ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

QUÁCH CÔNG HOÀNG

ỨNG DỤNG CẢM BIẾN 3D KINECT TRONG NHẬN DIỆN NGÔN NGỮ CỬ CHỈ TIẾNG VIỆT HỖ TRỢ VIỆC GIAO TIẾP VỚI NGƯỜI KHUYẾT TẬT KHIẾM THÍNH

LUẬN VĂN THẠC SĨ CÔNG NGHỆ ĐIỆN TỬ - VIỄN THÔNG

HÀ NỘI – 2014

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

QUÁCH CÔNG HOÀNG

ỨNG DỤNG CẢM BIẾN 3D KINECT TRONG NHẬN DIỆN NGÔN NGỮ CỬ CHỈ TIẾNG VIỆT HỖ TRỢ VIỆC GIAO TIẾP VỚI NGƯỜI KHUYẾT TẬT KHIẾM THÍNH

Ngành: Công Nghệ Điện Tử - Viễn Thông

Chuyên ngành: Kỹ thuật Điện tử

Mã số: 60 52 0203

LUẬN VĂN THẠC SĨ CÔNG NGHỆ ĐIỆN TỬ - VIỄN THÔNG

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. TRẦN QUANG VINH

HÀ NỘI – 2014

LỜI CAM ĐOAN

Tôi xin cam đoan nội dung của luận văn ***"Ứng dụng cảm biến 3D Kinect trong nhận diện ngôn ngữ cử chỉ tiếng Việt hỗ trợ việc giao tiếp với người khuyết tật khiếm thính"*** là sản phẩm do tôi thực hiện dưới sự hướng dẫn của PGS.TS. Trần Quang Vinh. Trong toàn bộ nội dung của luận văn, những điều được trình bày hoặc là của cá nhân hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

.

Hà Nội, ngày 17 tháng 10 năm 2014

TÁC GIẢ

Quách Công Hoàng

LỜI CẢM ƠN

Trước tiên tôi xin gửi lời cảm ơn chân thành tới tập thể các các thầy cô giáo trong Khoa Điện tử - Viễn thông, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội đã giúp đỡ tận tình và chu đáo để tôi có môi trường tốt học tập và nghiên cứu.

Đặc biệt, tôi xin bày tỏ lòng biết ơn sâu sắc tới PGS.TS. Trần Quang Vinh và NCS. Phùng Mạnh Dương, người trực tiếp đã hướng dẫn, chỉ bảo tôi tận tình trong suốt quá trình nghiên cứu và hoàn thiện luận văn này.

Một lần nữa tôi xin được gửi lời cảm ơn đến tất cả các thầy cô giáo, bạn bè và gia đình đã giúp đỡ tôi trong thời gian vừa qua. Tôi xin kính chúc các thầy cô giáo, các anh chị và các bạn mạnh khỏe và hạnh phúc.

Hà Nội, ngày 17 tháng 10 năm 2014

TÁC GIẢ

Quách Công Hoàng

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Kí hiệu	Tiếng anh	Tiếng việt
CPU	Cental processing unit	Khối xử lý trung tâm
GPU	Graphic processing unit	Khối xử lý đồ họa
3D	Three-dimensional space	Không gian ba chiều
DOF	Degree of freedom	Bậc tự do
PSO	Particle swarm optimization	Giải thuật tối ưu bầy đàn
RGB-D	Red Green Blue - Depth	Ảnh màu – độ sâu
ROS	Robotics Operating System	Hệ điều hành robot
OpenGL	Open Graphic Library	Thư viện đồ họa mở
OpenCL	Open Computing Language	Ngôn ngữ tính toán mở
OpenCV	Open Computer Vision	Thư viện thị giác máy tính mở
OpenNI	Open Natural Interaction	Thư viện tương tác tự nhiên mở
API	Application programming interface	Giao diện lập trình ứng dụng
SDK	Software development kit	Bộ công cụ phát triển phần mềm
CUDA	Compute Unified Device Architecture	Kiến trúc tính toán trên thiết bị đồng nhất
RAM	Random access memory	Bộ nhớ truy cập ngẫu nhiên
GA	Genetic algorithms	Giải thuật di truyền
ACO	Ant colony optimization	Giải thuật tối ưu đàn kiến
GPGPU	General purpose Graphic processing uint	Khối xử lý đồ họa mục đích chung
ALU	Arithmetic logic unit	Đơn vị logic số học

DANH MỤC HÌNH VẼ VÀ ĐỒ THỊ

MỞ ĐẦU

Khi máy tính ngày càng thu nhỏ kích thước như một chiếc kính hay chiếc đồng hồ đeo tay thì việc sử dụng bàn phím, chuột hay màn hình cảm ứng trở nên không thích hợp. Thay vào đó, những cách thức tương tác người – máy mới cần được thúc đẩy nghiên cứu. Bàn tay, bộ phận hoạt động chính xác và hiệu quả nhất khi con người sử dụng công cụ, được đánh giá nhiều tiềm năng. Và thực tế bài toán nhận dạng tư thế tay đã nhận được nhiều sự quan tâm nghiên cứu và đã có những ứng dụng cụ thể như tương tác robot, nhận diện ngôn ngữ cử chỉ, hay điều khiển thiết bị [1]… Tuy nhiên, các ứng dụng tương tác qua bàn tay thường đòi hỏi độ chính xác cao cùng số bậc tự do lớn khiến các phương pháp truyền thống tỏ ra kém hiệu quả. Thay vào đó, phương pháp mô hình được xem là hướng tiếp cận khả thi hiện nay [1] – [4].

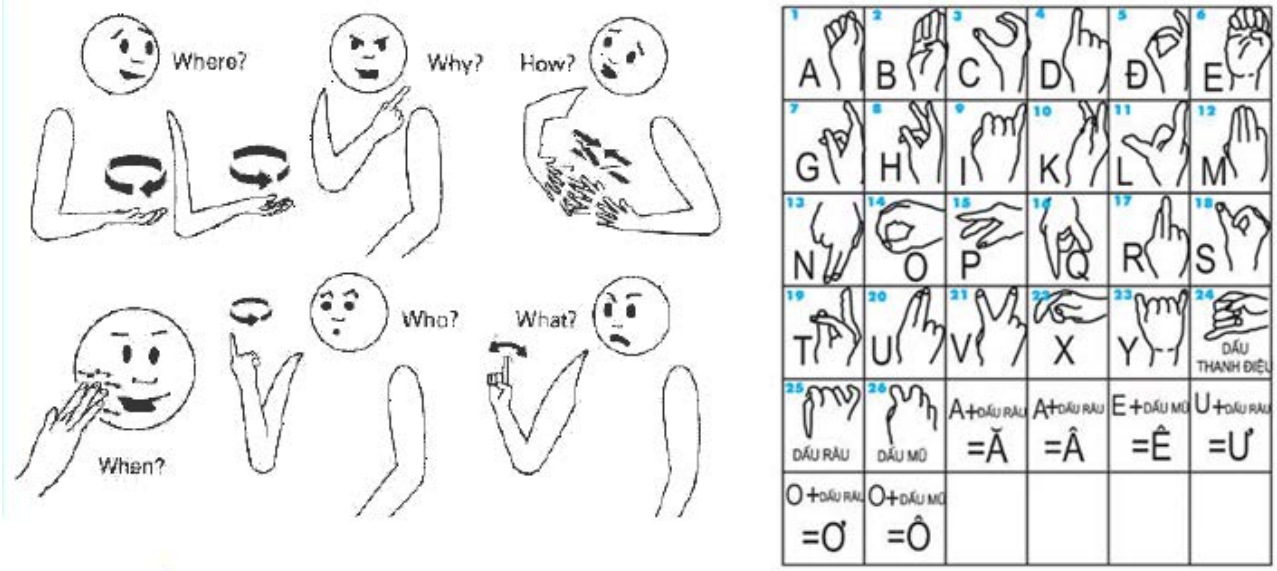
Trong luận văn này, tôi tiếp cận theo hướng mô hình để giải quyết bài toán nhận dạng tư thế bàn tay, hay cụ thể hơn là trạng thái các khớp nối của bàn tay trong các tư thế tay của ngôn ngữ kí hiệu tiếng Việt. Vấn đề nhận dạng được xây dựng như một bài toán tối ưu với mục tiêu là tối thiểu sự sai khác giữa ảnh mô hình của các thế tư thế tay giả định với ảnh quan sát thu được từ cảm biến ảnh Kinect. Giải thuật bầy đàn cải tiến sau đó được sử dụng để giải bài toán tối ưu này. Đồng thời, các tác vụ đòi hỏi tính toán lớn được đưa vào khối xử lý đồ họa GPU của máy tính để tính toán song song. Kết quả thực nghiệm hiện tại cho thấy hệ thống có thể nhận dạng được 26 bậc tự do của bàn tay trong thời gian 0.8s. Kết quả nhận dạng kém nhạy với nhiễu môi trường và yêu cầu phần cứng đơn giản.

Chương 1: TỔNG QUAN

1.1 Mục tiêu và đối tượng nghiên cứu

Mục tiêu của nghiên cứu này là có thể nhận dạng các tư thế của bàn tay người trong không gian ba chiều (3D) trong một khung dữ liệu thu được từ cảm biến Kinect. Địa chỉ ứng dụng của bài toán bước đầu là nhận dạng các kí tự chữ cái trong bảng ngôn ngữ kí hiệu và tôi mong muốn có thể mở rộng sang các ứng dụng tương tác thực tế ảo, thực tế tăng cường và điều khiển thiết bị.

1.1.1 Nhận dạng ngôn ngữ kí hiệu Tiếng Việt



Hình 1: Một số từ ngữ cơ bản trong ngôn ngữ kí hiệu và bảng chữ cái tiếng Việt.

Nhận diện và thông dịch ngôn ngữ kí hiệu nói chung cũng như ngôn ngữ kí hiệu tiếng Việt nói riêng cần kết hợp nhiều công việc như: nhận diện tư thế bàn tay và nhận diện các chuyển động của cánh tay, khẩu hình. Khả năng nhận diện các đặc trưng này càng tốt thì khả năng thông dịch sang ngôn ngữ kí hiệu càng chính xác. Trong đó các vấn đề nhận dạng

kể trên, nhận dạng các tư thế bàn tay hiện nay là bài toán còn có nhiều vấn đề cần phải nghiên cứu và phát triển.

1.1.2 Nhận dạng tư thế bàn tay

Cách làm có độ chính xác cao nhất hiện nay cho bài toán này là sử dụng găng tay chuyên dụng. Thiết bị này được đeo vào tay và có rất nhiều cảm biến để do vị trí bàn tay và độ mở của các khớp ngón tay. Cách làm này có thể xử lý theo thời gian thực và có khả năng phát hiện đầy đủ các hoạt động của bàn tay trong tương tác. Tuy nhiên hạn chế của phương pháp này là găng tay chuyên dụng này rất đắt tiền và đòi hỏi quá trình hiệu chỉnh ban đầu rất phức tạp. Ngoài ra, việc đeo găng tay cản trở các di chuyển tự nhiên của bàn tay, làm giảm phạm vi ứng dụng của phương pháp này.

Nhận diện bàn tay sử dụng các kĩ thuật của xử lý ảnh được xem là một giải pháp thay thế khi không yêu cầu người sử dụng đeo găng tay, tương tác từ bàn tay sẽ thoải mái và tự nhiên hơn. Tuy nhiên việc nhận diện bàn tay bằng kĩ thuật xử lý ảnh gặp phải những vấn đề khó khăn cơ bản như sau:

- ***Bài toán có quá nhiều chiều***: bàn tay là một đối tượng có nhiều khớp nối với hơn 20 bậc tự do. Mặc dù có những ràng buộc giữa các ngón tay và các khớp trên cùng một ngón tay khiến cho chuyển động thực tế ít hơn 20 bậc tự do, tuy nhiên các nghiên cứu cho thấy việc mô tả tư thế bàn tay với ít hơn 6 bậc tự do là không thể. Kết hợp với vị trí và hướng của bàn tay, tương đương 6 bậc tự do, việc nhận dạng tư thế bàn tay đòi hỏi phải ước lượng rất nhiều tham số.
- ***Hiện tượng bị che khuất***: bởi lẽ tay là một đối tượng nhiều khớp nối, nên khi quan sát bàn tay từ cảm biến ảnh, kết quả của phép chiếu từ không gian 3D xuống mặt phẳng 2D khiến một lượng lớn thông tin về hình khối bị che khuất. Điều này gây khó khăn cho việc phân hoạch và trích chọn các đặc trưng cơ bản từ bàn tay.
- ***Tốc độ xử lý***: ngay cả trong một khung hình, một hệ thống xử lý ảnh thời gian thực cần xử lý một lượng lớn dữ liệu. Nói cách khác, độ trễ yêu cầu trong một vài ứng dụng đòi hỏi khả năng tính toán lớn. Với công nghệ phần cứng hiện nay, một số giải thuật đòi hỏi phần cứng chuyên dụng, đắt tiền và có khả năng xử lý song song để thực thi thời gian thực.
- ***Môi trường không kiểm soát***: để mở rộng ứng dụng, nhiều hệ thống tương tác người máy được kì vọng có thể hoạt động trong một môi trường có nền không bị giới hạn và điều kiện ánh sáng thay đổi lớn. Nói cách khác, làm việc trên một nền tùy ý luôn là thách thức với hầu hết các hệ thống xử lý ảnh.
- ***Chuyển động nhanh của bàn tay***: bàn tay có khả năng chuyển động rất nhanh, ở cổ tay tốc độ lên tới 5 mét/giây cho dịch chuyển tịnh tiến và 300 độ / giây cho chuyển động xoay. Hiện nay, cảm bến ảnh thông dụng có thể hỗ trợ tốc độ khung hình 30-60 Hz. Mặt khác, rất nhiều giải thuật khó có thể đạt đến tốc độ 30Hz. Tốc độ di chuyển nhanh của bàn tay kết hợp với tần số lấy mẫu thấp từ cảm biến gây thêm những khó khăn cho thuật

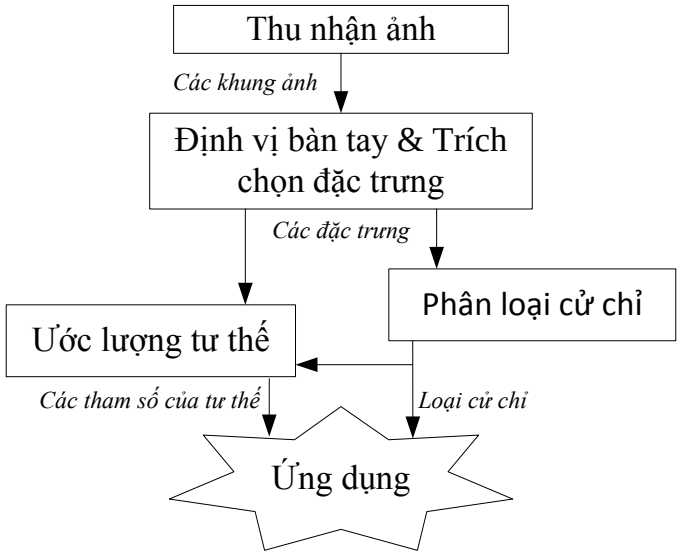
toán nhận dạng. Thí dụ: các ảnh thu được trong các khung hình liên tiếp có độ tương quan thấp dần khi tốc độ di chuyển của bàn tay tăng lên.

Rất khó có thể thỏa mãn tất cả các vấn đề trên, vì vậy một số nghiên cứu đã tìm cách giới hạn với người dùng hoặc môi trường. Thí dụ như coi nền môi trường là không thay đổi và bàn tay là một đối tượng duy nhất có màu da. Tuy nhiên không phải ứng dụng nào cũng chấp nhận những giới hạn này, đặc biệt là trong các ứng dụng tương tác thực tại ảo, tốc độ xử lý cần phải đạt tới 100 Hz.

Hai vấn đề đầu tiên được xem là khó khăn nhất khi phải đối mặt với bài toán có quá nhiều chiều và thông tin bị thiếu hụt do bị che khuất giữa các ngón tay. Một số nghiên cứu muốn giảm thiểu hiện tượng bị che khuất bằng cách quan sát bàn tay bằng hệ thống nhiều camera. Một số nghiên cứu khác tiếp cận theo cách giới hạn số tư thế của bàn tay: thí dụ như giới hạn tư thế trong các kí tự trong bảng chữ cái, hoặc giới hạn các tư thế cầm nắm, kéo thả cơ bản để tăng tốc độ xử lý. Tuy nhiên việc xây dựng một hệ thống tương tác người máy không giới hạn chuyển động của bàn tay cần phải được nghiên cứu.

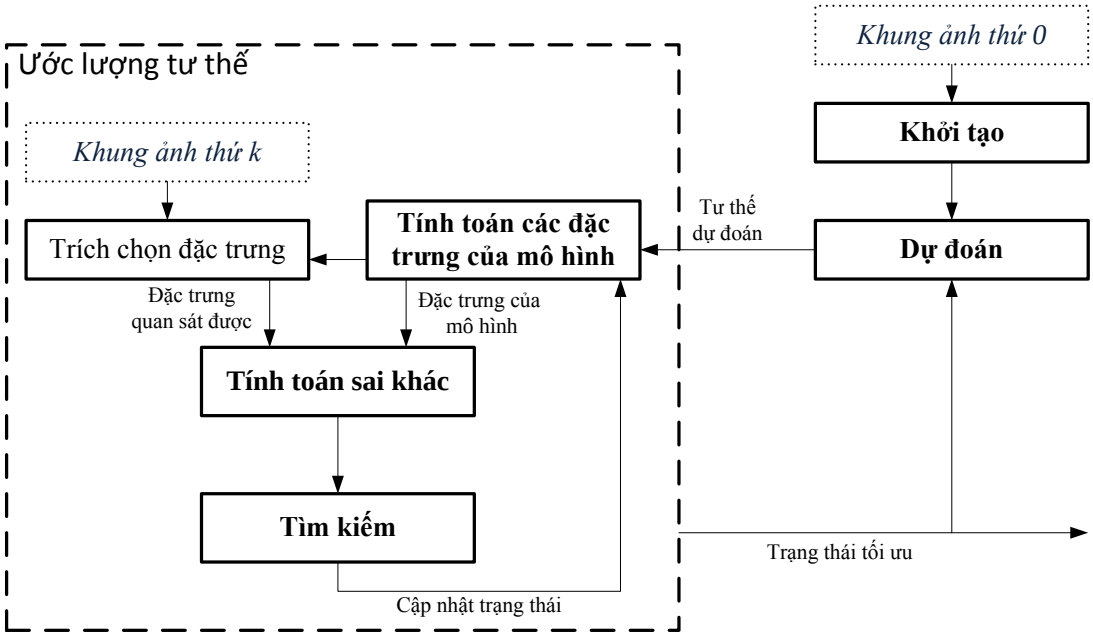
1.2 Các nghiên cứu liên quan

Để nhận diện đầy đủ các bậc tự do của bàn tay có 2 phương pháp chính là phương pháp dựa vào hình dáng quan sát được (*appearance-based method*) và phương pháp dựa trên thông tin mô hình (*model-based method*).



Hình 2: Lưu đồ tổng quát của phương pháp nhận dạng theo hình dáng.

Phương pháp nhận dạng dựa vào hình dáng quan sát bao gồm việc phân hoạch và trích chọn đặc trưng, sau đó phân loại thành các tư thế tay. Ưu điểm của phương pháp này là không tốn nhiều tài nguyên tính toán và phần cứng phức tạp.Tuy nhiên số lượng tư thế tay nhận dạng được chỉ là một tập hữu hạn các tư thế đã được xây dựng trước để đưa vào để huấn luyện.



Hình 3: Lưu đồ tổng quát của phương pháp nhận dạng mô hình

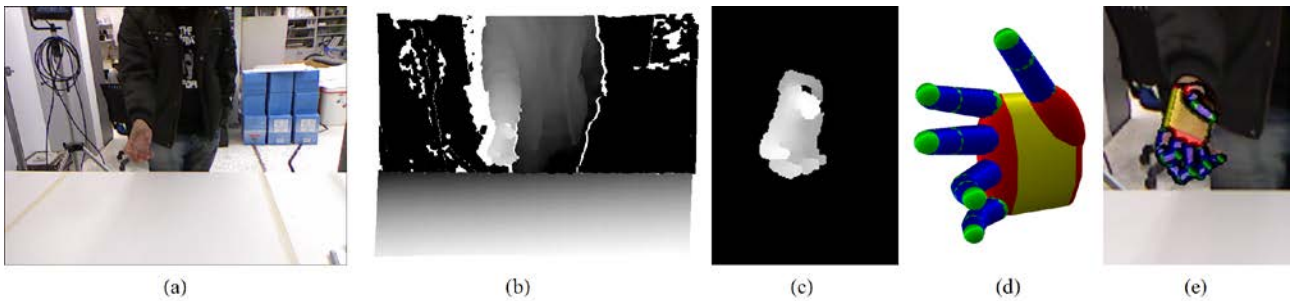
Phương pháp mô hình nhận dạng dựa trên so sánh ảnh quan sát với ảnh mô hình 3 chiều. Ảnh quan sát là hình ảnh thu được từ hệ một hoặc nhiều camera và có thể kèm thông tin độ sâu. Phương pháp này đòi hỏi tài nguyên tính toán lớn và độ chính xác phụ thuộc nhiều vào lượng thông tin mô hình quan sát được. Trong khi đó, ảnh mô hình được xây dựng dựa trên cấu trúc giải phẫu học cùng các ma trận đồ họa. Tùy mục đích ứng dụng và giải thuật, ảnh mô hình có thể khác nhau giữa các nhóm nghiên cứu.

Trong [3], mô hình bàn tay được xây gồm 12 bậc tự do với 10 bậc dành cho các ngón tay và 2 bậc dành cho chuyển động tịnh tiến trong không gian. Để nhận dạng tư thế tay, có hai phép đo được sử dụng. Phép đo thứ nhất đo mức độ chồng chập về diện tích giữa ảnh quan sát và ảnh mô hình chiếu trên mặt phẳng quan sát. Phép đo thứ hai đánh giá sự sai khác về khoảng cách giữa các đường biên của hai ảnh. Kĩ thuật tối ưu xuống dốc đơn hình (downhill simplex) sau đó được sử dụng để tìm tư thế cho sai khác nhỏ nhất. Các ràng buộc cơ sinh học cũng được sử dụng nhằm thu hẹp không gian tìm kiếm và loại bỏ các trường hợp không thực. Kết quả thực nghiệm cho thấy giải thuật đã nhận dạng được chuyển động đơn giản của bàn tay trong điều kiện nền đồng màu.

Hình 4: Kết quả nhận dạng trong nghiên cứu của Stenger.

Trong một nghiên cứu khác [4], Stenger đề xuất mô hình bàn tay với 27 bậc tự do được biểu diễn bởi 39 mặt bậc hai cụt. Việc sử dụng mặt bậc hai giúp đơn giản quá trình khởi tạo mô hình 3 chiều đồng thời dễ dàng thực hiện các phép chiếu. Bộ lọc Kalman sau đó được sử dụng để ước lượng và tối thiểu sai số hình học giữa các đường biên của ảnh quan sát và ảnh mô hình. Kết quả cho thấy giải thuật có thể nhận dạng được 7 bậc tự do với tốc độ 3 hình/giây. Để nâng cao độ chính xác, Stenger sau đó đã đề xuất sử dụng tập hợp mẫu gồm 16.055 tư thế bàn tay kết hợp với bộ lọc Bayes phân cấp [5]. Các hàm so sánh tương quan cũng được cải tiến để có thể làm việc được điều kiện nhiễu môi trường lớn. Giải thuật thành công với tỉ lệ nhận dạng hơn 90% và độ chính xác 9.3 điểm ảnh cho ảnh 320x240. Tuy nhiên, quá trình cài đặt thuật toán tương đối phức tạp với nhiều bước căn chỉnh thủ công đồng thời yêu cầu phải có tập dữ liệu quan sát lớn.

Gần đây, Oikonomidis đã đề xuất mô hình bàn tay gồm 26 bậc tự do được xây dựng từ các hình đồ họa cơ bản là hình cầu, hình trụ và hình elipsoid [2]. Ảnh quan sát được sử dụng bao gồm ảnh màu và ảnh độ sâu thu thập bởi cảm biến ảnh Kinect. Giải thuật tối ưu bầy đàn sau đó được áp dụng để tìm nghiệm cho bài toán cực tiểu sự sai khác giữa ảnh quan sát và ảnh mô hình. Kết quả cho thấy giải thuật đã nhận diện được đầy đủ 26 bậc tự do của bàn tay với tốc độ 15 hình/giây.



Hình 5: Kết quả nhận dạng trên cảm biến Kinect của Oikonomidis [2].

1.3 Nội dung nghiên cứu

Trong luận văn này, tôi tiếp cận theo hướng mô hình với ý tưởng chính phát triển từ nghiên cứu của Oikonomidis để giải quyết bài toán nhận dạng tư thế bàn tay dựa trên những thiết bị phần cứng sẵn có. Nội dung chính của các chương được trình bày như sau:

Chương 2. Phân tích các đặc trưng cấu trúc động học của bàn tay và các đặc trưng của cảm biến RGB-D như Kinect. Từ những đánh giá này, nghiên cứu xây dựng mô hình 3D của bàn tay bằng đồ họa máy tính và mô phỏng các dữ liệu thu được từ cảm biến Kinect với góc nhìn và khoảng cách khác nhau trong không gian.

Chương 3. Xây dựng giải thuật nhận dạng theo mô hình bao gồm hai công công việc chính: xây dựng hàm đánh giá mô hình với quan sát; xây dựng phương pháp nhận dạng sử dụng giải thuật tối ưu bầy đàn PSO để thay đổi các tham số của khớp ngón tay và tối thiểu hóa sai khác.

Chương 4. Trình bày về các giải pháp xử lý song song trên CPU/GPU. Ứng dụng nền tảng tính toán song song OpenCL để tăng cường tốc độ xử lý của thuật toán trên từng công đoạn cụ thể.

Chương 5. Mô phỏng hệ thống nhận dạng và thực nghiệm trên dữ liệu thật thu được từ cảm biến Kinect. Đánh giá khả năng nhận dạng, độ chính xác cũng như thời tài nguyên tính toán của giải thuật.

Chương 6. Kết luận và đánh giá ưu nhược điểm của hệ thống. Bàn luận và đề xuất các phương hướng phát triển nghiên cứu này trong tương lai.

Chương 2: MÔ HÌNH BÀN TAY

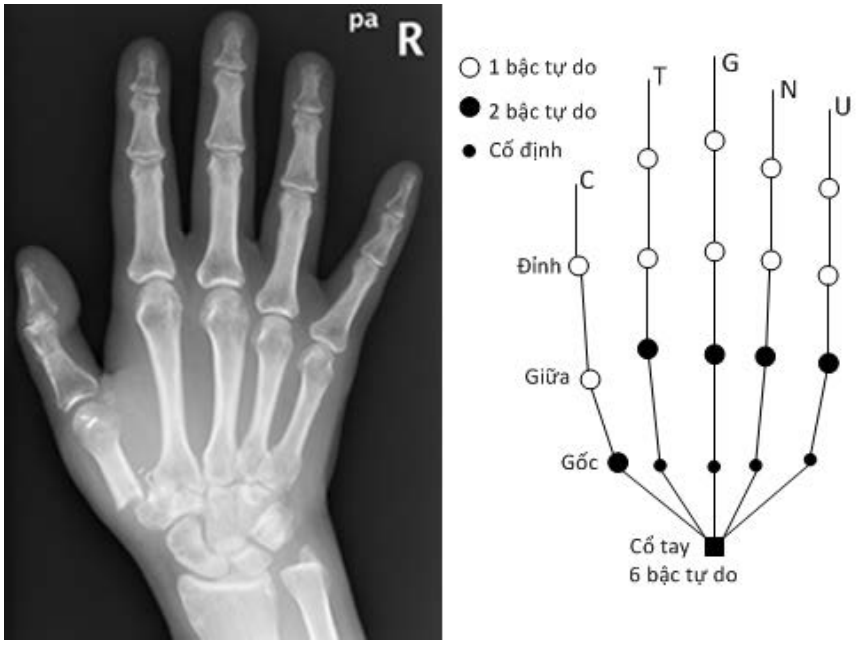
Phương pháp nhận dạng theo mô hình có mục tiêu chính là khôi phục lại mô hình 3D của tư thế bàn tay. Mô hình này được dùng để mô tả khái quát những đặc điểm của bề mặt bàn tay và các tư thế tay hợp lý.

Phần đầu của chương này trình bày cách làm thế nào để xây dựng một mô hình 3D bàn tay dựa trên các khối hình cở bản, đánh giá các đặc điểm động học của bàn tay trong thực tế. Phần tiếp theo của chương trình bày về các đặc trưng của của cảm biến Kinect, các vấn đề lý thuyết khi ánh xạ mô hình 3D xuống mặt phẳng 2D. Mục đích nghiên cứu của vần vấn đề này để xây dựng những mô hình giả định tương ứng với góc nhìn của cảm biến. Phần cuối của chương trình bày giải thuật xác định mô hình quan sát từ cảm biến bao gồm thông tin màu sắc và độ sâu.

Mô hình quan sát và mô hình giả định được xây dựng trong chương này phục vụ cho giải thuật nhận dạng sẽ được đề cập trong chương 3.

2.1 Mô hình động học của bàn tay

Bàn tay con người bao gồm 27 xương, trong đó có 8 xương ở cổ tay và 19 xương cho lòng bàn tay và ngón tay. Các xương này được kết nối với nhau bởi các khớp nối có một hoặc nhiều bậc tự do. Hình 6 biểu diễn các khớp nối cùng số bậc tự do tương ứng tạo thành tổng cộng 26 bậc tự do [1]. Trong đó, cổ tay có 6 bậc tự do với 3 bậc tự do cho chuyển động tịnh tiến trong không gian và 3 bậc tự do cho chuyển động xoay quanh các trục. Năm ngón tay mỗi ngón có 4 bậc tự do với 2 bậc cho khớp gốc ngón tay (gập/ngửa và khép/mở) và 1 bậc cho mỗi khớp còn lại.



Hình 6: Cấu trúc xương và mô hình động học của bàn tay

Với cách biểu diễn như vậy, động học của mỗi ngón tay được xác định bởi một vector gồm 4 tham số góc:

$$q_i = (\theta_{MP}^x, \theta_{MP}^z, \theta_{PIP}, \theta_{DIP}) \quad (2.1)$$

trong đó θ_{MP}^x và θ_{MP}^z là hai góc quay của khớp gốc, θ_{PIP} là góc quay của khớp giữa và θ_{DIP} là góc quay của khớp đỉnh.

Tương tự, vị trí và hướng của bàn tay được xác định qua cổ tay bởi vectơ gồm 6 tham số:

$$q_c = (x_c, y_c, z_c, \theta_c^x, \theta_c^y, \theta_c^z) \quad (2.2)$$

trong đó (x_c, y_c, z_c) là tọa độ của cổ tay trong không gian và $(\theta_c^x, \theta_c^y, \theta_c^z)$ là hướng của bàn tay quay quanh các trục tương ứng. Như vậy, tư thế của bàn tay hoàn toàn xác định khi biết 26 tham số góc:

$$h = (q_i, q_c),\ i = 1, 2, ..., 5 \quad (2.3)$$

Do đặc điểm giải phẫu học, chuyển động của các khớp ngón tay bị ràng buộc bởi các cơ giằng dẫn tới các góc quay của cổ tay và các đốt ngón tay bị giới hạn. Đặc điểm này là quan trọng bởi nó giúp giới hạn đáng kể không gian tìm kiếm của giải thuật bầy đàn sau này. Bảng 1 trình bày giới hạn của các tham số góc của ngón tay. Bảng 2 trình bày giới hạn các tham số của cổ tay. Lưu ý rằng giới hạn của vị trí (x_c, y_c, z_c) được xác định bởi thị trường của camera.

BẢNG 1: GIỚI HẠN CÁC THAM SỐ GÓC CỦA NGÓN TAY

	θ_{MP}^x	θ_{MP}^z	θ_{PIP}	θ_{DIP}
Ngón cái	$0^0 – 90^0$	$-15^0 – 60^0$	$0^0 – 50^0$	$-15^0 – 70^0$
Ngón trỏ	$0^0 – 90^0$	$-15^0 – 15^0$	$0^0 – 100^0$	$0^0 – 60^0$
Ngón giữa	$0^0 – 90^0$	$-10^0 – 10^0$	$0^0 – 100^0$	$0^0 – 60^0$
Ngón đeo nhẫn	$0^0 – 90^0$	$-30^0 – 0^0$	$0^0 – 100^0$	$0^0 – 60^0$
Ngón út	$0^0 – 90^0$	$-45^0 – 0^0$	$0^0 – 100^0$	$0^0 – 60^0$

BẢNG 2: GIỚI HẠN CÁC THAM SỐ GÓC VÀ VỊ TRÍ CỦA CỔ TAY

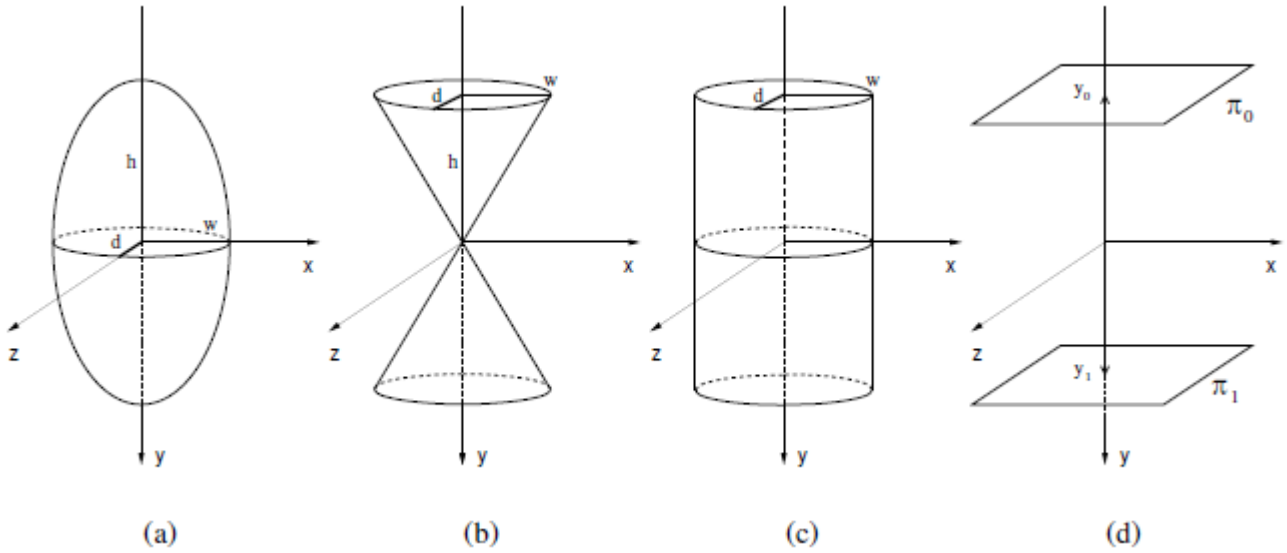
x_c	y_c	z_c	θ_c^x	θ_c^y	θ_c^z
-0,9 *m* – 0,9 *m*	-0,68 *m* – 0,68 *m*	0,5 *m* – 1,5 *m*	$-30^0 – 120^0$	$-70^0 – 75^0$	$-35^0 – 20^0$

2.2 Xây dựng mô hình giả định bằng đồ họa máy tính

Trên thực tế, mô hình bàn tay vừa có các thành phần khớp nối, lại vừa có các thành phần đàn hồi. Tuy nhiên, để giảm bớt khối lượng tính toán, các mô hình quá phức tạp của bàn tay không được sử dụng. Trong nhiều nghiên cứu, mô hình 3D bàn tay cần phải ánh xạ

lên ảnh 2D của ảnh quan sát để tìm ra các đặc trưng. Bên cạnh đó cần tính đến yếu tố hiển thị khi một phần bàn tay bị che khuất. Chính vì những lý do như trên nên mô hình 3D của bàn tay nên được xây dựng từ những khối hình thô, bao gồm những khối hình cơ bản như hình cầu, hình trụ, hình nón cụt và hình ellipsoid.

2.2.1 Các khối hình học cơ bản



Hình 7: Thí dụ về các mặt bậc hai: (a) ellipsoid, (b) hình nón, (c) hình trụ, (d) mặt phẳng

Các khối hình cơ bản nêu trên về mặt toán học được coi là các mặt bậc hai được biểu diễn trên tọa độ đồng nhất bằng các ma trận **Q** có kích thước 4x4. Bề mặt được định nghĩa bởi một tập các điểm **X** thỏa mãn phương trình:

$$\mathbf{X^T Q T = 0} \tag{2.4}$$

với **X = [x,y,z,1]^T** là một vector đồng nhất biểu diễn một điểm trong không gian 3D. Phương trình trên là cách viết đơn giản của phương trình:

$$q_{1,1}\,x^2+q_{2,2}\,y^2+q_{3,3}\,z^2+2q_{1,2}\,xy+2q_{1,3}\,xz+2q_{2,3}\,yz+2q_{1,4}\,x+2q_{2,4}\,y+2q_{3,4}\,z+\,q_{4,4}=0 \tag{2.5}$$

Các họ của mặt bậc hai thu được từ ma trận **Q** với các hạng khác nhau, cụ thể là:

- **Ellipsoid:** được biểu diễn bởi ma trận Q với hạng cao nhất. Vector pháp tuyến của được xuất pháp từ tâm và tập điểm trên bề mặt thỏa mãn phương trình:

$$\frac{x^2}{w^2}+\frac{y^2}{h^2}+\frac{z^2}{d^2}=1 \tag{2.6}$$

  phương trình này tương đương với ma trận:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{w^2} & 0 & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 & 0 \\ 0 & 0 & \frac{1}{d^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad (2.7)$$

Những thí dụ khác của bề mặt có thể biểu diễn với ma trận kiểu này là paraboloids và hyperboloid. Nếu ma trận Q là ma trận đơn nhất, mặt bậc hay này được xem như là suy biến.

- **Hình nón và hình trụ**: được biểu diễn bởi ma trận Q với hạng bằng 3.
  Phương trình của hình nón với trục chính 0y có dạng

$$\frac{x^2}{w^2} - \frac{y^2}{h^2} + \frac{z^2}{d^2} = 0 \quad (2.8)$$

và ma trận tương đương là:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{w^2} & 0 & 0 & 0 \\ 0 & -\frac{1}{h^2} & 0 & 0 \\ 0 & 0 & \frac{1}{d^2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.9)$$

Phương trình hình trụ có trục chính Oy là:

$$\frac{x^2}{w^2} + \frac{z^2}{d^2} = 1 \quad (2.10)$$

và ma trận tương ứng:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{w^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad (2.11)$$

- **Cặp mặt phẳng π_o và π_1**: được biểu diễn bởi $\mathbf{Q} = \pi_o \pi_1^T + \pi_1 \pi_o^T$ có hạng ma trận bằng 2, một cặp mặt phẳng song song với mặt xz được biểu diễn bởi phương trình:

$$(y - y_0)\ (y - y_1) = 0 \quad (2.12)$$

trong trường hợp này $\pi_o = [0,1,0,-y_0]^T$ và $\pi_1 = [0,1,0,-y_1]^T$.

Ma trận tương đương sẽ là:

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{(y_0+y_1)}{2} \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{(y_0+y_1)}{2} & 0 & y_0\, y_1 \end{bmatrix} \tag{2.13}$$

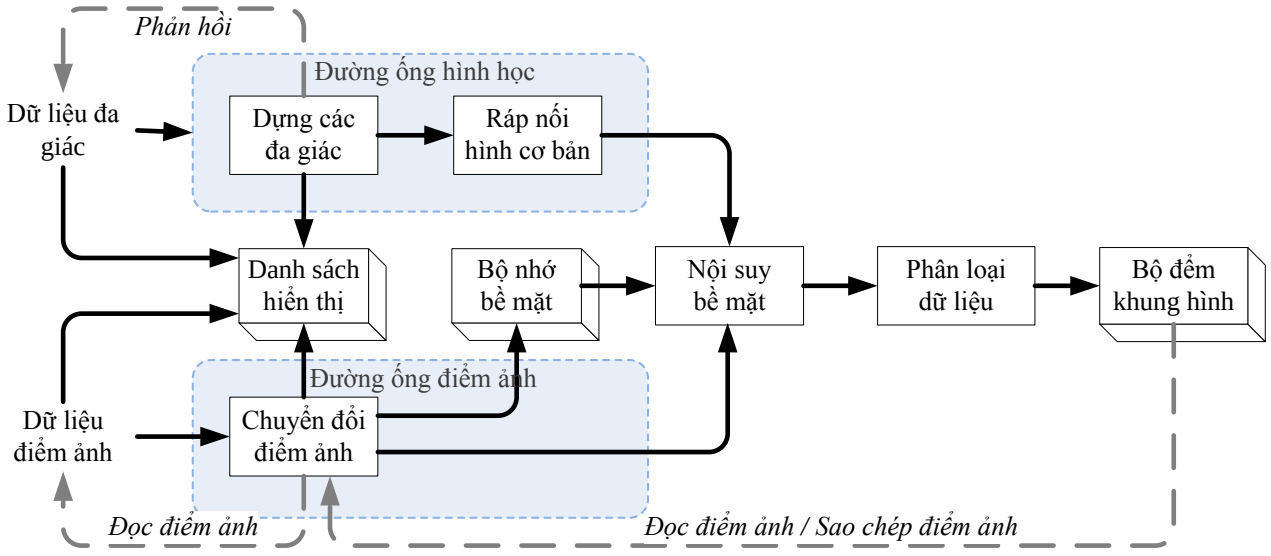
Chuyển đổi tọa độ trong không gian 3 chiều: khi chuyển đổi tọa độ trong không gian Euclid, hình dạng của các khối hình này vẫn giữ nguyên nhưng ma trân biểu diễn bị thay đổi. Sự biến đổi này được đại diện trong hệ tọa độ đồng nhất bởi một ma trận chuyển đổi **T** có kích thước 4x4:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \tag{2.14}$$

trong đó **R** là ma trận 3x3 đại diện cho phép xoay của 3 trục và **t** biểu diễn sự dịch chuyển theo 3 chiều xyz.

2.2.2 Phương pháp xây dựng mô hình trên các thư viện phần mềm đồ họa

Hiện nay có hai thư viện đồ họa chính thường được sử dụng là DirectX và OpenGL. Hai thư viện này cung cấp cho chúng ta những giao diện lập trình (API) cho khối xử lý đồ họa (GPU). Giao diện lập trình của chúng được thiết kế theo quy chuẩn để có thể chạy được trên nhiều nền tảng phần cứng khác nhau. Các thức hoạt động của DirectX / OpenGL trong máy tính giống mô hình client-server. Client là phần trên máy tính mà các tập lệnh trên thư viện thực thi gửi yêu cầu xuống server. Server là phần trên máy tính – thường là GPU – thực hiện các hàm vẽ được yêu cầu từ client và trả về các kết quả. Trong nghiên cứu này tôi sử dụng thư viện OpenGL do khả năng tương thích với nhiều hệ điều hành như Windows, Linux hay Android.

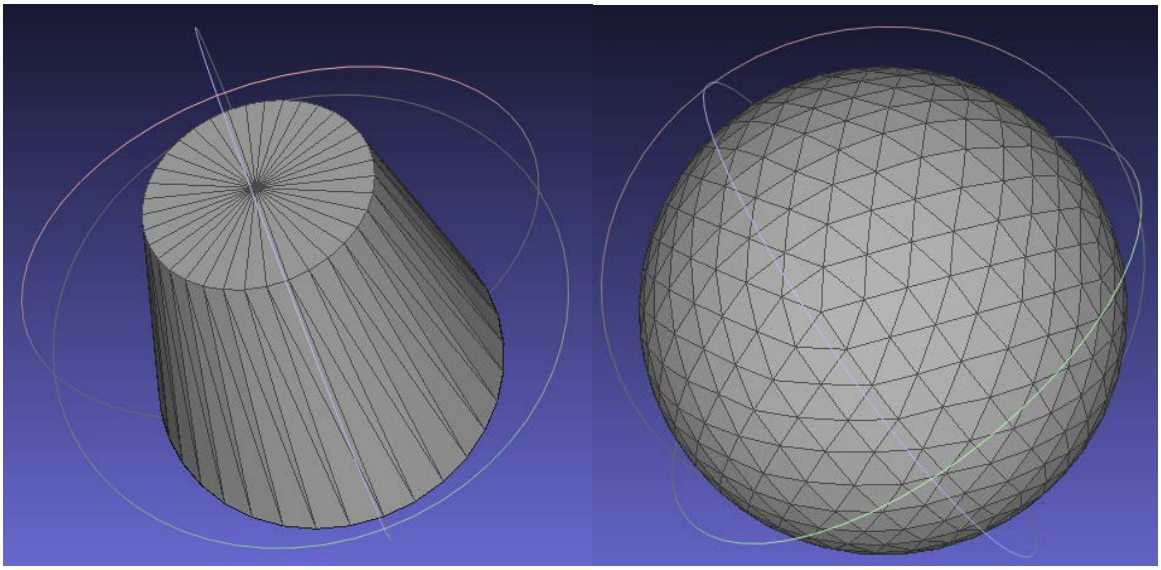


Hình 8: Đường ống lệnh của OpenGL

Đường ống lệnh trong OpenGL là một chuỗi các trạng thái xử lý theo thứ tự. Đường dữ liệu Image-Path được dùng để xử lý dữ liệu điểm ảnh, đường dữ liệu Geometry Path dùng để xử lý dữ liệu hình học.

Để xây dựng được các mặt bậc hai trên OpenGL với yêu cầu về tính thời gian thực, cần phải biểu diễn rời rạc hóa các mặt bậc hai này dưới dạng đa giác. Một đa giác phức tạp lại có thể được biểu diễn bằng nhiều các tam giác và tứ giác, vì vậy từ phương trình đặc trưng của các mặt bậc hai, có thể tính được đỉnh của đa giác tương ứng với các tam giác và tứ giác này.

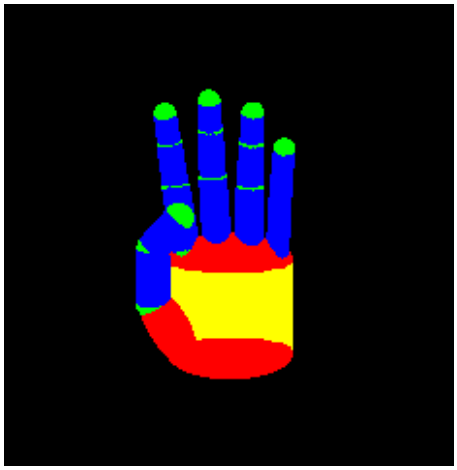
GPU được thiết kế tối ưu để nội suy các đỉnh tam giác thành các mặt, vì vậy từ các đỉnh tam giác hay tứ giác tính được ở trên chúng ta có thể xấp xỉ các mặt bậc hai. Sai số của việc xấp xỉ này phụ thuộc vào số đa giác để biểu diễn các mặt bậc hai. Do tính thời gian thực của thuật toán nên trong nghiên cứu này, một mặt cầu hay ellipsoid tùy kích thước được biểu diễn bởi 150 đến 200 đỉnh, còn các mặt nón cụt và hình trụ được biểu diễn bằng 48 đỉnh.



Hình 9: Biểu diễn các mặt nón cụt và mặt cầu bằng OpenGL

Từ cấu trúc giải phẫu học và động học đã nêu trong phần trước, tôi biểu diễn ảnh mô hình của bàn tay gồm 2 phần: lòng bàn tay và năm ngón tay. Lòng bàn tay được biểu diễn bởi một hình trụ elip bao hai đầu là 2 khối ellipsoid (hình 10).

Mỗi ngón tay được biểu diễn bởi 3 hình nón cụt tương ứng với các đốt ngón tay và 4 hình cầu tương ứng với các khớp ngón tay và đầu ngón tay. Riêng ngón cái có cấu tạo hơi khác nên đốt ngón tay lớn nhất được biểu diễn bởi một khối ellipsoid thay vì hình nón cụt. Kích thước và tỉ lệ giữa các phần của bàn tay được xác định dựa trên đo đạc bàn tay thực.



Hình 10: Ảnh mô hình bàn tay tạo bởi các khối hình học cơ bản

Ảnh mô hình cho phép biểu diễn hình ảnh 3 chiều của bàn tay trong không gian dưới các góc nhìn khác nhau của cảm biến ảnh. Khi đó ảnh 2D được tạo thành bởi một phép chiếu hình học lên mặt phẳng ảnh quan sát được biểu diễn bởi ma trận sau:

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (2.15)$$

Ý nghĩa của các tham số của ma trận này:

n – khoảng cách từ camera tới mặt cắt gần (near clipping plane).

f – khoảng cách từ camera tới mặt cắt xa (far clipping plane).

t – khoảng cách từ trục chính tới biên trên.

b – khoảng cách từ trục chính tới biên dưới.

r – khoảng cách từ trục chính tới biên phải.

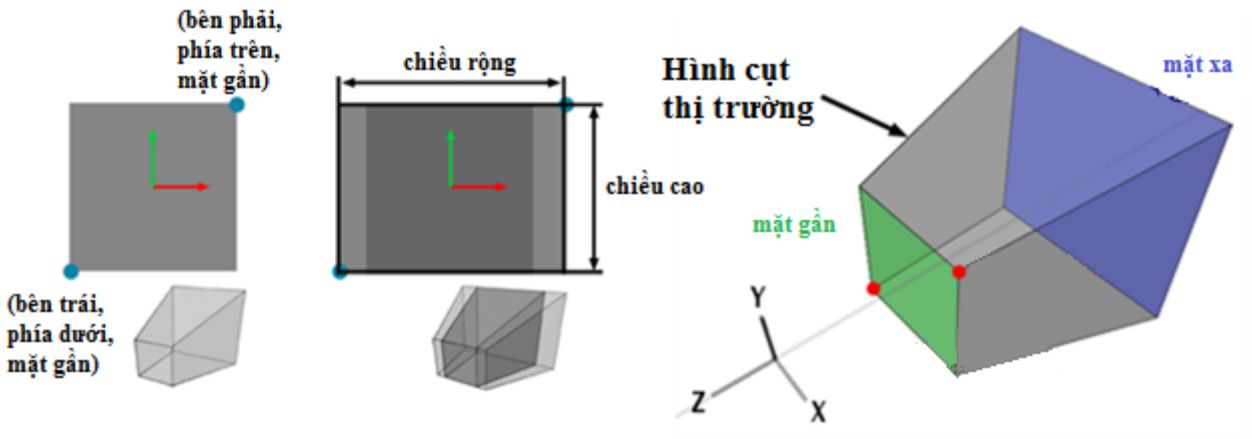
l – khoảng cách từ trục chính tới biên trái.

Dựa vào các thông số của cảm biến Kinect, để mô phỏng phép chiếu từ cảm biến màu RGB của Kinect:

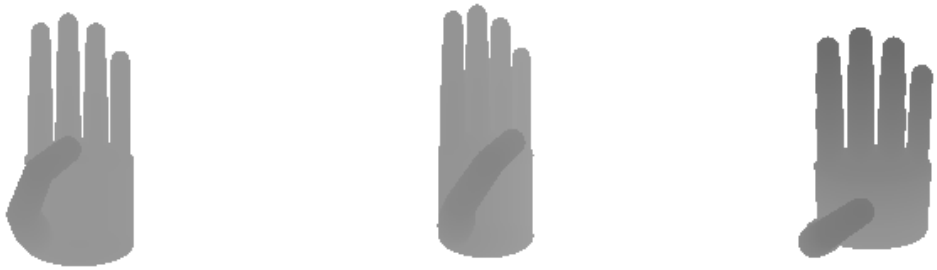
n = 531.15

f = 531.15 + d (d tùy thuộc vào không gian mô phỏng)

l = -320; r = 320; b = -240; t = 240



Hình 11: Mô tả về phép chiếu 3D xuống 2D trong không gian



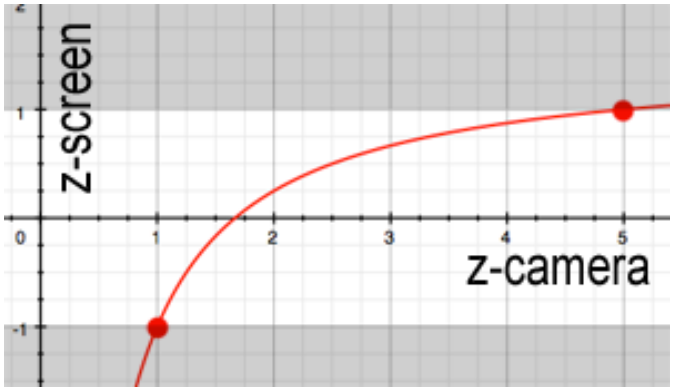
Hình 12: Bề mặt độ sâu của bàn tay dưới các góc nhìn khác nhau (màu càng đậm khoảng cách càng gần)

Có một điểm cần lưu ý là độ sâu trong mô phỏng bằng OpenGL (z-screen) không tuyến tính tính theo khoảng cách thực mà được chuẩn hóa từ -1 tới 1 phụ thuộc vào công thức sau:

$$Z = \frac{\frac{f}{n}}{\frac{range}{depth - offset} + \frac{f}{n} - 1} \tag{2.16}$$

Trong đó:

- depth là độ sâu thực của điểm ảnh cần tính
- offset là độ sâu thực của mặt cắt gần
- range là khoảng cách từ mặt cắt gần tới mặt cắt xa



Hình 13: Biến đổi của z-screen trong không gian mô phỏng n=1 và f=5

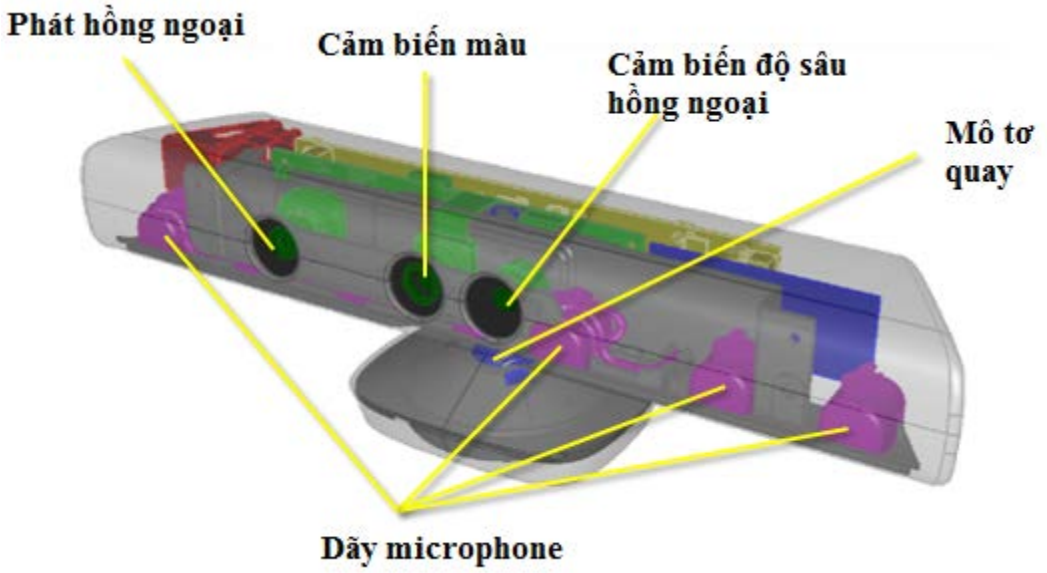
2.3 Xác định mô hình quan sát bàn tay trên cảm biến

2.3.1 Tóm lược về cảm biến Kinect

Cảm biến Kinect được Microsoft nghiên cứu và phát triển với mục đích tương tác người sử dụng với máy tính trong môi trường trong nhà từ khoảng cách 0.5m đến 3.5m. Cảm biến có khả năng thu thập dữ liệu màu RGB và độ sâu với độ phân giải điểm ảnh là 640x480 và tốc độ khung hình lên tới 30Hz.

Cấu trúc phần cứng của Kinect bao gồm:

- **Bộ cảm biến độ sâu** dựa trên nguyên lý stereo sử dụng một cặp thu phát hồng ngoại. Thị trường của cảm biến đô sâu là 58.5 độ theo chiều ngang và 45.6 độ theo chiều dọc
- **Cảm biến ảnh màu RGB:** có nhiều chế độ phân giải từ 320x240, 640x480 đến 1280x960 (tại độ phân giải cao nhất tốc độ khung hình chỉ đạt 12Hz). Thị trường của cảm biến ảnh màu là 62 độ theo chiều ngang và 48.6 độ theo chiều dọc.
- **Motor điều chỉnh góc nghiêng**: phạm vi từ -27 độ đến +27 độ
- **Bộ cảm biến âm thanh**: bao gồm 4 microphone 24bit ADC có khả năng lọc nhiễu tiếng vọng và ồn từ môi trường xung quanh.
- **Cảm biến gia tốc 3 chiều**: có độ chính xác trên 1 độ.



Hình 14: Sơ đồ phần cứng của cảm biến Kinect

Ưu điểm lớn nhất của công nghệ cảm biến Kinect là có thể đo được ảnh độ sâu với độ phân giải rất lớn trong khi đó giá thành lại rất rẻ: từ 150 đến 250 USD. Trong khi đó, với công nghệ laser giá thành thiết bị có thể lên tới hàng nghìn USD.

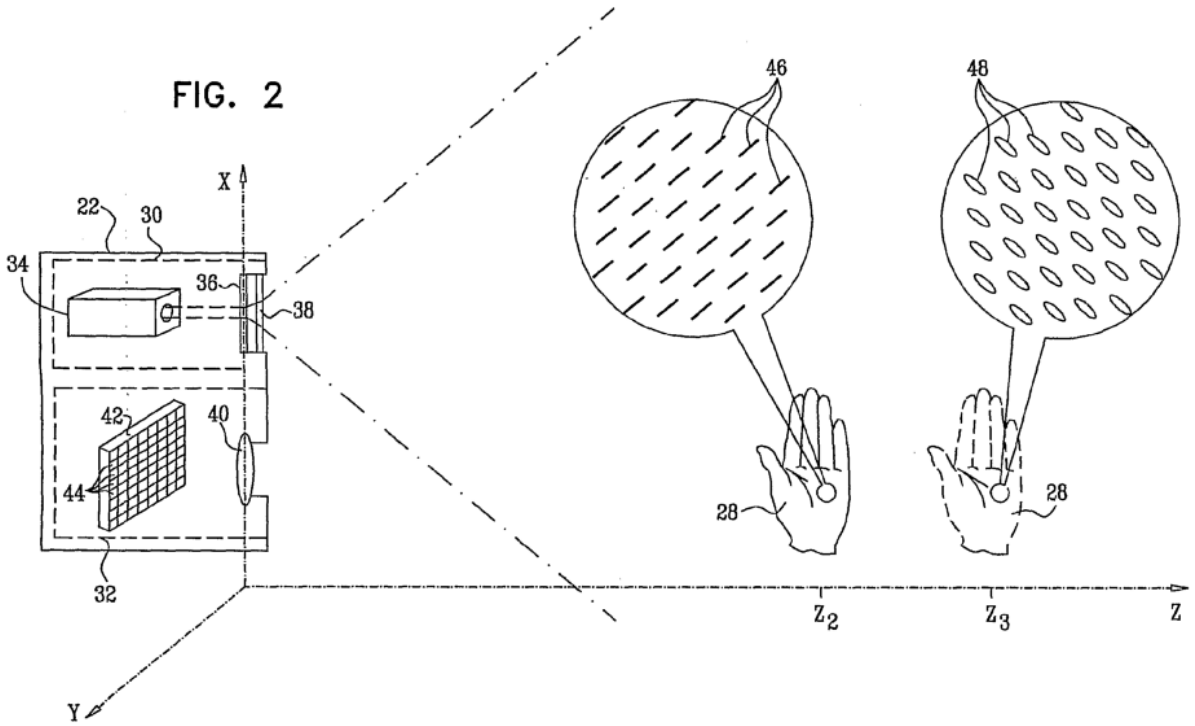
Theo các nghiên cứu John MacCormic, cảm biến Kinect có thể thu thập được thông tin độ sâu dựa vào một số nguyên lý chính sau:

- **Chùm sáng cấu trúc (structured-lights):** nếu chúng ta biết được cấu trúc một chùm sáng khi chiếu lên trên mặt phẳng, thì khi chùm sáng này chiếu lên một bề mặt lồi lõm nào, có thể phân tích và đánh giá được bề mặt này thông qua sự biến dạng của chùm sáng.

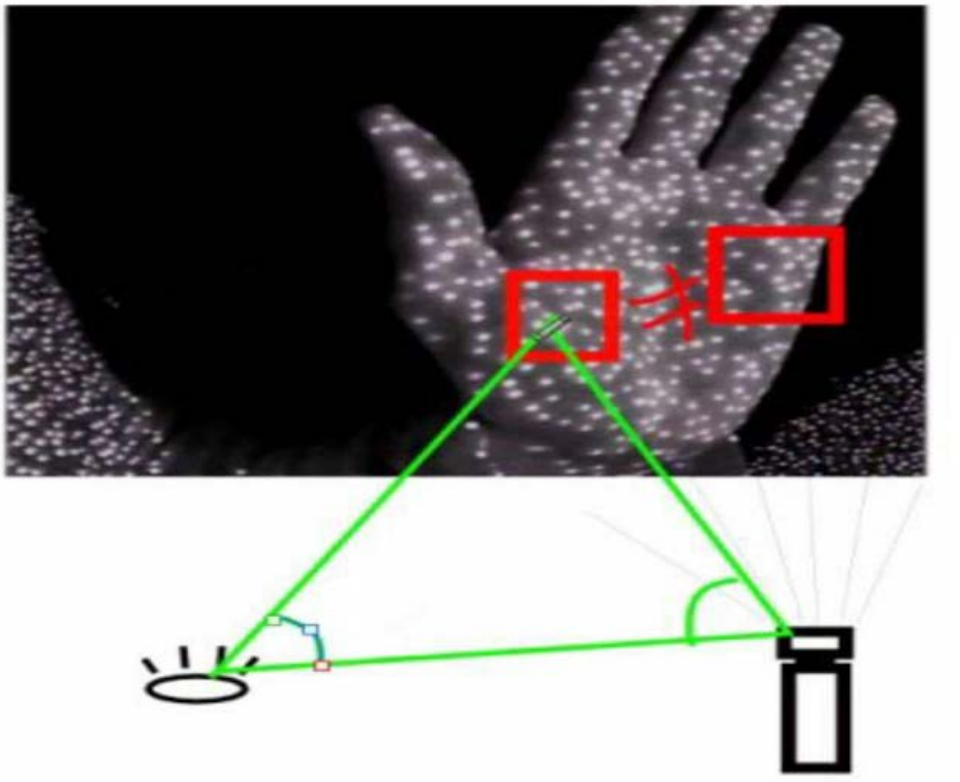
Hình 15: Tái tạo lại bề mặt sử dụng nguyên lý của ánh sáng cấu trúc

- **Độ sâu từ tiêu cự:** nguyên lý này xác định độ sâu của ảnh dựa vào đặc trưng của ảnh là càng xa điểm tiêu cự thì ảnh càng mờ.
- **Độ sâu từ ảnh stereo:** xác định độ sâu dựa trên chênh lệch giữa 2 ảnh của cùng một phối cảnh. Đối tượng có khoảng cách càng gần thì thay đổi càng lớn và ngược lại, đối tượng ở khoảng cách càng xa thì càng ít thay đổi.

Cảm biến Kinect đã cải thiện độ chính xác của phương pháp tính độ sâu bằng tiêu cự lên rất nhiều. Trong thiết bị này sử dụng một thấu kính méo có độ dài trục x và y không bằng nhau. Tia sáng có dạng tròn đi qua thấu kính này sẽ bị biến dạng thành hình elip với độ xoay của hai bán trục phụ thuộc vào khoảng cách từ thấu kính tới điểm tới của tia sáng.



Hình 16: Thiết kế của thấu kính Kinect



Hình 17: Xác định độ sâu bằng nguyên lý stereo

Việc xác định độ sâu được thực hiện trên bộ thu phát chùm tia hồng ngoại. Khoảng cách giữa bộ thu phát là 7.5cm. Việc ước lượng biến dạng theo nguyên lý stereo được thực hiện trên một cửa sổ có kích thước 9x9 pixel hoặc 9x7 pixel.

2.3.2 Xác định mô hình bàn tay từ cảm biến Kinect

Để có được mô hình bàn tay quan sát được từ cảm biến, hiểu một cách đơn giản là tách vùng độ sâu tương ứng với bàn tay là đủ vì độ phân giải độ sâu của Kinect lên tới 1-2 mm. Tuy nhiên trên thực tế Kinect là cảm biến được thiết kế để phù hợp với những đối tượng có kích thước lớn trong môi trường trong nhà như bàn ghế, con người. Vì vậy mô hình bàn tay quan sát được từ ảnh độ sâu của cảm biến Kinect là không đủ do mất mát thông tin từ nhiễu, giải thuật của Kinect không đủ thông tin để nội suy được độ sâu.

Các khu vực mất mát thông tin tường nằm ở các vùng biên, đầu ngón tay, ở tư thế hướng vuông góc với mặt phẳng ảnh. Trong trường hợp này diện tích tiếp xúc với các tia sáng cấu trúc là quá nhỏ, thông tin trong cửa số ước lượng theo stereo không đủ để đánh giá độ sâu. Trong trường hợp này ảnh độ sâu thường trả về giá trị là 0. Để giải quyết vấn đề này, cần thiết phải có sự kết hợp ảnh màu và ảnh độ sâu để có được thông tin đầy đủ về mô hình bàn tay.

Việc trích chọn mô hình quan sát của bàn tay được thực hiện qua ba bước chính:

- Biến đổi ảnh màu và ảnh độ sâu về cùng một hệ trục tọa độ.
- Xác định vùng bàn tay trong ảnh độ sâu.
- Xác định vùng bàn tay trong ảnh màu bằng phương pháp nhận diện màu da.

a) Biến đổi ảnh màu và ảnh độ sâu về cùng trục tọa độ:

Ảnh màu và ảnh độ sâu Kinect có cùng độ phân giải 640x480 tuy nhiên hai cảm biến có thị trường khác nhau nên không biểu diễn cùng một đối tượng. Theo tài liệu khảo sát cảm biến Kinect của ROS, hai thấu kính IR và RGB cách khoảng 2.5cm theo trục x, hướng nhìn hai thấu kính gần như song song với sai khác khoảng 1 độ - tương đương với độ dịch chuyển 1cm ở khoảng cách 1.5m.

Việc calibrate ảnh độ sâu vè ảnh màu được thực hiện khi biết trước các tham số sau:

BẢNG 3: BẢNG CÁC THAM SỐ CALIBARATION

Tham số	Giải thích
d_{off}	Offset của dữ liệu độ sâu.
f_{ir}	Tiêu cự của cảm biến ảnh độ sâu (dùng hồng ngoại).
b	Khoảng cách giữa 2 cảm biến ảnh.
f_{rgb}	Tiêu cự của cảm biến ảnh màu RGB.
k_1, k_2, c_x, c_y	Hệ số biến dạng của cảm biến ảnh màu.
R, t	Ma trận chuyển đổi từ ảnh độ sâu sang ảnh màu RGB.
u,v	Tọa độ của các điểm ảnh trong ma trận ảnh số.

Các bước biến đổi ảnh độ sâu về không gian thực hiện theo lưu đồ sau:

[u,v,kd]ir -- (doff, fir, b) --> [XYZ]ir -- (R,t) --> [XYZ]rgb -- (frgb) --> [u,v]rgb	(2.17)

Các quá trình này có thể biển diễn bằng một ma trận ***D*** có kích thước 4x3 dùng để biến đổi các điểm ảnh trong hệ tọa độ đồng nhất:

$$\mathbf{[u,v,w]rgb} = \boldsymbol{D} * \mathbf{[u,v,k_d,1]ir} \quad (2.18)$$

Việc tính toán ma trận ***D*** có thể thực hiện được bằng các công cụ calibration sẵn có trên mạng hoặc sử dụng kết quả sẵn có trong SDK của Kinect do Microsoft cung cấp.

Hình 18: Kết quả trước và sau khi calibration dữ liệu ảnh độ sâu về không gian của ảnh màu

b) Xác định vùng bàn tay trên ảnh độ sâu.

Vấn đề xác định vùng bàn tay có thể xem như bài toán tracking một đối tượng đang chuyển động. Từ thông tin độ sâu có thể dễ dàng tách được bàn tay với nền nếu biết trạng thái trước đó của bàn tay. Vì đây không phải là vấn đề quan trọng cần phải quyết trong luận văn nên tôi sử dụng thư viện NITE2 của Primesense để tracking chuyển động của bàn tay.

So với thư viện tracking khung xương của Microsoft, NITE2 có ưu điểm chính là tốn rất ít tài nguyên của CPU. Tuy nhiên hiện nay, nhà phát triển của NITE2 đã không còn hoạt động và giải thuật của thư viện cũng không được công bố. Vì vậy trong tương lai cần phải tự phát triển thêm modun giải thuật thay thế cho thư viện này.

c) Xác định vùng bàn tay trong ảnh màu bằng phương pháp nhận diện màu da.

Phân tích thiết kế của Kinect trong phần 2.3.1 cho thấy cảm biến Kinect hoạt động dựa trên công nghệ xử lý ảnh và công nghệ phát chùm tia hồng ngoại. Vì vậy ảnh độ sâu thu được có thể mất mát thông tin vì nhiều lý do có thể kể đến như: bề mặt quá nhỏ không thể nội suy chính xác, tia hồng ngoại từ cảm biến bị che khuất, nhiễu hồng ngoại từ môi trường bên ngoài v.v… Cảm biến ảnh màu ít bị tác động bởi yếu tố này vì vậy có thể bổ sung sự thiếu hụt thông tin từ ảnh độ sâu.

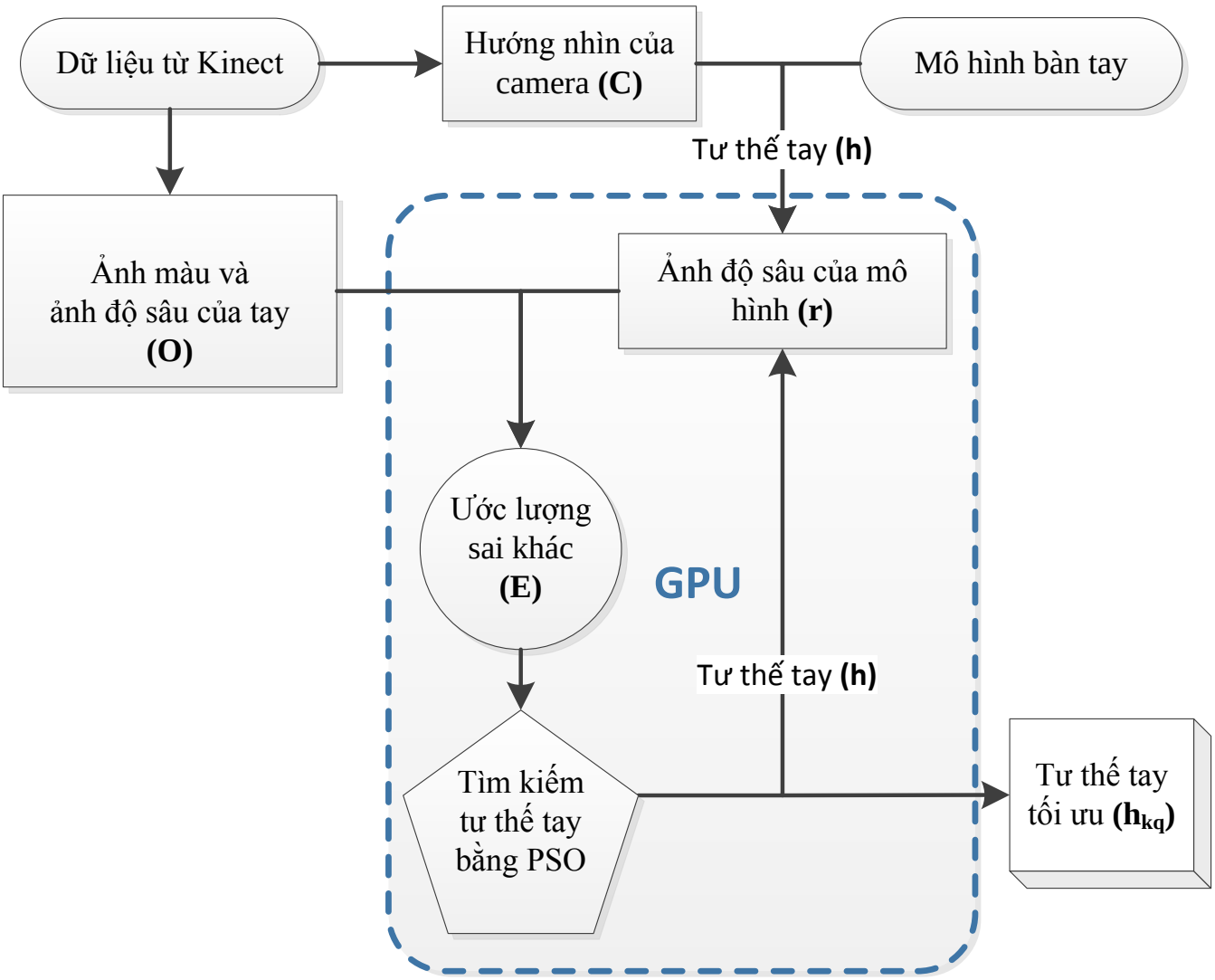
Mô hình tay được xác định từ ảnh độ sâu trước được mở rộng với bán kính là 5 pixel. Giải thuật nhận diện màu da theo nghiên cứu của Antonis Argyros [2] được thực thi để đánh giá các điểm ảnh nào là màu da tay. Kết hợp vùng da tay với ảnh độ sâu ta có được quan sát tốt nhất về mô hình bàn tay trên cảm biến Kinect.

Hình 19: Kết quả nhận dạng mô hình quan sát của bàn tay kết hợp ảnh màu và độ sâu từ cảm biến Kinect

Chương 3: GIẢI THUẬT NHẬN DẠNG

Hình 20 trình bày sơ đồ giải thuật nhận dạng được đề xuất trong luận văn. Giải thuật bao gồm 3 giai đoạn chính:

- Trích chọn ảnh quan quan sát của bàn tay.
- Xây dựng ảnh mô hình giả định của bàn tay tương ứng với góc nhìn quan sát.
- Tìm tư thế tối ưu – 26 bậc tự do của mô hình - bằng giải thuật bầy đàn.



Hình 20: Sơ đồ giải thuật nhận dạng theo phương pháp mô hình đề xuất

Phần xây dựng ảnh mô hình và ảnh quan sát đã được trình bày trong chương 2. Chương này trình bày bước tiếp theo của giải thuật: tìm tư thế tối ưu thực hiện qua hai giai đoạn. Giai đoạn thứ nhất là xây dựng hàm mục tiêu để đánh giá sự sai khác giữa ảnh quan sát và ảnh mô hình. Qua đó, chuyển bài toán nhận dạng thành bài toán tối ưu. Giai đoạn thứ hai là giải bài toán tối ưu sử dụng giải thuật bầy đàn PSO.

3.1 Xây dựng hàm mục tiêu

Theo phương pháp mô hình trình bày trong chương 1, với mô hình quan sát *O*, mục tiêu bài toán là tìm một bộ 26 tham số động học của bàn tay (q_i^{kq}, q_c^{kq}) sao cho ảnh mô hình h_{kq} tạo bởi bộ tham số này giống với ảnh tạo bợi mô hình quan sát *O* nhất. Tiêu chí để so sánh sự sai khác giữa ảnh mô hình và ảnh quan sát được xây dựng theo [2] như sau.

Xét một ảnh mô hình *h* bất kì, bằng phép chiếu hình học lên mặt phẳng quan sát với thông tin về tiêu cự và góc nhìn của camera *C*, ta thu được ảnh độ sâu $r_d(h,C)$. Ảnh độ sâu này sau đó được so sánh với ảnh độ sâu quan sát O_d để tìm ảnh tương quan nhị phân $r_m(h,C)$. Quy tắc tính ảnh tương quan như sau:

Giá trị của mỗi điểm ảnh của $r_m(h,C)$ bằng "1" khi tại vị trí đó sai khác giữa $r_d(h,C)$ và O_d nhỏ hơn một khoảng d_m hoặc tại đó O_d không xác định; trong các trường hợp còn lại, giá trị của $r_m(h,C)$ bằng "0".

Ảnh tương quan này sau đó tiếp tục được so sánh với ảnh màu O_s để loại bớt những vùng độ sâu không thích hợp. Kết quả dẫn đến hàm đánh giá sai khác của toàn bộ mô hình như sau:

$$D(O,h,C) = \frac{\sum \min\left(\left|o_d - r_d\right|, d_M\right)}{\sum (o_s \vee r_m)} + \lambda\left(1 - \frac{2\sum (o_s \wedge r_m)}{\sum (o_s \wedge r_m) + \sum (o_s \vee r_m)}\right) \tag{3.1}$$

trong đó $\vee$ kí hiệu phép HOẶC lôgic; $\wedge$ kí hiệu phép VÀ lôgic; d_M là hằng số dương giới hạn khác biệt về độ sâu; λ là hằng số chuẩn hóa sai khác diện tích; tổng Σ được tính trên toàn bộ các điểm ảnh.

Xét về mặt ý nghĩa, tỉ số: $\frac{\sum \min\left(\left|o_d - r_d\right|, d_M\right)}{\sum (o_s \vee r_m)}$ thể hiện sự sai khác về độ sâu giữa ảnh quan sát và ảnh mô hình; còn tỉ số $\frac{2\sum (o_s \wedge r_m)}{\sum (o_s \wedge r_m) + \sum (o_s \vee r_m)}$ thể hiện sự sai khác về diện tích giữa hai ảnh. Nói cách khác, một tư thế bàn tay *h* được xem là nghiệm cần tìm nếu ảnh mô hình tạo bởi nó có sự sai khác về độ sâu và về diện tích với ảnh quan sát là nhỏ nhất.

Để loại trừ những tư thế bàn tay vô lý ví dụ như ngón trỏ và ngón giữa xuyên qua nhau, một lượng $\lambda_k.kc(h)$ được thêm vào để tăng giá trị sai khác trong những trường hợp trên. Kết quả là hàm mục tiêu sau cùng được biểu diễn như sau:

$$E\left(h,O\right)=D\left(O,h,C\right)+\lambda_k.kc\left(h\right) \tag{3.2}$$

trong đó các tham số cho $D(O,h,C)$ và $E(h,O)$ được chọn như sau:

$$d_m=1cm,\ d_M=4cm$$

$$\lambda=20,\ \lambda_k=10$$

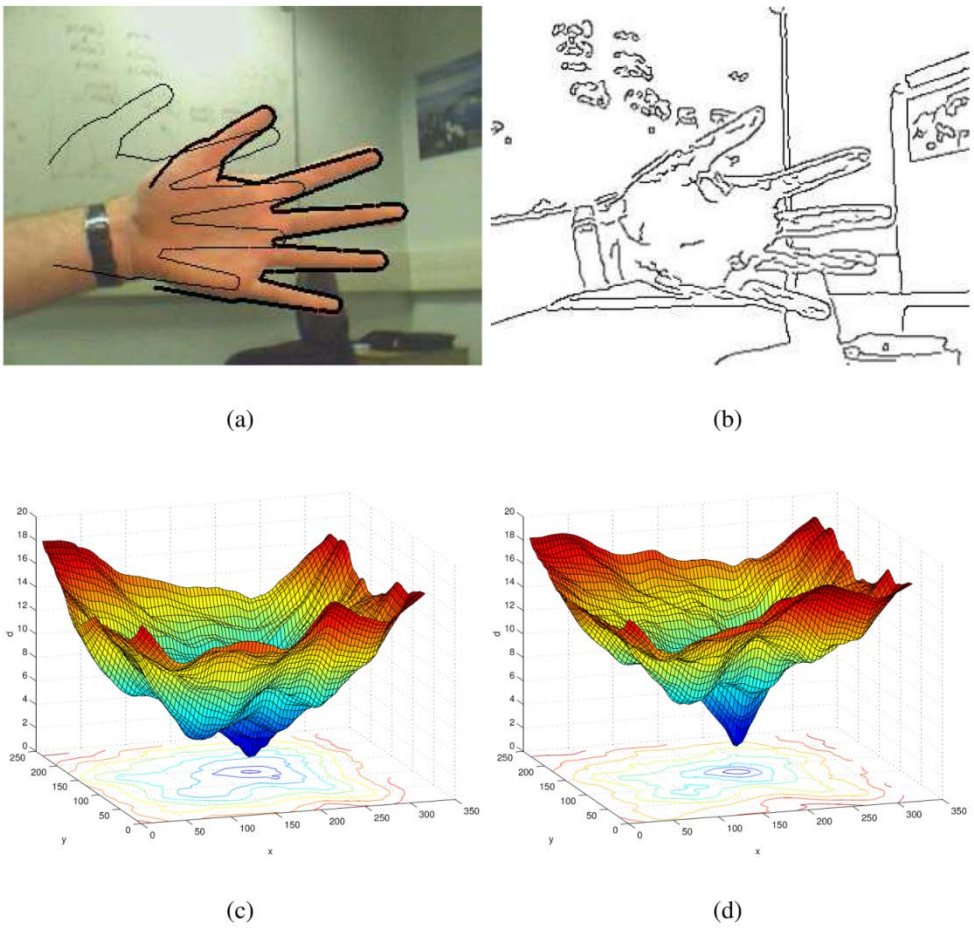
$$kc(h)=\sum_{p\in Q}-\min(\phi(p,h),0)$$

Q là 3 cặp ngón tay không tính ngón cái và ϕ biểu diễn sự sai khác về góc giữa 2 ngón tay trong mỗi cặp.

Với hàm mục tiêu (5), bài toán nhận dạng trở thành bài toán tối ưu trong đó cần tìm 26 tham số của tư thế h để $E\left(h,O\right)$ cực tiểu. Để giải bài toán này, chúng tôi sử dụng phương pháp tối ưu bầy đàn.

3.2 Nhận dạng sử dụng phương pháp tối ưu bầy đàn

Về lý thuyết, rất khó để tìm lời giải tường minh cho phương trình (5) khi mà không gian các lời giải của bài toán là dạng hàm mũ của số bậc tự do của ngón tay. Thật vậy, trong khảo sát của Stenger với hàm mục tiêu Chamfer Distance (Hình 21) cho thấy ngay cả khi các ngón tay của mô hình giả định trùng với mô hình quan sát, việc dịch chuyển của sáu bậc tự do của cổ tay sẽ cho kết quả đồ thị là một hàm có nhiều cực trị.



Hình 21: Đồ thị của Stenger biểu diễn sự biến thiên của hàm đánh giá mục tiêu Chamfer Distance khi bàn tay di chuyển trong không gian.

(a) ảnh đầu vào với 2 mô hình đường biên được vẽ chồng lên, (b) kết quả tách biên của ảnh đầu vào, (c) đồ thị biến thiên của hàm mục tiêu khi dịch chuyển mô hình giả định trong không gian – mô hình giả định này trùng với mô hình quản sát trong hình a , (d) đổ thị biến thiên của hàm mục tiêu khi dịch chuyển mô hình giả định có trạng thái không giống với mô hình quan sát.

Một xu hướng mới hiện nay đó là áp dụng các thuật toán heuristic vào các bài toán tối ưu nhằm tìm ra những giá trị gần với giá trị tối ưu. Các thuật toán xấp xỉ được phát triển để tìm ra những lời giải tốt nằm trong một cận xấp xỉ đối với lời giải tối ưu. Trong bài toán nhận dạng bàn tay, các phương pháp giải thống kê thường được sử dụng như giải thuật Powell [6], giải thuật Nelder – Mead [7], hay giải thuật di truyền [8]. Trong luận văn này, chúng tôi sử dụng phương pháp tối ưu bầy đàn nhờ tốc độ hội tụ nhanh và đơn giản trong cài đặt [9].

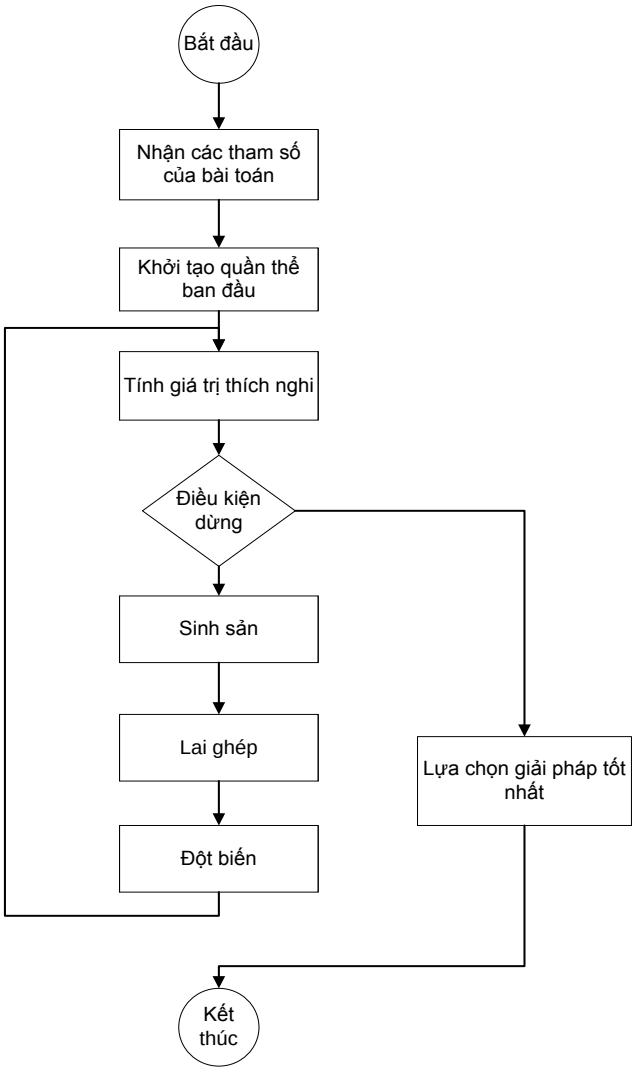
3.2.1 Giới thiệu về giải thuật tối ưu bầy đàn PSO

Phương pháp tối ưu bầy đàn là một dạng của các thuật toán tiến hóa quần thể đã được biết đến trước đây như thuật giải di truyền (Genetic algorithm), thuật toán đàn kiến (Ant colony algorithm).

a) Giải thuật di truyền - GA

Giải thuật di truyền là một phân ngành của giải thuật tiến hóa vận dụng các nguyên lý của tiến hóa như di truyền, đột biến, chọn lọc tự nhiên, và trao đổi chéo để tìm kiếm các giải pháp tích hợp cho bài toán tối ưu. Giải thuật di truyền cũng như các thuật toán tiến hoá đều được hình thành dựa trên một quan niệm được coi là một tiên đề phù hợp với thực tế khách quan. Đó là quan niệm *"Quá trình tiến hoá tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và tự nó đã mang tính tối ưu".* Quá trình tiến hoá thể hiện tính tối ưu ở chỗ thế hệ sau bao giờ cũng tốt hơn thế hệ trước. Giải thuật di truyền có các khái niệm cơ bản sau:

- **Cá thể, nhiễm sắc thể**: Trong giải thuật di truyền, một cá thể biểu diễn một giải pháp của bài toán. Không giống với trong tự nhiên, một cá thể có nhiều nhiễm sắc thể (NST), ở đây ta quan niệm một cá thể có một nhiễm sắc thể. Do đó khái niệm cá thể và nhiễm sắc thể trong giải thuật di truyền coi như là tương đương. Một NST được tạo thành từ nhiều gen, mỗi gen có thể có các giá trị khác nhau để quy định một tính trạng nào đó. Trong GA, một gen được coi như một phần tử trong chuỗi NST.
- **Quần thể**: là một tập hợp các cá thể có cùng một số đặc điểm nào đấy. Trong giải thuật di truyền ta quan niệm quần thể là một tập các lời giải của một bài toán.
- **Chọn lọc**: trong tự nhiên, quá trình chọn lọc và đấu tranh sinh tồn đã làm thay đổi các cá thể trong quần thể. Những cá thể tốt, thích nghi được với điều kiện sống thì có khả năng đấu tranh lớn hơn, do đó có thể tồn tại và sinh sản. Các cá thể không thích nghi được với điều kiện sống thì dần mất đi. Dựa vào nguyên lý của quá trình chọn lọc và đấu tranh sinh tồn trong tự nhiên, chọn lựa các cá thể trong GA chính là cách chọn các cá thể có độ thích nghi tốt để đưa vào thế hệ tiếp theo hoặc để cho lai ghép, với mục đích là sinh ra các cá thể mới tốt hơn. Có nhiều cách để lựa chọn nhưng cuối cùng đều nhằm đáp ứng mục tiêu là các cá thể tốt sẽ có khả năng được chọn cao hơn.
- **Lai ghép**: trong tự nhiên là sự kết hợp các tính trạng của bố mẹ để sinh ra thế hệ con. Trong giải thuật di truyền, lai ghép được coi là một sự tổ hợp lại các tính chất (thành phần) trong hai lời giải cha mẹ nào đó để sinh ra một lời giải mới mà có đặc tính mong muốn là tốt hơn thế hệ cha mẹ. Đây là một quá trình xảy ra chủ yếu trong giải thuật di truyền.
- **Đột biến**: là một sự biến đổi tại một (hay một số) gen của nhiễm sắc thể ban đầu để tạo ra một nhiễm sắc thể mới. Đột biến có xác suất xảy ra thấp hơn lai ghép. Đột biến có thể tạo ra một cá thể mới tốt hơn hoặc xấu hơn cá thể ban đầu. Tuy nhiên trong giải thuật di truyền thì ta luôn muốn tạo ra những phép đột biến cho phép cải thiện lời giải qua từng thế hệ.



Hình 22: Sơ đồ mô tả giải thuật di truyền

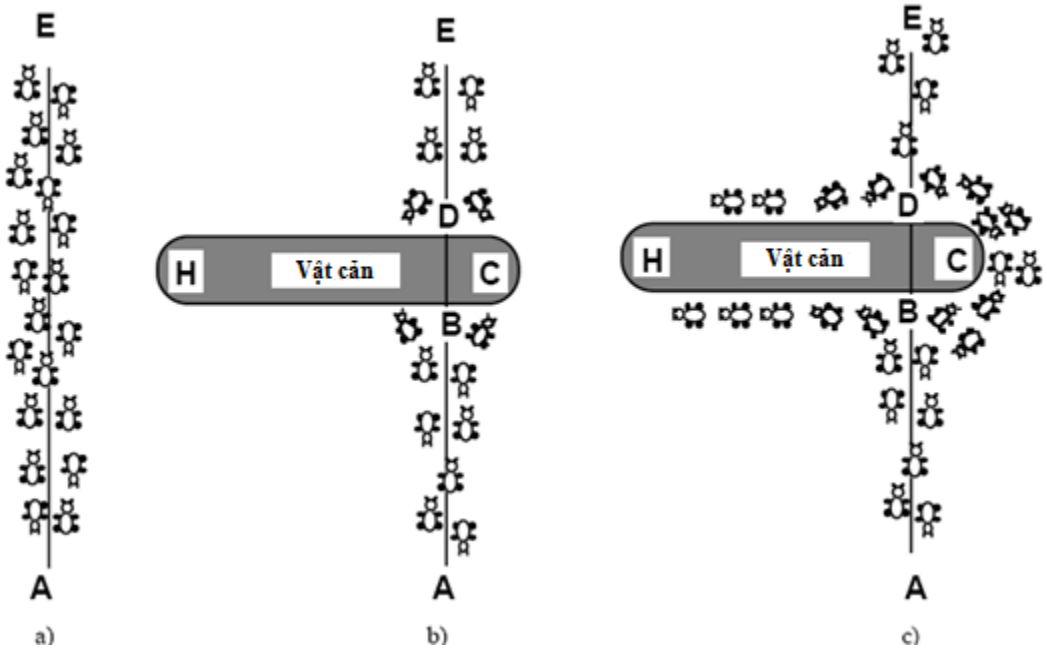
Thông thường, những giải pháp được thể hiện dưới dạng nhị phân với những chuỗi 0 và 1, nhưng lại mang nhiều thông tin mã hóa khác nhau. Quá trình tiến hóa xảy ra từ một tập hợp những cá thể hoàn toàn ngẫu nhiên ở tất cả các thế hệ. Trong từng thế hệ, tính thích nghi của tập hợp này được ước lượng, nhiều cá thể được chọn lọc định hướng từ tập hợp hiện thời (dựa vào thể trạng), được sửa đổi (bằng đột biến hoặc tổ hợp lại) để hình thành một tập hợp mới. Tập hợp này sẽ tiếp tục được chọn lọc lặp đi lặp lại trong các thế hệ kế tiếp của giải thuật.

b) Giải thuật tối ưu đàn kiến – ACO

Các thuật toán kiến lần đầu tiên được giới thiệu bởi Dorigo và các cộng sự như là cách tiếp cận đa tác tử tới các vấn đề về tối ưu tổ hợp khó, như bài toán người du lịch (TSP), bài toán người đưa thư. Hiện nay số lượng các ứng dụng càng ngày càng tăng và các nhà khoa học đã ứng dụng nó vào rất nhiều các vấn đề tối ưu rời rạc. Các ứng dụng gần đây có thể kể đến như các bài toán lập lịch, tô màu đồ thị, định hướng trong mạng truyền thông, v.v…

Các thuật toán kiến là các thuật toán dựa vào sự quan sát các bầy kiến thực. Kiến là loại cá thể sống bầy đàn. Chúng giao tiếp với nhau thông qua mùi mà chúng để lại trên hành trình mà chúng đi qua. Mỗi kiến khi đi qua một đoạn đường sẽ để lại trên đoạn đó một chất mà chúng ta gọi là mùi. Số lượng mùi sẽ tăng lên khi có nhiều kiến cùng đi qua. Các con kiến khác sẽ tìm đường dựa vào mật độ mùi trên đường, mật độ mùi càng lớn thì chúng càng có xu hướng chọn. Dựa vào hành vi tìm kiếm này mà đàn kiên tìm được đường đi ngắn nhất từ tổ đến nguồn thức ăn và sau đó quay trở tổ của mình.

Sau đây là ví dụ về luồng đi của đàn kiến thực tế:



Hình 23: Hành vi của bầy kiến khi gặp vật cản

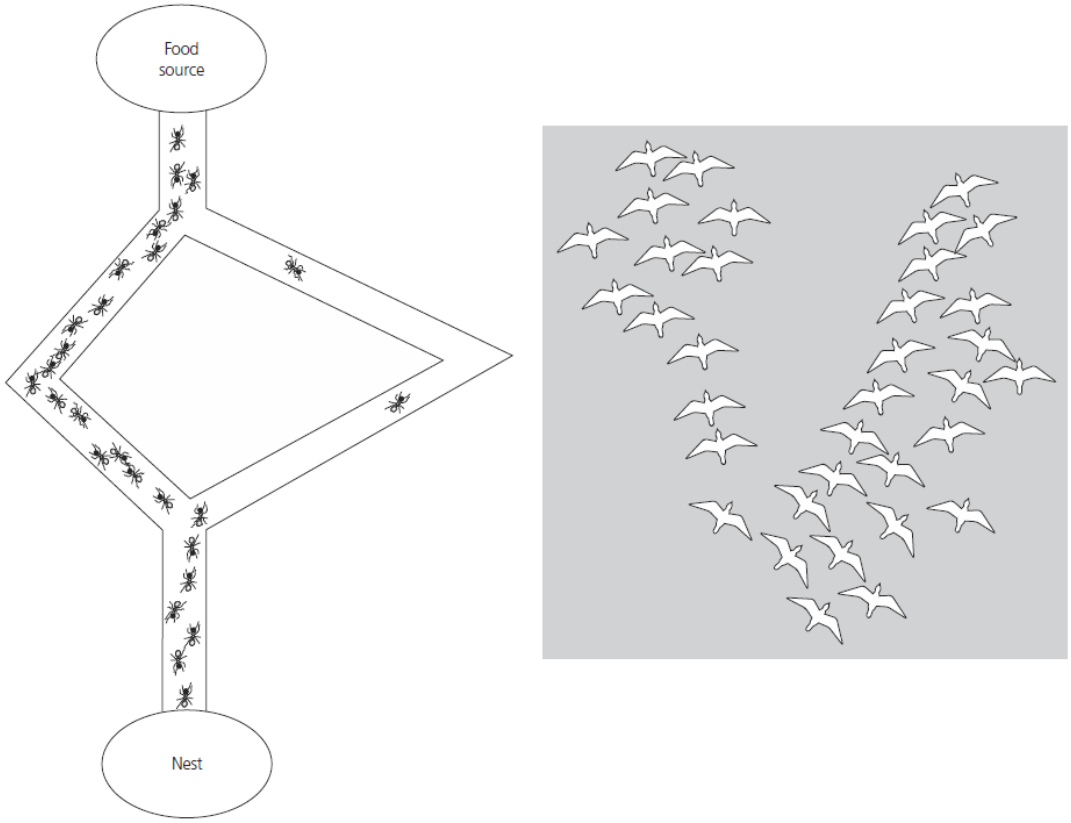
- Kiến đi theo đường thẳng giữa A và E
- Khi có chướng ngại vật kiến sẽ chọn hướng đi, có hai hướng với khả năng kiến sẽ chọn là như nhau.
- Trên đường ngắn hơn thì nhiều mùi (pheromone) hơn

Thuật toán tối ưu bầy kiến (ACO) nghiên cứu các hệ thống nhân tạo dựa vào hành vi tìm kiếm của bầy kiến thực và được sử dụng để giải quyết các vấn đề về *tối ưu rời rạc*. Thuật toán bầy kiến siêu tìm kiếm (ACO meta_heuristic) lần đầu tiên được Dorigo, Di Caro và Gambardella đề xuất vào năm 1999.

c) Giải thuật tối ưu bầy đàn - PSO

Tuy vậy PSO khác với GA ở chỗ nó thiên về sử dụng sự tương tác giữa các cá thể trong một quần thể để khám phá không gian tìm kiếm. PSO là kết quả của sự mô hình hóa việc đàn chim bay đi tìm kiếm thức ăn cho nên nó thường được xếp vào các loại thuật toán có sử dụng trí tuệ bầy đàn. Được giới thiệu vào năm 1995 tại một hội nghị của IEEE bởi

James Kennedy và kỹ sư Russell C. Eberhart. Thuật toán có nhiều ứng dụng quan trọng trong tất cả các lĩnh vực mà ở đó đòi hỏi phải giải quyết các bài toán tối ưu hóa.



Hình 24: So sánh di chuyển của đàn kiến và đàn chim trong không gian 2 chiều

Ví dụ đơn giản về PSO là quá trình tìm kiếm thức ăn của một đàn chim. Không gian tìm kiếm thức ăn lúc này là toàn bộ không gian ba chiều mà chúng ta đang sinh sống. Tại thời điểm bắt đầu tìm kiếm cả đàn bay theo một hướng nào đó, có thể là rất ngẫu nhiên. Tuy nhiên sau một thời gian tìm kiếm một số cá thể trong đàn bắt đầu tìm ra được nơi có chứa thức ăn. Tùy theo số lượng thức ăn vừa tìm kiếm, mà cá thể gửi tín hiệu đến các các cá thể khác đang tìm kiếm ở vùng lân cận. Tín hiệu này lan truyền trên toàn quần thể. Dựa vào thông tin nhận được mỗi cá thể sẽ điều chỉnh hướng bay và vận tốc theo hướng về nơi có nhiều thức ăn nhất. Cơ chế truyền tin như vậy thường được xem như là một kiểu hình của trí tuệ bầy đàn. *Cơ chế này giúp cả đàn chim tìm ra nơi có nhiều thức ăn nhất trên không gian tìm kiếm vô cùng rộng lớn.*

Giải thuật bầy đàn giải bài toán tối ưu bằng cách tạo ra một tập hợp gồm n phần tử, mỗi phần tử di chuyển và tiến hóa qua mỗi bước để rồi cuối cùng hội tụ tại điểm tối ưu. Ban đầu, các phần tử được gán một vị trí và vận tốc ngẫu nhiên. Sau đó, tại mỗi bước, mỗi phần tử cập nhật vị trí tốt nhất của nó, P_k, và vị trí tốt nhất của cả đàn, G_k. Gọi x_k và v_k lần lượt là vị trí và vận tốc hiện tại của mỗi phần tử. Khi đó, vị trí và vận tốc tiếp theo của phần tử đó được cập nhật như sau:

$$v_{k+1} = w\left(v_k + c_1 r_1 \left(P_k - x_k\right) + c_2 r_2 \left(G_k - x_k\right)\right) \quad (3.3)$$

$$x_{k+1} = x_k + v_{k+1} \quad (3.4)$$

với w là hệ số giảm vận tốc, c_1 là hằng số đặc trưng cho yếu tố cá thể, c_2 là hằng số đặc trưng cho yếu tố bầy đàn, r_1 và r_2 là hai biến ngẫu nhiên phân phối đều trong khoảng [0,1]. Phương trình (6) và (7) hàm ý mỗi phần tử sẽ di chuyển ngẫu nhiên nhưng có khuynh hướng tiến về vị trí tốt nhất của cả đàn và vị trí tốt nhất mà nó đã đi qua. Tương quan giữa yếu tố bầy đàn và yếu tố cá thể được thể hiện qua các hệ số c_1 và c_2.

3.2.2 Ứng dụng giải thuật tối ưu bầy đàn vào nhận dạng

Áp dụng vào bài toán nhận dạng, vị trí của mỗi phần tử được định nghĩa là vectơ 26 chiều ứng với 26 tham số động học của bàn tay hay chính là tư thế h của bàn tay. Vận tốc được định nghĩa là vectơ 26 chiều thể hiện sự thay đổi tư thế của bàn tay qua mỗi bước. Khi khởi tạo, vị trí của mỗi phần tử được gieo ngẫu nhiên tạo thành các tư thế h_1, h_2, …h_n. Vận tốc ban đầu được đặt bằng 0. Từ phương trình (5), giá trị của hàm mục tiêu $E(h_i, O)$ được tính cho mỗi tư thế. Từ đó, vị trí tốt nhất của mỗi phần tử P_k và vị trí tốt nhất của cả đàn G_k được xác định. Vận tốc của mỗi phần tử ở thế hệ tiếp theo sau đó được xác định bởi phương trình (6) và vị trí tiếp theo được xác định bởi phương trình (7). Trải qua các bước tiến hóa, vị trí (hay tư thế bàn tay) của mỗi phần tử sẽ tiến dần tới tư thế thực quan sát bởi camera. Thuật toán dừng khi sai số hàm mục tiêu nhỏ hơn giá trị đặt hoặc số bước tiến hóa đạt tới giá trị tối đa cho phép.

Trong hệ thống của chúng tôi, số phần tử của đàn được đặt là 64. Không gian tìm kiếm được giới hạn bởi khoảng giá trị của các phần tử theo bảng 1 và bảng 2. Điều kiện dừng là khi giá trị hàm mục tiêu nhỏ hơn 1.0 hoặc số bước tiến hóa đạt 30. Các hệ số của phương trình (6) được đặt như sau:

$c_1 = 2.8,$

$c_2 = 1.3,$

$w = 2 / \left|2 - \psi - \sqrt{\psi^2 - 4\psi}\right|$ với $\psi = c_1 + c_2$

Trong quá di chuyển theo giải thuật PSO, do số chiều lớn nên các vị trí đốt ngón tay thường bị kẹt tại các đỉnh tối ưu cục bộ thay vì tiến tới đỉnh tối ưu toàn cục. Để giải quyết vấn đề này, các phần tử được tạo đột biến (mutation). Cứ sau 3 bước tiến hóa, một nửa số phần tử kém nhất trong đàn được gieo lại ngẫu nhiên 20 chiều tương ứng với các tham số góc của các đốt ngón tay.

Chương 4: TĂNG TỐC THUẬT TOÁN SỬ DỤNG KHỐI XỬ LÝ ĐỒ HỌA GPU

Do không gian tìm kiếm lớn với 26 chiều tương ứng với 26 bậc tự do của bàn tay, giải thuật bầy đàn cần sử dụng số lượng phép tính lớn để tìm kiếm mà nếu xử lý tuần tự bằng CPU sẽ không đảm bảo yếu tố thời gian thực. Để giải quyết vấn đề này, khả năng xử lý song song của khối xử lý đồ họa GPU trên máy tính được tận dụng để cải thiện khả năng xử lý của hệ thống nhận dạng.

4.1 Xử lý song song trên máy tính và tiêu chuẩn OpenCL

Trong hơn hai thập kỉ gần đây, nền công nghiệp máy tính phát triển dưới ảnh hưởng của định luật Gordon Moore: mật độ bán dẫn trong một inch vuông tăng gấp đôi sang mỗi 18 tháng. Theo dự đoán này thì hiệu năng của một ứng dụng thông thường sẽ được tăng gấp đôi mỗi hai năm khi thế hệ vi xử lý tiếp theo ra đời. Sự cải thiện liên tục trong quá trình sản xuất bán dẫn và công nghệ thiết kế bộ vi xử lý là nguyên nhân chính của xu thế này, bởi lẽ nhờ chúng mà vi xử lý thế hệ tiếp theo có thể thu nhỏ tất cả các bóng bán dẫn và giảm lượng điện năng tiêu thụ. Vì vậy, các bộ vi xử lý đời mới có thể tăng gấp đôi mật độ bóng bán dẫn để cải thiện tốc độ (xung nhịp) lên 50% trong khi đó lượng điện năng tiêu thụ không thay đổi. Khi có nhu cầu về hiệu năng cao hơn, các kiến trúc sư máy tính chỉ tập trung vào việc tăng số bóng bán dẫn để đẩy số xung nhịp cao thêm, và thêm vào các thuộc tính kiến trúc có mục tiêu chính là cải thiện thêm hiệu năng cho ứng dụng mới.

Trong những năm 2000, bóng bán dẫn đã trở nên quá nhỏ đến mức các định luật về nhiệt động học và lượng tử bắt đầu ảnh hưởng tới đặc tính của toàn bộ con chip. Vì vậy khi tần số xung nhịp và mật độ bóng bán dẫn tăng, đồng nghĩa với việc phải tăng lượng tiêu thụ điện. Nhiều hãng công nghệ cho rằng, định luật Moore sẽ không còn đúng trong tương lai không xa nữa.

Để giải đáp ứng kì vọng duy trì định luật Moore - tăng gấp đôi hiệu năng sau mỗi chu kì thời gian (không phải là 2 năm như trước nữa), có hai hướng thay đổi chính đã sảy ra (1) là thay vì tăng tần số xung nhịp, vi xử lý hiện đại tăng số lượng lõi xử lý trên một đế. Xu thế này bắt buộc phần mềm phải có sự thay đổi tương thích. Do chúng ta không thể kì vọng một phần cứng có thể hoạt động tốt với bất cứ phần mềm nào, vì vậy chúng ta cần phải phát triển các cách thức triển khai mới cho cùng một ứng dụng mà đảm bảo khả năng phát huy được ưu điểm của kiến trúc nhiều nhân trong vi xử lý, và (2) nhiệt độ và công suất tiêu thụ phải là ưu tiên hàng đầu cho mỗi thiết kế phần cứng trong tương lai. Xu thế này thúc đẩy cộng đồng bắt đầu tìm kiếm một giải pháp cho tính toán không đồng nhất: một hệ thống được tổ hợp bởi các hệ thống con khác nhau, mỗi hệ thống con lại tối ưu cho một công việc. Thí dụ điển hình

nhất là rất nhiều hệ thống kết hợp CPU truyền thống với khối xử lý đồ họa GPU hay các bo mạch FPGA. Sự tích hợp này có thể thực hiện ờ nhiều cấp độ khác nhau: cấp độ hệ thống, cấp độ bo mạch và hiện nay là cấp độ lõi.

Việc phát triển các phần mềm cho hệ thống song song đồng nhất và phân tán được cho là công việc không đơn giản, ngay cả khi có rất nhiều ngôn ngữ lập trình, phương thức phát triển, thuật toán và công cụ phát hiện lỗi đã được nghiên cứu. Phát triển phần mềm cho hệ thống không đồng nhất – đa dụng vẫn còn tương đối mới và do đó chưa hoàn thiện và còn gặp nhiều khó khăn.

Bởi việc thiết kế các hệ thống không đồng nhất là không thể tránh khỏi, rất nhiều các nhà sản xuất phần mềm và phần cứng lớn như AMD, Nvidia, Intel, IBM bắt đầu tạo ra môi trường phần mềm hỗ trợ chúng.

Ngôn ngữ tính toán mở (OpenCL) được thiết kế để đáp ứng nhu cầu cần thiết trên. Chúng được định nghĩa và quản lý bởi tổ chức công nghệ phi lợi nhận Khronos. Ngôn ngữ này và môi trường phát triển của chúng được vay mượn từ rất nhiều các thiết kế cơ bản từ rất nhiều tiêu chuẩn phần cứng nổi tiếng và thành công như CUDA, CAL, CTM, và tập hợp lại để tạo ra một môi trường phát triển phần mềm độc lập cho phần cứng. Nó hỗ trợ các cấp độ khác nhau của xử lý song song và các phương thức chia sẻ hiệu quả cho hệ thống đồng nhất hoặc không đồng nhất, hệ thống sử dụng một hoặc nhiều CPU, GPU, FPGA và hứa hẹn thêm nhiều các thiết bị trong tương lai. Để hỗ trợ nhiều thiết bị trong tương lai, OpenCL định nghĩa một run-time cho phép quản lý tài nguyên và kết hợp các loại phần cứng khác nhau dưới cùng một môi trường thực thi và hi vọng rằng trong tương lai nó cho phép cân bằng động các quá trình tính toán, năng lượng tiêu thụ và tài nguyên như bộ nhớ.

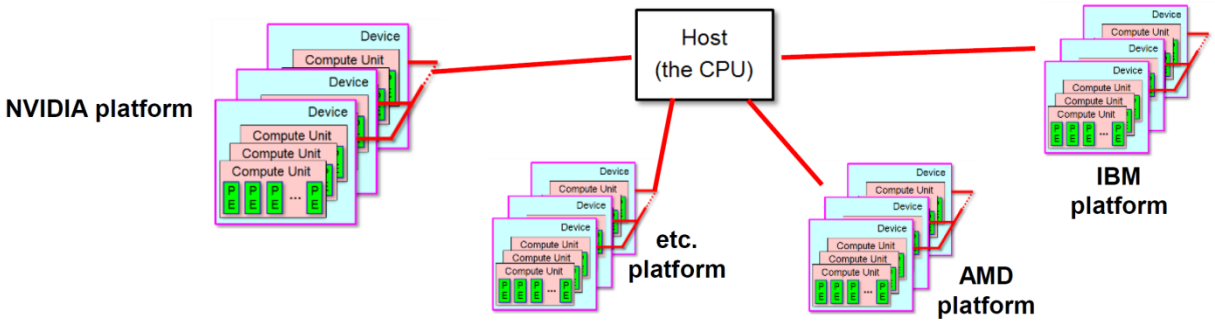
Tiêu chuẩn OpenCL cung cấp một giao diện lập trình ứng dụng (API) tổng quát đủ để chạy trên các kiến trúc phần cứng khác nhau và có khả năng thích nghi với từng nền tảng để đạt tới hiệu năng tốt nhất. Sử dụng một ngôn ngữ lõi và chuẩn hóa theo các đặc điểm kĩ thuật, mỗi chương trình phần mềm được thiết kế bởi một nhà sản xuất phần mềm có thể hoạt động trên các thiết bị phần cứng khác nhau. Một tập bốn mô hình của OpenCL tạo nên tính dễ chuyển đổi, độc lập giữa thiết bị và phần mềm vì vậy phần mềm có khả năng tăng tốc bằng cách thực thi trên nhiều thiết bị khác.

Giao diện lập trình OpenCL viết bằng ngôn ngữ C và có bọc thêm giao diện lập trình C++. Ngoài ra rất nhiều ứng dụng bên thứ ba viết thêm giao diện cho OpenCL từ những ngôn ngữ như Java, Python và .NET. Mã nguồn hoạt động trên thiết bị của OpenCL được viết bằng ngôn ngữ C – theo một phiên bản giới hạn của tiêu chuẩn C99 với các mở rộng phù hợp với dữ liệu song song và với nhiều loại thiết bị khác nhau.

Tiêu chuẩn OpenCL được định nghĩa làm 4 phần, gọi là 4 mô hình, có thể tổng kết ngắn như sau:

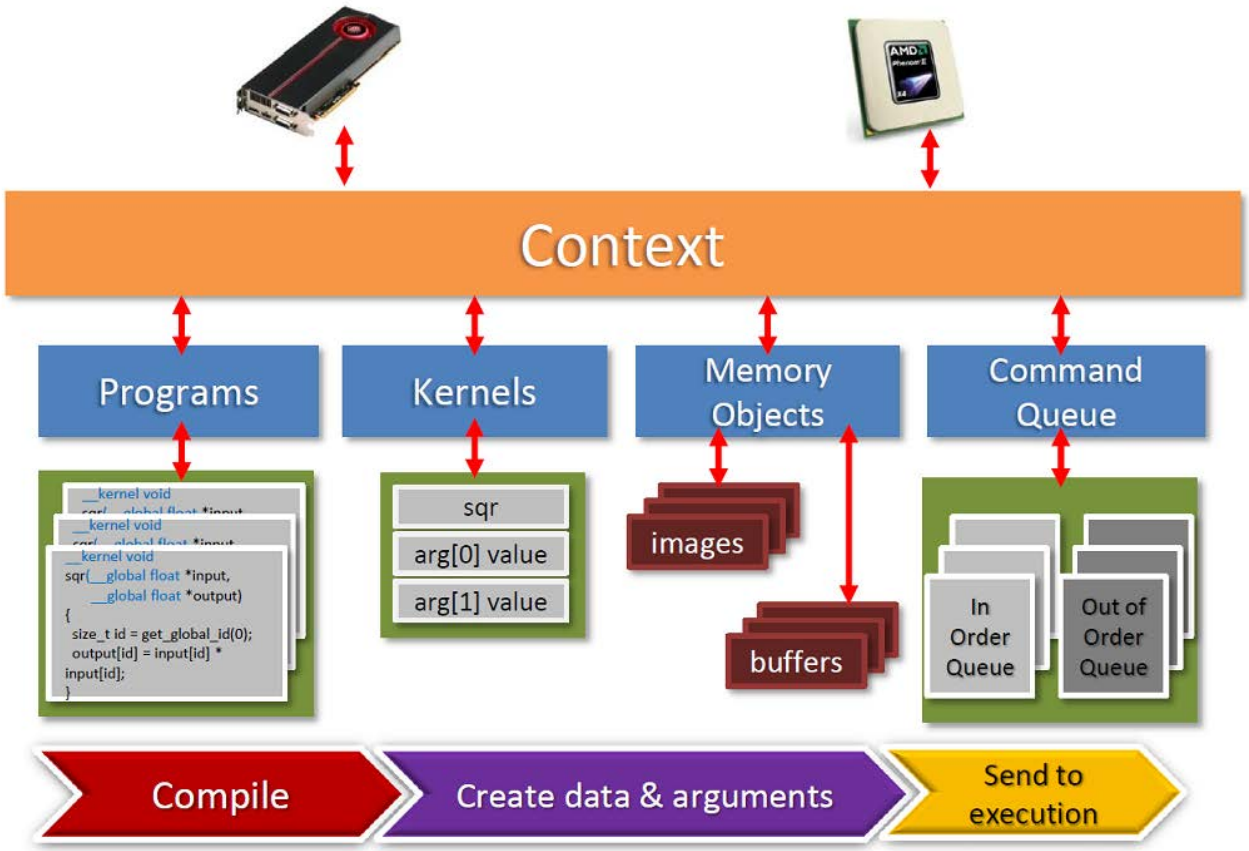
1. **Mô hình nền tảng**: định nghĩa rằng có một vi xử lý thực hiện công việc điều phối các thực thi (gọi là host) và nhiều vi xử lý có khả năng thực thi các mã C của OpenCL (gọi là các

thiết bị). Nó định nghĩa một mô hình phần cứng trừu tượng được sử dụng bởi các lập trình viên khi viết các hàm chức năng bằng OpenCL C (gọi tắt là nhân / kernel) để thực hiện trên các thiết bị.



Hình 25: Mô hình nền tảng với một host điều phối các thiết bị tính toán

2. Mô hình thực thi: định nghĩa OpenCL được cấu hình trên môi trường của host như thế nào và làm thế nào các nhân được thực hiện trên thiết bị. Điều này bao gồm việc thiết lập một bối cảnh (context) trên host, cung cấp cơ chế tương tác giữa host với thiết bị, và xác định một mô hình đồng thời được sử dụng để thực thi nhân chương trình trên các thiết bị.



Hình 26: Lưu đồ thực thi của OpenCL với Host là CPU và thiết bị tính toán là một GPGPU

Chú thích lưu đồ hình 26:

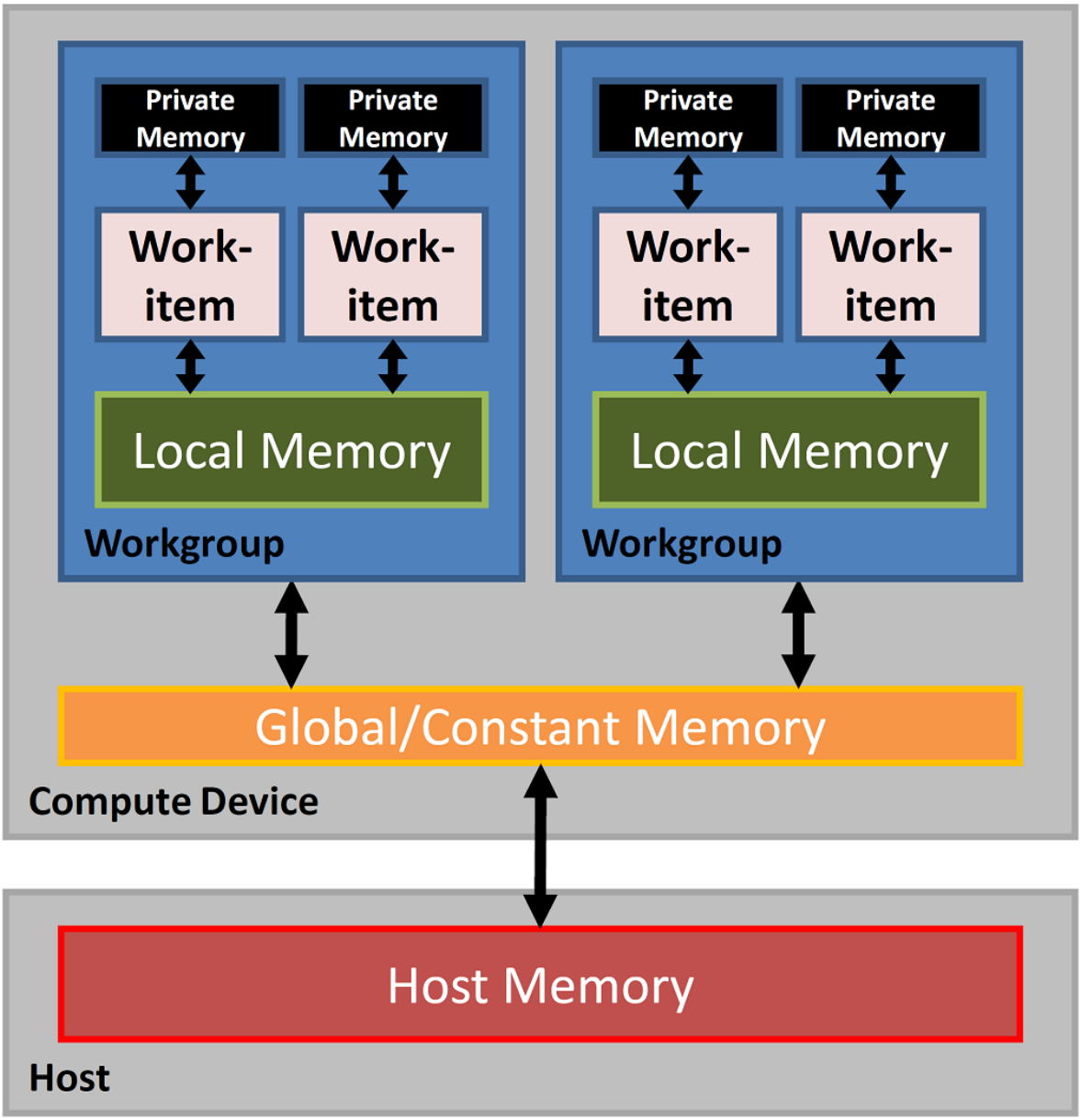
- ***Kernels***: hay nhân chương trình là các hàm OpenCL chạy trên các thiết bị.
- ***Program objects***: là mã nguồn của chương trình được chạy trên nhân.
- ***Memory objects***: là một tập các đối tượng trong bộ nhớ vật lý mà các thiết bị OpenCL có thể truy cập vào được. Nó chứa các tham số đầu vào để thực thi một nhân chương trình. Đối tượng bộ nhớ có thể là một mảng một chiều (buffer) hay một mảng hai chiều (image).
- ***Command-Queues***: Sự tương tác giữa host và các thiết bị OpenCL xảy ra thông qua các lệnh gửi bởi host đến các *Command-Queues* – hay hàng đợi lệnh. Các lệnh chờ đợi trong hàng đợi lệnh cho đến khi họ thực hiện trên thiết bị OpenCL. Một hàng đợi lệnh được tạo ra bởi host và gắn liền với một thiết bị OpenCL duy nhất sau khi một context đã được xác định. Host đưa các lệnh vào hàng đợi, sau đó lên kế hoạch thực thi tới các thiết bị tính toán. OpenCL hỗ trợ ba loại lệnh chính:
    - ***Kernel execution commands***: lệnh để thực thi một nhân trên một phần tử tính toán của thiết bị.
    - ***Memory commands***: truyền dữ liệu giữa host và các đối tượng bộ nhớ khác nhau (*Memory objects*), chuyển tiếp dữ liệu giữa các đối tượng bộ nhớ, hoặc ánh xạ và ngừng ánh xạ từ không gian địa chỉ của host tới đối tượng bộ nhớ.
    - ***Synchronization commands***: đặt ra các giới hạn hay các trình tự để các lệnh thực thi.

3. Mô hình bộ nhớ: xác định hệ thống phân cấp bộ nhớ trừu tượng mà các nhân chương trình sử dụng, không phụ thuộc vào kiến trúc bộ nhớ cơ bản thực tế. Mô hình bộ nhớ gần giống với hệ thống phân cấp bộ nhớ GPU hiện tại, mặc dù vậy nhưng điều này không giới hạn khả năng thích ứng với các thiết bị khác.

- **Host memory**: vùng nhớ này chỉ hiển thị đối với host. Như với hầu hết các chi tiết liên quan đến host, OpenCL định nghĩa duy nhất một cách thực tương tác của bộ nhớ máy chủ với các đối tượng OpenCL và cấu trúc.
- **Global memory**: vùng nhớ này cho phép đọc/ghi và truy cập tới tất các các work-items trong tất cả các work-groups.Work-items có thể đọc và ghi vào tất cả các phần tử của đối tượng bộ nhớ nằm trong global memory. Việc đọc ghi của vùng nhớ này có thể lưu trữ tạm thời tùy thuộc vào khả năng của thiết bị.
- **Constant memory**: vùng nhớ lưu trữ những hằng số trong suốt quá trình thực thi cả một nhân. Host cấp phát và khởi tạo đối tượng bộ nhớ và đặt vào constant memory. Work-items chỉ có quyền truy cập vào đối tượng bộ nhớ này.
- **Local memory**: vùng nhớ này là cục bộ đối với mỗi work-group. Bộ nhớ này có thể sử dụng để cấp phát các giá trị được chia sẻ bởi tất cả các work-items trong work-

groups đó. Nó có thể được sử dụng như vùng nhớ chuyên dụng trên thiết bị OpenCL. Ngoài ra, các khu local memory có thể được ánh xạ lên các phần của global memory.

- **Private memory**: vùng nhớ này là vùng nhớ riêng của mỗi work-items. Các work-item không thể nhìn thấy các giá trị được định nghĩa trong vùng nhớ đó.

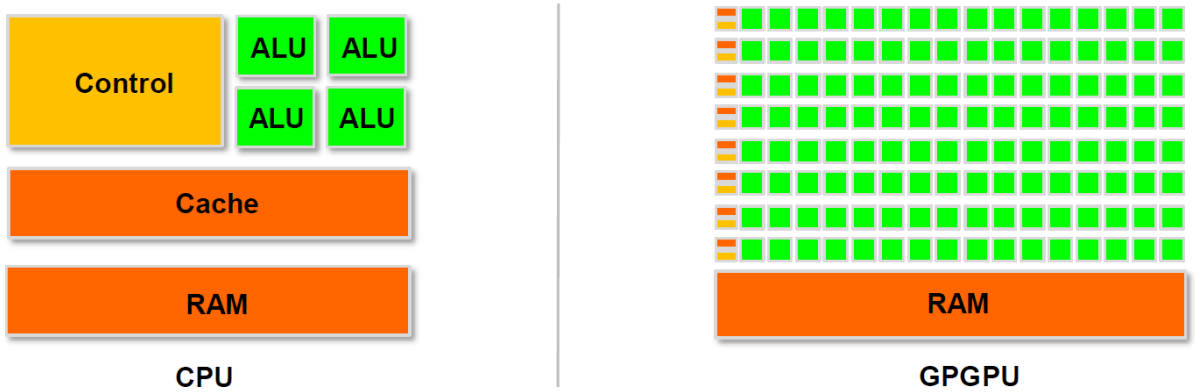


Hình 27: Mô hình bộ nhớ của host và thiết bị trong OpenCL

4. Mô hình lập trình: Xác định làm thế nào mô hình đồng thời được ánh xạ tới phần cứng vật lý.

4.2 Tăng tốc thuật toán trên GPU

Với mục đích sử dụng khác nhau nên CPU và GPU (và hiện nay là GPGPU) có thiết kế vật lý rất khác nhau:

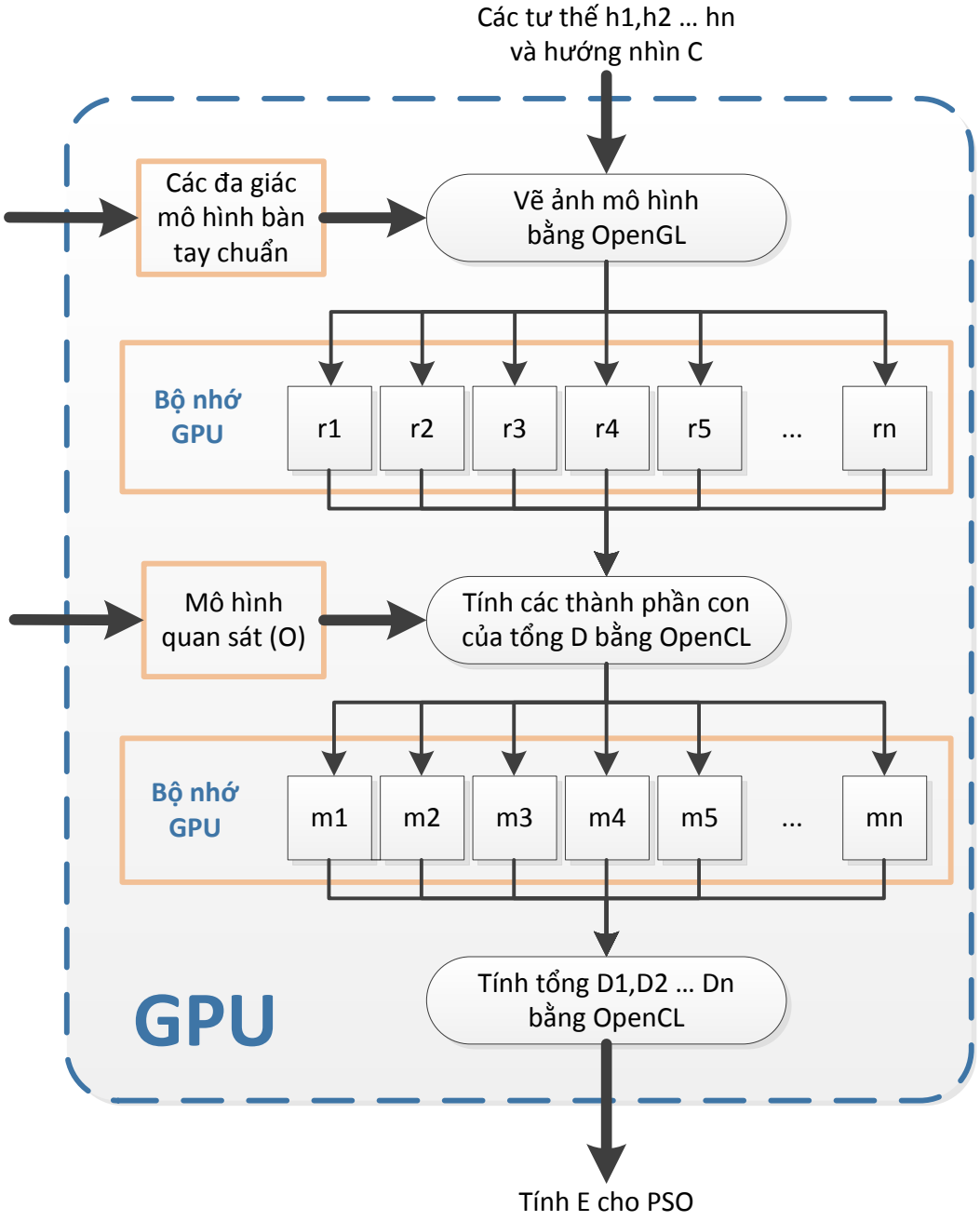


Hình 28: So sánh kiến trúc của CPU và GPGPU

BẢNG 4: SO SÁNH CPU VÀ GPU

	CPU	GPU
Đặc điểm thiết kế	- Thiết kế chip ưu tiên nhiều hơn cho khu vực điều khiển (Control) thay vì các bộ logic số học (ALU). - Tộc độ xử lý với một luồng rất nhanh do: khả năng dự đoán đường ống lệnh và rẽ nhánh tốt, khả năng điều khiển tuần tự và song song tốt. - Nhiều cấp độ bộ nhớ cache để giảm bớt độ trễ truy cập. - Tệp tin thanh ghi nhỏ do số lượng luồng hoạt động ít. Luồng quản lý bởi hệ điều hành.	- Thiết kế với rất nhiều bộ ALU - Ít không gian cho bộ điều khiển logic và các bộ nhớ caches. - Nhiều thanh ghi để hỗ trợ nhiều luồng hoạt động. - Quản lý và lập kế hoạch cho các luồng trên phần cứng. - Băng thông bộ nhớ rất lớn để phục vụ hoạt động của nhiều ALU
Ưu nhược điểm	- Ưu điểm: rất nhanh và linh hoạt với một luồng. Phù hợp với các công việc có tính tuần tự cao. - Nhược điểm: công suất tính toán thấp hơn do thiết kế ưu tiên thời gian đáp ứng.	- Ưu điểm: công suất và thông lượng tính toán lớn. - Nhược điểm: xử lý một luồng đơn giản và không nhanh bằng CPU.

Vì những đặc điểm phần cứng nêu trên nên một phần giải thuật nhận dạng bằng phương pháp mô hình sử dụng giải thuật PSO được thực hiện trên GPU của máy tính. Các tính toán song song được thực hiện *giữa các phần tử* và *trong mỗi phần tử* của đàn, cụ thể tiến trình song song gồm 3 bước như sau:



Hình 29: Sơ đồ khối quy trình tính toán trên GPU

- *Bước 1:* Mỗi phần tử của đàn được cấp một vùng nhớ riêng trên GPU. Vị trí của mỗi phần tử chính là một tư thế *h* của bàn tay. Bằng thư viện đồ họa OpenGL [12] và mô hình bàn tay định nghĩa trong phần II.A, một ảnh mô hình 3 chiều của bàn tay được tạo ra với tư thế *h*. Bằng phép chiếu hình học với thông tin hướng nhìn *C* và thông số camera đã biết, ta tính được ảnh độ sâu *r* từ ảnh mô hình. Với 64 phần tử của đàn, ta tạo được 64 ảnh độ sâu

r_1, r_2, …, r_{64} để dùng cho bước tính hàm mục tiêu tiếp theo. Toàn bộ tiến trình trên và cả những tiến trình ở các bước tiếp theo được thực hiện đồng thời cho 64 phần tử trên 64 vùng nhớ riêng của GPU. Vì vậy, giải thuật bầy đàn được song song hóa giữa các phần tử.

- *Bước 2:* Bây giờ, với mỗi phần tử, ta cần tính giá trị sai khác $D(O,h,C)$ theo phương trình (4) để từ đó tính giá trị hàm mục tiêu $E(h,O)$ theo phương trình (5). Phương trình (4) yêu cầu cần phải thực hiện các phép tính $|o_d - r_d|$, $O_s \wedge r_m$, $O_s \vee r_m$ với từng điểm ảnh. Với độ phân giải 640x480 của Kinect, số điểm ảnh của vùng bàn tay khi đó là rất lớn và không phù hợp cho tính tuần tự. Song song hóa các phép tính này do đó cũng cần được thực hiện. Ý tưởng của tôi là áp dụng các phép toán xử lý logic trực tiếp giữa hai vùng nhớ thay vì truy vấn lần lượt mỗi ô nhớ để xử lý. Cụ thể, ảnh quan sát màu O_s và ảnh độ sâu O_d được chuyển từ CPU vào bộ nhớ của GPU. Các vùng nhớ này sau đó được sao chép ra 64 vùng tương ứng với số phần tử của đàn để tránh hiện tượng tranh đua tài nguyên giữa các luồng xử lý. Các phép tính khi đó được thực hiện cho đồng thời tất cả các điểm ảnh trong các vùng nhớ lưu O_s, O_d và r_i.
- *Bước 3:* Kết quả của mỗi phép tính ở bước 2 được lưu ở một vùng nhớ có kích thước bằng với vùng nhớ r_i. Để tính $D(O,h,C)$, ta còn cần phải tính tổng giữa các phần tử của từng vùng nhớ này (phương trình (4)). Để tận dụng triệt để khả năng song song của GPU, tôi tiếp tục sử dụng giải thuật tính tổng song song theo mô hình kim tự tháp [13] (Parallel Reduction). Kết quả là, với cấu hình GPU hỗ trợ tới 256 luồng xử ly song song, thanh ghi kích thước 128 bit, kiểu số thực 32 bit, có 1024 phép tính tổng sẽ được thực hiện đồng thời. Giá trị cuối cùng của $D(O,h,C)$ sau đó được chuyển sang bộ nhớ của CPU để tiếp tục các bước của giải thuật bầy đàn.

Như vậy, bằng cách sử dụng khối xử lý đồ họa GPU, giải thuật bầy đàn PSO đã được song song hóa hoàn toàn. Qua thử nghiệm của chúng tôi, phương pháp này đã giảm thời gian xử lý một khung hình 450 lần từ 6 phút xuống còn 0.8 giây và nhờ đó đảm bảo yếu tố thời gian thực.

Chương 5: MÔ PHỎNG VÀ THỰC NGHIỆM

Để đánh giá hiệu quả của phương pháp đề xuất, chúng tôi đã tiến hành mô phỏng với dữ liệu tổng hợp và thực nghiệm với dữ liệu thật. Hệ thống của chúng tôi được cài đặt trên máy tính xách tay có cấu hình

- CPU Intel i7 740qm,
- 8 GB RAM,
- GPU AMD HD5870m, 800 MADD core, năng lực xử lý 1.12 TFlops cùng 1 GB bộ nhớ.

Chương trình phần mềm được viết trên nền tảng Visual C++ 2010 [14] kết hợp với thư viện xử lý ảnh OpenCV 2.4.9 [15], thư viện đồ họa OpenGL 4.3 [12] và thư viện tính toán song song trên GPU OpenCL 1.2 [11]. Giải thuật bầy đàn được thực hiện với 64 phần tử tiến hóa qua 30 thế hệ trong đó một nửa số phần tử được đột biến cứ mỗi 3 thế hệ.

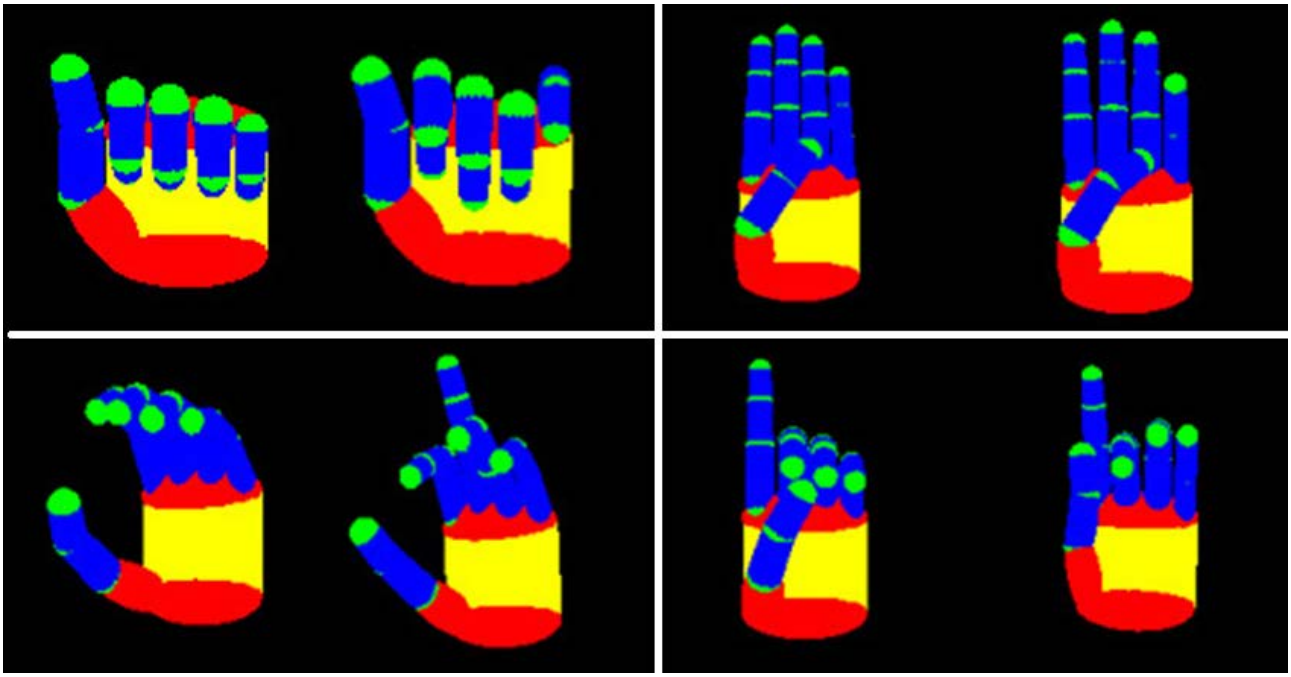
5.1 Mô phỏng

Mô phỏng được thực hiện với mục đích đánh giá và hoàn thiện giải thuật trước khi áp dụng với dữ liệu thực. Chi tiết như sau:

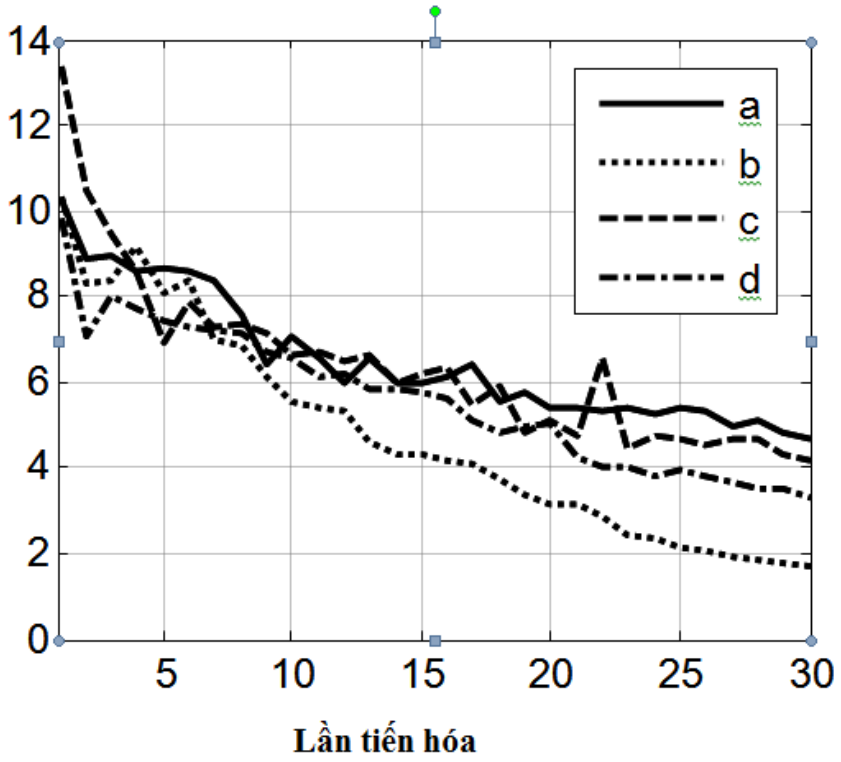
1) *Cài đặt mô phỏng*: Để thực hiện mô phỏng, một tư thế tay tùy chọn được đưa vào hệ thống để tạo ảnh mô hình h_{ref}. Giả thiết có hướng nhìn C, thông tin camera và mặt phẳng quan sát, khi đó ảnh màu và ảnh độ sâu được tạo ra từ ảnh mô hình bằng phép chiếu hình học. Các ảnh này được giả thiết như là ảnh quan sát màu O_s và ảnh độ sâu O_d thu được từ camera. Với các ảnh giả thiết này, giải thuật nhận dạng có thể thực hiện như với dữ liệu thực.

2) *Kết quả mô phỏng*: hình 30 trình kết quả nhận dạng 26 bậc tự do đối với các tư thế tay tương ứng với 4 chữ cái đầu của bảng chữ cái ngôn ngữ kí hiệu trong đó ảnh bên trái biểu diễn tư thế tay quan sát h_{ref} và ảnh bên phải biểu diễn tư thế tay nhận dạng h_{kq}. Có thể nhận thấy giải thuật đã xác định được chính xác 26 bậc tự do của bàn tay với một số tư thế. Trong một số tư thế khác, các bậc tự do gắn với các đốt ngón tay cho kết quả chưa chính xác do các phần tử của đàn bị tắc ở điểm tối ưu cục bộ.

Hình 31 trình bày sự thay đổi giá trị hàm mục tiêu qua từng bước tiến hóa. Có thể nhận thấy giá trị hàm mục tiêu giảm khi số bước tiến hóa tăng hay nói cách khác vị trí các phần tử của đàn tiến dần tới tư thế cần tìm. Trung bình, việc nhận dạng một tư thế được thực hiện trong 0.8 giây, trong đó 0,45 giây tiêu tốn cho việc xây dựng ảnh mô hình, ảnh quan sát, ảnh màu và 0,35 giây cho việc tính toán hàm mục tiêu E.



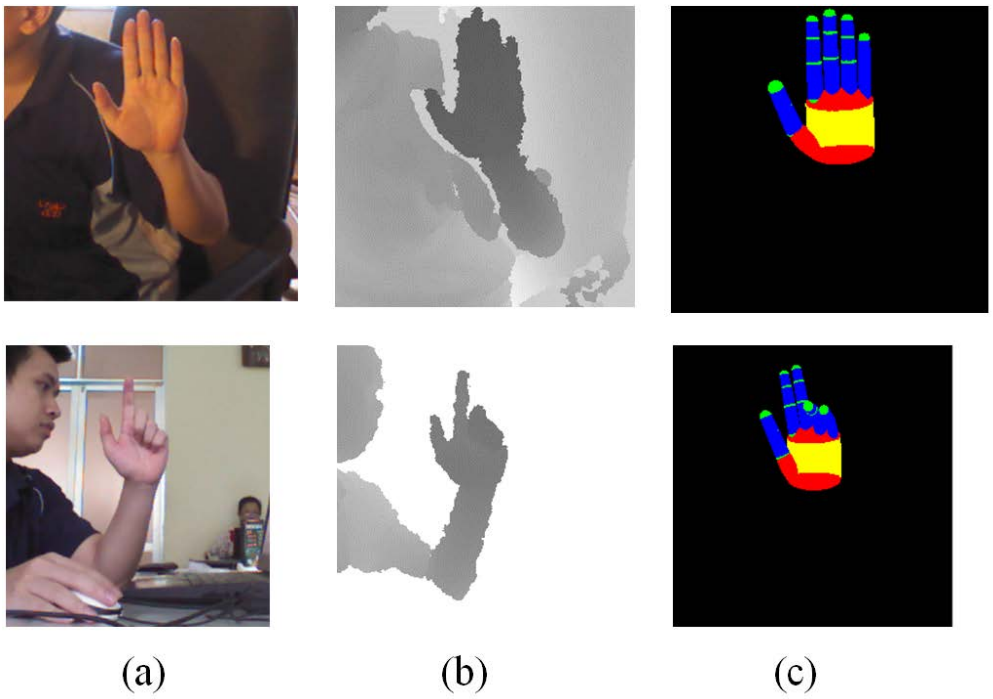
Hình 30: Kết quả nhận dạng 26 bậc tự do bàn tay với 4 tư thế trong đó bên trái là ảnh quan sát và bên phải là ảnh nhận dạng.



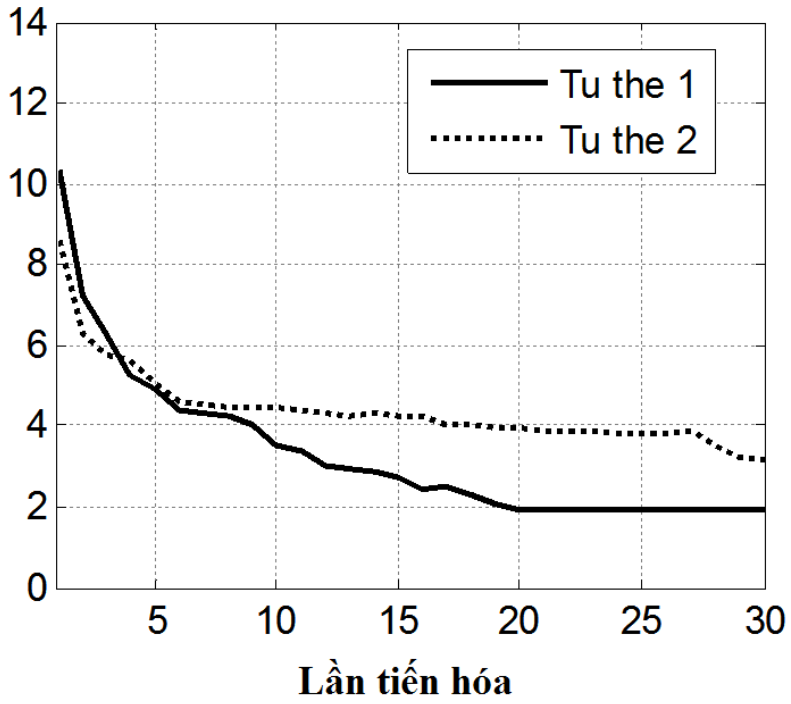
Hình 31: Biến thiên giá trị hàm mục tiêu theo bước tiến hóa với 4 tư thế ứng với các chữ cái "a", "b", "c", "d"

5.2 Thực nghiệm

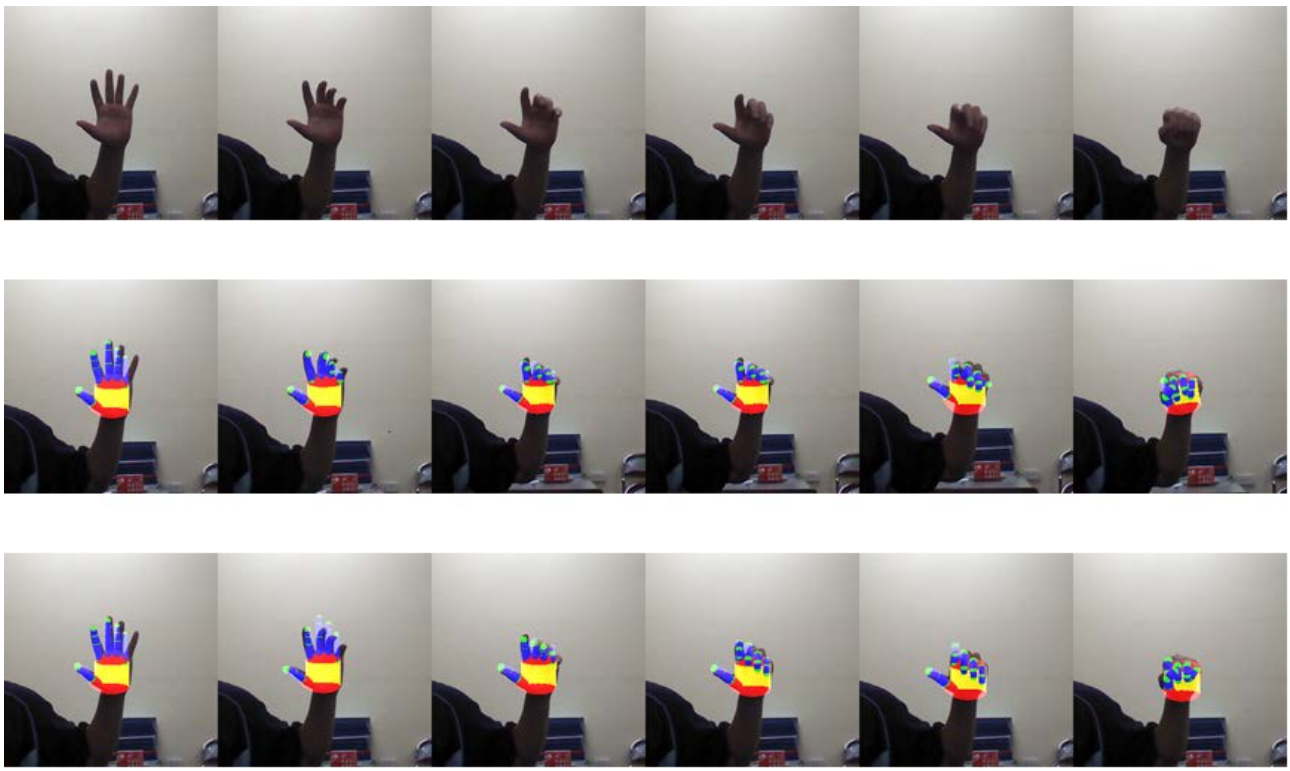
Trong thực nghiệm, ảnh quan sát là dữ liệu thu được từ cảm biến ảnh Kinect phiên bản 1.5 trong điều kiện trong nhà và ánh sáng ổn định. Vị trí bàn tay đặt cách cảm biến trong khoảng từ 0.5 *m* đến 1.5 *m*.



Hình 32: Kết quả thực nghiệm nhận dạng tư thế tay: (a) Ảnh quan sát màu; (b) Ảnh quan sát độ sâu; (c) Ảnh kết quả nhận dạng



Hình 33: Biến thiên giá trị hàm mục tiêu theo bước tiến hóa với 2 tư thế trong hình 32



Hình 34: Kết quả nhận dạng một chuỗi các ảnh chuyển động của bàn tay

Hình 32 trình bày kết quả nhận dạng. Trong cả hai tư thế, 6 bậc tự do của cổ tay và 4 bậc tự do của ngón cái được nhận dạng chính xác. Với bậc tự do của các ngón tay còn lại, chỉ các tư thế duỗi ngón được nhận dạng chính xác. Các tư thế gập ngón do một phần ngón tay bị che khuất nên thường dẫn tới kết quả không chính xác. Đồ thị giá trị hàm mục tiêu (hình 33) cho thấy, việc thay đổi tư thế các đốt ngón tay không dẫn tới sự thay đổi lớn của giá trị hàm mục tiêu và do đó không giúp các phần tử thoát khỏi đỉnh tối ưu cục bộ. Để khắc phục vấn đề này, cần thiết phải cải tiến hàm mục tiêu để nâng cao khả năng phân biệt. Việc này sẽ được thực hiện trong nghiên cứu tiếp theo.

Chương 6: KẾT LUẬN

6.1 Tổng kết

Kết quả chính của luận văn xây dựng thành công giải thuật nhận có thể nhận dạng được 26 bậc tự do của bàn tay theo phương pháp mô hình. Các công việc chính bao gồm từ khâu thu thập dữ liệu, xây dựng mô hình, xây dựng hàm mục tiêu tới triển khai giải thuật tối ưu bầy đàn. Việc tính toán song song giải thuật bầy đàn trên khối xử lý đồ họa GPU đã giúp cải thiện đáng kể hiệu năng xử lý của hệ thống mà không yêu cầu phần cứng bổ sung. Trong quá trình nghiên cứu tìm hiểu giải thuật nhận dạng, nhiều phép mô phỏng và thực nghiệm đã được tiến hành để khẳng định tính đúng đắn của phương pháp mô hình trong việc xử lý thông tin từ ảnh RGB-D.

Nghiên cứu trong luận văn được báo cáo trong Hội thảo quốc gia 2014 về Điện tử, Truyền thông và Công nghệ thông tin (REV-ECIT2014). Trên cơ sở các kết quả ban đầu của luận văn, đề xuất nghiên cứu phát triển ứng dụng đã được tập đoàn Samsung đã chấp nhận.

6.2 Hạn chế và hướng phát triển

Do thời gian có hạn, luận văn vẫn chưa xây dựng một hệ thống có ứng dụng hoàn chỉnh. Bên cạnh đó giải thuật nhận dạng còn tồn tại nhiều vấn đề như mới nhận diện được một bàn tay và thời gian xử lý còn cao v.v... Trong thời gian tiếp theo, tôi mong muốn hệ thống sẽ tiếp tục được cải tiến theo các hướng chính:

- Cải thiện phần tiền xử lý và trích chọn thêm đặc trưng của bàn tay để cải thiện khả năng đánh giá của hàm mục tiêu.
- Tăng cường phần cứng và tối ưu lập trình cho các phần của giải thuật song song để có thể xử lý được 30 hình/giây.
- Xây dựng quá trình hiệu chuẩn tự động bằng phương pháp tối ưu bầy đàn để giải thuật nhận dạng phù hợp với các mô hình kích thước bàn tay khác nhau.
- Nghiên cứu phát triển giải thuật để nhận dạng được chuyển động và tương tác của hai bàn tay.

Bằng việc kết hợp với các giải thuật khác như nhận dạng khuôn mặt, nhận dạng chuyển động cánh tay người, mạng neuron v.v… tôi hi vọng xây dựng được một hệ thống tương tác người máy có khả năng nhận diện được ngôn ngữ kí hiệu tiếng Việt trong tương lai không xa.

TÀI LIỆU THAM KHẢO

[1] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, X. Twombly, “Vision-based Hand Pose Estimation: A review”, J. Computer Vision and Image Understanding, vol.108(1-2), pp.52–73, 2007.

[2] I. Oikonomidis, N. Kyriazis, A. Argyros, “Efficient model-based 3D tracking of hand articulations using Kinect”, Proceedings of the British Machine Vision Conference, pp 101.1-101.11, 2011.

[3] H. Ouhaddi, P. Horain, “3D Hand gesture tracking by model registration”, International Workshop on Synthetic—Natural Hybrid Coding and Three Dimensional Imaging, 1999.

[4] B. Stenger, P.R.S. Mendonca, R. Cipolla, Model-based 3D tracking of an articulated hand, IEEE Computer Society Conference on Computer Vision and Pattern Recognition 02 (2001) 310.

[5] B. Stenger, Model-based hand tracking using a hierarchical bayesian filter, Ph.D. thesis, Department of Engineering, University of Cambridge, 2004.

[6] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, Numerical Recipes in C, Cambridge University Press, 1992.

[7] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization”, Computer Journal, vol. 7, 1965, pp. 308-313.

[8] David Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Professional, 1989.

[9] J. Kennedy, R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[10] Nicholas Wilt, The CUDA Handbook: A Comprehensive Guide to GPU Programming, Addison-Wesley Professional, 1 edition, 2013.

[11] Matthew Scarpino, OpenCL in Action: How to Accelerate Graphics and Computations, Manning Publications, 1 edition, 2011.

[12] Dave Shreiner, Graham Sellers, John M. Kessenich, Bill M. Licea-Kane, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Addison-Wesley Professional, 8 edition, 2013.

[13] L. Williams, Pyramidal parametrics, In ACM SIGGRAPH Computer Graphics, vol. 17, pp 1–11, 1983.

[14] Ivor Horton, Ivor Horton's Beginning Visual C++ 2010, Wrox; 1 edition, 2010.

[15] Samarth Brahmbhatt, Practical OpenCV (Technology in Action), Apress, 1 edition, 2013

[16] Jana Abhijit, Kinect for Windows SDK Programming Guide (Community Experience Distilled), Packt Publishing, 2012.