HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO
BÀI TẬP LỚN

MÔN: HỆ CƠ SỞ DỮ LIỆU ĐA PHƯƠNG TIỆN

ĐỀ TÀI: NHẬN DIỆN VÂN TAY

Giảng viên: Nguyễn Đình Hóa
Nhóm môn học: 02
Nhóm sinh viên thực hiện: 08

1. **Chu Văn Đăng** **B16DCCN052**
2. **Nguyễn Thị Minh Phương** **B16DCCN275**
3. **Trần Văn Tâm** **B16DCCN308**

Hà Nội, tháng 7 năm 2020

Lời cảm ơn

Trước hết chúng em xin được gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Đình Hóa. Thầy đã giao cho bọn em 1 đề tài rất hay, cô đọng không những đầy đủ kiến thức môn hệ cơ sở dữ liệu đa phương tiện mà còn giúp chúng em cải thiện kỹ năng làm việc nhóm và đặc biệt là kỹ năng lập trình – một kỹ năng không thể thiếu đối với một kĩ sư công nghệ thông tin.

Trong quá trình làm bài tập lớn chúng em đã gặp rất nhiều khó khăn nhưng chủ yếu là kiến thức của bọn em còn rất hạn chế. Chúng em đã nhận rất nhiều sự giúp đỡ đặc biệt đó là Google, giúp chúng em tìm ra những tài liệu rất quý để có thể hoàn thành được đề tài.

Một lần nữa chúng em xin cảm ơn thầy Nguyền Đình Hóa.

Mục lục

1. Tổng quan về vân tay

1.1 Vân tay

Vân tay theo nghĩa hẹp của nó là một ấn tượng để lại bởi các đường vân ma sát của ngón tay người. Việc thu hồi dấu vân tay từ hiện trường vụ án là một phương pháp quan trọng của khoa học pháp y. Dấu vân tay dễ dàng lắng đọng trên các bề mặt phù hợp (như thủy tinh hoặc kim loại hoặc đá đánh bóng) bởi các chất tiết mồ hôi tự nhiên từ các tuyến eccrine có trong các đường vân biểu bì. Đôi khi chúng được gọi là "Ấn tượng cơ hội" [1]. Độ rộng của các đường vân có kích thước từ 100-300 μm. Nói chung, chu vi của vân/rãnh là khoảng 500 μm [4]

Hình 1.1.1 Các Vân ma sát trên 1 ngón tay.

Vân tay thì được chia ra làm 3 chủng vân cơ bản là:

* Whorl (khoảng 35% dân số thế giới).

* Loop (khoảng 65% dân số thế giới).

* Arch (khoảng 5% dân số thế giới).

Hình 1.1.2 3 nhóm chính của 1 vân tay whorl, loop và arch

Độ trùng lặp của vân tay theo lí thuyết xác suất và thống kê Người ta đã chứng minh rằng dấu vân tay của mỗi cá nhân là độc nhất (trên dân số 7 tỷ người). Xác suất hai cá nhân - thậm chí ngay cả anh em (hoặc chị em) sinh đôi cùng trứng - có cùng một bộ dấu vân tay là 1 trên 64 tỉ. Ngay cả các ngón trên cùng bàn tay cũng có vân khác nhau. Dấu vân tay của mỗi người là không đổi trong suốt cuộc đời. Người ta có thể làm phẫu thuật thay da ngón tay, nhưng chỉ sau một thời gian dấu vân tay lại được hồi phục như ban đầu.

Trên đầu ngón tay mỗi người dều có những đường vằn nhỏ bởi đặc điểm này rất có lợi cho tổ tiên loài người, giúp họ dễ cầm nắm các vật dụng. Những vân tay hoàn toàn ngẫu nhiên. Như mọi thứ trong cơ thể con người, những đường vân tay là sự kết hợp của sự di truyền và những nhân tố môi trường. Ngoài ra cũng như mọi bộ phận khác trên cơ thể, các đường vằn này tạo thành một nhân tố tự nhiên đặc trưng của mỗi người, kể cả đối với những cặp song sinh giống hệt nhau. Tuy mới nhìn qua, hai vân tay trông có vẻ giống nhau nhưng một điều tra viên chuyên nghiệp hoặc một phần mềm cao cấp có thể nhìn thấu sự khác nhau rõ ràng giữa chúng. Tóm lại những vân tay của những người khác nhau là hoàn toàn khác nhau kể cả sinh đôi. Đây là ý tưởng cơ bản của hệ thống phân tích vân tay, cả trong việc điều tra tội phạm và bảo vệ an ninh. Máy quét vân tay có nhiệm vụ thay thế con người trong việc thu thập mẫu vân tay và so sánh nó với các mẫu khác trong hồ sơ.

1.2 Ảnh vân tay và các cách lưu trữ vân tay

1.2.1 Lưu trữ vân tay

Ngày xưa ngành công nghệ xử lí ảnh cũng như chưa có máy vi tính thì con người đã sử dụng tấm thẻ bằng đất sét (dùng để cho việc giao dịch kinh doanh ở thời Babylon cổ xưa). Hay xưa họ sẽ lưu trữ bằng cách điểm chỉ qua giấy.

Ngày nay với công nghệ số ngày càng phát triển thì 1 vân tay có thể lưu trữ được rất nhiều dạng nhưng chủ yếu sẽ là ở dạng hình ảnh.

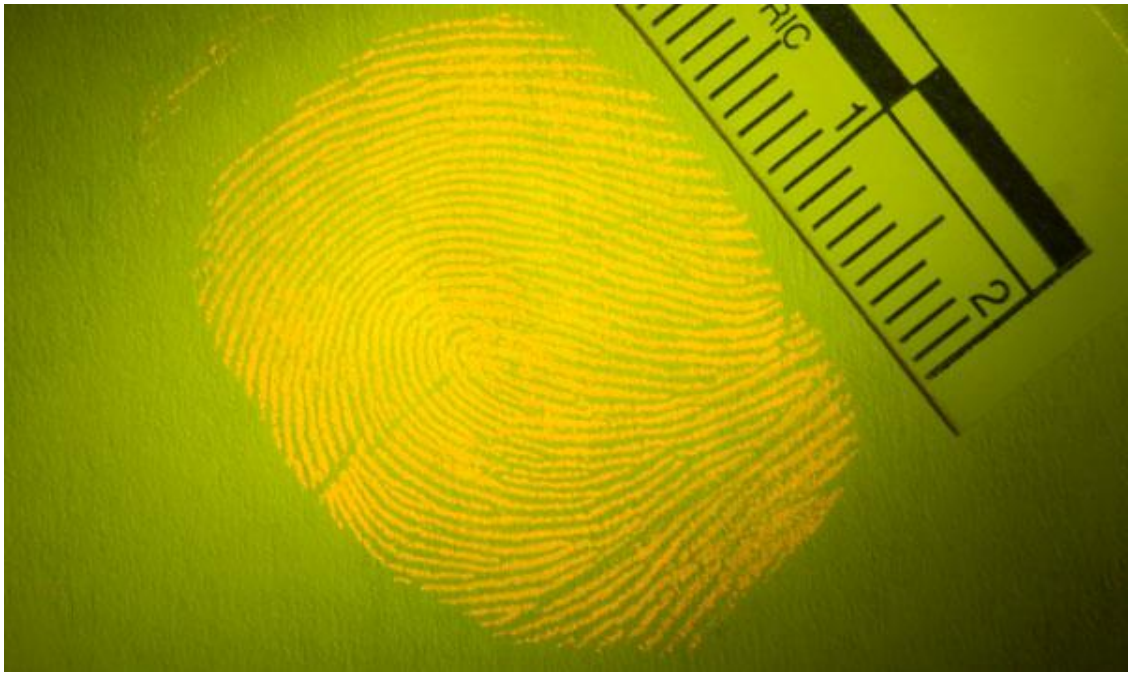
1.2.2 Thu thập vân tay

1.2.2.1 Lấy gián tiếp

Chúng có thể tìm thấy ở mọi nơi, miễn là một bề mặt rắn, bao gồm cả trên cơ thể người. Các nhà phân tích chia ra 3 loại vân tay tuỳ theo nơi mà chúng được tìm thấy. Trước hết là dấu vân trên những bề mặt mềm như xà phòng, sáp nến, khăn ướt... còn được gọi là vân "mềm" 3 chiều. 2 loại còn lại là vân trên bề mặt cứng và chia thành vân nhìn thấy (hay vân nổi - patent print) hoặc không nhìn thấy (hay vân chìm - latent print). Vân nổi có thể dễ dàng tìm thấy được vì chúng nổi lên trên các bề mặt khi máu, bụi đất, mực, sơn, dầu... mà hung thủ để lại trên bề mặt. Còn vân chìm rất khó nhận thấy bằng mắt thường, chúng được hình thành khi dầu và mồ hôi của 1 người bám lên một bề mặt khác. Để tìm thấy vân chìm, các nhà điều tra phải dùng đến nhiều công cụ khác nhau để làm nổi bật chúng lên.

Một phương pháp phổ biến nhất để lấy vân chìm là phủ bụi lên khu vực có dấu vân bằng các loại bột lưu vân tay. Chúng sẽ bám lên các vị trí có vân tay và biến vân chìm thành vân nổi. Tiếp đấy các nhà điều tra sẽ chụp hình lại y như cách thu thập vân nổi. Sau đó, chúng còn được lưu mẫu bằng cách áp băng dính để lưu trữ về sau này.

Tuy nhiên, các loại bột này có thể làm hư hại đến các bằng chứng liên quan khác có ở hiện trường khiến cho công tác điều tra bị ảnh hưởng. Vì thế, trước khi sử dụng loại bột trên, nhà điều tra có thể sẽ dùng các phương pháp lấy vân chìm khác để hỗ trợ.



Hình 1.2.2.1 Ánh sáng giúp giúp cải thiện chất lượng hình ảnh vân tay trên đồ vật [3]

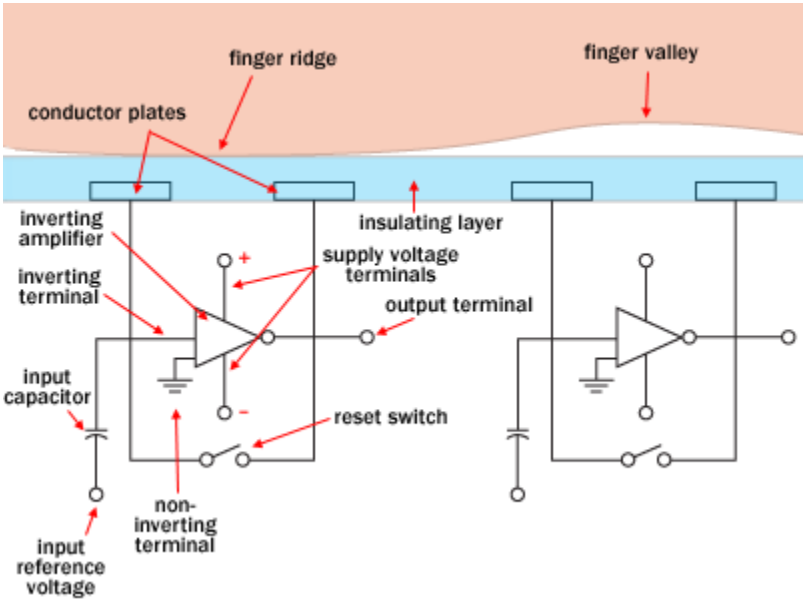
Hoặc chúng ta có thể lấy vân tay trên các tờ giấy điểm chỉ. Tuy nhiên phương pháp này bị vô hiệu hoá trong trường hợp kẻ gian dùng một bao cao su giả mạo dấu vân tay hay cũng không loại trừ trường hợp bọn chúng "mượn" luôn ngón tay của người khác để phục vụ cho mục đích riêng.

1.3 Lấy trực tiếp

Máy quét quang học - Một hệ thống máy quét quang học có hai nhiệm vụ cơ bản – lấy hình ảnh ngón tay, và quyết định xem liệu dấu vân tay trong ảnh có khớp với mẫu có sẵn hay không. Có nhiều cách khác nhau để lấy hình ảnh ngón tay, trong đó cách phổ biến nhất là quét quang học và quét điện dung. Cả hai đều đưa ra cùng một hình ảnh, nhưng bằng hai phương pháp khác nhau hoàn toàn. Trung tâm của máy quét quang học là CCD (Charge Coupled Device) – hệ thống cảm biến ánh sáng sử dụng trong camera kỹ thuật số. CCD là một mảng diode nhạy cảm với ánh sáng gọi là photosite, có nhiệm vụ tạo tín hiệu điện tương ứng với những photon ánh sáng. Mỗi photosite ghi lại một pixel, tức một chấm nhỏ thể hiện rằng ánh sáng đã chạm đến điểm đó. Các pixel sáng và tối sẽ tổng hợp thành một hình ảnh của vật thể được quét (như ngón tay). Thường thì Bộ chuyển đổi ADC (từ analog sang digital) trong hệ thống quét sẽ xử lý tín hiệu điện analog để tạo ra bức ảnh dạng số hóa. Quy trình quét bắt đầu khi bạn đặt ngón tay lên một đĩa thủy tinh, và để CCD chụp ảnh. Máy quét có nguồn điện riêng, thường là một mảng diode phát sáng để tỏa sáng các đường vân trên ngón tay. Cuối cùng hệ thống CCD sẽ tạo ra hình ảnh đảo ngược của ngón tay. Trước khi so sánh hình ảnh nhận được với dữ liệu có sẵn, bộ xử lý quét sẽ đảm bảo rằng hình ảnh thu được đủ rõ bằng cách kiểm tra độ tối pixel trung bình, hay tổng giá trị của một mẫu nhỏ, và sẽ từ chối quét hình nếu nó quá sáng hoặc quá tối. Nếu ảnh bị từ chối, máy quét sẽ điều chỉnh thời gian

phơi sáng, rồi quét lại lần nữa. Còn nếu độ tối đã đủ thì hệ thống sẽ tiếp tục kiểm tra độ phân giải ảnh. Bộ xử lý sẽ quan sát một số đường thẳng di chuyển ngang dọc trên ảnh. Nếu ảnh có độ phân giải tốt, đường thẳng chạy vuông góc với vân tay sẽ gồm các phần xen kẽ gồm các pixel rất tối và rất sáng. Nếu bộ xử lý cảm thấy hình ảnh đã đủ sắc nét và độ sáng, nó sẽ tiếp tục so sánh hình ảnh đó với vân tay trong hồ sơ.

Máy quét điện dung, cũng như máy quét quang học, máy quét điện dung tạo ra hình ảnh vân tay, nhưng không phải bằng ánh sáng mà bằng dòng điện. Biểu đồ dưới đây minh họa một cảm biến điện dung đơn giản, gồm môt hoặc vài chip bán dẫn chứa một mảng gồm nhiều ngăn nhỏ. Mỗi ngăn có hai đĩa dẫn điện (Comductor Plate), được phủ một lớp cách điện. Mỗi ngăn này nhỏ hơn chiều rộng của một đường vân trên ngón tay.



Hình 1.2.2.2 Hệ thống của máy quét điện dung [2]

Cảm biến được nối với một mạch điện nối xung quanh một bộ khuếch đại toán học đảo ngược (Inverting Amplifier) – một thiết bị bán dẫn phức tạp gồm nhiều transistor, điện trở và tụ điện. Cũng như các bộ khuếch đại khác, một bộ khuếch đại ngược sẽ thay đổi dòng điện dựa trên dao động của một dòng điện khác. Trong trường hợp này, bộ khuếch đại ngược sẽ thay đổi điện áp nguồn dựa trên điện áp tương đối của hai đầu vào, gọi là đầu đảo ngược và đầu không đảo ngược. Ở đây, đầu không đảo ngược được nối đất (Non-Inverting Terminal), còn đầu đảo ngược được nối với nguồn điện áp tham chiếu và một vòng lặp phản hồi. Vòng lặp này cũng được nối với đầu ra của bộ khuếch đại, bao gồm hai đĩa dẫn điện. Hai đĩa dẫn điện này hợp thành một tụ điện, có khả năng lưu trữ điện năng. Bề mặt của ngón tay có vai trò như một tụ điện thứ ba, được ngăn cách bởi lớp cách điện trong hệ thống ngăn – tức các túi khí trong đường rãnh trên đầu ngón tay. Khi thay đổi khoảng cách giữa các đĩa tụ điện (bằng cách di chuyển ngón tay ra xa hoặc gần đĩa dẫn điện) thì khả năng trữ điện của tụ điện cũng thay đổi theo. Chính vì tính chất này nên tụ điện trong một ngăn nằm trong đường vân nổi sẽ có điện dung lớn hơn so với tụ điện trong

ngăn nằm ở đường rãnh. Để quét ngón tay, đầu tiên bộ xử lý sẽ đóng công tắc của mỗi ngăn, làm đoản mạch từng Đầu Vào (input) và Đầu Ra (output) để "cân bằng" mạch điện. Khi mở lại công tắc, bộ xử lý sẽ đưa một dòng điện cố định vào mạch điện để truyền điện cho tụ. Điện dung của tụ điện trong vòng lặp phản hồi sẽ ảnh hưởng đến điện áp tại input, rồi ảnh hưởng đến output. Do khoảng cách đến ngón tay sẽ làm thay đổi điện dung nên đường vân sẽ có điện áp cao hơn đường rãnh. Bộ xử lý quét sẽ đọc output điện áp và quyết định xem đây là đường rãnh hay đường vân. Bằng cách đọc từng ngăn trong mảng cảm biến, bộ xử lý có thể tổng hợp thành một hình ảnh vân tay chung, tương tự như hình ảnh do máy quét quang thu được. Ưu điểm chính của máy quét điện dung là nó cần có hình ảnh vân tay thực chứ không chỉ là một tấm ảnh 2 màu, vì thế khó qua mặt hơn. Ngoài ra do chúng sử dụng chip bán dẫn thay vì CCD nên máy quét điện dung cũng gọn nhẹ hơn máy quét quang học.

1.4 Sinh trắc vân tay

Công nghệ Sinh trắc học (Biometric) - là một công nghệ sử dụng những thuộc tính vật lý hoặc các mẫu hành vi, các đặc điểm sinh học đặc trưng như dấu vân tay, mẫu mống mắt, giọng nói, khuôn mặt, dáng đi, ... để nhận diện con người. Sinh trắc học (biometric) là một công cụ kiểm tra cá nhân hữu hiệu chưa từng có trong lịch sử.

Công nghệ sinh trắc học được áp dụng phổ biến và lâu đời nhất là công nghệ nhận dạng vân tay. Dấu vân tay là một đặc điểm quan trọng để phân biệt giữa người này và người khác. Sự phát triển của công nghệ thông tin có thể giúp thu nhận và ghi nhớ được hàng triệu ghi chép dưới dạng số hoá. Kỹ thuật này được đánh giá sẽ là chìa khoá của một cuộc cách mạng công nghệ mới, khi những thiết bị có khả năng nhận dạng vân tay để bảo vệ dữ liệu được ứng dụng ngày càng nhiều.

Nguyên lý hoạt động của Công nghệ nhận dạng vân tay: Khi đặt ngón tay lên trên một thiết bị nhận dạng dấu vân tay, ngay lập tức thiết bị này sẽ quét hình ảnh ngón tay đó và đối chiếu các đặc điểm của ngón tay đó với dữ liệu đã được lưu trữ trong hệ thống. Quá trình xử lý dữ liệu sẽ được thiết bị chuyển sang các dữ liệu số và ra thông báo rằng dấu vân tay đó là hợp lệ hay không hợp lệ để cho phép hệ thống thực hiện các chức năng tiếp theo.

Đặc điểm của dấu vân tay dù chỉ gồm có 7 loại (vòng móc đơn, vòng móc kép, vòng tập trung ở giữa, vòng cung, vòng cung hình lều, vòng xoắn, vòng bất thường) nhưng thể hiện về chi tiết khác nhau muôn hình muôn vẻ và không thay đổi từ khi mới sinh ra cho đến khi về già. Khai thác tính độc nhất về cấu tạo hình dạng vân tay của mỗi người, các nhà sinh trắc học sẽ biến nó thành chiếc chìa khoá riêng mà chỉ bạn mới có thể sử dụng, giúp bạn tránh được nhiều phiền toái trong cuộc sống như bị trộm cắp, lạm dụng hoặc giả mạo các loại giấy tờ tuỳ thân, thẻ ngân hàng, hộ chiếu... đảm bảo an ninh và bảo mật.

Hệ thống sinh trắc học sẽ ghi nhận mẫu vân tay của người dùng và lưu trữ tất cả

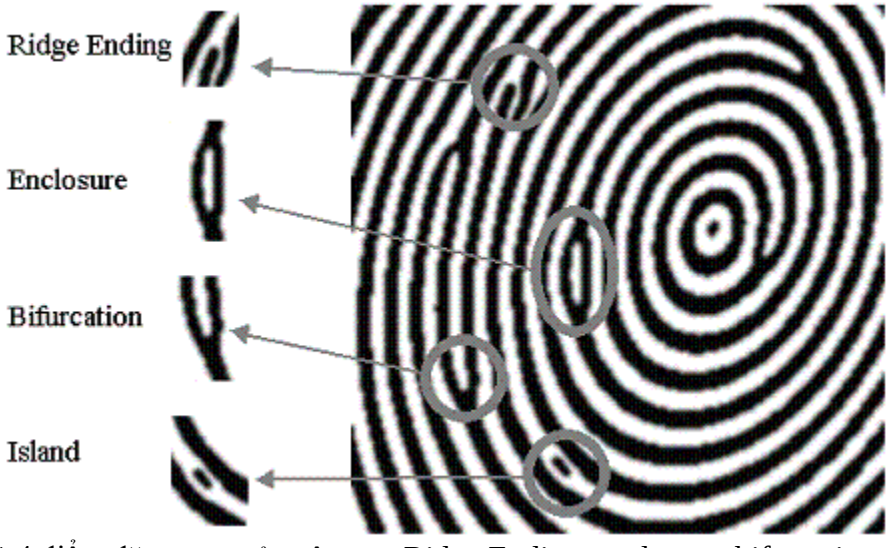
những dữ liệu đặc biệt này thành một mẫu nhận diện được số hoá toàn phần. Có hai phương pháp để lấy dấu vân tay:

Ngày nay việc ứng dụng công nghệ sinh trắc học về nhận dạng vân tay được sử dụng ngày càng nhiều vì nó đáp ứng yêu cầu bảo vệ dữ liệu, đảm bảo an ninh an toàn với độ chính xác cao.

1.5 Đặc trưng của vân tay.

Trên phim ảnh, máy phân tích vân tay tự động thường chồng các lớp vân tay lên nhau để tìm vân tay khớp. Nhưng trong thực tế thì đây không phải là cách so sánh vân tay thường dùng. Nếu hình bị nhòe, cùng một mẫu vân tay có thể trông rất khác nhau nên bạn hiếm khi có được hình ảnh trùng khớp hoàn hảo. Ngoài ra, nếu so sánh toàn bộ hình ảnh vân thì sẽ mất rất nhiều năng lượng xử lý, và ai đó có thể dễ dàng đánh cắp dữ liệu vân tay.

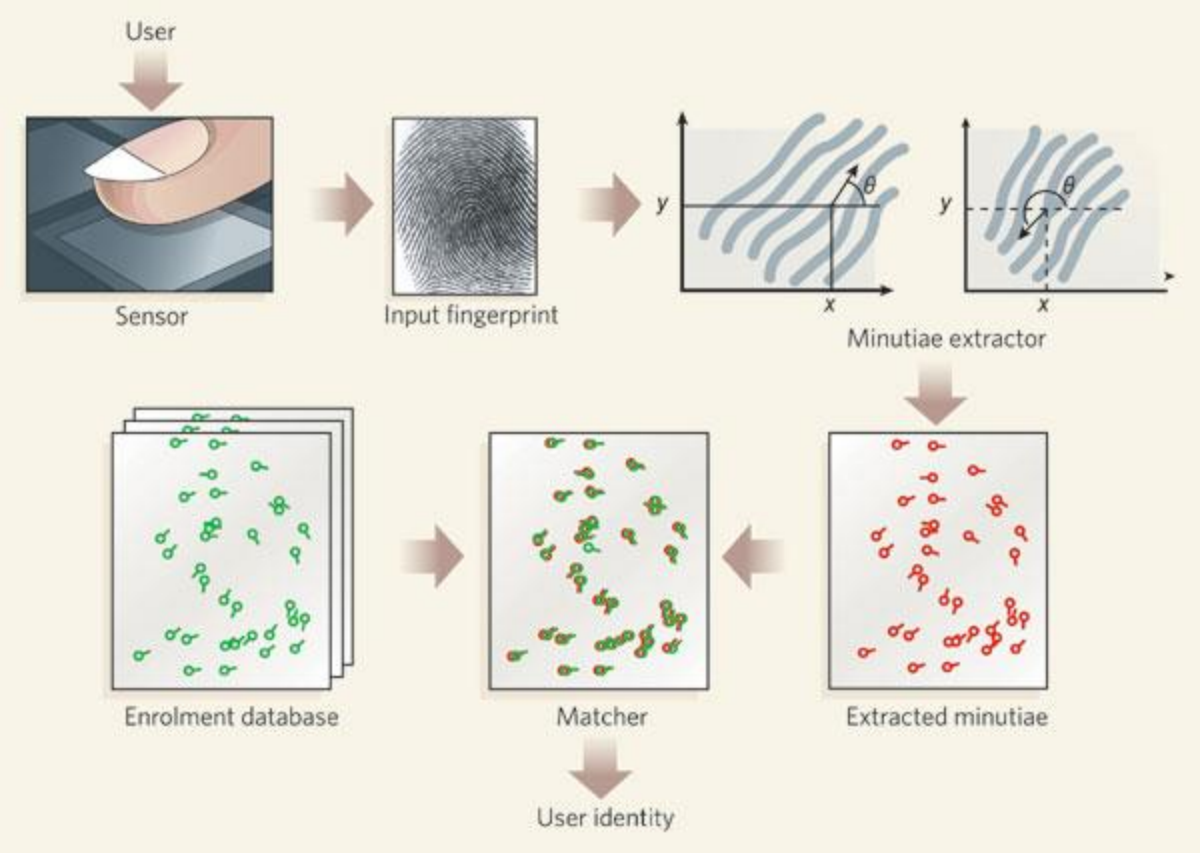
Thay vào đó, hầu hết các máy quét vân tay chỉ so sánh các đặc điểm đặc trưng của vân tay, ví dụ như điểm bắt đầu của đường vân nổi, hay điểm mà đường vân chia tách làm 2 (điểm rẽ nhánh).



Hình 1.4.1 4 điểm đặc trưng của vân tay: Ridge Ending, enclosure, bifurcation, island [2]

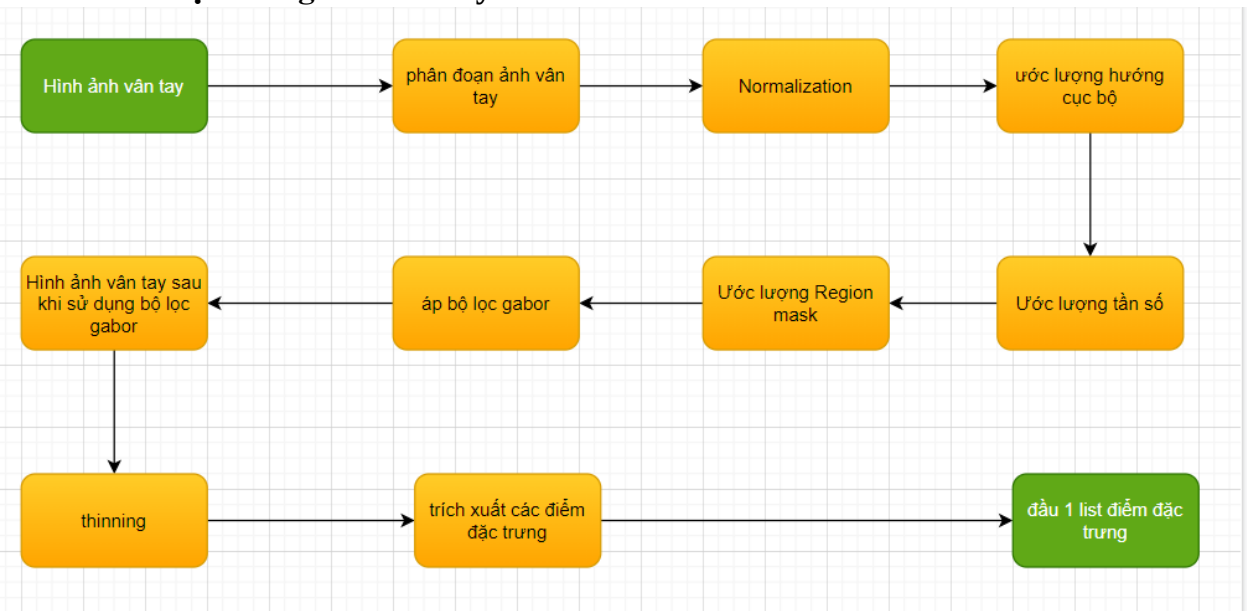
Phần mềm quét thường sử dụng các thuật toán đặc biệt phức tạp để nhận diện và phân tích. Nhiệm vụ cơ bản của chúng là đo đạc vị trí tương đối của các điểm, cũng như bạn nhận diện bầu trời đêm bằng vị trí của các chòm sao vậy. Một cách đơn giản là xem xét hình dạng tạo thành khi nối các điểm lại với nhau. Nếu 2 vân tay có 3 điểm nhỏ và 2 điểm rẽ nhánh, tạo thành một hình giống hệt nhau với cùng kích thước thì nhiều khả năng chúng là một.

Để khớp 2 dấu vân tay, hệ thống không cần tìm toàn bộ tất cả các điểm nối của cả 2 dấu vân tay, mà chỉ cần tìm một số lượng đủ lớn các điểm chung như vậy. Con số này còn tùy thuộc vào từng chương trình quét.



Hình 1.4.2 Hệ Thống đối chiếu vân tay sử dụng điểm đặc trưng [2]

2. Trích rút đặc trưng của vân tay.



Hình 2.1 các giai đoạn trích rút điểm đặc trưng của 1 ảnh vân tay

Như đã đề cập ở phần mục thì tiểu luận xin được sử dụng cách trích rút đặc trưng bằng cách trích rút ra cách điểm kì dị hay nói cách khác là các điểm cụt hoặc điểm rẽ ba của các đường vân. Các giai đoạn để trích rút đặc trưng thể hiện qua hình 2.1. Khi đó thì ảnh đầu vào sẽ được chuẩn hóa ảnh để phục vụ cho việc phân đoạn ảnh. Sau đó hình ảnh tiếp tục được cho qua bộ lọc gabor trước khi cho vào để làm mảnh (thinning). Điểm đặc trưng được trích xuất dựa vào bộ ảnh là ảnh dạng binary.

2.1 Kĩ thuật nâng cao chất lượng ảnh.

2.1.1 Tổng quan về nâng cao chất lượng ảnh dành cho ảnh vân tay.

Nâng cao chất lượng ảnh là 1 bước quan trọng tạo tiền đề cho xử lí ảnh với mục đích là làm nổi bật một số đặc tính quan trọng của ảnh: thay đổi độ tương phản, lọc nhiễm, nổi biên, làm trơn biên, khuyếch đại ảnh, …

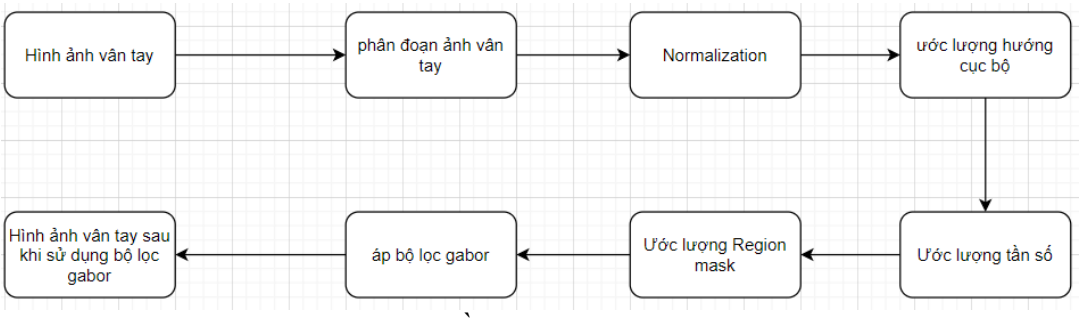
Hình 2.1.1. a) ảnh bên trái là ảnh thô ban đầu b) ảnh bên phải là ảnh sau khi sử dụng bộ lọc gabor

Một yêu cầu quan trọng với hầu hết các kỹ thuật đối chiếu vân tay hiện tại là tất cả chúng đều yêu cầu tiêu chuẩn hình ảnh đầu vào hợp lý. Trong thực tế, một dấu vân tay có thể bị hỏng do nhiễu nhiễu có thể là do thiết bị, tay bị dính bẩn hay ảnh được thu từ nguồn có chất lượng thấp như chụp lại ảnh vân tay ín trên giấy hay lấy lại dấu vân tay ở trên các đồ vật. những điềunày có thể dẫn đến các bức ảnh có chất lượng xấu. hình ảnh xấu có thể làm giảm hiệu quả của các hệ thống. Do đó kỹ thuật tăng cường vân tay được sử dụng để cải thiện hiệu suất. Tăng cường vân tay được sử dụng để khôi phục cấu trúc cấu trúc liên

kết của các đường vân và rãnh từ hình ảnh nhiễu. Nhiều thuật toán tăng cường vân tay tiểu luận xin được phép sử dụng bộ lọc gabor.

2.1.2 Bộ lọc gabor.

2.1.2.1 Thuật toán.



Hình 2.1.2.1.1 Sơ đồ các giai đoạn của bộ lọc gabor

1. Normalization: một bức ảnh thô đầu vào được chuẩn hóa để nó có giá trị trung bình và phương sai mong muốn.
2. Phân đoạn ảnh vân tay: một bức ảnh sau khi chuẩn hóa thì sẽ được phân đoạn vùng nào là vùng chứa vân tay.
3. Ước lược hướng cục bộ: hướng cục bộ của các khối sẽ được ước lượng từ ảnh đã được phân đoạn.
4. Ước lượng tần số cục bộ: tần số của các khối cục bộ được tính từ hình ảnh dấu vân tay đầu vào được chuẩn hóa và hình ảnh định hướng ước tính.
5. ước lượng Region mask: region mask có được bằng cách phân loại mỗi block trong ảnh đầu vào đã được chuẩn hóa thành block có thể tái tạo được hoặc không thể tái tạo được.
6. Bộ lọc gabor: Một bộ lọc Gabor được điều chỉnh theo hướng sườn núi địa phương và tần số sườn được áp dụng cho các pixel sườn núi và thung lũng trong hình ảnh dấu vân tay đầu vào được chuẩn hóa để có được hình ảnh vân tay nổi bật các đường vân.

2.1.2.2 Normalize ảnh đầu vào.

Để tiện cho việc đối chiếu các ảnh thì các ảnh đầu vào phải cùng 1 form đầu vào có thể là size hoặc là độ đậm nhạt của hình ảnh. Chuẩn hóa được sử dụng để chuẩn hóa các giá trị của các pixel trong ảnh bằng cách điều chỉnh phạm vi của các giá trị mức xám để chúng mở rộng trong phạm vi giá trị mong muốn và cải thiện độ tương phản của ảnh. Mục tiêu chính của chuẩn hóa là giảm phương sai của giá trị mức xám dọc theo các đường vân để tạo điều kiện cho các bước xử lý tiếp theo.

Thuật toán normalize:

Đầu vào: im , $Mean_0, VAR_0$

Bước 1: khởi tạo

tính:

- N, M là kích cỡ của bức ảnh im

- $Mean = \frac{1}{N*M}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} im(i,j)$
- $VAR = \frac{1}{N*M}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\left(im(i,j) - Mean\right)^2$
- Im_new = copy(im) // copy lại im lưu vào trong im_new

Bước 2: tính toán lại các giá trị

for i in range(0, N) do

for j in range(0, M) do

$$\text{im\_new[i][j]} = \begin{cases} Mean_0 + \sqrt{\frac{VAR_0\left(im[\text{i}][\text{j}] - \text{Mean}\right)^2}{VAR}} & if\ im[\text{i}][\text{j}] > Mean \\ Mean_0 - \sqrt{\frac{VAR_0\left(im[\text{i}][\text{j}] - \text{Mean}\right)^2}{VAR}} & if\ im[\text{i}][\text{j}] \le Mean \end{cases}$$

endfor;

endfor;

Bước 3: trả lại:

Return im_new

2.1.2.3 Phân đoạn hình ảnh.

Bước này làm bước tiền xử lí lọc bớt nhiễu ở background đi. Ta chỉ cần quan đến khu vực thực sự chứ ảnh vân tay.

Vì đặc điểm của ảnh vân tay nền của ảnh thường chỉ màu đen hoặc trắng nên ta suy ra được ở khu vực background là phương sai hay độ lệch chuẩn sẽ có giá trị thấp. Và ở khu vực có đường vân, cấu trúc của đường vân là đường vân mang giá trị xám (hoặc trắng) và rãnh vân có độ xám ngược so với đường vân và trùng với background nên độ lệch chuẩn (phương sai) ở khu vực này sẽ có giá trị cao. Lợi dụng được tính chất đó tiểu luận xin để xuất lọc bằng cách lấy ngưỡng độ lệch chuẩn (std). vì nếu ảnh vân tay là ảnh đường vân xen kẽ đường rảnh vẫn (cùng độ xám với nền). Vậy nên độ lệch chuẩn ở khu vực này sẽ thường có giá trị cao. Đổi lại nếu khu vực ở vị trí nền thì thường là độ xám màu đen or màu trắng nên suy ra độ lệch chuẩn ở khu vực này thường có giá trị thấp (hay nói cách khác là độ xám ở khu vực này đồng đều không có nhiều sự khác biệt) . do đó chúng ta có thể đặt T = λ*std_{im} . trong đó thì std_{im} là độ lệch chuẩn của cả bức ảnh, λ là ngưỡng.

Thuật toán phân đoạn ảnh:

Đầu vào: im , λ

Bước 1: khởi tạo

tính:

- N, M là kích cỡ của bức ảnh im
- $blocks = block\_slpit(im)$: chia ảnh ra thành các bock 16*16

- im_ $std = std\left(im\right)$ tính độ lệc chuẩn của im
- $blocks\_std = std\left(blocks\right)$
- threshor_ $std = \lambda$*im_std
- threshor_ $std = matrix\_one\left(N, M\right)$ lấy ra ma trận N*M
- Im_segment = copy(im)
- Im_normal = copy(im)

Bước 2: tính toán lại các giá trị

for i in range(0, N) do

for j in range(0, M) do

mask[i][j]= 0 *if* $blocks\_std$[i/ /16][j/ /16] < threshor_ std

endfor;

endfor;

mask = smoot_im(mask) // làm mượt bằng cách giảm nhiễm muối tiêu và lấp đầy điểm trong đường vân.

Im_segment = im_segment*mask

$$Im\_normal = \frac{Im\_normal - mean\left(im\left[mask == 0\right]\right)}{std\left(im\left[mask == 0\right]\right)}$$

Bước 3: trả lại:

Return Im_segment, Im_normal, mask

2.1.2.4 Ước lượng hướng vân tay

Ở bước này tiểu luận xin được trình bày 1 thuật toán ước lượng hướng là thuật toán ước lượng hướng dựa trên bình phương trung bình tối thiểu:

Đầu vào: im: hình ảnh vân tay

Đầu ra: các góc so với phương Ox đối với các block.

Bước 1: chia hình ảnh thành ra các khối nhỏ có kích cỡ w *w (w có thể là 16 hoặc 32)

Bước 2: với mỗi 1 pixel trong block tính Gradient ∂_x và ∂_y

Bước 3: ước lượng hướng của 1 block được tính bằng:

$$V_x = \sum_{v=i-\frac{w}{2}}^{v=i+\frac{w}{2}} \sum_{v=j-\frac{w}{2}}^{v=j+\frac{w}{2}} 2\partial_x(u,v)\partial_y(u,v)$$

$$V_y = \sum_{v=i-\frac{w}{2}}^{v=i+\frac{w}{2}} \sum_{v=j-\frac{w}{2}}^{v=j+\frac{w}{2}} \partial_x^2(u,v)\partial_y^2(u,v)$$

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{V_y(i,j)}{V_x(i,j)}\right)$$

Trong đó $\theta(i,j)$ là ước tính bình phương nhỏ nhất của hướng sườn địa phương tại khối được căn giữa tại pixel (i, j). Về mặt toán học, nó đại diện cho hướng trực giao với hướng chi phối của phổ Fourier của block.

Bước 4:

Do sự hiện diện của nhiễu, cấu trúc sườn núi và rãnh vân bị hỏng, các chi tiết vụn vặt, v.v. trong hình ảnh đầu vào, hướng sườn núi địa phương ước tính, $\theta(i,j)$ có thể không phải lúc nào cũng đúng. Do hướng sườn núi cục bộ thay đổi chậm trong vùng lân cận cục bộ nơi không có điểm kỳ dị xuất hiện, bộ lọc thông thấp có thể được sử dụng để sửa đổi hướng sườn núi cục bộ không chính xác. Để thực hiện lọc thông thấp, hình ảnh định hướng cần được chuyển đổi thành trường vectơ liên tục, được xác định như sau:

$$\phi_x(i,j) = \cos\left(2\theta(i,j)\right), \phi_y(i,j) = \sin\left(2\theta(i,j)\right)$$

Bước 5: tính toán lại hướng của block:

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{\phi_x}{\phi_y}\right)$$

Bước 6: trả lại

Return hướng của các block

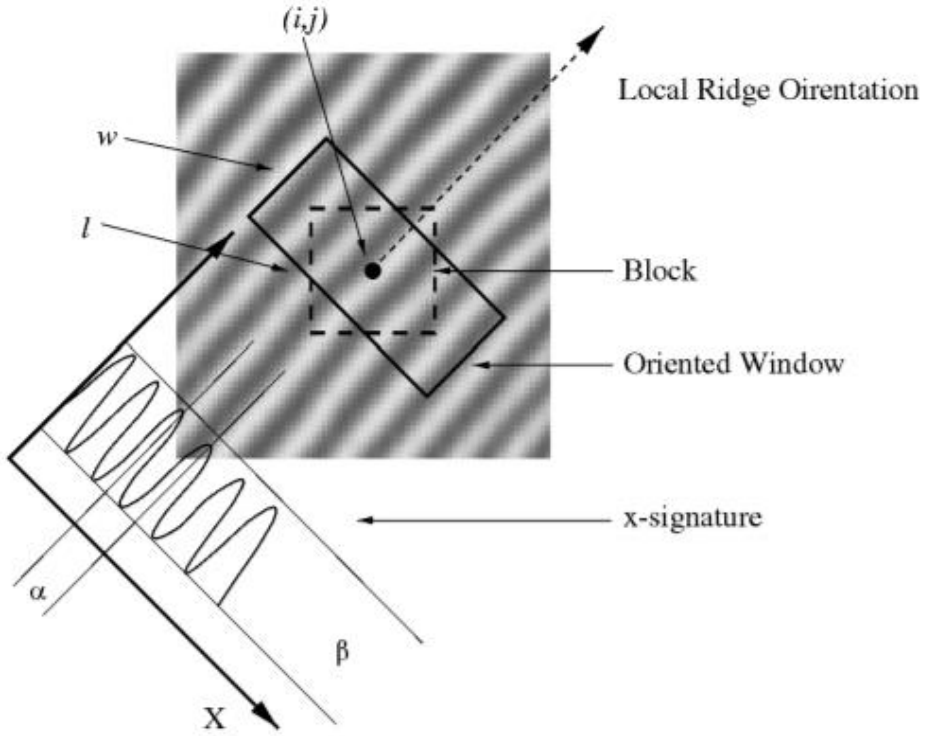
2.1.2.5 Ước lượng tần số.

Để ước lượng tần số địa phương, một cửa sổ tần số địa phương được đưa ra Các bước liên quan đến việc ước lượng tần số, theo Jain A., Ross A., Prabhakar S. các đường vân địa phương được trình bày như sau:

1- Chia ảnh thành các khối ảnh kích thước w × w, sao cho tâm tại điểm ảnh (i, j).

2 - Tính các cửa sổ tần số địa phương kích thước w × l, sao cho tâm tại điểm ảnh (i, j), thỏa mức bình thường để định hướng địa phương.

3- Đối với mỗi khối có tâm ở pixel (i, j), hãy tính X signature, X [0], X [1], ... X [l - 1], của các đường vân và thung lũng trong cửa sổ định hướng, trong đó



hình ảnh 2.1.5.1 thể hiện cách tính cách tín hiệu X [0], X [1], ... X [l - 1] [5]

$$X[k] = \frac{1}{w}\sum_{d=0}^{w-1} G(u, v),$$

$$u = i + \left(d - \frac{w}{2}\right)\cos O(i, j) + \left(k - \frac{l}{2}\right)\sin O(i, j),$$

$$v = j + \left(d - \frac{w}{2}\right)\sin O(i, j) + \left(\frac{l}{2} - k\right)\cos O(i, j).$$

Với θ(i, j) là định hướng địa phương và G(u,v) là hình ảnh bình thường. Các X [0], X [1], ..., X [λ -1] tạo thành một dạng sóng hình sin rời rạc, trong đó có cùng tần số như của các đường vân và các chỗ lõm trong cửa sổ tần số Đặt T(i, j) là số điểm ảnh giữa hai đỉnh liên tiếp (hình 2.1.5.2).

Bằng cách sử dụng các thông tin về sự định hướng và tần số, bộ lọc Gabor cho thấy có hiệu suất chất lượng hình ảnh địa phương tốt hơn là chất lượng hình ảnh định hướng trong phép lọc định hướng. Tuy nhiên, cách tính phức tạp của bộ lọc Gabor không thể đáp ứng yêu cầu về thời gian thực hiện trong hệ thống nhận dạng vân tay tự động.

4 - Đối với hình ảnh dấu vân tay được quét ở độ phân giải cố định, giá trị tần số của các đường vân và thung lũng trong một khu phố địa phương nằm trong một phạm vi nhất định. Đối với hình ảnh 500dpi, phạm vi này là [1/3, 1/25]. Do đó, nếu giá trị ước tính của tần số nằm ngoài phạm vi này, thì tần số được gán giá trị -1 để chỉ ra rằng tần số hợp lệ không thể có được.

5 - Các khối trong đó các chi tiết vụn vặt và các điểm kỳ dị xuất hiện và hoặc các rặng núi và thung lũng bị hỏng không tạo thành một sóng hình sin được xác định rõ. Các giá trị tần số cho các khối này cần được nội suy từ tần số của các khối lân cận có tần số được xác định rõ. chi tiết được mô tả như sau [6]:

For each block centered at (i, j) do

$$\Omega'[i][j]=\begin{cases}\Omega'[i][j] & \text{if } \Omega[i][j]=1\\ \dfrac{\sum_{u=-w_{\Omega/2}}^{w_{\Omega/2}}\sum_{v=-w_{\Omega/2}}^{w_{\Omega/2}} W_g(u,v)\,\mu\big(\Omega(i-uw, jvw)\big)}{\sum_{u=-w_{\Omega/2}}^{w_{\Omega/2}}\sum_{v=-w_{\Omega/2}}^{w_{\Omega/2}} W_g(u,v)\,\delta\big(\Omega(i-uw, jvw)\big)} & \textit{otherwise}\end{cases}$$

endfor;

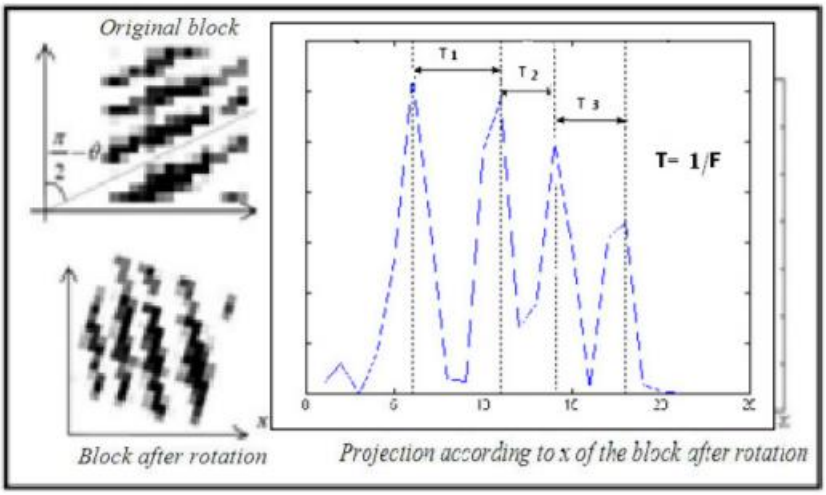
Trong đó:

- $\mu(x)=\begin{cases}0 & \textit{if } x\le 0\\ x & \textit{otherwise}\end{cases}$
- $\delta(x)=\begin{cases}0 & \textit{if } x\le 0\\ 1 & \textit{otherwise}\end{cases}$
- W_g là bộ lọc Gaussian rời rạc với giá trị trung bình và phương sai bằng 0 và 9 , và $w_\Omega = 7$ là kích cỡ của bộ lọc

6 - Khoảng cách Interridge thay đổi chậm trong một khu vực địa phương. Một bộ lọc thông thấp có thể được sử dụng để loại bỏ các ngoại lệ:

$$F[i][j]=\sum_{u=-w_{l/2}}^{w_{l/2}}\sum_{v=-w_{l/2}}^{w_{l/2}} W_l(u,v)\,\Omega'(i-uw, jvw)$$

Trong đó W_l là bộ lọc thông thấp hai chiều với tích phân đơn vị và với $w_l = 7$ là kích cỡ của bộ lọc



Hình 2.1.5.2 Trích xuất bước sóng sau khi quay phương đường vân song song với trục oy

2.1.2.6 Ước lượng Region Mask.

Một pixel (hoặc một khối) trong hình ảnh dấu vân tay đầu vào có thể ở trong vùng có thể phục hồi hoặc vùng không thể phục hồi. Việc phân loại pixel thành các loại có thể phục hồi và không thể phục hồi có thể được thực hiện dựa trên đánh giá hình dạng của sóng được hình thành bởi các rặng núi và thung lũng địa phương. Trong thuật toán của chúng tôi, ba tính năng được sử dụng để mô tả sóng hình sin: biên độ (a), tần số (b) và phương sai (g). Đặt X [1], X [2], ..., X [l] là chữ ký x của một khối có tâm tại (i, j). Ba tính năng tương ứng với pixel (khối) (i, j) được tính như sau:

1- α (chiều cao trung bình của các đỉnh - độ sâu trung bình của các thung lũng).

2- $\beta = \frac{1}{T(i,j)}$ Trong đó T (i, j) là số pixel trung bình giữa hai đỉnh liên tiếp

3- $\gamma = \frac{1}{l}\sum_{i=1}^{l}\left(X[\mathrm{i}]-\left(\frac{1}{l}\sum_{i=1}^{l} X[\mathrm{i}]\right)\right)^2$

Thuật toán cố gắng chọn một số hình ảnh dấu vân tay điển hình trong đó cả hai khu vực có thể phục hồi và không thể phục hồi được gắn nhãn thủ công và tính toán ba tính năng cho các khu vực đó. Sau đó thuật toán sử dụng KNN để xác định xem 1 khu vực có phải là khu vực phục hồi hay không ? thuật toán được thực hiện với K = 1 và gán nhãn vào $\Re(i,j)=1$ nếu điểm I,j không thuộc đường vân ngược lại $\Re(i,j)=0$.

2.1.2.7 Áp dụng gabor.

Các cấu hình của các đường vân và thung lũng song song với tần số và định hướng được xác định rõ trong hình ảnh dấu vân tay cung cấp thông tin hữu ích giúp loại bỏ nhiễu không mong muốn. Các sóng hình sin của các rặng núi và thung lũng thay đổi chậm theo hướng không đổi cục bộ. Do đó, bộ lọc thông dải được điều chỉnh theo tần số và hướng tương ứng có thể loại bỏ hiệu quả nhiễu không mong muốn và giữ cấu trúc sườn núi và thung lũng. Bộ lọc Gabor có cả hai thuộc tính chọn lọc tần số và định hướng và có độ phân giải khớp tối ưu trong cả hai miền không gian và tần số [7], [8]. Do đó, thích hợp sử dụng

bộ lọc Gabor làm bộ lọc thông dải để loại bỏ nhiễu và bảo tồn cấu trúc sườn núi / thung lũng.

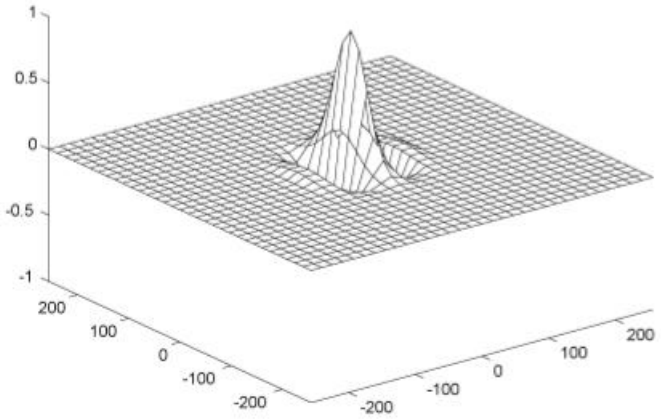
Bộ lọc Gabor có dạng chung là [8]:

$$h(x, y, \phi, f) = e^{-\frac{1}{2}\left(\frac{x_\phi^2}{\delta_x^2}+\frac{y_\phi^2}{\delta_y^2}\right)} \cos(2\pi f x\phi)$$

$$x_\phi = x\cos\phi + y\sin\phi$$

$$y_\phi = -x\cos\phi + y\sin\phi$$

Trong đó ϕ là hướng của bộ lọc Gabor, f là tần số của sóng phẳng hình sin và δ_x và δ_y là các hằng số không gian của đường bao Gaussian dọc theo trục x và y. Hàm truyền điều chế (MTF) của bộ lọc Gabor có thể được biểu diễn dưới dạng



Bộ lọc gabor với f = 10 và $\phi = 0$

Để áp dụng bộ lọc Gabor cho hình ảnh, phải có 3 tham số ba tham số:

1- Tần số khu vực f
2- Hướng khu vực ϕ
3- Và δ_x và δ_y là các hằng số không gian của đường bao Gaussian dọc theo trục x và y

Cuối cùng bước tính giá trị mức xám ảnh sau khi sử dụng gabor:

$$im\_gabor[i][j] = \begin{cases} 255 & \text{if} \quad \Re(i,j) = 0 \\ \sum_{u=-w_g/2}^{w_g/2} \sum_{v=-w_g/2}^{w_g/2} h(u,v,\theta(i,j),\mathrm{F}(i,j))G(u,v) & \textit{if} \quad \textit{otherwise} \end{cases}$$

2.1.3 Thinning

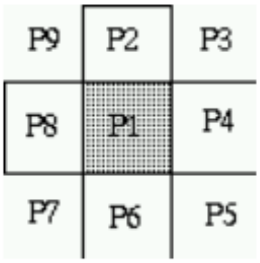
Để có thể làm mỏng được hình ảnh trước tiên phải được tạo thành nhị phân, tức là hình ảnh ở 256 cấp độ màu xám mà chúng ta có ở giai đoạn này được chuyển thành hình ảnh nhị phân trong đó các pixel đen tương ứng với các vệt và Các pixel trắng cho các thung lũng. Có rất nhiều kỹ thuật của binarization hình ảnh [8], tiểu luận sử dụng một phương

pháp ngưỡng đơn giản và hiệu quả. Để thực hiện quá trình xử lý này, giá trị của mỗi pixel P (x, y) được so sánh với ngưỡng M và nếu giá trị này lớn hơn ngưỡng thì pixel lấy giá trị của một (màu đen), nếu không, nó sẽ lấy giá trị bằng 0 (trắng).

Để tạo điều kiện trích xuất các điểm đặc trưng, hình ảnh phải được cấu trúc: các đường vân chỉ có độ rộng tối đa là 1 pixel (Điều đó không có nghĩa là các đường vân sẽ bị đứt đoạn mà vẫn phải dữ được cấu trúc là 1 đường). Tiểu luận đã sử dụng thuật toán Rosenfeld [10] vì tính đơn giản của nó và vì nó được áp dụng tốt khi triển khai phần cứng vì nó có thời gian tính toán giảm so với các thuật toán khác. Việc sử dụng thuật toán Rosenfeld cho phép tối ưu hóa thời gian xử lý chung.

Hình 2.1.3.1 ảnh sau khi sử dụng thinning

Thuật toán được mô tả ở đây là một thuật toán song song đơn giản do Rosenfeld đề xuất . Thuật toán này hoạt động bằng cách loại bỏ liên tiếp trong một tập hợp song song của "ranh giới" của P. Trước tiên, chúng tôi xác định từ của ranh giới. Đặt P là, tập hợp các pixel (một) màu đen. Phần bù của P là nền (các pixel như vậy có màu trắng (zero's)). Hãy xem xét một pixel p1 của P. Thuật toán sử dụng filter bank với điểm P1 với các trường hợp sảy ra như sau:



Hình ảnh 2.1.3.1 quan hệ P1 và các 8 điểm lân cận

1- p1 là điểm north border nếu p2 = 0.
2- p1 là điểm east border nếu p4 = 0.
3- p1 là điểm west border nếu p8 = 0.
4- p1 là điểm south border nếu p6 = 0.
5- p1 là điểm 4-endpoint nếu chính xác một trong 4 người hàng xóm của nó là điểm đen
6- p1 là điểm 8-endpoint với điều kiện chính xác một trong 8 người hàng xóm của mình là điểm màu đen
7- p1 là điểm 4-isolated nếu không có ai trong số 4 hàng xóm của nó là màu đen.
8- p1 là điểm 8-isolated nếu không có hàng xóm 8 của nó là màu đen.
9- p1 là điểm bên trong 4-simple nếu thay đổi nó từ đen sang trắng (một thành 0) sẽ không làm thay đổi kết nối 4 của các pixel đen còn lại trong vùng lân cận của p1
10- Điểm viền p1 là 8-simple nếu thay đổi nó từ đen thành trắng (một thành 0) không làm thay đổi kết nối 8 của các pixel đen còn lại trong vùng lân cận của p1.

Thuật toán làm mảnh:

Bắt đầu

Lặp lại cho đến khi không có pixel nào bị thay đổi từ đen sang trắng:

Bước 1: Thay đổi tất cả các điểm north borther từ đen thành trắng, những điểm là 4-simple chứ không phải là 4-isolated hay 4-endpiont.

Bước 2: Thay đổi tất cả các điểm điểm south borther từ đen thành trắng, những điểm là 4-simple chứ không phải là 4-isolated hay 4-endpiont.

Bước 3: Thay đổi tất cả các điểm east borther từ đen thành trắng, những điểm là 4-simple chứ không phải là 4-isolated hay 4-endpiont.

Bước 4: Thay đổi tất cả các điểm west borther từ đen thành trắng, những điểm là 4-simple chứ không phải là 4-isolated hay 4-endpiont.

Kết thúc lặp lại

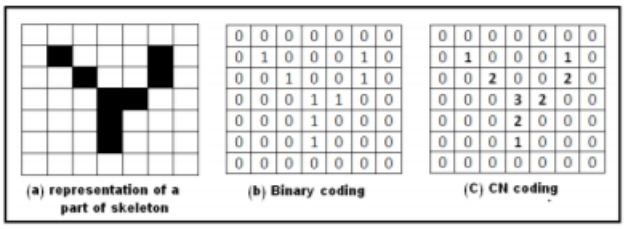
Kết thúc

2.1.4 Trích xuất điểm đặc trưng.

2.1.4.1 Trích xuất vị trí.

2.1.4.1.1 Trích xuất vị trí của điểm kì dị.

Ảnh sau khi được làm mảnh thì sẽ được trích xuất điểm đặc trưng bằng cách tìm ra 2 điểm đặc trưng là điểm ngã ba và điểm cụt. Phương pháp được sử dụng là Crossing Number (CN) [11]. Nó là phương pháp được sử dụng nhiều nhất vì sự đơn giản của nó. Người ta phải có một hình ảnh khung. Điều này phải có 0 cho pixel trắng và 1 cho pixel đen. Các chi tiết vụn vặt được trích xuất bằng cách kiểm tra vùng lân cận địa phương của từng pixel.



Hình 2.1.4.1.1 thể hiện các biến đổi của 1 đường vân

Công thức tính Crossing Number được mô tả như sau:

$$CN = \sum_{i=0}^{8} \left| P_{(i+1) \bmod 8+1} - P_{i \bmod 8+1} \right|$$

P1	P2	P3
P8	P	P4
P7	P6	P5

Hình 2.1.4.1.2 thể hiện mối quan hệ của điểm P với tám điểm xung quanh

Với những điểm có giá trị:

- CN = 2 thì điểm đó là điểm cụt
- CN = 6 điểm đó là điểm rẽ ba
- CN = 4 điểm đó nằm trên đường vân.

2.1.4.1.2 Loại bỏ điểm kì dị nhiễm.

Đầu tiên loại bỏ tất cả điểm nằm ngoài viền vân tây tuy rằng khi sử dụng Crossing Number thì phát hiện ra các điểm biên là điểm cụt nhưng trong lí thuyết [12] chỉ ra rằng những điểm đó không phải điểm kì dị. Thuật toán để lọc những điểm này cực kì đơn giản ta xét 4 hướng theo chiều x, ngược chiều x, theo chiều y và ngược chiều y có thành viên nào thuộc đường vân hay không? Nếu chỉ tồn 1 hướng mà không có điểm nằm trong đường vân thì ta xác định luôn điểm đó là điểm viền.

Thứ 2 loại bỏ những điểm quá ngắn nhau theo như tiểu luận lấy ngưỡng là khoảng cách bằng 9 những điểm mà có khoảng cách euclit nhỏ hơn hoặc bằng 9 điều được loại bỏ.

2.1.4.2 Trích xuất hướng.

2.1.4.2.1 Trích xuất hướng cho điểm cụt.

Sau khi trích xuất điểm kì dị thì chúng ta sẽ đến giai đoạn tính hướng của đường vân của điểm kì dị đó.

Đối với điểm cụt đơn giản ta chỉ cần lấy 1 mặt na 6*6 lấy điểm cụt là tâm (O) sau đó tìm xung quanh biên của mặt nạ điểm nào là điểm cuối (A) rồi ta tính vector $\overrightarrow{OA}$ khi đó:

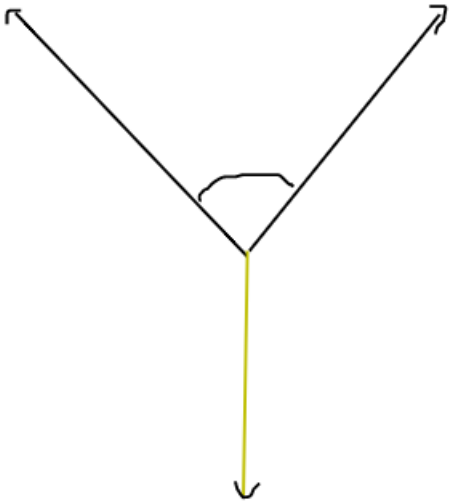
$$\cos\theta = \frac{x_{\overrightarrow{OA}}}{\left\|\overrightarrow{OA}\right\|} \Rightarrow \theta = ar\cos\left(\frac{x_{\overrightarrow{OA}}}{\left\|\overrightarrow{OA}\right\|}\right)$$

Tuy nhiên việc tính θ như thế thì lúc đó θ chỉ thể hiện phương của đường vân điều chúng ta đang cần đó là hướng vậy nên :

$$\theta = \begin{cases} ar\cos\left(\frac{x_{\overrightarrow{OA}}}{\left\|\overrightarrow{OA}\right\|}\right) & if\ y_A > y_O \\ ar\cos\left(\frac{x_{\overrightarrow{OA}}}{\left\|\overrightarrow{OA}\right\|}\right) + \pi & otherwise \end{cases}$$

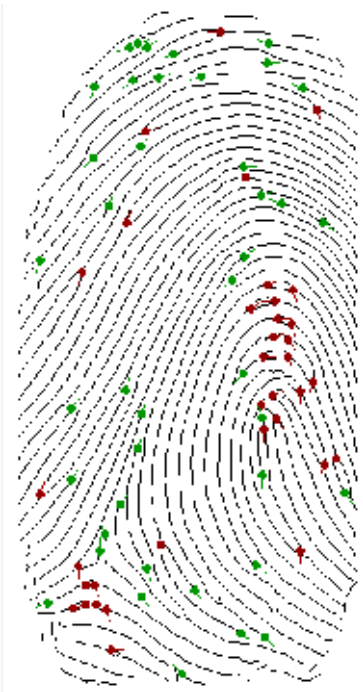
2.1.4.2.2 Trích xuất hướng cho điểm ngã ba.

Điều đáng lo ngại ở đây điểm ở ngã ba không biết chọn theo hướng của đường nào? Vậy tiểu luận xin đề xuất lấy đường mà đối góc đối diện của đường đó qua điểm ngã ba là nhỏ nhất.



Hình 2.1.4.2.2.1 ví dụ về cách lấy đường làm đường lấy hướng cho điểm ngã ba.

Sau khi loại được 2 đường bên rồi thì bài toán tìm hướng lại quay lại bài toán tìm hướng cho điểm cụt.



Hình 2.1.4.2.2.2 điểm đặc trưng và hướng của nó.

Sau khi trích xuất hướng của điểm đặc trưng thì lúc đó 1 vân tay sẽ có danh sách nhiều điểm đặc mỗi một loại điểm đặc trưng có 3 tham số:

- Loại điểm đặc trưng 0 là điểm cụt 1 là điểm ngã ba.
- Tọa độ điểm đặc trưng (y,x)
- Hướng điểm đặc trưng đại diện là θ

2.2 Matching (đối chiếu)

Các điểm minutiae là khi dò theo đường vân sẽ thấy những điểm đường vân kết thúc (Ridge Ending) hoặc rẽ nhánh (Bifuraction)

Vân tay được biểu diễn với 1 vector đặc trưng có 3 tham số (x, y, θ)

Tương ứng với vị trí x, y và góc (hướng) của điểm minuiae trên vân tay đó

Số lượng các điểm minutiae trong 1 vân tay là biến thiên

Bài toán đối sánh dựa vào đặc trưng được phát biểu thành:

+ Input: Ảnh đầu vào I

+ Output: Ảnh T trong cơ sở dữ liệu có số điểm minutiae so khớp với nhau nhiều nhất

Ta có:

$$T = \{ m_1, m_2, \ldots m_m \}; m_I = \{ x_i, y_i, \theta_i \}, \qquad i = 1 \ldots m$$

$$I = \{ m_1', m_2' \ldots m_n' \}; m_j' = \{ x_j', y_j', \theta_j' \} \qquad j = 1 \ldots n$$

Trong đó m, n là số các chi tiết (minutiae) trong ảnh T và I

Một chi tiết trong I và một chi tiết trong T được xem là so khớp nếu:

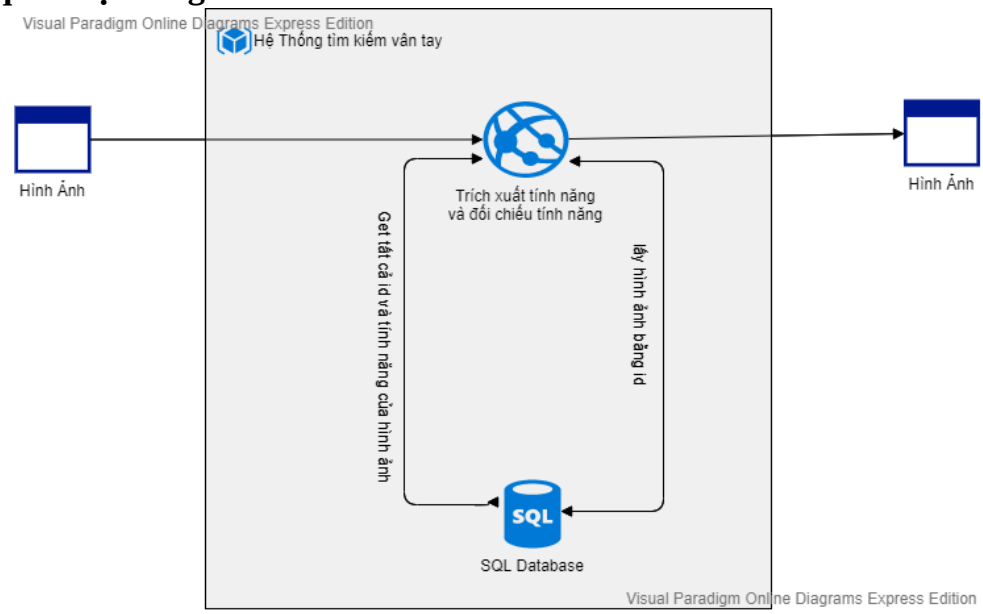
$$sd(m_j^{'}, m_i) = \sqrt{\left(x_j^{'} - x_i\right)^2 + \left(y_j^{'} - y_i\right)^2} \le r_0$$

$$dd(m_j^{'}, m_i) = \min\left(|\theta_j^{'} - \theta_i|, 360^{\circ} - |\theta_j^{'} - \theta_i|\right) \le \theta_0$$

Lấy min độ chênh lệch 2 góc và phần bù của độ chênh lệch 2 góc vì góc 358 độ và góc 2 độ chỉ chênh lệch nhau 4 độ

3. Hệ thống tìm kiếm vân tay

3.1 Tổng quan hệ thống



Hình 3.1 hệ thống tìm kiếm vân tay giống nhất.

Hệ thống tìm kiếm và đối chiếu vân tay thực hiện các công việc lưu trữ hình ảnh vân tay, trích xuất tính năng, thực hiện việc đối chiếu vân tay qua các đặc trưng của vân tay (các điểm đặc trưng) sau đó kết quả trả về 1 id vân tay có đặc điểm (đặc trưng) giống bức ảnh đầu vào nhất sau đó truy cập vào cơ sở dữ liệu lấy ra hình ảnh có id mà hệ thống yêu cầu. Sau đó hệ thống sẽ trả lại hình ảnh giống nhất với hình ảnh vừa mới cho vào.

3.2 Soucre code của hệ thống.

3.2.1 Normalize ảnh.

```
from math import sqrt
import numpy as np


def normalize_pixel(x, v0, v, m, m0):
    dev_coeff = sqrt((v0 * ((x - m)**2)) / v)
    return m0 + dev_coeff if x > m else m0 - dev_coeff

def normalize(im, m0, v0):
    m = np.mean(im)
    v = np.std(im) ** 2
    (y, x) = im.shape
    normilize_image = im.copy()
    for i in range(x):
        for j in range(y):
            normilize_image[j, i] = normalize_pixel(im[j, i], v0, v, m, m0)

    return normilize_image
```

3.2.2 Phân đoạn hình ảnh.

```
import numpy as np
import cv2 as cv
import math

def normalise(img):
    return (img - np.mean(img))/(np.std(img))


def create_segmented_and_variance_images(im, w, threshold=.3):
    (y, x) = im.shape
    threshold = np.std(im)*threshold

    image_variance = np.zeros(im.shape)
    segmented_image = im.copy()
    mask = np.ones_like(im)
    for i in range(0, x, w):
        for j in range(0, y, w):
            box = [i, j, min(i + w, x), min(j + w, y)]
            block_stddev = np.std(im[box[1]:box[3], box[0]:box[2]])
            image_variance[box[1]:box[3], box[0]:box[2]] = block_stddev

    # loc theo nguong threshold
    mask[image_variance < threshold] = 0
    # lam muot anh
    kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE,(w*2, w*2))
    mask = cv.morphologyEx(mask, cv.MORPH_OPEN, kernel)
    mask = cv.morphologyEx(mask, cv.MORPH_CLOSE, kernel)

    # normalize segmented image
    segmented_image *= mask
    im = normalise(im)
    mean_val = np.mean(im[mask==0]) # trung binh diem anh la nhieu
    std_val = np.std(im[mask==0])# tinh do lech chuan
    if  math.isnan(mean_val):
        mean_val = 0
        std_val = 1
    norm_img = (im - mean_val)/(std_val)

    return segmented_image, norm_img, mask
```

3.2.3 Ước lượng hướng vân tay.

```
import numpy as np
import cv2 as cv
def calculate_angles(im, W):
    """
    :param im:
    :param W: int width of the ridge
    :return: array
    """
    j1 = lambda x, y: 2 * x * y
    j2 = lambda x, y: x ** 2 - y ** 2
    j3 = lambda x, y: x ** 2 + y ** 2

    (y, x) = im.shape

    sobelOperator = [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]
    ySobel = np.array(sobelOperator).astype(np.int)
    xSobel = np.transpose(ySobel).astype(np.int)

    result = [[] for i in range(1, y, W)]

    Gx_ = cv.filter2D(im/125,-1, ySobel)*125
    Gy_ = cv.filter2D(im/125,-1, xSobel)*125

    for j in range(1, y, W):
        for i in range(1, x, W):
            nominator = 0
            denominator = 0
            for l in range(j, min(j + W, y - 1)):
                for k in range(i, min(i + W , x - 1)):
                    Gx = round(Gx_[l, k])  # horizontal gradients at l, k
                    Gy = round(Gy_[l, k])  # vertial gradients at l, k
                    nominator += j1(Gx, Gy)
                    denominator += j2(Gx, Gy)

            if nominator or denominator:
                angle = (math.pi + math.atan2(nominator, denominator)) / 2
                orientation = np.pi/2 + math.atan2(nominator,denominator)/2
                result[int((j-1) // W)].append(angle)
            else:
                result[int((j-1) // W)].append(0)

    result = np.array(result)

    return result
```

3.2.4 Ước lượng tần số.

```
import numpy as np
import math
import scipy.ndimage


def frequest(im, orientim, kernel_size, minWaveLength, maxWaveLength):
    """
    """
    rows, cols = np.shape(im)

    cosorient = np.cos(2*orientim) # np.mean(np.cos(2*orientim))
    sinorient = np.sin(2*orientim) # np.mean(np.sin(2*orientim))
    block_orient = math.atan2(sinorient,cosorient)/2

    rotim = scipy.ndimage.rotate(im,block_orient/np.pi*180 + 90,axes=(1,0),reshape = False,order = 3,mode = 'nearest')


    cropsze = int(np.fix(rows/np.sqrt(2)))
    offset = int(np.fix((rows-cropsze)/2))
    rotim = rotim[offset:offset+cropsze][:,offset:offset+cropsze]


    ridge_sum = np.sum(rotim, axis = 0)
    dilation = scipy.ndimage.grey_dilation(ridge_sum, kernel_size, structure=np.ones(kernel_size))
    ridge_noise = np.abs(dilation - ridge_sum); peak_thresh = 2;
    maxpts = (ridge_noise < peak_thresh) & (ridge_sum > np.mean(ridge_sum))
    maxind = np.where(maxpts)
    _, no_of_peaks = np.shape(maxind)

    if(no_of_peaks<2):
        freq_block = np.zeros(im.shape)
    else:
        waveLength = (maxind[0][-1] - maxind[0][0])/(no_of_peaks - 1)
        if waveLength>=minWaveLength and waveLength<=maxWaveLength:
            freq_block = 1/np.double(waveLength) * np.ones(im.shape)
        else:
            freq_block = np.zeros(im.shape)
    return(freq_block)

def ridge_freq(im, mask, orient, block_size, kernel_size, minWaveLength, maxWaveLength):
    # Function to estimate the fingerprint ridge frequency across a
    # fingerprint image.
    rows,cols = im.shape
    freq = np.zeros((rows,cols))

    for row in range(0, rows - block_size, block_size):
        for col in range(0, cols - block_size, block_size):
            image_block = im[row:row + block_size][:, col:col + block_size]
            angle_block = orient[row // block_size][col // block_size]
            if angle_block:
                freq[row:row + block_size][:, col:col + block_size] = frequest(image_block, angle_block, kernel_size,
                                                                               minWaveLength, maxWaveLength)

    freq = freq*mask
    freq_1d = np.reshape(freq,(1,rows*cols))
    ind = np.where(freq_1d>0)
    ind = np.array(ind)
    ind = ind[1,:]
    non_zero_elems_in_freq = freq_1d[0][ind]
    medianfreq = np.median(non_zero_elems_in_freq) * mask

    return medianfreq
```

3.2.5 Ước lượng Region Mask và Gabor.

```
def gabor_filter(im, orient, freq, kx=0.65, ky=0.65):
    angleInc = 3
    im = np.double(im)
    rows, cols = im.shape
    return_img = np.zeros((rows,cols))

    freq_1d = freq.flatten()
    frequency_ind = np.array(np.where(freq_1d>0))
    non_zero_elems_in_freq = freq_1d[frequency_ind]
    non_zero_elems_in_freq = np.double(np.round((non_zero_elems_in_freq*100)))/100
    unfreq = np.unique(non_zero_elems_in_freq)


    sigma_x = 1/unfreq*kx
    sigma_y = 1/unfreq*ky
    block_size = np.round(3*np.max([sigma_x,sigma_y])).astype('int')
    array = np.linspace(-block_size,block_size,(2*block_size + 1))
    x, y = np.meshgrid(array, array)

    reffilter = np.exp(-(((np.power(x,2))/(sigma_x*sigma_x) + (np.power(y,2))/(sigma_y*sigma_y)))) * np.cos(2*np.pi*unfreq[0]*x)
    filt_rows, filt_cols = reffilter.shape
    gabor_filter = np.array(np.zeros((180//angleInc, filt_rows, filt_cols)))


    for degree in range(0,180//angleInc):
        rot_filt = scipy.ndimage.rotate(reffilter,-(degree*angleInc + 90),reshape = False)
        gabor_filter[degree] = rot_filt

    maxorientindex = np.round(180/angleInc)
    orientindex = np.round(orient/np.pi*180/angleInc)
    for i in range(0,rows//16):
        for j in range(0,cols//16):
            if(orientindex[i][j] < 1):
                orientindex[i][j] = orientindex[i][j] + maxorientindex
            if(orientindex[i][j] > maxorientindex):
                orientindex[i][j] = orientindex[i][j] - maxorientindex

    block_size = int(block_size)
    valid_row, valid_col = np.where(freq>0)
    finalind = \
        np.where((valid_row>block_size) & (valid_row<rows - block_size) & (valid_col>block_size) & (valid_col<cols - block_size))

    for k in range(0, np.shape(finalind)[1]):
        r = valid_row[finalind[0][k]]; c = valid_col[finalind[0][k]]
        img_block = im[r-block_size:r+block_size + 1][:,c-block_size:c+block_size + 1]
        return_img[r][c] = np.sum(img_block * gabor_filter[int(orientindex[r//16][c//16]) - 1])

    gabor_img = 255 - np.array((return_img < 0)*255).astype(np.uint8)

    return gabor_img
```

3.2.6 Thinning.

```
import numpy as np
import cv2 as cv
from skimage.morphology import skeletonize as skelt
from skimage.morphology import thin
def skeletonize(image_input):
    """
    :param image_input: 2d array uint8
    :return:
    """
    image = np.zeros_like(image_input)
    image[image_input == 0] = 1.0
    output = np.zeros_like(image_input)
    skeleton = skelt(image)

    output[skeleton] = 255
    cv.bitwise_not(output, output)

    return output
```

3.2.7 Trích xuất điểm đặc trưng.

```
import numpy as np
import math
import cv2 as cv
from queue import Queue
from math import acos
import math
# hough
#
pi = acos(-1)
def check_border(biniry_image,x,y):
    (x_max,y_max) = biniry_image.shape
    i = x - 1
    count = 0
    while i >= 0:
        if biniry_image[i][y] == 1:
            count += 1
            break
        i -= 1
    i = x + 1
    while i < x_max:
        if biniry_image[i][y] == 1:
            count += 1
            break
        i += 1
    i = y - 1
    while i >= 0:
        if biniry_image[x][i] == 1:
            count += 1
            break
        i -= 1
    i = y + 1
    while i < y_max:
        if biniry_image[x][i] == 1:
            count += 1
            break
        i += 1
    if count == 4:
        return True
    return False
def get_matrix_pad_zeros(matrix):
    (x,y) = matrix.shape
    res = []
    tmp = np.zeros(y+2)
    res.append(tmp)
    for i in range(x):
        tmp = [0]
        for j in range(y):
            tmp.append(matrix[i][j])
        tmp.append(0)
        res.append( np.array(tmp) )
    res.append(np.zeros(y+2))
    return np.array(res)
def get_line_matrix(biniry_image,point,w = 3):
    res_matrix = np.zeros((2*w+1,2*w+1))
    queue = Queue(maxsize = 10000)
    (x,y) = point
    queue.put(point)
    dx = [-1, -1, -1, 0, 1, 1, 1, 0]
    dy = [-1, 0, 1, 1, 1, 0, -1, -1]

    while queue.empty() != True:
        (get_x,get_y) = queue.get()
        res_matrix[get_x - x + w ][get_y - y + w] = 1
        for i in range(8):
            if biniry_image[get_x + dx[i]][get_y + dy[i]] == 1:

                if abs( get_x + dx[i] - x) <= w and abs(get_y + dy[i]-y) <= w  and res_matrix[get_x + dx[i] - x + w ][get_y+dy[i] - y + w]  == 0:
                    new_point = (get_x + dx[i],get_y + dy[i])
                    queue.put(new_point)
    return res_matrix

def norm_l2(point):
    (x,y) = point
    return math.sqrt(x**2 + y**2)

def scalar_multiplication(point_x , point_y):
    return point_x[0] * point_y[0] + point_x[1] * point_y[1]
def calculation_cos(vector_1, vector_2):
    res = scalar_multiplication(vector_1,vector_2)/(norm_l2(vector_1) * norm_l2(vector_2))
    if res > 1:
        return 1.0
    elif res < -1:
        return -1.0
    else:
        return res
```

```python
def find_point_of_vetor(matrix, minunatiae_tpye):
    (n,m) = matrix.shape
    if minunatiae_tpye == 0:
        i = 0
        for i in range(m):
            if matrix[0][i] == 1:
                return (0,i)
        for i in range(n):
            if matrix[i][m-1] == 1:
                return (i,m-1)
        for i in range(m):
            if matrix[n-1][i] == 1:
                return (n-1,i)
        for i in range(n):
            if matrix[i][0] == 1:
                return (i,0)
        return None
    else:
        points = []
        matrix_pad_zeros = get_matrix_pad_zeros(matrix)
        i = 0
        while i < m:
            if matrix[0][i] == 1 and check_minunatiae_at(matrix_pad_zeros,1,i+1) == 0:
                points.append( (0,i) )
            i += 1
        i = 1
        while i < n:
            if matrix[i][m-1] == 1 and check_minunatiae_at(matrix_pad_zeros,i+1,m) == 0:
                points.append(  (i,m-1) )
            i += 1
        i = m - 2
        while i >  -1:
            if matrix[n-1][i] == 1 and check_minunatiae_at(matrix_pad_zeros,n,i+1) == 0:
                points.append(  (n-1,i) )
            i -= 1
        i = n - 2
        while i >  0:
            if matrix[i][0] == 1 and check_minunatiae_at(matrix_pad_zeros,i+1,1) == 0:
                points.append( (i,0) )
            i -= 1
        # print(str(n) + ' ' + str(m))
        centroi = (n//2,m//2)
        min_theta = pi
        res_point = None
        vector_1 = ( points[0][0] - centroi[0], points[0][1] - centroi[1] )
        vector_2 = ( points[1][0] - centroi[0], points[1][1] - centroi[1] )

        tmp_theta = acos(calculation_cos(vector_1, vector_2))
        if tmp_theta < min_theta:
            res_point = points[2]
            min_theta = tmp_theta

        # print(matrix)
        vector_1 = ( points[1][0] - centroi[0], points[1][1] - centroi[1] )
        vector_2 = ( points[2][0] - centroi[0], points[2][1] - centroi[1] )
        # print(points[0])
        # print(points[1])
        # print(points[2])
        tmp_theta = acos(calculation_cos(vector_1, vector_2))
        if tmp_theta < min_theta:
            res_point = points[0]
            min_theta = tmp_theta


        vector_1 = ( points[2][0] - centroi[0], points[2][1] - centroi[1] )
        vector_2 = ( points[0][0] - centroi[0], points[0][1] - centroi[1] )
        tmp_theta = acos(calculation_cos(vector_1, vector_2))
        if tmp_theta < min_theta:
            res_point = points[0]
            min_theta = tmp_theta

        return res_point
```

```
def get_orient(point, minunatiae_tpye, matrix,orientation):

    point_two = find_point_of_vetor(matrix,minunatiae_tpye)
    if  point_two == None:
        print('loi tim diem thu 2')
        return None
    (y,x) = point

    (x_centroi,y_centroi) = matrix.shape
    x_centroi = x_centroi // 2
    y_centroi = y_centroi // 2
    # print(' ass '+  str (point_two ))
    vector_1 = (0,1)
    vector_2 = (x_centroi - point_two[0], y_centroi - point_two[1])
    # print(vector_2)

    theta = acos(calculation_cos(vector_1, vector_2))
    # print(theta)
    if vector_2[0] >= 0:
        # print(vector_2[0])
        if vector_2[0] == 0:
            if vector_2[1] < 0:
                return pi
            else:
                return 0.0
        else:
            return 2*pi - theta
    else:
        return theta
def check_minunatiae_point(biniry_image,x,y, w = 9):

    (x_max,y_max) = biniry_image.shape
    if x - w < 0 or x + w >= x_max or y - w < 0 or y + w >= y_max:
        return False
    if check_border(biniry_image,x,y) == False:
        return False
    count = 0
    for i in range (-w,w+1):
        for j in range( -w, w+1):
            if i**2 + j** 2 <= 36:
                if check_minunatiae_at(biniry_image, x+ i,y+j) != -1:
                    count += 1
    if count > 1:
        return False

    return True

def check_minunatiae_at(biniry_image,x,y):

    # for i in range(1 , i_max - 1):
    #   for j in range(1, j_max - 1):
    #       print(biniry_image[i][j] , end = ' ')
    #   print()
    # if biniry_image.shape[0] != 160:
    #   print(biniry_image)
    dx = [-1, -1, -1, 0, 1, 1, 1, 0, -1]
    dy = [-1, 0, 1, 1, 1, 0, -1, -1,-1]
    if biniry_image[x][y] == 1:
        sum_number = 0
        for k in range(0,8):
            sum_number += abs(biniry_image[ x + dx[k] ][ y + dy[k]] - biniry_image[ x + dx[k+1]][ y + dy[k+1]]  )
        if sum_number//2 == 1:
            return 0
        if sum_number//2 == 3:
            return 1
    return -1
```

```
def get_minunatiaes_point(im,orientation):
    biniry_image = np.zeros_like(im)
    biniry_image[im<10] = 1.0
    biniry_image = biniry_image.astype(np.int8)
    result = []
    i_max, j_max = biniry_image.shape
    result_im = cv.cvtColor(im, cv.COLOR_GRAY2RGB)
    for i in range(2,i_max-1):
        for j in range(2,j_max-1):
            if biniry_image[i][j] == 1:
                minunatiae_tpye = check_minunatiae_at(biniry_image,i,j)
                if minunatiae_tpye != -1 and check_minunatiae_point(biniry_image.copy(),i,j) == True:
                    # for k1 in range(-2,3):
                    #   for k2 in range(-2,3):
                    #       print(biniry_image[i+k1][j+k2] , end = " ")
                    #   print()
                    #print(str(i) +' '+ str(j))
                    #print('')
                    matrix = get_line_matrix(biniry_image,(i,j))
                    result.append((minunatiae_tpye,(i,j), get_orient((i,j), minunatiae_tpye, matrix, orientation )  ) )
                    if minunatiae_tpye == 0:
                        cv.circle(result_im, (j,i), radius=2, color=(0, 150, 0), thickness=2)
                    else:
                        cv.circle(result_im, (j,i), radius=2, color=(150, 0, 0), thickness=2)

    return result,result_im
```

3.2.8 Matching.

```
def main(path):
    w = 16
    image = read_image_rgb(path)
    print(image.shape)
    image = normalize(image,m0 = float(100), v0 = float(100))
    image_segment,norm_img,mask = create_segmented_and_variance_images(image, w = w, threshold=.4)
    image_oriented = calculate_angles(image_segment,w)
    print(image.shape)
    freq = ridge_freq(norm_img, mask, image_oriented, w, kernel_size = 5, minWaveLength = 5, maxWaveLength = 15)
    gabor_img = gabor_filter(norm_img, image_oriented, freq)
    image_thinning = skeletonize(gabor_img)
    list_point_minunate,result_im = get_minunatiaes_point(image_thinning,image_oriented)
    for i  in list_point_minunate:
        print(i)
    result = []
    for i in list_point_minunate:
        result.append([i[0], i[1][0], i[1][1], i[2]])
    return result

def search_image(I, data):
    no_max = 0
    point = []
    for T in data:
        check = np.zeros(len(T['points']))
        count = 0
        for mi in I:
            for i, mt in enumerate(T['points']):
                (sd, dd) = calculate_distance(mt, mi)
                if check[i] == 0 and mt[0] == mi[0] and sd < 50 and dd < pi/24:
                    count += 1
                    check[i] = 1
        if(no_max < count):
            no_max = count
            point = T
    return (point, no_max)


def path_tokenizer(source, rel):
    source_token = source.split('/')[-2:]
    rel_token = rel.split('/')[-2:]
    if(source_token[0] == rel_token[0] and source_token[1].split('_')[0] == rel_token[1].split('_')[0]):
        return True
    return False
```

4. Kết quả thực nghiệm.

4.1 Kết quả.

Tiểu luận đã sử dụng 235 ảnh làm ảnh nằm trong cơ sở dữ liệu và 78 file làm ảnh đánh giá hệ thống. Kết quả thu được:

- Acuracy: 23/78 đạt 30%.
- Tốc độ search trung bình mỗi bức ảnh 20.10709738731843 milliseconds.

4.2 Đánh giá.

Kết quả thực nghiệm cho khá là thấp. Lí do chính là do tiểu luận vẫn còn áp cách tính duyệt brute force và thực hiện lấy ngưỡng nên thứ nhất ngưỡng đưa ra rất chủ quan, thứ 2 với cùng vâ tay nhưng vị trí khác nhau thì hệ thống không cho kết quả cao. Cuối cùng cũng như việc dịch chuyển vân tay thì nếu cùng 1 vân tay nhưng vân tay bị xoay góc ấn vào thì cũng hệ thống cũng không cho kết quả khả quan.

4.3 Hướng phát triển của đề tài.

Nhận thấy tiểu luận còn nhiều thiếu sót kết quả hệ thống không cao do những nguyên nhân trong mục 4.2 đã đề ra thì tiểu luận muốn phát triển đề tài thêm thuật toán Hough.

Thuật toán Hough là thuật toán tìm ra các giá trị tịnh tiến theo trục x, trục y, góc quay ngược chiều kim đồng hồ va đổ nở của vân (Δx, Δy, θ, s) sao cho khi thực hiện các phép biến đổi đó thực hiện trên tập điểm đặc trưng của vân tay này sẽ thu được tập điểm đặc trưng mới mà có số lượng điểm đặc trưng trùng khớp với tập điểm đặc trưng của vân tay kia là lớn nhất.

Tài liệu tham khảo

[1] Vân Tay theo wikipedia: https://vi.wikipedia.org/wiki/V%C3%A2n_tay

[2] Hỏi đáp về công nghệ vân tay http://www.adel.vn/hoi-dap-ve-cong-nghe/cong-nghe-van-tay/

[3] Lấy Dấu Vân Tay truy tìm tội phạm như thế nào https://vnreview.vn/tin-tuc-khoa-hoc-cong-nghe/-/view_content/content/1594917/lay-dau-van-tay-de-truy-tim-toi-pham-nhu-the-nao

[4] NGHIÊN CỨU MỘT SỐ GIẢI THUẬT PHÂN TÍCH ĐẶC TRƯNG CỦA VÂN TAY VÀ THỬ NGHIỆM TRONG NHẬN DẠNG VÂN TAY – Nguyễn Thị Huệ 5

[5] Fingerprint Image Enhancement: Algorithm and Performance - Evaluation Lin Hong, Student Member, IEEE, Yifei Wan, and Anil Jain, Fellow, IEEE

[6] G. Sapiro and A. Bruckstein, “A B-Spline Based Affine Invariant Multiscale Shape Representation,” Proc. Seventh Int’l Conf. Image Analysis and Processing III, pp. 20-22, Monopoli, Italy, 1993

[7] J.G. Daugman, “Uncertainty Relation for Resolution in Space, Spatial-Frequency, and Orientation Optimized by TwoDimensional Visual Cortical Filters,” J. Optical Soc. Am., vol. 2, pp. 1,160-1,169, 1985.

[8] A.K. Jain and F. Farrokhnia, “Unsupervised Texture Segmentation Using Gabor Filters,” Pattern Recognition, vol. 24, no. 12, pp. 1,1671,186, 1991.

[9] http://www.referencement-internet-web.com/15777-Passeport-biometrique-empreintes-digitalesnumerisees. html

[10]http://cgm.cs.mcgill.ca/~godfried/student_projects/laleh_pat_rec/rosen_alg.html#:~:text=ALGORITHM%20OF%20ROSENFELD-,The%20Algorithm%20of%20Rosenfeld,define%20the%20word%20of%20boundary.

[11] W. Zhao, R. Chellappa, P.J. Phillips and A.Rosenfeld, Face recognition: A literature survey, ACM Computing Surveys (CSUR), Volume 35, Issue 4, December 2009.

[12] FINGERPRINT RECOGNITION ALGORITHM Farah Dhib Tatar Department of Electrical Engineering, National school of the studies of engineer of Tunis, Tunisia