ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ**

---------------o0o---------------

****

**ĐỀ CƯƠNG LUẬN VĂN TỐT NGHIỆP**

**NGHIÊN CỨU**

**XE LĂN TỰ HÀNH TRONG NHÀ**

**GVHD: TS. NGUYỄN LÝ THIÊN TRƯỜNG**

**SVTH: NGUYỄN DUY LINH**

**MSSV: 1711957**

**TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2021**

***LỜI CẢM ƠN***

*Tp. Hồ Chí Minh, ngày tháng năm .*

**Sinh viên**

**MỤC LỤC**

[1. GIỚI THIỆU 1](#_Toc78112002)

[1.1 Tổng quan 1](#_Toc78112003)

[1.2 Tình hình nghiên cứu trong và ngoài nước 2](#_Toc78112004)

[1.3 Mục tiêu đề tài 4](#_Toc78112005)

[2. NỘI DUNG ĐỀ TÀI 5](#_Toc78112006)

[3. GIẢI PHÁP THỰC HIỆN 5](#_Toc78112007)

[4. DỰ KIẾN KẾT QUẢ ĐẠT ĐƯỢC 8](#_Toc78112008)

[4.1 Kết quả sơ khởi đã đạt được 8](#_Toc78112009)

[4.1.1 Nội dung 1 8](#_Toc78112010)

[4.1.2 Nội dung 2 20](#_Toc78112011)

[4.1.3 Nội dung 3 20](#_Toc78112012)

[4.1.3.1 ROS Filesystem Level 21](#_Toc78112013)

[*4.1.3.2* *ROS Computation Graph Level* 22](#_Toc78112014)

[4.1.3.3 ROS *Community Level* 23](#_Toc78112015)

[4.1.3.4 RVIZ 24](#_Toc78112016)

[4.1.3.5 TF Message 24](#_Toc78112017)

[4.1.4 Nội dung 5 24](#_Toc78112018)

[4.2 Kết quả dự kiến đạt được 25](#_Toc78112019)

[4.2.1 Nội dung 4 25](#_Toc78112020)

[4.2.2 Nội dung 6 25](#_Toc78112021)

[5. KẾ HOẠCH THỰC HIỆN 26](#_Toc78112022)

[6. TÀI LIỆU THAM KHẢO 26](#_Toc78112023)

**DANH SÁCH HÌNH**

[*Hình 1‑1 Minh hoạ khả năng nhận biết vật thể trong môi trường phức tạp của xe tự hành Tesla.* 1](#_Toc78111982)

[Hình 1‑2 Phân loại các giải thuật Path Planning [4]. 4](#_Toc78111983)

[Hình 3‑1 Sơ đồ tổng quan hệ thống phần cứng của đề tài. 8](#_Toc78111984)

[Hình 4‑1 Tập hợp partiles thể hiện một phân bố xác xuất ngẫu nhiên. 14](#_Toc78111985)

[Hình 4‑2 Minh hoạ phân bố xác xuất của vị trí robot qua các bước. Biến k đại diện cho lần lặp thứ k, thay cho t. [11] 15](#_Toc78111986)

[Hình 4‑3 Một số features giống nhau xuất hiện trong hai ảnh khác nhau. 16](#_Toc78111987)

[Hình 4‑4 Minh hoạ toạ độ camera. [9] 17](#_Toc78111988)

[*Hình 4‑5 Minh hoạ vị trí 2 camera và điểm P. [9]* 18](#_Toc78111989)

[Hình 4‑6 Minh hoạ cho quá trình reprojection.[10] 19](#_Toc78111990)

[Hình 4‑7 Cấp quản lý Filesystem của ROS. 21](#_Toc78111991)

[Hình 4‑8 Các thành phần của cấp Computation Graph 22](#_Toc78111992)

**DANH SÁCH BẢNG**

[Bảng 4‑1 Pseudo code của giải thuật Bayes filter. 16](#_Toc78112198)

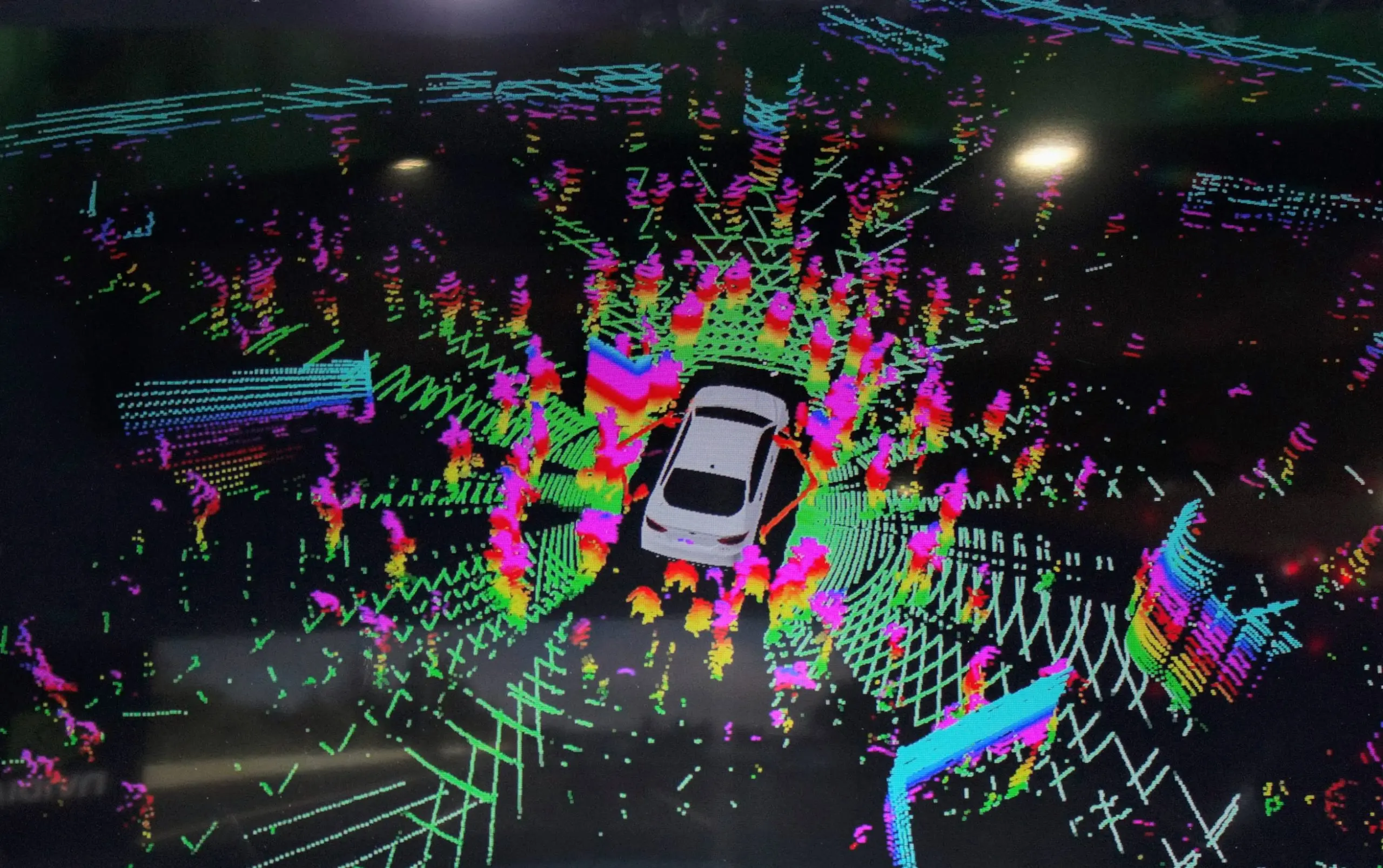
# GIỚI THIỆU

## Tổng quan

Mô tả tổng quan về lĩnh vực liên quan đến đề tài và những mục tiêu cần nghiên cứu. Từ đó giới thiệu mục tiêu cần đặt ra cho đề tài.

Trang thiết bị tự động được nghiên cứu và phát triển từng ngày để hỗ trợ con người trong sinh hoạt, sản xuất. Chúng ngày càng thông minh để thoả mãn được nhu cầu mạnh mẽ mà chúng ta đặt ra. Bây giờ, các thiết bị tự động đã có thể tự thực hiện những phần công việc độc hại, khó khăn thay con người một cách chính xác và năng suất hơn bao giờ hết. Hơn thế nữa, chúng có khả năng thực hiện cả những công việc mà con người không thể. Công nghệ không ngừng phát triển, mở rộng khả năng của máy móc đồng thời cũng kéo theo viễn cảnh lao động con người bị thay thế hoàn toàn.

Để đạt được tới viễn cảnh đó, trước hết, máy móc cần phải tự mình di chuyển mà không cần sự điều khiển từ con người, hay nói các khác là nó phải có khả năng tự hành. Đây là khả năng cơ bản nhất mà máy móc cần có để thực sự thế chân con người. Nhưng thực tế chưa có nghiên cứu nào đặt ra khả năng tự hành hoàn toàn ổn định trong mọi điều kiện vận hành cho máy móc. Dẫu vậy lĩnh vực nghiên cứu phương tiện tự hành vẫn đạt được nhiều thành tựu lớn. Những nỗ lực tiêu biểu trong nghiên cứu và ứng dụng khả năng tự hành là hệ thống xe tự lái của Tesla và robot Perseverance của NASA. Ngoài ra, rất nhiều robot nhỏ với khả năng tự hành trong môi trường đặc thù đang được ứng dụng hiệu quả như nhiều robot hút bụi trong gia đình, đã được thương mại hoá, hoặc robot kho bãi và giao hàng của Alibaba.



*Hình 1‑1 Minh hoạ khả năng nhận biết vật thể trong môi trường phức tạp của xe tự hành Tesla.*

Bên cạnh việc nghiên cứu phát triển các thiết bị tự hành thay thế lao động con người, cũng có những nghiên cứu thiết bị như xe lăn tự hành hỗ trợ người khuyết tật về thị lực, hỗ trợ họ di chuyển dễ dàng hơn trong môi trường phức tạp hoặc không quen thuộc. Đề tài cũng chọn hướng ứng dụng như vậy. Mục tiêu là thiết kế một thiết bị tương tự xe lăn có khả năng tự hành với điều kiện môi trường trong nhà. Vậy nên, đề cương luận văn này sẽ nghiên cứu, đánh giá và đưa vào ứng dụng một số giải thuật giải quyết bài toán lớn gồm 3 vấn đề sau đây:

1. Làm sao để thiết bị có khả năng tự lập bản đồ không gian xung quanh (Mapping).
2. Làm sao để thiết bị định vị được chính nó trong không gian đó (Localization).
3. Làm sao để thiết bị có thể tự lên kế hoạch di chuyển đến điểm cần đến. (Path Planning).

Bài toán lớn này là bài toán chung của lĩnh vực nghiên cứu thiết bị/robot tự hành. Trong phần tiếp theo, tình hình nghiên cứu tìm ra giải pháp cho chúng sẽ được trình bày.

## Tình hình nghiên cứu trong và ngoài nước

Các nghiên cứu phương tiện tự hành đều triển khai giải thuật làm cả hai việc: lập bản đồ và định vị thiết bị, cùng một lúc. Cái tên chung cho các giải thuật giải quyết bài toán liên quan đến hai vấn đề đầu tiên (trong ba vấn đề đã nêu ở mục 1.1) là **S**imultaneous **L**ocalization **A**nd **M**apping, hay gọi tắt là **SLAM**. Nhìn chung, các giải thuật giải quyết bài toán chia làm 2 hướng tiếp cận chính:

1. Dùng cảm biến khoảng cách để thu thập dữ liệu về khoảng cách gữa thiết bị tới nhiều điểm trong không gian xung quanh.
2. Dùng camera như cảm biến chính để thu thập nhiều loại thông tin về môi trường xung quanh. Hình ảnh từ một hay nhiều camera được dùng để trích xuất thông tin cần thiết để triển khai các phần của giải thuật.

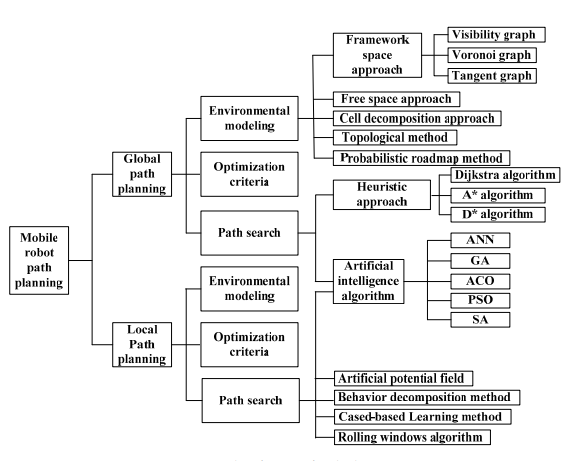
Mỗi cách tiếp cận đều có ưu điểm và nhược điểm riêng.

Nhóm giải thuật thứ nhất là những giải thuật SLAM cơ bản nhất và đã được nghiên cứu từ rất lâu. Dữ liệu đầu vào của các giải thuật loại này là khoảng cách tới các điểm không gian, một mặt phẳng 2D, vốn là dữ liệu trực tiếp và cần thiết nhất cho tác vụ lập bản đồ và định vị. Những giải thuật 2D-SLAM, vì vậy, không phải là tác vụ quá nặng so với các vi xử lý hiện có. Do chúng đều dùng dữ liệu ở dạng 2D nên có thể được gọi chung là các giải thuật 2D-SLAM. 2D-SLAM lại có nhiều hướng tiếp cận như Grid-based, Graph-based hoặc Feature-based. Grid-based SLAM vốn đã được ứng dụng thực tế trong gói phần mềm Gmapping (bởi nhóm nghiên cứu của Giorgio Grisetti [1]) được tích hợp trong hệ điều hành cho robot - ROS. 2D-SLAM dùng các loại cảm biến khoảng cách có thể kể tới là: RADAR, LiDAR, ToF. Tuy nhiên những loại cảm biến kể trên thường phức tạp, ít thông dụng và phải có tính chính xác nhất định nên giá thành là một rào cản để có thể đưa vào ứng dụng rộng rãi.

Các giải thuật ở nhóm thứ hai thì ngược lại. Camera thường là cảm biến chính được dùng và giá thành camera thì rất hợp lý, do độ phổ biến cao. Do giải thuật dùng dữ liệu ảnh trực quan nên được gọi là Visual SLAM hay V-SLAM. Các giải thuật V-SLAM, đầu tiên, phải xuất những thông tin cần thiết từ ảnh. Thông tin thường là khoảng cách tới cách điểm trong hình ảnh thu được và vị trí của chúng so với camera. Khác biệt giữa V-SLAM so với 2D-SLAM là vị trí của các điểm so với camera là vị trí trong không gian 3D, thay vì một mặt cắt 2D so với 2D-SLAM. Thiết bị sẽ thu được nhiều thông tin về không gian xung quanh hơn. Số lượng điểm cần để giải thuật tính toán do vậy cũng trở nên rất nhiều để có được lượng thông tin cần thiết. Trong khi đó, các phương pháp trích xuất khoảng cách từ ảnh cũng đòi hỏi phải trải qua nhiều bước dẫn đến khối lượng xử lý rất lớn khi triển khai giải thuật. Nhưng, vi xử lý ngày càng mạnh hơn theo quy luật Moore và giải thuật thì luôn có thể cải tiến, V-SLAM vẫn đang là lĩnh vực nghiên cứu năng động. Hai nghiên cứu tiêu biểu có thể kể tới là giải thuật ORB-SLAM 3 (do nhóm nghiên cứu của Carlos Campos [3]) và RTAB-MAP (do nhóm nghiên cứu của Mathieu Labbé [4]). Riêng RTABMAP là giải thuật đã được ứng dụng và được nhóm tác giả đóng thành gói phần mềm của hệ điều hành cho robot – ROS.

So sánh giữa 2D-SLAM và V-SLAM, 2D-SLAM đang là hướng tiếp cận phổ biến và đã được ứng dụng, thương mại hoá. Lĩnh vực nghiên cứu 2D-SLAM có thể nói đã đạt được những thành tựu tương đối. Trong khi đó, V-SLAM phần nhiều vẫn còn là những giải thuật nghiên cứu mang tính lý thuyết, chưa được sử dụng rộng như 2D-SLAM. Nhưng V-SLAM lại rất hứa hẹn khi dữ liệu ảnh mang được rất nhiều thông tin trực quan, có thể đem lại cho thiết bị khả năng ứng dụng rộng hơn nhờ những nghiên cứu mới của lĩnh vực Computer Vision. Và cũng do vậy mà lĩnh vực nghiên cứu V-SLAM vẫn còn nhiều khả năng để phát triển các thuật toán tốt hơn nữa. Đồng thời lĩnh vực này cũng tạo ra thách thức để những kỹ sư có thể ứng dụng thành quả nghiên cứu V-SLAM vào những ứng dụng thực tế thay thế 2D-SLAM.

Bài toán SLAM giải quyết được hai vấn đề đầu tiên là lập bản đồ (mapping) và định vị (localization). Thiết bị phải có khả năng giải quyết thêm bài toán thứ hai là lên kế hoạch di chuyển (path planning) thì mới đạt được khả năng tự hành. Lĩnh vực nghiên cứu cứu thuật toán tìm đường đi tối ưu nhất thì đã có nhiều nghiên cứu tương đối hoàn thiện. Thành quả nghiên cứu trong lĩnh vực từ lâu đã được ứng dụng rộng rãi trong nhiều lĩnh vực và vẫn còn được phát triển. Các giải thuật đã có thể lên kế hoạch di chuyển trong nhiều môi trường từ đơn giản đến phức tạp. Có hai hướng nghiên cứu cho bài toán tìm đường đi tối ưu dựa trên hai mục đích: global navigation và local navigation. Khả năng dựa vào vị trí hiện tại và vị trí đích đến trong bản đồ có sẵn để lên kế hoạch di chuyển tới điểm đến gọi là global navigation. Trong khi đó, local navigation giúp thiết bị có khả năng ứng biến khi có thay đổi xảy ra trên đường đi trong quá trình thực hiện global navigation. Những giải thuật cho global navigation tiêu biểu có thể kể đến như giải thuật Dijkstra cùng cải tiến như giải thuật A\*. Giải thuật RRT (Rapidly-Exploring Random Trees) và giải thuật cải tiến RRT\* cũng được ứng dụng cho mục đích global navigation nhờ ưu điểm nhanh và khối lượng tính toán nhẹ. Mỗi giải thuật sẽ đặc biệt hiệu quả trong mỗi môi trường đặc thù khác nhau. Đối với local navigation thì nhiều giải thuật như Dynamic Window Approach (DWA), D\*, ..



Hình 1‑2 Phân loại các giải thuật Path Planning [4].

## Mục tiêu đề tài

Nhận thấy cơ hội và thách thức trong khả năng ứng dụng V-SLAM, đề tài quyết định chọn hướng tiếp cận này.

Về mục tiêu lớn, đề tài sẽ xem xét, lựa chọn những phương án thiết kế một nên hệ thống để triển khai các giải thuật lý thuyết sẵn có. Từ đó lựa chọn phương án tốt nhất, phù hợp nhất để ứng dụng vào thi công một thiết bị có khả năng tự hành trong nhà. Thiết bị phải đạt được khả năng di chuyển từ điểm đầu tới điểm đích định sẵn và có khả năng ứng phó với các vật cản tĩnh và động.

Từ mục tiêu lớn đó, đề tài xác định các mục tiêu nhỏ như sau:

* Nghiên cứu lý thuyết toán xác suất nền tảng để giải quyết bài toán SLAM nói chung và lý thuyết thị giác máy tính/xử lý ảnh được dùng trong các giải thuật Visual SLAM có tiềm năng ứng dụng cao nhất.
* Tìm hiểu lý thuyết về các thuật toán tìm đường đi tối ưu (optimal path planning) thông dụng để chọn ra phương án thực hiện global navigation và local navigation.
* Tìm hiểu cấu trúc Robotic Operating System (ROS) và cách phát triển một hệ thống dùng ROS để triển khai giải thuật SLAM và giải thuật path planning.
* Mô phỏng và đánh giá trước các giải thuật với ROS để đánh giá và chọn ra một giải thuật có thể ứng dụng.
* Thiết kế và thi công một thiết bị phần cứng để có thể triển khai hệ thống.
* Triển khai thuật toán trên thiết bị thực tế và đánh giá kết quả.

# NỘI DUNG ĐỀ TÀI

Từ mục tiêu đề tài đặt ra, sinh viên triển khai thành các nội dung của luận văn.

**Nội dung 1:** Tìm hiểu lý thuyết toán xác suất giải quyết bài toán SLAM và lý thuyết về xử lý ảnh, thị giác máy tính được dùng trong các giải thuật Visual SLAM.

**Nội dung 2:** Tìm hiểu về các thuật toán tìm đường đi tối ưu cho global navigaton và local navigation.

**Nội dung 3:** Tìm hiểu về cấu trúc của ROS và lý thuyết về các khối chức năng/cơ chế thông tin của ROS.

**Nội dung 4:** Phác hoạ thiết kế các khối phần mềm để chạy trên ROS.

**Nội dung 5:** Thực hiện mô hình.

**Nội dung 6:** Triển khai hệ thống phần mềm lên phần cứng thực tế đã thực hiện và đánh giá kết quả đạt được so với mục tiêu ban đầu đã đề ra.

# GIẢI PHÁP THỰC HIỆN

Mô tả giải pháp thực hiện cho từng nội dung đã đề ra.

**Nội dung 1:** Tìm hiểu lý thuyết toán xác suất giải quyết bài toán SLAM và lý thuyết về xử lý ảnh, thị giác máy tính được dùng trong các giải thuật Visual SLAM.

Cụ thể, tìm hiểu các lý thuyết nền trong lĩnh vực Probabilistic Robotics như Bayesian Filter, Kalman Filter, Extended Kalman Filter, EKF Localization…trong bài toán SLAM. Các nội dung lý thuyết có thể tham khảo từ sách Probabilistic Robotics [4].

Sau đó, tìm hiểu lý thuyết xử lý hình ảnh được dùng trong các giải thuật V-SLAM như ORB Feature [5] dùng trong ORB-SLAM [3] hoặc SIFT/SURF Feature [6] dùng trong RTAB-MAP [4]. Đồng thời là các lý thuyết đặc thù thường dùng trong nhiều giải thuật Visual SLAM như Key-frame, Triangulation, Bundle Adjustment, Loop Closure Detection…

ORB-SLAM và RTAB-MAP là hai giải thuật Visual SLAM tốt nhất và đa dụng nhất hiện tại. Đề tài sẽ lựa chọn một trong hai giải thuật để ứng dụng. Vì vậy lý thuyết về hai giải thuật này cũng sẽ được tìm hiểu.

**Nội dung 2:** Tìm hiểu về các thuật toán tìm đường đi tối ưu như Dijkstra, A\*, RRT\* cho global navigation và Dynamic Window Approach, D\* cho local navigation.

Bài báo [4] là một bài báo review tổng quan về lĩnh vực nghiên cứu giải thuật path planning cho mobile robot. Dựa vào bài báo [4] ta có thể có cái nhìn sơ lược về ưu điểm và nhược điểm của mỗi thuật toán. Từ đó ta có thể chọn ra một số thuật toán có khả năng ứng dụng cao nhất, phổ biến nhất để tìm hiểu sâu hơn và đưa vào ứng dụng.

**Nội dung 3:** Tìm hiểu về cấu trúc của ROS và lý thuyết về các khối chức năng/cơ chế thông tin của ROS.

ROS - Robotic Operating System là một gói nhiều phần mềm nguồn mở có chức năng như một hệ điều hành cho robot. Chính vì là phần mềm nguồn mở nên source code và tài liệu hướng dẫn của ROS được các tác giả soạn thảo đầy đủ và công khai tại trang web [http://wiki.ros.org/ROS/Tutorials](http://wiki.ros.org/vn/ROS/H%C6%B0%E1%BB%9Bng%20d%E1%BA%ABn). Đề tài chọn ROS làm nền tảng để triển khai ứng dụng nên đây sẽ là nguồn tài liệu tham khảo chính trong quá trình phát triển phần mềm cho thiết bị.

**Nội dung 4:** Phác hoạ thiết kế các khối chức năng để chạy trên ROS.

Các khối chức năng là các phần mềm được phát triển thành các node chức năng theo cấu trúc của ROS. Nhiệm vụ của các node chủ yếu là xử lý thông tin thô thành thông tin đầu vào cho SLAM và path planning. Chúng cũng xử lý thông tin đầu ra của giải thuật để áp dụng lên phần cứng thực tế. Những node này thực hiện nhiệm vụ như:

* Node xử lý dữ liệu encoder gửi lên từ MCU để tính toán.
* Node tính toán dữ liệu đầu vào cho giải thuật: coordinate transform, wheel odometry, IMU data filtering, ...
* Node xử lý lệnh điều khiển vận tốc nhận được từ giải thuật path planning.

Các node khởi tạo topic, subscribe và publish dữ liệu qua cơ chế truyền thông rosserial. Mỗi topic sẽ có định dạng gói thông tin khác nhau.

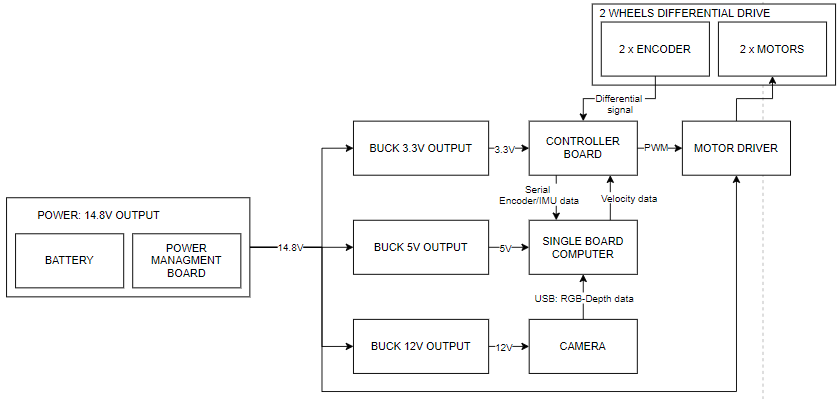
**Nội dung 5:** Tìm hiểu lý thuyết phần cứng và thiết kế mô hình.

Cần thiết phải thi công thiết bị phần cứng để đưa giải pháp (bao gồm sự kết hợp của các giải thuật lý thuyết và hệ thống tự thiết kế) vào ứng dụng mới có thể đánh giá tính hiệu quả của giải pháp trong thực tế so trong nghiên cứu.

Đề tài sẽ thi công một thiết bị có bánh xe mô phỏng theo xe lăn. Thiết bị được điều khiển theo cơ chế lái vi sai.

Về camera cho Visual SLAM, đề tài có 3 lựa chọn để triển khai: monocular camera (1 camera RGB), stereo camera (cặp camera RGB) và RGB-D camera (1 camera RGB + cảm biến chiều sâu như ToF). Ba lựa chọn hiện có cũng là 3 hướng tiếp cận của hướng nghiên cứu Visual SLAM. Những giải thuật Visual SLAM đa phần đều hỗ trợ 2-3 lựa chọn trên. Đề tài nhận thấy những nghiên cứu theo hướng monocular camera vẫn còn một số vấn đề chưa thể giải quyết bằng lý thuyết (vấn đề sai số tích luỹ của phép tính khoảng cách và vị trí). Trong khi đó, so sánh giữa 2 hướng còn lại, việc dùng camera stereo cho phép đo khoảng cách có sai số không tối ưu bằng phép đo khoảng cách bằng cảm biến ToF từ RGB-D. Ngược lại, cảm biến ToF trên RGB-D camera dựa trên hồng ngoại nên bị giới hạn khoảng cách đo và nhạy cảm với ánh sáng mặt trời cường độ cao. Những đặc điểm trên cho thấy: stereo camera thích hợp cho ứng dụng SLAM trong môi trường ngoài trời rộng lớn và RGB-D camera phù hợp cho các ứng dụng trong nhà. Một ưu điểm khác của RGB-D camera là chúng có thể tính toán khoảng cách chính xác ngay cả trong các góc tối nhất của môi trường nhờ cảm biến ToF, trong khi các giải thuật cho RGB camera tỏ ra rất nhạy cảm với ánh sáng môi trường. Vì vậy đề tài chọn RGB-D camera vì những ưu điểm trên, đặc biết là tính phù hợp với môi trường mục tiêu của đề tài - môi trường trong nhà.

Từ lựa chọn trên, đề tài đặt ra nhiệm vụ thiết kế một hệ thống phần cứng bao gồm MCU, Motors, IMU, Motor Driver, Pin, Máy tính nhúng và Camera RGB-D theo những yêu cầu ban đầu. Sơ đồ khối tổng quan của hệ thống phần cứng được phát thảo như dưới đây:



Hình 3‑1 Sơ đồ tổng quan hệ thống phần cứng của đề tài.

**Nội dung 6:** Triển khai hệ thống phần mềm lên phần cứng thực tế đã thực hiện và đánh giá kết quả đạt được so với mục tiêu ban đầu đã đề ra.

Từ những kết quả là phần mềm và phần cứng đã được hoàn thành từ những nội dung trước, công việc cuối cùng là kết nối chúng lại và đánh giá chi tiết khả năng đáp ứng yêu cầu ban đầu đề ra.

# DỰ KIẾN KẾT QUẢ ĐẠT ĐƯỢC

## Kết quả sơ khởi đã đạt được

## Nội dung 1

***Các lý thuyết cơ bản lĩnh vực Probabilistic Robotics:***

Lĩnh vực *Probabilistic Robotics* là lĩnh vực nghiên cứu phát triển nhận thức, hành động của robot dựa trên lý thuyết xác suất. Nhận thức của robot về môi trường xung quanh thường dựa trên thông tin từ cảm biến, vốn bị ảnh hưởng bởi nhiễu. Vì vậy, lý thuyết xác xuất được ứng dụng và biểu diễn thông tin thành một dạng phân phối xác xuất để thể hiện mức độ không chắc chắn của thông tin do nhiễu. Dựa vào đó robot có thể tính toán và đưa ra quyết định linh hoạt hơn nhờ dự đoán được các khả năng có thể xảy ra do sực không chắc chắn đó. Hơn thế nữa, khi robot hành động nhờ vào thông tin dạng xác suất sẽ giảm được sự ảnh hưởng từ nhiều nguồn nhiễu [1, p.3].

Quy ước trình bày trong báo cáo về ký hiệu toán như sau:

Với một biến ngẫu nhiên xác định trong miền D, có không gian mẫu là Ω ↦D. là giá trị cụ thể nằm trong không gian mẫu Ω. Thì hàm phân bố xác xuất của xác xuất để biến cố xảy ra sẽ là sao cho:

|  |  |  |
| --- | --- | --- |
|  |  | (0) |

mang đủ tính chất của xác xuất:

**Khái niệm cơ bản** ***Probabilistic Robotics***

1. State: Tất cả yếu tố liên quan đến môi trường xung quanh robot và bản thân robot là *state*. Chính vì vậy state bị ảnh hưởng bởi rất nhiều biến ngẫu nhiên trong và ngoài hệ thống. State cũng được xem là môt biến ngẫu nhiên, ký hiệu . Các biến cố của được kí hiệu là và biến cố xảy ra ở thời điểm sẽ là . Do robot thường là các hệ thống số nên rời rạc kéo theo state cũng không liên tục và các biến cố ảnh hưởng đến state cũng xảy ra không liên tục.

Một state gọi là hoàn chỉnh khi là dự đoán tốt nhất và các biến ngẫu nhiên ảnh hưởng đến trong quá khứ không giúp dự đoán tốt hơn trong tương lai. Khái niệm này được đưa ra để đơn giản hoá các xác xuất điều kiện được sử dụng trong *Probabilistic Robotics*.

Trong báo cáo, thuật ngữ *trạng thái* được sử dụng thay cho thuật ngữ *state* trong một số trường hợp để dễ hiểu hơn.

1. Measurement: Measurement còn được đề cập là Observation hoặc Percept trong các bài báo để chỉ thông tin về trạng thái môi trường xung quanh mà robot thu thập được từ sensor. Mesurement cũng là biến ngẫu nhiên, được ký hiệu là . Các biến cố của sẽ là . ở dạng rời rạc theo thời gian và thường được kí hiệu là .
2. Control: Control là thông tin về sự thay đổi của môi trường xung quanh do robot chủ động tác động. Ví dụ như robot di chuyển với tốc độ 10 cm/s trong khoảng thời gian 5 giây thì trạng thái mới của robot sẽ cách trạng thái cũ 50 cm. Vậy nên, về bản chất, control mang thông tin về sự thay đổi của state. Ký hiệu thông dụng dùng cho biến control là U và biến cố theo thời gian là là .
3. Belief: Belief là thông tin về state mà robot suy ra dựa trên dữ liệu mà nó đang có. Phân phối xác xuất của belief thể hiện những khả năng có thể xảy ra với trạng thái thực. Belief tồn tại dưới dạng xác xuất hậu nghiệm dựa trên tiền nghiệm là measurement và control:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

là kí hiệu của belief của . Về mặt ý nghĩa, chính là ước lượng của state dựa trên measurement và control trong quá khứ, kèm theo là control tại thời điểm : . Ước lượng của được thực hiện trước khi thu được mesurement tại thời điểm (). Một khi đã biết , sẽ được chính xác hoá lại thành . chính là ước lượng hoàn chỉnh của robot về state tại thời điểm .

**Bayes Filter**

Ý tưởng về belief trong *Probabilistic Robotics* dựa trên nguyên lý của Bayes Filter.

Bayes filter dựa trên định lý Bayes cho xác xuất có điều kiện, rằng:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

Nếu xem là state và là measurement thì công thức trên cho thấy ta có thể tính được xác xuất của state khi đã có measurement (dựa trên xác xuất thu được measurement tại state ). Dữ liệu đi kèm theo là xác xuất tiền nghiệm , đại diện cho mọi ước lượng về state trước khi measurement được thực hiện. chỉ đóng vai trò chuẩn hoá cho phép nhân ở tử. Vậy nên có thể nói, quá trình tính xác xuất hậu nghiệm thực tế là quá trình chính xác hoá belief trước đó là dựa trên xác xuất thu được measurement tại state .

Thực tế, state bị ảnh hưởng bởi nhiều hơn một biến ngẫu nhiên. Thật vậy, trong lý thuyết *Probabilistic Robotics* tính tới hai yếu tố measurement và control nên định lý Bayes cho 3 biến ngẫu nhiên sẽ được quan tâm. Cụ thể, biến ngẫu nhiên Z có biến cố là thì:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Công thức trên suy ra từ định lý Bayes cho 2 biến và được ứng dụng cho như sau:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
|  | |  | (4) |

Trong đó: là đóng vai trò là hệ số chuẩn hoá.

Nếu giả thuyết state tại thời điểm t là hoàn chỉnh, đồng nghĩa là hoàn chỉnh thì xác xuất dựa trên sẽ không phụ thuộc vào các biến cố trong quá khứ của các biến ngẫu nhiên khác. Vậy nên:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | = | | (5) | |
|  | |  | |  | |

Phương trình (5) cho thấy xác xuất có được measurement chỉ phụ thuộc vào state , bỏ qua mọi thông tin về measurement và control trong quá khứ. Điều này giúp rút gọn phương trình (4) thành:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Có thể thấy từ (6), chính là xác xuất hậu nghiệm của khi đã thu được measurement và là xác xuất tiền nghiệm khi chưa thu được measurement . Nói cách khác, là và là . Từ đó (6) trở thành:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Trong đó:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Để tìm ra mối quan hệ giữa và state trước đó , ta cần phải đưa vào phương trình (8). Áp dụng tổng xác xuất của trong không gian mẫu là 1 ().

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
|  |  |  | |
|  |  | (9) |

Tiếp tục áp dụng giả thuyết state là hoàn chỉnh, khi đã biết mọi thông tin khác không giúp dự đoán xác xuất tốt hơn, nên ta có:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

Phương trình (10) cho thấy dự đoán chỉ phụ thuộc vào state trước đó và tín hiệu control . Từ (8) (9) và (10) có thể kết luận theo công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

Trong đó: là vì không thể tác động đến state trong quá khứ nên được bỏ qua. Cuối cùng trong Bayes filter được tính bằng công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

Tổng quan lại, từ phương trình (7) và (12), giải thuật của Bayes filter ước lượng state trên dữ liệu measurement và control theo phương pháp hồi quy có thể được thể hiện qua pseudo-code như sau:

|  |  |
| --- | --- |
| 1: | **Algorithm Bayes\_filter**(): |
| 2: | for all do |
| 3: |  |
| 4: |  |
| 5: | endfor |
| 6: | return |

Bảng 4‑1 Pseudo code của giải thuật Bayes filter.

Dựa theo pseudo-code có thể hiểu rằng, Bayes filter dựa trên 2 bước chính:

* + - Dự đoán: là quá trình ước lượng trạng thái , hay ,dựa trên xác xuất hậu nghiệm của (là ) và control .
    - Cập nhật: là quá trình chính xác hoá lại dự đoán thành sau khi đã thu được measurement .

Tuy có thể biểu diễn dễ dàng dưới dạng pseudo code, nhưng Bayes filter về bản chất chỉ là giải thuật trên lý thuyết. Thực tế, hai giải thuật Particle filter và Kalman filter là hai giải thuật hiện thực hoá ý tưởng của Bayes filter, và cũng là hai giải thuật được ứng dụng nhiều nhất cho vấn đề localization của mobile robot.

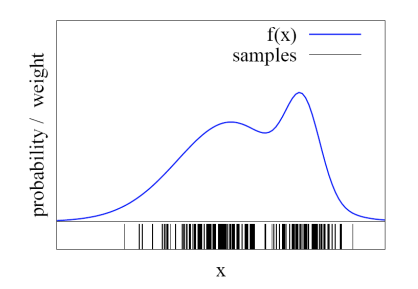
**Particle Filter**

Để ứng dụng Bayesian filter vào giải quyết bài toán dự đoán state của robot có hai phương thức chính.

Truyền thống, Kalman Filter được sử dụng bám theo giả thuyết rằng: các phân bố xác xuất của các biến ngẫu nhiên đều là phân bố Gaussian và các biến ngẫu nhiên thay đổi theo một hàm tuyến tính. Vì thực tế biến ngẫu nhiên thay đổi bất tuyến tính như giả thuyết. Nên bộ lọc cải tiến Extended Kalman filter (EKF) được sử dụng thay và bỏ qua giả tuyết state thay đổi theo hàm tuyến tính. Thay vào đó, EKF dựa trên khai triển Taylor bậc 1 của hàm trạng thái (state). Bộ lọc EKF rất hiệu quả khi áp dụng trong các hệ thống có nhiễu theo phân bố Gaussian. Tuy nhiên, thực tế các phân bố xác xuất là ngẫu nhiên và không phải là phân bố Gaussian (non-gaussian).

Do trên, các bộ lọc Particle filtersinh ra để giải quyết bài toán dự đoán trạng thái trong trường hợp nhiễu phân bố xác xuất nhiễu non-gaussian. Loại bộ lọc này đang được sự dụng rộng rãi do mang nhiều ưu điểm khi ứng dụng thực tế.

Các bộ lọc thuộc loại Particle filter theo cơ chế: lấy một tập hợp hữu hạn mẫu (sampling) ngẫu nhiên từ phân phối xác suất state trước đó. Các mẫu xác xuất là rời rạc nên được gọi là particles. Do đó, từ mật độ của các particles, ta có thể thể hiện lại phân phối xác xuất của state trước đó đó một cách gần đúng. Mỗi particle đi kèm trọng số thể hiện độ tin cậy của nó. Particle có trọng số cao nhất đồng nghĩa với xác xuất xảy ra của particle này là cao nhất. Hình dưới đây thể hiện sự phân bố của các particles (samples) theo hàm phân bố xác xuất f(x) ngẫu nhiên mà chúng đã được lấy mẫu.



Hình 4‑1 Tập hợp partiles thể hiện một phân bố xác xuất ngẫu nhiên.

Phân bố xác xuất trên cho thấy một số vùng rìa có xác xuất thấp, xuất hiện ít particles. Để tối ưu, các particles ở vùng quanh chóp sẽ được giữ lại và particles vùng rìa sẽ bị loại bỏ nếu có trọng số thấp hơn ngưỡng định trước (threshold). Nói cách khác, bộ lọc sẽ chỉ giữ lại các particles có khả năng xảy ra cao nhất để tiếp tục quá trình ước lượng state. Nhờ đó mà giảm được một khối lượng tính toán lớn không cần thiết.

Dựa vào ưu điểm lấy mẫu rời rạc, khả năng thích ứng với phân bố xác xuất bất kì và khối lượng tính toán hữu hạn, Particle filter trở thành lựa chọn tối ưu nhất để triển khai lên hệ thống số như robot. Thực tế hoá cho ý tưởng của Particle filter là giải thuật Monte Carlo filter dùng để ước lượng phân bố xác xuất của state. Tên gọi của giải pháp này là Monte Carlo Localization.

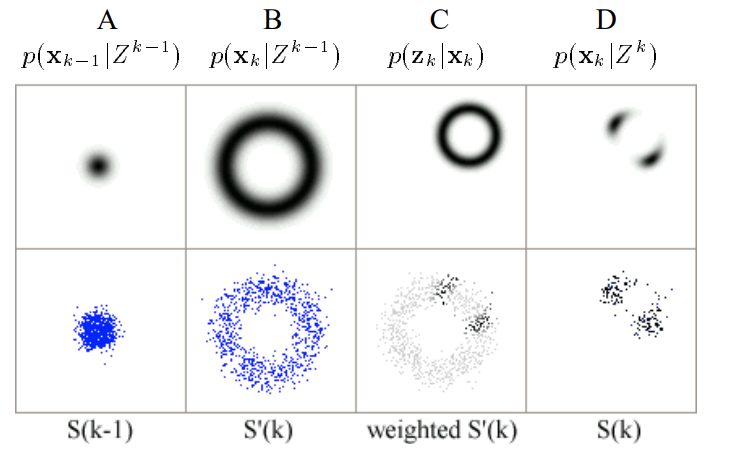
**Monte Carlo Localization**

Monte Carlo Localization [11] là giải thuật ước lượng state của mobile robot dựa trên Monte Carlo filter (Particle filter). Về tổng thể, Monte Carlo Localization dựa trên ý tưởng chung của nhóm bộ lọc Particle filter và Bayes filter. Như Bayes filter cho ước lượng trạng thái , Monte Carlo filter cũng theo 2 bước:

* + - 1. Dự đoán: Dựa vào hàm phân bố xác xuất , một nhóm particles là được lấy mẫu ngẫu nhiên. Mỗi particle kí hiệu là . Quá trình này được xem như áp dụng tín hiệu control vào các mẫu particle để đạt được các mẫu dự đoán . Gọi chung nhóm particle dự đoán là
      2. Cập nhật: Khi đã có dữ liệu measurement , các particles nhóm được cập nhật trọng số. Trong số này là xác xuất để xảy ra tại particle . Gọi trọng số của particle là , ta có:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Cuối cùng, chỉ các particle có được lấy để tiếp tục ước lượng state. Trải qua hữu hạn lần lặp hồi quy, các particle sẽ dần hội tụ tại state gần đúng với thực tế nhất. Nói cách khác, phương sai của hàm phân bố xác xuất sẽ đạt tối ưu.



Hình 4‑2 Minh hoạ phân bố xác xuất của vị trí robot qua các bước. Biến k đại diện cho lần lặp thứ k, thay cho t. [11]

***Lý thuyết các khái niệm trong Visual SLAM***

**Feature – đặc trưng**

Trong lĩnh vực thị giác máy tính, feature (đặc trưng) là thông tin đặc biệt xuất hiện trong ảnh, thường là những vùng ảnh cụ thể mang một tính chất đặc trưng. Các feature có thể là những cấu trúc hình học đặc biệt, khả kiến, xuất hiện trong ảnh như điểm, cạnh, đường cong hoặc một vật thể nhất định. Feature cũng có thể là những cấu trúc phức tạp mà mắt thường không nhận ra được. Các feature được xác định bằng các phép toán đại số áp dụng lên ma trận ảnh. Các phép này được biết tới là các phép trích đặc trưng ảnh – feature extraction. Các feature này do tính chất đặc trưng nên không thay đổi khi góc ảnh thay đổi hoặc ánh sáng thay đổi. Vì vậy chúng được dùng cho mục đích tính toán mối quan hệ chuyển động giữa nhiều ảnh chụp liên tiếp nhau. Nói cách khác, nhờ những đặc trưng này ta có thể biết được cách mà camera chuyển động.



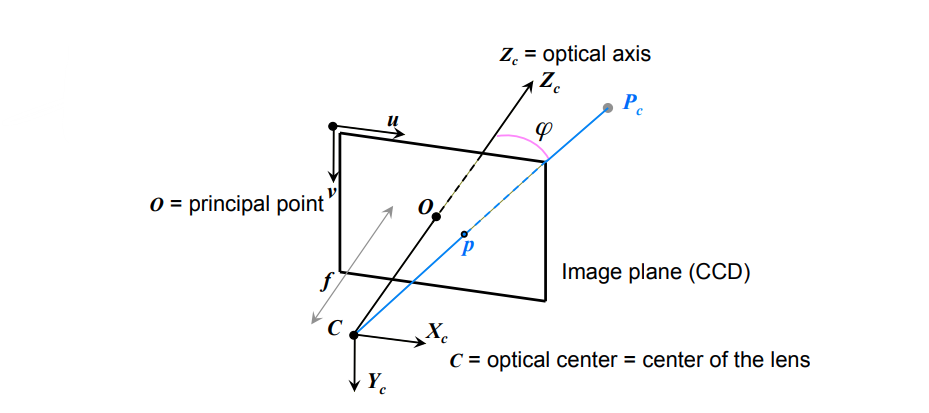
Hình 4‑3 Một số features giống nhau xuất hiện trong hai ảnh khác nhau.

Từ khả năng trên, các feature được dùng để tính toán chuyển động của camera thông qua chuỗi ảnh liên tiếp được chụp trong quá trình di chuyển. Đó vốn là vấn đề Visual Odometry, một phần của bài toán Visual SLAM.

**Triangulation**

Triangulation là kỹ thuật ứng dụng lý thuyết về camera để tìm được của độ 3D của các feature xuất hiện trong hai ảnh. Tạm gọi **P** là điểm thực tế tạo nên feature xuất hiện trên hai ảnh. Triangulation khôi phục lại tọa độ 3D **[X,Y,Z]** của một điểm **P** khi đã biết hình chiếu của **P** lên ảnh của nhiều camera, cũng có thể hiểu là nhiều ảnh của cùng một camera cùng chụp điểm **P**. Quy ước chung, mỗi ảnh sẽ xem như được chụp từ một camera.

Toạ độ 3D của **P** được khôi phục dựa trên vị trí tương đối của ít nhất 2 camera với nhau. Thường mối quan hệ vị trí tương đối này sẽ diễn theo dạng (R | T). Trong đó, chọn một camera làm gốc và hệ tọa độ của nó làm chuẩn. Hệ toạ độ của camera sẽ có trục x và y cùng nằm trên mặt phẳng ảnh và trục z sẽ vuông góc với 2 trục còn lại tại gốc toạ độ là *quang tâm* của camera.



Hình 4‑4 Minh hoạ toạ độ camera. [9]

Trong ảnh minh họa trên, **C** là quan tâm của camera cũng là gốc toạ độ của hệ. **O** là tâm của ảnh và đường thẳng **CO** là quang tâm của camera, cùng là **Zc** . **Pc**đại diện cho một điểm ảnh như **P** và **f** là tiêu cự của camera. Hai vector u và vector v đại diện cho hệ tọa độ 2D của ảnh có gốc toạ độ tại góc trên bên trái.

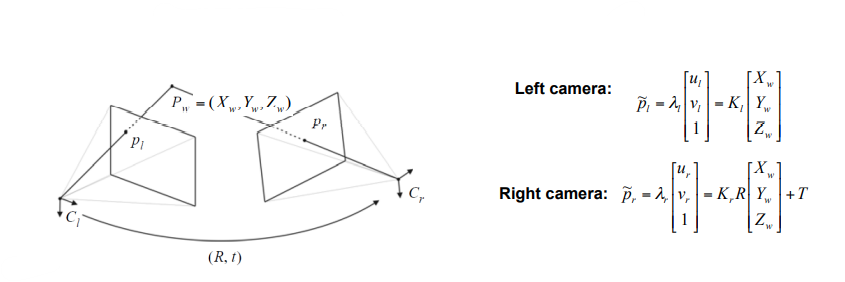
Xét trường hợp cụ thể, hai camera **Cl** (camera bên trái – left camera) và **Cr** (camera bên phải – right camera), ta có thể lập một hệ gồm hai phương trình mà trong đó toạ độ [X, Y, Z] của P là ẩn. Chọn gốc tọa độ gốc tham chiếu trùng với hệ toạ độ camera bên trái, minh hoạ và biểu diễn của hai phương trình như sau:

Camera bên trái – left camera:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

Camera bên phải – right camera:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |



*Hình 4‑5 Minh hoạ vị trí 2 camera và điểm P. [9]*

Trong hệ phương trình trên, Kl và Kr là hai ma trận kích thước (3x3), lần lượt mang thông số nội (intrinsic parameters) của camera trái và phải. Hai ma trận thông số này mang thông tin về tiêu cự và kích thước cảm biến, là những thông số đã biết. Còn lại, R và T là ma trận xoay (3x3) và ma trận chuyển vị (3x1). lần lượt là toạ độ hai điểm ảnh bên trái và bên phải (theo hệ toạ độ camera mỗi bên).

Như vậy, với chuỗi nhiều ảnh, toạ độ thực của các feature sẽ được ước lượng sơ bộ phục vụ cho quá trình tối ưu lại về sau để phục vụ cho quá trình Mapping. Đồng thời, mối quan hệ về R và T giữa mỗi ảnh sẽ có thể khôi phục lại một cách tương đối. Nhờ vậy mà quỹ đạo di chuyển gần đúng của camera sẽ góp một phần vào tiến trình Localization của giải thuật SLAM.

**Bundle Adjustment**

Bundle Adjustment là quá trình tối ưu hoá lại những ước lượng có được từ Triangulation hàng loạt ảnh trong chuỗi ảnh.

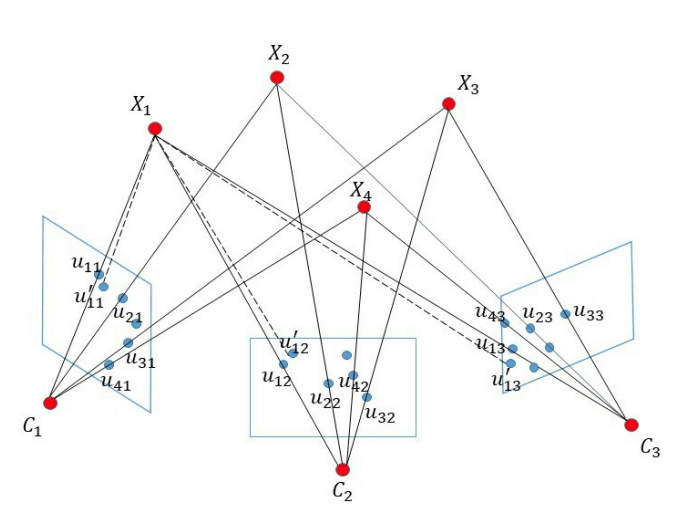
Bundle Adjustment về bản chất là quá trình tối thiểu hoá sai số giữa toạ độ feature thu được và toạ độ feature dựa trên toạ độ ước lượng (3D) của feature đó. Cụ thể, một điểm có toạ độ thực ước lượng là Xi, khi tính lại hình chiếu lên ảnh (bằng phương trình (14) và (15)) có toạ độ ảnh là . Quá trình tính lại hình chiếu gọi là reprojection. Do cả hai đều là toạ độ trên hệ toạ độ ảnh () nên sai số sẽ tính theo khoảng các Euclide, sai số này là reprojection error:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

Và min E:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

Trong đó là toạ độ pixel của feature thu được, và là hình chiếu của điểm Xi lên camera Cj. là toạ độ pixel thu được từ reprojection.



Hình 4‑6 Minh hoạ cho quá trình reprojection.[10]

Có thể thấy phương trình (4) khi áp dụng cho *n* features và *m* ảnh thì sẽ tạo thành phương trình phi tuyến phức tạp. Trong đó, để đạt được sai số cực tiểu, các biến thông số nội trong ma trận K được điều chỉnh trong quá trình reprojection. Đây chính là sự điều chỉnh (adjustment).

Mặc dù phương trình (4) lấy tổng E từ camera C1 tới Cn, nhưng thực tế feature Xi không thể thấy được từ tất cả camera. Nói cách khác, chỉ một nhóm ảnh (bundle) có hình chiếu của Xi sẽ đóng góp vào quá trình điều chỉnh.

Vì các phương pháp tìm cực tiểu cho phương trình phi tuyến lớn như (17) khá phức tạp và là lĩnh vực thuần toán nên đề tài sẽ không đi sâu vào chi tiết. Bài báo [10] là review một cách chi tiết về mặt toán cho tiến trình Bundle Adjustment này.

**Loop Closure Detection**

**TODO:**

**ORB-SLAM và RTAB-MAP**

**TODO: Phần này phải trình bày chi tiết zô**

## Nội dung 2

*Tìm hiểu về các thuật toán tìm đường đi tối ưu như Dijkstra, A\*, RRT\* cho global navigation và Dynamic Window Approach, D\* cho local navigation.*

**Global Navigation:** Dijkstra, A\*, RRT\***.**

**Local Navigation:** DWA và D\*.

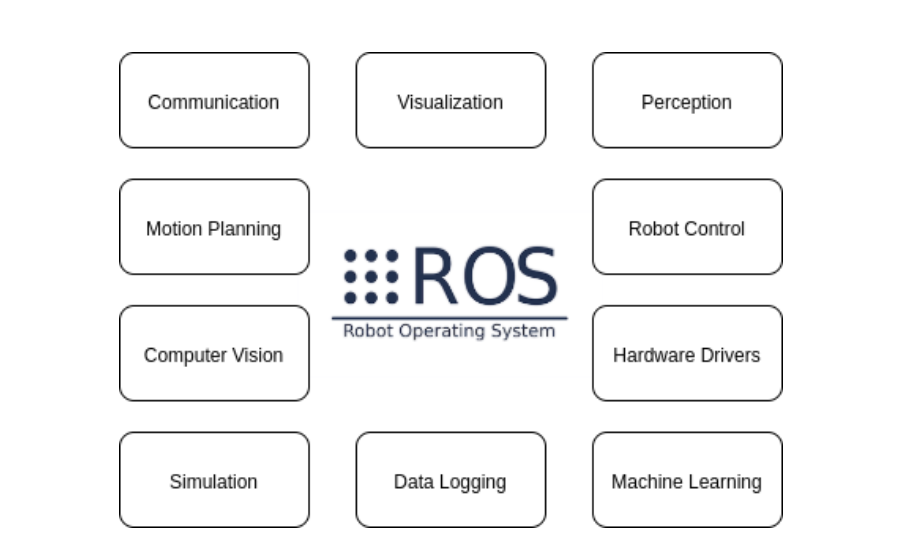
## Nội dung 3

*Tìm hiểu về cấu trúc của ROS và lý thuyết về các khối chức năng/cơ chế thông tin của ROS.*

Robot Operating System – ROS là một gói nhiều phần mềm bao gồm triển khai của các giải thuật mã nguồn mở, driver phần cứng và rất nhiều công cụ phát triển phần mềm cho robot. Mặc dù được gọi là “hệ điều hành” (Operating System) nhưng ROS không có cấu trúc như một hệ điều hành. ROS dựa trên 3 concept chính, theo 3 level:

1. *ROS Filesystem Level*: Thành phần này quản lý, phân cấp mã nguồn phần mềm và dữ liệu lưu trên bộ nhớ theo thứ tự, phương thức quy định bởi ROS. Nó tổ chức phần mềm/dữ liệu thành các Packages, Metapackages, Message types, Service types, …
2. *ROS Computation Graph Level*: Thành phần này quản lý việc trao đổi dữ liệu giữa các phần mềm và giữa các hệ thống trong ROS. Chúng được tổ chức thành các thành phần như Nodes, Master, Parameter Server, messages, services, topics, …
3. *ROS Community Level*: Thành phần này giúp quản lý sự phân tán trong mã nguồn mở. Nó giúp việc sử dụng, phát triển ROS trở nên thuận tiện hơn bằng cách quản lý việc phát triển các lớp (layers) phần mềm và tài liệu, đóng gói các phần mềm ổn định thành các “stack” trong ROS và phân phối tới các nhà phát triển thông qua GIT (một hệ thống quản lý source code). Mã nguồn, tài liệu phát triển của các stack, thư viện, framework được quản lý công khai qua GIT.

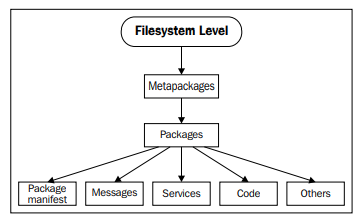
Nói chung, những phần mềm được tích hợp vào ROS vốn là những phần mềm dành cho các lĩnh vực hẹp hơn thị giác máy, điều khiển, giả lập và định hướng chuyển động. Trong đó, RVIZ là một phần mềm hạt nhận cực kỳ quan trọng của ROS. RVIZ giúp trực quan hoá số liệu, dữ liệu làm đơn giản hóa quá trình phát triển robot. ROS cũng tích hợp thư viện xử lý ảnh OpenCV hỗ trợ phát triển robot ứng dụng thị giác máy tính. Các bản phân phối ROS mới ngày càng tích hợp nhiều frameworks, thư viện mã nguồn mở trong lĩnh vực Machine Learning hỗ trợ việc triển khai trí tuệ nhân tạo cho robot.



*Hình 4-4 Những gói phần mềm được tích hợp trong ROS.*

## ROS Filesystem Level

Cấp quản lý Filesystem của ROS đặt ra quy chuẩn trong cách tổ chức hệ thống file/folder khi phát triển phần mềm.

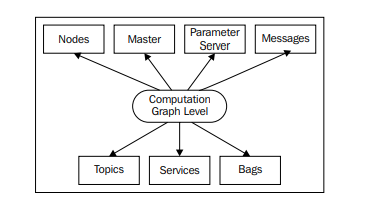


Hình 4‑7 Cấp quản lý Filesystem của ROS.

1. Packages: Packages là hạt nhân của ROS. Package có vai trò là thành phần tối thiểu cấu thành nên một chương trình trong ROS. Nó thường chứa các chương trình, file cấu hình, …
2. Package manifests: Package manifests chứa thông tin mô tả về package, về license, về các phần mềm phụ thuộc, về thông tin biên dịch, … Chúng được quản lý bới file package.xml
3. Metapackages: Là tập hợp của nhiều packages được đóng gói và biên dịch thành một Metapackage.
4. Metapackage manifests: Tương tự như package manifests về chức năng, cũng là file xml nhưng có một số ràng buộc nhất định về cấu trúc.
5. Message (msg) types: Message là gói thông tin trao đổi giữa các tiến trình (processes) đang hoạt động. ROS có nhiều loại message types, mỗi loại message type có cấu trúc riêng và được mô tả trong một file định dạng “.msg”. File mô tả cấu trúc cho các message types thường nằm trong folder “package/msg/”.
6. Service (src) types: Service types mô tả cấu trúc dữ liệu trong các gói thông tin dạng request-response được trao đổi giữa các tiến trình. Mô tả cho các Service types thường nằm trong folder “package/srv/” và có định dạng “.msg”.

## *ROS Computation Graph Level*

Đây là cấp quản lý các tiến trình chạy trong hệ thống. Cấp quản lý này tạo ra một mạng lưới kết nối chặt chẽ các thành phần để xử lý dữ liệu một cách khoa học.



Hình 4‑8 Các thành phần của cấp Computation Graph

Ở cấp này, các thành phần chức năng bao gồm:

1. Nodes: Nodes là tiến trình có thể chạy trên CPU. Các node liên kết với nhau thành một mạng lưới liên kết và có thể liên lạc bằng messages, sevices. Trong hệ thống, mỗi node sẽ đảm nhiệm một vai trò riêng. Các nodes được viết theo thư viện roscpp (C/C++) và rospy (Python)
2. Master: Master là chương trình quản lý toàn bộ các node. Nó thiết lập kết nối giữa các node. Chỉ một master cho hệ thống thì có thể quản lý nhiều nodes trên nhiều thiết bị, miễn là các thiết bị kết nối với nhau qua cơ chế TCP/IP.
3. Parameter server: Parameter server là chương trình quản lý việc lưu trữ dữ liệu. Nhờ server này, các thông số của node sẽ không cố định mà có thể thay đổi khi node đang chạy, giúp cho node hoạt động linh hoạt hơn.
4. Messages: Các nodes thông tin với nhau thông qua messages. Message là các gói dữ liệu có cấu trúc thống nhất theo message types giữa các nodes.
5. Topics: Topics là các luồng message được các nodes tạo ra và đăng kí tới Master. Master điều hướng messages theo cơ chế publish/subscribe. Nghĩa là, một node sẽ khởi tạo và gửi messages lên topic và có thể có một hoặc nhiều node khác subscribe tới topic đó. Khi có message mới được gửi (publish) lên thì tất cả các node subscribe đều nhận được message này. Mỗi topic chỉ trao đổi loại message theo message type của node tạo ra nó.
6. Services: Không giống như nodes, services là luồng thông tin được tạo bời node theo cơ chế request/response. Nói cách khác, khi một node khởi tạo sevice, nó sẽ nhận các yêu cầu (requests) từ một node A và trả về chính node A đó các gói thông tin hồi âm (responses). Các responses tuân theo cấu trúc được mô tả trong các file service types.
7. Bags: Bags là định dạng file ghi lại nội dung, thời gian và nhiều thông tin khác của message. Nhờ vậy các thông tin quan trọng như dữ liệu cảm biến có thể được lưu lại phục vụ quá trình tìm lỗi hoặc thử nghiệm.

## ROS *Community Level*

1. Distributions: Distribution là tập hợp của các thành phần hạt nhân của ROS kèm theo các gói metapackages lớn được biên dịch sẵn để người dùng cài đặt.
2. Repositories: Là các địa chỉ lưu trữ mã nguồn giúp chia sẻ, đóng góp mã nguồn phần mềm ROS. Mã nguồn của nhiều giải thuật triển khai trên ROS được tổ chức thành các Repositories này.
3. The ROS Wiki: ROS Wiki là trang web chứa tài liệu phát triển, sử dụng của các phần mềm ROS.
4. Bug Ticket System: Đây là hệ thống thông báo lỗi của ROS.
5. ROS Answer: ROS Answer là forum nơi để thảo luận về các vấn đề liên quan tới ROS.
6. ROS Blog: ROS Blog là trang web tin tức về ROS và lĩnh vực Robotics.

## RVIZ

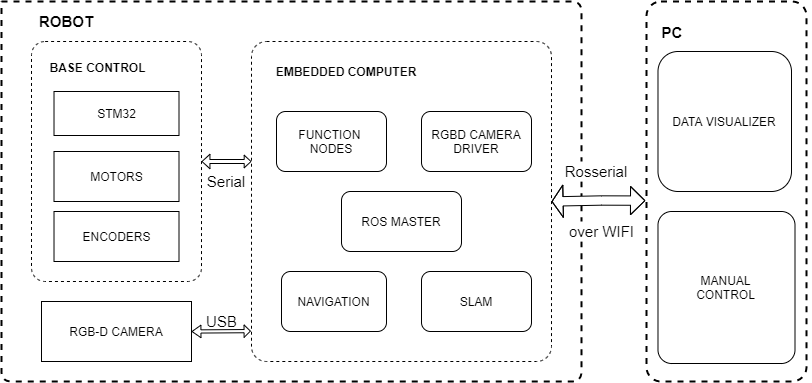
## TF Message

## Nội dung 5

*Tìm hiểu lý thuyết phần cứng và thiết kế mô hình*

## Tổng quan hệ thống

Để có hướng đi đúng trong quá trình phát triển mô hình, cấu trúc tổng quan của hệ thống cần được thiết kế trước hết. Về chức năng hệ thống cấu thành từ các khối liên kết như sau:



Hình 4‑9 Sơ đồ tổng quan của hệ thống.

Về mặt chức năng hệ thống bao gồm 3 khối lớn:

Khối Embedded Computer là máy chủ chạy ROS và các giải thuật quan trọng. Khối này có các khối nhỏ là:

* + 1. Khối Function Nodes: là đại diện cho các node làm nhiệm vụ truyền nhận thông tin với Base Control và PC thông qua giao thức Rosserial.
    2. Khối RGBD Camera Driver: là khối đại diện cho các node quản lý, xử lý thông tin được gửi về từ RGB-D Camera thông qua giao tiếp USB.
    3. Khối Navigation là khối chạy giải thuật Path planning dựa trên thông tin về bản đồ được cung cấp bởi khối SLAM.
    4. Khối SLAM là khối chạy giải thuật SLAM. Input là thông tin wheel odometry từ Base Control, dữ liệu từ RGB-D Camera. Output là thông tin về bản đồ, vật cản, vị trí robot được publish lên các topic.
    5. Khối ROS Master là trung tâm thông tin của toàn hệ thống. Các topics, services, nodes, ROS client được quản lý bởi ROS Master.

Khối Base Control điều khiển phần cứng. Input là thông tin về vận tốc mỗi bánh xe. Output là thông tin vận tốc thực thu được từ encoder.

1. Khối STM32 là khối phần cứng để triển khai PI Controller, tính vận tốc bánh xe từ encoder
2. Khối Motors cũng là khối phần cứng bao gồm 2 motor để di chuyển khung robot theo cơ chế lái vi sai.
3. Khối Encoders bao gồm 2 encoders cho mỗi bên motor. Dữ liệu từ encoder là dữ liệu hồi tiếp cho PI Controller, cũng là dữ liệu để tính vận tốc phục vụ wheel odometry.

Khối PC là khối giao tiếp với người dùng, hiển thị trực quan dữ liệu về bản đồ và vị trí robot, đường đi và vật cản. Khối này có input là thông tin về bản đồ, vị trí robot, đường đi,.. Output là lệnh điều khiển base control trực tiếp từ người dùng hoặc đích đến được đặt trước.

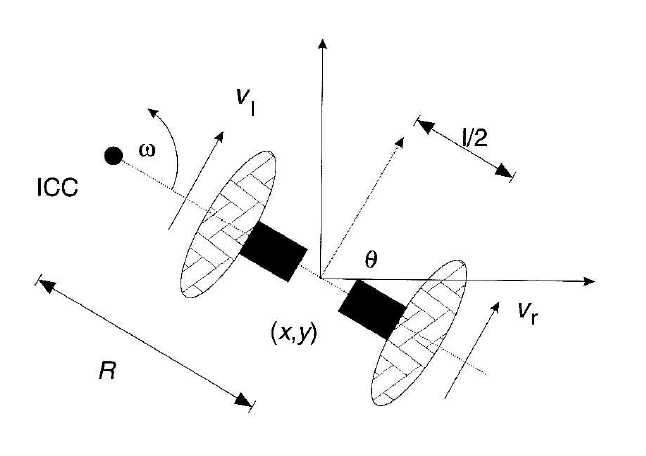
1. Khối Data Visualizer chứa các phần mềm trực quan hoá dữ liệu như RVIZ phục vụ hiển thị cho người dùng.
2. Khối Manual Control nhận thông tin điều khiển từ người dùng khi khối SLAM và Navigation chưa sẵn sàng.

4.1.4.2 Lý thuyết phần cứng

a) Differential drive – Lái vi sai

Nhiều robot sử dụng cơ chế lái vi sai – differential drive. Để ngắn gọn, đề tài sẽ đề cập đến cơ chế này bằng cả thuật ngữ tiếng anh lẫn tiếng việt.

Cơ chế lái vi sai thường được triển khai cho robot dùng hai motors có trục nằm trên cùng một đường thẳng. Mỗi motor được điều khiển tiến – lùi với vận tốc độc lập nhau. Do vậy, khi vận tốc mỗi bên bánh xe khác nhau dẫn đến robot di chuyển xoay quanh một điểm nằm trên đường thẳng qua trục. Điểm đó là ICC, viết tắt của thuật ngữ tiếng anh Instantaneous Center of Curvature.



Hình 4‑10 Differential Drive kinematics (from Dudek and Jenkin, Computational Principles of Mobile Robotics).

Bằng cách thay đổi tốc độ quay của motor, ta có thể thay đổi cách robot chuyển động. Vì cùng nằm trên đường thẳng qua ICC, tốc độ gốc quanh ICC của hai bánh xe như nhau, kí hiệu . Khi đó, vận tốc dài của hai bên được tính theo công thức:

|  |  |  |
| --- | --- | --- |
|  |  | () |

Trong đó:

* : Khoảng cách giữa hai bánh xe.
* , : là lần lượt vận tốc dài của bánh xe bên phải và bánh xe bên trái.
* R: là khoảng cách từ ICC đến trung điểm giữa hai bánh xe.

Từ () có thể suy ra mối quan hệ giữa vận tốc dài của robot, vận tốc góc và vận tốc mỗi bánh xe. Công thức biểu diễn như sau:

|  |  |  |
| --- | --- | --- |
|  |  | () |
|  |  | () |

Hai công thức trên đặc biệt quan trọng cho vấn đề điều khiển base control. Bời vì message điều khiển gửi từ ROS luôn ở dạng tốc độ dài (m/s) và tốc độ góc (rad/s). Từ đó, tốc độ dài mỗi bánh xe sẽ tính được sau đó quy đổi đơn vị sang vòng/phút để làm input cho PI controller.

b) Pulse Width Modulation – PWM.

Pulse Width Modulation là

c) Quadrature Encoder Interface

d) Quadrant Chopper Controller

e) PI Controller

4.1.4.2 Triển khai phần cứng

a) Khung

b) Mạch điều khiển Quadrant chopper

c) STM32F103 MCU

d) Nguồn

e) Motors

f) Máy tính nhúng

## Kết quả dự kiến đạt được

## Nội dung 4

*Phác hoạ thiết kế các khối phần mềm để chạy trên ROS.*

Ở nội dung này, nhiệm vụ đặt ra là thiết kế một số ROS node đảm nhiệm vai trò thông tin giữa khối điều khiển phần cứng (base control) và khối phần mềm (SLAM & Path Planning). Trước mắt, đề tài cần:

1. Node 1: Subscribe vào topic “/cmd\_vel” để nhận message điều khiển vận tốc hai bánh xe và gửi cho base control làm input cho PI controller.
2. Node 2: Nhận thông tin encoder về vận tốc mỗi bánh xe, tính toán vị trí robot theo dạng tf\_message để publish lên topic “/tf” phục vụ cho khối phần mềm SLAM.

Hiện tại hai node trên tạm thời đã được được triển khai lên ESP-8266 để thử nghiệm khối base control. Nhưng để hoàn thiện hệ thống, một số vấn đề có thể phát sinh khiến đề tài phải triển khai một số node khắc phục. Cụ thể, máy tính nhúng tài nguyên hạn chế, có khả năng không thể vừa chạy giải thuật vừa hiển thị dữ liệu. Vậy nên cần thiết phải dùng tới một máy tính khác để theo dõi dữ liệu. Một số node sẽ cần cho việc truyền nhận, nén và giải nén dữ liệu để phù hợp băng thông của kết nối giữa hai máy tính.

Do vậy, nội dung 4 vẫn chưa hoàn thiện.

## Nội dung 6

*Triển khai hệ thống phần mềm lên phần cứng thực tế đã thực hiện và đánh giá kết quả đạt được so với mục tiêu ban đầu đã đề ra.*

Nội dung 6 phụ thuộc vào tất cả các nội dung còn lại. Dự kiến khi hoàn thiện nội dung 6, một số điều kiện hoạt động sẽ được thử nghiệm để đánh giá độ chính xác của robot.

# KẾ HOẠCH THỰC HIỆN

Sinh viên trình bày tiến độ thực hiện luận văn, liệt kê các nội dung đã hoàn thành.

Với những nội dung chưa thực hiện xong trong quá trình làm đề cương luận văn, sinh viên lập kế hoạch để hoàn thành các nội dung còn lại. Các nội dung có thể chia thành nhiều bước nhỏ để lập kế hoạch.

Thời gian thực hiện luận văn: 4 tháng

Kế hoạch:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nội dung | Tháng 2 | Tháng 3 | Tháng 4 | Tháng 5 |
| 1. |  |  |  |  |
| 1.1 |  |  |  |  |
| 1.2 |  |  |  |  |
| 2. |  |  |  |  |
| 2.1 |  |  |  |  |
| 2.2 |  |  |  |  |

# TÀI LIỆU THAM KHẢO

1. Tống Văn On, “Thiết kế mạch số với VHDL & Verilog”, Nhà xuất bản Lao động Xã Hội, 2007.
2. Grisetti G, Stachniss C, Burgard W (2007) Improved techniques for grid mapping with rao-8 blackwellized particle filters. IEEE Transactions on Robotics 23(1), software: www.openslam.org/ gmapping.html
3. Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, Juan D. Tardos. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM”. 2021.
4. Han-ye Zhang, Wei-ming Lin and Ai-xia Chen. “Path Planning for the Mobile Robot: A Review”.
5. Sebastian THRUN, Wolfram BURGARD, Dieter FOX. “Probabilistic Robotics”. 1999-2000.
6. Rublee, Ethan; Rabaud, Vincent; Konolige, Kurt; Bradski, Gary (2011). "ORB: an efficient alternative to SIFT or SURF". IEEE International Conference on Computer Vision (ICCV).
7. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded Up Robust Features", ETH Zurich, Katholieke Universiteit Leuven
8. Labbé, RTAB-MAP.
9. Michael Bosse. “Perception II: Pinhole camera and Stereo Vision Autonomous Mobile Robots”.
10. Yu Chen, Yisong Chen, Guoping Wang. “Bundle Adjustment Revisited”. Peking University.
11. Frank Dellaert , Dieter Fox , Wolfram Burgard , Sebastian Thrun . “Monte Carlo Localization for Mobile Robots”. Carnegie Mellon University, University of Bonn.