PHẦN MỞ ĐẦU

Trong thời đại ngày nay công nghệ thông tin hầu như đã thâm nhập vào toàn bộ các lĩnh vực đời sống xã hội. Xã hội càng phát triển thì nhu cầu về công nghệ thông tin ngày càng cao, do vậy dữ liệu số hầu như không còn xa lạ đối với mỗi người chúng ta. Trong mọi lĩnh vực các ứng dụng công nghệ thông tin đã trợ giúp con người rất nhiều.

Hiện nay, thông tin hình ảnh đóng vai trò rất quan trọng trong trao đổi thông tin, bởi phần lớn các thông tin mà con người thu nhận được đều thông qua thị giác. Trong các lĩnh vực công nghệ thông tin thì lĩnh vực giám sát tự động đã và đang thu hút được nhiều sự quan tâm của các nhóm nghiên cứu trong và ngoài nước. Cùng với sự phát triển của sức mạnh máy tính, các hệ thống giám sát tự động ngày càng tinh vi và hiện đại đã trợ giúp con người rất nhiều trong việc bảo vệ an ninh, giám sát giao thông, v.v

Ở nước ta hiện nay, lĩnh vực giám sát tự động cũng đã có những bước phát triển đáng kể. Tuy nhiên, nó chỉ mới dựa trên nền tảng phần cứng và cũng chưa được áp dụng nhiều trong thực tế. Việc giải quyết bài toán này theo hướng tiếp cận sử dụng phần mềm chưa được quan tâm phát triển. Do vậy em lựa chọn đề tài: "Tìm hiểu bài toán phát hiện đối tượng chuyển động". Trong khuôn khổ khóa luận này em tập trung trình bày về các kỹ thuật trừ ảnh và ứng dụng các kỹ thuật này để giải quyết một bài toán quan trọng và then chốt trong lĩnh vực giám sát tự động đó là bài toán phát hiện tự động đối tượng chuyển động thông qua web camera.

Nội dung chính của khóa luận bao gồm các phần sau: phần mở đầu, phần kết luận, ba chương nội dung, cụ thể:

- ***Chương 1:*** Khái quát về xử lý ảnh và bài toán phát hiện đối tượng chuyển động
- ***Chương 2:*** Phát hiện đối tượng chuyển động dựa vào kỹ thuật trừ ảnh
- ***Chương 3:*** Chương trình thử nghiệm

Chương 1:
KHÁI QUÁT VỀ XỬ LÝ ẢNH VÀ PHÁT HIỆN ĐỐI TƯỢNG

1.1. KHÁI QUÁT VỀ XỬ LÝ ẢNH

1.1.1. Xử lý ảnh là gì?

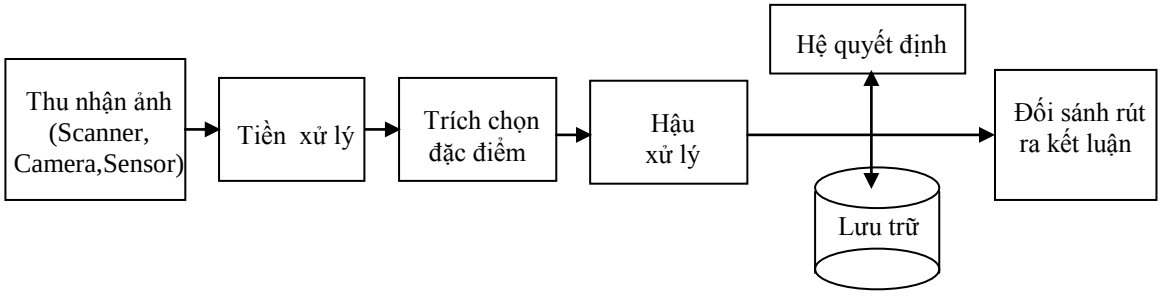
Con người thu nhận thông tin qua các giác quan, trong đó thị giác đóng vai trò quan trọng nhất. Những năm trở lại đây với sự phát triển của phần cứng máy tính, xử lý ảnh và đồ hoạ đó phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ hoạ đóng một vai trò quan trọng trong tương tác người máy.

Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh "tốt hơn" hoặc một kết luận.

Hình 1.1. Quá trình xử lý ảnh

Ảnh có thể xem là tập hợp các điểm ảnh và mỗi điểm ảnh được xem như là đặc trưng cường độ sáng hay một dấu hiệu nào đó tại một vị trí nào đó của đối tượng trong không gian và nó có thể xem như một hàm n biến $P(c_1, c_2,..., c_n)$. Do đó, ảnh trong xử lý ảnh có thể xem như ảnh n chiều.

Sơ đồ tổng quát của một hệ thống xử lý ảnh:



Hình 1.2. Các bước cơ bản trong một hệ thống xử lý ảnh

1.1.2. Các vấn đề cơ bản trong xử lý ảnh

1.1.2.1. Một số khái niệm

**** Ảnh và điểm ảnh:***

Điểm ảnh được xem như là dấu hiệu hay cường độ sáng tại 1 toạ độ trong không gian của đối tượng và ảnh được xem như là 1 tập hợp các điểm ảnh.

**** Mức xám, màu***

Là số các giá trị có thể có của các điểm ảnh của ảnh

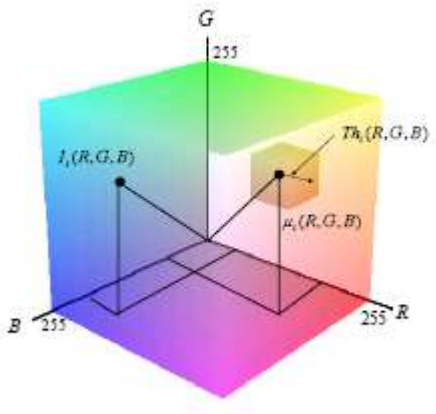
1.1.2.2 Thu nhận ảnh

Như ta đã biết, ảnh trong thực tế là ảnh liên tục cả về không gian lẫn giá trị độ sáng. Muốn xử lý ảnh trên máy tính ta cần phải số hóa ảnh, tức là đưa ảnh từ thực tế vào máy tính. Để đưa ảnh vào trong máy tính chúng ta có thể dùng các thiết bị thu nhận như: camera cộng với bộ chuyển đổi tương tự số AD (Analog to Digital) hoặc máy quét chuyên dụng.

Các thiết bị thu nhận có thể cho ảnh trắng đen B/W với mật độ từ 400 đến 600 dpi. Với ảnh B/W mức màu z là 0 hoặc 1. Với ảnh đa cấp xám, mức xám biến thiên từ 0 đến 255.

1.1.2.3. Biểu diễn ảnh

Sau quá trình số hóa ta sẽ thu được một ma trận tương ứng với ảnh cần xét, mỗi phần tử của ma trận tương ứng với một điểm ảnh. Các điểm này thường được đặc trưng bởi tọa độ màu RGB tương ứng với nó trong hệ tọa độ màu cơ bản sau:



Hình 1.3: Hệ tọa độ màu RGB

Về mặt toán học ta có thể xem ảnh như là một hàm hai biến f(x,y) với x,y là các biến tọa độ. Giá trị số tại điểm (x,y) tương ứng với giá trị xám hoặc độ sáng của ảnh. ảnh có thể được biểu diễn theo một trong hai mô hình sau đây:

- Mô hình Raster: là mô hình biểu diễn ảnh phổ biến nhất hiện nay. ảnh được biểu diễn dưới dạng ma trận các điểm ảnh. Tùy theo nhu cầu thực tế mà mỗi điểm ảnh có thể được biểu diễn bởi một hay nhiều bit. Mô hình Raster phù hợp cho việc thu nhận và hiển thị ảnh.
- Mô hình vector: bên cạnh mục đích tiết kiệm không gian lưu trữ, dễ dàng hiển thị và in ấn, các ảnh biểu diễn theo mô hình vector còn có ưu điểm cho phép dễ dàng lựa trọn, sao chép, di chuyển, tìm kiếm, v.v… Trong mô hình này người ta sử dụng hướng vectot của các điểm ảnh lân cận để mã hóa và tái tạo ảnh ban đầu. Các ảnh vector được thu nhậnh trực tiếp từ các thiết bị số hóa như Digitalize hoặc chuyển đổi từ các ảnh Raster thông qua các chương trình vector hóa.

Khi xử lý các ảnh Raster chúng ta có thể quan tâm đến mối quan hệ trong vùng lân cận của các điểm ảnh. Các điểm ảnh có thể xếp hàng trên một lưới hình vuông, hoặc lưới lục giác hoặc theo một cách hoàn toàn ngẫu nhiên với nhau. Cách sắp xếp theo lưới hình vuông được quan tâm nhiều nhất và có hai khái niệm sau: điểm 4 – láng giềng và điểm 8 – láng giềng. Hình vẽ 1.4 dưới đây mô tả các khái niệm này:

8 láng giềng *4 láng giềng*

Hình 1.4: Điểm 4 láng giềng và 8 láng giềng

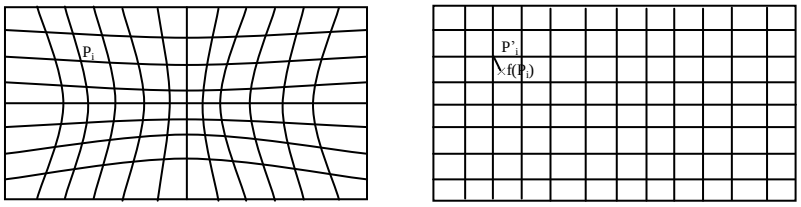
1.1.2.4. Khử nhiễu

Có 2 loại nhiễu cơ bản trong quá trình thu nhận ảnh

- Nhiều hệ thống: là nhiễu có quy luật có thể khử bằng các phép biến đổi
- Nhiễu ngẫu nhiên: vết bẩn không rõ nguyên nhân → khắc phục bằng các phép lọc

1.1.2.5. Nắn chỉnh biến dạng

Ảnh thu nhận thường bị biến dạng do các thiết bị quang học và điện tử.



Ảnh thu nhận **Ảnh mong muốn**

Hình 1.5. Ảnh thu nhận và ảnh mong muốn

Để khắc phục người ta sử dụng các phép chiếu, các phép chiếu thường được xây dựng trên tập các điểm điều khiển.

Giả sử (P_i, P_i') $i = \overline{1,n}$ có n các tập điều khiển

Tìm hàm f: $P_i \mapsto f(P_i)$ sao cho

$$\sum_{i=1}^{n} \left| \left| f(P_i) - P_i' \right| \right|^2 \to \min$$

Giả sử ảnh bị biến đổi chỉ bao gồm: Tịnh tiến, quay, tỷ lệ, biến dạng bậc nhất tuyến tính. Khi đó hàm f có dạng:

$f(x, y) = (a_1x + b_1y + c_1, a_2x + b_2y + c_2)$

Ta có:

$$\phi = \sum_{i=1}^{n} (f(Pi) - Pi')^2 = \sum_{i=1}^{n} \left| a_1x_i + b_1y_i + c_1 - x_i' \right|^2 + \left| a_2x_i + b_2y_i + c_2 - y_i' \right|^2$$

Để cho $\phi \to \min$

$$\begin{cases} \dfrac{\partial \phi}{\partial a_1} = 0 \\ \dfrac{\partial \phi}{\partial b_1} = 0 \\ \dfrac{\partial \phi}{\partial c_1} = 0 \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^{n} a_1x_i^2 + \sum_{i=1}^{n} b_1x_iy_i + \sum_{i=1}^{n} c_1x_i = \sum_{i=1}^{n} x_ix_i' \\ \sum_{i=1}^{n} a_1x_iy_i + \sum_{i=1}^{n} b_1y_i^2 + \sum_{i=1}^{n} c_1y_i = \sum_{i=1}^{n} y_ix_i' \\ \sum_{i=1}^{n} a_1x_i + \sum_{i=1}^{n} b_1y_i + nc_1 = \sum_{i=1}^{n} x_i' \end{cases}$$

Giải hệ phương trình tuyến tính tìm được a_1, b_1, c_1

Tương tự tìm được a_2, b_2, c_2

$\Rightarrow$ Xác định được hàm f

1.1.2.6. Chỉnh mức xám

Nhằm khắc phục tính không đồng đều của hệ thống gây ra. Thông thường có 2 hướng tiếp cận:

- Giảm số mức xám: Thực hiện bằng cách nhóm các mức xám gần nhau thành một bó. Trường hợp chỉ có 2 mức xám thì chính là chuyển về ảnh đen trắng. Ứng dụng: In ảnh màu ra máy in đen trắng.
- Tăng số mức xám: Thực hiện nội suy ra các mức xám trung gian bằng kỹ thuật nội suy. Kỹ thuật này nhằm tăng cường độ mịn cho ảnh

1.1.2.7. Phân tích ảnh

Là khâu quan trọng trong quá trình xử lý ảnh để tiến tới hiểu ảnh. Trong phân tích ảnh việc trích chọn đặc điểm là một bước quan trọng. Các đặc điểm của đối tượng được trích chọn tuỳ theo mục đích nhận dạng trong quá trình xử lý ảnh. Có thể nêu ra một số đặc điểm của ảnh sau đây:

Đặc điểm không gian: Phân bố mức xám, phân bố xác suất, biên độ, điểm uốn v.v..

Đặc điểm biến đổi: Các đặc điểm loại này được trích chọn bằng việc thực hiện lọc vùng (zonal filtering). Các bộ vùng được gọi là "mặt nạ đặc điểm" (feature mask) thường là các khe hẹp với hình dạng khác nhau (chữ nhật, tam giác, cung tròn v.v..)

Đặc điểm biên và đường biên: Đặc trưng cho đường biên của đối tượng và do vậy rất hữu ích trong việc trích trọn các thuộc tính bất biến được dùng khi nhận dạng đối tượng. Các đặc điểm này có thể được trích chọn nhờ toán tử gradient, toán tử la bàn, toán tử Laplace, toán tử "chéo không" (zero crossing) v.v..

Việc trích chọn hiệu quả các đặc điểm giúp cho việc nhận dạng các đối tượng ảnh chính xác, với tốc độ tính toán cao và dung lượng nhớ lưu trữ
giảm xuống.

1.1.2.8. Nhận dạng

Nhận dạng tự động (automatic recognition), mô tả đối tượng, phân loại và phân nhóm các mẫu là những vấn đề quan trọng trong thị giác máy, được ứng dụng trong nhiều ngành khoa học khác nhau. Tuy nhiên, một câu hỏi đặt ra là: mẫu (pattern) là gì? Watanabe, một trong những người đi đầu trong lĩnh vực này đã định nghĩa: "Ngược lại với hỗn loạn (chaos), mẫu là một thực thể (entity), được xác định một cách ang áng

(vaguely defined) và có thể gán cho nó một tên gọi nào đó". Ví dụ mẫu có thể là ảnh của vân tay, ảnh của một vật nào đó được chụp, một chữ viết, khuôn mặt người hoặc một ký đồ tín hiệu tiếng nói. Khi biết một mẫu nào đó, để nhận dạng hoặc phân loại mẫu đó có thể:

Hoặc **phân loại có mẫu** (supervised classification), chẳng hạn phân tích phân biệt (discriminant analyis), trong đó mẫu đầu vào được định danh như một thành phần của một lớp đã xác định.

Hoặc **phân loại không có mẫu** (unsupervised classification hay clustering) trong đó các mẫu được gán vào các lớp khác nhau dựa trên một tiêu chuẩn đồng dạng nào đó. Các lớp này cho đến thời điểm phân loại vẫn chưa biết hay chưa được định danh.

Hệ thống nhận dạng tự động bao gồm ba khâu tương ứng với ba giai đoạn chủ yếu sau đây:

1^o. Thu nhận dữ liệu và tiền xử lý.

2^o. Biểu diễn dữ liệu.

3^o. Nhận dạng, ra quyết định.

Bốn cách tiếp cận khác nhau trong lý thuyết nhận dạng là:

1^o. Đối sánh mẫu dựa trên các đặc trưng được trích chọn.

2^o. Phân loại thống kê.

3^o. Đối sánh cấu trúc.

4^o. Phân loại dựa trên mạng nơ-ron nhân tạo.

Trong các ứng dụng rõ ràng là không thể chỉ dùng có một cách tiếp cận đơn lẻ để phân loại "tối ưu" do vậy cần sử dụng cùng một lúc nhiều phương pháp và cách tiếp cận khác nhau. Do vậy, các phương thức phân loại tổ hợp hay được sử dụng khi nhận dạng và nay đã có những kết quả có triển vọng dựa trên thiết kế các hệ thống lai (hybrid system) bao gồm nhiều mô hình kết hợp.

Việc giải quyết bài toán nhận dạng trong những ứng dụng mới, nảy sinh trong cuộc sống không chỉ tạo ra những thách thức về thuật giải, mà còn đặt ra những yêu cầu về tốc độ tính toán. Đặc điểm chung của tất cả những ứng dụng đó là những đặc điểm đặc trưng cần thiết thường là nhiều, không thể do chuyên gia đề xuất, mà phải được trích chọn dựa trên các thủ tục phân tích dữ liệu.

1.1.2.9. Nén ảnh

Nhằm giảm thiểu không gian lưu trữ. Thường được tiến hành theo cả hai cách khuynh hướng là nén có bảo toàn và không bảo toàn thông tin. Nén không bảo toàn thì thường có khả năng nén cao hơn nhưng khả năng phục hồi thì kém hơn. Trên cơ sở hai khuynh hướng, có 4 cách tiếp cận cơ bản trong nén ảnh:

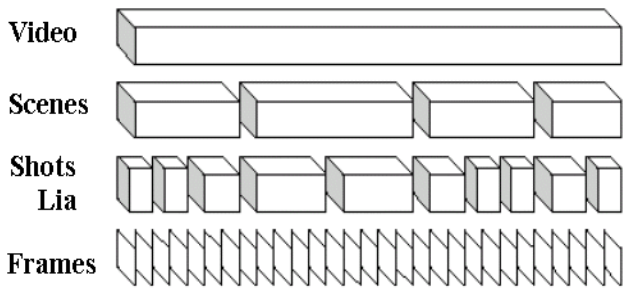
- Nén ảnh thống kê: Kỹ thuật nén này dựa vào việc thống kê tần xuất xuất hiện của giá trị các điểm ảnh, trên cơ sở đó mà có chiến lược mã hóa thích hợp. Một ví dụ điển hình cho kỹ thuật mã hóa này là *.TIF
- Nén ảnh không gian: Kỹ thuật này dựa vào vị trí không gian của các điểm ảnh để tiến hành mã hóa. Kỹ thuật lợi dụng sự giống nhau của các điểm ảnh trong các vùng gần nhau. Ví dụ cho kỹ thuật này là mã nén *.PCX
- Nén ảnh sử dụng phép biến đổi: Đây là kỹ thuật tiếp cận theo hướng nén không bảo toàn và do vậy, kỹ thuật thướng nến hiệu quả hơn. *.JPG chính là tiếp cận theo kỹ thuật nén này.
- Nén ảnh Fractal: Sử dụng tính chất Fractal của các đối tượng ảnh, thể hiện sự lặp lại của các chi tiết. Kỹ thuật nén sẽ tính toán để chỉ cần lưu trữ phần gốc ảnh và quy luật sinh ra ảnh theo nguyên lý Fractal

1.2.VIDEO VÀ BÀI TOÁN PHÁT HIỆN ĐỐI TƯỢNG CHUYỂN ĐỘNG

1.2.1. Một số khái niệm

Video là tập hợp các khung hình (frames), mỗi khung hình là một ảnh. Shot (lia) là một đơn vị cơ sở của video. Một lia là một đơn vị vật lý của dòng video, gồm chuỗi các khung hình liên tiếp, không thể chia nhỏ hơn.

Scene (cảnh) là các đơn vị logic của dòng video, một cảnh gồm các lia liên quan về không gian và liền kề về thời gian, cùng mô tả một nội dung ngữ nghĩa hoặc một tình tiết. Cấu trúc phân cấp của Video được mô tả trong hình vẽ 1.6:



Hình 1.6: Cấu trúc phân cấp của video

Khi phim được chiếu các khung hình lần lượt được hiển thị ở một tốc độ nhất định. Tốc độ thường thấy ở các định dạng video là 25 hình/s hoặc 30 hình/s. Như vậy trong một giờ video số khung hình tương ứng là 108000 hoặc 9000.

Phân đoạn video là quá trình phân tích và chia nội dung hình ảnh video thành các đơn vị cơ sở gọi là các lia (shot). Việc lấy mẫu chính là chọn gần đúng một khung hình video đại diện cho mỗi lia (hoặc nhiều hơn tùy theo độ phức tạp của nội dung hình ảnh của lia), và được gọi là các khung khoá. Khung khoá là khung hình đại diện mô tả nội dung chính của shot. Quá trình phân đoạn dữ liệu video tiến hành phân tích, phát hiện sự chuyển đổi từ lia này sang lia khác hay chính là sự phát hiện ranh giới giữa các lia (đó chính là sự khác nhau giữa các khung hình liền kề). Hình vẽ 1.7 sau đây mô tả sự chuyển đổi giữa các lia

Hình 1.7: minh hoạ về việc chuyển đổi giữa các lia

Trong hình vẽ trên sự chuyển đổi lia xảy ra giữa khung hình thứ 3 và thứ 4

1.2.2. Một số thuộc tính đặc trưng của video

Video có 4 đặc tính chung là: màu (color), kết cấu (texture), hình dáng (shape), chuyển động (motion). Sau đây chúng ta sẽ lần lượt tìm hiểu từng đặc tính.

1.2.3. Chuyển động (Motion)

Motion là một thuộc tính quan trọng của video. Thông tin về chuyển động có thể được sinh ra bằng các kỹ thuật ghép khối hoặc luồng ánh sáng. Các đặc trưng chuyển động như mômen của trường chuyển động, biểu đồ chuyển động hoặc các tham số chuyển động toàn cục có thể được trích chọn tử vector chuyển động. Các đặc trưng mức cao phản ánh di chuyển camera như quét camera (pan), nghiêng (tilt), phóng to (zoom in),thu nhỏ (zoom out) cũng có thể được trích chọn.

1.2.4. Bài toán phát hiện đối tượng chuyển động

Sự phát triển của công nghệ thông tin đẩy nhanh sự phát triển của các lĩnh vực xã hội khác. Với sự phát triển của phần cứng cả về phương diện thu nhận và hiển thị cũng như tốc độ xử lý đã mở ra nhiều hướng cho sự phát triển phần mềm. Trong số đó phải kể đến lĩnh vực giám sát tự động.

Một trong những bài toán quan trọng và then chốt trong lĩnh vực giám sát tự động đó là bài toán phát hiện đối tượng chuyển động. Đối với bài toán phát hiện đối tượng chuyển động thường có hai cách tiếp cận chính sau đây:

- Dựa hoàn toàn vào phần cứng
- Dựa vào các kỹ thuật xử lý ảnh trên cơ sở xử lý các hình ảnh thu được, phân tích và kết luận xem có đối tượng đột nhập hay không

Ở nước ta hiện nay, việc giải quyết bài toán phát hiện đối tượng chuyển động còn chủ yếu dựa vào phần cứng và cũng chưa được áp dụng nhiều trong thực tế. Trong chương tiếp theo chúng ta sẽ tìm hiểu chi tiết từng cách tiếp cận để giải quyết bài toán này.

Chương 2: PHÁT HIỆN ĐỐI TƯỢNG CHUYỂN ĐỘNG DỰA VÀO KỸ THUẬT TRỪ ẢNH

2.1. K Ỹ THUẬT TRỪ ẢNH DỰA VÀO ĐIỂM ẢNH

Phương pháp đơn giản nhất để trừ hai khung hình là tính giá trị biểu diễn sự chênh lệch tổng cộng về cường độ của tất cả các điểm ảnh tương ứng trên hai khung hình:

$$D(f_1, f_2) = \frac{1}{X \times Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} |f_1(x,y) - f_2(x,y)|$$

So sánh giá trị tìm được với ngưỡng chuyển cảnh T_b để xác định xem có chuyển cảnh hay không.

Kỹ thuật trừ ảnh dựa vào điểm ảnh rất đơn giản. Nhược điểm lớn nhất của kỹ thuật này la không phân biệt được sự thay đổi lớn trong một vùng ảnh nhỏ và thay đổi nhỏ trong một vùng ảnh lớn. Nói chung tất cả các kỹ thuật trừ giá trị điểm ảnh đều nhạy với nhiễu và các di chuyển camera. Có thể cải tiến kỹ thuật này bằng cách đếm tổng số điểm ảnh có thay đổi lớn hơn một ngưỡng nào đó và so sánh giá trị tính được với một ngưỡng khác để phát hiện chuyển cảnh

$$DP(x,y) = \begin{cases} 1 & \text{, Nếu } |f_1(x,y) - f_2(x,y)| > T_1 \\ 0 & \text{, ngược lại} \end{cases}$$

$$D(f_1, f_2) = \frac{1}{X \times Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} DP(x,y)$$

Nếu tỷ lệ số điểm ảnh thay đổi $D(f_1,f_2)$ lớn hơn ngưỡng T_1 thì đã có sự chuyển cảnh do cắt. Tuy các thay đổi không liên quan trong khung hình đã được loại bỏ bớt nhưng hướng tiếp cận này vẫn nhạy với các di chuyển camera và đối tượng. Chẳng hạn, khi camera quay theo đối tượng, rất nhiều điểm ảnh được cho là thay đổi, dù cho có ít điểm ảnh dịch chuyển. Có thể giảm tác động này bằng cách sử dụng một bộ lọc trơn: trước khi so sánh, mỗi điểm ảnh được thay thế bằng giá trị trung bình của các điểm ảnh lân cận.

Một nhược điểm khác của kỹ thuật trừ điểm ảnh là độ nhạy của điểm ảnh với việc chiếu sáng. Khi đó người ta điều chỉnh độ sai khác giá trị điểm ảnh bằng cách chia nó cho cường độ của điểm ảnh trên khung hình thứ hai . Hampapur gọi ảnh thu được từ độ chênh lệch hiệu chỉnh là ảnh chromatic:

$$D(f_1, f_2) = \frac{1}{X \times Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \frac{|f_1(x,y) - f_2(x,y)|}{f_2(x,y)}$$

Phương pháp trừ giá trị điểm ảnh cơ bản là tính toán từ các giá trị điểm ảnh, nhưng có thể mở rộng đối với các ảnh màu. Ví dụ với ảnh màu RGB, ta tính tổng có trọng số các sai khác của ba giá trị Red, Green, Blue của các điểm ảnh.

$$D(f_1, f_2) = \sum_{x=0}^{X} \sum_{y=0}^{Y} \sum_{i \in \{R,G,B\}} w_i \mid f_{1i}(x,y) - f_{2i}(x,y) \mid$$

2.2. TRỪ ẢNH PHÂN KHỐI

Trái ngược với hướng tiếp cận sử dụng các đặc tính toàn cục của cả khung hình, hướng tiếp cận phân khối sử dụng các đặc tính cục bộ nhằm tăng tính độc lập với các di chuyển của camera và đối tượng. Mỗi khung hình được chia thành b khối. Các khối trên khung hình f_1 được so sánh với các khối tương ứng trên khung hình f_2. Về cơ bản, độ chênh lệch giữa hai khung hình được tính như sau:

$$D(f_1, f_2) = \sum_{k=1}^{b} C_k . DP(f_1, f_2, k)$$

Trong đó C_k là hệ số cho trước, $DP(f_1,f_2,k)$ là độ chênh lệch giữa hai khối thứ k của hai khung hình f_1 và f_2.

Kasturi đưa ra công thức :

$$\lambda_k = \frac{\left[\frac{\sigma_{1k} + \sigma_{2k}}{2} + \left(\frac{\mu_{1k} - \mu_{2k}}{2} \right)^2 \right]^2}{\sigma_{1k} . \sigma_{2k}}$$

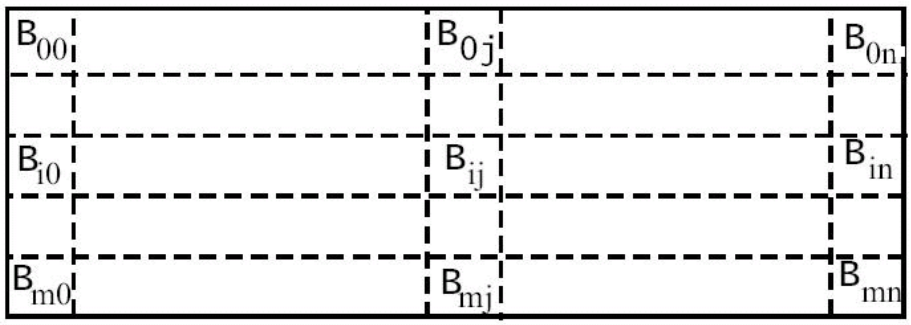
Trong đó μ_{1k}, μ_{2k} là giá trị cường độ trung bình của khối thứ k σ_{1k}, σ_{2k} là độ chênh lệch tương ứng với hai khối đó.

$$DP(f_1, f_2, k) = \begin{cases} 1 & \text{Nếu } \lambda_k > T_1 \\ 0 & \text{Nếu ngược lại} \end{cases}$$

Một cắt cảnh xảy ra khi số các khối thay đổi đủ lớn, nghĩa là $D(f_1,f_2) > T_2$ và $C_k=1$ cho tất cả các khối.

Một hướng tiếp cận khác với kỹ thuật trừ ảnh phân khối do Shaharay đưa ra. ông chia khung hình thành 12 miền và tìm miền thích hợp nhất cho mỗi miền ở khung hình kia. Độ chênh lệch tính bằng kỹ thuật trừ ảnh dựa vào điểm ảnh của từng miền được sắp xếp. Tổng có trọng số của các chênh lệch đã sắp xếp cho ta kết quả D cuối cùng.

Xiong phát triển phương pháp trừ ảnh, gọi là so sánh thực, phát hiện chuyển cảnh do ngắt chỉ bằng việc so sánh một phần của ảnh. Phương pháp này chỉ ra rằng, sai sót mắc phải hoàn toàn có thể bỏ qua nếu ít hơn một nửa số các cửa sổ cơ sở (các ô vuông chồng nhau) đều được kiểm tra. Với giả thiết rằng, trong trường hợp thay đổi nhiều nhất giữa hai khung hình thì kích thước các cửa sổ được chọn đủ lớn để bất biến với các thay đổi không làm vỡ và đủ nhỏ để có thể chứa thông tin về không gian nhiều chừng nào có thể. Các cửa sổ cơ sở được so sánh và tính độ chênh lệch mức xám hoặc giá trị màu của các điểm ảnh. Khi giá trị chênh lệch lớn hơn một ngưỡng nào đó thì xem như miền đang xét đã thay đổi. Khi số miền thay đổi lớn hơn một ngưỡng khác thì sự chuyển cảnh do ngắt đã xảy ra. Thực nghiệm cho thấy rằng hướng tiếp cận này cho tốc độ nhanh hơn phương pháp so sánh từng cặp điểm



Hình 2.1: Các cửa sổ cơ sở trong thuật toán so sánh thực

Một số nghiên cứu đã mở rộng ý tưởng lấy mẫu theo không gian thành lấy mẫu theo không gian và thời gian. Thuật toán có sử dụng bước nhảy phát hiện cả chuyển cảnh đột ngột và chuyển cảnh dần dần. Thuật toán này đi so sánh hai khung hình i và j, ở đó j = i + step. Nếu không có sự thay đổi đáng kể nào, thì chuyển sang so sánh các khung hình cách nửa bước nhảy, nghĩa là so sánh hai khung hình i + step/2 và j + step/2. Ngược lại, tìm kiếm nhị phân được dùng để định vị chuyển cảnh. Nếu i và j liên tiếp nhau và sự chênh lệch của hai khung hình lớn hơn ngưỡng thì đó là chuyển cảnh đột ngột do ngắt. Nếu không, sử dụng thuật toán trừ ảnh dựa trên việc phát hiện cạnh để phát hiện chuyển cảnh dần dần. Hiển nhiên, thuật toán này phụ thuộc vào bước nhảy step: bước nhảy lớn thì tăng hiệu quả nhưng tăng khả năng sai sót, bước nhảy nhỏ quá sẽ bỏ qua những chuyển cảnh dần dần. Thuật toán này có độ nhạy rất cao với sự di chuyển của dối tượng và sự di chuyển của camera.

2.3. PHƯƠNG PHÁP BIỂU ĐỒ

Một bước xa hơn để giảm ảnh hưởng của sự chuyển camera và đối tượng là thực hiện trừ ảnh dựa vào biểu đồ. Biểu đồ mô tả sự phân bố giá trị điểm ảnh của khung hình. ý tưởng của cách tiếp cận này là các ảnh có nền không đổi và đối tượng không đổi sẽ có chênh lệch ít trong biểu đồ. Hơn nữa biểu đồ bất biến với việc quay ảnh và thay đổi ít khi góc nhìn thay đổi.

Có thể dùng biểu đồ màu hoặc biểu đồ mức xám để tính sự sai khác giữa hai khung hình. Biểu đồ màu (mức xám) của khung hình i là một vector G chiều H_i = ($H_i(1)$, $H_i(2)$, … , $H_i(G)$). Trong đó G là số màu (mức xám), $H_i(j)$ là số điểm ảnh của khung hình i có màu (mức xám) j. Phương pháp trừ ảnh dựa trên biểu đồ có thể sử dụng biểu đồ toàn cục hoặc biểu đồ cục bộ. Biểu đồ toàn cục là biểu đồ biểu diễn sự phân bố giá trị màu(mức xám) của toàn bộ khung hình. Còn biểu đồ cục bộ chỉ mô tả sự phân bố của một phần nào đó của khung hình mà thôi.

2.3.1. Biểu đồ toàn cục

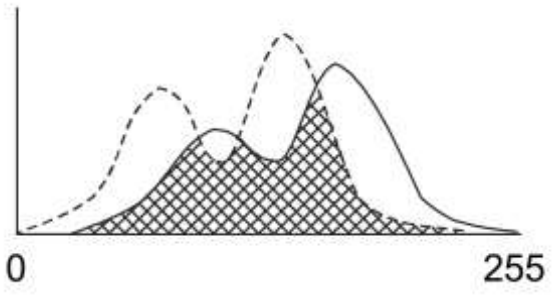
Phương pháp đơn giản nhất là tính tổng sự sai khác các cột của biểu đồ.

$$D(f_1, f_2) = \sum_{k=0}^{G} | H_1(k) - H_2(k) |$$

Có thể sử dụng thêm trọng số nếu có một số màu (mức xám) quan trọng hơn với mục tiêu so sánh.

$$D(f_1, f_2) = \sum_{k=0}^{G} w(k) | H_1(k) - H_2(k) |$$

Trong đó W(k) là trọng số ứng với giá trị màu (mức xám) k.



Hình 2.2: so sánh biểu đồ giữa hai ảnh

Cách thứ ba là sử dụng phần giao nhau của hai biểu đồ. Vùng biểu đồ chồng nhau, phần gạch chéo trong hình 2.2, cho biết độ tương tự về nội dung hai ảnh có thể được định nghĩa như sau:

$$S(f_1, f_2) = \sum_{k=0}^{G} \min\left(H_1(k), H_2(k)\right)$$

Độ tương tự còn có thể định nghĩa như sau:

$$S(f_1, f_2) = \frac{\sum_{k=0}^{G} \min(H_1(k), H_2(k))}{\sum_{k=0}^{G} \max(H_1(k), H_2(k))}$$

Như vậy, dựa vào phần giao nhau của hai biểu đồ, có thể tính độ chênh lệch biểu đồ hai khung hình theo công thức:

$$D(f_1, f_2) = 1 - S(f_1, f_2) = 1 - \frac{\sum_{k=0}^{G} \min(H_1(k), H_2(k))}{\sum_{k=0}^{G} \max(H_1(k), H_2(k))}$$

Một hướng tiếp cận sử dụng biểu đồ khác là xem xét biểu đồ là vector và sử dụng tích vô hướng của chúng:

$$D(f_1, f_2) = 1 - \frac{\vec{h_1} \bullet \vec{h_2}}{\left\|\vec{h_1}\right\| . \left\|\vec{h_2}\right\|}$$

Để biểu diện sự phân bố của màu với ảnh 24 bit, phải tạo biểu đồ với 256^3 cột, mỗi cột ứng với một bộ ba RGB có thể có. Có thể dùng thuật toán nhanh tính toán với biểu đồ, nhưng ta thường áp dụng giải pháp thô: dùng biểu đồ với số cột ít hơn. Yihong dùng giải pháp biểu đồ 8 mức RGB kết quả là biểu đồ có $2^8 = 256$ cột.

$$D(f_1, f_2) = w_R D_R + w_G D_G + w_B D_B$$

Trong đó, D_G, D_B, D_R là chênh lệch biểu đồ màu thành phần green, blue, red. Jyrki sử dụng các trọng số như sau:

$$D(f_1, f_2) = 0.2125 D_R + 0.7154 D_G + 0.0721 D_B$$

Nói chung, người thường chỉ dùng 20 cột có số điểm ảnh nhiều nhất để so sánh. Còn có một cách khách làm giảm số cột của biểu đồ là chỉ dùng 2 bit cao nhất cho

cường độ mỗi màu thành phần để mã hoá màu của điểm ảnh. Như vậy việc so sánh biểu đồ chỉ cần thực hiện với 64 cột. Sawhney đề xuất rằng 256 màu là đủ biểu diễn sự phân bố màu của các cảnh. Novak và Safer thì chỉ chia các cột biểu đồ thành hai loại "full" và "Empty" để ước lượng thuộc tính bề mặt và điều kiện ánh sáng cho các đối tượng đơn.

Chênh lệch biểu đồ có thể được tính bằng công thức Kolmogorov – Sminov như sau:

$$D_{K-S}(f_1, f_2) = \max_j \left| \sum_{k=0}^{j} H_1(k) - H_2(k) \right|$$

Nói cách khác, chênh lệch tích luỹ lớn nhất giữa hai biểu đồ phân bố cho đến j được tính toán. Giá trị $D_{K\text{-}S}$ lớn xác định ranh giới chuyển cảnh. Để nhấn mạnh sự sai khác giữa hai khung hình khi chuyển cảnh qua cắt cứng, một số tác giả đề xuất thuật toán χ^2 để so sánh biểu đồ màu:

$$D(f_1, f_2) = \sum_{k=0}^{G} \frac{|H_1(k) - H_2(k)|^2}{H_2(k)}$$

Thuật toán χ^2 không những nhấn mạnh độ sai khác giữa hai khung hình qua cắt cứng, nó còn nhấn mạnh độ sai khác giữa hai khung hình khi di chuyển camera hay đối tượng.

Yakimovsky đưa ra công thức:

$$D(f_1, f_2) = \frac{\sigma_0^{2\,\overline{m+n}}}{\sigma_1^{2\,\overline{m}}\ \sigma_2^{2\,\overline{n}}}$$

Trong đó : σ_0^2 *là phần chung giữa hai biểu đồ*

$\sigma_1^2, \sigma_{02}^2$ *là phần khác nhau của hai biểu đồ.*

m,n là số cột tương ứng của hai biểu đồ.

Công thức này có thể áp dụng cho cả trường hợp hai biểu đồ có số cột khác nhau.

2.3.2. Biểu đồ cục bộ

Như đã đề cập, phương pháp trừ ảnh dựa vào biểu đồ là phương pháp ít chịu ảnh hưởng của nhiễu và di chuyển đối tượng. Tuy vậy cũng có một số trở ngại. Đầu tiên, biểu đồ chỉ mô tả sự phân bổ các giá trị màu hay mức xám mà không bao hàm bất cứ thông tin nào về không gian. Hai ảnh có cùng biểu đồ màu nhưng có nội dung rất khác nhau. Trở ngại khác là rất có thể các vùng ảnh nhỏ khi thay đổi sẽ gây chú ý nhưng lại không có vai trò gì trong biểu đồ và do đó có thể bị bỏ qua khi thực hiện trừ ảnh. Để giải quyết vấn đề đó chúng ta sẽ kết hợp trừ ảnh dựa vào biểu đồ với kỹ thuật trừ ảnh phân khối. Trừ ảnh phân khối quan tâm đến thông tin về không gian. Về cơ bản phương pháp này tốt hơn việc so sánh từng cặp điểm ảnh, nhưng nó vẫn chịu tác động của sự di chuyển camera và di chuyển của đối tượng. Bằng cách kết hợp hai ý tưởng, chúng ta vừa có thể giảm được sự tác động của các di chuyển camera và đối tượng, vừa sử dụng thông tin về không gian ảnh, và do đó cho kết quả phân đoạn tốt hơn.

ý tưởng là chúng ta sẽ chia khung hình thành b khối, đánh số từ 1 đến b. So sánh biểu đồ của các khối tương ứng rồi tính tổng chênh lệch để có kết quả trừ ảnh cuối cùng

$$D(f_1, f_2) = \sum_{k=1}^{b} DP(f_1, f_2, k)$$

$$DP(f_1, f_2) = \sum_{j=0}^{G} | H_1(j,k) - H_2(j,k) |$$

Trong đó H(j,k) là giá trị biểu đồ tại màu (mức xám) j ứng với khối thứ k

Hướng tiếp cận khác trong kỹ thuật trừ ảnh dựa vào biểu đồ cục bộ được Swanberg đưa ra. Sự chênh lệch DP(f_1,f_2,k) giữa các khối được tính bằng cách so sánh biểu đồ màu RGB sử dụng công thức sau:

$$DP(f_1, f_2, k) = \sum_{c \in \{R,G,B\}} \sum_{j=0}^{G} \frac{(H_1^{\,c}(j,k) - H_2^{\,c}(j,k))^2}{H_2^{\,c}(j,k)}$$

2.4. PHƯƠNG PHÁP THỐNG KÊ

Phương pháp sai khác thống kê dựa vào phương pháp trừ giá trị điểm ảnh, nhưng thay vì tính tổng sự sai khác của tất cả các điểm ảnh, ta chia ảnh thành các miền rồi so sánh các đại lượng thống kê điểm ảnh của miền đó. Một cách là ta sử dụng thống kê tỉ lệ số điểm ảnh thay đổi trên toàn bộ khung hình. Ta sử dụng một giá trị d là ngưỡng sai khác được tính giữa hai điểm ảnh tương ứng. Gọi S là tập các điểm ảnh có sai khác lớn hơn g d:

$$S = \{(x, y) \mid |f_1(x, y) - f_2(x, y)| > d\}$$

Độ sai khác giữa hai khung hình được tính bằng tỷ lệ các điểm ảnh có độ chênh lệch lớn hơn d.

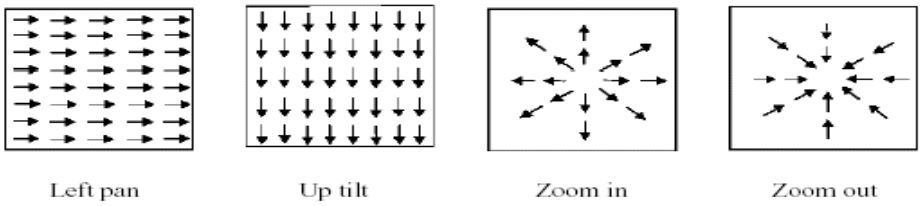
$$D(f1, f2) = \frac{S.count}{X * Y}$$

Cách khác, chúng ta có thể sử dụng các đại lượng thống kê cho từng miền, như biểu đồ chẳng hạn.

2.4.1. Đặc trưng là vector chuyển động

Trong các đoạn video, người ta thường thấy các hiệu ứng do chuyển động của camera, như pan (quét), zoom (zoom in – phóng tó, zoom out – thu nhỏ), tilt (nghiêng). Để nâng cao hiệu quả phân đoạn, kỹ thuật trừ ảnh dựa vào đặc trưng là vector chuyển động được sử dụng để phát hiện các hiệu ứng kiểu này.

Các mẫu vector chuyển động thu được từ các di chuyển camera khác nhau được thể hiện trên hình 2.3 sau đây



Hình 2.3: Mẫu vector cho các di chuyển camera.

Một số nhà nghiên cứu đã sử dụng vector chuyển động xác định từ việc ghép khối để phát hiện xem shot được phóng to, thu nhỏ hay quét camera. Một số nghiên cứu khác lại sử dụng vector chuyển động như là một phần của việc trừ ảnh phân khối dựa vào điểm ảnh để quyết định xem có phải có một lượng lớn các di chuyển đối tượng hay camera trong shot.

2.4.2. Đặc trưng là cạnh

Một hướng tiếp cận khác cho việc phân loại và phát hiện chuyển cảnh là sự phát hiện sự xuất hiện các cạnh (biên cường độ) trong một khung hình, chúng cách các cạnh trong khung hình trước một khoảng nhất định. Kỹ thuật này không chỉ phát hiện mà còn có thể phân loại được các loại chuyển cảnh: cắt cứng, chồng mờ, fade, wipe. Phương pháp này tỏ ra chính xác hơn phương pháp dựa vào biểu đồ và độ nhạy với chuyển động thấp hơn nhiều so với gam màu.

Zabih, Miller và Mai[14] không so sánh biểu đồ màu , gam màu. Thuật toán của họ dựa trên kỹ thuật phát hiện cạnh. Họ căn chỉnh các khung hình để giảm các tác động của sự di chuyển camera và so sánh số lượng vị trí các cạnh trong các ảnh đã phát hiện cạnh. Tỉ lệ phần trăm của các cạnh vào và ra giữa hai khung hình liên tiếp được tính toán. Biên của shot được phát hiện bằng cách tìm tỷ lệ phần trăm thay đổi cạnh lớn.

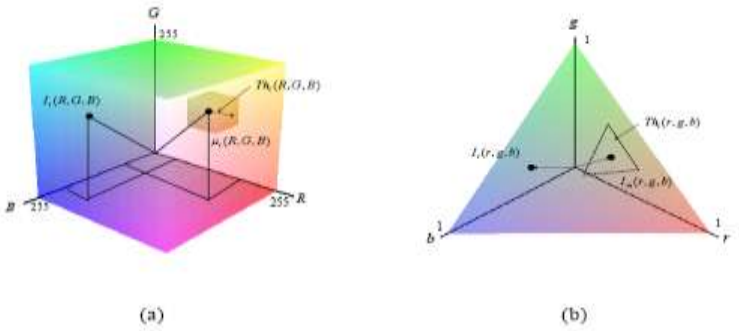
2.5. KỸ THUẬT TRỪ NỀN (Background subtraction)

Kỹ thuật trừ nền thông thường thực hiện việc trừ ảnh hiện tại cho ảnh tham chiếu. Mặc dù vậy một số yếu tố (color, motion, block, v.v…) được sử dụng trong một số nghiên cứu, phương pháp đề xuất ở đây tận dụng các đặc tính giá trị màu của điểm ảnh trong hai hệ tọa độ màu RGB và RGB chuẩn hóa. Nó cần thiết để xác định các giá trị ngưỡng tối ưu trong kỹ thuật trừ nền. Trong mục này chúng ta sẽ giải thích các thuộc tính của mỗi không gian màu và việc xác định các giá trị ngưỡng tối ưu cho điểm ảnh như thế nào. ở đây, chúng ta cho thấy việc sử dụng giá trị ngưỡng xác định như thế nào trong thuật toán đề xuất.

2.5.1. Không gian màu (Color space)

Hệ thống thị giác của con người nhận dạng màu sắc của các đối tượng dựa trên độ kết tủa màu sắc (chromaticity) và độ chói (luminance). Do đó, chúng ta sử dụng hai hệ tọa độ màu quen thuộc là RGB và RGB chuẩn hóa. Trong hệ tọa độ màu RGB, mỗi điểm ảnh đều có các phần tử chromaticity và luminance. Do đó, trong không gian màu này hai màu được coi là khác nhau nếu hoặc là chromaticity hoặc là luminance khác nhau. Do đó, khi mà kỹ thuật trừ nền được thực hiện trong hệ tọa độ màu RGB thì bóng của đối tượng, hoặc vùng sáng được nhận dạng như là đối tượng thật thậm chí chúng chỉ khác nhau về luminance nhưng hầu hết có cùng chromaticity. Việc loại bỏ các ảnh hưởng của ánh sáng sẽ khó nếu như chúng ta chỉ sử dụng hệ tọa độ RGB. Vấn

đề này đã làm nảy sinh nhiều nghiên cứu về các mô hình màu. Biểu diễn riêng biệt chromaticity và luminance trong một mô hình màu có khả năng xác định mỗi điểm ảnh một cách chính xác có thể. Tuy nhiên, nó yêu cầu việc tính toán phức tạp và chi phí đắt. Trong hệ tọa độ màu RGB chuẩn hóa, mỗi điểm ảnh chỉ có một phần tử chromaticity. Trong hệ tọa độ màu này, chúng ta có thể loại bỏ được hiện tượng giao thoa ánh sáng bởi vì chúng chỉ có luminance là khác với cảnh nền. Trong hình vẽ dưới đây, hệ tọa độ màu RGB là một khối lập phương ba chiều, còn RGB chuẩn hóa là một tam giác hai chiều:



(a) (b)

Hình 2.4: Các không gian màu và phân lớp điểm ảnh của nó.

(a) hệ tọa độ RGB, (b) hệ tọa độ RGB chuẩn hóa

2.5.2. Mô hình nền (Background modeling)

Trong phương pháp đề xuất, chúng ta quan tâm đến các ảnh nền trong hệ tọa độ màu RGB và RGB chuẩn hóa. Chúng ta có thể xác định giá trị trung bình và độ lệch tiêu chuẩn của các kênh màu (R,G,B) tại điểm ảnh i trong ảnh tham chiếu. Mỗi điểm ảnh của ảnh tham chiếu được mô hình hóa như sau:

$$Rf_i = < \mu_i, \sigma_i, \bar{\mu}_i, \bar{\sigma}_i >, \qquad I_i = \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}, \qquad \bar{I}_i = \begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix} = \frac{1}{|I_i|} \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix} \qquad (1)$$

Trong đó:

Rf_i là bộ dữ liệu của ảnh tham chiếu

$\mu_i, \bar{\mu}_i$ là vector giá trị trung bình của các kênh màu tại điểm ảnh i trong hệ tọa độ màu RGB và RGB chuẩn hóa

$\sigma_i, \bar{\sigma}_i$ là vector độ lệch tiêu chuẩn của các kênh màu tại điểm ảnh i trong hệ tọa độ màu RGB và RGB chuẩn hóa.

Các phương trình sau đây cho thấy cách tính toán vector giá trị trung bình và độ lệch tiêu chuẩn tại điểm ảnh i trong không gian màu RGB và RGB chuẩn hóa:

$$\mu_i = \frac{1}{N}\sum_{j=0}^{N-1} I_j \,,\ \bar{\mu}_i = \frac{1}{N}\sum_{j=0}^{N-1} \bar{I}_j \tag{2}$$

$$\sigma_i = \frac{1}{\sqrt{N}}\left|I_i - \mu_i\right|,\ \ \bar{\sigma}_i = \frac{1}{\sqrt{N}}\left|I_i - \mu_i\right| \tag{3}$$

Trong đó: N là số ảnh đang xét

2.5.3. Lựa chọn ngưỡng (Threshold selection)

Khi chúng ta quan sát sự thay đổi của các điểm ảnh trong ảnh của cảnh nền tĩnh, chúng được mô hình hóa một cách đơn giản như là một phân phối Gaussian. Từ quan sát này, giá trị ngưỡng của điểm ảnh i được ánh xạ bởi hàm của độ lệch tiêu chuẩn của điểm ảnh này

$$Th_i = \alpha.\sigma_i\,,\quad \bar{Th}_i = \beta.\bar{\sigma}_i \tag{4}$$

Th_i và $\bar{Th}_i$ là giá trị ngưỡng của điểm ảnh i trong các hệ tọa độ màu RGB và RGB chuẩn hóa. Các hằng số α, β cho trước, nó xác định độ tin cậy. Ví dụ với $\alpha = \beta = 1$ thì độ tin cậy là 68%. Nếu $\alpha = \beta = 2$ thì độ tin cậy là 95%. Ngoài ra α, β còn xác định miền giá trị của ngưỡng. Chúng ta có thể tính được giá trị ngưỡng tại điểm ảnh i một cách đơn giản bằng cách sử dụng $\sigma_i, \bar{\sigma}_i$ và các hằng số α và β. Hầu hết các kỹ thuật trừ nền đánh địa chỉ việc xác định giá trị ngưỡng, có một vài phương pháp lại cho thấy cách sử dụng các giá trị ngưỡng định trước trong thao tác trừ ảnh. Trong phương pháp đề xuất, chúng ta thấy được hiệu quả của việc sử dụng các giá trị ngưỡng định trước để trừ đối tượng cho cảnh nền. Các phương trình (5), (6) là hàm quyết định, nó so sánh sự khác nhau giữa các kênh màu của điểm ảnh i và các giá trị ngưỡng định trước trong hệ tọa độ màu RGB và RGB chuẩn hóa

$$F_i = \sum_{c=1}^{3} u\left(\left|D_{i,c}\right| - \left|Th_{i,c}\right|\right) \tag{5}$$

$$f_i = \sum_{c=1}^{3} u\left(\left|\bar{D}_{i,c}\right| - \left|\bar{Th}_{i,c}\right|\right) \tag{6}$$

$$D_i = I_i - \mu_i \qquad \bar{D}_i = \bar{I}_i - \bar{\mu}_i \tag{7}$$

Trong đó: $|.| = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$. $F_i (0 \le F_i \le 3)$ và $f_i (0 \le f_i \le 3)$ là các hàm quyết định mô tả điểm ảnh i trong mỗi không gian màu và c số lượng kênh màu. ở đây, u là một hàm đơn vị bước nhảy và nó bằng 0 hoặc 1. D_i và $\bar{D}_i$ là các vector sai

khác giữa ảnh hiện tại và ảnh tham chiếu tại điểm ảnh i trong hệ toạn độ màu RGB và RGB chuẩn hóa. Do đó, nếu $D_i > Th_i$ thì nó là 1. Ngược lại, nó bằng 0

Sử dụng các phương trình (5), (6), chúng ta có thể xác định điểm ảnh i như sau:

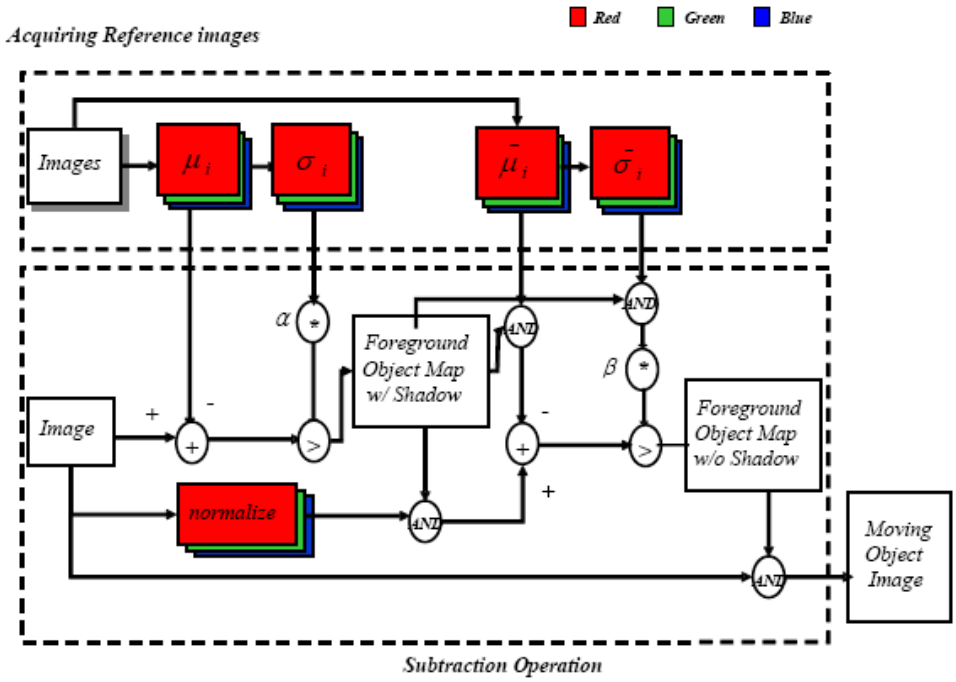
$$Obj_i = \begin{cases} B: & F_i = c_1 \\ H^s: & 0 \le F_i < c_1 \\ B^s: & f_i = c_2 \\ H: & 0 \le f_i < c_2 \end{cases} \quad (3\text{-}8)$$

Trong đó B là ảnh nền và B^s là ảnh nền ứng với bóng. H^s là ảnh phân đoạn đối tượng ứng ứng với bóng, H là ảnh phân đoạn đối tượng không có bóng. C_1, c_2 là số lượng các kênh màu. Trong hệ tọa độ RGB và RGB chuẩn hóa, thì khoản biến thiên của chúng là $0 \le c_1 \le 3$ và $0 \le c_2 \le 3$

Phương pháp đã đề xuất sử dụng phương trình (3-8) để phân biệt một cách hính xác H và B bằng cách điều chỉnh c_1, c_2. Ví dụ, nếu chúng ta xem xét tất cả các kênh màu trong mỗi hệ tọa độ thì $c_1 = c_2 = 3$. Điều này chỉ ra rằng tất cả các kênh màu của điểm ảnh i thỏa mãn $D_i > Th_i$. Hoặc nếu chúng ta chỉ có hai kênh màu thì $c_1 = c_2 = 2$. Trong trường hợp chúng ta đang xét các đặc tính của mỗi hệ tọa độ màu, ta có thể xác định được c_1, c_2.

2.5.4. Thao tác trừ (Subtraction operation)

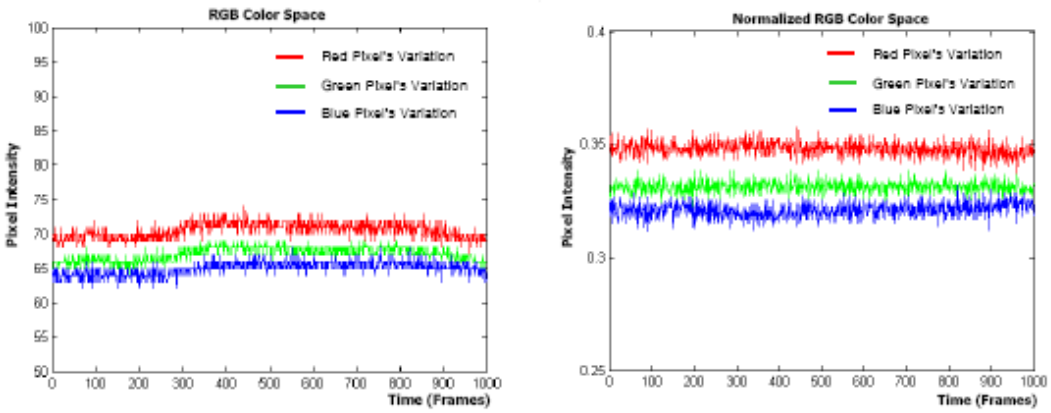
Thao tác trừ nền được mô tả như trong hình vẽ dưới đây:



Hình 2.5:Sơ đồ thuật giải kỹ thuật trừ nền

Trong đó: μ_i và σ_i là vectơ giá trị trung bình và độ lệch tiêu chuẩn của các kênh màu của điểm ảnh i trong hệ tọa độ màu RGB. $\overline{\mu}_i$ và $\overline{\sigma}_i$ là vectơ giá trị trung bình và độ lệch tiêu chuẩn của các kênh màu tại điểm ảnh i trong hệ tọa độ màu RGB chuẩn hóa. α và β là các hằng số ngưỡng xác định trong mỗi không gian màu. Dấu '-' biểu diễn thao tác trừ ảnh hiện tại cho ảnh nền. Dấu '>' so sánh sự khác nhau.

Phương pháp chúng ta đang xét cũng gần giống với kỹ thuật trừ nền thông thường và có hai bước. Bước một là xâu chuỗi nền và bước tiếp theo là trừ nền đã được xâu chuỗi. Tuy nhiên, như chúng ta thấy trong hình 2.6, mỗi một bước lại có hai bước nhỏ trong thuật toán đề xuất. Trong bước đầu tiên, chúng ta xâu chuỗi các ảnh nền và tạo ảnh tham chiếu trong hệ tọa độ màu RGB và RGB chuẩn hóa. Trong bước thứ hai, chúng ta thực hiện việc trừ ảnh hiện tại cho ảnh tham chiếu trong mỗi hệ tọa độ màu. Trong bước xâu chuỗi nền, chúng ta mô hình hóa nền sử dụng phương trình (1). Tiếp đó chúng ta xác định ngưỡng tại điểm ảnh i thông qua phương trình (4). Sauk hi mô hình nền được thực hiện trong mỗi không gian màu, ta phân biệt đối tượng với bóng từ cảnh nền trong hệ tọa độ RGB sử dụng phương trình (5). Tiếp đó, chúng ta lượng tử hóa ảnh kết quả thành một ảnh nhị phân. Như chúng ta thấy trong hình vẽ 2.6, ảnh nhị phân được tạo ra được sử dụng như là một ảnh mặt nạ trong hệ tọa độ RGB chuẩn hóa. Khi chúng ta áp dụng ảnh mặt nạ vào ảnh tham chiếu và ảnh hiện tại trong hệ tọa độ RGB chuẩn hóa tại cùng một thời điểm, chúng ta sẽ loại bỏ được bóng của đối tượng một cách đơn giản bởi vì bóng chỉ có ảnh hưởng trên luminance. Thông qua hai bước này, chúng ta có thể dễ dàng đạt được ảnh của đối tượng (H) không có bóng. Hình vẽ 2.7 dưới đây cho thấy sự thay đổi của điểm ảnh i theo thời gian trong các hệ tọa độ màu RGB và RGB chuẩn hóa. Sự biến đổi của điểm ảnh i theo thời gian là khác nhau trong mỗi kênh màu



(a) (b)

Hình 2.6:Sự biến đổi của điểm ảnh i trong mỗi không gian màu

(a)hệ tọa độ RGB, (b)hệ tọa độ RGB chuẩn hóa

Trong hệ tọa độ RGB : Red $\sigma = 1.1436$, Green $\sigma = 0.9665$, Blue $\sigma = 0.9734$. Trong hệ tọa độ RGB chuẩn hóa: Red $\sigma = 0.0031$, Green $\sigma = 0.0025$, Blue $\sigma = 0.003$

Chương 3:
CHƯƠNG TRÌNH THỬ NGHIỆM

3.1. KỸ THUẬT BẮT GIỮ HÌNH ẢNH QUA CAMERA

Môi trường Windows đã cung cấp cho ta hai cách lập trình với video: cách thứ nhất dùng VFW (Video For Windows) API. Cách thứ hai dùng lớp AVICap của Windows. VFW API hỗ trợ cho quá trình bắt giữ (capture) video từ webcam. AVICap cung cấp cách tiếp cận dựa trên thông điệp đơn giản, cho phép chúng ta truy cập, điều khiển luồng dữ liệu audio, video. Một ứng dụng xây dựng trên AVICap có một số khả năng như:

- Thu dữ liệu audio, video vào một file có đuôi mở rộng là avi
- Kết nối và hủy kết nối với các thiết bị vào trong thời gian thực thi
- Xem trực tiếp dữ liệu video từ thiết bị đầu vào theo phương pháp preview hoặc overlay
- Chỉ định tốc độ thu dữ liệu
- Hiển thị các dialogbox cho phép người sử dụng điều khiển dữ liệu video đầu vào
- Sao chép các hình ảnh và palette lên clipboard
- Thu một ảnh đơn và lưu dưới dạng DIB

AVICap hỗ trợ các khả năng thu dữ liệu dưới dạng một ảnh tĩnh đơn hay theo dạng stream với nhiều frame ảnh. Các frame ảnh có thể cách nhau một khoảng thời gian xác định hay tùy ý. Việc thu các stream ảnh cũng có thể không cần lưu trên đĩa mà có thể được sử dụng trực tiếp từ buffer trên bộ nhớ, điều này cho phép lập trình viên mềm dẻo trong việ xử lý trong các ứng dụng khác nhau. Ngoài ra lớp AVICap cho phép ứng dụng chỉ định các hàm callback được sử dụng trong quá trình bắt giữ

- ***Status Callback:*** được gọi khi có sự thay đổi trạng thái của quá trình thu video
- ***Error Callback:*** được gọi khi có lỗi xảy ra trong quá trình thu video
- ***Frame Callback:*** được gọi trước khi một frame ảnh được preview
- ***Video Stream Callback:*** được gọi khi thu được các frame ảnh trong quá trình streaming video
- ***Audio Stream Callback:*** được gọi khi dữ liệu audio được ghi đầy trong buffer

Khi xây dựng một ứng dụng video dùng lớp AVICap, các ứng dụng thường được thực hiện theo các trình tự sau:

- Tạo một capture window
- Kết nối vào một capture driver
- Liệt kê các capture driver đã cài đặt trong hệ thống
- Lấy thông tin về khả năng của một capture driver
- Lấy thông tin trạng thái của một capture window
- Trình bày dialogbox để thiết lập các thông số video
- Lấy cũng như thiết lập các thông số của video format
- Cho phép preview video
- Cho phép overlay video
- Đặt tên cho capture file
- Cấp phát trước vùng nhớ trên đĩa cho capture file
- Định dạng audio capture
- Thay đổi các thông số video capture
- Thu dữ liệu
- Thêm các chuỗi thông tin vào capture file
- Thêm các hàm callback vào ứng dụng

Tiếp theo, chúng ta sẽ tìm hiểu một số hàm AVICap Windows thường dùng:

- *Hàm tạo capture window*

```
hWndC = capCreateCaptureWindow(
        (LPSTR) “My Capture Window”, // tên cửa sổ
        WS_CHILD | WS_VISIBLE,   // kiểu cửa sổ
        0, 0, 160, 120,          // vị trí cửa sổ
        (HWND) hwndParent,
        (int) nID );
```

- *Kết nối vào capture driver*

```
fOK = capDriverConnect(hWndC,0);
```

- *Hủy bỏ kết nối với capture driver*

```
capDriverDisconnect(hWndC);
```

- *Kích hoạt chế độ Preview video:*

Đầu tiên chúng ta cần phải đặt tốc độ bắt giữ hình ảnh, sau đó kích hoạt chế độ Preview video. Thí dụ dưới đây thiết lập tốc độ hiển thị frame ở chế độ preview là 66 miliseconds mỗi frame(tức là khoảng 15 fps) và thiết lập chế độ preview cho capture window

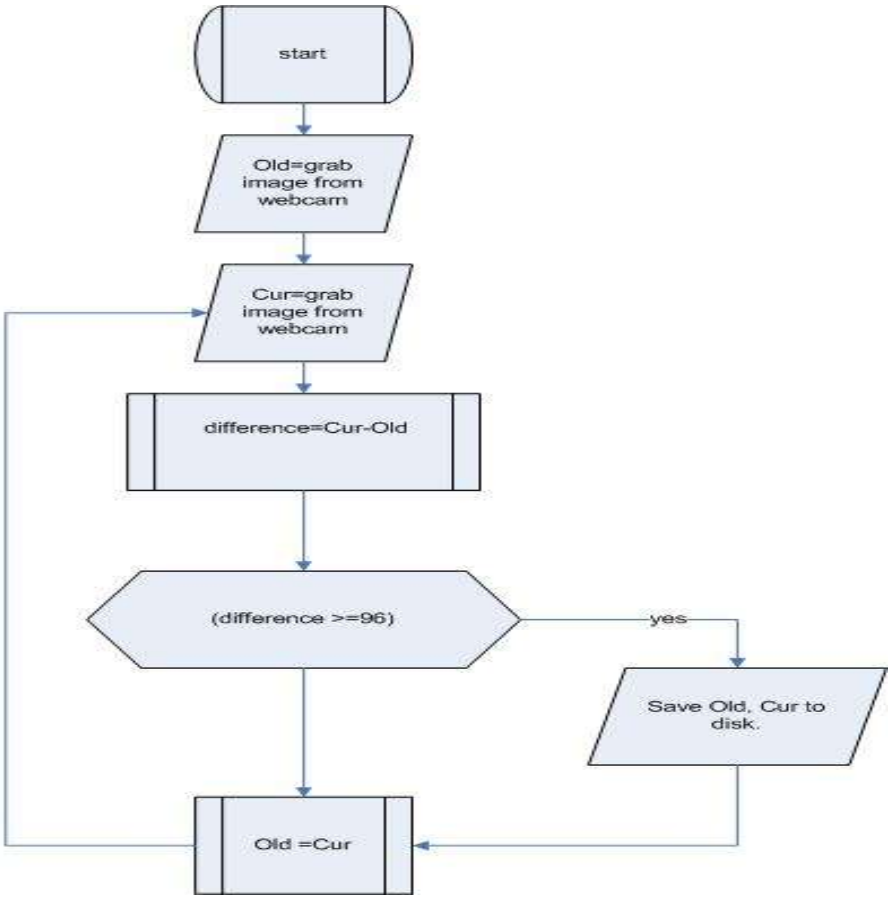
```
CapPreviewRate(hWndC,66);
CapPreview(hWndC, TRUE);
CapPreview(hWndC, FALSE);
```

3.2. PHÂN TÍCH YÊU CẦU BÀI TOÁN VÀ THUẬT GIẢI ĐỀ XUẤT

Như ta đã biết, bài toán phát hiện đối tượng chuyển động là một trong những bài toán quan trọng và then chốt trong lĩnh vực giám sát tự động. Yêu cầu của bài toán là phát hiện các đối tượng chuyển động tại các nơi đặt camera quan sát như hành lang, cầu thang, v.v Tại các vị trí này nền hầu như là không thay đổi. Do đó ta có thể áp dụng kỹ thuật trừ ảnh để phát hiện ra có đối tượng chuyển động hay không. Để đơn giản ta sử dụng kỹ thuật trừ điểm ảnh. Kỹ thuật này nhằm tìm ra sự sai khác giữa hai ảnh.

Trên cở sở phân tích yêu cầu của bài toán, chúng ta hãy xem xét thuật giải được đề xuất sau đây. Ý tưởng của thuật giải là lấy hình ảnh trực tiếp từ webcam trong mọi khoảng thời gian (xem nó như là ảnh hiện tại) và so sánh nó với ảnh trước đó bằng kỹ thuật trừ điểm ảnh. Nếu tìm thấy sự sai khác lớn giữa chúng thì ta sẽ lưu lại hai ảnh này. Ngược lại, giải phóng bộ nhớ mà ảnh cũ đang chiếm giữ và xem ảnh mới nhận được là ảnh hiện tại.

Sơ đồ thuật giải như sau:



Hình 3.1: Sơ đồ thuật giải

Đầu tiên ảnh chụp được từ webcam được lưu vào biến old. Tiếp theo, một ảnh khác lấy từ webcam trong khoảng thời gian liền sau đó được lưu vào biến cur. So sánh cur và old bằng cách so sánh màu của mỗi điểm ảnh. Nếu sự sai khác lớn hơn một ngưỡng nào đó (tùy theo chất lượng của camera và ánh sáng hệ thống mà các ngưỡng có thể khác nhau) thì chúng ta sẽ đưa ra cảnh báo đồng thời lưu lại hai ảnh này. Cuối cùng gán old = cur và quay trở lại bước 2.

3.3. CÁC HÀM VÀ LỚP CHÍNH TRONG CHƯƠNG TRÌNH

Chương trình phát hiện đối tượng chuyển động AntiThief được cài đặt trên ngôn ngữ lập trình C#. Sau đây là một số hàm và lớp chính trong chương trình. ***Lớp ImageProcessing:*** Chức năng chính của lớp này là xử lý các hình ảnh thu được từ camera.

- Các biến và kiểu dữ liệu trong lớp

```
Bitmap flag,flag2,flag3;
int width,width2,width3;
BitmapData bitmapData = null,bitmapData2= null,bitmapData3= null;
Byte* pBase = null,pBase2=null,pBase3=null;

public struct Pixel
{
    public byte blue;
    public byte green;
    public byte red;
}
```

- Một số phương thức trong lớp
  - ***Ph□ơng thức PixelSize:*** cho kích cỡ của điểm ảnh. Cấu trúc của ph- ơng thức nh- sau:

```
public Point PixelSize
{
    get
    {
        GraphicsUnit unit = GraphicsUnit.Pixel;
        RectangleF bounds = flag.GetBounds(ref unit);

        return new Point((int) bounds.Width, (int) bounds.Height);
    }
}
```

  - ***Ph□ơng thức PixelAt:*** cho vị trí của điểm ảnh. Cú pháp của ph- ơng thức nh- sau:

```
public Pixel* PixelAt(int x, int y)
{
    return (Pixel*) (pBase + y * width + x * sizeof(Pixel));
}
```

- Một số hàm trong lớp
  - ***Hàm LockBitmap:*** đ- ợc sử dụng trong việc trừ hai ảnh. Cú pháp của hàm nh- sau:

```
public void LockBitmap()
{
    GraphicsUnit unit = GraphicsUnit.Pixel;
    RectangleF boundsF = flag.GetBounds(ref unit);
    Rectangle bounds = new Rectangle((int) boundsF.X,
        (int) boundsF.Y,
        (int) boundsF.Width,
        (int) boundsF.Height);


    width = (int) boundsF.Width * sizeof(Pixel);
    if (width % 4 != 0)
    {
        width = 4 * (width / 4 + 1);
    }

    bitmapData =
        flag.LockBits(bounds, ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);

    pBase = (Byte*) bitmapData.Scan0.ToPointer();

}
```

- ***Hàm UnlockBitmap:*** được sử dụng trong việc trừ hai ảnh. Cú pháp như sau:

```
public void UnlockBitmap()
{
    flag.UnlockBits(bitmapData);
    bitmapData = null;
    pBase = null;

}
```

- ***Hàm Save:*** l- u lại ảnh. Cú pháp nh- sau:

```
public void Save(string filename)
{
    flag3.Save(filename, ImageFormat.Jpeg);
}
```

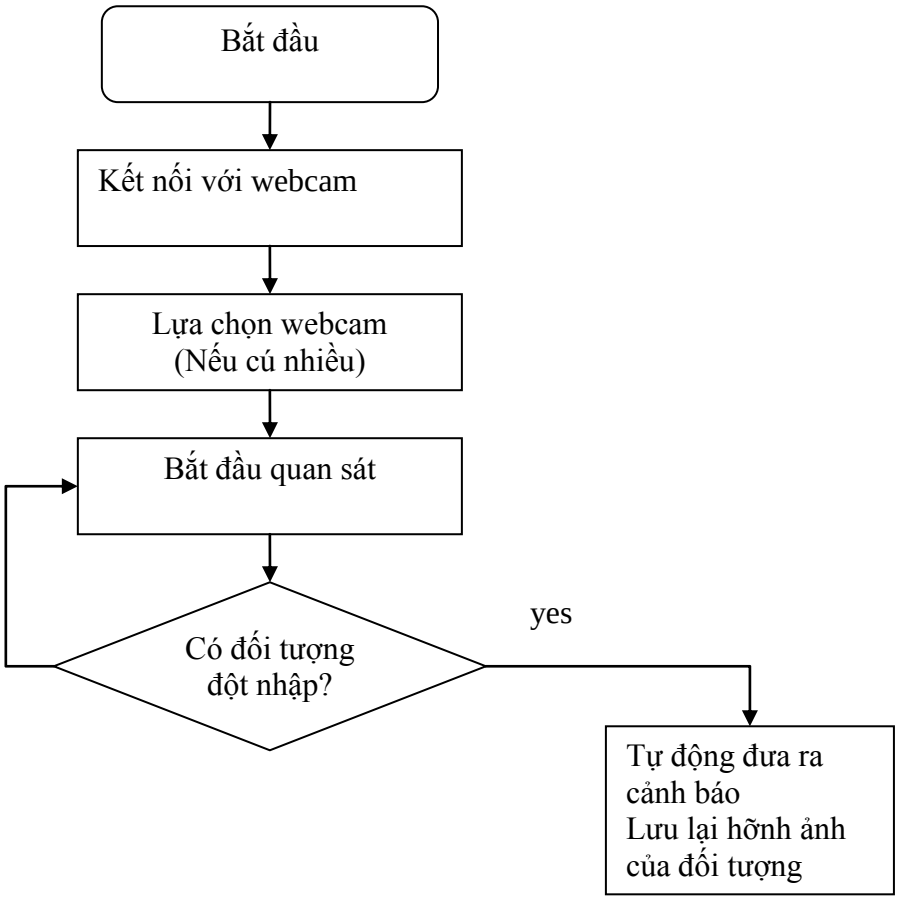
- ***Hàm CompareUnsafeFaster:*** thực hiện việc trừ hai ảnh. Cú pháp của hàm như sau:

```
public void CompareUnsafeFaster(out Int32
percent)
        {
            Point size = PixelSize;
            percent=0;
            LockBitmap();
            LockBitmap2();
            LockBitmap3();
```

```
                for (int y = 0; y < size.Y; y++)
                {
                     Pixel* pPixel = PixelAt(0, y);
                     Pixel* pPixel2 = PixelAt2(0, y);
                     Pixel* pPixel3 = PixelAt3(0, y);
                     for (int x = 0; x < size.X; x++)
                     {
                      if(!(pPixel->green==pPixel2->green))
                          {
                               pPixel3->red = pPixel2->red;
                               pPixel3->green = pPixel2-
>green;
                               pPixel3->blue = pPixel2->blue;
                               percent++;
                          }
                          pPixel++;
                          pPixel2++;
                          pPixel3++;
                     }
                }
                UnlockBitmap3();
                UnlockBitmap2();
                UnlockBitmap();
                 }
```

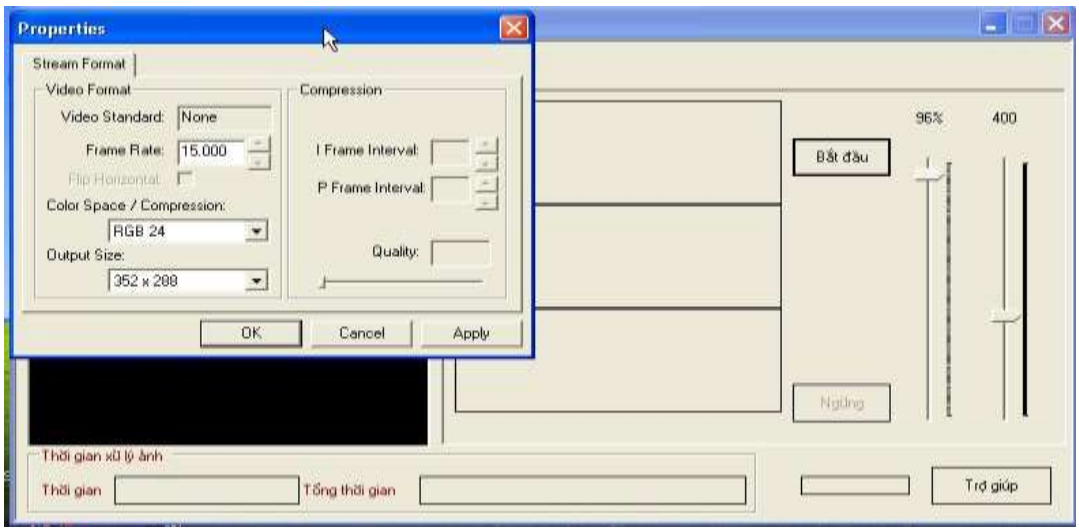
3.4. Chức năng và cách sử dụng chương trình

Chức năng chính của chương trình là kết nối với các camera, hiển thị hình ảnh thu từ webcam lên trên form. Khi phát hiện có đối tượng chuyển động chương trình sẽ tự động đưa ra cảnh báo và lưu lại ảnh có chứa đối tượng chuyển động. Sau đây sơ đồ hoạt động của chương trình:



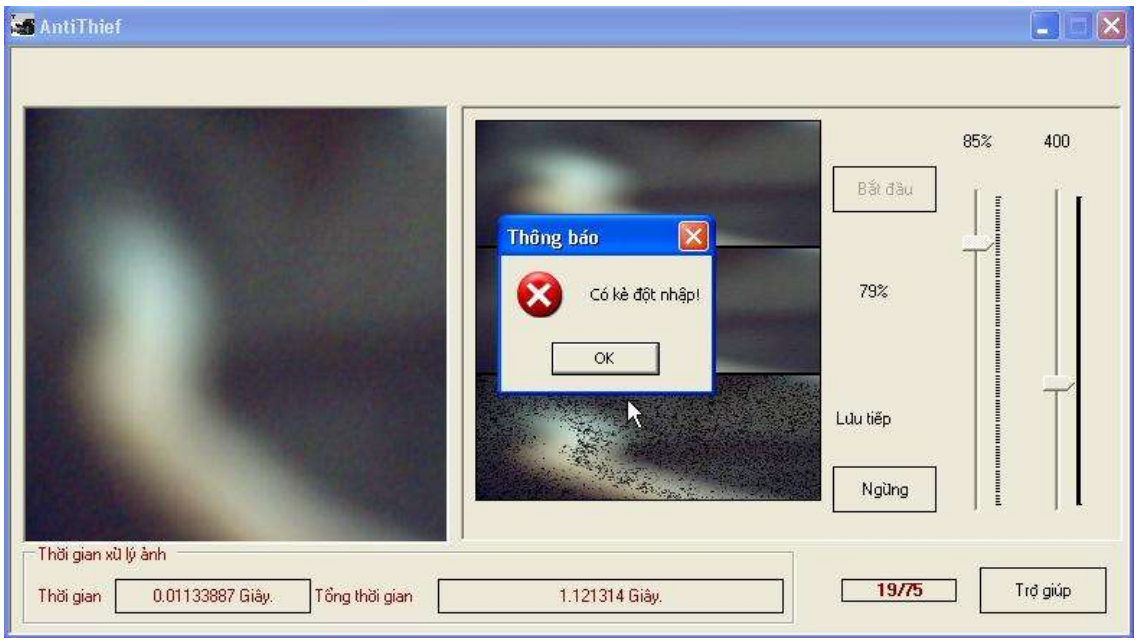
Hỗnh 3.2: Sơ đồ hoạt động của chương trỗnh

Khi khởi động chương trình có giao diện như sau:



Trong hộp thoại properties cho phép chúng ta thiết lập các thông số cho ảnh đầu ra. Chúng ta có thể thay đổi loại ảnh, kích cỡ ảnh đầu ra, chất lượng của ảnh (tùy theo từng camera mà các thông số này có thể khác nhau). Nhấn nút Apply để áp dụng và nhân nút OK để đóng hộp thoại này. Khi đó chương trình bắt đầu hiển thị hình ảnh thu được lên form:

Khi chúng ta nhấn vào nút bắt đầu chương trình sẽ bắt đầu quan sát, và khi phát hiện có đối tượng chuyển động nó sẽ xuất ra thông báo và lưu lại hình ảnh có chứa đối tượng chuyển động vào thư mục wanted:



KẾT LUẬN

Trong quá trình làm khóa luận tốt nghiệp với đề tài: "Tìm hiểu bài toán phát hiện đối tượng chuyển động", tôi đã giải quyết được một số vấn đề cơ bản trong việc xây dựng một hệ thống giám sát tự động:

Phần lý thuyết

- Tìm hiểu về một hệ thống xử lý ảnh cũng như các kỹ thuật trừ ảnh. Cách thức xây dựng một hệ thống giám sát tự động
- Tìm hiểu về ngôn ngữ lập trình Csharp và kỹ thuật bắt giữ hình ảnh thông qua camera trên môi trường Windows

Phần thực nghiệm

- Cài đặt thành công chương trình ứng dụng phát hiện đối tượng chuyển động thông qua webcamera
- Giảm dung lượng lưu trữ trên không gian đĩa, bằng cách chỉ lưu lại những hình ảnh có chứa đối tượng chuyển động thay vì phải lưu lại cả đoạn video

TÀI LIỆU THAM KHẢO

[1] Bài giảng xử lý ảnh – TS. Đỗ Năng Toàn, Viện công nghệ thông tin Việt Nam – Viện khoa học và công nghệ Việt Nam

[2] Nhập môn xử lý ảnh số – Lương Mạnh Bá, Nguyễn Thanh Thủy

[3] Hampapur, A., Jain, R., Weymouth, T., Digital Video Segmentation, Proc. ACM Multimedia 94, San Francisco CA, 1994, pp. 357 – 364

[4] Jyrpi Korki - Anttila (2002), "Automatic color enhancement and sence change detection of digital video", Dept of Automation and Systems, Lab of Media Technology, Hensiki University of Technology

[5] Shahraray, B., Scene Change Detection and Content-Based Sampling of Video Sequences, Digital Video Compression: Algorithms and Technologies, A. Rodriguez, R. Safranek, E. Delp, Editors, Proc. SPIE 2419, 1995, pp. 2 – 13

[6] Xiong, W., Lee, J. C.-M., Ip, M.C., Net comparison: a fast and effective method for classifying image sequences, SPIE Conf. Storage and Retrieval for Image and Video Databases III, Proceedings, San Jose, CA, 1995, pp. 318 – 328

[7] Background Subtraction for 3D Vision – based user Interfaces – Dongpyo Hong, Thesis for Master's Degree

HƯỚNG PHÁT TRIỂN TIẾP THEO CỦA ĐỀ TÀI

Do thời gian có hạn nên vẫn còn một số vấn đề mà đề tài chưa giải quyết được. Sau đây, tôi xin đưa ra một số hướng có thể mở rộng và hoàn thiện đề tài nhằm áp dụng cho thực tiễn:

- Kết hợp kỹ thuật trừ ảnh và các kỹ thuật xử lý ảnh khác như trích chọn đặc trưng ảnh, dò biên, v.v để nhận dạng được đối tượng chuyển động
- Nghiên cứu và phát triển chương trình để có thể hoạt động trên một mạng máy tính

PHỤ LỤC

1. MÃ NGUỒN LỚP IMAGEPROCESSING.CS

```
using System;
using System.Drawing;
using System.Drawing.Imaging;


namespace ImgProcessing
{
	public unsafe sealed class ImageProcessing
	{
		Bitmap flag,flag2,flag3;
		int width,width2,width3;
		BitmapData bitmapData = null,
		bitmapData2= null,
		bitmapData3= null;
		Byte* pBase = null,pBase2=null,pBase3=null;

		#region ImageProcessing Constructor
		public ImageProcessing(Bitmap picOld,Bitmap picNew,
			Bitmap target)
		{
			this.flag=picOld;
			this.flag2=picNew;
			this.flag3=target;
		}
		public ImageProcessing(Bitmap source,Bitmap target)
		{
			this.flag=source;
			this.flag2=target;
		}
		#endregion

		#region internal methods
		#region PixelSize
		public Point PixelSize
		{
			get
			{
				GraphicsUnit unit = GraphicsUnit.Pixel;
				RectangleF bounds = flag.GetBounds(ref unit);
				return new Point((int) bounds.Width,
				(int) bounds.Height);
			}
		}
```

```
public Point PixelSize2
{
	get
	{
		GraphicsUnit unit = GraphicsUnit.Pixel;
		RectangleF bounds = flag2.GetBounds(ref unit);
		return new Point((int) bounds.Width,
		(int) bounds.Height);
	}
}
public Point PixelSize3
{
	get
	{
		GraphicsUnit unit = GraphicsUnit.Pixel;
		RectangleF bounds = flag3.GetBounds(ref unit);
		return new Point((int) bounds.Width,
		(int) bounds.Height);
	}
}
#endregion

#region PixelAt
public Pixel* PixelAt(int x, int y)
{
	return (Pixel*) (pBase + y * width + x * sizeof(Pixel));
}
public Pixel* PixelAt2(int x, int y)
{
	return (Pixel*) (pBase2 + y * width2 + x * sizeof(Pixel));
}
public Pixel* PixelAt3(int x, int y)
{
	return (Pixel*) (pBase3 + y * width3 + x * sizeof(Pixel));
}

#endregion

#region LockBitMap
public void LockBitmap()
{
	GraphicsUnit unit = GraphicsUnit.Pixel;
	RectangleF boundsF = flag.GetBounds(ref unit);
	Rectangle bounds = new Rectangle((int) boundsF.X,
		(int) boundsF.Y,
		(int) boundsF.Width,
```

```
            (int) boundsF.Height);
            width = (int) boundsF.Width * sizeof(Pixel);
            if (width % 4 != 0)
            {
                  width = 4 * (width / 4 + 1);
            }

            bitmapData =
                  flag.LockBits(bounds, ImageLockMode.ReadWrite,
                  PixelFormat.Format24bppRgb);

            pBase = (Byte*) bitmapData.Scan0.ToPoànter();

}
public void UnlockBitmap()
{
            flag.UnlockBits(bitmapData);
            bitmapData = null;
            pBase = null;

}
public void LockBitmap2()
{
            GraphicsUnit unit = GraphicsUnit.Pixel;
            RectangleF boundsF = flag2.GetBounds(ref unit);
            Rectangle bounds = new Rectangle((int) boundsF.X,
                  (int) boundsF.Y,
                  (int) boundsF.Width,
                  (int) boundsF.Height);


            width2 = (int) boundsF.Width * sizeof(Pixel);
            if (width2 % 4 != 0)
            {
                  width2 = 4 * (width2 / 4 + 1);
            }

            bitmapData2 =
                  flag2.LockBits(bounds, ImageLockMode.ReadWrite,
                  PixelFormat.Format24bppRgb);

            pBase2 = (Byte*) bitmapData2.Scan0.ToPointer();
}

public void UnlockBitmap2()
```

```
{
        flag2.UnlockBits(bitmapData2);
        bitmapData2= null;
        pBase2= null;
}
public void LockBitmap3()
{
        GraphicsUnit unit = GraphicsUnit.Pixel;
        RectangleF boundsF = flag3.GetBounds(ref unit);
        Rectangle bounds = new Rectangle((int) boundsF.X,
                (int) boundsF.Y,
                (int) boundsF.Width,
                (int) boundsF.Height);


        width3 = (int) boundsF.Width * sizeof(Pixel);
        if (width3 % 4 != 0)
        {
                width3 = 4 * (width3 / 4 + 1);
        }

        bitmapData3 =
                flag3.LockBits(bounds, ImageLockMode.ReadWrite,
                PixelFormat.Format24bppRgb);

        pBase3 = (Byte*) bitmapData3.Scan0.ToPointer();
}

public void UnlockBitmap3()
{
        flag3.UnlockBits(bitmapData3);
        bitmapData3= null;
        pBase3= null;
}
#endregion

#region Save
public void Save(string filename)
{
        flag3.Save(filename, ImageFormat.Jpeg);
}
#endregion

#region Dispose
public void Dispose()
```

```
{
        flag=null;
        flag2=null;
        flag3=null;
}
#endregion

public Bitmap bitmap
{
        get
        {
                return(this.flag3);
        }
}


#endregion

#region Magic code for CompareUnsafeFaster
public void CompareUnsafeFaster(out Int32 percent)
{
        Point size = PixelSize;
        percent=0;
        LockBitmap();
        LockBitmap2();
        LockBitmap3();

        for (int y = 0; y < size.Y; y++)
        {
                Pixel* pPixel = PixelAt(0, y);
                Pixel* pPixel2 = PixelAt2(0, y);
                Pixel* pPixel3 = PixelAt3(0, y);
                for (int x = 0; x < size.X; x++)
                {
                 if(!(pPixel->green==pPixel2->green))
                {
                                pPixel3->red = pPixel2->red;
                                pPixel3->green = pPixel2->green;
                                pPixel3->blue = pPixel2->blue;
                                percent++;
                }
                        pPixel++;
                        pPixel2++;
                        pPixel3++;
                }
```

```
            }
            UnlockBitmap3();
            UnlockBitmap2();
            UnlockBitmap();
}

        #endregion

        #region Magic code for ComplementUnsafeFaster
        public void ComplementUnsafeFaster()
        {
            Point size = PixelSize;

            LockBitmap();
            LockBitmap2();


            for (int y = 0; y < size.Y; y++)
            {
                Pixel* pPixel = PixelAt(0, y);
                Pixel* pPixel2 = PixelAt2(0, y);

                for (int x = 0; x < size.X; x++)
                {

                    pPixel2->red =(byte)(255-(int)pPixel->red);
                    pPixel2->green = (byte)(255-(int)pPixe>green);
                    pPixel2->blue = (byte)(255-(int)pPixel->blue);

                    pPixel++;
                    pPixel2++;

                }
            }
            UnlockBitmap2();
            UnlockBitmap();
            flag3=new Bitmap(flag2);

        }

        #endregion

        #region Magic code for ColorBallUnsafeFaster
        public void ColorBallUnsafeFaster(double red,double green,
                double blue)
```

```
{
        Point size = PixelSize;
        double fr, fg, fb;
        if(red<0)
        {
                fr=red/100+1;
        }
        else
        {
                fr=red/100;
        }
        if(green<0)
        {
                fg=green/100+1;
        }
        else
        {
                fg=green/100;
        }
        if(blue<0)
        {
                fb=blue/100+1;
        }
        else
        {
                fb=blue/100;
        }
        LockBitmap();
        LockBitmap2();


        for (int y = 0; y < size.Y; y++)
        {
                Pixel* pPixel = PixelAt(0, y);
                Pixel* pPixel2 = PixelAt2(0, y);

                for (int x = 0; x < size.X; x++)
                {

                        if(  red < 0 )
                        {
                                pPixel2->red =
                                        (byte)((int)pPixel->red * fr);
                        }
                        else
```

```
                        {
                                pPixel2->red =
                                        (byte)((int)pPixel->red
                                        + (255 - (int)pPixel->red) * fr);
                        }
                        if(  green < 0 )
                        {
                                pPixel2->green =
                                        (byte)((int)pPixel->green * fg);
                        }
                        else
                        {
                                pPixel2->green =
                                        (byte)((int)pPixel->green
                                        + (255 - (int)pPixel->green) * fg);
                        }
                        if(  blue < 0 )
                        {
                                pPixel2->blue =
                                        (byte)((int) pPixel->blue * fb);
                        }
                        else
                        {
                                pPixel2->blue=
                                        (byte)((int)pPixel->blue
                                        + (255 - (int)pPixel->blue) * fb);
                        }

                        pPixel++;
                        pPixel2++;

                }
        }
        UnlockBitmap2();
        UnlockBitmap();
        flag3=new Bitmap(flag2);

}

#endregion

#region Magic code for Brightness
public void Brightness(double brightness)
{
        Point size = PixelSize;
```

```
double f;
if(brightness<0)
{
	f=brightness/100+1;
}
else
{
	f=brightness/100;
}

LockBitmap();
LockBitmap2();


for (int y = 0; y < size.Y; y++)
{
	Pixel* pPixel = PixelAt(0, y);
	Pixel* pPixel2 = PixelAt2(0, y);

	for (int x = 0; x < size.X; x++)
	{
		if(  brightness < 0 )
		{
			pPixel2->red =
				(byte)( (int)pPixel->red * f);
			pPixel2->green =
				(byte)( (int)pPixel->green * f);
			pPixel2->blue =
				(byte)( (int)pPixel->blue * f);
		}
		else
		{
			pPixel2->green =
				(byte)((int)pPixel->red
				+ (255 - (int)pPixel->red) * f);
			pPixel2->green =
				(byte)((int)pPixel->green
				+ (255 - (int)pPixel->green) * f);
			pPixel2->blue=
				(byte)((int)pPixel->blue
				+ (255 - (int)pPixel->blue) * f);
		}

		pPixel++;
		pPixel2++;
```

```
                }
            }
            UnlockBitmap2();
            UnlockBitmap();
            flag3=new Bitmap(flag2);

        }

        #endregion

        #region MakeGreyUnsafeFaster
        public void MakeGreyUnsafeFaster()
        {
            Point size = PixelSize;

            LockBitmap();
            LockBitmap2();

            for (int y = 0; y < size.Y; y++)
            {
                Pixel* pPixel = PixelAt(0, y);
                Pixel* pPixel2 = PixelAt2(0, y);
                for (int x = 0; x < size.X; x++)
                {
                    byte value =
                        (byte) ((pPixel->red + pPixel->green +
                                 pPixel->blue) / 3);
                    pPixel2->red =  value;
                    pPixel2->green = value;
                    pPixel2->blue = value;
                    pPixel++;
                    pPixel2++;
                }
            }
            UnlockBitmap2();
            UnlockBitmap();
            flag3=new Bitmap(flag2);
        }
        #endregion

}

#region Pixel struct
public struct Pixel
```

```
        {
            public byte blue;
            public byte green;
            public byte red;
        }
        #endregion
}
```

2. Mã nguồn lớp Counter.cs

```
using System;

namespace Timing
{
    public class Counter
    {
        long elapsedCount = 0;
        long startCount = 0;

        public void Start()
        {
            startCount = 0;
            QueryPerformanceCounter(ref startCount);
        }

        public void Stop()
        {
            long stopCount = 0;
            QueryPerformanceCounter(ref stopCount);

            elapsedCount += (stopCount - startCount);
        }

        public void Clear()
        {
            elapsedCount = 0;
        }

        public float Seconds
        {
            get
            {
                long freq = 0;
                QueryPerformanceFrequency(ref freq);
                return((float) elapsedCount / (float) freq);
            }
        }
```

```
        public override string ToString()
        {
            return String.Format("{0} Giõy.", Seconds);
        }

        static long Frequency
        {
            get
            {
                long freq = 0;
                QueryPerformanceFrequency(ref freq);
                return freq;
            }
        }
        static long Value
        {
            get
            {
                long count = 0;
                QueryPerformanceCounter(ref count);
                return count;
            }
        }

        [System.Runtime.InteropServices.DllImport("KERNEL32")]
        private static extern bool QueryPerformanceCounter
                                    (  ref long lpPerformanceCount);

        [System.Runtime.InteropServices.DllImport("KERNEL32")]
        private static extern bool QueryPerformanceFrequency
                                    ( ref long lpFrequency);
    }
}
```

LỜI CẢM ƠN

Sau một thời gian học tập và nghiên cứu em đã hoàn thành khóa luận tốt nghiệp với đề tài: Tìm hiểu bài toán phát hiện đối tượng chuyển động. Đầu tiên em xin bày tỏ lòng biết ơn chân thành đến thầy giáo PGS. TS. Đỗ Năng Toàn là người đã trực tiếp hướng dẫn và tạo điều kiện cho em được thực tập tại Viện công nghệ thông tin Viện khoa học và công nghệ Việt Nam để hoàn thành khóa luận này. Đồng thời em cũng xin chân thành cảm ơn các thầy cô giáo đang giảng dạy tại khoa công nghệ thông tin, trường đại học dân lập hải phòng trong suốt thời gian học tập vừa qua đã trang bị cho em những kiến thức cần thiết và bổ ích giúp em hoàn thành khóa luận này cũng như những kỹ năng nghề nghiệp sau này. Cuối cùng em xin cảm ơn gia đình và bạn bè đã luôn cổ vũ và động viên em trong suốt thời gian vừa qua. Do trình độ có hạn nên nội dung khóa luận còn sơ sài rất mong nhận được sự góp ý của các thầy cô giáo và các bạn.

Hải Phòng , ngày ,tháng ,năm 2010

Sinh viên

Bùi Cao Phát

MỤC LỤC