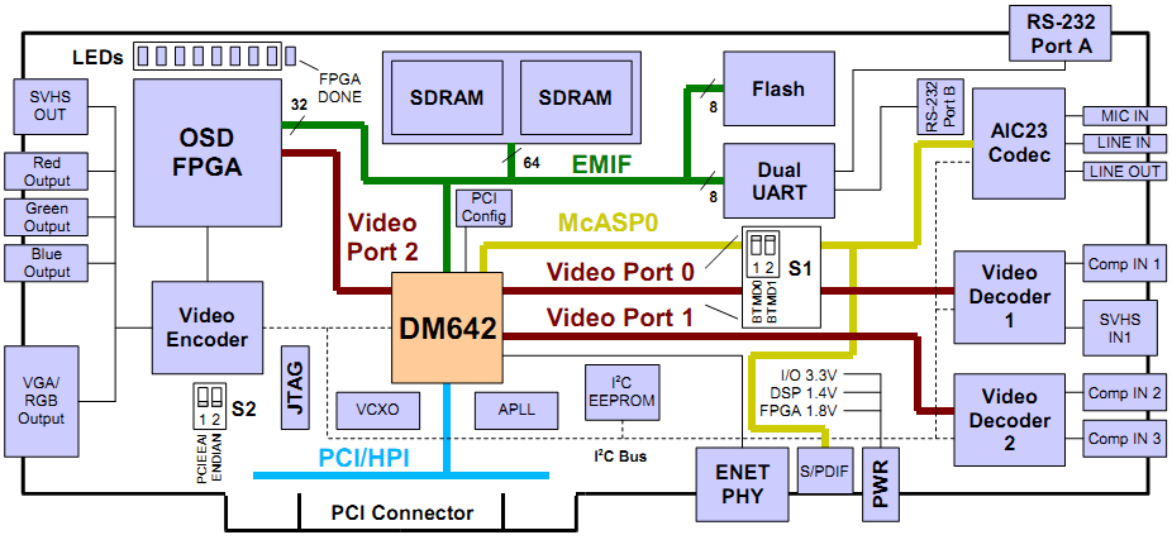
- DSP có khả năng thực hiện đa tác vụ từ điều khiển đến xử lý tín hiệu, với giá thành rẻ hơn so với FPGA.
- Để đạt được hiệu suất tối đa cho hệ thống sử dụng FPGA cần nhiều thời gian và kiến thức để tối ưu, trong khi đó tốc độ xử lý của hệ thống dựa trên DSP chỉ phụ thuộc chủ yếu vào xung nhịp của chip, do đó có thể đạt được hiệu suất cao hơn trong thời gian ngắn.
- DSP sử dụng ngôn ngữ lập trình C, Assembly tương đối phổ biến, không đòi hỏi hiểu biết ngôn ngữ mô tả phần cứng VHDL hay Verilog như khi sử dụng FPGA, khi cần thay đổi, lập trình lại, chip DSP cũng tỏ ra mềm dẻo hơn do chỉ cần chỉnh sửa code, trong khi đó với FPGA gặp khó khăn hơn do phải tái cấu trúc lại các cổng logic.

Dựa trên những phân tích trên, cùng với mục tiêu của đề tài là tập trung vào nghiên cứu thuật toán nhận dạng cử chỉ, không đòi hỏi tối ưu điện năng tiêu thụ. Tác giả chọn giải pháp xây dựng đề tài trên chip DSP TMS320DM642 của Texas Instrument, cụ thể là kit EVMDM642 của Digital Spectrum Incooperated.

4.1. **Board EVMDM642**

TMS320DM642 EVM thuộc dòng sản phẩm low-cost standalone development platform cho phép người sử dụng thử nghiệm và phát triển ứng dụng

cho dòng chip TI C64x DSP. Nó cũng là thiết kế phần cứng mang tính tham khảo dành cho TMS320DM642 DSP.



Hình 4.1. Block Diagram EVMDM642

TMS320DM642 EVM được trang bị tương đối đầy đủ với các thiết bị phổ biến để có thể xây dựng và phát triển các ứng dụng khác nhau. Các thành phần bao gồm:

- TI TMS320DM642 DSP hoạt động ở 720MHz.
- Khe cắm Standalone/Standard PCI.
- 3 video port với 2 decoder và 1 encoder onboard.
- 32Mb SDRAM.
- On-Screen-Display FPGA.
- 4Mb Flash memory.
- AIC23 stereo codec.
- Giao tiếp Ethernet.
- Cấu hình phần mềm thông qua các thanh ghi trong FPGA.
- 8 User Led.
- Nguồn 5V.
- Các đế mở rộng cho Daughter card.
- Dual UART với RS-232.

4.1.1. *Tổng quan các chức năng*

DSP trên EVMDM642 giao tiếp với thiết bị bên ngoài thông qua đường 64-bit EMIF hoặc một trong ba đường 8/16-bit video port. SDRAM, Flash, FPGA, và UART được kết nối vào các đường bus riêng biệt. EMIF cũng được kết nối với các đế cắm mở rộng dùng cho Daughter card gắn vào board.

Các video encoder và decoder trên board kết nối với các video port và đế cắm mở rộng, EVM có 2 decoder và 1 encoder. Chức năng on-screen-display được thực hiện trong 1 FPGA gắn ngoài, FPGA này được đặt ở giữa output video port và video decoder.

AIC23 codec trên board cho phép DSP truyền nhận tín hiệu audio analog. I^2C bus được dùng cho việc điều khiển giao tiếp codec và McASP được dùng cho đường dữ liệu. Analog giao tiếp bên ngoài thông qua 3 jack audio 3.5mm tương ứng với microphone input, line in, line out. Codec có thể chọn microphone hoặc line in làm tín hiệu input. Tín hiệu analog output được truyền đến line out với 1 độ khuếch đại cố định. McASP có thể gửi tín hiệu analog ra bằng đường khác đến các đế cắm mở rộng.

FPGA được sử dụng như cầu nối để gắn kết các thành phần trên board lại với nhau. FPGA có những thanh ghi cấu hình mà người dùng có thể đọc và ghi dữ liệu để cấu hình hoạt động cho board.

EVM được thêm vào 8 led, các led này được điều khiển bằng cách đọc ghi các thanh ghi trên FPGA.

Bộ cấp nguồn 5V cung cấp điện thế hoạt động cho board trong các ứng dụng chạy độc lập, với việc sử dụng PCI card cắm trên PC thì PC bus sẽ đóng vai trò là nguồn cung cấp. Trên board có những bộ điều chỉnh điện thế để cung cấp điện thế 1.4V cho DSP và 3.3V cho I/O. Board được giữ ở trạng thái reset cho đến khi các bộ điều chỉnh điện thế được thiết lập đúng như trong bảng đặt tả hoạt động. EVM

cũng có một bộ điều chỉnh LDO để cung cấp 1.8V cho FPGA và 3.3V cho encode và decoder.

Code Composer Studio giao tiếp với EVM thông qua emulator gắn ngoài với giao tiếp JTAG 14/60 chân.

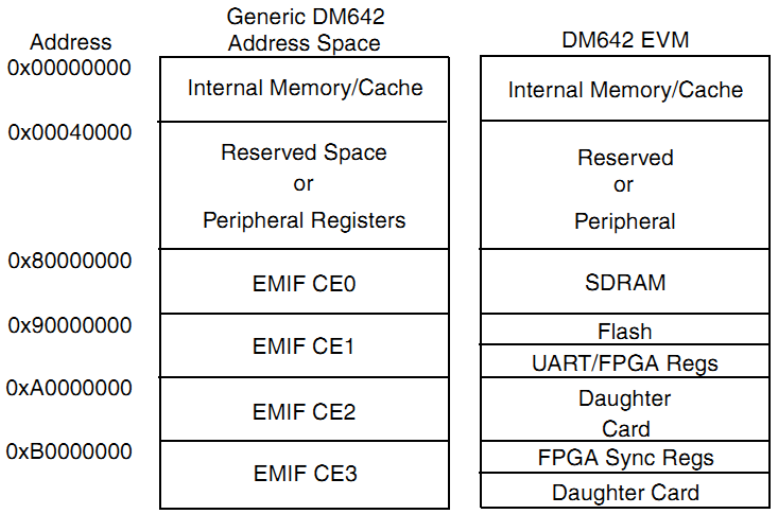
EVM được thiết kế để làm việc với môi trường phát triển Code Composer Studio của TI. Code Composer Studio giao tiếp với board thông qua JTAG emulator gắn ngoài.

4.1.2. *Memory map*

Dòng C64xx DSP có không gian địa chỉ lớn và độ rộng địa chỉ là 32-bit. Code chương trình và dữ liệu có thể được đặt bất kỳ đâu trong không gian địa chỉ đó.

Memory map cho ta thấy vùng địa chỉ của DM642 với thông tin được đặc tả cụ thể cho mỗi vùng địa chỉ. Vị trí của vùng nhớ cũng có thể được map lại trên phần mềm.

EMIF có 4 vùng được định địa chỉ phân biệt được gọi là vùng Chip Enable (CE 0 đến CE 3). SDRAM chiếm giữ vùng CE 0 trong khi Flash, UART, và FPGA dược map vào CE 1. Các Daughter card sử dụng vùng CE 2 và CE 3. CE 3 được cấu hình cho hoạt động đồng bộ của OSD FPGA và những thanh ghi đồng bộ khác trong FPGA.



Hình 4.2. Memory map DM642 EVM

4.1.3. *Thiết lập cho các switch cấu hình*

EVM có 2 switch cấu hình cho phép người dùng điều khiển trạng thái hoạt động của DSP. Những switch cấu hình này là S1 và S2 trên EVM.

Bảng 4-1. Thiết lập của Switch cấu hình S1

S1-2	S1-1	Configuration Description
Off	Off	No Boot
Off	On	HPI/PCI Boot
On	Off	Reserved
On *	On *	EMIF boot from 8-bit Flash

S1 cấu hình cho boot mode, cấu hình này sẽ được thực thi trong quá trình DSP khởi động. Ở chế độ mặc định, các Switch được cấu hình để DSP boot từ EMIF (tức boot từ Flash) ở chế độ Little Endian.

Switch cấu hình S2 điều khiển đặc tính endian của DSP và tín hiệu cho phép PCI ROM. Bảng bên dưới sẽ mô tả thiết lập cho switch S2.

Bảng 4-2. Thiết lập của Switch cấu hình S2-1

S2-1	Configuration Description
Off	PCI EEPROM Disabled
On *	PCI EEPROM Enabled

Bảng 4-3. Thiết lập của Switch cấu hình S2-2

S2-1	Configuration Description
Off *	Little Endian Mode
On	Big Endian Mode

4.1.4. *Nguồn cung cấp*

EVM hoạt động ở 5V được cấp từ nguồn chính tại J5 hoặc từ khe cắm PCI. Nguồn 5V được chuyển đổi thành 1.4V và 3.3V bằng bộ điều chỉnh điện thế của TI. Nguồn 1.4V được sử dụng cho DSP trong khi nguồn 3.3V được sử dụng cho các I/O của DSP cũng như tất cả các chip trên board. Bộ điều chỉnh LDO được sử dụng để tạo ra điện thế hoạt động cho FPGA và video I/O.

Có 5 điểm kiểm tra nguồn trong EVM: TP 4, TP 8, TP 13, TP 15, và TP 16. Các vị trí này giúp thuận tiện cho việc kiểm tra nguồn cung cấp của EVM.

Bảng 4-4. Các điểm kiểm tra nguồn

Test Point	Voltage	Voltage Use
TP4	+1.4 V	DSP Core
TP8	+3.3 V	DSP I/O and logic
TP13	+1.8 V	FPGA
TP15	+3.3 V	Video encoder
TP16	+3.3 V	Video decoder

4.1.5. *Giao tiếp EMIF*

DM642 được gắn với giao tiếp bộ nhớ có độ rộng 64-bit. 4 chip enable chia nhơ không gian địa chỉ và cho phép truy cập đồng bộ và bất đồng bộ với độ rộng bit 8, 16, 32, 64. EVM sử dụng CE 0, CE 1, CE 2, CE 3. CE 0 được nối với SDRAM bus 64-bit. CE 1 được dùng cho 8-bit Flash, UART, và FPGA. CE 3 được dùng cho chức năng đồng bộ. Cả CE 2 và CE 3 được nối với các đế cắm mở rộng cho các Daughter card.

Bảng 4-5. EMIF Interfaces

Chip Select	Function
CE0	SDRAM bus
CE1	8 bit Flash, UART, FPGA functions
CE2	Daughter Card Interface
CE3	FPGA Sync Registers
	Daughter Card Interface

4.1.5.1. Giao tiếp bộ nhớ SDRAM

EVM giao tiếp với đường bus SDRAM rộng 64-bit trong không gian CE 0. Đây là không gian 32MB SDRAM được dùng để lưu trữ chương trình, dữ liệu và video. Bus sử dụng một PLL ngoài giúp SDRAM hoạt động ở 133MHz nhằm tối ưu hiệu suất. Quá trình làm tươi cho SDRAM được thực hiện tự động bởi DM642.

PLL được sử dụng cho EMIF là chip ICS512. Clock vào PLL là 25MHz.

Bảng 4-6. Tần số của PLL

S1 Input	S2 Input	Multiplier	Output Frequency
0	0	4x	100 Mhz
0	Open	5.33x	133.25 Mhz *
0	1	5x	125 Mhz
Open	0	2.5x	62.5 Mhz
Open	Open	2x	50 Mhz
Open	1	3.33x	83.25 Mhz
1	0	6x	150 Mhz
1	Open	3x	75 Mhz
1	1	8x	200 Mhz

DM642 có thể cấu hình làm nguồn clock của EMIF, mặc định trên EVM là ECLKIN. Tuy nhiên, cũng có thể điều khiển clock EMIF chia từ clock CPU. Việc cấu hình này thông qua chân ECLKINSEL0 và ECLKINSEL1 được chia sẻ với địa chỉ của EMIF ở chân EA19 và EA20. Cấu hình có hiệu lực khi reset.

Bảng 4-7. Cấu hình clock trong EMIF

ECLKINSEL0	ECLKINSEL1	Mode
0	0	ECLKIN *
0	1	CPUCLK/4
1	0	CPUCLK/6
1	1	ECLKIN

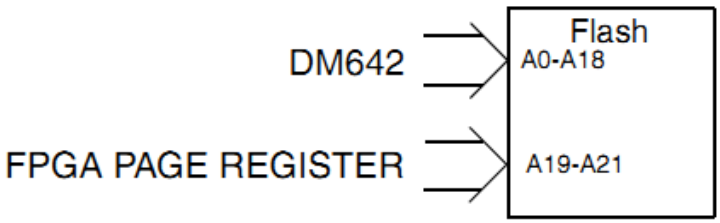
4.1.5.2. Giao tiếp bộ nhớ Flash

DM642 có 4MB bộ nhớ Flash được map tại vị trí thấp nhất trong không gian CE 1. Bộ nhớ Flash này được dùng làm vùng boot load chính và lưu giữ thông tin cấu hình của FPGA. Trên DM642 EVM, không gian CE 1 được cấu hình 8-bit và Flash cũng có độ rộng 8-bit. Không gian địa chỉ khả dụng trong CE 1 nhỏ hơn kích thước của Flash nên FPGA được sử dụng để tạo 3-bit địa chỉ thêm vào. Đường địa

chỉ được thêm vào được định địa chỉ thông qua thanh ghi Flash base trong FPGA và mặc định là 000B khi reset.

Bảng 4-8. Giao tiếp bộ nhớ Flash

Address Range	Page Number	Contents
0x8000 0000 0x8007 FFFF	Page 0	000B
	Page 1	001B
	Page 2	010B
	Page 3	011B
	Page 4	100B
	Page 5	101B
	Page 6	110B
	Page 7	111B



Hình 4.3. Giao tiếp bộ nhớ Flash

4.1.5.3. Giao tiếp UART

Dual UART (TLC16C752) chiếm vùng nhớ được map vào nửa trên của CE 1 cùng với những thanh ghi bất đồng bộ của FPGA. Mỗi UART (A và B) chiếm 8 ô nhớ. CE 1 được cấu hình để truy xuất 8-bit trên DM642 EVM.

Bảng 4-9. Địa chỉ UART

UART	Address
A	0x9008 0000 - 0x9008 0007
B	0x9008 0008 - 0x9008 000F

Giao tiếp UART với đường điều khiển RS-232. UART A đặt ở đế DB-9, J11, và UART B được kết nối với double row header trên board, J12.

4.1.6. *Giao tiếp video port/McASP*

Trên core DM642 có 3 video port. Những port này có thể được chia nhỏ cho phép các chức năng tùy chọn như McASP hoặc SPDIF trên port 0 và 1. DM642 EVM sử dụng tất cả 3 port này. Video port 0 và 1 được sử dụng như những input video và Video port 2 được sử dụng như output video. Trong cấu hình chuẩn của EVM, Video port 0 và 1 được lập trình để có thể chia nhỏ ra cho phép chức năng McASP được thiết lập và giao tiếp với TLV320AIC23 strereo codec, hoặc để giao tiếp với ngõ ra SPDIF, J9.

4.1.6.1. Video decoder port

Trên DM642 EVM, video port 0 và 1 được sử dụng như các capture input. Những port này giao tiếp với video decoder SAA7115H của Phillips. Các video port 0 và 1 được cho chạy trên CBT, các port còn lại được sử dụng cho giao tiếp McASP trên board. Video port 1 giao tiếp với nguồn video thông qua cáp RCA được kết nối với qua jack cắm J15 và 4 chân S-Video mini-din J16. Input này được lấy từ các nguồn video như DVD player hoặc video camera. SAA7115 được lập trình thông qua bus I^2C của DM642 và có thể giao tiếp với tất cả các chuẩn composite video như NTSC, PAL và SECAM bằng cách cấu hình tương ứng vào những thanh ghi bên trong decoder.

4.1.6.2. Video encoder port

Video port 2 của DM642 được sử dụng để điều khiển video encoder. Nó được nối thông quá FPGA U8 để thiết lập các chức năng đặc biệt như OSD FPGA, nhưng ở chế độ mặc định, video được đưa trực tiếp đến SA7105 video encoder của Phillips. Bộ encoder này có thể điều khiển xuất RGB, HD video, NTSC/PAL, hoặc S-Video phụ thuộc vào cách cấu hình các thanh ghi trong SA7105. SA7105 được cấu hình bằng cách lập trình vào các thanh ghi thông qua bus I^2C của DM642.

Encoder giao tiếp với các thành phần hiển thị chuẩn RGB hoặc composite video. Những chuẩn video RGB kết nối qua jack RCA được hỗ trợ qua J2, J3, J4. J3 là Green output và cũng có thể được sử dụng để giao tiếp với chuẩn composite video. Một ngõ ra S-Video 4 chân mini-din ở J1 cũng được hỗ trợ. 15 chan HD được hỗ trợ ở J5 cho phép EVM điều khiển các loại màn hình VGA.

DM642 EVM hỗ trợ HDTV output nhưng yêu cầu một số bộ lọc biến đổi như được đặc tả trong tài liệu HDTV support.

4.1.6.3. FPGA video functions

DM642 EVM sử dụng dòng FPGA Xilinx XC2S300E để thiết lập tăng cường cho các chức năng video cùng với một số các chức năng lập trình được khác. Trong chế độ mặc định, FPGA đi qua video từ video port 2 của DM642 đến SAA7105 video encoder. Với HDTV, FPGA hỗ trợ chế độ tăng cường clock cao và để hỗ trợ OSD FPGA thì FPGA có những FIFO phục vụ mục đích tổng hợp dữ liệu từ video port 2 và dữ liệu từ FIFO nội. FIFO bên trong FPGA được truy cập qua EMIF của DM642 trong chế độ đồng bộ ở vùng không gian CE 3.

4.1.7. ***Giao tiếp PCI/HPI/Ethernet***

DM642 hỗ trợ đa dạng các loại giao tiếp ngoại vi. DSP đa hợp giữa PCI brigde, HPI và EMAC.

4.1.7.1. Giao tiếp PCI

DM642 hỗ trợ giao tiếp PCI một cách trực tiếp. CBT được sử dụng để phân chia bus PCI từ DM642 nhằm giúp EVM có thể sử dụng được cả giao tiếp PCI và Ethernet. CBT cũng cấp logic 5V hỗ trợ cho giao tiếp PCI.

Thông qua tín hiệu PCI-detect, CBT được cấu hình tự động cho hoạt động PCI khi board được gắn vào khe PCI.

4.1.7.2. Giao tiếp với EEPROM

DM642 EVM hỗ trợ một EEPROM ngoài. Khi được dùng, nó lưu giữ những giá trị tùy chọn của cấu hình PCI. EEPROM được kích hoạt từ switch cấu hình S2. Khi S2-1 on, cấu hình PCI sử dụng những thông số trong EEPROM làm cấu hình. Nếu S2-1 off, những thanh ghi mặc định trong DM642 được sử dụng cho việc cấu hình PCI.

Bảng 4-10. EEPROM memory map

Address	Value	Description of Contents
0x00	0x104C	Vendor ID
0x01	0x9065	Device ID
0x02	0x0000	Calls Code [7:0}/Revision ID
0x03	0xFF00	Class Code [23:8]
0x04	0x1652	Subsystem Vendor ID
0x05	0x0642	Subsystem ID
0x06	0x0000	Max_Latency/Min_Grant
0x07	0x0000	PC_D1/PC_D0 Power Consumed
0x08	0x0000	PC_D3/PC_D2 Power Consumed
0x09	0x0000	PD_D1/PC_D0 Power Dissipated
0x0A	0x0000	PD_D3/PC_D2 Power Dissipated
0x0B	0x0000	Data_Scale (PD_D3 ... PC_D0)
0x0C	0x0000	0000 0000 PMC[14:9], PMC[5], PMC[3]
0x00D	0xC593	Checksum

4.1.7.3. Giao tiếp Ethernet

Ở chế độ standalone, EMAC được chọn một cách tự động và được nối đến PHY thông qua những bộ CBT. DM642 EVM sử dụng LXT971 PHY của Intel. Giao tiếp 10/100Mbit này được tách riêng và kết nối ra ngoài cổng RJ-45, J18. PHY giao tiếp trực tiếp với DM642. Địa chỉ MAC được lưu giữ trong I^{2}C ROM được ghi trong quá trình sản xuất.

RJ-45 có 2 led, xanh và vàng, được tích hợp vào bên trong jack cắm để hiển thị trạng thái của ethernet. Led xanh sáng để báo hiệu liên kết và khi nhấp nháy nghĩa là liên kết đang hoạt động. Led vàng khi sáng sẽ báo hiệu chế độ truyền full-duplex.

4.1.8. *Giao tiếp I^{2}C*

Bus I^{2}C trên DM642 được sử dụng cho giao tiếp để điều khiển các thanh ghi của nhiều thiết bị. Bus I^{2}C được dùng để cấu hình video encoder, video decoder, và stereo codec. I^{2}C ROM cũng được giao tiếp thông qua bus nối tiếp.

Start	Slave Address	W	ACK	Sub Address	ACK-S	Data	ACK-S	Stop

Write Sequence

Start	Slave Address	W	Sub Address	ACK-S	
Start	Slave Address	R	Data	ACK-S	STOP

Read Sequence

Hình 4.4. I^{2}C Bus format

Bảng 4-11. I^{2}C memory map

Device	Address	R/W	Function
SAA7115	0x42	R/W	Capture 1 Decoder
SAA7115	0x40	R/W	Capture 2 Decoder
SAA7105	0x88	R/W	Encoder
TLV320AIC23	0x1A	R/W	CODEC
24WC256	0x50	R/W	I^{2}C EEPROM

4.1.9. *DM642 EVM Software:*

- Installation support for CCStudio v2.2 and v3.0.
- DM642 Chip Support Package, DSP/BIOS kernel support for the new peripherals (Video Port, VIC, EMAC, MDIO)
- Device Driver Kit (DDK), which includes drivers for DM642 video capture and display drivers for the DM642 EVM card, AIC23 audio codec driver, Dual UART driver, Generic McASP data mover driver for C64x DSPs, and PCI target-side driver from Valley Technologies.
- eXpressDSP Reference Framework 5
- Video Application Demos including source code for: Capture/Display Pass-thru Demos (Composite/S-Video In/Out), On-Screen Display (OSD), Audio Loopback
- Streaming Media Executable Demos (with object code libraries) Video Loopback demos
- (H.263, MPEG-2 video and JPEG), TCP/IP Networking stack (Client/Configuration), Network Camera Demo and Audio Loopback (G.729)

4.2. **TMS320DM642 Video/Imaging Fixed-Point DSP**

Họ TMS320C64x DSP (bao gồm TMS320DM642) là họ DSP fixed-point có hiệu suất cao nhất trong dòng TMS320C6000 DSP. Chip TMS320DM642 (DM642) dựa trên kiến trúc VelociTI™ nâng cao (VelociTI.2™) very-long-instruction-word (VLIW), là một sự lựa chọn tuyệt vời cho các ứng dụng xử lý ảnh và xử lý âm thanh.

4.2.1. *Features*

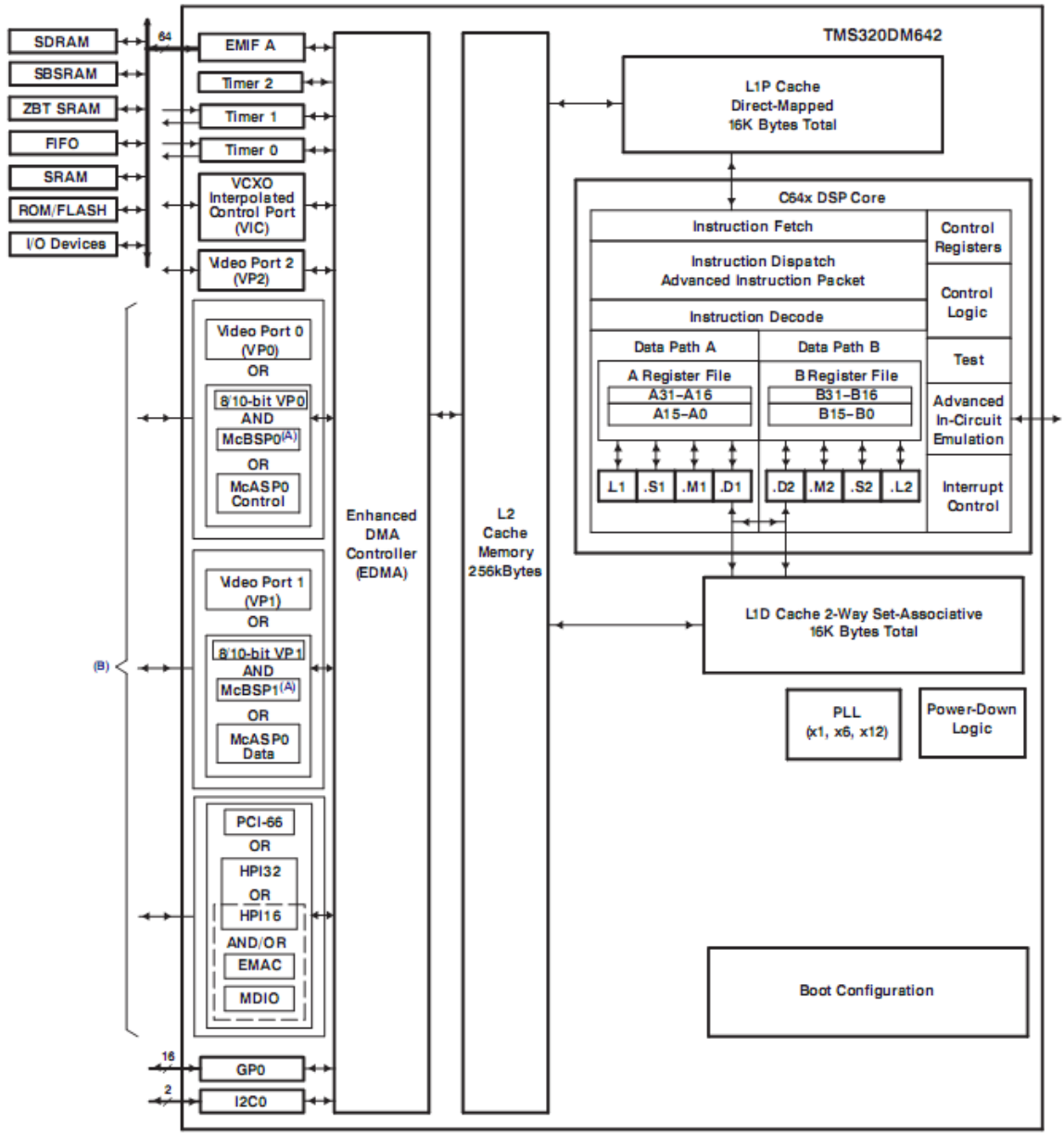
- **High-Performance Digital Media Processor**
  - 1.39-ns Instruction Cycle Time
  - 720-MHz Clock Rate
  - Eight 32-Bit Instructions/Cycle
  - 5760 MIPS
  - Fully Software-Compatible With C64x™
- **VelociTI.2™ Extensions to VelociTI™ Advanced Very-Long-Instruction-Word (VLIW) TMS320C64x™ DSP Core**

- Eight Highly Independent Functional Units With VelociTI.2™ Extensions:
    - Six ALUs (32-/40-Bit), Each Supports Single 32-Bit, Dual 16-Bit, or Quad 8-Bit Arithmetic per Clock Cycle
    - Two Multipliers Support Four 16 x 16-Bit Multiplies (32-Bit Results) per Clock Cycle or Eight 8 x 8-Bit Multiplies (16-Bit Results) per Clock Cycle
- Load-Store Architecture With Non-Aligned Support
- 64 32-Bit General-Purpose Registers
- Instruction Packing Reduces Code Size
- All Instructions Conditional

- **Instruction Set Features**
    - Byte-Addressable (8-/16-/32-/64-Bit Data)
    - 8-Bit Overflow Protection
    - Bit-Field Extract, Set, Clear
    - Normalization, Saturation, Bit-Counting
    - VelociTI.2™ Increased Orthogonality
- **L1/L2 Memory Architecture**
    - 128K-Bit (16K-Byte) L1P Program Cache (Direct Mapped)
    - 128K-Bit (16K-Byte) L1D Data Cache (2-Way Set-Associative)
    - 2M-Bit (256K-Byte) L2 Unified Mapped RAM/Cache (Flexible RAM/Cache Allocation)
- **Endianess**: Little Endian
- **64-Bit External Memory Interface (EMIF)**
    - Glueless Interface to Asynchronous Memories (SRAM and EPROM) and Synchronous Memories (SDRAM, SBSRAM, ZBT SRAM, and FIFO)
    - 1024M-Byte Total Addressable External Memory Space
- **Enhanced Direct-Memory-Access (EDMA) Controller (64 Independent Channels)**
- **10/100 Mb/s Ethernet MAC (EMAC)**
    - IEEE 802.3 Compliant
    - Media Independent Interface (MII)
    - 8 Independent Transmit (TX) Channels and 1Receive (RX) Channel
- **Management Data Input/Output (MDIO)**
- **Three Configurable Video Ports**
    - Providing a Glueless I/F to Common Video Decoder and Encoder Devices
    - Supports Multiple Resolutions/Video Stds

- **VCXO Interpolated Control Port (VIC)**
  - Supports Audio/Video Synchronization
- **Host-Port Interface (HPI) [32-/16-Bit]**
- **32-Bit/66-MHz, 3.3-V Peripheral Component Interconnect (PCI) Master/Slave Interface Conforms to PCI Specification 2.2**
- **Multichannel Audio Serial Port (McASP)**
  - Eight Serial Data Pins
  - Wide Variety of I2S and Similar Bit Stream Formats
  - Integrated Digital Audio I/F Transmitter Supports S/PDIF, IEC60958-1, AES-3, CP-430 Formats

4.2.2. *Device Overview*



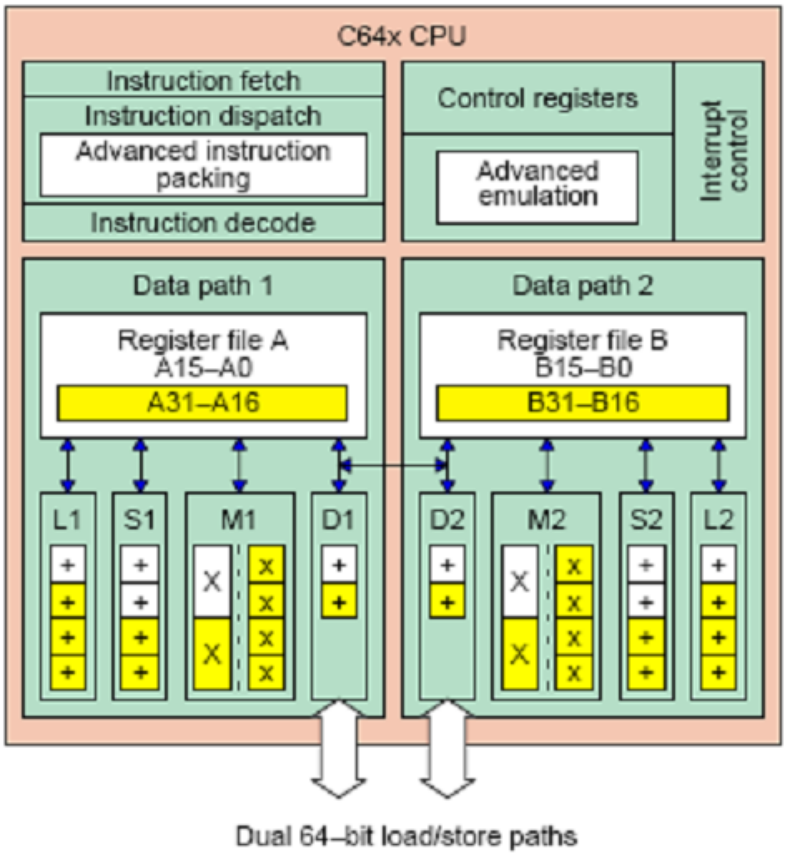
Hình 4.5. TMS320DM642

4.2.2.1. CPU (DSP core)

Với kiến trúc VelociTI™ nâng caoVLIWs, CPU có thể nạp đến 8 lệnh 32-bit vào 8 đơn vị chức năng trong mỗi chu kỳ clock. Bit đầu tiên của mỗi lệnh 32-bit xác định lệnh tiếp theo thuộc cùng execute packet với lệnh trước đó, hay xác định nó có

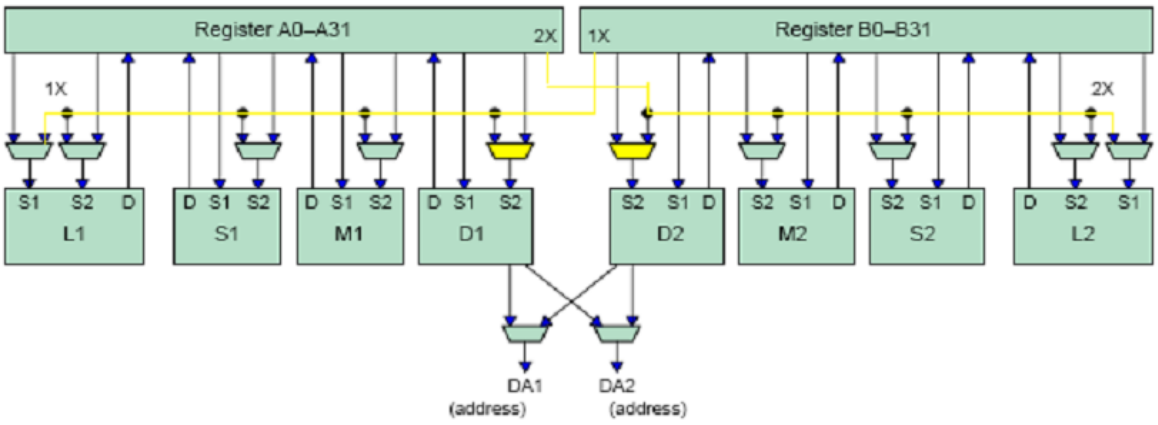
được thực hiện trong clock sau như một phần của execute packet kế tiếp. Các fetch packet thường có độ rộng 256-bit, tuy nhiên, các execute packet có thể có nhiều kích thước khác nhau. Execute packet với chiều dài thay đổi để tiết kiệm bộ nhớ là tính nằng then chốt để phân biệt dòng CPU C64x với những CPU có kiến trúc VLIW khác. C64x™ VelociTI.2™ được cải tiến từ kiến trúc TMS320C62x™ DSP VelociTI™. Những cải tiến đó bao gồm:

- Register file enhancements
- Data path extensions
- Quad 8-bit and dual 16-bit extensions with data flow enhancements
- Additional functional unit hardware
- Increased orthogonality of the instruction set
- Additional instructions that reduce code size and increase register flexibility



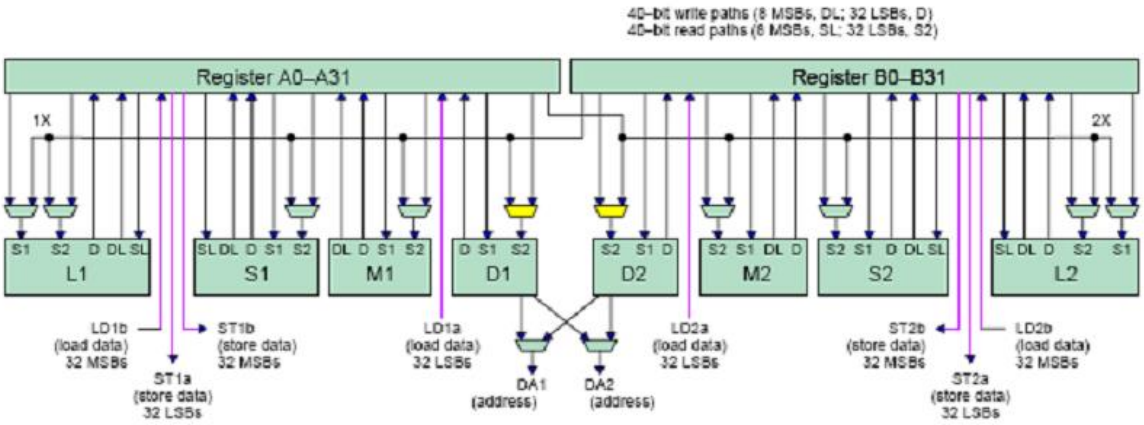
Hình 4.6. C64x CPU

CPU gồm 2 bộ đơn vị chức năng. Mỗi bộ gồm 4 unit và 1 register file. Một bộ chứa các unit .L1, .S1, .M1, và .D1. Bộ còn lại chứa .D2, .M2, .S2, và .L2. Mỗi register file chứa 32 thanh ghi 32-bit để có tổng cộng 64 thanh ghi general-purpose. Ngoài việc hỗ trợ packed 16-bit và 32-/40-bit fixed-point data types như trong kiến trúc C62x™ VelociTI™ VLIW, C64x™ register files còn hỗ trợ cho packed 8-bit data và 64-bit fixed-point data types. Hai bộ functional unit cùng với 2 register file chia thành 2 bên A và B của CPU. Các functional unit ở cùng 1 bên sử dụng chung 32 thanh ghi thuộc về bên đó. Ngoài ra, mỗi bên có 1 "data cross path" – một data bus nối với tất cả các thanh ghi ở bên kia, do đó, các functional unit ở bên này có thể truy xuất dữ liệu từ các thanh ghi ở bên kia. C64x CPU pipelines data-cross-path truy xuất qua nhiều chu kỳ clock. Điều này cho phép cùng một thanh ghi được dùng như một data-cross-path operand bởi nhiều functional unit trong cùng execute packet. Tất cả functional unit trong C64x CPU có thể truy xuất operands thông qua data cross path. Register file có thể đáp ứng tất cả các yêu cầu truy xuất từ các functional unit ở cùng bên trong một chu kỳ clock. Trong C64x CPU, nếu có lệnh yêu cầu đọc thanh ghi khi thanh ghi này vừa được cập nhật từ chu kỳ clock trước đó thì CPU sẽ yêu cầu delay 1 chu kỳ clock.



Hình 4.7. C64x Data Cross Paths

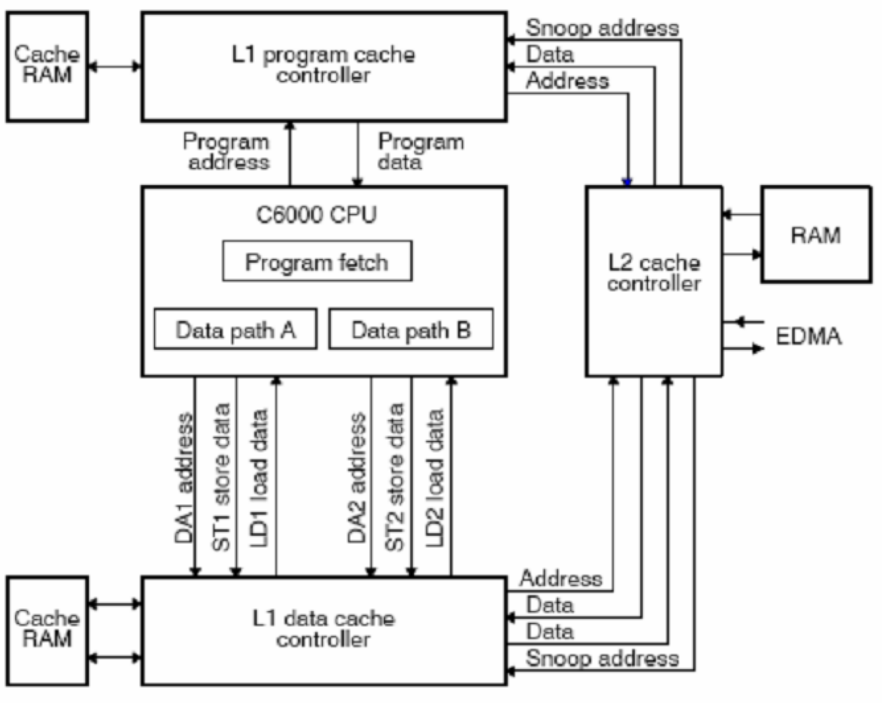
Một tính năng quan trọng khác của C64x CPU là kiến trúc load/store, tất cả lệnh này đều hoạt động trên thanh ghi. 2 bộ data-addressing unit (.D1 và .D2) chịu trách nhiệm cho tất cả các quá trình trao đổi dữ liệu giữa register file và bộ nhớ. Dữ liệu địa chỉ điều khiển bởi .D unit cho phép các dữ liệu địa chỉ tạo ra từ một register file được dùng để load hay store dữ liệu từ một register file khác. C64x .D unit có thể load và store byte (8-bit), half-word (16-bit), word (32-bit) trong một lệnh. Và với data path mới, C64x .D unit có thể load và store double-word (64-bit) trong một lệnh. Hơn thế nữa, lênh load và store non-aligned cho phép .D unit truy xuất một word và một double-word trong bất kỳ byte boundary.



Hình 4.8. C64x Memory Load and Store Paths

Hai .M functional unit thực hiện tất cả các phép toán nhân. Mỗi C64x .M unit có thể thực hiện 2 phép nhân 16 × 16 bit hay 4 phép nhân 8 × 8 bit trên một chu kỳ clock. .M unit có thể thực hiện phép nhân 16 × 32 bit, 2 phép nhân 16 × 16 bit với phép cộng/trừ, và 4 phép nhân 8 × 8 bit với phép cộng. Ngoài phép nhân chuẩn, C64x .M unit bao gồm bit-count, rotate, nhân Galois field, và phần cứng ghi dịch 2 chiều.

Hai .S và .L functional unit thực hiện 1 tập arithmetic, logical, chức năng rẻ nhánh với kết quả sẵn sàng sau một chu kỳ clock. Chức năng arithmetic và logical trên C64x CPU bao gồm phép toán single 32-bit, dual 16-bit, và quad 8-bit.



Hình 4.9. TMS320C64x Two Level Internal Memory Block Diagram

4.2.2.2. Enhanced Direct Memory Access (EDMA) Controller

EDMA controller điều khiển tất cả các quá trình trao đổi dữ liệu giữa bộ nhớ đệm cấp 2 (*L2 cache/memory*) và những thiết bị ngoại vi. Các quá trình trao đỗi dữ liệu này bao gồm: các dịch vụ của bộ nhớ đệm, truy xuất bộ nhớ không thể sử dụng bộ nhớ cache, các quá trình trao đổi dữ liệu do người dùng lập trình và các truy xuất từ máy tính (*host*).

DM642 hổ trợ đến 64 kênh EDMA phục vụ cho các thiết bị ngoại vi và bộ nhớ ngoài. Mỗi kênh EDMA có một sự kiện đồng bộ tương ứng, các sự kiện này được xác định trong thanh ghi sự kiện của EDMA (ERL và ERH) ngay cả khi sự kiến đó bị vô hiệu hóa bởi thanh ghi kích hoạt sự kiện (EERL và EERH). Độ ưu tiên của mỗi sự kiện có thể được xác định một cách độc lập bằng các thông số của quá trình trao đổi được lưu trong EDMA RAM.

4.2.2.3. Video Port

Mỗi Video Port có khả nằng gửi và nhận dữ liệu video dạng số. Các Video Port cũng có thể capture/display dữ liệu RAW. Các Video Port ngoại vi phải tuân theo các chuẩn video như BT.656 và SMPTE296.

DM642 có ba video port ngoại vi. Video port ngoại vi có thể hoạt động như một video capture port, video display port.

Mỗi port có 2 kênh: A và B. Một capture/display buffer 5120 byte được dùng cho cả 2 kênh. Tất cả port chỉ có thể cấu hình hoặc là capture video hoặc là display video.

Đối với hoạt động video capture, video port có thể hoạt động như 2 kênh 8/10-bit BT.656 hay raw video capture, hoặc như 1 kênh riêng lẻ 8/10-bit BT.656, 8/10-bit raw video, 16/20-bit Y/C video, 16/20-bit raw video.

Đối với hoạt động video display, video port có thể hoạt động như một kênh riêng lẻ 8/10-bit BT.656 hay 8/10-bit raw video, 16/20-bit Y/C video, hay 16/20 raw video. Nó cũng có thể hoạt động trong chế 2 kênh 8/10-bit raw video nhưng khi đó 2 kênh có cùng timing.

4.2.2.4. Development Support

TI cung cấp đầy đủ các công cụ phát triển cho dòng TMS320C6000™ DSP, bao gồm công cụ đánh giá hiệu năng của bộ xử lý, generate code, triển khai thuật toán và đầy đủ các module hổ trợ debug.

Sau đây là những sản phẩm hổ trợ cho việc phát triển ứng dựng dựa trên nền tảng C6000™ DSP:

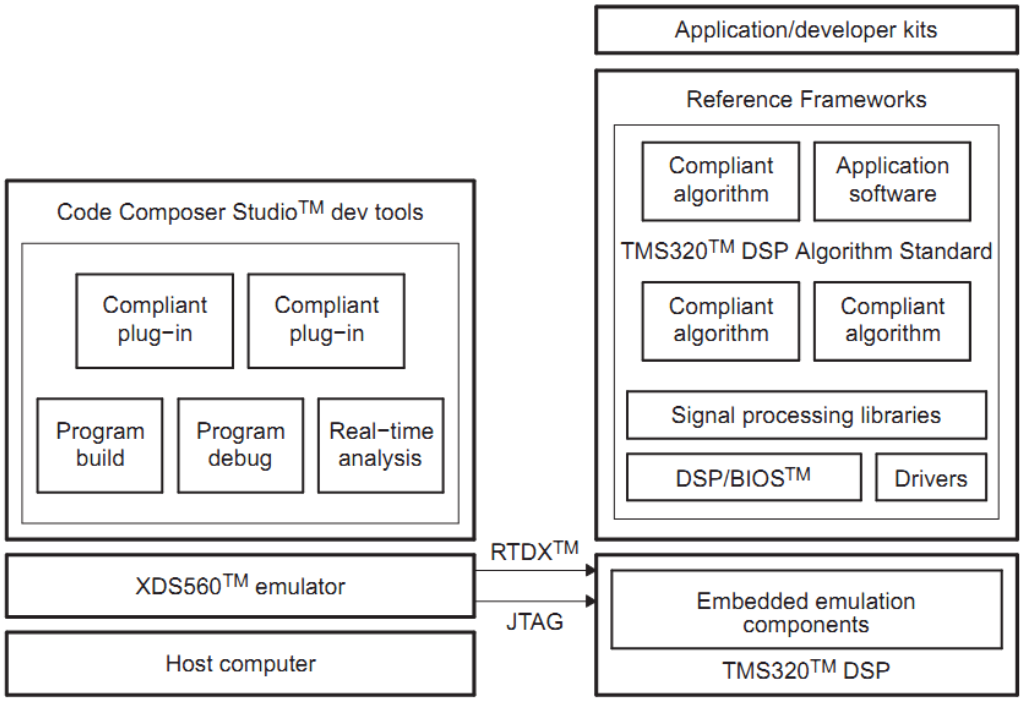
Công cụ phát triển phần mềm:

- Code Composer Studio™ Integrated Development Environment (IDE): bao gồm trình soạn thảo.

- C/C++/Assembly Code Generation, và Debug cùng với những công cụ phát triển bổ sung.
- Real-Time Foundation Software (DSP/BIOS™ kernel)

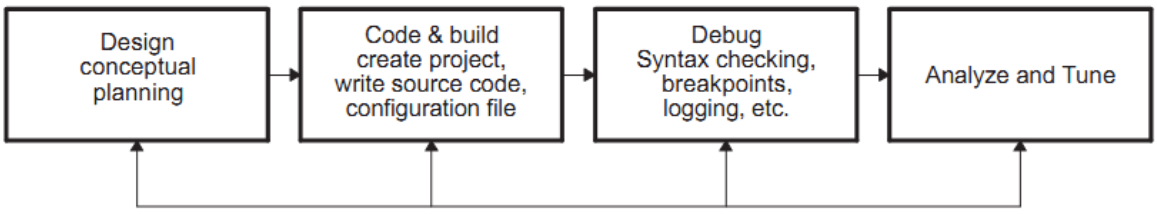
Công cụ phát triển phần cứng:

- Extended Development System (XDS™) Emulator
- EVM (Evaluation Module)



Hình 4.10. Software and Development Tools

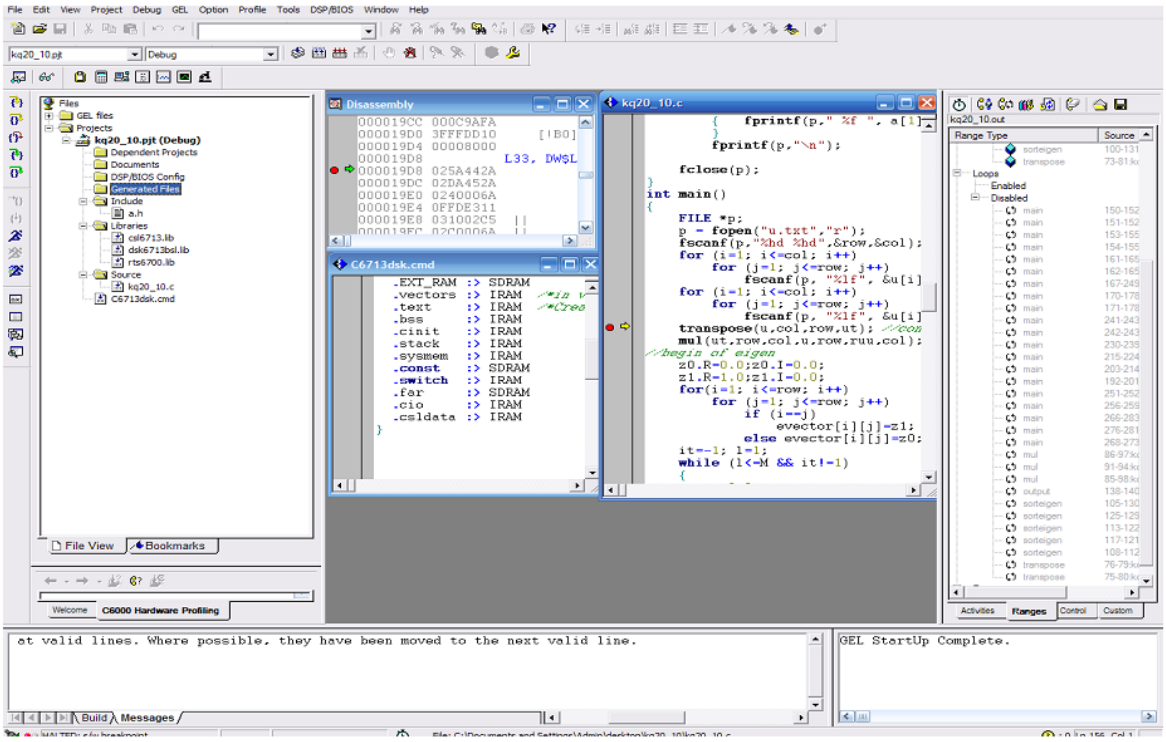
Quá trình phát triển của hầu hết các ứng dụng dựa trên nền tảng DSP gồm 4 bước: thiết kế ứng dụng, coding, debug và analysis/tuning.



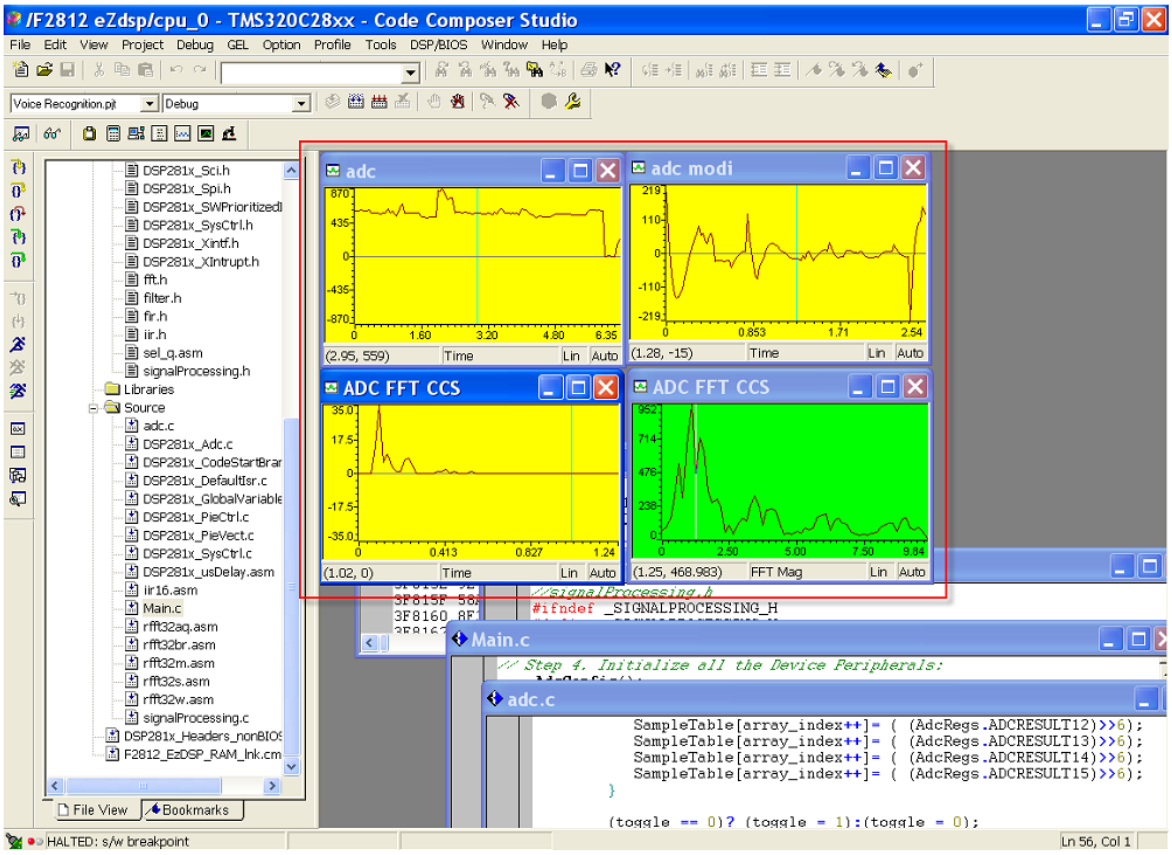
Hình 4.11. Development Flow

4.3. **Code Composer Studio**

Code Composer Studio (CCStudio) là IDE cho các dòng DSP của Texas Instruments (TI). CCStudio bao gồm một bộ công cụ để phát triển và debug các chương trình ứng dụng nhúng. CCStudio cung cấp trình biên dịch cho các dòng DSP khác nhau, trình soạn thảo, gỡ lỗi, đánh giá hiệu năng của mã nguồn , hệ điều hành thời gian thực (RTOS) và nhiều tính năng khác. Điều đặc biệt là CCStudio hỗ trợ các tính năng phân tích hệ thống thời gian thực giúp kiểm soát các thông số trong quá trình hoạt động, từ đó có thể thực hiện tối ưu hóa để hệ thống hoạt động ổn định và hiệu quả.



Hình 4.12. Giao diện chương trình Code Composer Studio



Hình 4.13. Vẽ đồ thị thời gian thực trên CCS

Trong CCS có nhiều dạng file với nhiều mục đích khác nhau. Bảng dưới đây giới thiệu một số dạng file và vai trò của nó [7]:

Bảng 4-12. Các dạng file sử dụng trong CCS

STT	*Dạng file*	*Vai trò*
1	file.pjt	Tập tin được tạo khi xây dựng 1 project để quản lý file và thông tin của project
2	file.c	Các tập tin chứa code C, phần thuật toán chính của chương trình
3	file.asm	Các tập tin chứa mã hợp ngữ được tạo ra bởi trình biên dịch C, hoặc do người sử dụng
4	file.h	Các tập tin tiêu đề hỗ trợ cho project, có thể sử dụng để chứa hàm hoặc dữ liệu
5	file.lib	Các thư viện hỗ trợ khởi tạo Chip, mạch, ADC và các thiết bị ngoại vi trên bảng mạch
6	file.cmd	Các tập tin liên kết giúp phân hoạch các vùng chương trình và dữ liệu vào bộ nhớ
7	file.out	Tập tin khả thi, được tạo bởi CMD file để có thể nạp vào kit
8	file.cdb	Tập tin cấu hình, sử dụng khi dùng tính năng DSP/BIOS

Ngoài ra, TI còn cung cấp rất nhiều các thư viện hổ trợ cho DSP, các hàm trong các thư viện này được dùng để tính toán chuyên sâu trong các ứng dụng real-time đòi hỏi tốc độ thực thi nhanh và độ chính xác cao. Các hàm trong các thư viện này đã được TI tối ưu vì thế tốc độ thực thi được cải thiện khá đáng kể so với các code tương đương được viết bằng ngôn ngữ C chuẩn. Với việc sử dụng các hàm có sẵn trong thư viện được TI cung cấp, thời gian cho việc thực hiện và phát triển các ứng dụng, thuật toán trên DSP sẽ rút ngắn đáng kể. Một số thư viện mà đề tài sử dụng như:

- **C64x+ IQMath Library:** Thư viện IQMath gồm các hàm toán học tính toán số fixed-point có độ chính xác cao và đã được tối ưu, hỗ trợ tính toán các biến dạng Q-point. Thư viện này bao gồm nhiều hàm số học, lượng giác và toán học khác.

- **FastRTS Library:** Thư viện FastRTS bao gồm nhiều hàm hỗ trợ tính toán các số floating-point trên nền DSP fixed-point.
- **DSPLIB:** DSPLIB bao gồm rất nhiều các hàm xử lý tín hiệu số phục vụ cho việc xây dựng các ứng dụng xử lý tín hiệu. DSPLIB hỗ trợ một số hàm như:
    - Adaptive Filtering
    - Correlation
    - Fast Fourier Transform
    - Filtering and Convolution
    - Matrix Computations
- **Image and Video Processing Libraries (IMGLIB):** IMGLIB bao gồm hơn 70 hàm hỗ trợ cho việc xây dựng các ứng dụng xư lý ảnh. IMGLIB bao gồm các hàm:
    - Compression and Decompression
        - Forward and Inverse DCT
        - Motion Estimation
        - Quantization
        - Wavelet Processing
    - Image Analysis
        - Boundary and Perimeter Estimation
        - Morphological Operations
        - Edge Detection
        - Image Histogram
        - Image Thresholding
    - Image Filtering & Format Conversion
        - Image Convolution
        - Image Correlation
        - Median Filtering
        - Color Space Conversion
        - Error Diffusion
        - Pixel Expansion
- **Video Analytics & Vision Library (VLIB):** VLIB là thự viện phần mềm với hơn 40 hàm giúp rút ngắn thời gian phát triển các ứng dụng phân tích video với hiệu suất nhanh hơn gấp 10 lần. VLIB có khả năng thực hiện:
        - Background Modeling & Subtraction
        - Object Feature Extraction
        - Tracking & Recognition
        - Low-level Pixel Processing

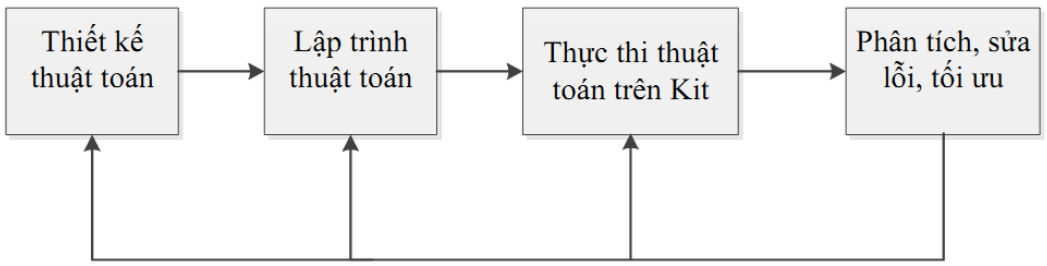
VLIB tạo nền tảng để phát triển các ứng dụng:

- Video Analytics
- Video Surveillance
- Automotive Vision
- Embedded Vision
- Game Vision
- Machine Vision
- Consumer Electronics

CHƯƠNG 5. XÂY DỰNG HỆ THỐNG NHẬN DẠNG CỬ CHỈ TRÊN DSP

Ngày nay với sự phát triển không ngừng của các phần mềm mô phỏng như Matlab, chúng ta có khả năng thiết kế thuật toán chỉ dựa vào các công cụ Simulink, sau đó sử dụng các toolbox như Real-Time Workshop để sinh ra code C, từ đó đưa vào kit để thực hiện. Tuy nhiên code được tạo ra có dung lượng lớn và khó để tùy biến tối ưu về sau. Vì vậy em lựa chọn giải pháp tự thiết kế thuật toán trên C, sau đó chuyển code vào môi trường CCS để biên dịch thành chương trình chạy cho kit DSP.

Các bước xây dựng thuật toán trên nền tảng DSP bao gồm 4 bước được mô tả bởi lưu đồ dưới đây:



Hình 5.1. Mô hình xây dựng thuật toán

5.1. Thiết kế thuật toán

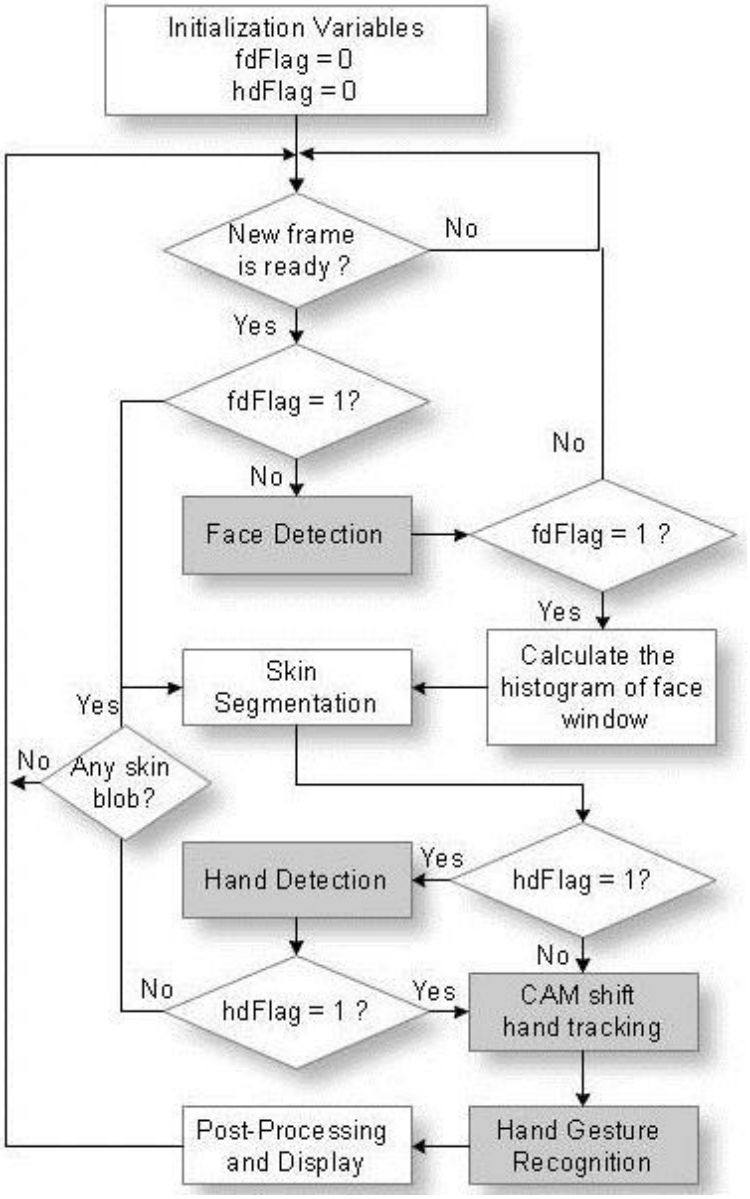
Cụ thể về thuật toán nhận dạng và phân loại cử chỉ đã được nghiên cứu ở Chương 1. Tuy nhiên khi nhúng vào DSP, ta phải làm sao cho chương trình không quá nặng nề với chip DSP, bảo đảm chương trình vẫn cho kết quả chính xác với tốc độ nhanh nhất. Ta có thể mô phỏng thuật toán trên PC sử dụng thư viện OpenCV để kiểm tra khả năng hoạt động cũng như độ chính xác của thuật toán, trước khi porting chương trình lên kit DSP.

5.2. **Lập trình thuật toán**

Sau khi đã có được giải thuật đúng đắn, đã được kiểm chứng trên PC bằng OpenCV, ở bước này ta tiến hành chuyển đổi code C được viết bằng cách sử dụng các hàm của OpenCV thành code C chuẩn. OpenCV là thư viện rất mạnh hỗ trợ gần như đầy đủ các hàm cần thiết cho các chương trình xử lý ảnh và video. Việc chuyển đổi này đòi hỏi người lập trình phải hiểu rõ thuật toán của các bước xử lý. Ngoài ra người lập trình cần hiểu rõ cấu trúc của DSP và cách thức quản lý bộ nhớ trên DSP để có thể tận dụng hết sức mạnh xử lý tín hiệu số của chip DSP.

Chip DSP được sử dụng trong đề tài này là chip DSP fixed-point. Tuy nhiên hầu hết các thuật toán xử lý ảnh đều đòi hỏi khả năng tính toán các số floating-point, do đó trong quá trình lập trình thuật toán trên DSP cần thực hiện chuyển đổi các phép tính floating-point sang fixed-point để chương trình hoạt động nhanh hơn giảm gánh nặng tính toán cho DSP.

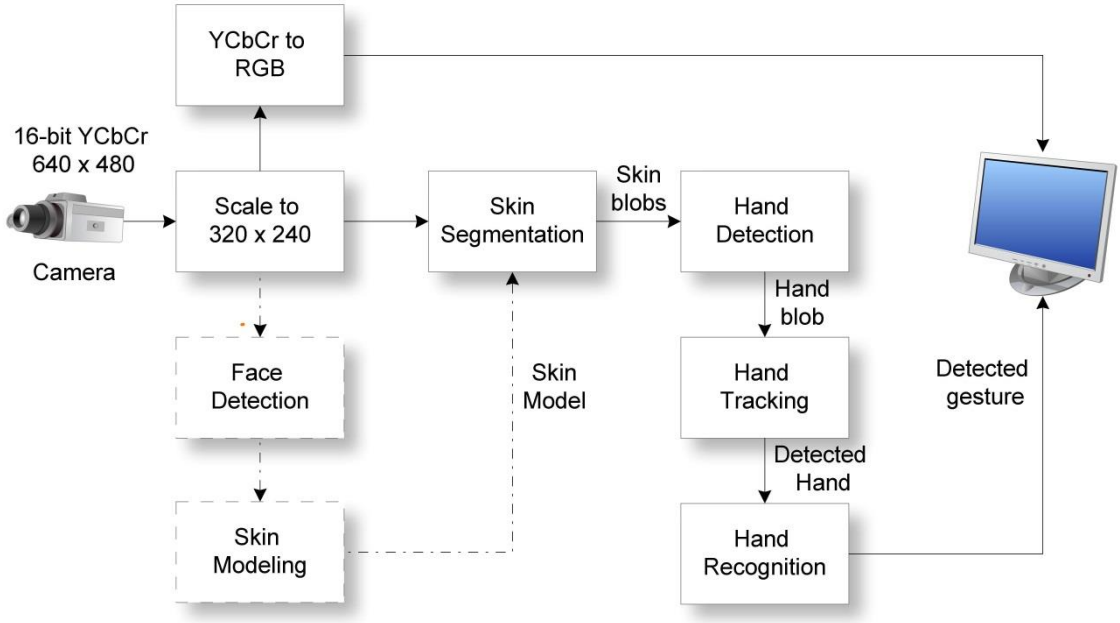
Giải thuật cài đặt trên DSP có lưu đồ như sau:



Hình 5.2. Lưu đồ chương trình trên DSP

5.3. **Thực hiện thuật toán trên board**

Với mục đích tập trung vào thuật toán và khả năng thực hiện thuật toán trên board DSP, đề tài không đi sâu vào việc phát triển các khối thực thi ngoại vi như khối gửi tín hiệu điều khiển về máy tính sau khi đã nhận dạng được một cử chỉ. Hệ thống sử dụng một camera thu hình ảnh đưa vào board DSP và dùng một màn hình VGA để hiển thị kết quả. Hệ thống gồm các khối cỏ bản như sau:



Hình 5.3. Sơ đồ khối hệ thống

5.3.1. *Image scale*

Camera kết nối với board EVM642 qua cổng Composite có thể thu được hình ảnh ở độ phân giải 640 × 480 pixel. Tuy nhiên, việc lưu trữ và xữ lý hình ảnh ở độ phân giải này chiếm quá nhiều tài nguyên hệ thống và không cần thiết cho quá trình phát hiện cử chỉ. Do đó, hình ảnh sau khi được camera đưa vào được scale xuống thành 320 × 240 pixel.

Thư viện VLIB cung cấp hàm Image Pyramid hỗ trợ việc down-scale hình ảnh xuống các kích thước khác nhau từ 1/2, 1/4 đến 1/8.

```
int VLIB_imagePyramid16(
            unsigned short * restrict pIn,
            unsigned short inCols,
            unsigned short inRows,
            unsigned short * restrict pOut);
```

5.3.2. *YcbCr to RGB convertion*

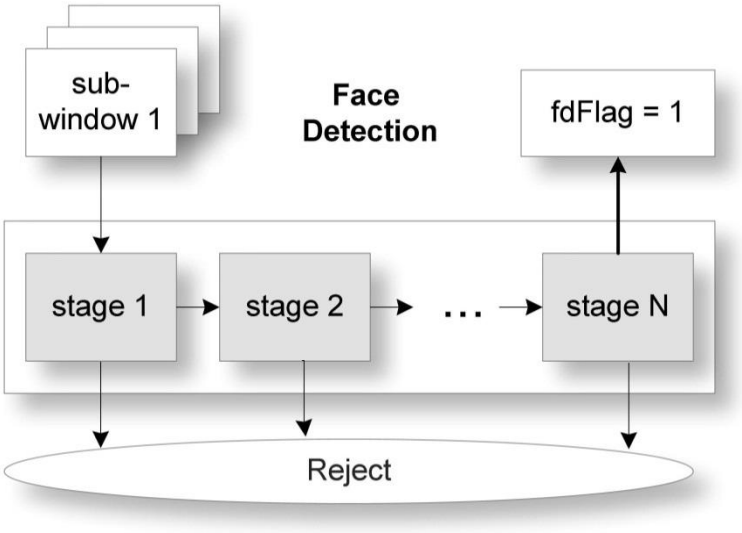
Hình ảnh từ camera đi vào ở không gian màu YCbCr, để có thể xuất ra màn hình VGA hình ảnh này cần được chuyển sang dạng RGB. Hình ảnh ở dạng RGB này, vừa được dùng để hiển thị trên màn hình VGA vừa được dùng cho việc xây dựng mô hình màu da và sử dụng trong thuật toán CAMshift.

Việc chuyển đổi được tiến hành bằng sự hỗ trợ của thư viện VLIB.

```
int VLIB_convertUYVYint_to_RGBpl(
            const unsigned char *yc,
            int width,
            int pitch,
            int height,
            const short coeff[5],
            unsigned char *restrict r,
            unsigned char *restrict g,
            unsigned char *restrict b);
```

5.3.3. *Phân vùng màu da*

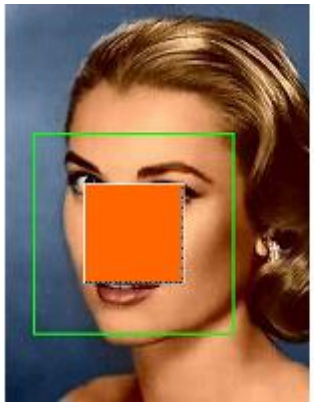
Như đã đề cập ở Chương 2, đề tài xây dựng khối phân vùng màu da có khả năng tương thích với các điều kiện ánh sáng khác nhau cũng như màu da của từng người dùng. Ý tưởng chính là sử dụng thông tin phân phối màu da trên gương mặt của người dùng để xây dựng thành một mô hình màu da cụ thể. Mô hình màu da chứa các thông tin được lấy ra từ gương mặt của người dùng và điều kiện ánh sáng hiện tại. Do đó, ta có một bộ phát hiện màu da khá mạnh, có thể đối phó với những điều kiện ánh sáng khác nhau hay với những chủng loại màu da khác nhau. Các tiếp cận này đòi hỏi phải có một bộ phát hiện gương mặt đáng tin cậy. Đề tài này chọn sử dụng phương pháp của Viola và Jones vì những ưu điểm đã được phân tích ở Chương 1.



Hình 5.4. Face Detector

Sau khi đã xác định được gương mặt trên hình ảnh từ camera đưa vào. Một vùng các pixel trên gương mặt được chọn ra để xây dựng mô hình màu da (skin model). Việc chọn ra vùng chứa các pixel để mô hình hóa được thực hiện trên ảnh RGB565 – 16bit. Tức là, mỗi pixel đươc chứa trong một biến kiểu *unsigned short* với giá trị gồm 5-bit Red, 6-bit Green, và 5-bit Blue. Giá trị mỗi pixel được lấy ra để tiến hành tính toán Histogram. Tuy nhiên, nếu thực hiện tính toán trực tiếp các pixel như thế này thì Histogram sẽ có tổng cộng 65535 bin, số lượng này quá lớn và không cần thiết. Nhằm tiết kiệm thời gian tính toán, trước khi đưa vào tính toán Histogram, mỗi pixel được chia cho 16 (thực hiện dịch sang phải 4 lần). Vậy Histogram lúc này có 4096 bin,vẫn đáp ứng được yêu cầu về độ chính xác.

Việc tính toán Histogram tức là thực hiện đếm số lần xuất hiện một giá trị của pixel. Số lần xuất hiện của một giá trị pixel trong vùng đang xét trên gương mặt có thể được xem là xác suất một giá trị pixel là màu da.



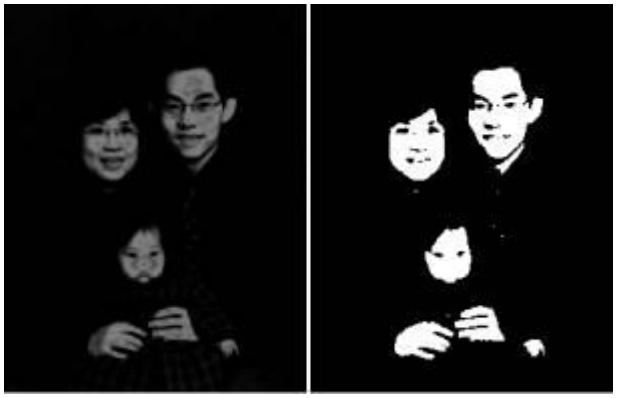
$$x_{0,orange} = x_{0,green} + 0.25 \cdot width_{green}$$
$$y_{0,orange} = y_{0,green} + 0.25 \cdot height_{green}$$
$$width_{orange} = 0.5 \cdot width_{green}$$
$$height_{orange} = 0.5 \cdot height_{green}$$

Hình 5.5. Chọn vùng trên gương mặt để làm mô hình màu da

Sau khi đã thực hiện tính toán Histogram, ta thu được một bảng tra (Lookup table) với giá trị của mỗi pixel (sau khi đã chia cho 16) ứng với một giá trị xác suất pixel đó là màu da.

Với bảng tra thu được, khi có những frame ảnh kế tiếp, ta thực hiện lấy từng pixel của frame ảnh so với bảng tra để thu được giá trị xác suất và thay vào giá trị của pixel đó. Với thao tác này, sau khi đã thực hiện với tất cả các pixel, ta thu được một hình ảnh với giá trị của mỗi pixel là giá trị xác suất pixel đó là màu da. Hình ảnh này, giống như một ảnh grayscale với pixel có xác suất là màu da càng lớn thì càng sáng và ngược lại.

Tiếp tục tiến hành nhị phân hóa hình ảnh thu được, ta được một ảnh nhị phân với các pixel có giá trị xác suất lớn hơn một giá trị ngưỡng được xác định trước là 1 còn các pixel khác là 0.



Hình 5.6. Phân vùng màu da

Ngoài ra, để giảm các pixel nhiễu nằm riêng lẻ, hình ảnh nhị phân này còn được tiến hành xử lý bằng các phép toán hình thái học như Erosion và Dilation như đã trình bày ở Chương 2. Bước này được tiến hành bằng cách sử dụng hàm được TI cung cấp trong thư viện VLIB:

```
int VLIB_dilate_bin_square(
            const unsigned char *restrict in_data,
            unsigned char *restrict out_data,
            int cols
            int pitch);
void VLIB_erode_bin_square(
            const unsigned char *restrict in_data,
            unsigned char *restrict out_data,
            int cols
            int pitch);
```

Sau đó, hình ảnh nhị phân tiếp tục được thực hiện gán nhãn cho các thành phần liên thống, nhằm loại bỏ các khối có màu giống màu da nhưng có tỉ lệ chiều dài và chiều rộng thấp hơn một ngưỡng xác định trước. Việc gán nhãn cho các thành phần liên thông được thực hiện bằng hàm:

```
int VLIB_calcConnectedComponentsMaxBufferSize(
            unsigned short imgWidth,
            unsigned short imgHeight,
            int minBlobArea,
```

```
            int *maxBytesRequired);

int VLIB_initConnectedComponentsList(
            VLIB_CCHandle * handle,
            void * pBuffer,
            int bytesBuffer);

int VLIB_createConnectedComponentsList(
            VLIB_CCHandle * handle,
            unsigned short width,
            unsigned short rowsInImg,
            int * p32BitPackedFGMask,
            int minBlobArea,
            int connected8Flag);

int VLIB_getNumCCs(
            VLIB_CCHandle * handle,
            int * numCCs);

int VLIB_getCCFeatures(
            VLIB_CCHandle * handle,
            VLIB_CC * cc,
            short listIndex);

int VLIB_createCCMap8Bit(
            VLIB_CCHandle * restrict handle,
            unsigned char * restrict pOutMap,
            const unsigned short outCols,
            const unsigned short outRows);
```

5.3.4. *Theo dõi bàn tay*

Như đã trình bày ở Chương 3, đề tài này sử dụng mô hình quan sát dựa vào histogram màu trong hệ màu RGB. Trong đó, một histogram là phân phối xác suất của 4096 bin. Histogram này được tính từ trước khi tiến hành xây dựng mô hình màu da cho khối Skin Segmentation. Khối này cũng sử dụng hình ảnh phân phối xác suất được tính từ trước để tiến hành thuật toán CAMshift.

Vị trí của vùng chứa bàn tay được cập nhật liên tục qua các frame. Nếu kích thước vùng đang theo dõi xuống nhỏ hơn một giá trị xác định trước thì hệ thống sẽ tiến hành tìm kiếm lại vị trí của cử chỉ trên hình.

Trong thực tế, chúng ta làm việc với digital video image do đó phân phối của chúng là rời rạc. Mà CAMshift là thuật toán tìm kiếm gradient của phân phối, do đó cửa sổ tìm kiếm nhỏ nhất phải lớn hơn một để phát hiện được gradient. Và để có thể dịch chuyển điểm giữa của cửa sổ thì nó nên có kích thước lẻ. Khi bắt đầu, chúng ta tính phân phối xác suất màu của toàn bộ frame và dùng zeroth moment để xác định kích thước cửa sổ tìm kiếm và điểm trọng tâm để đặt cửa sổ.

Mục tiêu của chúng ta là theo dõi toàn bộ vật thể vì vậy chúng ta cần mở rộng cửa sổ. Mỗi khi tính được kích thước của cửa sổ mới, ta nhân kết quả này cho 2 để kích thước cửa sổ tăng lên, bao trùm cả các vùng phân phối liên thông. Sau đó, ta làm tròn kết quả về số lẻ gần nhất để cửa sổ tìm kiếm có điểm giữa.

Với phân phôi xác suất màu có giá trị pixel lớn nhất là 255, ta xác đinh kích thước cửa sổ s như sau:

$$s = 2 \times \sqrt{\frac{M_{00}}{256}}$$

5.3.5. *Nhận dạng cử chỉ*

Theo phương pháp do Viola & Jones đề xuất, để tính được Integral Image $I_{m,n}$, ta cần phải tính $R_{m,n}$ ($R_{m,n}$ là tổng các điểm ảnh từ hàng đầu tiên đến hàng thứ m) trước, sau đó tính $I_{m,n}$ theo hai biểu thức đệ quy sau:

$$\begin{aligned} R_{m,n} &= R_{m,n-1} + A_{m,n} \\ I_{m,n} &= I_{m-1,n} + R_{m,n} \end{aligned}$$

Khi thực hiện phương pháp này trên board DSP, cách đơn giản nhất là lưu trữ tất cả dữ liệu cần tính toán trên bộ nhớ ngoài – SDRAM, vì bộ nhớ ngoài thường

có dung lượng lớn và chi phí thấp. Tuy nhiên, việc truy xuất đến bộ nhớ ngoài chậm hơn rất nhiều so với truy xuất bộ nhớ on-chip.

Việc tính Integral Image của dữ liệu chứa ở bộ nhớ ngoài thường được thực hiện theo các bước sau:

- *(a)* Chuyển một phần dữ liệu ảnh vào bộ nhớ on-chip.
- *(b)* Dữ liệu được lấy ra từ on-chip để thực hiện tính toán, sau đó lưu kết quả vào on-chip.
- *(c)* Chuyển kết quả từ bộ nhớ on-chip vào biến output chứa trên bộ nhớ ngoài.

Các bước được lặp lại tuần tự theo *(a)* – *(b)* – *(c)* cho đến khi đã xữ lý hết dữ liệu.

$$a_1 \;\; b_1 \;\; c_1 \;\; a_2 \;\; b_2 \;\; c_2 \;\; a_3 \;\; b_3 \;\; c_3 \;\; \ldots \;\; a_r \;\; b_r \;\; c_r$$

Tuy nhiên, có thể thấy rằng không phải tất cả các bước đều phụ thuộc vào bước xữ lý trước đó. Ví dụ, nếu sử dụng 2 buffer, *(a_2)* có thể bắt đầu chuyển dữ liệu mới vào buffer thứ 2 ngay khi *(a_1)* kết thúc, tức thực hiện song song với *(b_1)*. Trình tự các bước giống như sau:

$$\begin{array}{lccccccccccc} \text{CPU} & & b_1 & & b_2 & & \ldots & b_{r-1} & & & b_r & \\ \text{DMA} & a_1 & a_2 & c_1 & a_3 & c_2 & \ldots & a_r & c_{r-1} & & & c_r \end{array}$$

Bằng cách này, thời gian xữ lý khi thực hiện song song *(a)* và *(b)* bằng với thời gian thực hiện của *(a)* hoặc *(b)*.

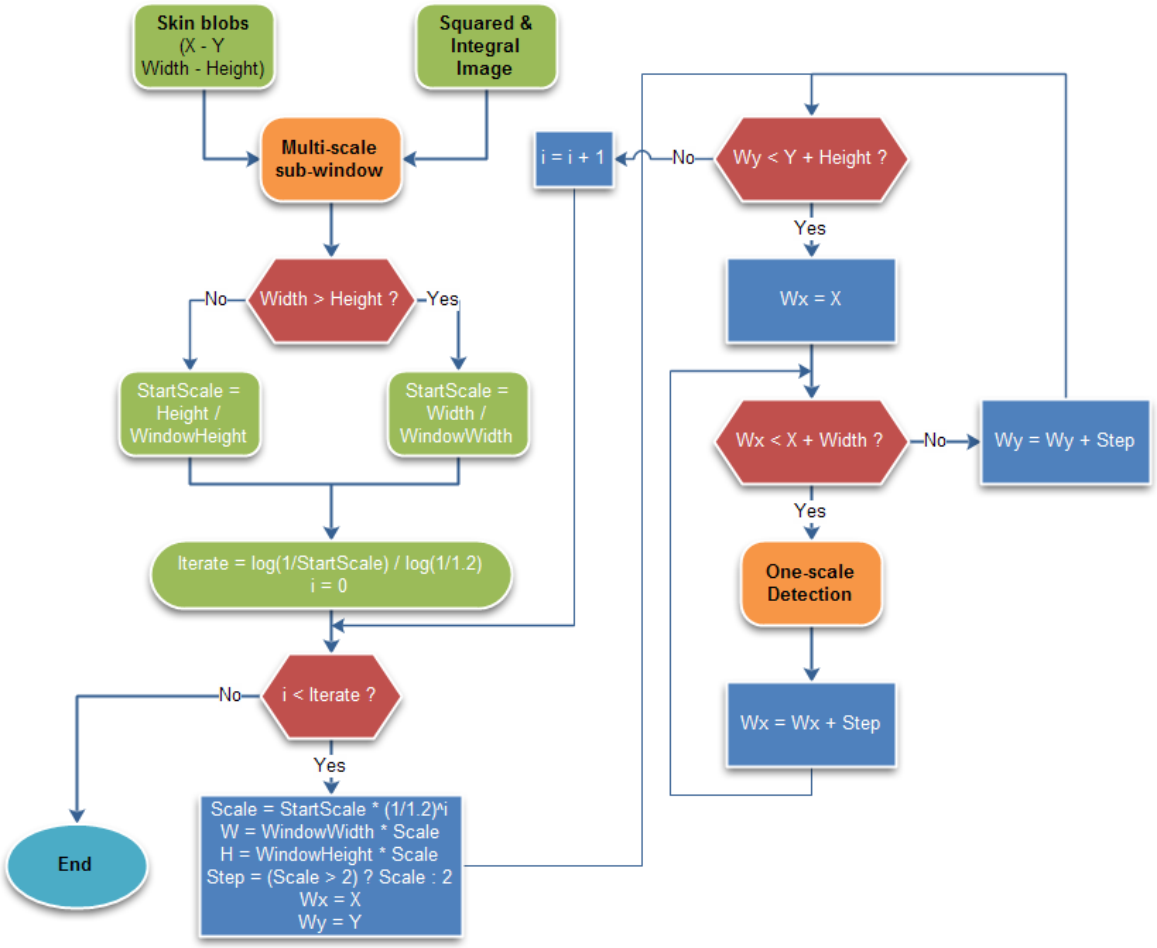
Sau khi đã có được Integral Image, kết hợp cùng vị trí các skin blobs do khối skin segmentation trả về để tiến hành tìm kiếm cử chỉ trên ảnh. Theo tác tìm kiếm được thực hiện như sau:

- Bước 1: Tính tỉ lệ kích thước giữa skin blobs đang xét với kích thước khung tìm kiếm chuẩn (do tập training quy định). Từ đó suy ra số vòng lặp để cử sổ

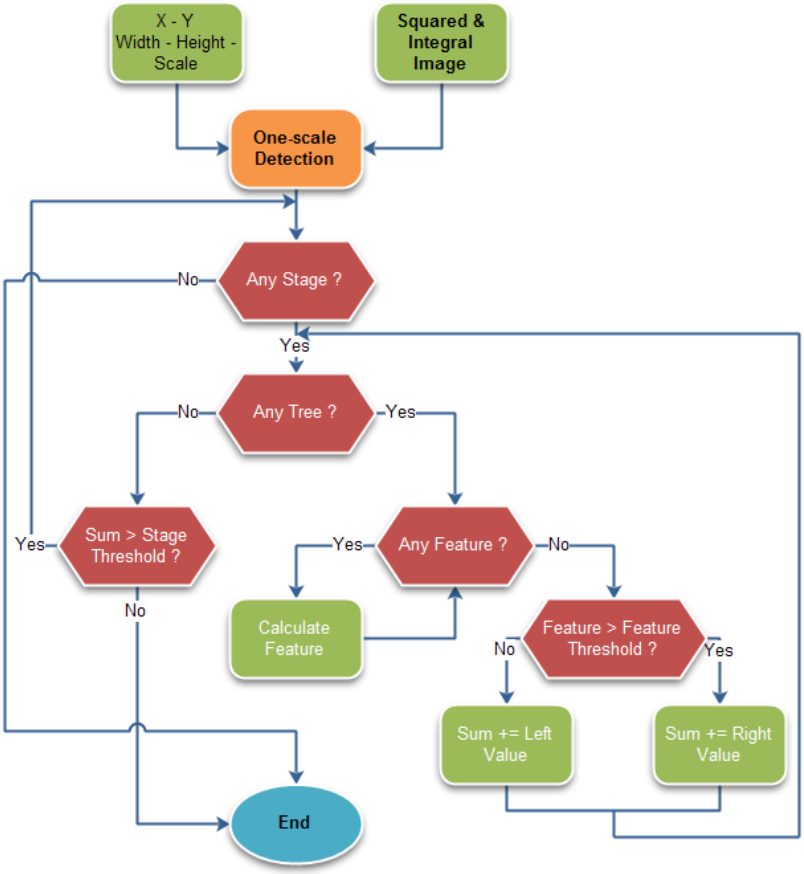
tìm kiếm được mở rộng dần ra (với scale factor là 1.2) cho đến khi nó lớn hơn skin blobs.

- Bước 2: Cho cửa sổ quét lần lượt trên ảnh với bước nhãy tùy thuộc vào tỉ lệ scale hiện tại.
- Bước 3: Tại mỗi vị trí của cửa sổ, căn cứ vào tọa độ hiện tại và tọa độ các feature được trích ra từ tập training để tính giá trị feature cho từng hình chữ nhật. Kết quả này được so sánh với giá trị ngưỡng để chọn một giá trị có sẵn trong tập training. Các giá trị này được cộng dồn cho đến khi hết các feature cần tính của một stage. Kết quả cuối cùng được mang so sánh với ngưỡng của stage đó, nếu lớn hơn thì tiếp tục chuyển sang xét stage tiếp theo.
- Bước 4: Lặp lại bước 3, nếu vị trí cửa sổ hiện tại có thể vượt qua tất cả các stage có trong tập training thì xác định vị trí đó là cử chỉ, nếu không vượt qua được tất cả các stage thì loại bỏ vị trí đó. Quay lại Bước 2.

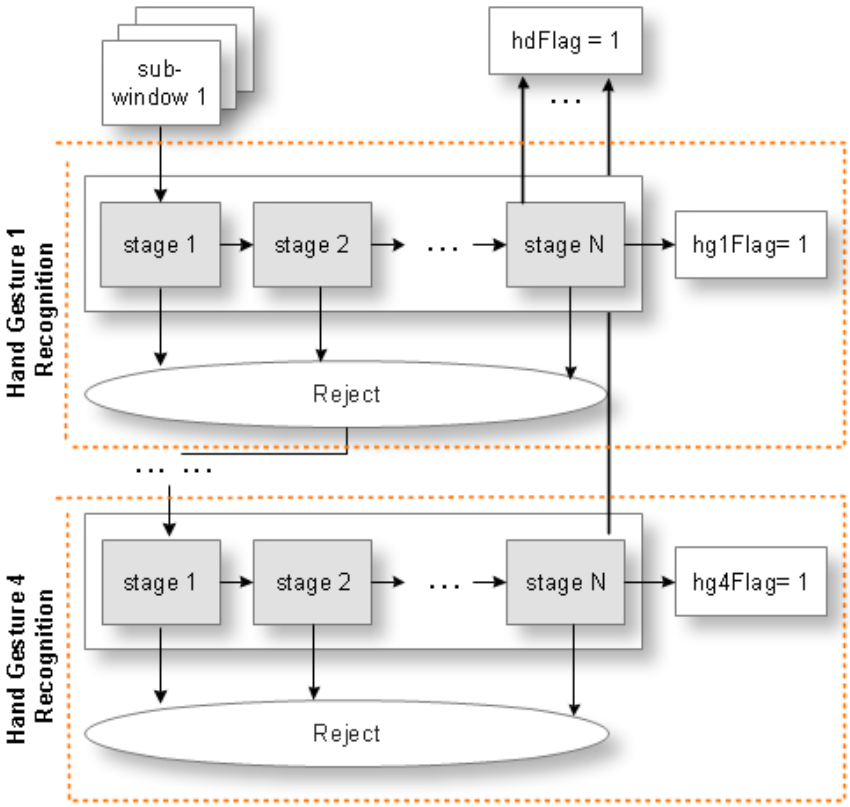
Lưu đồ giải thuật của quá trình trên như sau:



Hình 5.7. Lưu đồ giải thuật multi-scale



Hình 5.8. Lưu đồ giải thuật One-scale



Hình 5.9. Gesture Recognition

5.3.6. *Phân vùng bộ nhớ của board cho dữ liệu và chương trình*

Theo mặc định ban đầu, kit chỉ sử dụng bộ nhớ trong để lưu trữ các dữ liệu cũng như chương trình được nạp vào. Do bộ nhớ này khá nhỏ (256 KB), ta cần phải cấu hình vùng nhớ mà kit có thể sử dụng thông qua file liên kết (Linker File) [7].

Cấu trúc file liên kết:

```
MEMORY
{
        IVECS:  org=0h,      len=0x220
        IRAM:    org=0x00000220,  len=0x0002FDE0 /*internal memory*/
        EDATA:  org=0x80000000,  len=0x0100000
        SDRAM:  org=0x80F00000,   len=0x01000000 /*external memory*/
```

```
        FLASH:   org=0x90000000,   len=0x00020000 */ /*flash memory*/
}
SECTIONS
{
        .EXT_RAM :> SDRAM
        .vectors  :> IRAM
        .text      :> IRAM
        .bss       :> IRAM
        .cinit     :> IRAM
        .stack     :> IRAM
        .sysmem :> IRAM
        .const     :> SDRAM
        .switch   :> IRAM
        .far       :> SDRAM
        .cio       :> IRAM
        .csldata  :> IRAM
}
```

Để ánh xạ một trường bất kỳ sang bộ nhớ ngoài chỉ cần sửa “IRAM” thành “SDRAM”. Ví dụ: .const : > SDRAM

Để ánh xạ một biến bất kỳ trong code (có thể là biến cần kích cỡ lớn mà bộ nhớ trong không chứa được) ta làm như sau:

- Định nghĩa một vùng nhớ trong bộ nhớ ngoài. Ví dụ:

EDATA: org = 0x80000000, len = 0x0100000 định nghĩa phần nhớ ngoài EDATA kích thước 1Mb.

- Trong phần khai bào biến dùng cấu trúc #pragma để ánh xạ biến vào vùng nhớ vừa tạo. Ví dụ với biến var:

#pragma DATA_SECTION (var,“EDATA”);

5.4. **Đánh giá kết quả**

Thực hiện chạy thử nghiệm chương trình trên nền tảng PC và board DSP. Thông sô dùng để so sánh là CPU usage và số fps. Đối với PC, em sử dụng webcam có độ phân giải 320 x 240 và sử dụng thư viện OpenCV để xây dựng chương trình (tức bỏ qua các bước biến đổi floating-point về fixed-point). Kết quả so sánh được liệt kê bên dưới.

Bảng 5-1. Kết quả thử nghiệm

Platform	**CPU Usage**	**FPS**
Intel Core2Duo T5500 1.66 GHz 2 GB RAM Windows 7 x86	≈35%	14 - 16
Intel Core2Duo P7350 2 GHz 4 GB RAM Windows 7 x64	≈37%	15
Intel Core i3-370M 2.4 GHz 4 GB RAM Windows 7 x64	≈30%	22
DM642 EVM	≈80%	60

Hình 5.10. Hệ thống nhận dạng cử chỉ

Đánh giá kết quả nhận dạng 4 cử chỉ trên board DSP với điều kiện như sau:

- Người dùng đứng ở khoảng cách 1 – 2m so với camera.
- Chỉ sử dụng 1 tay.
- Ánh sáng không thay đổi quá lớn trong suốt quá trình test.

Bảng 5-2. Kết quả kiểm tra trên board DSP

Cử chỉ	Độ chính xác
A	97%
B	85%
Five	80%
OK	95%

Hình 5.11. Kết quả nhận dạng 4 cử chỉ

Do hạn chế trong số mẫu training, nên đòi hỏi người dùng phải trình diễn đúng cử chỉ về hình dạng, hướng và mặt của bàn tay thì hệ thống mới nhận dạng được. Ví dụ, nêu để nắm tay nằm ngang thì hệ thống sẽ nhận nhầm là cử chỉ OK.

Hình- Nhận dạng nhầm

Cử chỉ FIVE đòi hỏi phải mở rộng lòng bàn tay để không bị nhận nhầm là cử chỉ B

Hình- Nhận dạng nhầm

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. **Kết luận**

Trên cơ sở tìm hiểu về thuật toán AdaBoost, Haar Feature và mô hình Cascade of Classifiers, 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, đồng thời thực hiện thuật toán này trên board DSP và đã đạt những những kết quả bước đầu. Ngoài sử dụng phương pháp của Viola và Jones cho việc nhân dạng cử chỉ, em còn xây dựng thuật toán phân vùng màu da có khả năng tương thích với người dùng cũng thuật toán theo dõi bàn tay để tăng tốc độ hoạt động, giúp hệ thống đạt hiệu quả tốt nhất. Bên cạnh đó, em còn thực hiện việc tối ưu hóa các thuật toán để nó phù hợp với nền tảng DSP bằng việc thực hiện tính toán một số giá trị từ trước, chuyển các phép toán xử lý trên biến kiểu float về kiểu int.

Bộ nhận dạng cử chỉ với khả năng phân biệt được 4 cử chỉ khác nhau đã phần nào cho thấy khả năng ứng dụng của thuật toán này trên board DSP. Tuy hệ thống vẫn còn một số đòi hỏi về điều kiện ngoại cảnh cũng như yêu cầu về khoảng cách, nhưng nhìn chung hệ thống vẫn hoạt động chính xác và hiệu quả trong nhiều điều kiện môi trường khác nhau.

Tuy nhiên, do hạn chế về mặt thời gian một phần vì thời gian thực hiện không đủ, một phần vì khâu training chiếm quá nhiều thời gian nên đề tài này chỉ dừng lại ở việc xây dựng được bộ phân loại cử chỉ có khả năng phân biệt được 4 cử chỉ khác nhau.

6.2. **Hướng phát triển**

Hướng phát triển trước mắt là tiến hành training thêm nhiều cử chỉ khác để bộ phân loại cử chỉ có thể nhận dạng được nhiều cử chỉ hơn nữa. Tạo các ứng dụng

cụ thể cho hệ thống, sử dụng cử chỉ của người dùng để ra lệnh thực hiện một chức năng nào đó mà cụ thể là ứng dụng trong điều khiển robot.

Hệ thống hiện tại chỉ hoạt động đối với hình chụp chính diện, nó còn khá nhạy cảm với 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 [8]) 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. 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, kết hợp cùng thuật toán theo dõi chuyển động CAMshift.

Với những kết quả đạt được từ đề tài này, để có được một hệ thống thực sự có thể tương tác với con người thông qua cử chỉ thì chúng ta vẫn còn nhiều trở ngại cần phải vượt qua.

TÀI LIỆU THAM KHẢO

[1] P. Viola and M. J. Jones, "Robust real-time face detection", International Journal of Computer Vision, May. 2004, pp.137-154.

[2] P. Viola and M. J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.

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

[4] W. T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition", In *Proc. IEEE Intl, Wkshp. on Automatic Face and Gesture Recognition*, Zurich, June, 1995.

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

[6] M. Soriano, J.B. MartinKauppi, S. Huovinen, M. Laaksonen, "Adaptive skin color modeling using the skin locus for selecting training pixels", Pattern Recognition 36 (3) (2003), pp. 681-690.

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

[8] 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.

[9] V. Vezhnevets, V. Sazonov, and A. Adreeva, "A survey on pixel-based skin color detection techniques", In *Proceedings of the GraphiCon* 2003 (2003), pp. 85-92.

[10] P. Kakumanu, S. Makrogiannis, N. Bourbakis, “A survey of skin-color modeling and detection methods”, Pattern Recognition 2007, pp. 1106 - 1122.

[11] W. Krattenthaler, K. J. Mayer, and M. Zeiller, “Point Correlation: A Reduced-Cost Template Matching Technique”, In *Proc. ICIP*, 1994, pp.208-212.

[12] A. K. Jain, Y. Zhong, and S. Lakshmanan, “Object Matching Using Deformable Template”, IEEE Trans, Pattern Analysis and Machine Intelligence, vol. 18, no. 3, Mar. 1996, pp. 267-278.

[13] D. Morrell, “Filtering of Stochastic Processes”, EEE 581 – Lecture 2, 2003.

[14] V. A. Oliveira, A. Conci, “Skin Detection using HSV color space”, Computation Institute – Universidade Federal Fluminense – UFF – Niteri, Brazil.

[15] H. Francke, J. Ruiz del Solar, R. Verschae, “Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning”, In *Pacific Rim Symposium on Image Video and Technology,* 2007, pp. 533 – 547.

[16] C. Manresa, J. Varona, R. Mas, “Real-Time Hand Tracking and Gesture Recognition for Human-Computer Interaction”, 2000.