

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA

NGUYỄN TRỌNG TRÍ

THIẾT KẾ HỆ THỐNG NHẬN DẠNG CỬ CHỈ TAY
SỬ DỤNG CẢM BIẾN KINECT

Chuyên ngành: Kỹ Thuật Điện Tử

Mã số: 60 52 70

LUẬN VĂN THẠC SĨ

TP.HỒ CHÍ MINH, tháng 12 năm 2014

Công trình được hoàn thành tại: **Trường Đại học Bách Khoa-ĐHQG-HCM**

Cán bộ hướng dẫn khoa học:

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 1:

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 2:

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Luận văn thạc sĩ được bảo vệ tại Trường Đại học Bách Khoa, ĐHQG

Tp.HCM ngày tháng năm.

Thành phần Hội đồng đánh giá luận văn thạc sĩ gồm:

(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ luận văn thạc sĩ)

1.....

2.....

3.....

4.....

5.....

Xác nhận của Chủ tịch Hội đồng đánh giá LV và Trưởng Khoa quản lý chuyên ngành sau khi luận văn đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG

TRƯỞNG KHOA ĐIỆN-ĐIỆN TỬ

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên: NGUYỄN TRỌNG TRÍ MSHV: 12140051

Ngày, tháng, năm sinh: 12/03/1988 Nơi sinh: Nghệ An

Chuyên ngành: Kỹ Thuật Điện Tử Mã số: 60 52 70

I. TÊN ĐỀ TÀI: THIẾT KẾ HỆ THỐNG NHẬN DẠNG CỬ CHỈ TAY SỬ DỤNG CẢM BIẾN KINECT

II. NHIỆM VỤ VÀ NỘI DUNG:

NHIỆM VỤ: Thiết kế 1 hệ thống nhận dạng cử chỉ tay sử dụng Kinect có độ chính xác cao, không phụ thuộc vào môi trường và hiển thị cử chỉ thời gian thực.

NỘI DUNG: Tìm hiểu về Kinect, các thư viện xử lý ảnh, máy học vector hỗ trợ. Thiết kế giải thuật để hiện thực hệ thống và đánh giá kết quả của giải thuật đề nghị.

III. NGÀY GIAO NHIỆM VỤ:.....07/07/2014.....

IV. NGÀY HOÀN THÀNH NHIỆM VỤ: 07/12/2014.....

V. CÁN BỘ HƯỚNG DẪN:Tiến Sĩ Trương Quang Vinh.....

.....

Tp. HCM, ngày 07 tháng 12 năm 2014

CÁN BỘ HƯỚNG DẪN

CHỦ NHIỆM BỘ MÔN ĐÀO TẠO

(Họ tên và chữ ký)

(Họ tên và chữ ký)

TRƯỞNG KHOA ĐIỆN ĐIỆN TỬ

(Họ tên và chữ ký)

Lời cảm ơn

Trước hết em xin bày tỏ lòng biết ơn sâu sắc đến **T.S Trương Quang Vinh**, người đã tận tình hướng dẫn, giúp đỡ em trong suốt quá trình thực hiện đề tài.

Tôi xin chân thành cảm ơn các **Thầy Cô** Trường Đại học Bách Khoa TP.Hồ Chí Minh đã trang bị, truyền đạt những kiến thức bổ ích trong suốt thời gian học cao học.

Cảm ơn **gia đình và tất cả bạn bè**, những người đã luôn luôn động viên, giúp đỡ tôi trong suốt quá trình thực hiện đề tài.

Tp.Hồ Chí Minh, tháng 12 năm 2014

Học viên

Nguyễn Trọng Trí

TÓM TẮT

Nhận dạng cử chỉ tay là trong những cách phổ biến mà con người áp dụng ngày càng rộng rãi trong những sản phẩm công nghệ thông tin cần sự tương tác giữa người với máy tính. Các cảm biến hình ảnh 3D được phát triển gần đây, ví dụ Kinect, ngoài nhận ảnh màu, còn có thể nhận dạng chiều sâu của vật thể đã mở ra nhưng cơ hội mới trong việc phát triển những ứng dụng tương tác giữa con người và máy tính. Đề tài “THIẾT KẾ HỆ THỐNG NHẬN DẠNG CỬ CHỈ TAY SỬ DỤNG CẢM BIẾN KINECT” đưa ra một phương pháp nhận dạng cử chỉ tay mới dựa trên hình ảnh độ sâu thu được từ cảm biến. Đầu tiên, vùng bàn tay được lấy ra bằng cách đặt các ngưỡng trên điểm được theo dấu trên bàn tay sử dụng thư viện NITE 2 cung cấp bởi PrimeSense. Bước thứ hai, tìm véc tơ đặc trưng: đặc trưng đầu tiên đề cập đến là số ngón tay được mở ra, đặc trưng thứ hai mô tả góc giữa đầu ngón tay so với phương ngang của bàn tay với tâm ngón tay là gốc, đặc trưng cuối cùng là hiệu của khoảng cách từ tâm của bàn tay đến các ngón tay và bán kính vòng tròn nội tiếp lớn nhất của bàn tay. Cuối cùng, máy vector hỗ trợ (SVM) được áp dụng để nhận ra những cử chỉ khác nhau. Kết quả thực nghiệm đánh giá cho thấy việc nhận dạng chính xác trong thời gian thực.

ABSTRACT

Hand gesture is becoming one of common ways that people use in information technology products needing interaction between people and computer. 3D image sensors are developed recently, e.g. Kinect, not only provide color image, but also depth map. It opened new opportunities in development of human computer interaction (HCI) application. The thesis "HAND GESTURE RECOGNITION USING KINECT SENSOR" shows a novel hand gesture recognition method based on depth image obtained from the sensor. Firstly, the hand region is done by putting thresholds on hand point detected by using NITE 2 library. Secondly, the feature vector is extracted: the first feature mentions to the number of fingers opened at the shown gesture, second feature describes the angles between the fingertips from horizontal of the hand where hand center is original coordination, the last feature is the subtractions of the distances from the hand center to the fingertips and the radius of the biggest inscribed circle. Finally, a support vector machine (SVM) is applied to identify different gestures. The experimental result shows that the proposed method perform accurately with pleasing quality in real-time.

LỜI CAM ĐOAN

Tôi xin cam đoan ý tưởng giải thuật và kết quả hiện thực đề tài chưa được công bố trong bất kì công trình khoa học trước đây.

Người cam đoan

Nguyễn Trọng Trí

MỤC LỤC

CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI	12
1.1 Giới thiệu đề tài	12
1.2 Tình hình nghiên cứu	13
1.3 Mục tiêu luận văn và giới hạn đề tài	15
1.4 Bố cục luận văn	15
CHƯƠNG II: CƠ SỞ LÝ THUYẾT	16
2.1 Cảm biến Kinect và các thư viện hỗ trợ	16
2.1.1 Cảm biến Kinect	16
2.1.2 Các thư viện xử lý ảnh sử dụng cho lập trình với Kinect	19
2.2 Máy học vector hỗ trợ (Support Vector Machine)	21
2.2.1 SVM trong trường hợp mẫu phân hoạch tuyến tính	22
2.2.2 SVM trong trường hợp mẫu không phân hoạch tuyến tính	22
2.2.3 Các phương pháp phân loại SVM	28
CHƯƠNG III: HỆ THỐNG NHẬN DẠNG CỬ CHỈ TAY ĐỀ XUẤT	30
3.1 Tổng quan hệ thống đề xuất	30
3.2 Phương pháp trích ảnh bàn tay và tiền xử lý	31
3.2.1 Phát hiện bàn tay	31
3.2.2 Lọc hình thái và làm trơn ảnh	34
3.3 Phương pháp trích đặc trưng bàn tay	34
3.3.1 Tìm đường bao bàn tay	34
3.3.2 Tìm phương bàn tay và trọng tâm bàn tay	36
3.3.3 Tìm tâm và bán kính lòng bàn tay	37
3.3.4 Tìm các đầu ngón tay	38
3.4 Phương pháp phân loại cử chỉ	39
CHƯƠNG IV: KẾT QUẢ THỰC HIỆN GIẢI THUẬT	44

4.1	Kết quả hiện thực hệ thống	44
4.2	Đánh giá hệ thống nhận dạng cử chỉ tay đề xuất	45
4.3	Ứng dụng dựa trên hệ thống cử chỉ tay	50
CHƯƠNG V: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....		52
5.1	Kết luận	52
5.2	Hướng phát triển.....	52
Tài liệu tham khảo.....		54

DANH SÁCH HÌNH ẢNH

<i>Hình 1-1 Các ứng dụng nhận dạng cử chỉ tay</i>	<i>12</i>
<i>Hình 1-2 Ba bước chính để nhận dạng cử chỉ tay</i>	<i>13</i>
<i>Hình 2-1 Kinect xbox 360</i>	<i>16</i>
<i>Hình 2-2 Cấu tạo kinect</i>	<i>16</i>
<i>Hình 2-3 Mô tả thực hiện của cảm biến độ sâu</i>	<i>17</i>
<i>Hình 2-4 Sơ đồ phân cứng thu thập dữ liệu từ kinect</i>	<i>18</i>
<i>Hình 1-5 Hệ tọa độ độ sâu trong Kinect</i>	<i>19</i>
<i>Hình 2-6 Siêu phẳng phân cách hai tập mẫu</i>	<i>22</i>
<i>Hình 2-7 Khoảng cách phân hoạch</i>	<i>24</i>
<i>Hình 2-8 Không thể phân hoạch tập mẫu trên bằng một siêu phẳng</i>	<i>26</i>
<i>Hình 3-1 Sơ đồ khối hệ thống nhận dạng cử chỉ tay đề xuất</i>	<i>30</i>
<i>Hình 3-2 Điểm theo dấu trên bàn tay có được sử dụng thư viện NITE 2</i>	<i>31</i>
<i>Hình 3-3 Vùng bàn tay có được khi đặt 2 ngưỡng Th1 và Th2</i>	<i>32</i>
<i>Hình 3-4 Giải thuật cho phương pháp trích ảnh bàn tay</i>	<i>33</i>
<i>Hình 3-5 Ảnh vùng bàn tay sau khi khử nhiễu</i>	<i>34</i>
<i>Hình 3-6 Ảnh bàn tay với đường bao</i>	<i>35</i>
<i>Hình 3-7 Giải thuật tách ảnh bàn tay</i>	<i>35</i>
<i>Hình 3-8 Biểu diễn trục bàn tay và góc theo phương ngang</i>	<i>36</i>
<i>Hình 3-9 Trọng tâm (điểm màu đỏ), tâm bàn tay (màu tím) và vòng tròn nội tiếp lớn nhất (vòng tròn màu đỏ)</i>	<i>38</i>
<i>Hình 3-10 Thuật toán K-curvature</i>	<i>39</i>
<i>Hình 3-11 Các đặc trưng của bàn tay được trích ra</i>	<i>40</i>

<i>Hình 3-12 Mười một cử chỉ khác nhau được chứa trong cơ sở dữ liệu huấn luyện.</i>	41
<i>Hình 3-13 Giảm đồ biểu diễn véc tơ đặc trưng của 11 cử chỉ.....</i>	43
<i>Hình 4-1 Phần mềm hiện thực giải thuật đề xuất</i>	44
<i>Hình 4-2 Hình ảnh độ sâu hiển thị trên phần mềm.....</i>	45
<i>Hình 4-3 Dữ liệu huấn luyện.....</i>	46
<i>Hình 4-4 Độ chính xác của hệ thống qua phương pháp kiểm tra mẫu</i>	47
<i>Hình 4-5 Độ chính xác trung bình của hệ thống.....</i>	49
<i>Hình 4-6 Hệ thống điều khiển trò chơi</i>	50
<i>Hình 4-7 Bảng điều khiển cho trò chơi “búa, kéo, giấy”</i>	51
<i>Hình 4-8 Trò chơi “búa, kéo, giấy” sử dụng cử chỉ tay</i>	51

CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI

1.1 Giới thiệu đề tài

Ngày nay dưới sự phát triển rộng rãi của các ứng dụng công nghệ thông tin vào trong cuộc sống, việc tương tác giữa con người và thiết bị điện tử ngày càng trở nên quan trọng và phổ biến. Con người có thể điều khiển các thiết bị máy tính, điện tử bằng giọng nói hay cử chỉ giống như khi tương tác giữa người với người trong thế giới thực mà không cần thông qua các thiết bị điều khiển khác như bàn phím, chuột hay các nút nhấn. Điều này đem lại cho người sử dụng một sự trải nghiệm mới, thú vị với các sản phẩm này.

Nhận dạng các cử chỉ của tay người là cách tự nhiên khi tương tác người với máy tính và ngày nay được nghiên cứu càng nhiều trong trong các trường đại học trên thế giới và được ứng dụng càng nhiều trong các lĩnh vực khác nhau như thiết bị trò chơi, tivi thông minh, thiết bị y tế, điều khiển robot, tương tác thực ảo,... Ví dụ cho ứng dụng tivi thông minh như ta chỉ cần phẩy tay là có thể chuyển kênh tivi, tăng giảm âm lượng thay vì phải tự tay nhấn nút trên bộ điều khiển.



Hình 1-1 Các ứng dụng nhận dạng cử chỉ tay

Đã có nhiều đề tài, ứng dụng nhận dạng cử chỉ tay nhưng nó vẫn là một nghiên cứu đầy thách thức do sự phức tạp hình học của bàn tay và những khó khăn trong việc nhận dạng bàn tay trong điều kiện ánh sáng yếu hay là trong việc phân biệt bàn tay và các đối tượng khác cùng màu sắc nếu chỉ sử dụng camera thông thường. Năm 2010, Microsoft cho ra đời cảm biến Kinect dành cho máy Xbox 360, bên cạnh phục vụ cho mục đích chơi game, sản phẩm Kinect còn được dùng rộng rãi vào mục đích nghiên cứu bởi vì ngoài thu nhận hình ảnh màu, nó còn thu nhận được hình ảnh độ sâu dựa vào camera hồng ngoại.

Cử chỉ tay có thể là tĩnh hoặc động. Nhận dạng cử chỉ tĩnh yêu cầu tính phức tạp thấp hơn vì chỉ dựa vào phân tích một khung hình đơn lẻ. Nhận dạng cử chỉ động yêu cầu sự tính toán phức tạp hơn bởi vì cần tính toán trên các khung hình liên tục nhưng thích hợp cho tính toán thời gian thực. Trong luận văn này chúng tôi chỉ tập trung nghiên cứu và thiết kế hệ thống nhận dạng cho cử chỉ tĩnh.

Để nhận dạng cử chỉ tay tĩnh, bước đầu tiên trong các hệ thống nhận dạng là phát hiện ra vị trí bàn tay (segmentation); sau đó tiến hành xử lý hình ảnh, trích chọn đặc trưng (feature extraction) để tính toán các thông tin mô tả bàn tay, từ các dữ liệu quan sát và thực hiện công việc phân loại dựa vào các đặc tính đã được trích chọn để nhận dạng cử chỉ (classification). Đó là ba bước chính của một hệ thống nhận dạng cử chỉ tay.



Hình 1-2 Ba bước chính để nhận dạng cử chỉ tay

1.2 Tình hình nghiên cứu

Nhận dạng cử chỉ tay đã được nghiên cứu từ lâu nhưng nó vẫn là một đề tài đầy thách thức vì tích phức tạp hình học và những khó khăn trong việc tách vùng bàn tay trong điều kiện môi trường khác nhau nếu chỉ sử dụng camera thông thường. Phương pháp trích ảnh bàn dựa màu da người là một phương pháp khá đơn giản và được áp dụng ở nhiều bài báo nghiên cứu [1, 2, 3]. Phát hiện màu da là một bước xử lý tìm kiếm trong ảnh các vùng và điểm ảnh có màu da rồi đưa ra kết quả vùng bàn tay trên ảnh là vùng các điểm ảnh có màu da, phát hiện các điểm ảnh màu da có vẻ như khá dễ dàng. Tuy nhiên, do phương pháp này chỉ dựa vào thông tin về màu sắc nên các vùng ảnh không phải là bàn tay như khuôn mặt hay các vùng da khác trên cơ thể con người, thậm chí là các đối tượng khác có màu giống với màu da cũng bị nhận diện như là bàn tay. Hơn nữa khi ánh sáng thay đổi thì giá trị màu sắc trên da cũng thay đổi, và màu da lại có sự khác nhau ở mỗi người. Do đó, việc phát hiện bàn tay

dựa trên màu da chỉ có hiệu quả khi trong ảnh ngoài vùng bàn tay thì không chứa thêm các đối tượng khác có màu da và vùng bàn tay phải có sự tách biệt với nền.

Việc sử dụng bao tay và đánh dấu màu trên những vị trí cần nhận dạng trên bàn tay sẽ cung cấp thông tin chính xác về hướng, vị trí lòng bàn tay và từng ngón tay [4, 5]. Nhưng việc sử dụng bao tay hay các công cụ khác cũng đều không mang lại sự thoải mái cho người sử dụng và điều này dường như cũng không thể giúp nhận dạng bàn tay trong điều kiện ánh sáng yếu hoặc trong điều kiện tối. Năm 2010, Microsoft phát hành cảm biến Kinect, ngoài ảnh màu, nó còn cung cấp thêm ảnh độ sâu giúp cho việc nhận dạng bàn tay dễ dàng, chính xác, tin cậy hơn và việc nhận dạng cũng không bị ảnh hưởng bởi môi trường ánh sáng.

Một trong những cách phổ biến được sử dụng để nhận ra cử chỉ tay là phương pháp áp dụng thuật toán học máy (machine learning algorithm) dựa vào một số đặc trưng của bàn tay. Trong bài báo số [3], một tính chất hình học của bàn tay như trọng tâm bàn tay, số điểm ảnh,... được chuyển thành vector đặc tính, mạng nơ-ron nhân tạo được sử dụng để nhận dạng và phân loại cử chỉ. Tác giả Luigi Lamberti [4] miêu tả hình ảnh bàn tay của bằng 1 vector đặc trưng 9 phần tử là những góc và các khoảng cách Euclide, sau đó vector đặc trưng này được đưa vào một bộ phân loại thực hiện bởi phương pháp học vector lượng tử (Learning Vector Quantization). Các tác giả của bài báo số [6] lại tạo ra một mô hình bàn tay 3D mô tả bởi 21 phần khác nhau, bộ phân loại cử chỉ dựa trên máy vector hỗ trợ (Support Vector Machine (SVM)) sau đó được áp dụng để nhận ra cho mười chữ số trong tập ngôn ngữ cử chỉ Mỹ (American Sign Language(ASL)). Cả 2 bài báo [7], [8] tách hình ảnh sâu sử dụng một cách tiếp cận đơn giản: bàn tay phải đặt gần cảm biến nhất, vì vậy nó rất dễ dàng để trích xuất các khu vực tay bằng cách đặt một ngưỡng sâu lên điểm gần nhất thu được. Bài báo số [7] sử dụng đặc trưng là biểu đồ về khoảng cách từ tâm đến đường bao của bàn tay để nhận ra cử chỉ tay. Còn bài báo số [8] kết hợp 2 vector đặc trưng bao gồm đặc trưng khoảng cách và đặc trưng về đường bao và sau đó sử dụng một máy phân loại vector đa lớp (Multi-class Support Vector Machine) để phân biệt những cử chỉ. Trong bài báo số [9] đưa ra phương pháp nhận dạng cử chỉ tay sử dụng phép biến đổi Fourier của đường bao bàn tay để làm thông tin cho vector đặc trưng, sau đó thực

hiện thuật toán phân loại phần tử cận gần nhất (Nearest Neighbor classification) để nhận dạng ASL.

Cách tiếp cận mà chúng tôi đề xuất là một trong những phương pháp nhận dạng mà sử dụng kỹ thuật máy học để nhận ra cử chỉ tay. Trong luận văn này, chúng tôi đưa ra một sự phương pháp nhận dạng cử chỉ tay sử dụng một vector đặc trưng mới là sự kết hợp các đặc trưng bao gồm số đầu ngón tay, đặc trưng về góc và đặc trưng về khoảng cách. Căn cứ vào cơ sở dữ liệu được xây dựng từ các mẫu, một máy hỗ trợ vector được áp dụng để nhận ra những cử chỉ tay khác nhau.

1.3 Mục tiêu luận văn và giới hạn đề tài

Mục tiêu của luận văn là thiết kế 1 hệ thống nhận dạng cử chỉ tay mà không cần bất kì công cụ hỗ trợ nào, có độ chính xác cao, tương thích với điều kiện trong bóng tối, hiển thị cử chỉ thời gian thực:

- Hiện thực trên máy tính sử dụng các thư viện C#, C++ của Windows, các thư viện hỗ trợ xử lý ảnh OpenCV và thư viện cho Kinect: OpenNi2, Nite2.
- Hệ thống chỉ cho phép nhận dạng với điều kiện không có ánh sáng mặt trời.
- Vùng nhận dạng giới hạn trong phạm vi [50cm – 1m].
- Nhận dạng 11 cử chỉ và góc nghiêng bàn tay không quá 45° .

1.4 Bố cục luận văn

Luận văn được chia thành 5 chương với nội dung như sau:

Chương I: Giới thiệu tổng quan và giới thiệu về đề tài, tình hình nghiên cứu, mục tiêu và giới hạn đề tài.

Chương II: Cơ sở lý thuyết.

Chương III: Hệ thống nhận dạng cử chỉ tay đề xuất.

Chương IV: Kết quả thực hiện.

Chương V: Kết luận và hướng phát triển.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1 Cảm biến Kinect và các thư viện hỗ trợ

2.1.1 Cảm biến Kinect

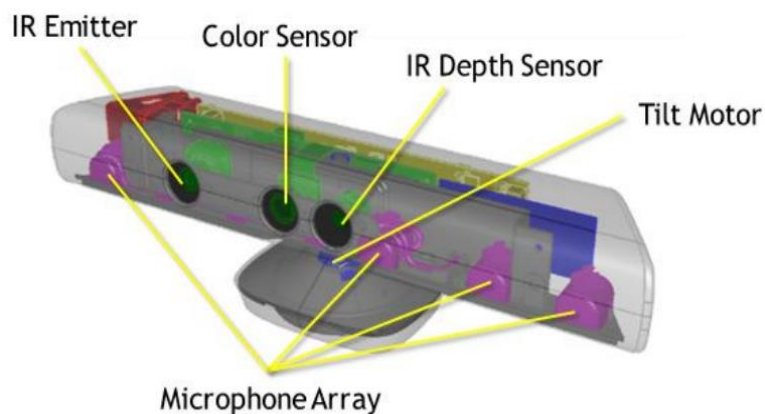
2.1.1.1 Giới thiệu



Hình 2-1 Kinect xbox 360

Kinect 360 là sản phẩm của Microsoft, dựa trên công nghệ camera được phát triển bởi PrimeSense. Khi kết nối vào Xbox 360, người chơi có thể điều khiển và tương tác với máy console mà không cần tay cầm điều khiển, thông qua giao diện người dùng tự nhiên sử dụng cử chỉ và lệnh thoại. Khả năng nhận dạng cử chỉ của Kinect dựa trên hai đặc tính chính sau: thông tin về độ sâu ảnh (depth map), khả năng phát hiện và bám theo đặc tính cơ thể người (body skeleton tracking). Bên cạnh phục vụ cho mục đích chơi game, sản phẩm Kinect còn được dùng vào mục đích nghiên cứu xử lý ảnh 3D, phát hiện cử chỉ (gesture recognition), bám theo người (body tracking) và nhiều mục đích khác.

2.1.1.2 Cấu tạo phần cứng



Hình 2-2 Cấu tạo kinect

Kinect bao gồm 1 RGB camera, các thành phần sử dụng để cảm biến độ sâu, một dãy các microphone và 1 động cơ điều khiển góc nâng.

- **RGB Camera:** là một camera thu nhận ảnh màu 8-bit thông thường tới bộ lọc màu bayer, tốc độ khung hình 30Hz, có độ phân giải chuẩn VGA (640×480 điểm ảnh).

- **Cảm biến độ sâu:** bao gồm một thiết bị chiếu tia laser hồng ngoại (IR projector) kết hợp với một bộ cảm biến CMOS đơn sắc (IR camera), cho phép thu thập dữ liệu độ sâu dưới hầu hết mọi điều kiện ánh sáng xung quanh.

Nguyên lý các cảm biến độ sâu:



Hình 2-3 Mô tả thực hiện của cảm biến độ sâu

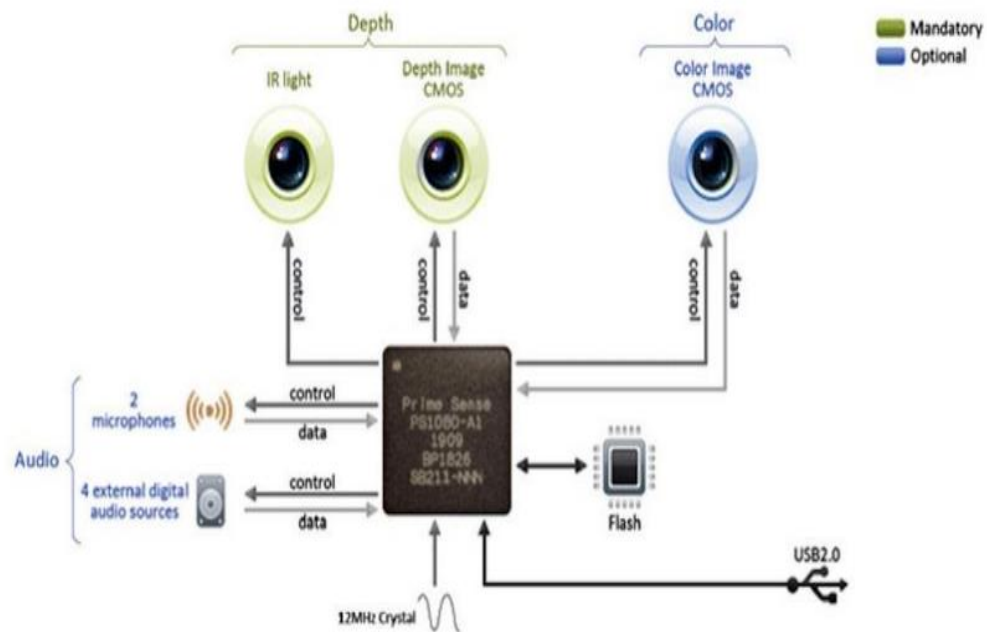
Cặp cảm biến IR camera và IR projector sẽ phối hợp với nhau để tạo ra giá trị độ sâu bằng công nghệ Light Coding của PrimeSense. Kỹ thuật Light Coding dùng nguồn sáng hồng ngoại chiếu liên tục kết hợp với một camera hồng ngoại để tính khoảng cách. Việc tính toán này được thực hiện bằng chip PS1080 Soc của PrimeSen. Projector sẽ chiếu một chùm sáng hồng ngoại, tạo nên những đốm sáng ở không gian phía trước Kinect, tập hợp đốm sáng được phát ra này là cố định. Những đốm sáng

này được tạo ra nhờ một nguồn sáng truyền qua lưới nhiễu xạ (diffraction gratings). Tập hợp các đốm sáng này được IR camera chụp lại, thông qua giải thuật đặc biệt được tích hợp trong PS1080 SoC cho ra bản đồ độ sâu. Bản chất của giải thuật này là các phép toán hình học dựa trên quan hệ giữa hai cảm biến IR camera và Projector.

- **Dãy Microphone:** Dây Micro bao gồm 4 micro được bố trí dọc theo thân Kinect có khả năng thu lại âm thanh đồng thời xác định hướng của âm thanh.
- **Ngoài ra,** Kinect còn có 1 cảm biến đo gia tốc để xác định hướng và 1 động cơ dùng để điều khiển góc nghiêng camera.

2.1.1.3 Dữ liệu thu được từ Kinect

Các cảm biến của Kinect được điều khiển đồng thời thu thập và xử lý dữ liệu thông qua chip PS1080 có tần số 12MHz, sau đó được lưu trữ vào bộ nhớ Flash. Các dữ liệu này có thể truyền vào máy tính thông qua cổng USB 2.0.

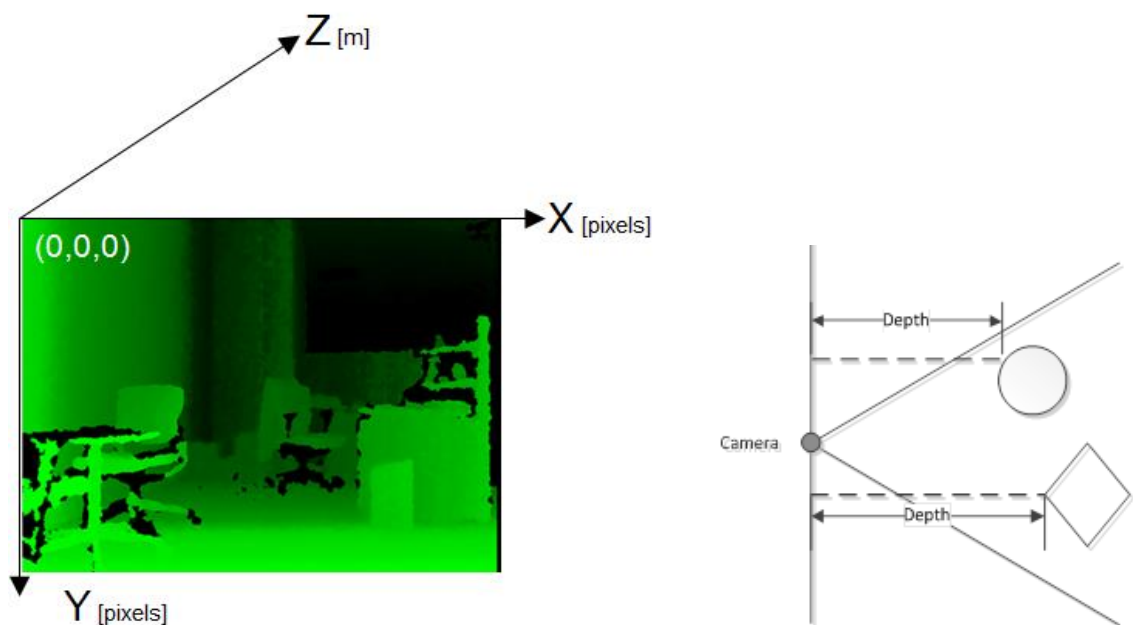


Hình 2-4 Sơ đồ khối phần cứng thu thập dữ liệu từ kinect

Các tín hiệu thu thập bao gồm dữ liệu về độ sâu, màu sắc và âm thanh trong đó dữ liệu về độ sâu là dữ liệu quan trọng có nhiều ứng dụng. Sở dĩ dữ liệu về chiều độ sâu có tầm quan trọng như vậy bởi nó giúp việc nhận dạng các vật thể đơn giản hơn nhiều so với xử lý ảnh thông thường.

Các thuật toán xử lý ảnh thông thường dựa vào sự tương đồng về màu sắc, tuy nhiên, có thể những vật có màu sắc tương tự nhau nhưng không cùng một vật thể hoặc các phần của cùng một đối tượng nhưng có màu khác nhau, do vậy gây khó khăn trong quá trình nhận dạng. Trong khi đó, với thông tin về độ sâu, các vật thể được phân biệt với nhau thông qua vị trí. Những điểm có khoảng cách gần nhau có xu hướng cùng một đối tượng mà không phân biệt màu sắc. Chỉ khi độ sâu giảm đột ngột như ở cạnh và ở một số phần nhỏ của đối tượng thì khi đó, hình ảnh trên bản đồ độ sâu mới có sự thay đổi.

Giá trị độ sâu hay là khoảng cách từ Kinect đến vật thể bản thân Kinect thực sự không tính toán độ sâu, nó trả về một giá trị trừu tượng hơn cho hệ thống xử lý và tùy vào driver sẽ xuất theo dạng dữ liệu nào. Trong khi OpenNI và MS Kineck SDK xuất giá trị này ra dưới khoảng cách tính bằng mm, còn libfreenect xuất ra các giá trị lượng tử 11-bit.



Hình 1-5 Hệ tọa độ độ sâu trong Kinect

2.1.2 Các thư viện xử lý ảnh sử dụng cho lập trình với Kinect

Các thư viện xử lý ảnh được sử dụng phổ biến: Libfreenect, Code Laboratories Kinect, OpenNI và Kinect SDK. Mỗi thư viện xử lý ảnh đi kèm với driver của từng thư viện. Riêng OpenNI phiên bản 2 hỗ trợ driver của nhà phát triển thứ 3.

Libfreenect:

Libfreenect là thư viện được phát triển bởi OpenKinect, do một cộng đồng những người quan tâm đến Kinect viết và chia sẻ. Libfreenect được đóng gói sử dụng trên nhiều ngôn ngữ khác nhau như Python, C, C++, C#, Java JNI, Java JNA, Javascript.

Code Laboratories Kinect:

Code Laboratories (CL) là một công ty về phần mềm chuyên hỗ trợ các nhà phát triển, lập trình viên khai thác các tính năng của các thiết bị xử lý ảnh. trong đó Kinect, CL cung cấp cho người sử dụng những tính năng cơ bản nhất của Kinect về camera, audio và motor.

Windows Kinect SDK:

Windows Kinect SDK chứa các trình điều khiển tương thích hệ điều hành Windows 7 cho Kinect. SDK cho phép nhà phát triển xây dựng các ứng dụng Kinect với C++, C# hoặc Visual Basic trên Microsoft Visual Studio 2010 và bao gồm các đặc tính:

- Stream dữ liệu cảm biến gốc: truy cập đến stream cấp thấp thu được từ cảm biến độ sâu, cảm biến màu sắc, và mảng micro bốn phần tử.
- Theo dõi cơ thể: khả năng theo dõi skeleton nhiều người di chuyển trong vùng nhận biết của Kinect phục vụ cho ứng dụng điều khiển bởi cử chỉ.
- Nâng cao khả năng xử lý âm thanh: khả năng xử lý âm thanh bao gồm bộ khử nhiễu và bộ lọc tiếng, hình thành chùm tia để xác định nguồn âm thanh, và tích hợp API nhận dạng giọng nói của Windows.

OpenNI/NITE:

OpenNI (Open Natural Interaction) là một framework đa nền tảng, đa ngôn ngữ, và nó định nghĩa giao diện lập trình ứng dụng (API) để viết các ứng dụng dùng tương tác tự nhiên. OpenNI API bao gồm một tập hợp các giao diện để viết các ứng dụng tương tác tự nhiên. Mục đích chính của OpenNI là tạo nên API tiêu chuẩn cho phép giao tiếp với:

- Cảm biến âm thanh và hình ảnh.

- Các middleware cảm nhận hình ảnh và âm thanh (các thành phần phần mềm mà phân tích các dữ liệu âm thanh và hình ảnh được ghi lại, và hiểu nó). Ví dụ, phần mềm nhận dữ liệu hình ảnh, chẳng hạn nhận một hình ảnh, trả về vị trí của lòng bàn tay của một bàn tay được phát hiện trong hình ảnh.

Tháng 4/2013 tổ chức OpenNI phát hành phiên bản OpenNI2/NiTE2 đơn giản hóa các APIs, kiểu dữ liệu, dễ dàng giao tiếp giữa các middleware và hỗ trợ driver từ các nhà phát triển thứ 3.

Các thư viện hỗ trợ xử lý ảnh khác:

Open CV:

Open CV là một thư viện mở gồm các hàm được xây dựng phục vụ cho việc xử lý ảnh thời gian thực. Các thuật toán xử lý ảnh thông thường lẫn cao cấp đều được tối ưu hóa bởi các nhà phát triển thành các hàm đơn giản rất dễ sử dụng. OpenCV hỗ trợ 2 ngôn ngữ chính : C/C++ và python.

OpenGL:

Là một giao diện phần mềm độc lập với phần cứng hỗ trợ cho lập trình đồ họa. Để làm được điều này, OpenGL không thực hiện các tác vụ thuộc về hệ điều hành cũng như không nhận dữ liệu nhập của người dùng (người dùng giao tiếp với OpenGL thông qua OpenGL API). Nó là lớp trung gian giữa người dùng và phần cứng. Nghĩa là nó giao tiếp trực tiếp với driver của thiết bị đồ họa.

Lựa chọn thư viện:

Vì những ưu điểm kể trên của bộ thư viện OPENNI2/NITE2, ngoài ra bộ thư viện này còn hỗ trợ cho cả ARM-linux sẽ giúp các nhà nghiên cứu dễ dàng phát triển phần cứng riêng, nên chúng tôi quyết định lựa chọn thư viện OPENNI2/NITE2 trong luận văn này.

2.2 Máy học vector hỗ trợ (Support Vector Machine)

Support Vector Machine (SVM), là một trong những thuật toán học máy tốt nhất, đã được đề xuất trong năm 1995 bởi Vapnik. Đây là một phương pháp dựa trên lý thuyết học thống kê (Statistical Learning Theory) nên có một nền tảng toán học chặt chẽ để bảo đảm rằng kết quả đạt được tối ưu. Ý tưởng chính của SVM là chuyển

tập mẫu từ không gian biểu diễn R_n của chúng sang một không gian R_d có số chiều lớn hơn. Trong không gian R_d tìm một siêu phẳng tối ưu để phân hoạch tập mẫu này dựa trên phân lớp của chúng, cũng có nghĩa là tìm ra miền phân bố của từng lớp trong không gian biểu diễn R_n , để từ đó xác định được phân lớp của một mẫu cần nhận dạng.

Cũng như mạng nơ-ron, phương pháp SVM là một phương pháp có tính tổng quát cao, có thể áp dụng cho nhiều loại bài toán nhận dạng khác nhau.

2.2.1 SVM trong trường hợp tập phân hoạch tuyến tính

Trong trường hợp này, tập mẫu là một tập có thể phân chia tuyến tính bằng một siêu phẳng và SVM đi tìm siêu phẳng này.

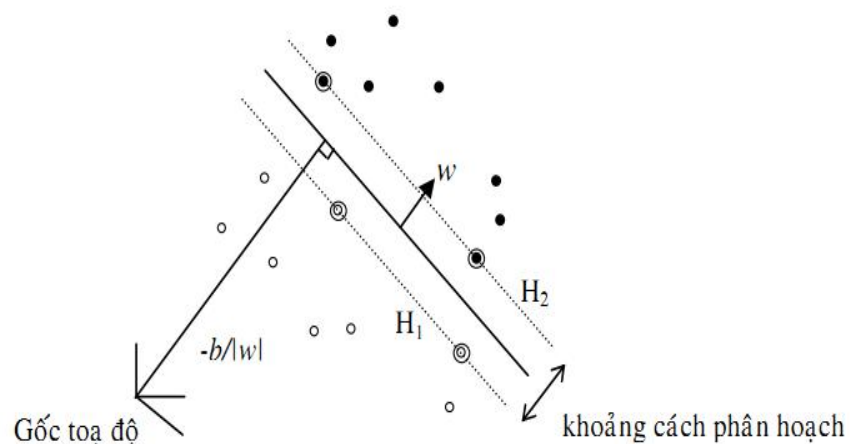
❖ *Giai đoạn huấn luyện SVM:*

Giả sử tập mẫu có được gồm l phần tử là:

$$(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)$$

Trong đó $x_i \in R^n$ còn $y_i \in \{-1; 1\}$ là phân lớp của x_i .

Cần xác định được siêu phẳng mà có thể tách biệt được hai lớp trên và vấn đề là cần tìm ra siêu phẳng nào làm cho khoảng cách Euclid giữa hai lớp trên là lớn nhất (hình 1). Lúc đó các vector có khoảng cách gần siêu phẳng nhất được gọi là support vector.



Hình 2-6 Siêu phẳng phân cách hai tập mẫu

Giả sử phương trình siêu phẳng cần tìm là $w \cdot x + b = 0$ trong đó w là pháp vector của siêu phẳng $w \in R^n$. Ta có hai bất phương trình sau:

$$w x_i + b \leq -1 \text{ với } y_i = -1$$

$$w x_i + b \geq +1 \text{ với } y_i = +1$$

Kết hợp hai bất phương trình trên:

$$(w x_i + b) y_i - 1 \geq 0$$

Lúc đó những support vector x_i thỏa mãn phương trình $w x_i + b = -1$ thì nằm trên siêu phẳng H_1 , phương trình $w x_i + b = +1$ thì nằm trên siêu phẳng H_2 .

Khoảng cách có dấu d_1 từ gốc tọa độ đến H_1 là: $d_1 = (1-b)/\|w\|$

Khoảng cách có dấu d_2 từ gốc tọa độ đến H_2 là: $d_2 = (-1-b)/\|w\|$

Suy ra khoảng cách phân hoạch d giữa H_1 và H_2 là: $d = |d_1 - d_2| = 2/\|w\|$

Do đó để có d lớn nhất thì $\|w\|$ phải nhỏ nhất hay nói cách khác phải đi tìm cực tiểu của $\frac{1}{2} \|w\|^2$ (được biến đổi một chút để dễ tìm w sau này).

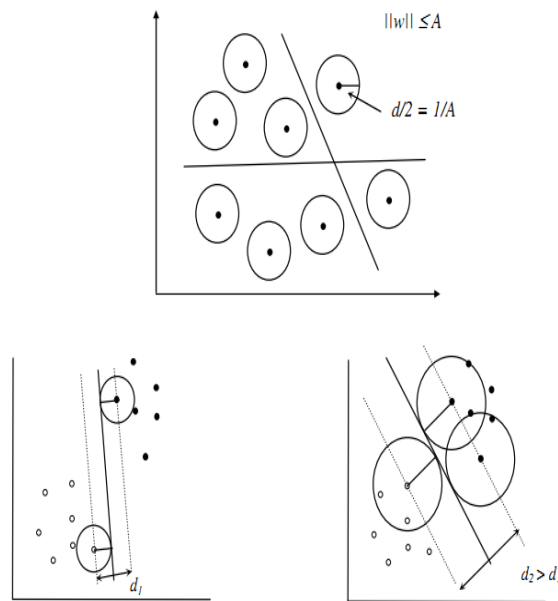
Tại sao phải cần tìm siêu phẳng ứng với khoảng cách phân hoạch lớn nhất? Có thể giải thích điều này từ lý thuyết của nguyên tắc SRM. Vì siêu phẳng này phân hoạch tập mẫu thành hai lớp tách biệt nên $R_{emp}(\alpha) = 0$. Còn VC Confidence thì sao? Vapnik đã chứng minh rằng nếu các điểm x_i nằm bên trong hình cầu bán kính R của không gian R^n và $\|w\| \leq A$ thì VC dimension của tập các siêu phẳng $w \cdot x + b = 0$ là:

$$h \leq \min\{[R^2 A^2], n\} + 1 \text{ trong đó } [R^2 A^2] \text{ là phần nguyên của } R^2 A^2.$$

Do đó nếu chọn d lớn thì $\|w\|$ nhỏ, suy ra A nhỏ vì VC dimension h sẽ nhỏ. Và đó là mục tiêu của nguyên tắc SRM. Hình 3 minh họa tính chất này của $\|w\|$.

Bao xung quanh các điểm mẫu x_i là các hình cầu bán kính $1/A$. Siêu phẳng tìm được phải không được cắt các hình cầu (như vậy thì mới có $d \geq 2/A$). Nếu không có ràng buộc này thì tập siêu phẳng bao gồm tất cả các siêu phẳng trong không gian R^n . Tuy nhiên với ràng buộc này, số lượng các siêu phẳng bị giới hạn lại. Có nghĩa là VC dimension của tập siêu phẳng giới hạn sẽ nhỏ hơn VC

dimension khi không bị giới hạn. $\|w\|$ càng nhỏ (hay d càng lớn) thì số các siêu phẳng càng ít đi dẫn đến VC dimension càng giảm.



Hình 2-7 Khoảng cách phân hoạch

Phương án để tìm cực tiểu của $\frac{1}{2} \|w\|^2$

Bài toán: Tìm cực tiểu của $\frac{1}{2} \|w\|^2$ theo w và b dựa trên: $(wx_i + b) y_i - 1 \geq 0$ $i=1,2,\dots$ là bài toán tối ưu có ràng buộc, trong đó hàm mục tiêu là một hàm lồi và miền ràng buộc cũng là một tập lồi. Do có tính lồi nên để giải bài toán trên ta có thể chuyển qua giải bài toán đối ngẫu tương ứng. Bài toán đối ngẫu sẽ là:

Tìm cực đại của: $\Theta(u) = \inf \left\{ \frac{1}{2} \|w\|^2 - \sum_{i=1}^l u_i [(wx_i + b) y_i - 1] : w \in R^n, b \in R \right\}$

Để giải bài toán đối ngẫu trước tiên ta phải tìm cực tiểu của

$$L(w, b) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l u_i [(wx_i + b) y_i - 1] = 0 \text{ theo } w \text{ và } b$$

Vì L là hàm hai biến w, b bậc hai không ràng buộc nên theo điều kiện Fermat, cực tiểu của L xảy ra tại w và b sao cho:

$$\frac{\partial L(w, b)}{\partial w} = w - \sum_{i=1}^l u_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^l u_i y_i x_i$$

$$\frac{\partial L(w,b)}{\partial b} = \sum_{i=1}^l u_i y_i = 0$$

Lúc đó giá trị cực tiểu của L là:

$$\begin{aligned} aL_o(w,b) &= \frac{1}{2} ||w||^2 - \sum_{i=1}^l u_i [(wx_i + b) y_i - 1] \\ &= \sum_{i=1}^l u_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l u_i y_i x_i y_j x_j u_j \end{aligned}$$

Như vậy bài toán đối ngẫu được viết lại thành:

$$\text{Tìm cực đại của } F(u) = \sum_{i=1}^l u_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l u_i y_i x_i y_j x_j u_j$$

Dựa trên $u_i \geq 0 \quad i=1, \dots, l$

$$\sum_{i=1}^l u_i y_i = 0$$

Giải bài toán này ta tìm được u và từ đó tính được w theo công thức

Để tính b, vận dụng điều kiện KKT cho bài toán gốc, ta có được:

$$\sum_{i=1}^l u_i [(wx_i + b) y_i - 1] = 0 \quad i=1, \dots, l$$

Do đó đối với một I thì có hai trường hợp:

$u_i=0$: trong trường hợp này $(wx_i + b) y_i - 1 > 0$ suy ra x_i không nằm trên siêu phẳng biên H_1 và H_2 . Vì $u_i=0$ nên x_i không tham gia vào việc cấu trúc w theo công thức. Những x_i này là không cần thiết và có thể được bỏ đi mà không ảnh hưởng đến w.

$u_i>0$: lúc này $(wx_i + b) y_i - 1 = 0$ suy ra x_i nằm trên siêu phẳng biên H_1 và H_2 . x_i được gọi là support vector và tham gia vào việc tính w.

Thường thì số lượng support vector nhỏ hơn nhiều so với số lượng mẫu.

Do đó để tính b chỉ cần chọn một I mà có $u_i>0$, lúc đó:

$$(wx_i + b) y_i - 1 = 0 \text{ nên } b = (1/y_i) - wx_i = y_i - wx_i$$

Vậy ta đã tính được w và b nên xác định được siêu phẳng phân hoạch.

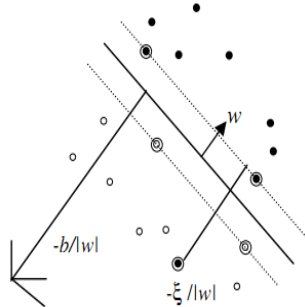
❖ *Giai đoạn nhận dạng:*

Bây giờ giả sử có một mẫu dữ liệu cần nhận dạng x^* nào đó, thì phân lớp y^* của x^* (-1 hay +1) được xác định thông qua công thức:

$$y^* = f(x^*) = \text{sign}(wx^* + b) = \text{sign}(\sum_{i=1}^l u_i y_i x_i x^* + b)$$

2.2.2 SVM trong trường hợp mẫu không phân hoạch tuyến tính được

Trường hợp tập mẫu có thể phân chia tuyến tính được chỉ là một trường hợp đặc biệt. Trong trường hợp tổng quát, tập mẫu là một tập không phân chia tuyến tính được bằng siêu phẳng



Hình 2-8 Không thể phân hoạch tập mẫu trên bằng một siêu phẳng

Tuy nhiên ta vẫn muốn dùng siêu phẳng để phân chia tập mẫu này.

Để có thể áp dụng được phương pháp trong phần trên, ta phải gán cho mỗi mẫu x_i một sai số ξ_i để “xem như có thể phân chia tuyến tính”.

$$awx_i + b \geq +1 - \xi_i \text{ với } y_i = +1$$

$$awx_i + b \leq -1 + \xi_i \text{ với } y_i = -1$$

$$\xi_i \geq 0 \quad i=1,2,\dots,l$$

Với cách đặt như vậy thì sai số thực nghiệm Empirical Risk là: $\sum_{i=1}^l \xi_i$

Cũng giống như phần trước, nếu cực tiểu $\|w\|$ thì sẽ làm cho VC dimension nhỏ và từ đó VC confidence nhỏ. Do đó để thực thi nguyên tắc SRM (cực tiểu cùng lúc Empirical risk và VC confidence), ta có thể cực tiểu biểu thức sau:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i$$

Trong đó C là một hằng số dương tự chọn. Việc chọn C thể hiện nguyên tắc SRM. Nếu chọn C lớn sẽ làm cho Empirical Risk nhỏ - do tập trung cực tiểu. Còn nếu C nhỏ thì làm cho VC confidence nhỏ - do cực tiểu w . (cách này được gọi là C -

SVM do dựa trên giá trị của khoảng cách lỗi, ngoài ra còn một phương án khác là tối thiểu số điểm lỗi, được gọi là v-SVM).

Vậy ta có thể phát biểu lại bài toán như sau:

$$\text{Cực tiểu: } f(w, b, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i$$

$$\text{Với } (wx_i + b) y_i - 1 + \xi_i \geq 0 \quad i=1, \dots, l$$

Bài toán đối ngẫu của bài toán trên sẽ là:

$$\text{Cực đại: } \theta(u, v) \text{ với } u \in \mathbb{R}^l, v \in \mathbb{R}^l, u \geq 0, v \geq 0$$

$$\text{Dựa trên } \theta(u, v) = \inf \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l u_i [(wx_i + b) y_i - 1 - \sum_{i=1}^l \xi_i v_i] \right\}$$

Cũng theo điều kiện Fermat, cực tiểu của

$$L(w, b, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l u_i [(wx_i + b) y_i - 1 - \sum_{i=1}^l \xi_i v_i]$$

Xảy ra tại w, b, ξ sao cho:

$$\frac{\partial L(w, b, \xi)}{\partial w} = w - \sum_{i=1}^l u_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^l u_i y_i x_i$$

$$\frac{\partial L(w, b, \xi)}{\partial b} = - \sum_{i=1}^l u_i y_i = 0$$

$$\frac{\partial L(w, b, \xi)}{\partial \xi_i} = C - u_i - v_i = 0 \Rightarrow 0 \leq u_i \leq C$$

Lúc đó giá trị cực tiểu của L là:

$$\begin{aligned} L_0(w, b, \xi) &= \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l u_i [(wx_i + b) y_i - 1 - \sum_{i=1}^l \xi_i v_i] \\ &= \sum_{i=1}^l u_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l u_i y_i x_i y_j x_j u_j \end{aligned}$$

Như vậy bài toán được viết lại thành:

$$\text{Tìm cực đại của } F(u) = \sum_{i=1}^l u_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l u_i y_i x_i y_j x_j u_j$$

$$\text{Dựa trên } 0 \leq u_i \leq C \quad i=1, \dots, l$$

$$\sum_{j=1}^l u_i y_i = 0$$

Bài toán này giống bài toán phân chia tuyến tính chỉ có thêm điều kiện $u_i \leq C$. Giải bài toán này ta tìm được w theo công thức, tức $w = \sum_{j=1}^{N_s} u_j y_j x_j$. Trong đó x_i là các vector support và các vector lỗi (ứng với $0 \leq u_i \leq C$). N_s là số lượng support vector và các vector lỗi.

Để tìm b , sử dụng điều kiện Karush-Kuhn-Tucker cho bài toán gốc ta có:

$$u_i [(wx_i + b) y_i - 1 + \xi_i] = 0$$

$$v_i \xi_i = 0$$

Chọn một u_i sao cho $0 \leq u_i \leq C$. Lúc đó $(wx_i + b) y_i - 1 + \xi_i = 0$ để điều kiện xảy ra. Do $C - u_i - v_i = 0$ nên có thêm $\xi_i = 0$ để điều kiện xảy ra. Lúc đó có thể tính được $b = y_i - wx_i$.

Vậy là đã tìm được siêu phẳng phân chia tập dữ liệu không thể phân hoạch tuyến tính. Siêu phẳng như vậy được gọi là siêu phẳng khoảng cách phân hoạch mềm (soft-margin hyperplane).

2.2.3 Các phương pháp phân loại SVM

SVM có hai phương pháp phân loại:

✓ *One-vs-rest*

Giả sử cần phải nhận dạng K lớp thì cần sử dụng K SVM. SVM thứ I sẽ phân biệt lớp thứ i và các lớp còn lại. Một mẫu thử x sẽ được cho kiểm tra trên K SVM này. Hàm nhận dạng của SVM thứ I nào cho ra kết quả giá trị bé nhất thì mẫu thử x thuộc lớp thứ I đó.

✓ *One-vs-one*

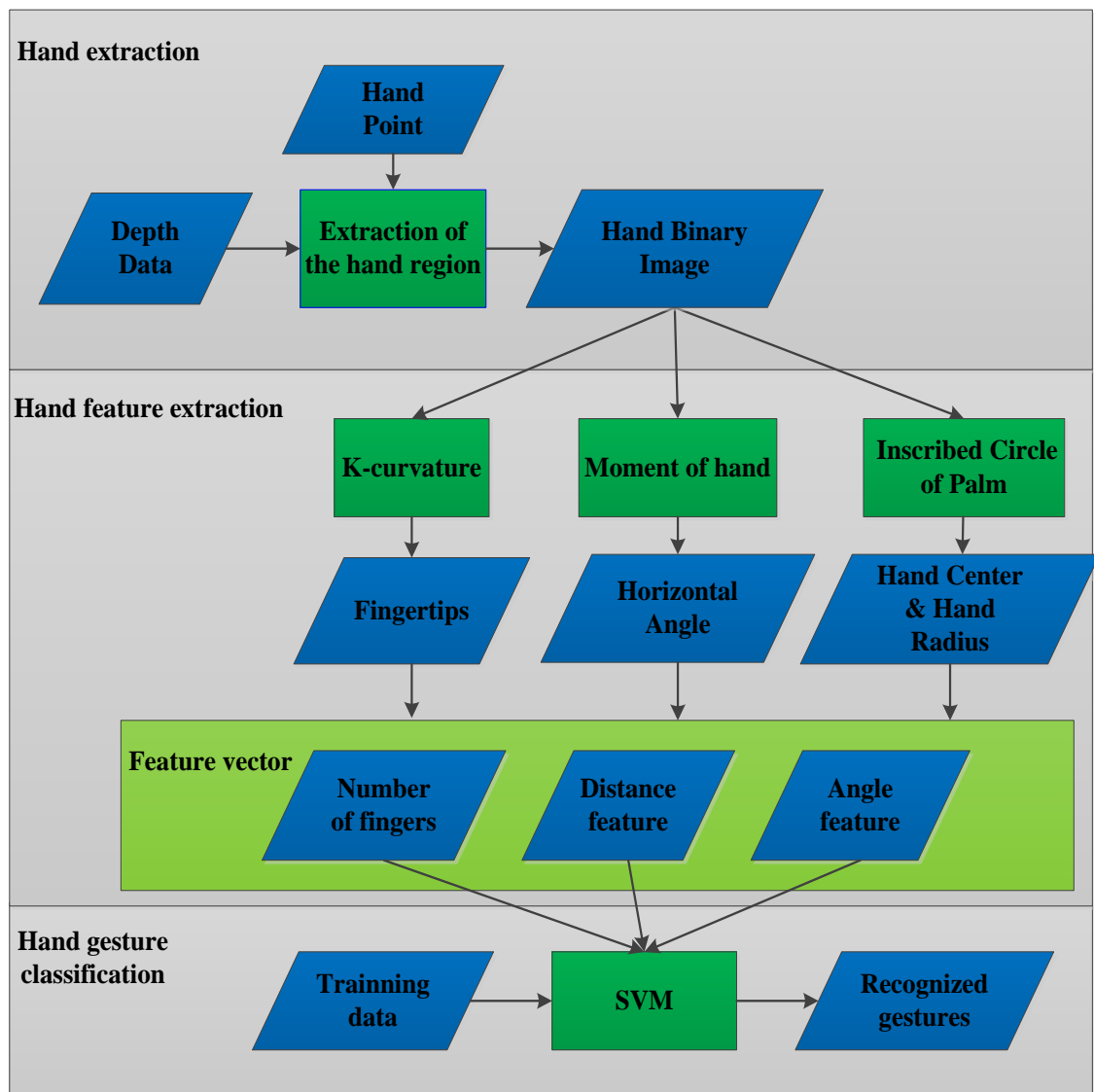
Phương pháp one-vs-one do John Platt và Nello Cristianini đưa ra. Trong phương pháp này, các lớp không phải là kết quả nhận dạng được loại trừ dần dần. Ứng với mỗi cặp lớp i, j , một SVM (i, j) được thiết lập chỉ dựa trên những mẫu của hai lớp này (i là lớp -1 , j là lớp $+1$). Đặc điểm của SVM này là nếu một mẫu thử có kết quả âm thì nó không kết luận mẫu này thuộc lớp I mà nó chỉ kết luận là mẫu này không thuộc lớp j và ngược lại. Có $K(K-1)/2$ SVM như vậy.

Có thể thấy rằng việc nhận dạng một mẫu không cần phải trải qua hết $K(K-1)/2$ SVM mà chỉ cần trải qua $K-1$ SVM mà thôi. Do đó số lượng các giá trị các kernel $K(x, y)$ cần phải tính chỉ là $(K-1) * M_s$ với M_s là số lượng support vector trung bình của một SVM hai lớp i, j . Vì M_s nhỏ hơn nhiều so với N_s nên thời gian nhận dạng sẽ được giảm bớt so với phương pháp one-vs-rest.

CHƯƠNG III: HỆ THỐNG NHẬN DẠNG CỬ CHỈ TAY ĐỀ XUẤT

3.1 Tổng quan hệ thống đề xuất

Hệ thống nhận dạng cử chỉ sử dụng cảm biến Kinect đề xuất cũng bao gồm 3 bước như các hệ thống nhận dạng khác: Tách ngưỡng ảnh bàn tay, trích đặc trưng và phân loại cử chỉ. Sơ đồ khối của hệ thống nhận dạng cử chỉ tay đề xuất được thể hiện ở hình dưới đây.



Hình 3-1 Sơ đồ khối hệ thống nhận dạng cử chỉ tay đề xuất

Bước đầu tiên của hệ thống là cần phát hiện được vùng bàn tay và tách ảnh bàn tay ra khỏi hình nền. Dựa vào ảnh độ sâu thu được, vùng bàn tay được lấy ra bằng cách đặt các ngưỡng độ sâu và ngưỡng khoảng cách tọa độ trên một điểm thuộc bàn

tay được theo dấu sử dụng thư viện NITE 2 cung cấp bởi PrimeSense. Bước thứ hai, các đặc trưng của bàn tay được tìm ra bao gồm: vị trí những đầu ngón tay sử dụng thuật toán K-Curvature mà đã được áp dụng trong các bài báo [18], [19], [20]; góc giữa phương ngang của hình và trục bàn tay sử dụng phương pháp phân tích moment [21]; bán kính đường tròn nội tiếp lớn nhất của lòng bàn tay và tâm bàn tay [11], [12], [20]. Từ những đặc trưng của bàn tay được tìm ra, vector đặc trưng gồm 15 phần tử được xây dựng, trong đó giá trị đầu tiên là số đầu ngón tay, 5 giá trị tiếp theo mô tả các góc giữa đầu ngón tay so với phương ngang của bàn tay với tâm bàn tay là góc, 4 giá trị kế tiếp là góc giữa các ngón tay với nhau, và 5 giá trị cuối cùng là những hiệu của khoảng cách từ tâm của bàn tay đến các đầu các ngón tay và bán kính vòng tròn nội tiếp lớn nhất của lòng bàn tay. Cuối cùng, máy vector hỗ trợ (SVM) được áp dụng cho véc tơ đặc trưng trên để nhận ra những cử chỉ khác nhau.

3.2 Phương pháp trình ảnh bàn tay và tiền xử lý

3.2.1 Phát hiện bàn tay

Bước đầu tiên trong nhận dạng cử chỉ tay là tách vùng bàn. Chúng tôi phát hiện vùng ảnh chứa bàn tay ra bằng cách sử dụng thư viện NITE2. Thư viện này cung cấp hàm theo dấu một điểm trên bàn tay sử dụng phương pháp Bayesian Object Localization. Người dùng đưa tay qua lại trước cảm biến, một điểm trên bàn tay sẽ được tìm ra sau một vài giây và sẽ được theo dấu đến khi nào bàn tay không còn hiện diện trước cảm biến.



Hình 3-2 Điểm theo dấu trên bàn tay có được sử dụng thư viện NITE 2

Giả sử điểm trên bàn tay sẽ có tọa độ ảnh $P_h(x_h, y_h)$ và độ sâu tại đó là D_h . Vùng bàn tay được tách ra bằng cách đặt 1 ngưỡng độ sâu T_{h1} và 1 ngưỡng khoảng cách tọa độ theo 2 phương x, y tính từ P_h . Gọi $P_i(x_i, y_i)$ là điểm ảnh trên ảnh độ sâu thu được và D_i là độ sâu tại đó. Chúng tôi định nghĩa vùng bàn tay theo công thức:

$$H = \{P_i | (|x_i - x_h| < T_{h1}) \cap (|y_i - y_h| < T_{h1}) \cap (D_i < D_h + T_{h2})\} \quad (1)$$

Giá trị của T_{h1} được thiết lập tùy thuộc vào độ sâu của bàn tay để làm sao tách được vùng bàn tay một cách thích hợp. Trong luận văn này, chúng tôi chọn các ngưỡng với $T_{h1} = \frac{64000}{D(x_h)}$, $T_{h2} = 100$ (mm). Vùng ảnh bàn tay H được chuyển qua ảnh nhị phân, nó sẽ bao gồm vùng trắng với diện tích lớn nhất chính là ảnh bàn tay, vùng đen là nền, ngoài ra còn có thể có một số ảnh nhiễu nhỏ.

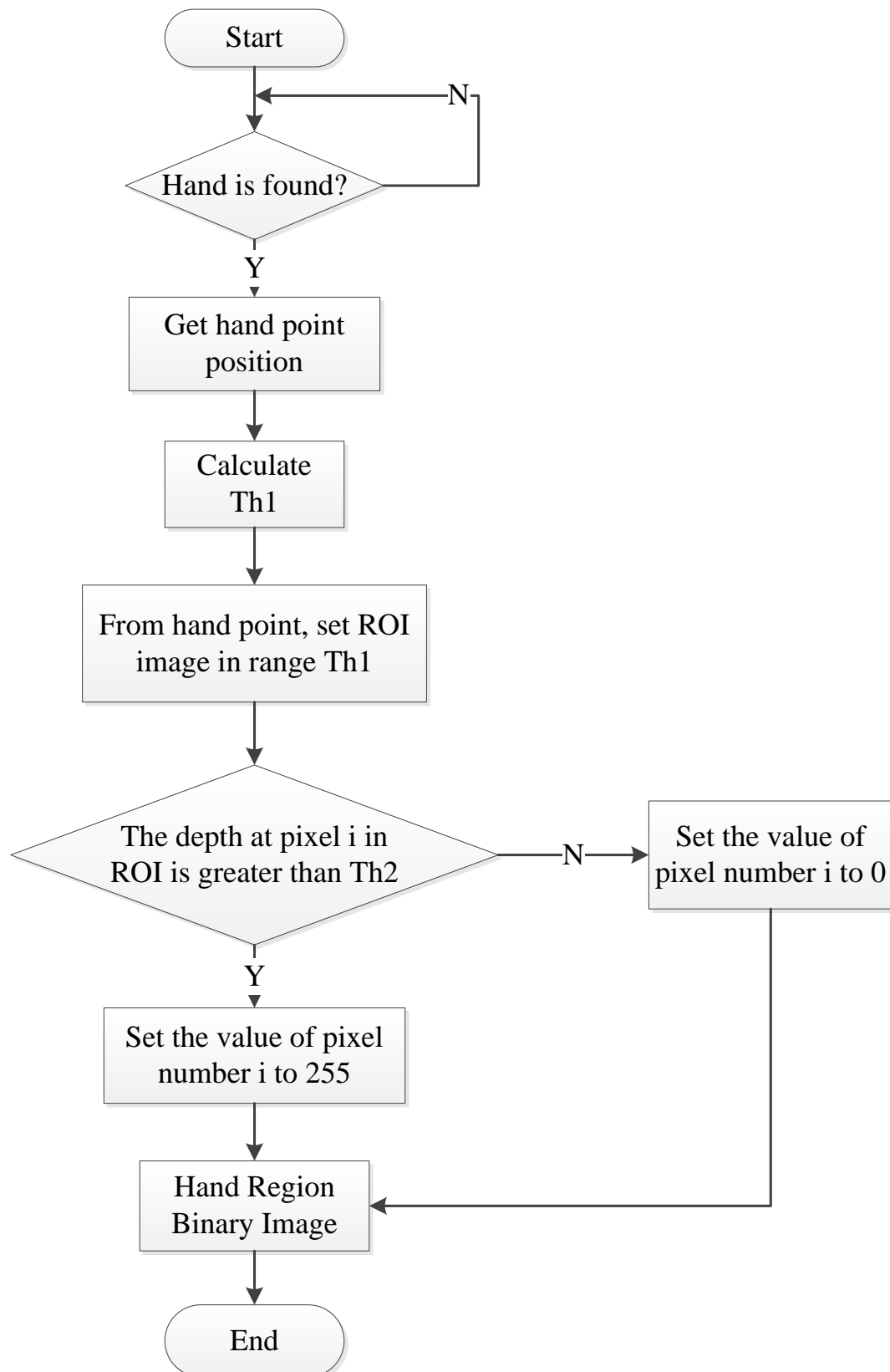


Hình 3-3 Vùng bàn tay có được khi đặt 2 ngưỡng T_{h1} và T_{h2}

Tổng số điểm ảnh trên vùng bàn tay sẽ thay đổi khi khoảng cách giữa bàn tay đến Camera thay đổi, chúng tôi đặt ra một hệ số tỉ lệ là Scl , nó sẽ áp dụng cho các thuật toán ở các bước xử lý sau.

$$Scl = \frac{5000}{D_h} \quad (2)$$

Giải thuật tách vùng bàn tay:



Hình 3-4 Giải thuật cho phương pháp trích ảnh bàn tay

3.2.2 Lọc hình thái và làm tròn ảnh

Trước khi triển khai các giải thuật trích đặc trưng cho ảnh vùng bàn tay được lấy ra ở trên, nhiều trên vùng ảnh bàn tay cần được loại bỏ. Để loại bỏ nhiều một cách hiệu quả, chúng tôi áp dụng lọc hình thái với phép đóng ảnh (giãn ảnh sau đó co ảnh) với phần tử cấu trúc hình elíp chiều rộng và cao đều bằng 4, gốc là tâm elíp. Tiếp theo chúng tôi sử dụng một bộ lọc trung vị với kích thước 7x7 để làm tròn biên ảnh.



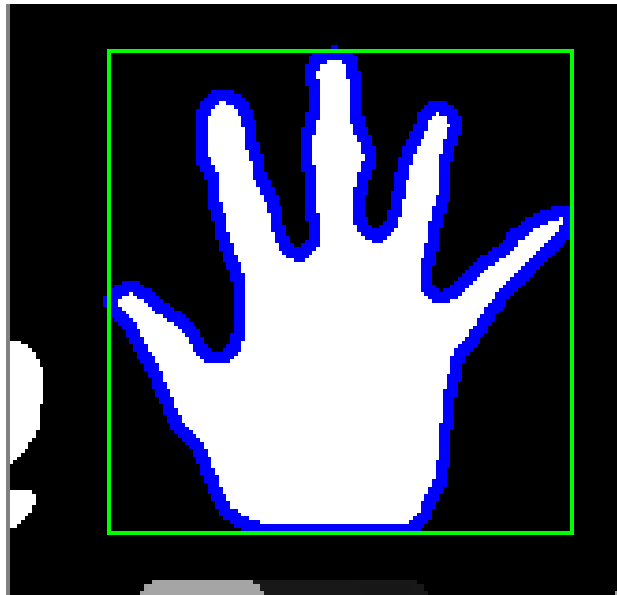
Hình 3-5 Ảnh vùng bàn tay sau khi khử nhiễu

3.3 Phương pháp trích đặc trưng bàn tay

3.3.1 Tìm đường bao bàn tay

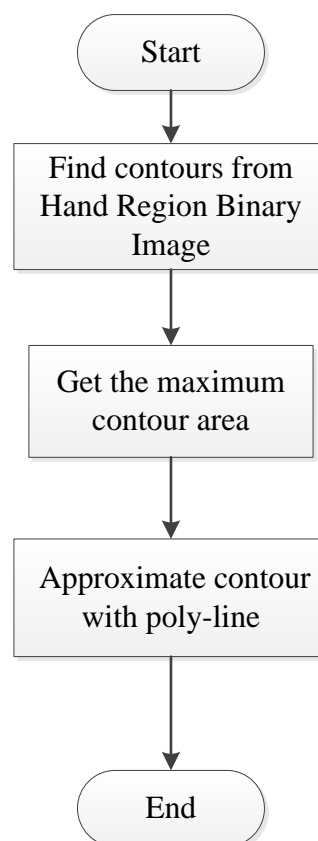
Vùng bàn tay sau khi được tách ra có thể còn bao gồm các ảnh nhỏ (nhiều), chúng tôi sử dụng thuật toán phân tích cấu trúc hình thái của tác giả SATOSHI SUZUKI và KEIICHI ABE [13] được tích hợp sẵn trong bộ thư viện mã nguồn mở OPENCV để tìm đường bao các ảnh. Ảnh có diện tích đường bao lớn nhất là chính là ảnh bàn tay. Ngõ ra của bước này là danh sách các điểm thuộc đường bao. Chúng tôi áp dụng thêm một phép xấp xỉ đa giác để giảm số lượng điểm ảnh, nhằm mục đích

giảm khối lượng tính toán cho các bước trích đặc trưng sau này mà có sử dụng đường bao.



Hình 3-6 Ảnh bàn tay với đường bao

Giải thuật thực hiện:



Hình 3-7 Giải thuật tách ảnh bàn tay

3.3.2 Tìm phương bàn tay và trọng tâm bàn tay

Trong luận văn này chúng tôi áp dụng tính chất mô ment của một hình [21], để tìm ra góc giữa phương ngang của bàn tay đối với trục nằm ngang của hình và trọng tâm của bàn tay.

Nhắc lại công thức tính mô ment của 1 ảnh xám:

$$m(p, q) = \sum_{i=1}^n I(x, y) x^p y^q \quad (3)$$

Trong đó $I(x, y)$ là giá trị tại tọa độ $I(x, y)$, n là tổng số điểm ảnh trên hình, $m(p, q)$ là mô ment bậc (p, q) . Trọng tâm của bàn tay sẽ được tính thông qua $m(1, 0)$, $m(0, 1)$ và $m(0, 0)$:

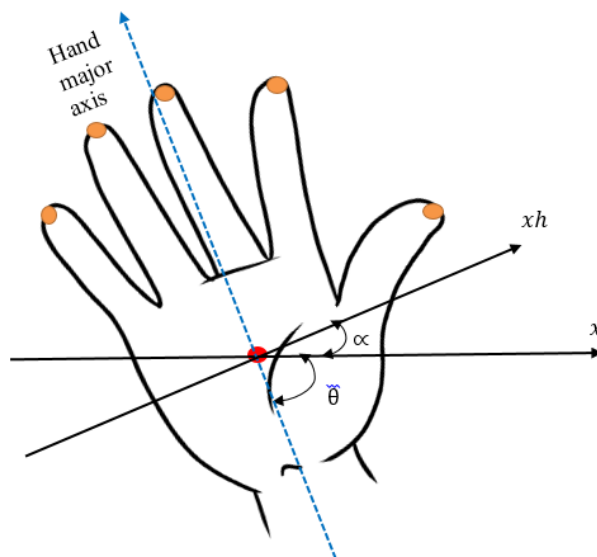
$$x_c = \frac{m(1, 0)}{m(0, 0)}, \quad y_c = \frac{m(0, 1)}{m(0, 0)} \quad (4)$$

Góc giữa trục đứng bàn tay và phương ngang của hình sẽ được tính theo công thức:

$$\theta = \frac{1}{2} \left(\arctan \left(\frac{2m(1, 1)}{m(2, 0) - m(0, 2)} \right) \right) \quad (5)$$

Từ đây góc giữa phương ngang bàn tay và phương ngang của hình sẽ được xác định:

$$\alpha = 90^\circ - \theta \quad (6)$$



Hình 3-8 Biểu diễn trục bàn tay và góc theo phương ngang

Góc phương ngang α sẽ được sử dụng để tính toán những đặc trưng về góc để khi bàn tay thay đổi hướng thì đặc trưng về góc thay đổi không nhiều.

3.3.3 Tìm tâm và bán kính lòng bàn tay

Tâm và bán kính lòng bàn tay được tìm ra nhằm mục đích chính để tính toán cho cái đại lượng đặc trưng khoảng cách. Vị trí trọng tâm ở mục 3.3.2 sẽ bị thay đổi khi chuyển đổi cử chỉ tay do mật độ điểm ảnh thay đổi khiến cho các đại lượng mô men thay đổi. Vì vậy cần có một phương pháp khác để tìm ra tâm lòng bàn tay với vị trí ổn định. Trong luận văn này chúng tôi sử dụng phương pháp tìm tâm bàn tay mà đã được áp dụng ở các bài báo [11], [12], [20]; đó là phương pháp tìm vòng tròn nội tiếp lớn nhất của bàn tay. Với mỗi điểm ảnh P_i trên bàn tay, khoảng cách nhỏ nhất D_{imin} của nó đến đường bao sẽ được tính toán. Điểm mà có giá trị khoảng cách lớn nhất trong tập khoảng cách nhỏ nhất đó chính là tâm lòng bàn tay P_h và khoảng cách đó chính là bán kính R . Vòng tròn nội tiếp lớn nhất đại diện cho vùng lòng bàn tay (Palm Region). Khối lượng tính toán giữa toàn điểm ảnh trên bàn tay và đường bao là rất lớn. Ví dụ, nếu chúng ta có N điểm ảnh trên bàn tay, M là số điểm ảnh trên đường bao, thì số vòng lặp sẽ là $N \times M + N$. Chúng tôi giảm khối lượng tính toán bằng cách chỉ xem xét những điểm ảnh lân cận trọng tâm bàn tay.

Giải thuật tìm tâm bàn tay và bán kính lòng bàn tay được mô tả theo những công thức như sau:

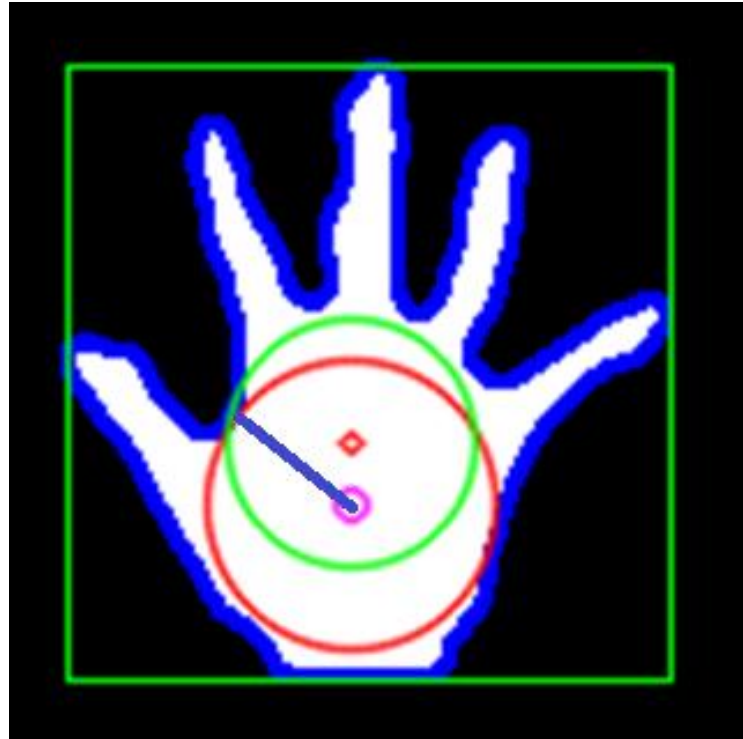
$$G = \{P_i \mid (d(P_i, P_c) < T_{h3})\} \quad (7)$$

G là tập hợp tất cả những điểm cần xét để tìm tâm và bán kính bàn tay. $d(P_i, P_c)$ là khoảng cách Euclid từ điểm trên bàn tay đến điểm trọng tâm của bàn tay P_c cái mà được tính ở công thức (4) mục 3.3.2. T_{h3} là ngưỡng khoảng cách Euclid với $T_{h3} = 4 * Scl$.

$$D_{imax} = \text{Min}(d(P_i, P_b(j)) \mid (P_i \in G \cap P_b(j) \in C)) \quad (8)$$

Trong đó C là tập hợp tất cả các điểm thuộc đường bao, $P_b(j)$ thứ j trên đường bao C đó. Bán kính lòng bàn tay R sẽ là:

$$R = \text{Max}(D_{imax} \mid P_i \in G) \quad (9)$$



Hình 3-9 Trọng tâm (điểm màu đỏ), tâm bàn tay (màu tím) và vòng tròn nội tiếp lớn nhất (vòng tròn màu đỏ)

Kết quả của giải thuật cho thấy tâm lòng bàn tay ổn định khi thay đổi các cử chỉ khác nhau.

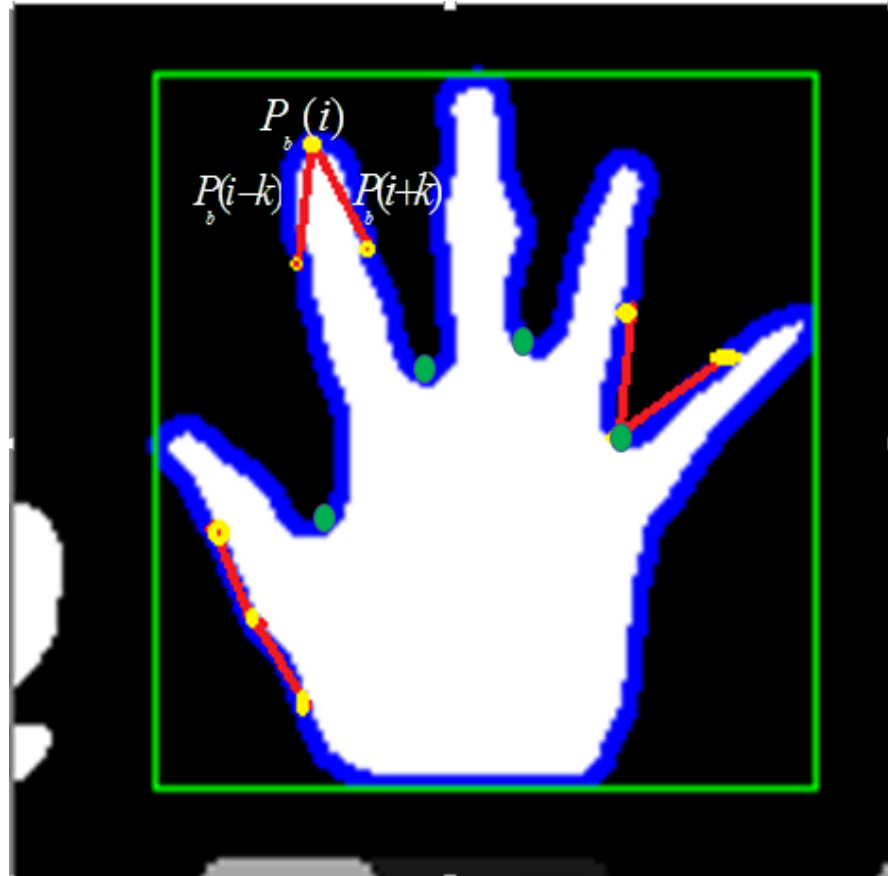
3.3.4 Tìm các đầu ngón tay

Chúng tôi cải tiến phương pháp k-curvature để tìm ra các đầu ngón tay mà các bài báo trước đó ([18], [19], [20]) đã áp dụng. K-curvature là phương pháp tính góc giữa 2 vector trên đường bao. Tại mỗi điểm $P_b(i)$ trên tập điểm mô tả đường bao, góc giữa 2 véc tơ $P_b(i-k)$ và $P_b(i+k)$ theo hướng ngược nhau sẽ được tìm ra. Những điểm đầu ngón tay và kẽ tay sẽ có góc k-curvature nhỏ thua một ngưỡng được chỉ ra. Thay vì chọn các hệ số k và góc ngưỡng là cố định (Bài báo số [18] chọn $k = 10$, góc ngưỡng là 30° ; còn báo số [20] chọn K bằng 20 và góc ngưỡng là 60°), trong luận văn này chúng tôi chọn chúng tùy thuộc vào sự thay đổi của độ sâu bàn tay đối với cảm biến mà được đại diện bởi hệ số Scl được chỉ ra ở công thức số (2) mục 3.3.1:

$$\text{Hệ số k-curvature} : k = 2 * Scl \quad (10)$$

Góc ngưỡng k- curvature: $\theta_k = \frac{75}{S_{cl}} + 60^\circ$ (11)

Hai công thức trên được chúng tôi rút ra bằng thực nghiệm khi thực hiện thuật toán trong khoảng cách từ 50cm – 1. Hình dưới đây miêu tả thuật toán k-curvature:



Hình 3-10 Thuật toán K-curvature

Những điểm ở kẽ tay (màu xanh lá) có khoảng cách nhỏ thua 1.2 lần bán kính vòng tròn nội tiếp tìm ra ở mục 3.3.3 sẽ bị loại bỏ. Những đầu ngón tay sử dụng thuật toán k-curvature được tìm ra cho kết quả một cách chính xác trong phạm vi 50 cm – 1 m.

3.4 Phương pháp phân loại cử chỉ

Từ những đặc trưng về hình ảnh bàn tay tìm được ở trên, chúng tôi định nghĩa 1 vector đặc trưng gồm 15 phần tử như sau:

$$\mathbf{Vf} = \{N, \alpha[i], \beta[j], D[i]\} \quad (12)$$

Trong đó:

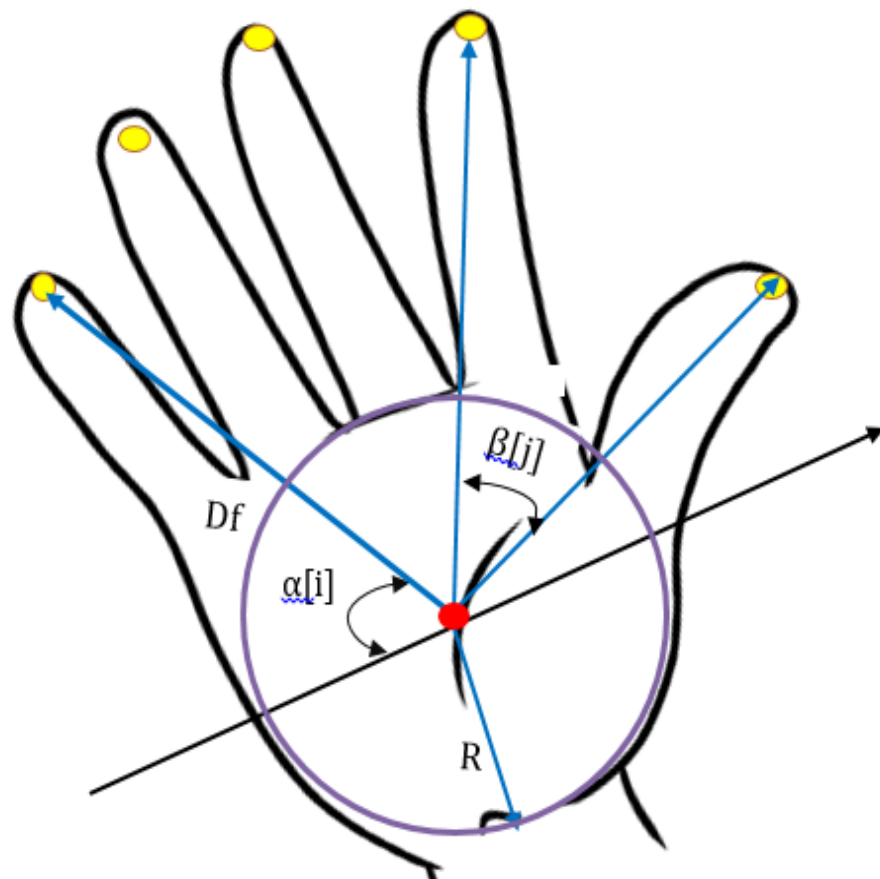
N : là số các ngón tay được mở ra.

$\alpha[i]$: là góc từ tâm bàn tay cho tới các đầu ngón tay so với trục ngang của bàn tay, $i = 0 \dots 4$

$\beta[j]$: là góc giữa các ngón tay liên tiếp với nhau, $j = 0 \dots 3$

$D[i]$: là hiệu của khoảng cách từ tâm bàn tay đến các đầu ngón tay thứ i ($Df[i]$) và bán kính đường tròn nội tiếp lớn nhất (R). Nó cũng được tỉ

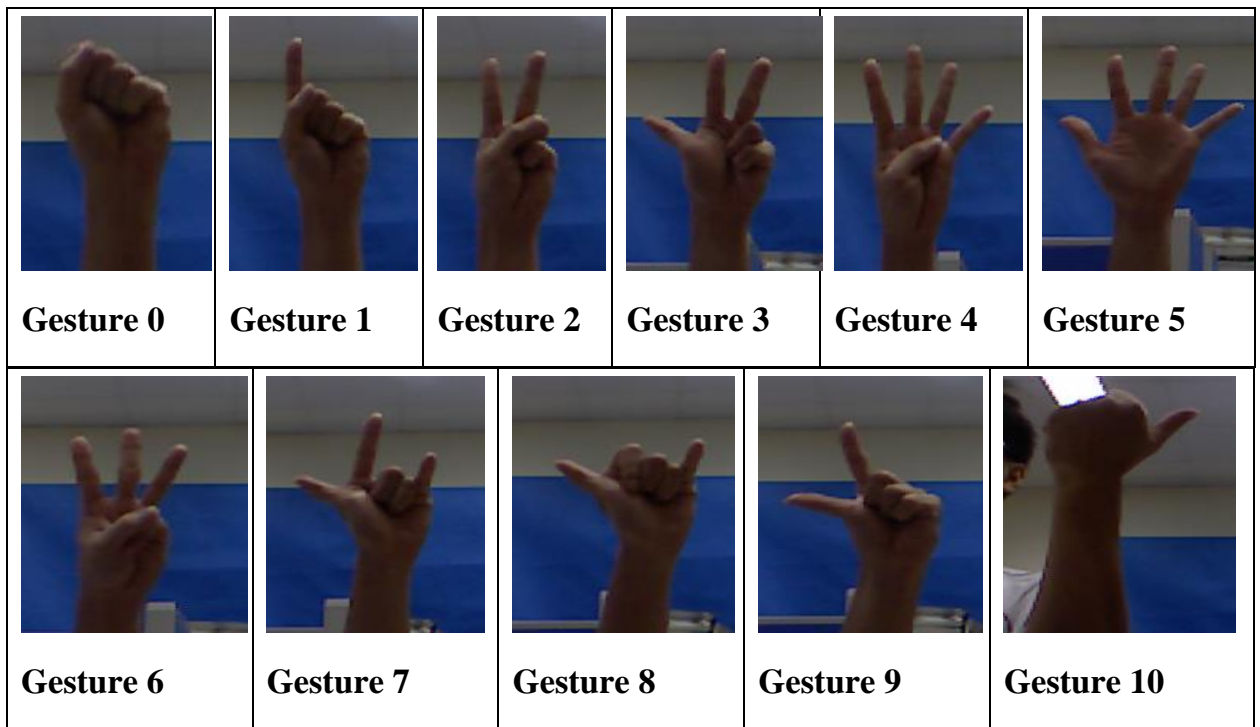
lệ với độ sâu: $D[i] = \frac{(Df[i] - R)}{Scl}$; $i = 0 \dots 4$ (13)



Hình 3-11 Các đặc trưng của bàn tay được trích ra

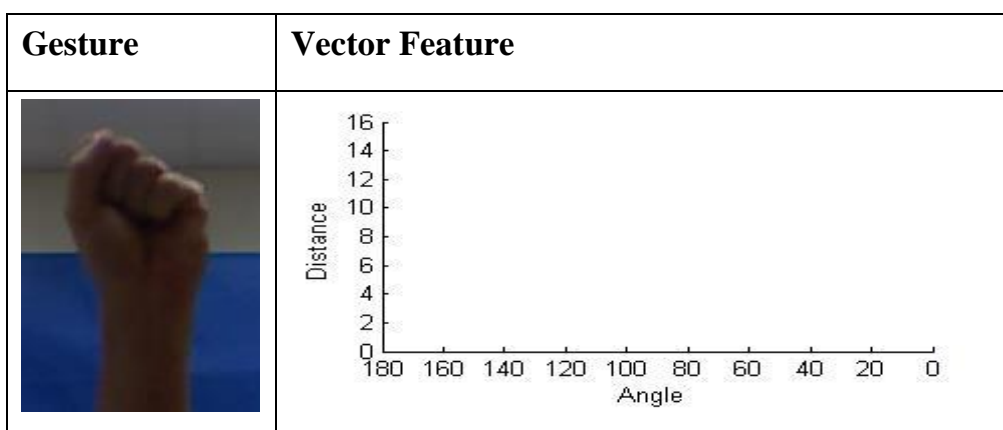
Mỗi cử chỉ tay sẽ tương ứng với một vector đặc trưng V_f . Sau khi xây dựng 1 dữ liệu huấn luyện từ các mẫu, chúng tôi áp dụng một máy hỗ trợ vector (SVM) dựa trên phương pháp one-vs-one (mục 2.2.3) để phân loại các cử chỉ khác nhau. Hệ

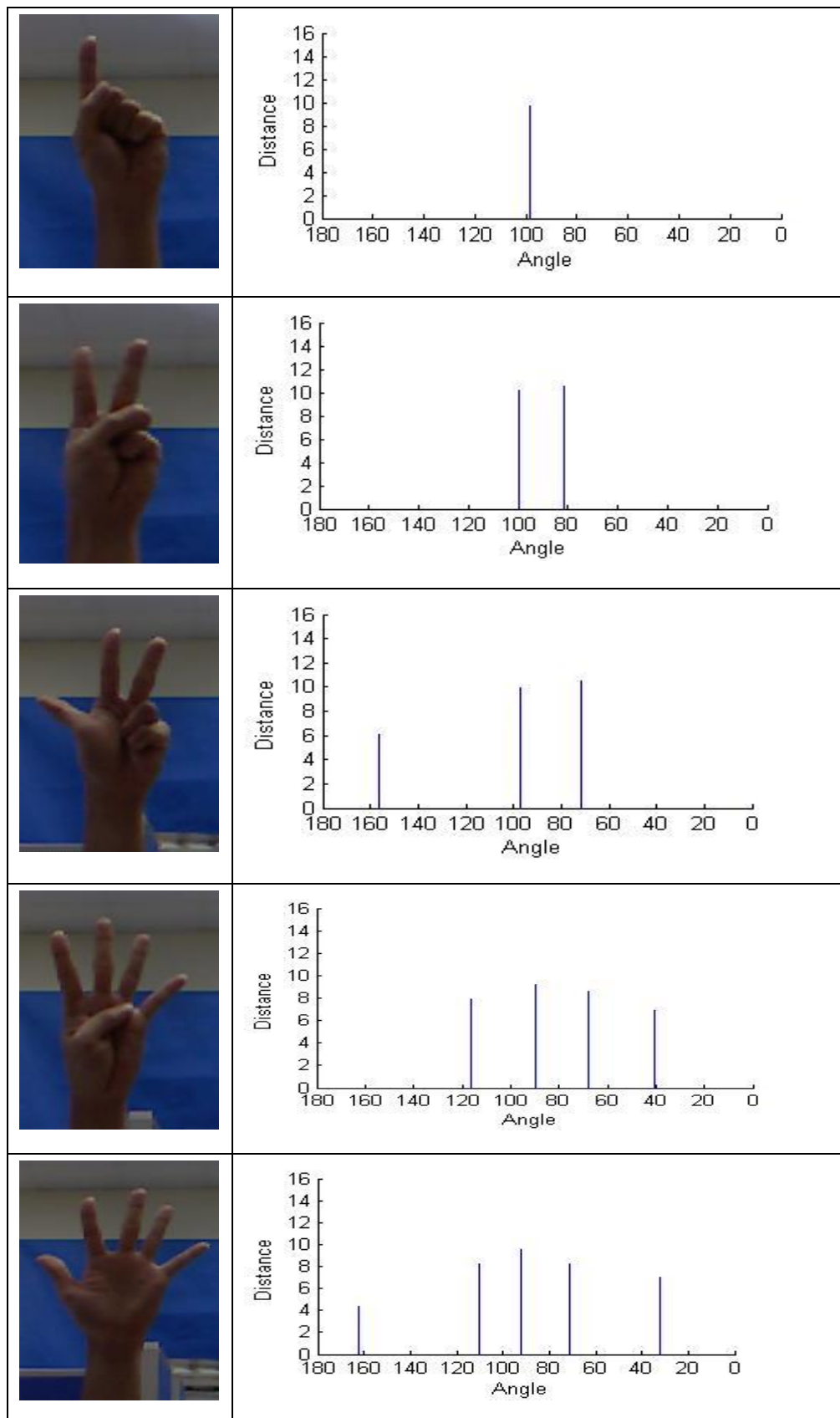
thống nhận dạng cử chỉ tay mà chúng tôi đề xuất cho phép nhận dạng 11 cử chỉ khác nhau dưới đây:

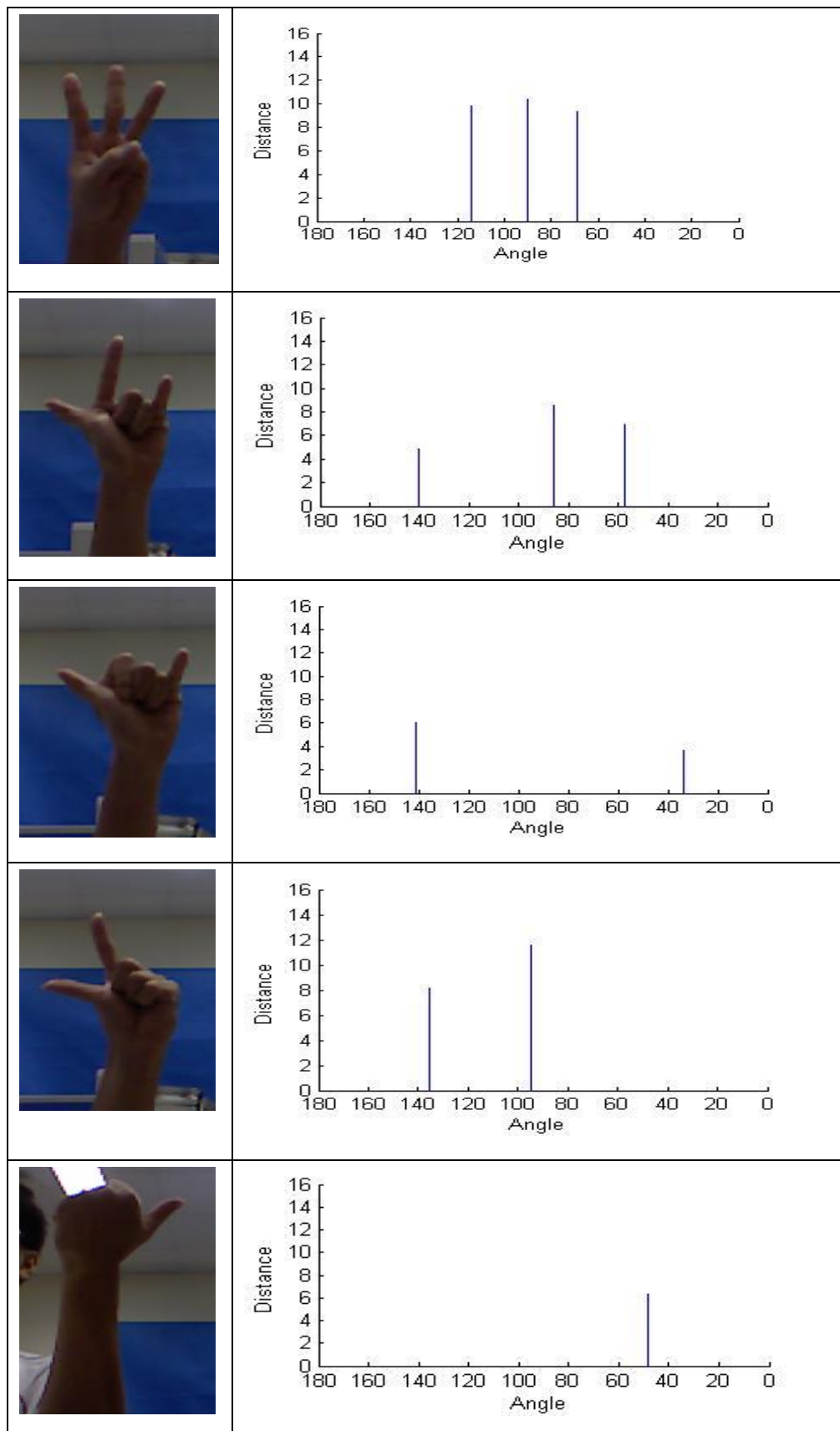


Hình 3-12 Mười một cử chỉ khác nhau được chứa trong cơ sở dữ liệu huấn luyện

Vectơ đặc trưng của một mẫu với cử chỉ tương ứng được miêu tả ở giản đồ ở hình 3-13. Trong đó, trục đứng là đặc trưng về hiệu khoảng cách, trục ngang là đặc trưng về góc.





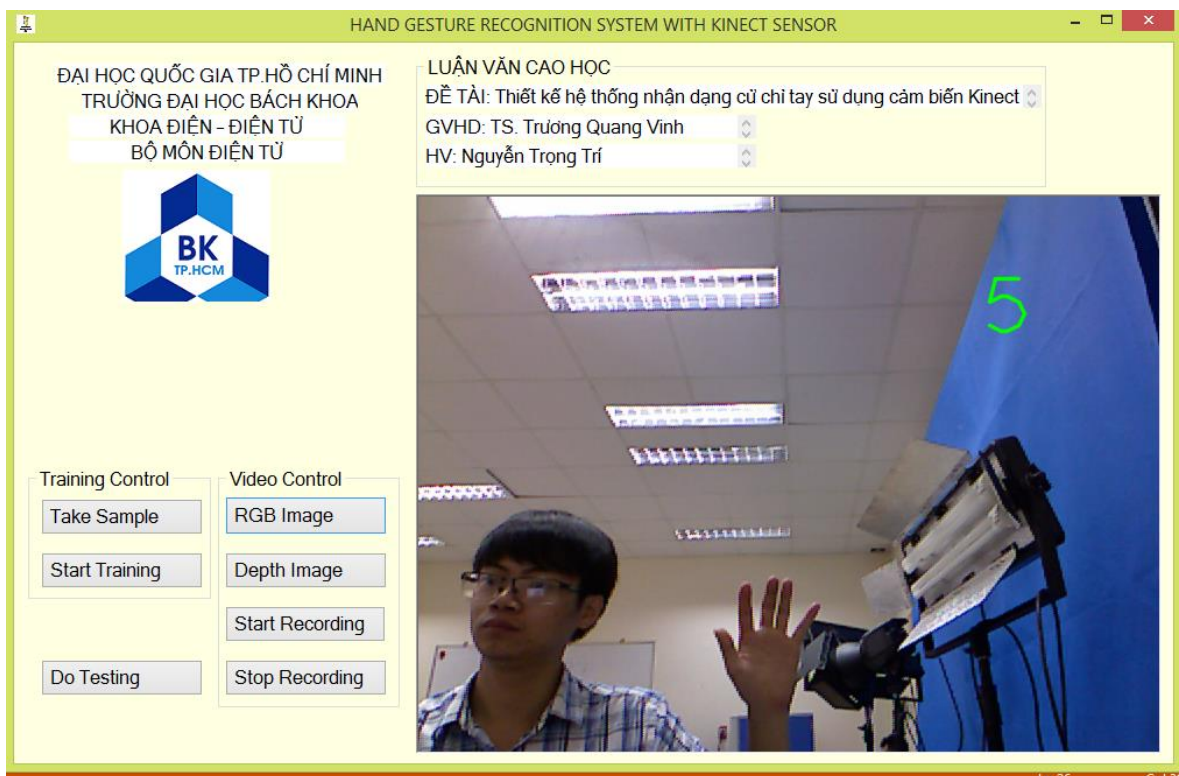


Hình 3-13 Giảm đồ biểu diễn véc tơ đặc trưng của 11 cử chỉ

CHƯƠNG IV: KẾT QUẢ THỰC HIỆN GIẢI THUẬT

4.1 Kết quả hiện thực hệ thống

Hệ thống nhận dạng cử chỉ tay của chúng tôi được hiện thực trên máy tính Intel core I5-560M, RAM 4G với hệ điều hành Windows 8.1. Giải thuật được phát triển trên môi trường Visual Studio 2012 với ngôn ngữ C++ cho phần giải thuật và C# cho phần giao diện. Chúng tôi sử dụng thư viện OPENNI2 cho phần kết nối cảm biến và máy tính, thư viện NITE2 cho phần theo dấu bàn tay và thư viện OPENCV cho một số thuật toán xử lý ảnh và phần nhận dạng sử dụng SVM. Kết quả phần mềm hiện thực giải thuật thể hiện ở hình dưới đây.



Hình 4-1 Phần mềm hiện thực giải thuật đề xuất

Các nút điều khiển trên phần mềm:

Phím điều khiển	Mô tả chức năng
RGB Image	Cho phép hiển thị ảnh màu
Depth Image	Cho phép hiển thị ảnh màu

Take Sample	Lấy mẫu ảnh độ sâu và vị trí bàn tay
Start Recording	Bắt đầu ghi video
Stop Recording	Dừng ghi video
Start Training	Cho phép huấn luyện mẫu
Do Testing	Cho phép thực hiện kiểm tra từ các mẫu thu được



Hình 4-2 Hình ảnh độ sâu hiển thị trên phần mềm

4.2 Đánh giá hệ thống nhận dạng cử chỉ tay đề xuất

Để đánh giá kết quả của hệ thống nhận dạng cử chỉ tay của chúng tôi đề xuất, chúng tôi mời 5 học viên trong IC-Lab trường đại học trong độ tuổi 20-30 thực hiện các 11 cử chỉ ở trên trong phạm vi 50cm-1m, mỗi cử chỉ thực hiện bởi từng người chúng tôi lấy mẫu ít nhất 10. Tổng cộng chúng tôi có 620 mẫu để huấn luyện cho SVM.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	96.436	0.000	0.000	0.000	0.000	8.316	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1
1	101.000	0.000	0.000	0.000	0.000	8.584	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1
1	97.106	0.000	0.000	0.000	0.000	9.929	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1
2	103.248	83.194	0.000	0.000	0.000	8.414	7.601	0.000	0.000	0.000	19.147	0.000	0.000	0.000	2
2	102.260	82.533	0.000	0.000	0.000	9.545	9.280	0.000	0.000	0.000	18.289	0.000	0.000	0.000	2
2	96.627	76.045	0.000	0.000	0.000	10.124	10.844	0.000	0.000	0.000	17.133	0.000	0.000	0.000	2
3	174.196	116.730	88.041	0.000	0.000	6.742	10.903	8.435	0.000	0.000	57.527	29.577	0.000	0.000	3
3	144.985	94.975	67.959	0.000	0.000	6.675	8.582	8.935	0.000	0.000	44.979	25.645	0.000	0.000	3
3	151.344	95.003	73.979	0.000	0.000	5.759	9.137	6.639	0.000	0.000	52.534	19.357	0.000	0.000	3
3	160.870	97.919	68.458	0.000	0.000	6.313	10.604	9.983	0.000	0.000	51.849	25.221	0.000	0.000	3
4	118.387	94.014	73.212	37.849	0.000	7.510	8.500	6.992	6.189	0.000	25.006	21.499	35.097	0.000	4
4	124.534	103.182	75.303	48.571	0.000	6.895	6.393	7.682	7.246	0.000	21.352	27.879	26.733	0.000	4
4	123.773	94.496	72.646	44.310	0.000	6.961	8.634	8.797	6.720	0.000	27.761	20.806	26.794	0.000	4
5	183.467	123.174	99.525	76.332	35.520	5.389	8.547	9.468	8.951	7.803	50.659	19.646	19.428	35.507	5
5	161.133	119.652	91.731	73.383	36.639	4.005	8.228	10.432	9.225	7.711	33.814	23.629	15.730	31.201	5
5	190.103	135.264	103.027	79.891	36.084	5.312	7.207	9.369	9.080	7.114	45.913	25.953	19.211	37.322	5
3	106.163	82.095	59.808	0.000	0.000	10.340	10.845	9.154	0.000	0.000	19.964	18.122	0.000	0.000	6
3	104.254	82.693	61.310	0.000	0.000	9.818	10.121	8.818	0.000	0.000	18.058	17.645	0.000	0.000	6
3	114.094	87.294	63.195	0.000	0.000	9.189	9.637	8.729	0.000	0.000	22.233	20.022	0.000	0.000	6
3	141.058	87.581	42.461	0.000	0.000	5.515	9.091	4.015	0.000	0.000	49.887	41.604	0.000	0.000	7
3	136.799	88.202	37.406	0.000	0.000	5.466	8.646	3.702	0.000	0.000	47.771	50.307	0.000	0.000	7
3	137.447	91.727	39.682	0.000	0.000	4.951	8.470	4.238	0.000	0.000	44.700	51.173	0.000	0.000	7
2	165.185	64.593	0.000	0.000	0.000	6.212	4.221	0.000	0.000	0.000	114.067	0.000	0.000	0.000	8
2	140.896	46.248	0.000	0.000	0.000	6.361	4.633	0.000	0.000	0.000	113.820	0.000	0.000	0.000	8
2	146.117	37.662	0.000	0.000	0.000	5.721	3.794	0.000	0.000	0.000	122.284	0.000	0.000	0.000	8
2	165.969	83.502	0.000	0.000	0.000	6.694	9.883	0.000	0.000	0.000	85.918	0.000	0.000	0.000	9
2	170.623	83.439	0.000	0.000	0.000	6.417	10.265	0.000	0.000	0.000	89.807	0.000	0.000	0.000	9
2	154.746	75.496	0.000	0.000	0.000	7.130	10.026	0.000	0.000	0.000	82.128	0.000	0.000	0.000	9
1	59.336	0.000	0.000	0.000	0.000	6.185	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	10
1	62.306	0.000	0.000	0.000	0.000	5.746	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	10
1	51.717	0.000	0.000	0.000	0.000	5.516	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	10

Hình 4-3 Dữ liệu huấn luyện

(Cột A-O là các đặc trưng, cột P là cử chỉ tương ứng)

Sau đó chúng tôi thực hiện 2 phương pháp để đánh giá cho hệ thống: phương pháp kiểm tra mẫu và phương pháp trực quan.

Phương pháp kiểm tra mẫu:


Phương pháp thứ nhất, chúng tôi cũng lấy mẫu trên 9 người khác nhau, mỗi người thực hiện 11 cử chỉ khác nhau, mỗi cử chỉ cũng lặp lại khoảng trên 10 lần, tổng cộng có 1926 mẫu ảnh độ sâu với độ sâu và tọa độ điểm trên bàn tay tương ứng. Kết quả cho việc đánh giá phương pháp pháp thứ nhất phản ánh bởi bảng Confusion Matrix dưới đây:

	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	112	3									
G1	1	105	4								8
G2		2	144						2	9	
G3				221				15			
G4					120	5	2				
G5					5	139	1				
G6				2			185	2			
G7				12	1		2	230		1	
G8			1		2				210	7	1
G9								1	14	235	
G10			1								121
Accuracy	99,12%	95,5%	96%	94%	93,8%	97%	97,4%	93%	93%	93,3%	93,08%

Hình 4-4 Độ chính xác của hệ thống qua phương pháp kiểm tra mẫu

Phương pháp trực quan:

Trong phương pháp đánh giá thứ 2, chúng tôi mời 10 người thực hiện các cử chỉ và lưu lại các video họ thực hiện. Ở phương pháp này chúng tôi thực hiện kết hợp các khung hình liên tiếp với nhau để cho ra kết quả cử chỉ. Nếu 3 khung hình liên tiếp cho ra cùng 1 cử chỉ thì cử chỉ mới được xuất ra, còn không thì vẫn giữ cử chỉ trước đó. Mặc dù có độ trễ trong việc nhận dạng, nhưng nó vẫn chấp nhận được khi hiển thị trong thời gian thực. Độ chính xác kiểm nghiệm của phương pháp thứ 2 đối với từng người sử dụng:

Người dùng	Kết quả										
	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
	1										0.1
		1									
			1				0.1				
				1							
					1						
						1					
							0.9				
								1			
									1		
										1	
											0.9

	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										
G1		1									
G2			1								
G3				1							
G4					1						
G5						1					
G6							1				
G7								1			
G8									1		
G9										1	
G10											1

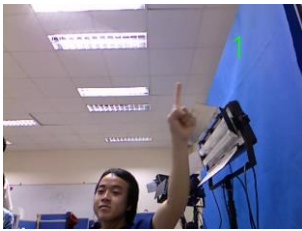
	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										
G1		1									
G2			1								
G3				1							
G4					0.9						
G5						1					
G6							1				
G7								1			
G8					0.1				1		
G9										1	
G10											1


	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										
G1		1									
G2			1								
G3				1							
G4					1						
G5						1					
G6							1				
G7								1			
G8									1		
G9										1	
G10											1

	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										
G1		1									
G2			1	0.1							
G3				0.9			0.2	0.2			
G4					0.8						
G5						1					
G6					0.1		0.7				
G7							0.1	0.7			
G8									0.9		
G9					0.1			0.1		1	
G10											1

	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										
G1		1									
G2			1								
G3				1							
G4					1						
G5						1					
G6							1				
G7								1			
G8									1		
G9										1	
G10											1

	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										
G1		1									
G2			1								
G3				1							
G4					1						
G5						1					
G6							1				
G7								1			
G8									1		
G9										1	
G10											1

		G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
	G0	1										
	G1		1									
	G2			1								
	G3				1							
	G4					1						
	G5						1					
	G6							1				
	G7								1			
	G8									1		
	G9										1	
	G10											1

		G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
	G0	1										
	G1		1									
	G2			1				0.1				
	G3				1							
	G4					1						
	G5						1					
	G6							1				
	G7								1			
	G8									1	0.2	
	G9										0.8	
	G10											1

		G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
	G0	1										
	G1		1									
	G2			1								
	G3				1							
	G4					1						
	G5						1					
	G6							1				
	G7								1			
	G8									1		
	G9										1	0.3
	G10											0.7

Hình 4-5 Độ chính xác của hệ thống qua phương pháp đánh giá trực quan của từng người sử dụng

Trung bình kết quả của tất cả các lần thử nghiệm ở trên:

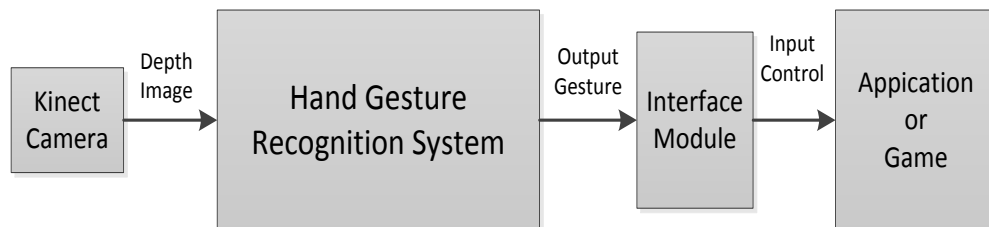
	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G0	1										0.01
G1		1									
G2			1	0.01			0.02				
G3				0.99			0.02	0.02			
G4					0.97						
G5						1					
G6					0.01		0.95				
G7								0.97			
G8					0.01		0.01		0.99	0.02	
G9					0.01			0.01	0.01	0.98	0.03
G10											0.96

Hình 4-5 Độ chính xác trung bình của hệ thống qua phương pháp đánh giá trực quan

Hệ thống cho phép nhận dạng 11 cử chỉ tay với độ chính xác trên 95% trong thời gian thực quan sát.

4.3 Ứng dụng dựa trên hệ thống cử chỉ tay


Dựa vào những cử chỉ có được qua hệ thống nhận dạng, chúng tôi sử dụng những cử chỉ đó làm điều khiển ngõ vào cho các ứng dụng hoặc trò chơi trên máy tính. Vì giới hạn về thời gian thực hiện, chúng tôi chỉ hiện thực 1 ứng dụng đơn giản búa, kéo, giấy.

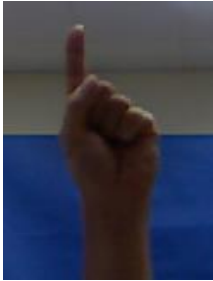
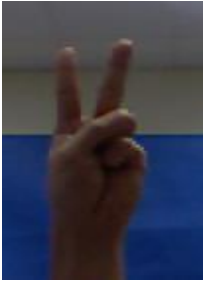




Hình 4-6 Hệ thống điều khiển trò chơi

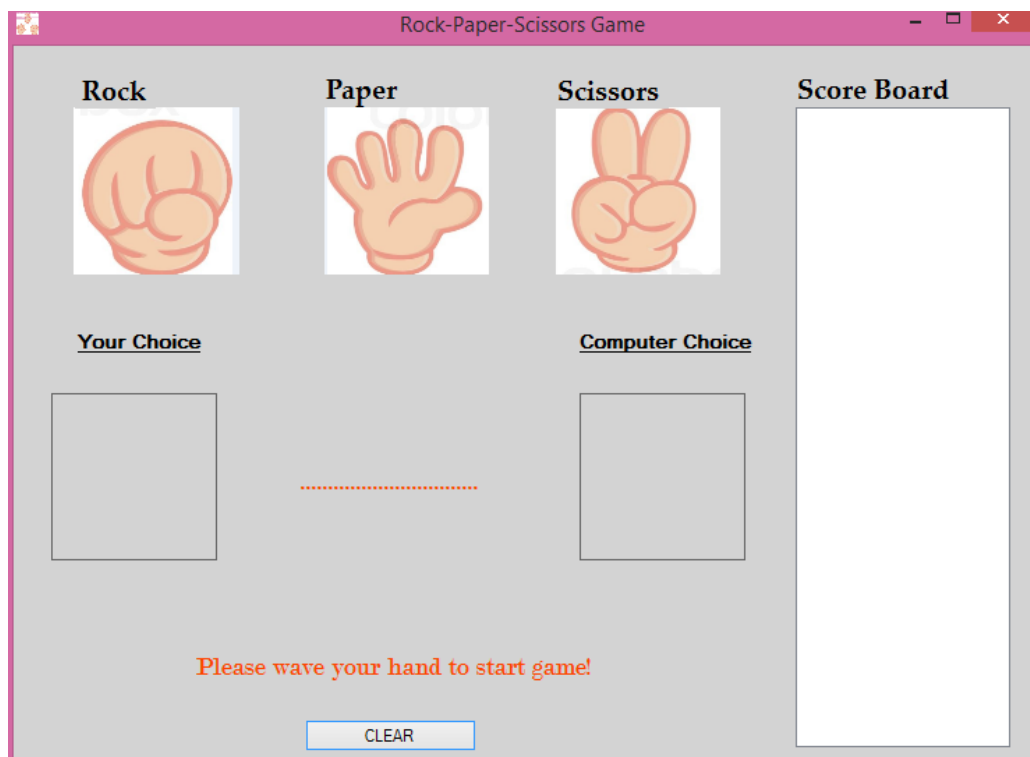
hoặc ứng dụng trên máy tính sử dụng Kinect

Trong luận văn này chúng tôi thử nghiệm hệ thống nhận dạng cử chỉ tay trên trò chơi “búa, kéo, giấy”, trò chơi mà cũng được thử nghiệm ở bài báo số [22], với các cử chỉ điều khiển như hình 4-8. Sau khi người dùng tùy chọn cử chỉ của mình, máy tính sẽ chọn cử chỉ của nó một cách ngẫu nhiên, và kết quả sẽ được tính toán và được hiển thị trên bảng điểm.

Cử chỉ điều khiển	Mô tả chức năng	Cử chỉ điều khiển	Mô tả chức năng
	Bắt đầu trò chơi		Búa

	Bắt đầu lựa chọn cử chỉ		Kéo
	Xóa kết quả		Giấy

Hình 4-7 Bảng điều khiển cho trò chơi “búa, kéo, giấy”



Hình 4-8 Trò chơi “búa, kéo, giấy” sử dụng cử chỉ tay

Trò chơi được chúng tôi kiểm tra trên 5 người, kết quả thông qua quan sát việc trải nghiệm của người dùng, dường như không có sự sai sót về các cử chỉ nào. Người dùng trải nghiệm ứng dụng một cách thoải mái và tự nhiên.

CHƯƠNG V: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong luận văn này chúng tôi đã phát triển một phương pháp nhận dạng cử chỉ tay hiệu quả, chính xác trong thời gian thực sử dụng thông tin độ sâu từ cảm biến Kinect. Người dùng có thể tương tác với ứng dụng và trò chơi trên máy tính bằng cách hiện thực cử chỉ tay thay vì sử dụng chuột hay bàn phím. Điều này mang tới một trải nghiệm thú vị cho người sử dụng.

Hệ thống chúng tôi đề xuất được phân lớp qua ba bước chính: trích ảnh, tìm vector đặc trưng và nhận dạng sử dụng SVM. Bước đầu tiên, từ ảnh độ sâu thu được từ Kinect, chúng tôi tách vùng bàn tay một cách dễ dàng sử dụng thư viện OPEN2/NITE2. Sau đó các đặc trưng như phương bàn tay, bán kính, tâm lòng bàn tay và các đỉnh ngón tay được tìm ra. Dựa vào những đặc trưng đó chúng tôi xây dựng một vector đặc trưng bao gồm đặc trưng về khoảng cách và góc biểu diễn cho một cử chỉ. Cuối cùng chúng tôi áp dụng SVM cho việc nhận dạng các cử chỉ khác nhau

Phương pháp của chúng tôi đề xuất tương đối đơn giản nhưng vẫn mang tới chính xác cao trong việc nhận dạng, độ chính xác của việc nhận dạng đạt trên 98% trong thời gian thực quan sát và trong một số giới hạn về khoảng cách và cử chỉ. Hệ thống chúng tôi có thể hoạt động trong bóng tối, không phụ thuộc vào môi trường ánh sáng và nhận dạng một cách đáng tin cậy.

5.2 Hướng phát triển

Độ chính xác có thể nâng cao nếu cải thiện tốt hơn trong các quá trình tiền xử lý, trích chọn đặc trưng và bổ sung bộ dữ liệu huấn luyện đa dạng hơn. Áp dụng thuật toán huấn luyện khác như Hidden-Markov Model (HMM) có thể cho kết quả tốt hơn SVM. Hệ thống có thể mở rộng tập cử chỉ bằng cách lấy quy định các cử chỉ khác với một trong những cử chỉ trong tập 11 cử chỉ trên, sau đó lấy mẫu và huấn luyện.

Về việc phát triển ứng dụng, hiện tại ứng dụng trên máy tính cần lập trình chung với hệ thống nhận dạng cử chỉ tay mới có thể áp dụng cử chỉ tay được. Vì thế để mở rộng cho hệ thống nhận dạng cử chỉ có thể tương tác với các ứng dụng bên thứ ba cần sử dụng bộ thư viện TUIO để làm cầu nối cho các ứng dụng độc lập trên

Windows có thể tương tác với nhau. Khi sử dụng bộ thư viện này, bất kì ứng dụng nào cũng có thể sử dụng cử chỉ tay để điều khiển.

Thư viện OPENNI/OPENCV hỗ trợ cho việc phát triển trên ARM-LINUX, nên việc phát triển hệ thống trên nền ARM hoàn toàn có thể hiện thực được. Với việc triển khai trên ARM, các ứng dụng cho cử chỉ tay sẽ linh động hơn, có thể mở rộng các đề tài như điều khiển nhà thông minh, thang máy...

Tài liệu tham khảo

- [1] Mokhtar M. Hasan and Pramoud K. Misra, "Brightness Factor Matching for Gesture Recognition System using Scaled Normalization," *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 3, no.2, 2011.
- [2] Mokhtar M. Hasan and Pramoud K. Misra, "Robust Gesture Recognition Using Gaussian Distribution for Features Fitting," *International Journal of Machine Learning and Computing*, vol. 2, No. 3, June 2012.
- [3] Parul Chaudhary, Hardeep Singh Ryait, "Neural Network Based Static Sign Gesture Recognition System," *International Journal of Innovative Research in Computer and Communication Engineering*, February 2014 .
- [4] Luigi Lamberti, Francesco Camastra, "Real-Time Hand Gesture Recognition using a Color Glove," *Springer Proceedings of the 16th international conference on Image analysis and processing*., 2011.
- [5] Robert Y. Wang and Jovan Popović, "Real-Time Hand-Tracking with a Color Glove," *ACM Transactions on Graphics* 28(3), Article 63, 2009.
- [6] Lale Akarun *et al.*, "Real Time Hand Pose Estimation using Depth Sensors," *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Nov. 2011.
- [7] Z. Ren, J. Yuan, and Z. Zhang, "Robust Hand Gesture Recognition Based on FingerEarth Mover's Distance with a Commodity Depth Camera," *In Proc. of ACM Conference on Multimedia*, vol. ACM, pp. 1093 - 109, 2011.
- [8] Fabio Domini, Mauro Donadeo, Giulio Marin, "Hand Gesture Recognition with Depth Data," *ARTEMIS '13 Proceedings of the 4th ACM/IEEE international workshop on Analysis and retrieval of tracked events and motion in imagery stream*, pp. 9-16, 2013.

- [9] Abidatul Izzah and Nanik Suciati, "Translation of Sign Language using Generic Fourier Descriptor and Nearest Neighbour," *International Journal on Cybernetics & Informatics (IJCI)*, vol. 3, No.1, February 2014.
- [10] Zhou Ren, Jingjing Meng, Junsong Yuan, "Depth Camera Based Hand Gesture Recognition and its Applications in Human-Computer-Interaction," *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pp. 1 - 5, Dec. 2011.
- [11] Yan Wen, Chuanyan Hu, Guanghui Yu, Changbo Wang, "A Robust Method of Detecting Hand Gestures Using Depth Sensors," *Haptic Audio Visual Environments and Games (HAVE), 2012 IEEE International Workshop on*, 2012.
- [12] Li, Yi., "Hand gesture recognition using Kinect," *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, 2012.
- [13] Suzuki, S. and Abe, K., , "Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46," 1985.
- [14] K. Stergios Poularakis, "Finger Detection and Hand Posture Recognition Based on Depth Information," *IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 2014.
- [15] Jagdish L. Raheja *et al.*, "Tracking of Fingertips and Centers of Palm Using KINECT," *Computational Intelligence, Modelling and Simulation (CIMSIM), 2011 Third International Conference on*, 2011.
- [16] Arun Kulshreshth, Chris Zorn, Joseph J. LaViola Jr., "Real-time Markerless Kinect based Finger Tracking and Hand Gesture Recognition for HCI," *IEEE Symposium on 3D User Interfaces 2013*, 2013.
- [17] W.-c. Chen, "Real-Time Palm Tracking and Hand Gesture Estimation," *Master Thesis*, 2011.

- [18] Seung Il Han *et al.*, "Vision Based hand tracking for Interaction," <http://kowon.dongseo.ac.kr>, 2008.
- [19] Thiago R. Trigo and Sergio Roberto M. Pellegrino, "An Analysis of Features for Hand-Gesture Classification," *IWSSIP 2010 - 17th International Conference on Systems, Signals and Image Processing*, 2010 .
- [20] Hui-Shyong Yeo *et al.*, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *Springer Science+Business Media New York 2013*, 2013.
- [21] J. Kilian, "Simple Image Analysis By Moments," March 15, 2001.
- [22] Zhou Ren *et al.*, "Robust Hand Gesture Recognition with Kinect Sensor," *In Proc. of ACM MM*, 2011.