ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN SỸ ANH

NHẬN DIỆN CÁC DẠNG BỀ MẶT PHỤC VỤ PHÂN LOẠI VẬT THỂ SỬ DỤNG CAMERA RGB-D

LUẬN VĂN THẠC SĨ

CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ, TRUYỀN THÔNG

Hà Nội – 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN SỸ ANH

NHẬN DIỆN CÁC DẠNG BỀ MẶT PHỤC VỤ PHÂN LOẠI VẬT THỂ SỬ DỤNG CAMERA RGB-D

Ngành: Công nghệ Kỹ thuật Điện tử, Truyền thông
Chuyên ngành: Kỹ thuật Điện tử
Mã số : 60520203

LUẬN VĂN THẠC SĨ

CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ, TRUYỀN THÔNG

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. PHẠM MINH TRIỂN

Hà Nội – 2016

LỜI CAM ĐOAN

Tôi xin cam đoan nội dung của luận văn ***"nhận diện các dạng bề mặt phục vụ phân loại vật thể sử dụng camera RGB-D"*** là sản phẩm do tôi thực hiện dưới sự hướng dẫn của TS. Phạm Minh Triển. 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 14 tháng 12 năm 2016

TÁC GIẢ

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 TS. Phạm Minh Triển và ThS. Quách Công Hoàng, những người trực tiếp đã hướng dẫn, chỉ bảo tôi tận tình trong suốt quá trình nghiên cứu và hoàn thiện luận văn này.

Công trình này được tài trợ từ đề tài KHCN cấp ĐHQGHN, Mã số đề tài: QG.15.25.

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 14 tháng 12 năm 2016

TÁC GIẢ

Nguyễn Sỹ Anh

MỤC LỤC

DANH MỤC HÌNH VẼ VÀ ĐỒ THỊ

DANH MỤC BẢNG BIỂU

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Kí hiệu	Tiếng Anh	Tiếng Việt
RGB	Red Green Blue	Ảnh màu
RGB-D	Red Green Blue – Depth	Ảnh màu – độ sâu
VGA	Video Graphics Array	Chuẩn hiển thị hình ảnh 640x480
CAD	Computer-aided design	Thiết kế được hỗ trợ bởi máy tính
SVM	Support Véc tơ Machine	Máy véc tơ hỗ trợ
PFH	Point Feature Histogram	Lược đồ đặc trưng điểm
VFH	Viewpoint Feature Histogram	Lược đồ đặc trưng điểm nhìn
CVFH	Cluster Viewpoint Feature Histogram	Lược đồ đặc trưng điểm nhìn của nhóm điểm
GFPFH	Global Fast Point Feature Histogram	Lược đồ đặc trưng điểm nhanh toàn thể
RANSAC	Random Sample Consensus	Phương pháp đồng nhất mẫu ngẫu nhiên
PCA	Principal Component Analysis	Phép phân tích thành phần chính
RBF	Radial Basis Function	Hàm cơ sở bán kính
PCL	Point Cloud Library	Thư viện mở xử lý đám mây điểm

MỞ ĐẦU

Các robot với mục đích sử dụng hàng ngày trước đây thường sử dụng các loại cảm biến truyền thống như cảm biến hồng ngoại, cảm biến siêu âm. Nhược điểm của các loại cảm biến này là thông tin mang lại ít, gây ra nhiều giới hạn cho việc vận hành và khả năng của robot để thực hiện các tác vụ phức tạp.

Trong những năm gần đây, với sự đi lên của khoa học kĩ thuật, kĩ thuật xử lý ảnh và các hướng nghiên cứu về thị giác máy tính (Computer Vision) đang được phát triển, trong đó có thị giác máy tính sử dụng trong Robotics. Việc sử dụng camera giúp robot cảm nhận được môi trường xung quanh một cách đầy đủ và chi tiết hơn nhiều so với các loại cảm biến truyền thống. Từ đó tăng tính khả thi cho các bài toán phức tạp hơn về robot như khả năng nhận diện vật thể (màu sắc, hình khối), giúp phối hợp robot hoạt động bầy đàn (định vị, lập bản đồ, phối hợp tuần tra, tìm kiếm cứu nạn đa mục tiêu), …

Hiện nay, hướng nghiên cứu thị giác máy tính không chỉ tập trung vào việc xử lý ảnh màu 2D mà còn phát triển cả các công cụ xử lý ảnh 2.5D và 3D. Các loại camera 3D trên thị trường thường sử dụng kết hợp camera màu RGB truyền thống và camera độ sâu (Depth), cho đầu ra là ảnh kết hợp RGB-D.

Bài toán nhận diện và phân loại các bề mặt vật thể sử dụng camera RGB-D là một bước khởi đầu cho việc nghiên cứu thị giác máy tính theo xu hướng mới hiện nay. Mục tiêu của luận văn là xây dựng chương trình có khả năng phân biệt, nhận biết các bề mặt vật thể trong ảnh. Đề tài có thể được phát triển tiếp thành các ứng dụng trong lĩnh vực robot, hệ thống đa robot hay trong đời sống hàng ngày.

Chương 1: Giới thiệu

1.1. Giới thiệu về các ứng dụng của robot và đa robot

Với sự phát triển mạnh mẽ trong những năm gần đây, robot được kì vọng sẽ có vai trò ngày càng quan trọng trong sự phát triển của công nghệ và kĩ thuật. Robot thám hiểm có khả năng giúp ích con người nhiều hơn trong các công việc nguy hiểm như tìm kiếm cứu nạn hay làm việc trong các môi trường đặc biệt như môi trường độc hại. Robot công nghiệp có thể thay thế con người thực hiện các công nghiệp đơn điệu, lặp đi lặp lại trong nhà máy nhưng vẫn đảm bảo độ chính xác.

Hình 1.1: Robot turtle trong nhà được trang bị cảm biến Kinect

Robot trong nhà là một trong những chủ đề được nghiên cứu rộng rãi nhất. Ứng dụng của robot trong nhà có thể thấy trong cuộc sống hàng ngày như robot lau nhà trong các hộ gia đình thông thường, robot vận chuyển trong các kho bãi, robot phục vụ trong các bệnh viện … Nhu cầu về robot gia tăng đi kèm với sự phát triển của ngành robot theo nhiều hướng khác nhau. Những nghiên cứu về robot gần đây thường xoay quanh các chủ đề về thăm dò, khám phá những khu vực chưa biết, mô hình hóa môi trường, nhận diện vật thể và con người. Bên cạnh

đó, bài toán đa robot (nhiều robot phối hợp cùng thực hiện một nhiệm vụ) cũng được nghiên cứu rộng rãi.

Hình 1.2: Bài toán đa robot phối hợp thực hiện nhiệm vụ [11]

Vấn đề nhận biết môi trường xung quanh của robot là chủ đề nghiên cứu rộng rãi nhất hiện nay. Do các yêu cầu càng ngày càng cao trong việc thực hiện nhiệm vụ thì các cảm biến truyền thống dần dần không đáp ứng được nhu cầu của người phát triển. Robot ngày nay được trang bị nhiều thiết bị cảm nhận môi trường hiện đại hơn trong đó có camera RGB-D hay máy quét laser.

Các bài toán robot trong nhà với cảm biến ảnh nhờ đó có thể được cụ thể hóa thành các chủ đề nghiên cứu nhỏ hơn như xây dựng mô hình môi trường từ những hình ảnh thu thập được; định vị robot trong một môi trường đã biết trước; nhận diện, phân loại vật thể/con người trong môi trường xung quanh.

1.2. Camera RGB-D và đám mây điểm

Camera RGB-D là loại camera sử dụng đồng thời hai loại cảm biến: cảm biến ảnh màu thông thường như các loại camera truyền thống, cho ảnh đầu ra là ảnh RGB và một cảm biến độ sâu, cho ảnh đầu ra là ảnh độ sâu (Depth). Loại camera RGB-D phổ biến nhất trên thị trường là Kinect của Microsoft. Cảm biến độ sâu của Kinect sử dụng một cặp thu phát hồng ngoại.

Hình 1.3: Cảm biến Kinect

Bảng 1.1: Các thông số kĩ thuật của cảm biến Kinect

Độ phân giải ảnh màu	VGA (640x480)
Độ phân giải ảnh độ sâu	VGA (640x480)
Thị trường	43^o theo chiều dọc 57^o theo chiều ngang
Tốc độ ghi hình	30 khung hình/giây

Ảnh màu RGB và ảnh độ sâu Depth trên Kinect qua các bước xử lý tạo ra dữ liệu 3D dưới dạng point cloud (đám mây điểm). Đám mây điểm là một bộ các điểm trong không gian ba chiều, mỗi điểm bao gồm tọa độ XYZ của nó. Ngoài ra, mỗi điểm cũng có thể chứa thêm thông tin về màu.

Nói chung, đám mây điểm là kiểu dữ liệu thu được từ các thiết bị quét 3D. Các thiết bị này cảm nhận bề mặt các vật thể theo nguyên tắc phát ra một chùm sóng điện từ (hồng ngoại hoặc laser) và thu về sóng phản xạ. Kết quả của quá trình đo từ máy quét là tập dữ liệu gồm bộ các điểm thu được, dưới dạng đám mây điểm. Cảm biến RGB-D cũng là một dạng máy quét 3D khi sử dụng cảm biến độ sâu theo nguyên lý quét và kết hợp với cảm biến màu. Ngoài ra, dữ liệu kiểu đám mây điểm cũng có thể được tạo ra từ các mô hình 3D như mô hình CAD.

Dữ liệu kiểu đám mây điểm được sử dụng trong robot và đa robot với các cảm biến RGB-D, hay ngành viễn thám với các thiết bị quét 3D địa hình bằng máy quét gắn trên máy bay không người lái.

Hình 1.4: Ảnh đầu ra của Kinect (nguồn: internet)

ảnh độ sâu (phía trên bên trái), ảnh màu RGB (phía dưới bên trái) và đám mây điểm kết hợp ảnh độ sâu và RGB (bên phải)

1.3. Mục tiêu và đối tượng nghiên cứu

Mục tiêu của luận văn này là nhận diện các dạng bề mặt khác nhau trong đám mây điểm, với mục đích phân loại vật thể, phục vụ cho các ứng dụng về robot trong nhà. Bài toán nhận dạng và phân loại vật thể trên đám mây điểm đã được nghiên cứu và phát triển trong nhiều năm, với nhiều cách tiếp cận khác nhau. Một trong những phương pháp tiếp cận phổ biến và rõ ràng nhất là trích xuất các đặc trưng (feature) của đối tượng và sau đó dùng phương pháp máy véc tơ hỗ trợ (Support Véc tơ Machine – SVM) để nhận diện đối tượng.

Nội dung của luận văn này là giới thiệu phương pháp trích xuất lược đồ đặc trưng điểm (Point Feature Histogram) và sau đó sử dụng SVM để nhận diện bề mặt của điểm. Nội dung chính của các chương được trình bày như sau:

Chương 2: Nói về các kĩ thuật xử lý đám mây điểm, gồm có tiền xử lý và tính toán đặc trưng điểm. Tiền xử lý gồm có giảm mẫu (downsample), loại bỏ các điểm nhiễu không liên quan, phân đoạn và ghép nhóm. Mục đích của quá trình này là lọc đi những dữ liệu thừa, giảm dung lượng dữ liệu cần xử lý nhằm giảm thời gian tính toán cho các bước sau. Các đặc trưng điểm được sử dụng bao gồm véc tơ pháp tuyến và lược đồ đặc trưng điểm – là đặc trưng cần thiết để xác định bề mặt vật thể.

Chương 3: Khái niệm và phương pháp xây dựng mô hình học máy SVM, cũng như cách thức dùng mô hình SVM để nhận diện, phân loại đặc trưng vật thể.

Chương 4: Chương trình và thực nghiệm. Chương trình được thử nghiệm trên dữ liệu sạch (noiseless) và dữ liệu thật chụp bằng cảm biến Kinect. Phân tích và đánh giá hiệu năng khi thay đổi các tham số của giải thuật.

Chương 5: Kết luận và đánh giá, đồng thời đề xuất các hướng phát triển tiếp theo của đề tài.

1.4. Các nghiên cứu liên quan

Các phương pháp trích xuất đặc trưng của đối tượng từ đám mây điểm đã được nghiên cứu rộng rãi trong nhiều năm. Trong đó, hai đặc trưng về mặt hình học của các điểm trong đám mây điểm được sử dụng nhiều nhất là ước lượng pháp tuyến (normal estimation) và ước lượng độ cong (curvature estimation). Đây đều là những đặc trưng mang tính cục bộ bởi nó mô tả thông tin về môi trường (hay các điểm) xung quanh điểm cần khảo sát. Các đặc trưng mang tính cục bộ này sử dụng phương pháp khảo sát thông qua các điểm lân cận. Đặc điểm chung của các đặc trưng cục bộ là chúng dễ bị ảnh hưởng bởi nhiễu đến từ cảm biến.

Trái với các đặc trưng điểm mang tính cục bộ chỉ mô tả mối liên hệ giữa một điểm và các lân cận của nó, các đặc trưng điểm mang tính toàn thể mô tả đặc trưng của cả một nhóm điểm lớn biểu diễn một vật thể và có thể dùng trong các bài toán phân loại, nhận dạng vật thể. Một loại đặc trưng toàn thể liên quan là Viewpoint Feature Histogram (VFH) [12]. Đây là đặc trưng toàn thể có liên quan đến Fast Point Feature Histogram (FPFH) [14]. Với đặc trưng này, các góc sai lệch được tính dựa trên véc tơ pháp tuyến của điểm và véc tơ pháp tuyến của tâm

đám mây điểm. Điều đó khiến cho histogram trở nên hữu ích cho việc nhận diện vật thể và ước lượng tư thế.

Một giải thuật mở rộng của VFH là Cluster Viewpoint Feature Histogram (CVFH) được trình bày trong [13]. Giải thuật này dựa trên ý tưởng rằng mỗi vật thể đều có một cấu trúc nhất định cho phép chia vật thể đó ra thành một số N vùng mịn riêng biệt. Mỗi vùng đó lại được sử dụng độc lập để tính ra một bộ N histogram VFH riêng biệt.

Global Fast Point Feature Histogram (GFPFH) được trình bày trong [15] là giải thuật tổng quát hóa FPFH ở cấp độ toàn thể để tạo ra một đặc trưng điểm có thể bao gồm mối liên hệ của các phần hình học cục bộ của các vật thể.

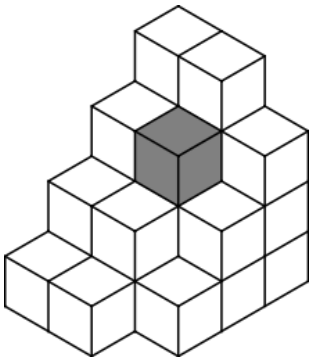
Chương 2: Các kỹ thuật xử lý đám mây điểm

2.1. Tiền xử lý

Việc lưu trữ và xử lý một đám mây điểm lớn với hàng trăm ngàn điểm dưới dạng các điểm trong không gian ba chiều là một tác vụ rất tiêu tốn tài nguyên phần cứng và cũng là nguyên nhân chính dẫn đến tình trạng thắt cổ chai trong các hệ thống. Trong khi đó, trong tập dữ liệu đám mây điểm, chúng ta chỉ cần các dữ liệu liên quan đến các vật thể cần xác định bề mặt. Chương này sẽ trình bày các kỹ thuật tiền xử lý đám mây điểm, qua đó chúng ta có thể giảm số lượng điểm cần tính toán đi nhiều lần mà vẫn giữ được các đặc tính hình học cần thiết.

2.1.1. Giảm mẫu

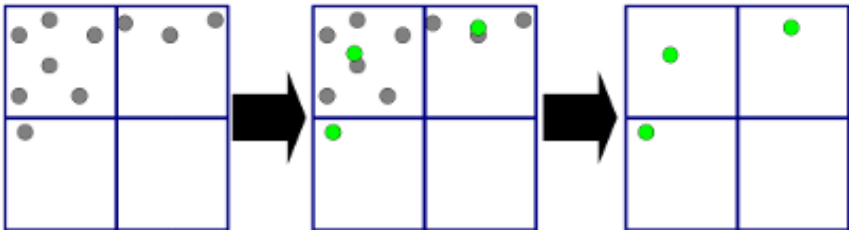
Giảm mẫu có mục đích giảm số lượng các điểm trong một đám mây điểm mà không làm mất các đặc trưng trong đám mây điểm. Một đám mây điểm sau khi giảm mẫu sẽ có số điểm ít hơn so với ban đầu, giúp giảm khối lượng tính toán cho các bước tiếp theo. Phương pháp giảm mẫu được dùng ở đây sử dụng bộ lọc lưới voxel (voxel grid filter) [7].



Hình 2.1. Voxel grid trong không gian ba chiều

Mỗi voxel là một hình hộp, biểu diễn một giá trị điểm trong không gian. Khái niệm voxel trong không gian ba chiều cũng giống như khái niệm điểm ảnh (pixel) trong mặt phẳng hai chiều. Voxel có thể được sử dụng như một cách biểu diễn đồ họa 3D, hay trong cách biểu diễn dữ liệu theo kiểu cây octree. Thông thường khi biểu diễn đồ họa bằng voxel thì mỗi voxel sẽ đại diện cho một điểm, tương đương với tọa độ và màu của điểm đó.

Bộ lọc lưới voxel là phương pháp giảm mẫu bằng cách đưa đám mây điểm vào trong một không gian gồm các lưới voxel, với kích thước của lưới lớn hơn so với khoảng cách giữa một điểm và điểm gần nó nhất. Nói cách khác, độ phân giải của đám mây điểm phải lớn hơn độ phân giải của lưới voxel. Sau đó với mỗi voxel, giải thuật sẽ tính toán tâm trung bình của các điểm bên trong và thay thế các điểm này bằng duy nhất một điểm tại tâm trung bình.

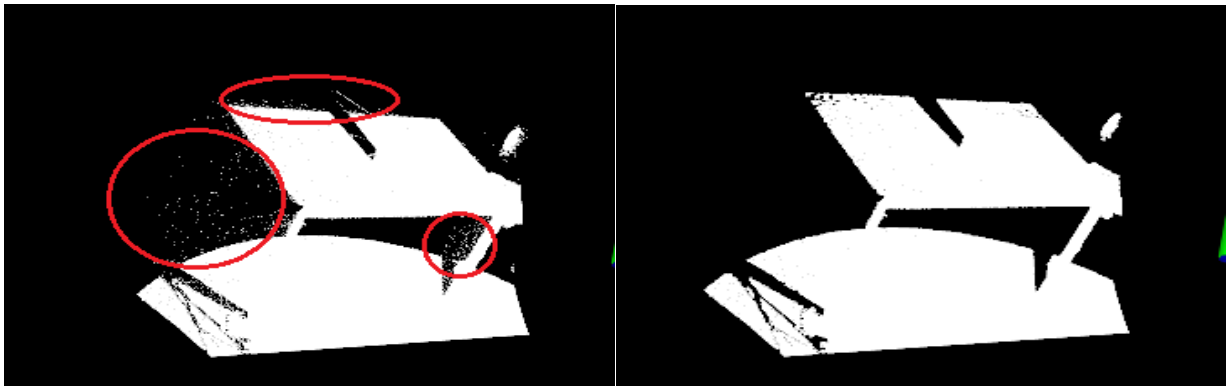


Hình 2.2: thay thế các điểm trong mỗi voxel bằng điểm trung bình

Phương pháp này có thời gian thực hiện lâu hơn so với phương pháp đơn giản là thay thế bằng điểm nằm ở trung tâm mỗi voxel. Tuy nhiên nó có thể giảm mẫu một đám mây điểm mà vẫn giữ lại nhiều hơn các đặc tính hình học. Một ưu điểm khác của phương pháp này là trong một số trường hợp, nó còn giúp giảm nhiễu ngẫu nhiên trên tập dữ liệu đầu vào nhờ vào nguyên tắc lấy trung bình các mẫu.

2.1.2. Loại bỏ những điểm không liên quan

Khi sử dụng các thiết bị quét 3D như camera RGB-D hoặc máy quét laser, trong dữ liệu thu được thường bị xuất hiện những điểm nằm lơ lửng, không nằm trong mặt phẳng nào. Những điểm này thường xuất hiện gần nơi giao tiếp giữa các bề mặt, hay gần các cạnh của vật thể. Khi tính toán các đặc trưng của đám mây điểm, các điểm nhiễu này có thể gây ra sai lệch vì lý do thực ra chúng không tồn tại mà chỉ là nhiễu tác động lên cảm biến. Hơn nữa, việc gia tăng số điểm trên đám mây điểm còn làm tăng thời gian tính toán. Do đó trước khi tính toán các đặc trưng, ta cần loại bỏ những điểm nhiễu không nằm trên các bề mặt này.



Hình 2.3: Trước và sau khi loại bỏ các điểm nhiễu

Phương pháp được sử dụng rộng rãi nhất để loại bỏ các điểm nhiễu ngoài bề mặt này là phương pháp phân tích thống kê. Đây là phương pháp tính toán khoảng cách trung bình từ một điểm đến các lân cận của nó để xác định các điểm không nằm trong bề mặt. Quá trình thực hiện phương pháp này như sau:

- Đầu tiên, với mỗi điểm p_q trong đám mây điểm, xác định k điểm lân cận của nó và tính giá trị khoảng cách trung bình d từ p_q đến các điểm lân cận đó. Quá trình này được thực hiện với tất cả các điểm trong đám mây điểm.
- Thiết lập một phân phối theo khoảng cách trung bình từ mỗi điểm đến các điểm lân cận. Từ phân phối đó, tính toán ra các giá trị trung bình μ_k và độ lệch chuẩn σ_k của phân phối.
- Những điểm có khoảng cách trung bình d đến k điểm gần nhất cao hơn một giá trị $\alpha\sigma_k$ sẽ được coi là điểm nằm ngoài bề mặt và bị loại bỏ khỏi đám mây điểm.

2.1.3. Phân đoạn và ghép nhóm

Phần này sẽ trình bày hai phương pháp xử lý các đám mây điểm lớn với mục đích giảm khối lượng tính toán cho các bước tính toán sau. Hai phương pháp này là phân đoạn (segmentation) và ghép nhóm (clustering).

Phân đoạn là quá trình ghép các điểm trong một đám mây điểm vào một mô hình hình học đơn giản như mặt phẳng, mặt trụ, mặt cầu, … sao cho các điểm trong đám mây điểm có khoảng cách đến mô hình nằm trong khoảng cho phép. Các điểm thuộc mô hình sau đó sẽ được đánh dấu để từ đó có thể thay thế các điểm bằng một mô hình đơn giản. Quá trình này có tác dụng đơn giản hóa dữ liệu đám mây điểm, giúp nâng cao hiệu quả xử lý của hệ thống.

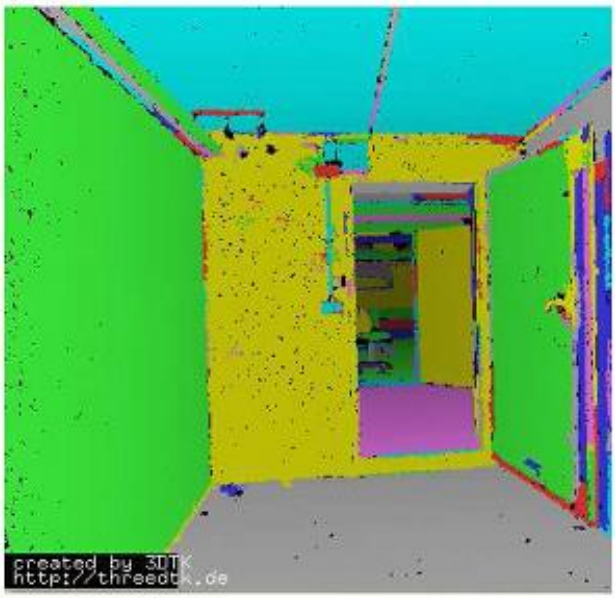
Ghép nhóm là phương pháp phân chia các điểm trong một đám mây điểm thành các nhóm nhỏ, qua đó giảm đáng kể thời gian để xử lý toàn bộ lượng dữ liệu ban đầu.

a. Phân đoạn

Các phương pháp phân đoạn dữ liệu đám mây điểm là chủ đề đã được nghiên cứu trong thời gian dài. Một cảnh ảnh *P* được thu thập từ cảm biến RGB-D sẽ thể hiện các vật thể được quét qua dưới dạng đám mây điểm. Trong điều kiện lý tưởng, với các mô hình vật thể đều đã có trong cơ sở dữ liệu thì tập dữ liệu *P* sau khi thu thập từ cảm biến sẽ có thể được đơn giản hóa đi đáng kể: Các điểm trong *P* thể hiện một mô hình (hay một phần của mô hình) sẽ được thay thế bởi mô hình đó với các thông số thể hiện vị trí, tư thế (hay góc nhìn) và kích cỡ thực tế. Điều này được thực hiện với tất cả các điểm trong tập dữ liệu *P* và sau đó dẫn đến một kết quả rằng thay vì lưu trữ một đám mây điểm *P* thì ta chỉ cần lưu trữ một bộ các thông số thể hiện những mô hình xuất hiện trong *P* với vị trí, góc nhìn và kích cỡ của chúng.

Tuy nhiên một cảnh ảnh quét từ cảm biến (2.5D hoặc 3D) bao giờ cũng xuất hiện nhiễu. Nhiễu lượng tử tác động lên cảm biến khiến cho từng điểm bị lệch đi so với giá trị thực tế của chúng một khoảng σ dao động tùy theo các loại cảm biến, khoảng cách từ vật đến cảm biến. Hơn nữa, với những khung cảnh phức tạp, chứa nhiều đồ vật thì việc các đồ vật che khuất nhau hay gây ra các hiện tượng vật lý như phản xạ, tán xạ ánh sáng làm cho việc kết hợp ảnh RGB và ảnh Depth được thực hiện không chính xác tại một số nơi. Các khung cảnh phức

tạp còn khiến cho đa số các vật thể không xuất hiện đầy đủ dưới cảm biến mà chỉ xuất hiện một phần, một phía.



Hình 2.4: Ví dụ về phân đoạn trong đám mây điểm
Các mặt phẳng được đánh dấu bằng màu khác nhau

Kỹ thuật phổ biến nhất trong việc phân đoạn đám mây điểm là đơn giản hóa dữ liệu bằng việc thay thế các điểm bằng các hình 3D cơ bản như mặt phẳng, mặt trụ, mặt cầu, mặt nón, hoặc thậm chí là các hình 3D với các đa thức bậc cao.

Việc thay thế (hay xấp xỉ) các điểm trong đám mây điểm bằng các bề mặt hình học 3D đơn giản là khả thi trong đa số các trường hợp thực tế. Khi sử dụng cảm biến RGB-D quét các khung cảnh trong nhà hay ngoài trời, ta có thể thấy rằng đa số những vật thể xuất hiện đều là sự kết hợp của các hình 3D cơ bản: bức tường hay sàn nhà, mặt tủ đều là các mặt phẳng; chướng ngại vật hay đồ đạc trong nhà lại là các mặt trụ, mặt nón hay mặt cầu kết hợp…

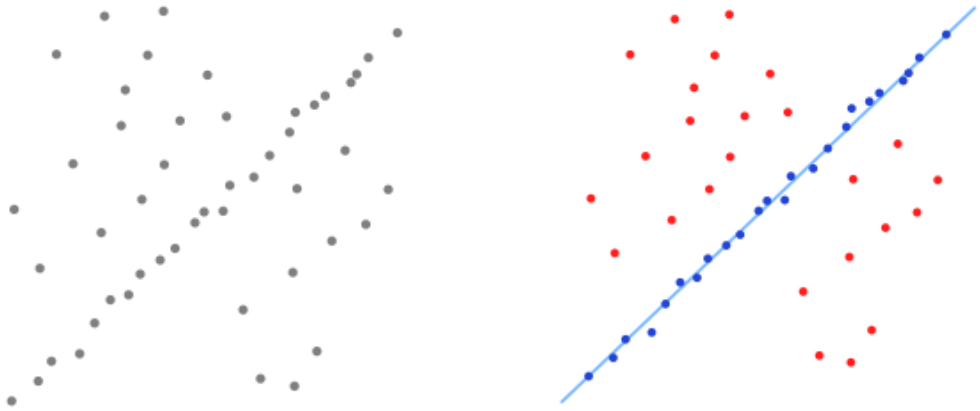
Chi tiết hơn về tác dụng của quá trình thay thế này cũng có thể được hiểu trong các trường hợp cụ thể. Ví dụ như xét bài toán robot di chuyển trong nhà, tránh vật cản và tính toán đường đi hợp lý. Nếu môi trường xung quanh được cung cấp dưới dạng đám mây điểm với hàng ngàn điểm, thì việc tính toán khoảng cách đến từng điểm trong số này là công việc nặng và tiêu tốn tài nguyên. Tuy nhiên khi đơn giản hóa môi trường bằng các mô hình với các hình khối 3D thì việc tính toán khoảng cách và xác định vị trí robot trở nên đơn giản hơn nhiều với các phép tính khoảng cách từ điểm đến các bề mặt, với số bề mặt được giới hạn.

Phương pháp phân đoạn được trình bày ở đây là phương pháp RANSAC (Random Sample Consensus). Thuật toán RANSAC là thuật toán lặp với mục đích ước lượng các thông số của một mô hình toán học từ một bộ dữ liệu thu thập được bao gồm cả các điểm trong và ngoài. Các điểm trong là những điểm trong bộ dữ liệu phù hợp (hay nằm trong) mô hình toán học đó, còn điểm ngoài là những điểm không nằm trong mô hình.

Thuật toán RANSAC được công bố lần đầu bởi Fischler và Bolles [9], với nguyên tắc ước lượng các thông số bằng cách chọn ngẫu nhiên các mẫu trong dữ liệu thu thập được. Với tập dữ liệu đầu vào bao gồm cả các điểm trong và ngoài mô hình, thuật toán RANSAC sử dụng nhiều lần thử để tìm ra mô hình có kết quả tốt nhất. Về cơ bản, thuật toán RANSAC lặp đi lặp lại hai quá trình:

- Chọn ngẫu nhiên một số lượng tối thiểu dữ liệu từ đầu vào (điểm mẫu), sau đó tính toán các thông số của mô hình chứa các điểm mẫu này.
- Thuật toán kiểm tra tất cả các điểm còn lại xem chúng nằm trong hay nằm ngoài mô hình. Một điểm được coi là nằm trong mô hình nếu sai số của nó so với mô hình nhỏ hơn sai số qui định.

Thuật toán RANSAC lặp lại quá trình trên cho đến khi bộ các điểm trong đủ lớn hoặc đạt giá trị lớn nhất sau một số lần lặp lại cho trước.



Hình 2.5: Thuật toán RANSAC ước lượng mô hình đường thẳng.

(a): Dữ liệu đầu vào

(b): Mô hình được ước lượng với các điểm màu xanh là điểm trong và màu đỏ là điểm ngoài

Thuật toán RANSAC có đầu vào là bộ dữ liệu thu thập được, phương pháp để khớp dữ liệu thu thập vào mô hình và các thông số tin cậy (như sai số cho phép, số lần lặp tối đa). Quá trình thực hiện của RANSAC như sau:

- Chọn một bộ nhỏ nhất các điểm mẫu bất kì từ dữ liệu đầu vào, giả thiết các điểm này đều là điểm trong.
- Tính toán các thông số của mô hình chứa các điểm đó.
- Kiểm tra tất cả các điểm còn lại trong dữ liệu đầu vào xem chúng nằm trong hay nằm ngoài mô hình dựa vào mô hình vừa tính được và sai số cho phép.
- Mô hình vừa xấp xỉ là tốt nếu số điểm trong thỏa mãn yêu cầu.
- Quá trình này được lặp lại để phát hiện các mô hình tốt hơn.

Các tham số tin cậy quan trọng trong việc thực hiện thuật toán RANSAC là ngưỡng sai số cho phép ε và số lần lặp lại N. Ngưỡng sai số cho phép ε là giá trị khoảng cách lớn nhất từ một điểm đến mô hình để điểm đó còn được coi là điểm trong. Việc xác định ε với từng trường hợp cụ thể thường dựa trên kinh nghiệm và ước lượng thực tế.

Số lần lặp lại N cũng có thể được ước lượng hợp lý để cân bằng giữa kết quả và thời gian tính toán. Số lần lặp N có thể được tính bằng phương pháp thống kê.

Giả sử p là xác suất thành công của thuật toán (thông thường $p = 0.99$). Gọi u là xác suất để một điểm trong tập dữ liệu thu được là điểm trong, v là xác suất để điểm đó là điểm ngoài, ta có $v = 1 - u$.

Gọi m là số điểm trong để mô hình thu được là tốt, khi đó xác suất để thu được một mô hình tốt là u^m.

Trong mỗi lần thử, thuật toán thất bại khi không tìm được mô hình nào đủ tốt, tức là không có mô hình nào có đủ m điểm trong. Xác suất để điều này xảy ra là $1 - u^m$.

Sau N lần lặp lại, xác suất thất bại là:

$$(1 - u^m)^N = 1 - p \qquad (2.1)$$

Qua đó ta có thể tính được số lần lặp lại tối ưu với xác suất thành công p:

$$N = \frac{\log(1-p)}{\log(1-u^m)} \quad (2.2)$$

Ưu điểm của thuật toán RANSAC là độ bền vững, nói cách khác là RANSAC có thể tìm ra mô hình thích hợp với độ chính xác rất cao trong bộ dữ liệu thu thập được mặc dù bộ dữ liệu đó chứa nhiều điểm ngoài.

Nhược điểm của RANSAC là ở chỗ không có giới hạn về thời gian tính toán các thông số mô hình. Khi số lần lặp lại bị giới hạn, giải thuật có thể không đưa ra được mô hình tối ưu. Do đó khi thực hiện RANSAC, người sử dụng phải cân bằng giữa thời gian tính toán và chất lượng mô hình ước lượng được. Với số lần lặp lại càng tăng, xác suất để RANSAC đưa ra các mô hình tốt hơn cũng tăng.

Một nhược điểm khác của RANSAC là giải thuật chỉ có thể ước lượng được một mô hình với mỗi bộ dữ liệu đầu vào. Nếu tập dữ liệu đầu vào chứa nhiều mô hình cần tìm (ví dụ như một đám mây điểm chứa hai hay nhiều mặt phẳng), RANSAC chỉ có thể tìm ra mặt phẳng lớn nhất, hay mặt phẳng chứa nhiều điểm nhất.

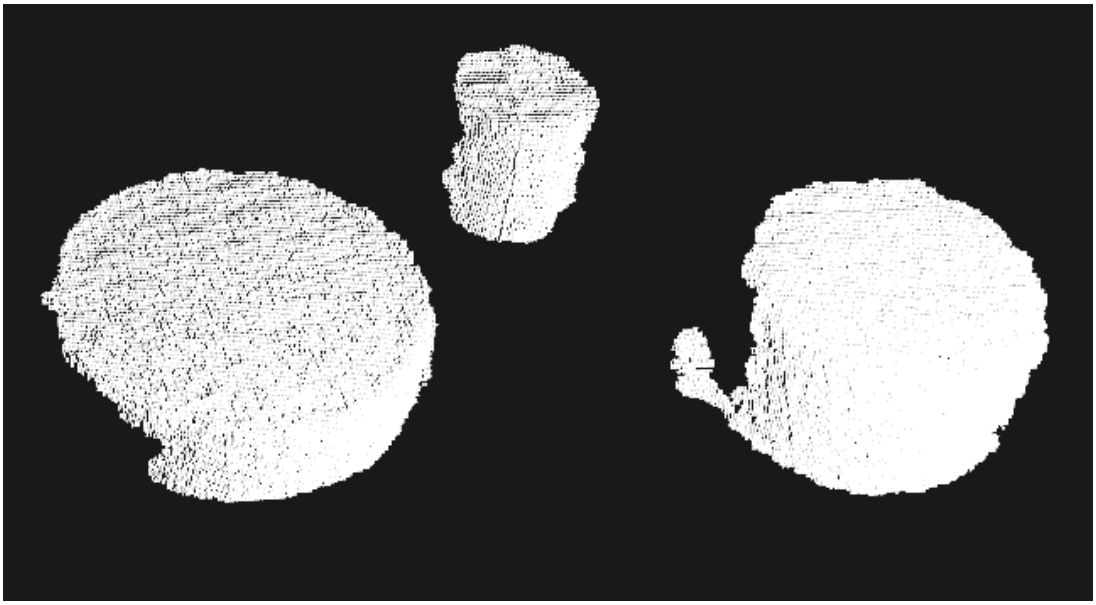
b. Ghép nhóm

Ghép nhóm là phương pháp phân chia các điểm trong một đám mây điểm thành các nhóm nhỏ, qua đó giảm đáng kể thời gian để xử lý toàn bộ lượng dữ liệu ban đầu. Các phương pháp ghép nhóm đơn giản sử dụng cách tìm các đường biên hay so sánh về khoảng cách đến các điểm lân cận để nhóm các điểm gần nhau lại với nhau.

Giả sử trong tập dữ liệu $\boldsymbol{P} = \{p_1, p_2, \dots, p_n\}$, hai nhóm $\boldsymbol{O_i} = \{p_i \in P\}$ và $\boldsymbol{O_j} = \{p_j \in P\}$ là hai nhóm riêng biệt nếu:

$$min\|p_i - p_j\| \geq \varepsilon$$

Với ε là ngưỡng khoảng cách giới hạn. Nói cách khác, nếu khoảng cách nhỏ nhất giữa hai tập các điểm $\boldsymbol{O_i} = \{p_i \in \boldsymbol{P}\}$ và $\boldsymbol{O_j} = \{p_j \in \boldsymbol{P}\}$ lớn hơn một ngưỡng giới hạn cho trước, thì khi đó $\boldsymbol{O_i}$ và $\boldsymbol{O_j}$ là hai nhóm khác nhau.



Hình 2.6: Các cụm điểm thành nhóm riêng biệt

Ví dụ về việc ghép nhóm trong một đám mây điểm là trong bài toán xác định các vật thể đặt trên bàn. Khi đó đám mây điểm đầu vào sẽ bao gồm một mặt phẳng lớn – là mặt bàn – và các nhóm điểm biểu diễn cho các vật nằm trên mặt bàn đó. Sau khi tách được mặt bàn thì đám mây điểm chỉ còn bao gồm các nhóm điểm riêng biệt rõ ràng, mỗi nhóm là dữ liệu quét của một vật thể.

Trong bài toán như trên, phương pháp ghép nhóm đơn giản nhất là phương pháp sử dụng khoảng cách vật lý và giải thuật tìm kiếm điểm lân cận. Các bước để thực hiện giải thuật này như sau:

- Với dữ liệu đầu vào là đám mây điểm $\boldsymbol{P}$, tạo cây kd-tree biểu diễn dữ liệu để thuận lợi cho việc tìm kiếm lân cận.
- Tạo ra một danh sách nhóm $\boldsymbol{C}$, và các điểm cần được khảo sát $\boldsymbol{Q}$. Ban đầu, các điểm cần được khảo sát là toàn bộ dữ liệu đầu vào.
- Với mỗi điểm $p_i \in \boldsymbol{P}$, thực hiện các bước sau:
    - Thêm p_i vào danh sách các điểm cần khảo sát $\boldsymbol{Q}$.
    - Với mỗi $p_i \in \boldsymbol{Q}$, tìm kiếm các điểm lân cận của p_i trong bán kính $r = \varepsilon$. Sau đó kiểm tra các điểm lân cận này đã được xử lý hay chưa, nếu chưa được xử lý thì thêm điểm đó vào $\boldsymbol{Q}$.
    - Khi tất cả các điểm trong $\boldsymbol{Q}$ đã được xử lý, điều đó có nghĩa là không còn điểm nào trong $\boldsymbol{P}$ có khoảng cách đến $\boldsymbol{Q}$ nhỏ hơn ε. $\boldsymbol{Q}$ được coi là một nhóm.

- Quá trình trên được lặp lại cho đến khi tất cả các điểm đều thuộc một nhóm nào đó.

2.2. Tính toán đặc trưng điểm

Các đặc trưng hình học trong đám mây điểm có thể được chia làm hai dạng: đặc trưng mang tính cục bộ (local feature) hoặc đặc trưng mang tính toàn thể (global feature). Các loại đặc trưng toàn thể đại diện cho một nhóm điểm, thường được sử dụng cho các bài toán phân loại, nhận diện vật thể. Các đặc trưng cục bộ thường được mô tả bằng mối quan hệ với các môi trường xung quanh một điểm. Chương này sẽ trình bày một số khía cạnh liên quan đến các đặc trưng điểm cục bộ và phương pháp xác định lược đồ đặc trưng điểm PFH, vai trò của PFH trong bài toán xác định bề mặt hình học.

2.2.1. Các điểm lân cận

Các điểm lân cận là một khái niệm cơ bản trong các quá trình xử lý đám mây điểm. Việc xác định các điểm lân cận không chỉ là tiền đề mà còn có thể quyết định đến chất lượng, độ chính xác, thời gian thực hiện của các thuật toán xử lý đặc trưng điểm về sau, qua đó ảnh hưởng đến toàn hệ thống.

Khái niệm các điểm lân cận được xác định bằng khoảng cách giữa điểm cần xem xét đến các điểm xung quanh nó. Giả sử điểm cần xem xét là p_q, và $\boldsymbol{P^k} = \{p_1^k, \dots, p_n^k\}$ là tập hợp *n* điểm xung quanh p_q. Khi đó, một điểm p_i^k là lân cận củap_q nếu

$$\left\| p_i^k - p_q \right\| \leq d \qquad (2.1)$$

Trong đó, d là giá trị độ dài lớn nhất có thể để xác định các điểm lân cận của một điểm, còn vế trái là khoảng cách Euclid giữa hai điểm $p_q$và p_i^k. Tập hợp các điểm p_i^k thỏa mãn điều kiện trên là các điểm lân cận của p_q. Các đặc trưng điểm trong đám mây điểm sau đó sẽ được mô tả bằng một hàm véc tơ $\boldsymbol{F}$, mô tả các thông tin về đặc trưng của điểm $\mathrm{p_q}$theo $\boldsymbol{P^k}$:

$$\boldsymbol{F}\left(p_q, \boldsymbol{P^k}\right) = (x_1, x_2, \dots, x_n)^T \qquad (2.2)$$

Với $(x_1, x_2, \dots, x_n)^T$ là một véc tơ *i* chiều, biểu diễn đặc trưng điểm của p_q.

Trong thực tế sử dụng, có hai phương pháp khác nhau để xác định các điểm lân cận của một điểm cần xem xét, đó là:

- Xác định bằng k điểm lân cận gần điểm cần xem xét nhất (tìm kiếm theo k);
- Xác định bằng tất cả k điểm lân cận trong một bán kính r tính từ điểm cần xem xét (tìm kiếm theo bán kính).

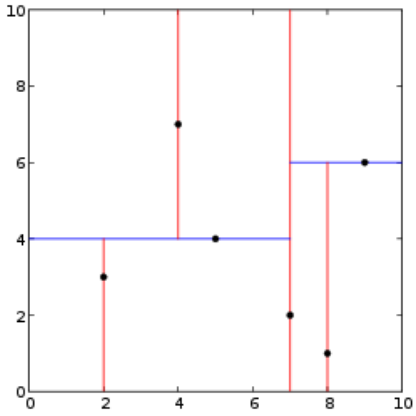
Phương pháp tìm kiếm theo bán kính có một số điểm mạnh nhất định trong việc xác định các đặc trưng của một điểm vì với phương pháp này, các điểm lân cận sẽ được xác định mà không phụ thuộc vào mật độ điểm xung quanh điểm cần xem xét, do đó nó không bị ảnh hưởng bởi khoảng cách cũng như góc nhìn từ điểm tới thiết bị quét. Phương pháp tìm kiếm thông dụng hơn là tìm kiếm theo số lân cận k được chọn từ trước.

Để xác định được k điểm gần nhất đối với điểm p_q cần xem xét từ đám mây điểm, ta sẽ phải tính khoảng cách từ tất cả các điểm trong đám mây điểm đến điểm p_q đó, sau đó chọn ra k điểm có khoảng cách nhỏ nhất. Quá trình này sẽ gây lãng phí tài nguyên máy tính khi thực hiện, vì việc tính toán quá nhiều khoảng cách là không cần thiết.

2.2.2. Tìm kiếm điểm lân cận bằng cây k-d tree

Cây k-d tree (hay k-dimensional tree) là một kiểu cấu trúc dữ liệu sử dụng trong ngành khoa học máy tính dành cho việc tổ chức dữ liệu điểm trong không gian k chiều. Cấu trúc dữ liệu cây k-d tree được sử dụng phổ biến trong nhiều ứng dụng, ví dụ như trong các thuật toán tìm kiếm trong không gian nhiều chiều.

Cây k-d tree là một cây nhị phân, trong đó mỗi nút lại là một điểm trong không gian k chiều. Mỗi điểm đó lại có thể sinh ra một siêu phẳng (hyperplane) chia không gian ra thành hai phần: phần bên trái và phần bên phải. Các điểm ở phần bên trái của siêu phẳng được biểu diễn ở nhánh bên trái của cây nhị phân, các điểm ở phần bên phải của siêu phẳng được biểu diễn ở nhánh bên phải của cây nhị phân. Mỗi nút của cây được chọn nhờ vào việc chia đôi một thuộc tính của một trong số k chiều, và siêu phẳng được tạo ra vuông góc với trục của chiều đó. Nhờ đó, các giá trị theo chiều đó của tập dữ liệu được phân loại thành hai phần: phần bên phải gồm các phần tử có giá trị lớn hơn và phần bên trái có giá trị nhỏ hơn.



Hình 2.7: Cây k-d trong không gian hai chiều.

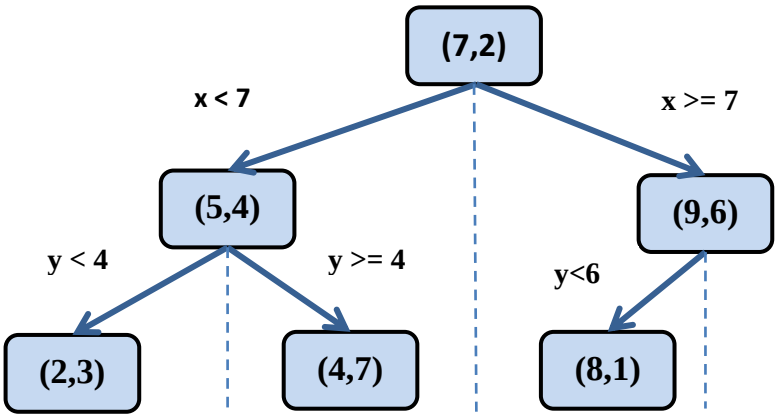
Quá trình tổ chức dữ liệu vào cây k-d được thực hiện như sau:

- Đầu tiên, một điểm X_1 trong không gian được chọn, điểm X_1 là nút của cây k-d tree.
- Sau đó, một thuộc tính *x* trong *k* thuộc tính được chọn ngẫu nhiên, từ đó, một siêu phẳng vuông góc với trục *x* tại điểm X_1 được thiết lập chia không gian thành hai phần.
- Các điểm trong không gian *k* chiều được xếp vào nhánh bên trái và bên phải cây nhị phân.
- Quá trình trên được lặp lại với mỗi nhánh con cho đến khi tất cả các điểm đều được xét tới.

Ví dụ về quá trình phân chia các điểm (7,2), (5,4), (9,6), (2,3), (4,7), (8,1) trong không gian hai chiều vào cây k-d tree:

Bảng 2.1: Quá trình sắp xếp dữ liệu vào cây k-d tree

Lần lặp lại	Điểm nút	Các điểm được xếp vào nhánh bên trái	Các điểm được xếp vào nhánh bên phải
1	(7,2)	(2,3), (5,4), (4,7)	(8,1), (9,6)
2	(5,4)	(2,3)	(4,7)
3	(9,6)	(8,1)	---



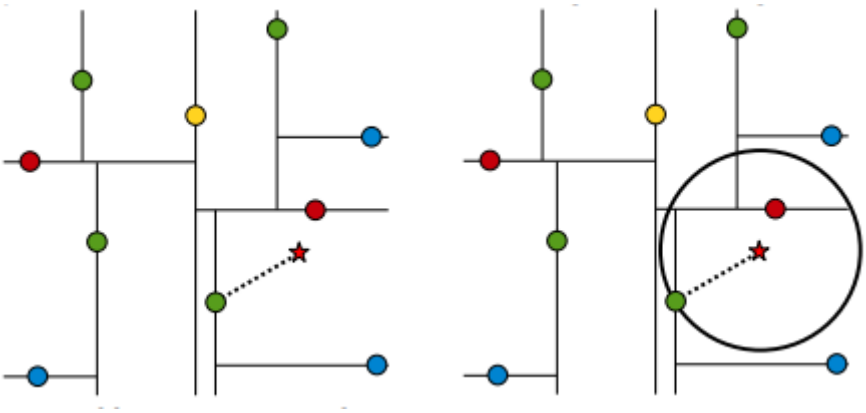
Hình 2.8: Phân chia các điểm vào cây k-d tree

Cấu trúc phân chia dữ liệu kiểu cây k-d tree được sử dụng cho việc tìm kiếm các lân cận của một điểm. Ưu điểm của phương pháp tìm kiếm lân cận trên cây k-d tree là hiệu quả tìm kiếm vì nó khai thác cách tổ chức dữ liệu trên cây k-d tree để loại bỏ một số lượng lớn các điểm trong không gian.

Quá trình tìm kiếm điểm lân cận trên cây k-d tree được thực hiện như sau:

- Thuật toán tìm kiếm theo các điểm nút trên cây k-d tree từ trên xuống dưới. Nguyên tắc tìm điểm cũng giống với nguyên tắc khi sắp xếp các điểm vào cây k-d tree: so sánh giá trị thuộc tính của điểm với điểm nút để đi xuống theo bên trái hoặc phải.
- Mỗi khi thuật toán đi qua một điểm nút, điểm nút đó được đánh dấu là điểm gần nhất hiện tại.
- Sau khi tìm đến điểm dưới cùng của cây k-d tree, thuật toán đi ngược lại từ dưới lên trên và thực hiện các bước sau:
    - Nếu nút hiện tại gần với điểm khảo sát hơn so, nút đó được đánh dấu là điểm gần nhất hiện tại.
    - Thuật toán kiểm tra xem có điểm nằm ở bên kia siêu phẳng phân chia mà gần hơn so với điểm gần nhất hiện tại hay không. Quá trình này được thực hiện bằng cách vẽ một siêu cầu bao quanh điểm cần khảo sát, có bán kính bằng với khoảng cách đến điểm gần nhất hiện tại và kiểm tra xem siêu cầu đó có cắt qua siêu phẳng phân tách hay không.
    - Nếu siêu phẳng và siêu cầu không cắt nhau, tất cả các điểm nằm bên kia siêu phẳng đều bị loại bỏ và thuật toán tiếp tục đi lên theo cây k-d tree.

- Nếu chúng cắt nhau, có thể sẽ tồn tại điểm gần nhất nằm về phía bên kia siêu phẳng. Khi đó, thuật toán sẽ duyệt nhánh bên kia của nút theo đúng trình tự tìm kiếm từ đầu.



Hình 2.9: Tìm kiếm điểm lân cận gần nhất trên cây k-d tree

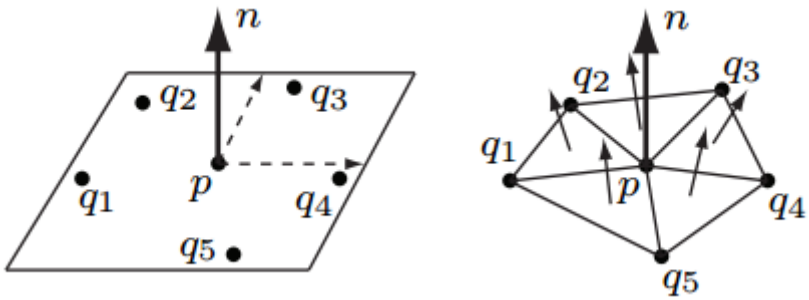
- Thuật toán kết thúc khi quá trình duyệt ngược đến nút cao nhất của cây k-d tree.

Thuật toán này có thể được mở rộng với việc tìm kiếm *k* điểm lân cận gần nhất. Trong trường hợp này, chương trình sẽ lưu lại *k* điểm gần nhất hiện tại thay vì 1. Mỗi nhánh sẽ bị loại khi có đã có *k* điểm gần nhất và nhánh đó không có điểm nào có khoảng cách đến tâm cầu nhỏ hơn các điểm đã chọn.

Khối lượng tính toán của thuật toán tìm lân cận gần nhất bằng cây k-d tree là tối đa *O(log n)* với tập dữ liệu phân phối ngẫu nhiên. Trường hợp này xảy ra khi thuật toàn phải duyệt hết toàn bộ cây nhị phân. Tuy nhiên trong các không gian có số chiều thấp như mặt phẳng hoặc không gian ba chiều, trường hợp này ít khi xảy ra.

2.2.3. Ước lượng véc tơ pháp tuyến

Các điểm lân cận có thể được sử dụng để mô tả không gian xung quanh một điểm, hay nói cách khác là biểu diễn bề mặt đi qua điểm đó. Phương pháp biểu diễn bề mặt là thông qua véc tơ pháp tuyến. Véc tơ pháp tuyến cũng là một trong những đặc trưng điểm cơ bản trong đám mây điểm. Việc ước lượng véc tơ pháp tuyến cho từng điểm cũng tương đương với việc xác định véc tơ pháp tuyến của mặt phẳng tiếp tuyến với bề mặt. Các phương pháp tính toán véc tơ pháp tuyến của một điểm đều dựa trên nguyên tắc chung là sử dụng các điểm lân cận. Các phương pháp chủ yếu thuộc hai dạng là phương pháp tối ưu (optimazation-based) và phương pháp lấy trung bình (average-based) [10].



(a) (b)

Hình 2.10: Hai phương pháp xác định véc tơ pháp tuyến.
(a): phương pháp tối ưu và (b): phương pháp lấy trung bình

Các phương pháp tối ưu ước lượng véc tơ pháp tuyến theo nguyên tắc ước lượng một mặt phẳng chứa điểm cần tìm và các lân cận của nó, sau đó tìm véc tơ pháp tuyến của mặt phẳng. Việc xác định mặt phẳng được thực hiện bằng cách tối thiểu sai số từ nó đến các lân cận.

Các phương pháp lấy trung bình được thực hiện dựa trên nguyên tắc lấy trung bình véc tơ pháp tuyến của các tam giác tạo thành từ điểm đó và một cặp điểm lân cận. Việc xác định véc tơ pháp tuyến ứng với các tam giác này có thể được thực hiện theo các tiêu chí khác nhau như theo diện tích (area-weighted), theo góc (angle-weighted), theo trọng tâm (centroid-weighted).

Trong các phương pháp xác định véc tơ pháp tuyến trên thì phương pháp được sử dụng rộng rãi nhất cho dữ liệu kiểu đám mây điểm là phương pháp dựa trên tối ưu. Với phương pháp này, việc xác định véc tơ pháp tuyến của bề mặt được chuyển thành bài toán khớp mặt phẳng bằng bình phương tối thiểu trong không gian ba chiều. Lời giải cho bài toán này là phương pháp PCA (Principal Component Analysis) với việc tìm trị riêng và véc tơ riêng của ma trận hiệp phương sai được tạo ra từ các điểm lân cận.

Mặt phẳng được biểu diễn bằng một điểm $\boldsymbol{x}$ và véc tơ pháp tuyến $\vec{n}$, k điểm lân cận xung quanh $\boldsymbol{x}$ là $\boldsymbol{P^k}$. Khoảng cách từ mỗi điểm p_i đến mặt phẳng là:

$$d_i = (p_i - x)\vec{n} \qquad (2.3)$$

Giá trị của $\boldsymbol{x}$ và $\vec{n}$ được tính bằng phương pháp bình phương tối thiểu để $d_i = 0$.

Lấy:

$$x = \bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{2.3}$$

Là tâm của tập $\boldsymbol{P^k}$ điểm, xác định ma trận phương sai *C:*

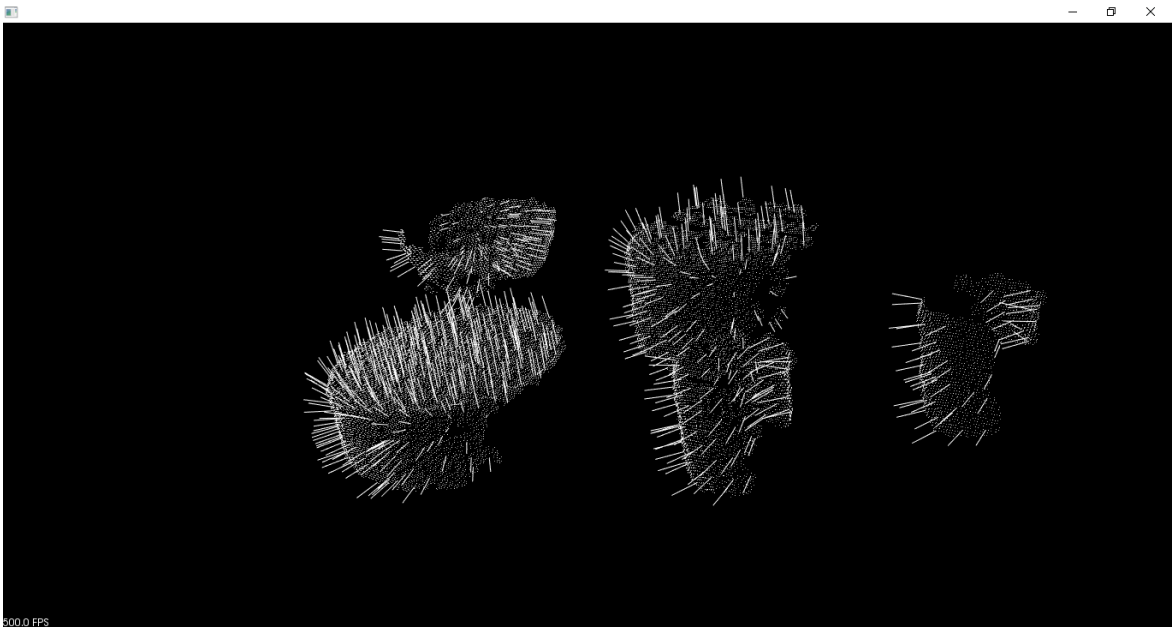
$$C = \frac{1}{k}\sum_{i=1}^{k} (p_i - \bar{p}).(p_i - \bar{p})^T \tag{2.4}$$

Véc tơ pháp $\vec{n}$ là các trị riêng của ma trận *C:*

$$C.\overrightarrow{v_j} = \lambda_j.\overrightarrow{v_j} \tag{2.5}$$

Với $\overrightarrow{v_j}$ là các véc tơ riêng, λ_j là các trị riêng, j = 0,1,2.

Tuy nhiên, có một vấn đề khi sử dụng phương pháp PCA để giải bài toán tìm véc tơ pháp tuyến trong đám mây điểm. Theo phương pháp này, dấu của véc tơ pháp tuyến không được xác định mà có thể theo cả hai chiều đối lập nhau. Trong khi đó ta cần các véc tơ pháp tuyến phải có dấu xác định một cách đồng nhất, nói cách khác, khi tính toán véc tơ pháp tuyến cho một bề mặt trong đám mây điểm, ta cần các điểm trong cùng một bề mặt có véc tơ pháp tuyến đều quay về một phía.



Hình 2.11: Ước lượng véc tơ pháp tuyến trong đám mây điểm

Để đồng nhất dấu của các véc tơ pháp tuyến, ta sử dụng một điểm nhìn (viewpoint) v_p. Dấu của các véc tơ pháp tuyến hướng về phía điểm nhìn sẽ thỏa mãn điều kiện:

$$\overrightarrow{n_i}(v_p - p_i) > 0 \quad (2.6)$$

Nhược điểm lớn nhất của phương pháp xác định véc tơ pháp tuyến thông qua các điểm lân cận là ở các cạnh, các vị trí mà tại có có sự thay đổi đột ngột về không gian. Tại những điểm này, lân cận của một điểm có thể thuộc về các bề mặt khác nhau, khiến cho véc tơ pháp tuyến xác định qua lân cận tại điểm đó không phản ánh đúng bề mặt. Điều này dẫn đến vấn đề lựa chọn các tham số phù hợp khi xử lý tính véc tơ pháp tuyến.

Khi xác định véc tơ pháp tuyến, số lượng các điểm lân cận dùng để tính toán *k* (với phương pháp tìm lân cận theo số lượng) hoặc bán kính tìm lân cận *r* (với phương pháp tìm lân cận theo bán kính) là các thông số cần được lựa chọn cẩn thận bằng thực nghiệm. Dữ liệu thật dưới dạng đám mây điểm thu thập từ các cảm biến thường chứa nhiều nhiễu từ môi trường bên ngoài. Sai lệch do nhiễu thống kê xuất hiện trên các bề mặt có thể được giảm bớt bằng cách tăng số lượng các điểm lân cận được chọn. Tuy nhiên việc này không chỉ làm tăng thời gian tính toán mà còn gây sai lệch nhiều hơn với các điểm nằm gần cạnh. Ngược lại, giảm số điểm lân cận sẽ giảm sai lệch với các điểm gần cạnh nhưng kết quả tổng thể bị ảnh hưởng nhiều hơn do nhiễu từ cảm biến.

2.2.4. Lược đồ đặc trưng điểm

Véc tơ pháp tuyến là một kiểu đặc trưng điểm mang tính cục bộ, sử dụng các điểm lân cận xung quanh và thể hiện tính chất của các điểm xung quanh điểm cần khảo sát. Tuy nhiên lượng thông tin trên véc tơ pháp tuyến là khá ít trong khi với nhiều trường hợp, người sử dụng cần biết thêm thông tin về điểm ví dụ như điểm đó nằm trên mặt phẳng, mặt trụ hay mặt cầu, … Từ đó có thể trích xuất thêm các thông tin về bề mặt hình học chứa điểm đó. Ở cấp độ tổng thể, các điểm có đặc trưng giống nhau sẽ thuộc về cùng một bề mặt và có thể được nhóm vào cùng một nhóm, từ đó hỗ trợ cho bài toán nhận diện và phân loại các vật thể. Phần này sẽ trình bày một đặc trưng điểm mạnh hơn là Point Feature Histogram (PFH) – lược đồ đặc trưng điểm. Đây cũng là đặc trưng điểm mang tính cục bộ và được tính toán dựa trên các điểm lân cận.

PFH là giải thuật được đề xuất bởi nhóm tác giả Rasu Bogdan Rusu [1]. PFH được tính toán dựa trên việc so sánh mối liên hệ giữa các véc tơ pháp tuyến của các cặp điểm với nhau trong cùng một lân cận. Nói cách khác, PFH tính toán độ sai lệch giữa các cặp véc tơ pháp tuyến với nhau, sau đó biểu diễn kết quả đó

dưới dạng histogram. Khi các điểm nằm trên các bề mặt hình học khác nhau như mặt phẳng, mặt cầu, mặt trụ, ... thì các véc tơ pháp tuyến của các lân cận điểm đó cũng có những sai khác với nhau theo một hình mẫu nhất định. Các điểm cùng nằm trên một mặt phẳng thường có véc tơ pháp tuyến song song với nhau; các điểm trên mặt trụ có véc tơ pháp tuyến thay đổi đều theo chiều quay trên một mặt phẳng, hay các điểm trên mặt cầu có véc tơ pháp tuyến lệch nhau theo cả ba chiều. PFH thể hiện điều này dưới dạng mỗi histogram cho từng điểm. Các điểm cùng nằm trên một bề mặt giống nhau sẽ có các histogram hình dạng giống nhau. Bằng cách khảo sát các histogram này, ta có thể biết được điểm đó đang nằm trên bề mặt hình học như thế nào.

PFH là một đặc trưng có thể được tính toán trong 3D đám mây điểm, là mở rộng của công trình nghiên cứu mối liên hệ giữa các cặp điểm trong hình khối 3D [2]. Mục đích của PFH là nó có thể giúp xác định được các hình khối không gian cơ bản (như mặt phẳng, hình trụ, hình nón …) trong 3D đám mây điểm. PFH là đặc trưng được tính riêng cho mỗi điểm trong không gian. Các điểm thuộc cùng một bề mặt trong không gian sẽ có PFH tương tự nhau, do đó có thể phân loại chúng.

Quá trình tính toán PFH sử dụng hai bán kính tìm lân cận r_1 và r_2. Trong đó $0 < r_1 < r_2$ và:

- Bán kính r_1 là bán kính tìm lân cận cho việc xác định véc tơ pháp tuyến.
- Bán kính r_2 là bán kính tìm lân cận cho việc tính toán PFH.

Tương ứng với hai bán kính r_1 và r_2 sẽ là hai tập điểm lân cận $\boldsymbol{P^{k_1}}$ và $\boldsymbol{P^{k_2}}$. Bước đầu tiên trong quá trình tính toán PFH là ước lượng véc tơ pháp tuyến của tất cả các điểm lân cận $\boldsymbol{P^{k_2}}$. Quá trình tìm véc tơ pháp tuyến này sử dụng bán kính r_1 để tìm lân cận. Thông thường khi tính PFH cho tất cả các điểm trong một tập dữ liệu thì các véc tơ cho tất cả các điểm được sử dụng như một dữ liệu đầu vào.

Quá trình tính PFH cho mỗi điểm được thực hiện bằng việc xét từng cặp hai điểm p_i và p_j (tương ứng là các véc tơ pháp tuyến n_i và n_j) trong $\boldsymbol{P^{k_2}}$. Hai điểm này được xác định một là điểm nguồn p_s và một là điểm đích p_t. Điểm nguồn được chọn sao cho góc giữa véc tơ pháp tuyến và đường nối hai điểm là nhỏ hơn.

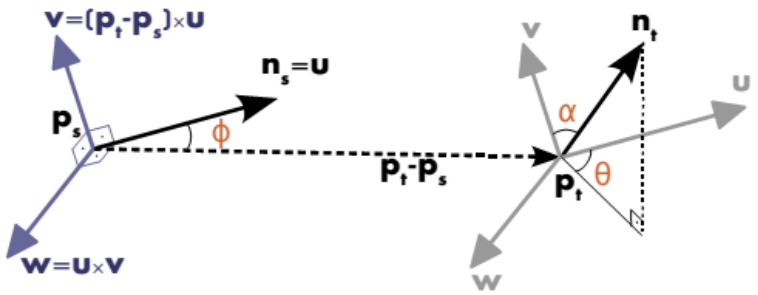
Nếu $\text{acos}(\overrightarrow{n_i}, \overrightarrow{p_{ji}}) \leq \text{acos}(\overrightarrow{n_j}, \overrightarrow{p_{ij}})$, $p_{ji} = p_j - p_i$, $p_{ij} = p_i - p_j$

Thì $\begin{cases} p_s = p_i, n_s = n_i \\ p_t = p_j, n_t = n_j \end{cases}$

Ngược lại, $\begin{cases} p_s = p_j, n_s = n_j \\ p_t = p_i, n_t = n_i \end{cases}$

Dựa trên hai điểm p_s, p_t và hai véc tơ pháp tuyến $\overrightarrow{n_s}$, $\overrightarrow{n_p}$, xây dựng hệ tọa độ Darboux [2] với các trục u, v, w như sau:

- $u = n_s$
- $v = \frac{(p_t - p_s) \times u}{\|p_t - p_s\|}$
- $w = u \times v$

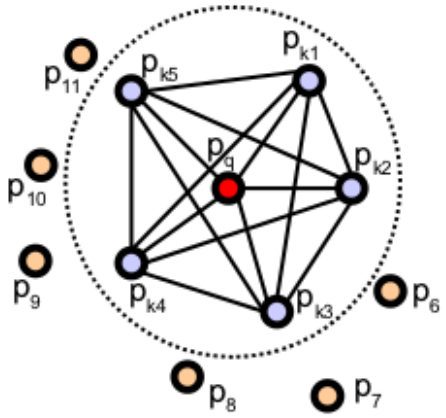


Hình 2.12: Tham số hóa mối liên hệ giữa hai véc tơ pháp tuyến [3]

Sử dụng hệ tọa độ Darboux, mối liên hệ giữa hai véc tơ pháp tuyến $\overrightarrow{n_s}$, $\overrightarrow{n_p}$ được biểu diễn bằng bộ các thông số sau:

- $\alpha = v.n_t$
- $\varphi = u.\frac{p_t - p_s}{\|p_j - p_i\|}$
- $\theta = \text{atan}(w.n_t, u.n_s)$
- $d = \|p_j - p_i\|$

Bộ bốn thông số $\langle \alpha, \varphi, \theta, d \rangle$ được tính cho tất cả các cặp điểm trong lân cận $\boldsymbol{P^{k_2}}$ của điểm cần khảo sát. Giả sử trong lân cận $\boldsymbol{P^k}$ có k điểm lân cận, khi đó số lượng bộ bốn thông số thu được sẽ là số cặp các điểm và bằng $\frac{k(k-1)}{2}$, độ phức tạp tính toán là $O(k^2)$. Với tập dữ liệu đầu vào đám mây điểm bao gồm n điểm, độ phức tạp tính toán sẽ là $O(nk^2)$.



Hình 2.13: điểm khảo sát p_q và các điểm lân cận

Cuối cùng, bộ thông số trên được đưa vào histogram. Quá trình này chia khoảng giá trị của mỗi thông số thành *b* phần bằng nhau, và đếm số lần xuất hiện của mỗi thông số trong mỗi phần. Trong bộ bốn thông số $\langle \alpha, \varphi, \theta, d \rangle$, khoảng cách *d* không được sử dụng trong quá trình đưa vào histogram này vì nó không thể hiện sự sai khác về hướng giữa hai véc tơ pháp tuyến. Đặt:

$$f_0 = \alpha$$
$$f_1 = \varphi$$
$$f_2 = \theta$$

Giá trị của mỗi cột trong histogram được thêm vào một nếu:

$$idx = \sum_{i=0}^{i \le 2} \left[\frac{f_i . n}{f_{imax} - f_{imin}} \right] . d^i \qquad (2.7)$$

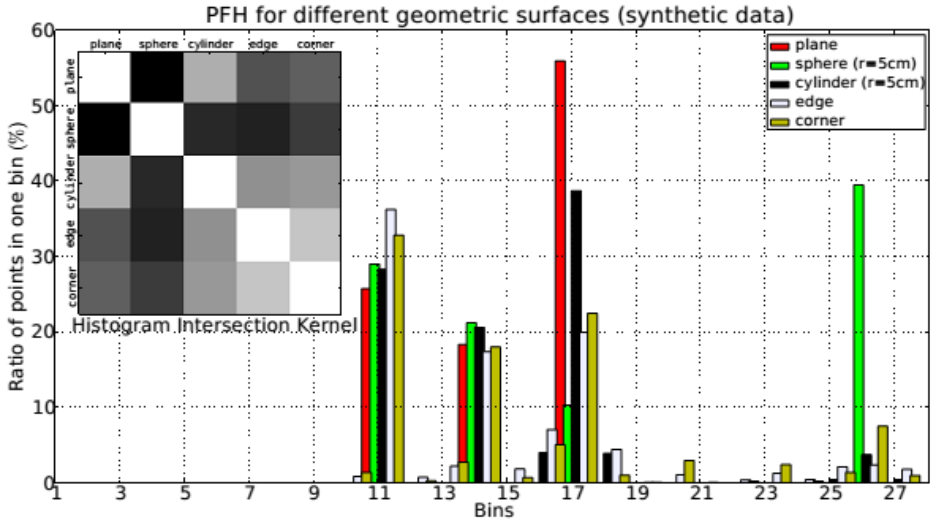
Với idx là số thứ tự của cột đó, f_{imax} và f_{imin} là giá trị lớn nhất và nhỏ nhất mà f_i có thể đạt được. Kết thúc quá trình này ta thu được một histogram đặc trưng cho điểm cần khảo sát. Số cột trong histogram được quyết định bởi số khoảng mà mỗi giá trị được chia ra. Ví dụ với khoảng giá trị được chia làm 5 phần thì số cột trong histogram sẽ là $5^3 = 125$ cột.

Đặc trưng histogram này là đặc trưng cho từng bề mặt. Nói cách khác khi điểm khảo sát nằm trên các bề mặt khác nhau thì histogram cho điểm đó sẽ có các hình dạng khác nhau, đặc trưng cho bề mặt chứa điểm đó. Để khảo sát sự khác nhau giữa histogram của những điểm nằm trên các bề mặt hình học khác nhau, ta sử dụng Histogram Intersection Kernel [4]. Đây là một giải thuật để khảo sát sự khác nhau giữa hai histogram bằng cách tính tổng các thành phần chung giữa hai

histogram đó. Giả sử có hai histogram A, B với số cột đều bằng *m*; gọi *ai*, *bi* (*i = 1…m)* là giá trị của cột thứ *i* trong histogram, khi đó Histogram Intersection Kernel được tính bằng:

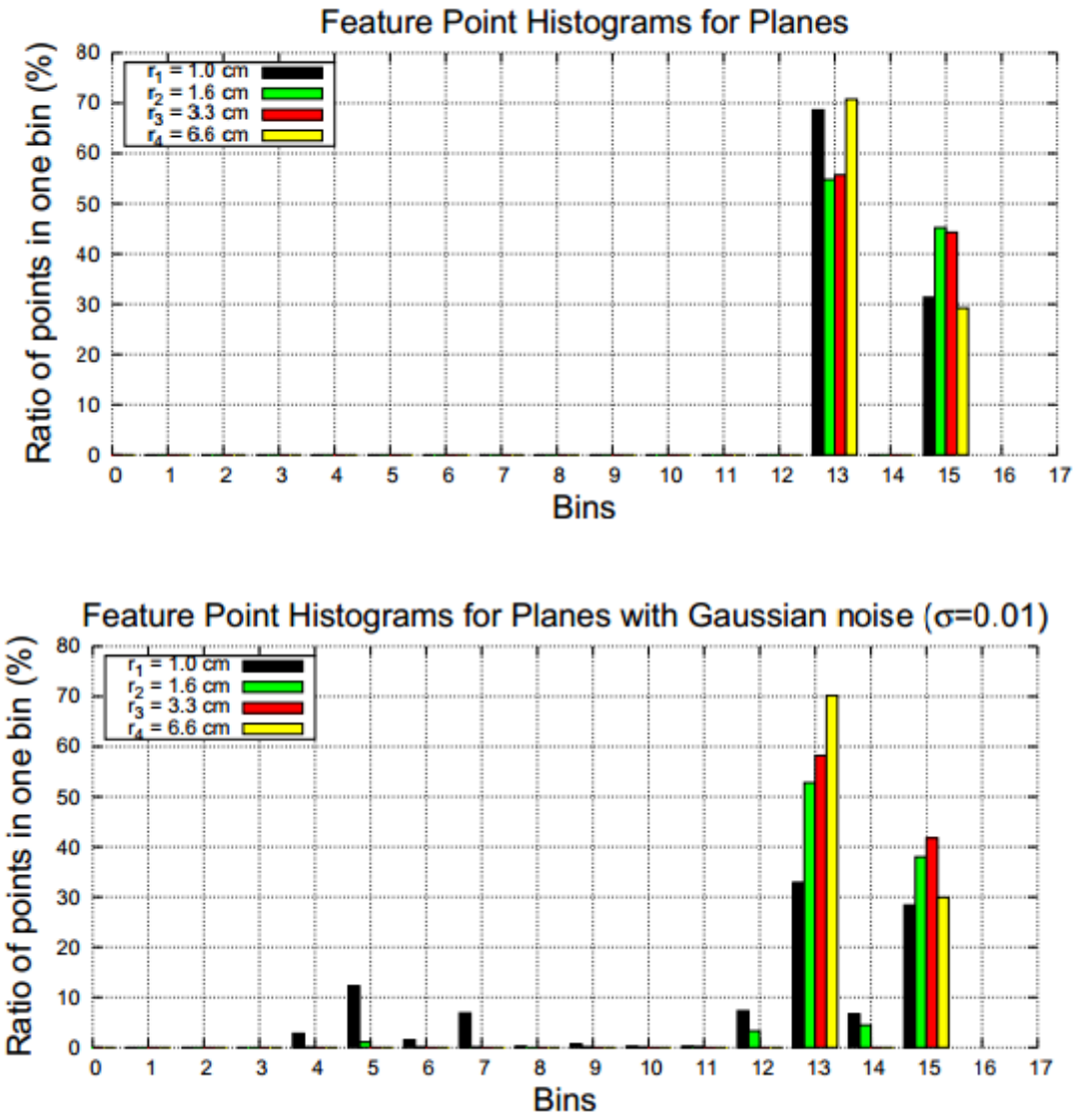
$$K(A,B) = \sum_{i=1}^{m} min\{a_i, b_i\} \tag{2.8}$$

Giá trị của $K(A,B)$ càng nhỏ thì độ sai khác giữa hai histogram A và B càng lớn, nghĩa là càng dễ phân biệt A và B. Ngược lại, $K(A,B)$ càng lớn thì A và B càng khó phân biệt.



Hình 2.14: PFH cho các bề mặt hình học khác nhau [3]

Khi tính toán PFH, sẽ xuất hiện trường hợp hai điểm p_1, p_2 là lân cận của nhau. Trong trường hợp này, sẽ có những điểm là lân cận chung của cả p_1 và p_2. Khi đó, quá trình tính toán PFH trong các lân cận của p_1 và p_2 sẽ gây ra hiện tượng nhiều PFH bị tính toán lại, gây tốn tài nguyên một cách không cần thiết. Điều đó đặt ra vấn đề lưu trữ các kết quả tính toán PFH của các cặp điểm lân cận trong bộ nhớ đệm để làm giảm độ phức tạp tính toán, từ đó cải thiện phần nào khối lượng và thời gian tính toán.



Hình 2.15: PFH cho mặt phẳng không nhiễu (hình trên) và có nhiễu (hình dưới)

Chất lượng tính toán PFH phụ thuộc rất nhiều vào chất lượng của quá trình tính toán véc tơ pháp tuyến trước đó, do đó nó có thể bị ảnh hưởng bởi nhiễu trên dữ liệu đám mây điểm. Sẽ có những trường hợp sai khác giữa hai giá trị là không cao, nhưng nó vượt quá ngưỡng xét khi đưa vào histogram, làm cho giá trị đó bị đưa vào một cột khác trong histogram.

Chương 3: Phân loại đặc trưng điểm bằng phương pháp học máy SVM

Sau khi trích xuất được các đặc trưng điểm từ dữ liệu đám mây điểm, các đặc trưng này sẽ được sử dụng để nhận diện các dạng bề mặt trên đám mây điểm cần khảo sát. Ý tưởng ở đây là so sánh các đặc trưng điểm thu được với một mô hình sẵn có và xếp chúng vào các loại đã được nhận diện. Để thực hiện điều này, tôi sử dụng phương pháp máy véc tơ hỗ trợ (SVM). Chương này sẽ trình bày khái niệm cũng như cách hoạt động của phương pháp SVM trong bài toán nhận diện và phân loại dữ liệu.

3.1. Khái niệm máy véc tơ hỗ trợ

Phương pháp Support Vector Machine (Máy Véc tơ hỗ trợ - SVM) là một phương pháp học máy được sử dụng cho các bài toán phân loại dữ liệu [5]. Về cơ bản, SVM nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau. Do đó SVM là giải thuật phân loại nhị phân. Với một bộ các ví dụ luyện tập (training data) trong không gian nhiều chiều, thuộc hai lớp cho trước, giải thuật luyện tập SVM xây dựng một mô hình để tìm ra một siêu phẳng (hyperplane) phân chia ranh giới giữa hai nhóm sao cho khoảng cách từ các véc tơ luyện tập tới ranh giới là xa nhất có thể. Sau khi đã có mô hình SVM, các ví dụ mới cũng được biểu diễn trong cùng một không gian và được mô hình dự đoán thuộc một trong hai lớp tùy vào ví dụ đó nằm ở phía nào của siêu phẳng.

3.2. Mô hình phân lớp SVM

Thuật toán được sử dụng với SVM là thuật toán tìm siêu phẳng phân chia dữ liệu đã có. Giả sử có *l* dữ liệu huấn luyện:

$$D = \{(\boldsymbol{x_i}, y_i) | (\boldsymbol{x_i} \in R^d), y_i \in \{-1,1\}\}, i = 1, \dots, l$$

Trong đó:

- *D* là tập dữ liệu đầu vào gồm có l mẫu.
- $\boldsymbol{x_i}$ là dữ liệu đầu vào. Dữ liệu này có dạng véc tơ trong không gian *d* chiều, với mỗi chiều biểu diễn một thuộc tính.
- y_i là nhãn của dữ liệu $\boldsymbol{x_i}$, nhận giá trị –1 và 1, thể hiện dữ liệu $\boldsymbol{x_i}$ thuộc lớp –1 hay 1.

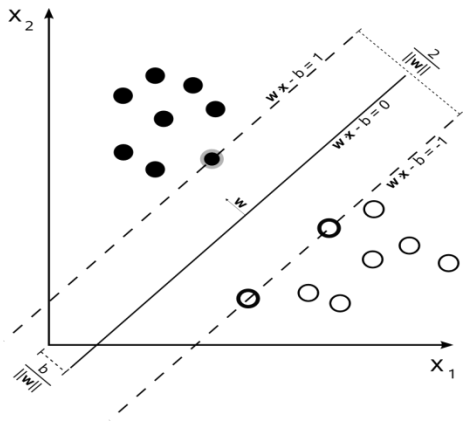
Sau quá trình luyện tập với tập dữ liệu đầu vào D, SVM sẽ đưa ra các tham số của phương trình siêu phẳng ranh giới giữa hai lớp, hai tham số này là véc tơ $\boldsymbol{w}$ có p chiều và một tham số b. Phương trình của siêu phẳng là:

$$\boldsymbol{w}.\boldsymbol{x} + b = 0 \quad (4.1)$$

Khi đó hàm số biểu diễn dấu của biểu thức:

$$f(x) = sign(\boldsymbol{w}.\boldsymbol{x} + b) \quad (4.2)$$

Là hàm số có khả năng phân chia hoàn toàn dữ liệu vào một trong hai lớp.



*Hình 3.1: Siêu phẳng (**w**,b) tối ưu phân chia 2 class.*

Sau quá trình luyện tập, các véc tơ mẫu thử có thể được phân vào một trong hai lớp bằng hàm số $f(x)$:

- $f(x) = sign(w.x + b) > 0$: xếp mẫu thử vào lớp 1.
- $f(x) = sign(w.x + b) < 0$: xếp mẫu thử vào lớp -1.

Theo thuật toán học SVM, hệ số w và b được tính theo công thức:

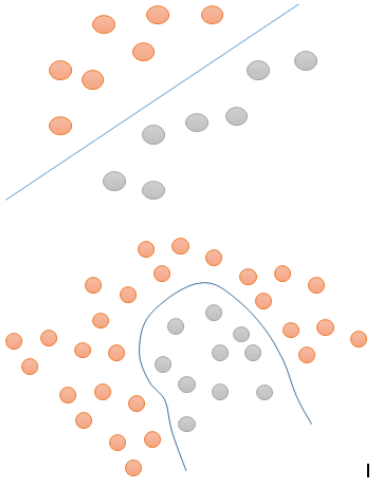
- $w = \sum_i \ \alpha_i y_i x_i$
- $b = -\frac{1}{2}(w.x^+ + w.x^-)$

Trong đó các véc tơ x_i là các véc tơ hỗ trợ. Các véc tơ hỗ trợ là các véc tơ nằm sát với siêu phẳng phân cách hai phân lớp này. x^+ là véc tơ hỗ trợ với nhãn +1, x^- là véc tơ hỗ trợ với nhãn -1.

3.3. Chuyển đổi không gian dữ liệu SVM

Các véc tơ thuộc tập D có thể được phân thành hai lớp trong không gian P chiều bởi một siêu phẳng hoặc không. Do SVM là một thuật toán sử dụng siêu phẳng để phân lớp, nếu các véc tơ thuộc tập D không thể được phân lớp trong không gian P chiều bởi một siêu phẳng thì cần chuyển đổi các véc tơ thuộc tập D sang một không gian mới bằng một phép chuyển đổi không phá vỡ tính phân lớp của tập D mà vẫn có thể phân lớp chúng bằng một siêu phẳng. Phép biến đổi này được kí hiệu như sau:

$$x \rightarrow \phi(x)$$



Hình 3.2: Chuyển đổi không gian dữ liệu SVM

Phép biến đổi từ x thành $\phi(x)$ có thể giữ nguyên hoặc làm tăng, giảm số chiều P của không gian x. Việc chuyển đổi các véc tơ x_i thành các véc tơ $\phi(x_i)$ theo các phép chuyển đổi được gọi là Kernel.

$$w = \sum_i \alpha_i y_i x_i \rightarrow w = \sum_i \alpha_i y_i \phi(x_i) \tag{4.3}$$

$$f(x) = sign(w.\phi(x) - b) = sign\left(\left[\sum_i \alpha_i y_i \phi(x_i)\right].\phi(x) - b\right)$$

$$= sign\left(\sum_i \alpha_i y_i \phi(x_i).\phi(x) - b\right)$$

$$= sign\left(\sum_i \alpha_i y_i [K(x_i, x)] - b\right)$$

Trong đó $K(x_i, x) = \phi(x_i).\phi(x)$ được gọi là hàm Kernel của véc tơ x.

Trong quá trình học SVM, đầu tiên ta coi tập huấn luyện và tập thử nghiệm là có thể được phân tách bởi siêu phẳng và không cần dùng hàm Kernel để chuyển đổi không gian của tập dữ liệu. Nếu sai số thử nghiệm là nhỏ, ta không cần chuyển đổi không gian. Tuy nhiên nếu sai số là lớn, ta cần chuyển đổi không gian tập dữ liệu sử dụng một số các hàm Kernel phổ biến.

3.4. Các hàm Kernel phổ biến

3.4.1. Kernel đa thức

Kernel đa thức là Kernel có dạng:

$$K(x_a, x_b) = (x_a . x_b + 1)^p \qquad (4.4)$$

Trong đó, *p* là một tham số có thể tùy chỉnh. Trong các ứng dụng thực tế, *p* thường dao động trong khoảng từ 1 đến 10.

Khai triển phép nhân véc tơ trong biểu thức trên, ta có:

$$K(x_a, x_b) = (x_{a1}x_{b1} + x_{a2}x_{b2} + \cdots + x_{ad}x_{bd} + 1)^p$$

Mỗi khi bậc của đa thức tăng, đa thức lại nhân thêm với *(d+1)* giá trị của chúng. Kết quả là sẽ có $\binom{d+p-1}{p}$ phần tử trong đa thức triển khai, tức là có ngần ấy cách biến đổi về bậc của các véc tơ dữ liệu đầu vào. Bằng cách sử dụng đa thức với bậc *p* cao, số chiều của miền các véc tơ đầu vào sẽ tăng lên nhiều lần, khi đó việc tìm ranh giới phân chia dữ liệu sẽ dễ dàng hơn. Tuy nhiên ở miền có nhiều chiều hơn thì số lượng các véc tơ hỗ trợ cũng tăng lên.

3.4.2. Kernel RBF

Một dạng Kernel phổ biến khác là Kernel Gaussian RBF, có dạng:

$$K(x_a, x_b) = exp\left(-\frac{\|x_a - x_b\|^2}{2\sigma^2}\right) \qquad (4.5)$$

Trong đó σ là một tham số có thể điều chỉnh được. Sử dụng Kernel này dẫn tới kết quả là hàm phân chia sẽ có dạng:

$$f(x) = sign\left[\sum_i \alpha_i y_i \, exp\left(-\frac{\|x_a - x_b\|^2}{2\sigma^2}\right) + b\right]$$

Về cơ bản, đây là một hàm RBF (Radical Basis Function), với các véc tơ hỗ trợ nằm ở tâm. Do đó ở không gian mới này, SVM chỉ hoàn toàn là tìm ra số các tâm cần thiết (và vị trí của chúng) để tạo thành một mạng lưới RBF với hiệu năng cao nhất có thể.

Thông thường khi thiết kế một mô hình SVM, dữ liệu đầu vào thường không được phân tách tuyến tính rõ ràng. Do đó, việc sử dụng Kernel để biến đổi dữ liệu đầu vào sang một không gian dữ liệu mới dễ dàng hơn cho việc phân tách là thường xảy ra trong thực tế. Câu hỏi được đặt ra là sử dụng Kernel nào thì thích hợp và với mỗi Kernel đó, các tham số điều chỉnh ra sao (với Kernel đa thức thì tham số cần điều chỉnh là p – bậc của đa thức, còn với Kernel RBF thì người thiết kế mô hình cần lựa chọn tham số σ). Việc chọn Kernel và điều chỉnh các thông số phù hợp được thực hiện bằng thực nghiệm. Người thiết kế có thể thử sử dụng các loại Kernel khác nhau và chọn ra Kernel với kết quả tốt nhất.

Chương 4: Kết quả thực nghiệm

Chương trình được viết bằng ngôn ngữ C++ trên môi trường Visual studio 2013. Chương trình sử dụng hai thư viện mở:

- Point Cloud Library (PCL) là thư viện hỗ trợ xử lý đám mây điểm.
- Libsvm là thư viện hỗ trợ xử lý liên quan đến SVM bao gồm xây dựng mô hình và thử nghiệm, phân loại.

4.1. Thư viện mở Point Cloud Library

Point Cloud Library (PCL) là một thư viện mã nguồn mở chứa các thuật toán về xử lý đám mây điểm và hình học 3D, chuyên phục vụ cho việc nghiên cứu thị giác máy tính trong không gian ba chiều [7]. Thư viện PCL được phát triển từ tháng 03/2010 bởi Willow Garage – một phòng thí nghiệm chuyên nghiên cứu về Robotics đặt tại Mĩ. PCL được ra mắt lần đầu vào tháng 05/2011.

Hình 4.1: Logo của Point Cloud Library

Thư viện PCL bao gồm các module:

- Filter: Thư viện phục vụ các chức năng thực thi các bộ lọc cơ bản trong xử lý đám mây điểm như giảm mẫu, lọc theo khoảng cách, trích xuất index, chiếu, …
- Feature: Thư viện phục vụ thực thi trích xuất các đặc trưng hình học trong không gian ba chiều như ước lượng pháp tuyến và độ cong, moment, PFH và FPFH, đặc trưng NARF, VFH, RIFT, …
- I/O: Thư viện phục vụ thực thi các chức năng vào/ra của chương trình như đọc, ghi đám mây điểm từ ổ đĩa.
- Phân đoạn: Thư viện phục vụ thực thi phân đoạn đám mây điểm, bao gồm các chức năng như khớp mô hình hình học, ghép nhóm, RANSAC, …
- Surface: Thư viện phục vụ thực thi các kỹ thuật khôi phục các bề mặt trên đám mây điểm.

- Registration: Thư viện phục vụ thực thi các phương pháp ghép đám mây điểm như ICP.
- Keypoints: Thư viện phục vụ thực thi các phương pháp tìm kiếm và trích xuất đặc điểm trong đám mây điểm.
- Rangeimage: Thư viện hỗ trợ tạo ảnh tầm xa (range image) từ dữ liệu đám mây điểm.

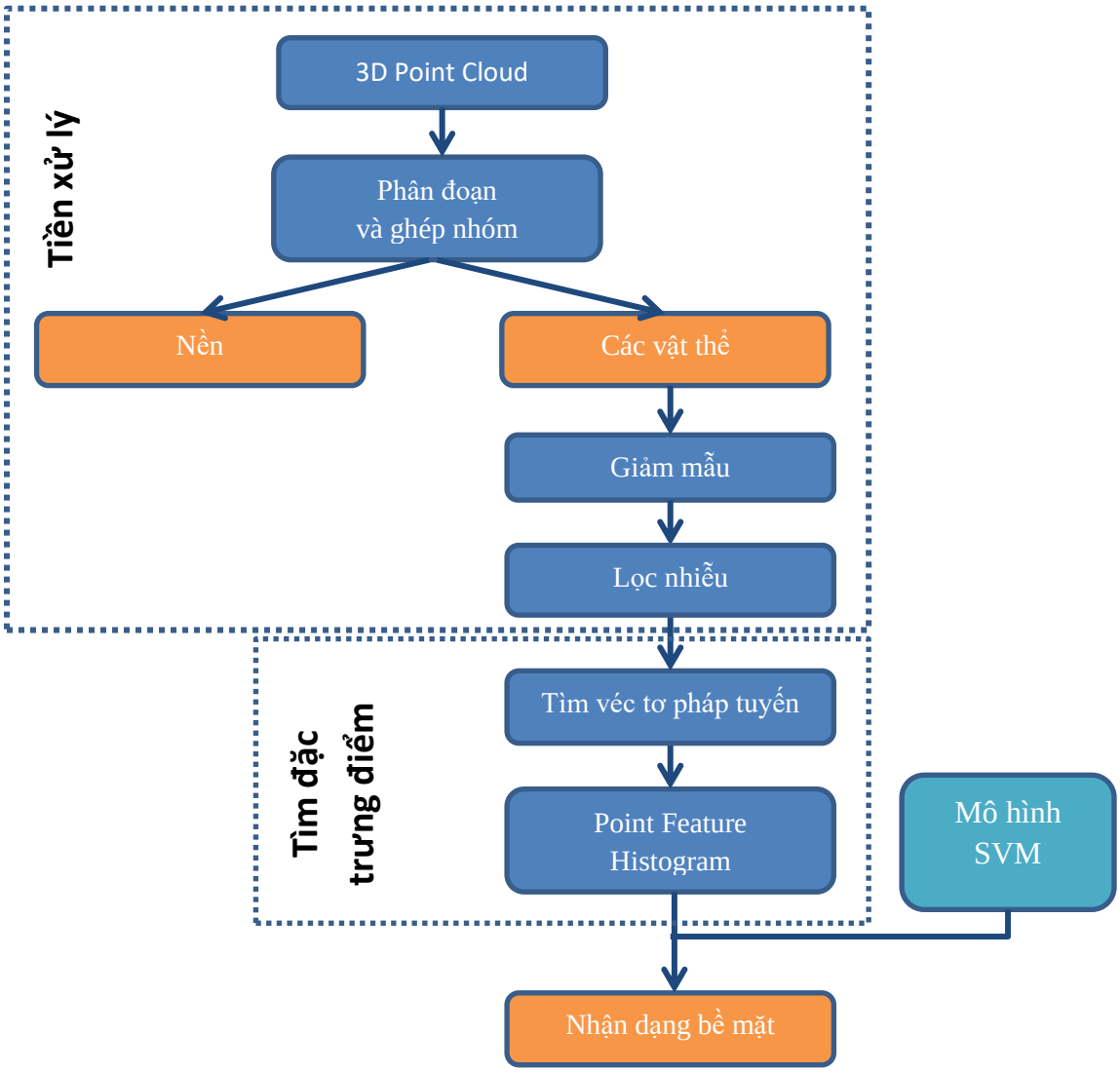
4.2. Thư viện mở libsvm

LIBSVM [6] là một thư viện mã nguồn mở về học máy, được phát triển bởi trường Đại học Quốc gia Đài Loan. Thư viện libsvm cung cấp việc thực hiện giải thuật tối thiểu tuần tự (SMO – sequential minimal optimization) cho SVM với kernel, sử dụng cho các bài toán phân loại và phân tích hồi quy. Thư viện libsvm gốc được viết trên ngôn ngữ C++. Mã nguồn của libsvm được sử dụng lại trong một số bộ công cụ về học máy như trong Matlab, OpenCV, …

4.3. Sơ đồ chương trình

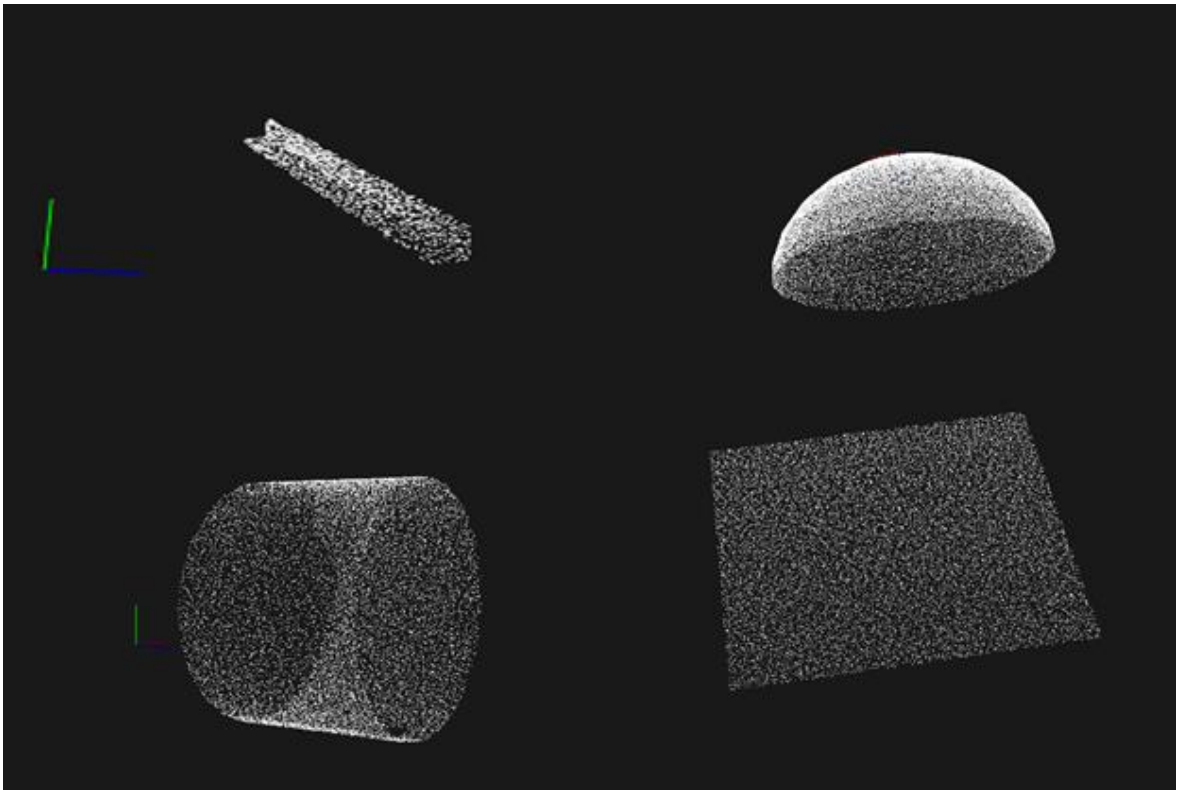
Hình 4.2 trình bày sơ đồ giải thuật của chương trình được thực hiện trong luận văn. Chương trình gồm ba thành phần chính:

- Tiền xử lý: bao gồm các bước giảm mẫu, lọc nhiễu, tách bề mặt khỏi vật thể.
- Tìm đặc trưng điểm: bao gồm có ước lượng véc tơ pháp tuyến và tính toán đặc trưng PFH.
- Nhận dạng bề mặt: Sử dụng mô hình SVM để kiểm tra, phân loại các dạng bề mặt.



Hình 4.2: Sơ đồ tổng thể chương trình

Đầu vào của chương trình là ảnh tĩnh dưới dạng 3D đám mây điểm. Trong quá trình tạo dữ liệu cho việc xây dựng mô hình SVM, chương trình sử dụng dữ liệu đám mây điểm không nhiễu, mô tả các bề mặt hình học ba chiều cơ bản bao gồm hình mặt cầu, hình trụ, mặt phẳng và cạnh. Dữ liệu này được lấy từ cơ sở dữ liệu 3D của đại học Washington tại địa chỉ http://rgbd-dataset.cs.washington.edu/dataset/rgbd-scenes-v2/.



Hình 4.3: các dữ liệu được sử dụng cho xây dựng mô hình SVM

Trong quá trình phân loại bề mặt vật thể, chương trình sử dụng dữ liệu không nhiễu lấy từ cơ sở dữ liệu của đại học Washington như trên, và dữ liệu thật quét từ cảm biến Kinect lấy từ cơ sở dữ liệu mOSD (modified Object Segmentation Dataset) của cộng đồng PCL tại địa chỉ https://github.com/PointCloudLibrary/data/tree/master/segmentation/mOSD.
Trong các dữ liệu thử nghiệm, khoảng cách trung bình từ camera cho đến các vật thể là khoảng 1.5 m.

Đám mây điểm đầu vào sau đó được tách vật thể và nền bằng phương pháp RANSAC. Trong thực nghiệm với các đám mây điểm đầu vào với khoảng hơn 300.000 điểm thì phần chứa vật thể bao gồm khoảng 50.000 đến 70.000 điểm tùy vào số lượng và kích cỡ các vật thể. Phần nền còn lại được loại bỏ khỏi dữ liệu xử lý.

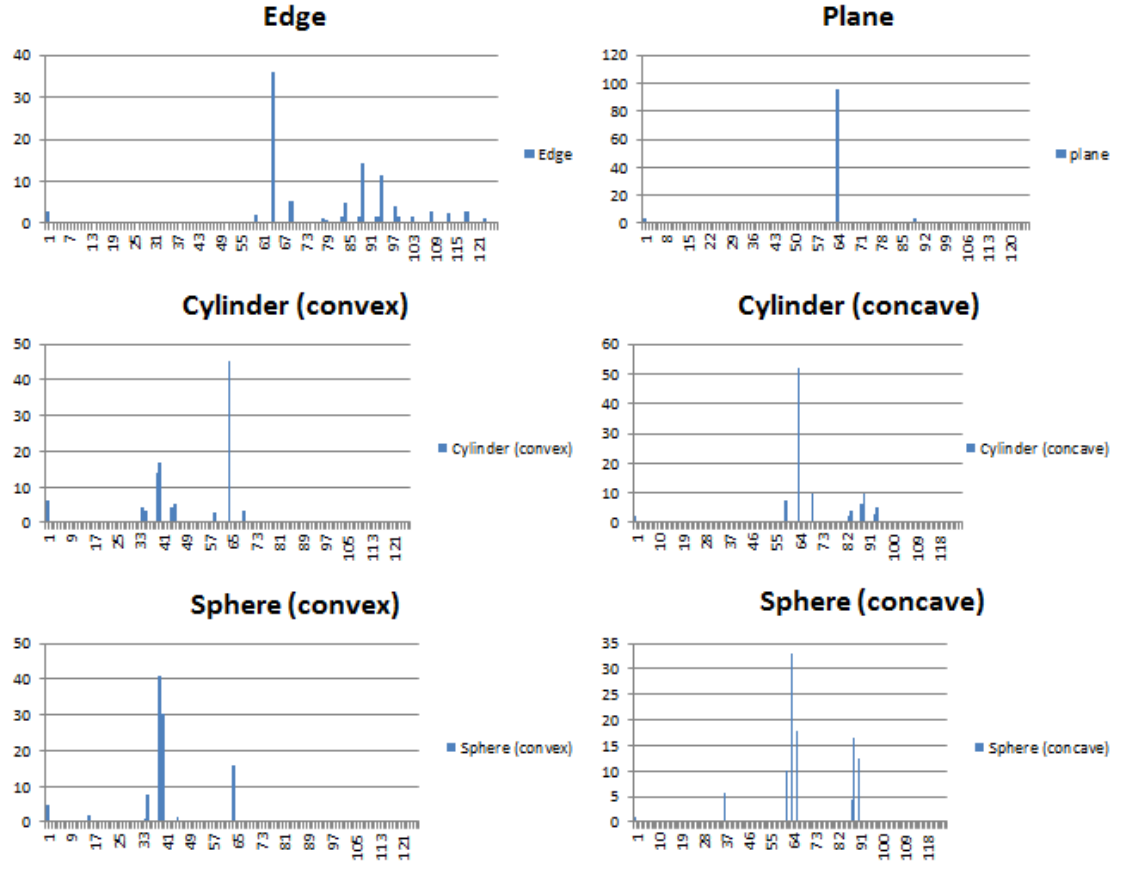
Quá trình tiền xử lý sau đó bao gồm giảm mẫu và lọc nhiễu đám mây điểm với mục đích giảm thời gian tính toán cho các bước xử lý tiếp theo. Trong giai đoạn giảm mẫu, kích thước lá voxelgrid là rất quan trọng vì cần phải đủ lớn để có mật độ điểm không quá dày nhưng vẫn không làm mất đặc trưng điểm. Chi tiết

về các giá trị là voxel grid được thử nghiệm trong phần kết quả. Công đoạn lọc nhiễu hay lọc những dữ liệu không liên quan được thực hiện sau khi giảm mẫu.

Giai đoạn tính toán đặc trưng điểm bắt đầu với việc xác định véc tơ pháp tuyến cho từng điểm trong đám mây điểm, sử dụng cấu trúc dữ liệu dạng cây k-d tree và phương pháp PCA để tìm trị riêng và véc tơ riêng. Bán kính xác định lân cận trong các trường hợp r_1 được tùy chỉnh theo bán kính tính toán PFH r_2 với $r_1 = r_2 - 5\text{mm}$. Các véc tơ pháp tuyến sau đó được đồng nhất hướng bằng điểm nhìn là điểm đặt camera.

Quá trình tính toán PFH là bước tiêu tốn tài nguyên chính của chương trình. Tham số quan trọng trong quá trình này là bán kính tính toán PFH r_2. Các giá trị r_2 khác nhau được thử nghiệm lần lượt trên dữ liệu từ Kinect.

Trong quá trình xây dựng mô hình SVM, các đặc trưng PFH của từng điểm ứng với các dạng bề mặt khác nhau được ghi ra file .csv. Hình sau thể hiện histogram của 6 dạng bề mặt khác nhau được khảo sát và thử nghiệm.



Hình 4.4: Các dạng histogram ứng với các bề mặt khác nhau

Mô hình học máy SVM nhận diện 6 dạng bề mặt cơ bản là mặt phẳng, cạnh, mặt trụ lồi, mặt trụ lõm, mặt cầu lồi, mặt cầu lõm. Kết quả nhận diện được thể hiện bằng màu gán cho từng điểm như trong bảng sau:

Bảng 4.1: Màu tương ứng với các dạng bề mặt

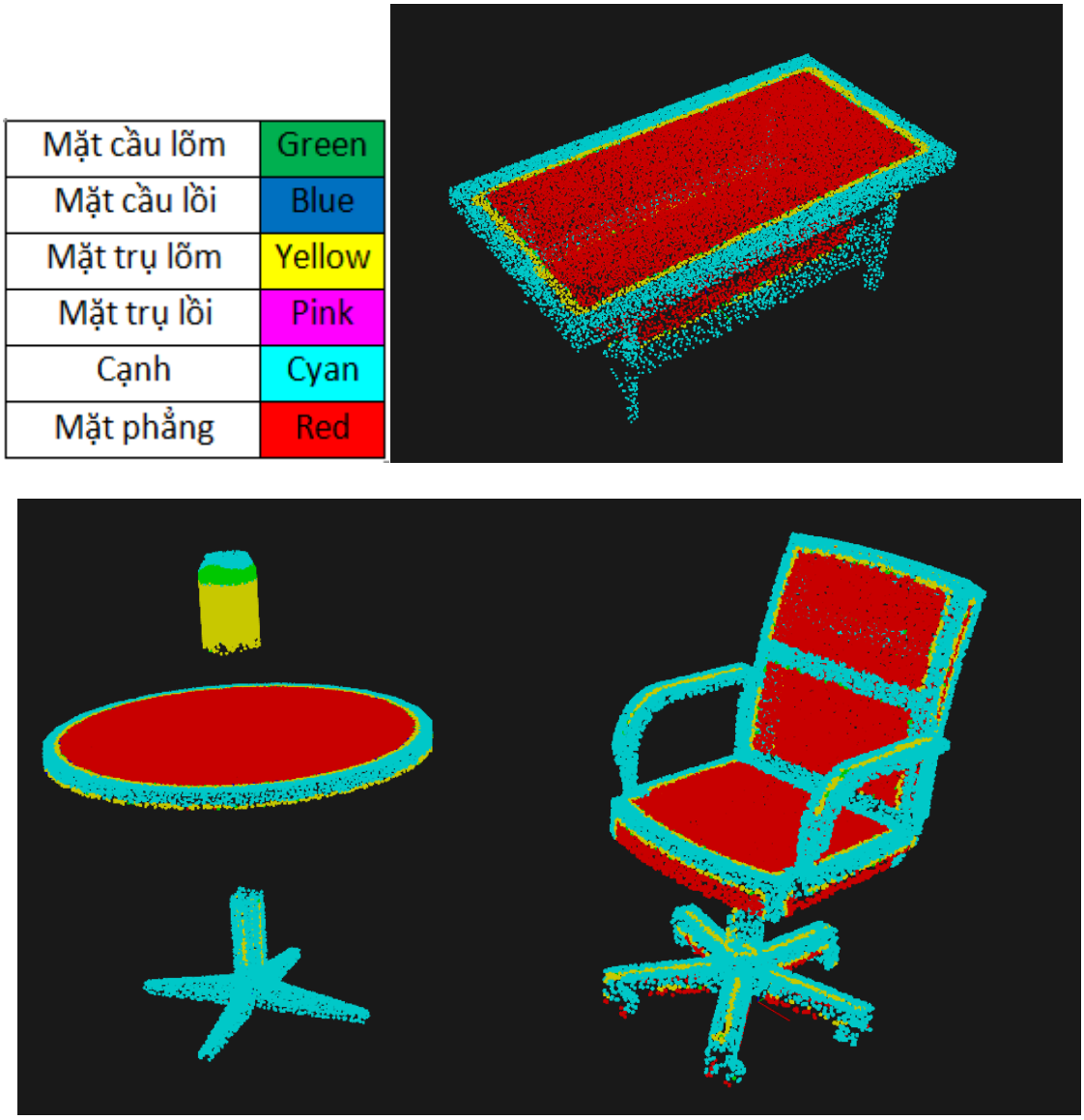
Mặt cầu lõm	Green
Mặt cầu lồi	Blue
Mặt trụ lõm	Yellow
Mặt trụ lồi	Pink
Cạnh	Cyan
Mặt phẳng	Red

4.4. Kết quả

4.4.1. Kết quả trên dữ liệu không nhiễu

Trước tiên, mô hình học máy SVM được thử trên các dữ liệu đầu vào không nhiễu. Dữ liệu được thử là dữ liệu vẽ 3D một số đồ vật trong nhà như bàn, ghế, … Mô hình tiến hành phân loại các bề mặt khác nhau trên đồ vật. Do các đồ vật này không bao gồm một bề mặt lớn làm nền nên chương trình không sử dụng chức năng phân đoạn để tách nền khỏi vật.

Kết quả cho một số đồ vật:

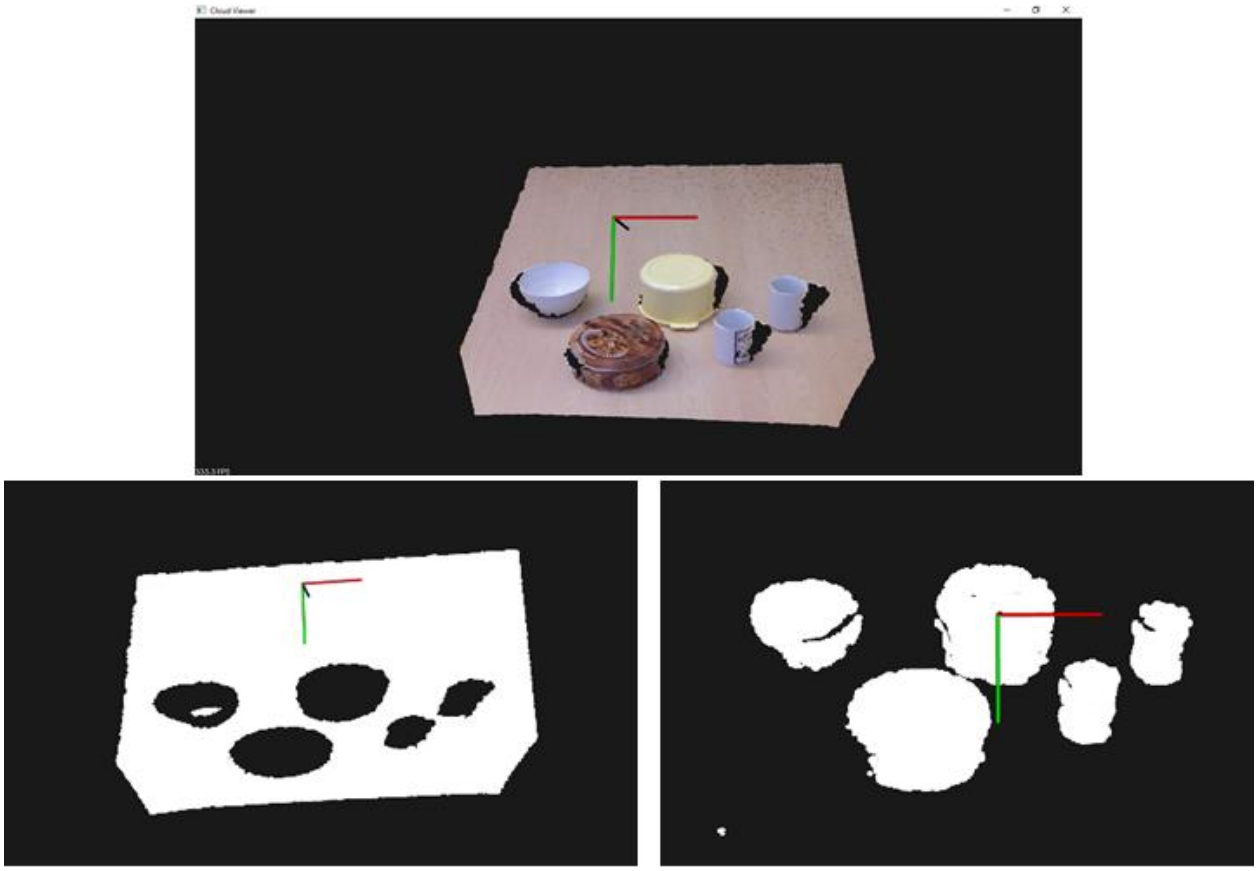


Hình 4.5: Kết quả thử nghiệm với dữ liệu không nhiễu

Trên dữ liệu không nhiễu, chương trình dự đoán khá chính xác các bề mặt được xuất hiện trên vật thể như mặt trụ, mặt phẳng, cạnh. Các chi tiết có kích thước nhỏ như chân bàn, chân ghế, tay vịn, được nhận ra là cạnh.

4.4.2. Kết quả trên đám mây điểm quét từ Kinect

Với dữ liệu vật thể quét từ Kinect, chương trình thực hiện đầy đủ các bước tiền xử lý bao gồm tách nền, ghép nhóm, giảm mẫu, lọc các điểm không liên quan.



Hình 4.6: đám mây điểm đầu vào và sau khi đã tách nền

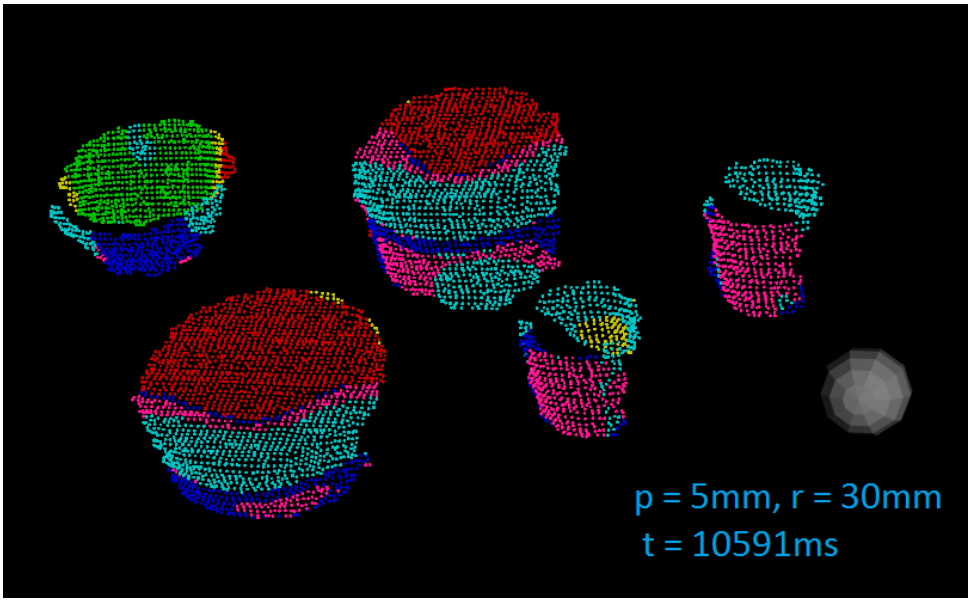
Đám mây điểm đầu vào (hình trên); nền (hình dưới bên trái); các vật thể được tách ra khỏi nền (hình dưới bên phải).

Hình 4.6 thể hiện dữ liệu đầu vào sau các bước tiền xử lý. Kết quả nhận diện bề mặt được thể hiện trong hình 4.7 với quả cầu ở góc dưới bên trái có bán kính bằng với bán kính tính đặc trưng PFH.

Về cơ bản, trên dữ liệu từ Kinect với nhiều nhiễu lượng tử thì chương trình vẫn có thể nhận diện phần lớn các bề mặt: mặt phẳng; mặt tròn lồi, lõm (thể hiện bằng màu xanh lá cây và xanh nước biển trên vật thể bát); mặt trụ lồi; cạnh (thể

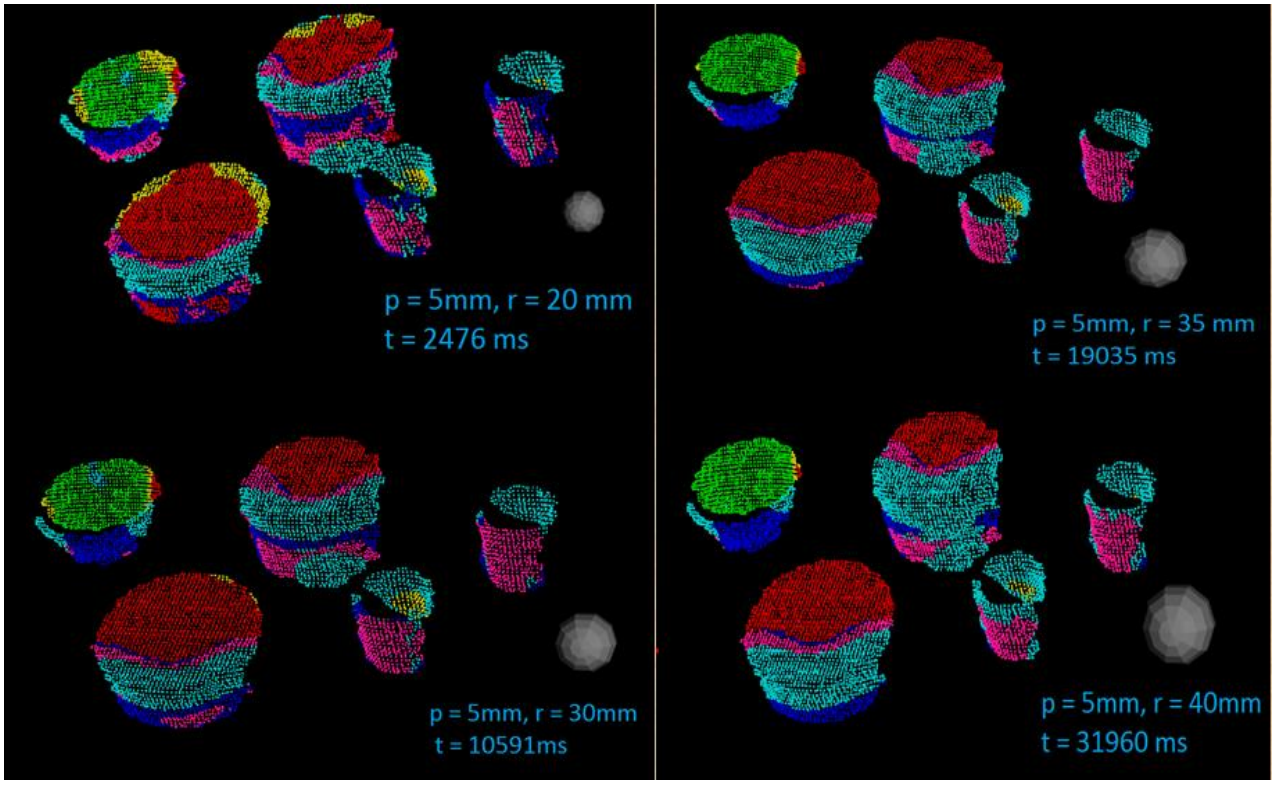
hiện bằng màu xanh lục lam ở các phần tiếp giáp giữa mặt phẳng và mặt trụ). Các điểm bị nhận diện sai chủ yếu nằm ở những phần có nhiều nhiễu lượng tử như phần rìa của các bề mặt, phần tiếp giáp, chuyển giao giữa hai bề mặt hay các phần có góc nhìn nhỏ từ cảm biến.

Đối với các bề mặt có diện tích thấp như mặt trụ của hộp bánh hay mặt trụ lõm, chương trình nhận diện chúng là hỗn hợp của nhiều bề mặt khác nhau, trong đó phần cạnh chiếm đa số diện tích. Với các chi tiết nhỏ như tay cầm (có kích thước nhỏ hơn hình cầu), chương trình đều nhận diện là cạnh.



Hình 4.7: Kết quả thử nghiệm với dữ liệu từ Kinect

Để khảo sát ảnh hưởng của việc giảm mẫu và chọn bán kính tính PFH, ta chạy chương trình với các kích thước voxel grid (p) và bán kính tính PFH (r) khác nhau. Đầu tiên, ta giữ nguyên tỉ lệ giảm mẫu và thay đổi bán tính tính PFH với các giá trị r lần lượt là 20, 30, 35 và 40mm. Kết quả thu được trong hình 5.8.

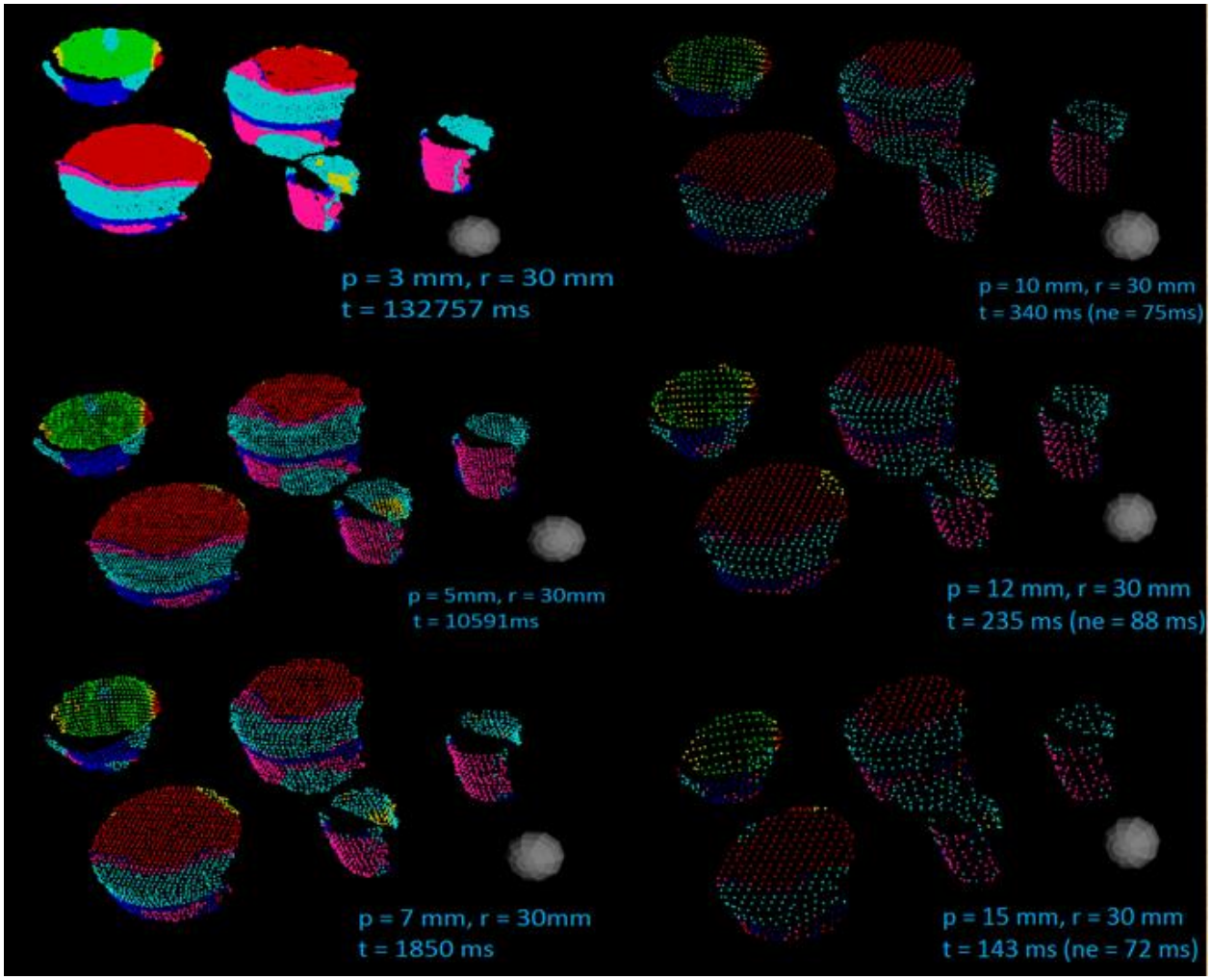


Hình 4.8: Kết quả thử nghiệm với các giá trị r khác nhau

Khi giảm bán kính tính PFH xuống còn 20mm, ta thấy thời gian thực hiện giảm rõ rệt xuống còn ¼. Tuy nhiên với bán kính nhỏ, các lân cận không đủ để thể hiện đặc trưng hình học của bề mặt, điều đó dẫn đến sai số tại các điểm rìa tăng lên đáng kể.

Khi tăng bán kính PFH trên 30mm, thời gian thực hiện cũng tăng thêm nhưng chất lượng không được cải thiện nhiều. Với các bề mặt có kích thước lớn (các mặt phẳng và mặt cầu trong hình), việc tăng bán kính tính PFH có cải thiện độ chính xác tại các điểm rìa. Tuy nhiên với các bề mặt có kích cỡ nhỏ hơn như mặt trụ của cốc hay các phần tiếp giáp giữa hai bề mặt, số điểm nhận dạng sai tăng lên.

Để khảo sát ảnh hưởng của giảm mẫu tới hiệu năng chương trình, ta giữ nguyên bán kính r = 30mm (là giá trị tốt nhất theo khảo sát ở trên) và thay đổi kích thước voxelgrid (p) trong quá trình giảm mẫu. Kết quả được cho trong hình 4.9



Hình 4.9: Kết quả thử nghiệm với các giá trị p khác nhau

Ảnh hưởng rõ ràng nhất khi thay đổi kích thước lưới lọc voxel grid là thời gian tính toán. Với kích cỡ lưới lọc 3mm, đám mây điểm chứa nhiều điểm hơn và mật độ điểm cao hơn, thời gian tính toán bị tăng lên tới 130 giây – rất lớn khi xử lý một ảnh quét từ Kinect. Ngược lại khi lưới lọc có kích thước lớn, tương đương với việc giảm mẫu mạnh hơn thì thời gian tính toán cũng giảm đáng kể. Cụ thể với kích thước voxel bằng 12 và 15mm, thời gian tính toán toàn bộ chỉ còn 235 và 143 ms, trong đó đã bao gồm 88 và 72 ms dành cho việc ước lượng véc tơ pháp tuyến. Tức là thời gian tính toán đặc trưng PFH chỉ còn tương ứng 147 và 71 ms.

Song song với việc giảm thời gian tính toán, khi giảm mẫu nhiều hơn thì đặc trưng PFH vẫn đủ mạnh để nhận diện các bề mặt. Ta thấy trong trường hợp kích thước lưới lọc tăng đến 15mm thì đặc trưng bề mặt hình học vẫn không thay đổi so với kích thước 5mm. Đây là một ưu thế lớn khi sử dụng trong các ứng dụng thời gian thực, khi thời gian tính toán là một vấn đề cốt lõi.

Kết quả được tổng hợp trong bảng dưới đây

Bảng 4.2: Kết quả với các giá trị p và r khác nhau

Kích thước voxel $p = 5mm$		**Bán kính PFH $r = 30\ mm$**	
Bán kính PFH	**Thời gian tính toán**	**Kích thước voxel**	**Thời gian tính toán**
20 mm	2476 ms	3 mm	132757 ms
30 mm	10591 ms	5 mm	10591 ms
35 mm	19035 ms	7 mm	1850 ms
40 mm	31960 ms	10 mm	340 ms
		12 mm	235 ms
		15 mm	143 ms

Chương 5: Kết luận

5.1. Kết luận

Luận văn đã trình bày phương pháp nhận diện bề mặt các vật thể sử dụng đám mây điểm thu thập từ camera RGB-D. Các bước thực hiện gồm có tiền xử lý bao gồm phân đoạn, giảm mẫu, lọc bớt nhiễu. Dữ liệu sau đó được tính toán trích xuất đặc trưng điểm và dùng mô hình SVM để nhận diện, phân loại bề mặt vật thể theo các bề mặt hình học trong không gian 3D như mặt cầu, mặt trụ, mặt phẳng và cạnh.

Chương trình được viết trên ngôn ngữ C++ và thử nghiệm với dữ liệu đám mây điểm không nhiễu và có nhiễu. Kết quả chương trình có thể nhận dạng được các bề mặt đã biết. Luận văn cũng đã trình bày kết quả thử nghiệm, đánh giá về hiệu năng của chương trình khi thay đổi các thông số trong quá trình xử lý.

5.2. Hạn chế và hướng phát triển

Do thời gian có hạn, luận văn vẫn chưa xây dựng được một hệ thống hoàn chỉnh có thể nhận diện, phân loại vật thể sử dụng camera RGB-D. Ngoài ra, giải thuật vẫn còn tồn tại một số vấn đề như thời gian xử lý cao, chưa nhận diện chính xác với các bề mặt nhiều nhiễu.

Trong thời gian tới, tôi đề xuất các hướng phát triển tiếp theo như sau:

- Hoàn thiện chương trình với một số giải thuật thích nghi với kích thước của vật, khoảng cách từ vật đến camera, …
- Nghiên cứu, thử nghiệm hệ thống nhận diện, phân loại vật thể với các đặc tính toàn thể như GFPFH, VFH, …
- Nghiên cứu, tìm hiểu và thử nghiệm các cách tiếp cận khác đối với bài toán nhận diện và phân loại vật thể sử dụng camera RGB-D.

Tài liệu tham khảo

1. Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Michael Beetz, *Learning Informative Point Classes for the Acquisition of Object Model Maps*, Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV), Hanoi, Vietnam, December 17-20, 2008

2. E. Wahl, U. Hillenbrand, and G. Hirzinger, *Surflet-Pair-Relation Histograms: A Statistical 3D-Shape Representation for Rapid Classification*, in 3DIM03, 2003, pp. 474–481

3. Radu Bogdan Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, PhD. Thesis, Institure of Informatic, Technical University of Munich.

4. Annalisa Barla1;2, Francesca Odone2, Alessandro Verri, *Histogram Intersection Kernel for image classification*, Proceedings of International Conference on Image Processing (ICIP), 2003.

5. Dustin Boswell, *Introduction to Support Véc tơ Machines*, www.dustwell.com/PastWork/IntroToSVM.pdf, August 2002.

6. Chih-Chung Chang, Chih-Jen Lin, *LIBSVM: A library for Support Véc tơ Machine*, 2001.

7. Radu Bogdan Rusu and Steve Cousins, *3D is here: Point Cloud Library (PCL)*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA ’11), Shanghai, China, May 2011

8. Andreas Richtsfeld, Thomas Morwald, Johann Prankl, Michael Zillich and Markus Vincze, *Segmentation of Unknown Objects in Indoor Environments*, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems

9. M.A. Fischler and R.C. Bolles. *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, 24(6):381–395, 1981.

10. Klaas Klasing, Daniel Althoff, Dirk Wollherr, Martin Buss, *Comparison of Surface Normal Estimation Methods for Range Sensing*, Applications 2009

IEEE International Conference on Robotics and AutomationKobe International Conference CenterKobe, Japan, May 12-17, 2009.

11. Alicja Wasik, Jose N. Pereira, Rodrigo Ventura, Pedro U. Lima and Alcherio Martinoli, *Graph-Based Distributed Control for Adaptive Multi-Robot Patrolling through Local Formation Transformation*, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, Korea, October 9-14, 2016.

12. Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, John Hsu , *Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram*, Intelligent Robots and Systems (IROS), 2010.

13. A. Aldoma, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, M. Vincze,and G. Bradski, *CAD-model recognition and 6 DOF pose estimationusing 3D cues*, in Proc. ICCV 2011, 3D Representation and Recognition(3D RR11), Barcelona, Spain, 2011, pp. 585–592.

14. R. B. Rusu, N. Blodow, and M. Beetz. *Fast PointFeature Histograms (FPFH) for 3D Registration*. In Proceedings of the International Conference onRobotics and Automation (ICRA), 2009

15. R.B. Rusu, A. Holzbach, M. Beetz. *Detecting and Segmenting Objects for Mobile Manipulation*, in the S3DV Workshop of the 12th International Conference on Computer Vision (ICCV), 2009.