Đánh Giá và Tối Ưu Thuật Toán Hector SLAM Ứng Dụng Lập Bản Đồ và Định Vị Trên Pimouse Robot

Đinh Bảo Minh, Đặng Anh Việt, Nguyễn Cảnh Thanh và Hoàng Văn Xiêm
Bộ môn Kỹ thuật Robot, Khoa Điện tử - Viễn Thông,
Trường Đại học Công Nghệ - Đại học Quốc Gia Hà Nội
Email: minhdinh@vnu.edu.vn, vietda@vnu.edu.vn, canhthanhlt@gmail.com, xiemhoang@vnu.edu.vn

Abstract—**Lập bản đồ và định vị là hai trong số các bài toán cơ bản của hệ thống Robot di động. Hai cách tiếp cận phổ biến hiện nay để giải quyết bài toán này là sử dụng hệ thống LiDAR hoặc/và hệ thống cảm biến hình ảnh cùng các thuật toán xử lý dữ liệu thu được. Hướng tiếp cận với LiDAR và thuật toán Hector SLAM cho kết quả tạo bản đồ với độ chính xác cao, nhưng đòi hỏi phải tối ưu các tham số của thuật toán. Để hiểu rõ vấn đề này, chúng tôi nghiên cứu và đánh giá các tham số chính ảnh hưởng tới hiệu năng thực thi của thuật toán Hector SLAM cho một hệ thống Robot di động sử dụng LiDAR. Hiệu năng của hệ thống được đánh giá trên hai khía cạnh: i) chất lượng của bản đồ thu được và ii) lượng CPU chiếm dụng. Với việc hiểu rõ ảnh hưởng của các tham số của thuật toán Hector SLAM tới hiệu năng của hệ thống, người dùng có thể thay đổi linh hoạt các tham số này tùy vào Robot sử dụng. Kết quả nghiên cứu được minh họa trên một hệ thống Robot di động được phát triển bởi công ty RT Corporation, Nhật Bản, Pimouse Robot.**

Keywords- Hector SLAM, ROS, Pimouse Robot.

I. GIỚI THIỆU

Robot di động là một lĩnh vực nghiên cứu của người máy và kỹ thuật thông tin. Robot di động có thể được điều khiển bởi con người hoặc tự động hoàn toàn (AMR - Autonomous Mobile Robot) với khả năng tự điều hướng trong môi trường mà không cần đến các thiết bị điều khiển. Ngày nay, Robot di động là một trong những hướng nghiên cứu đang rất được quan tâm và được ứng dụng rộng rãi trong nhiều lĩnh vực như công nghiệp, thương mại, quân sự và an ninh.

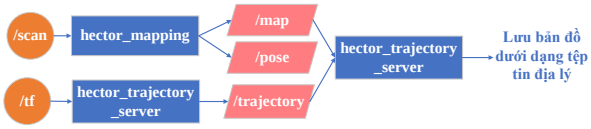
Đối với AMR, robot cần xây dựng bản đồ của môi trường làm việc và hiểu được nó. Đồng thời robot phải xác định được vị trí của mình cũng như các chướng ngại vật xuất hiện trong môi trường làm việc nêu trên. Lập bản đồ (Mapping) là quá trình AMR mô hình hóa môi trường làm việc của mình. Dựa vào bản đồ được tạo ra, các AMR có thể điều hướng tự động, từ đó ứng dụng trong các lĩnh vực như tìm kiếm cứu hộ, vận chuyển thông minh... Một phương pháp cho phép các AMR có thể thực hiện đồng thời hai tác vụ lập bản đồ cũng như định vị robot trong bản đồ cùng một lúc được biết với tên gọi SLAM (Simultaneous Localization and Mapping).

Nguyên tắc cơ bản của SLAM là cung cấp thông tin về môi trường xung quanh dựa trên hệ thống cảm biến của nó và xây dựng bản đồ của không gian làm việc trong khi ước tính vị trí và định hướng của robot [1]. Ngày nay, có rất nhiều thuật toán SLAM đã được phát triển như: GMapping [2], Karto [3], CartoGrapher [4], Hector SLAM [5], PTAM [6, 7], REMODE [8], ORB-SLAM [9, 10], DTAM [11], LSD-SLAM [12], Stereo LSD-SLAM [13], SVO [14], RTAB map [15], CNN-SLAM [16], DPPTAM [17], DSO [18], S-PTAM [19].

Hệ điều hành Robot (ROS) là một framework phổ biến nhất trong công nghệ robot ngày nay. Nó cung cấp một bộ công cụ, thư viện và trình điều khiển để giúp phát triển các ứng dụng robot với sự trừu tượng hóa phần cứng [20]. Với sự trợ giúp của ROS, các phương pháp SLAM nêu trên có thể dễ dàng được thực hiện, nghiên cứu và phát triển.

Các nghiên cứu [21-25] đã thực nghiệm và so sánh chất lượng của nhiều thuật toán SLAM khác nhau trong một điều kiện kiểm thử cụ thể. Ngoài ra, các nghiên cứu [26-28] còn thực hiện kết hợp cả việc sử dụng LiDAR và Camera hoặc các cảm biến khác như IMU cho việc nâng cao chất lượng quá trình bản địa hóa, tạo bản đồ hoặc định vị. Để dễ dàng thực hiện các nghiên cứu [26-28], việc hiểu rõ ưu, nhược điểm của từng thuật toán SLAM là vô cùng cần thiết. Bên cạnh đó, các nghiên cứu [21-25] chỉ thực hiện các đánh giá để so sánh độ chính xác trong việc xây dựng bản đồ của thuật toán hay mức độ sử dụng CPU của phần cứng mà không chỉ rõ về việc cấu hình bộ tham số cho từng thuật toán, yếu tố ảnh hưởng tới chất lượng của thuật toán trong các môi trường khác nhau. Do đó, việc nắm được sự ảnh hưởng của các tham số khác nhau tới thuật toán là cần thiết. Bài báo nghiên cứu và đánh giá hiệu năng của thuật toán HectorSLAM trên hệ thống Robot di động Pimouse Robot [29] khi thay đổi các tham số chính. Hai khía cạnh được đánh giá là chất lượng bản đồ thu được và lượng CPU chiếm dụng.

Phần còn lại của bài báo được tổ chức như sau: Phần II thảo luận về các công trình liên quan; Phần III giới



Hình 1: Các node của Hector SLAM

thiệu môi trường và thuật toán kiểm thử. Phần IV trình bày các kết quả và thảo luận của nghiên cứu này. Phần V là phần kết luận của bài báo.

II. CÁC NGHIÊN CỨU LIÊN QUAN

Nghiên cứu của Sankalprajan và các cộng sự [22] đã phân tích so sánh chất lượng và độ chính xác của các thuật toán 2D SLAM dựa trên ROS trong môi trường mô phỏng bãi đậu xe ngầm có các hệ số tỷ lệ giống như thực tế. Nghiên cứu kết luận rằng thuật toán Hector SLAM đưa ra các bản đồ cả trong thực tế cũng như mô phỏng rất giống nhau. Đồng thời chỉ ra rằng Hector SLAM cũng phụ thuộc vào các yếu tố như ngưỡng góc, ngưỡng tuyến tính và các yếu tố cập nhật.

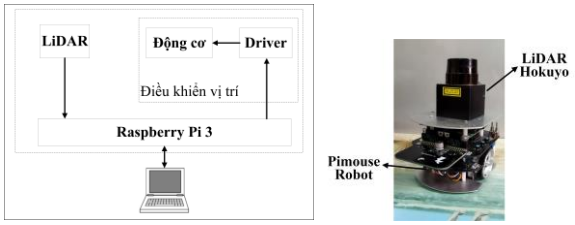
Nghiên cứu [23] đã so sánh quỹ đạo di chuyển của UGV. Các phương pháp SLAM được lựa chọn để so sánh sử dụng nhiều loại cảm biến khác nhau: Monocular camera, ZED camera, Kinect RGB-D camera và LiDAR. Hector SLAM được sử dụng như một ground truth nhằm đánh giá các thuật toán visual SLAM.

Một nghiên cứu khác [25] cũng so sánh về quỹ đạo của robot di động được tính toán bởi các hệ thống SLAM dựa trên ROS. Kết quả đánh giá cho thấy Hector SLAM và Cartographer mang lại hiệu quả rất tốt với độ chính xác với sai số RMSE là 0,024 m.

Thrilochan Sharma và các cộng sự [30] đã mô tả hiệu suất của bốn thuật toán GMapping, Hector SLAM, KartoSLAM và RTAB Map trong cả mô phỏng và thực tế. Kết quả trung bình cho thấy Hector SLAM cho hiệu suất tốt, tiêu tốn ít tài nguyên hơn.

Nghiên cứu [31] trình bày một phương pháp tối ưu thuật toán Hector SLAM thông qua việc tinh chỉnh các tham số của thuật toán. Tuy nhiên nghiên cứu trực tiếp đưa ra bộ tham số được cho là tối ưu chứ không thực hiện so sánh kết quả thu được từ bộ tham số đó với kết quả thu được từ các bộ tham số khác có thể có.

Qua kết quả từ các nghiên cứu [22, 23, 25, 30], Hector SLAM cho thấy khả năng xây dựng bản đồ với độ chính xác cao, trong khi mức CPU chiếm dụng là không đáng kể, điều này phù hợp cho việc tích hợp thuật toán trên các nền tảng Robot di động sử dụng máy tính nhúng Raspberry Pi 3 như Pimouse Robot. Tuy nhiên, các nghiên cứu này chỉ dừng ở việc so sánh Hector SLAM với các thuật toán SLAM khác mà chưa nêu rõ các thiết đặt cụ thể cho từng tham số sử dụng cho thuật toán SLAM. Mặt khác, nghiên cứu [31] chỉ ra rằng các tham số trong mỗi thuật toán SLAM đều có thể ảnh hưởng tới chất lượng ánh xạ cũng như độ chính xác của bản đồ thu được. Do đó nghiên cứu của chúng tôi sẽ tập trung phân tích mức độ ảnh hưởng của các tham số khác



Hình 2: Sơ đồ kết nối tổng quan hệ thống (bên trái), Mô hình PiMouse robot (bên phải)

nhau tới kỹ thuật Hector SLAM được tích hợp trong ROS thông qua thực nghiệm trên hệ thống Pimouse Robot. Kết quả thực nghiệm chỉ ra rằng việc thay đổi giá trị các tham số có thể nâng cao chất lượng bản đồ thu được, giảm thiểu sai số định vị robot đồng thời ảnh hưởng tới mức độ sử dụng của CPU.

III. THÔNG TIN HỆ THỐNG KIỂM THỬ

A. *Hector SLAM trong ROS*

Hector SLAM là kỹ thuật SLAM 2D được phát triển vào năm 2018 [5]. Trong ROS, hector_slam [32] là một gói nhỏ nhằm cài đặt hector_mapping và các gói liên quan. Các gói chính bao gồm [32]:

- hector_mapping: Node SLAM dựa trên LiDAR không cần odometry và tài nguyên tính toán thấp
- hector_geotiff: Lưu bản đồ và quỹ đạo robot vào các tệp hình ảnh địa lý.
- hector_trajectory_server: Lưu quỹ đạo dựa trên tf

Hình 1 biểu thị mối liên hệ giữa các node của Hector SLAM được tích hợp trong ROS. Trong đó, các tham số của node hector_mapping có thể phân loại như sau:

- Tham số tf: gồm các tham số để điều chỉnh khung tf;
- Tham số bản đồ: gồm các tham số để thiết lập các thuộc tính bản đồ như kích thước, vị trí xuất xứ, thời gian xuất bản bản đồ...;
- Thông số Laser: gồm các thông số để thiết lập các ngưỡng của máy quét laser. Các giá trị mặc định khớp với các thông số của cảm biến LiDAR Hokuyo [33].

Trong nghiên cứu này, chúng tôi thực hiện phân tích 04 tham số thuộc phân loại tham số bản đồ được mô tả trong Bảng I, là *DTh*, *ATh*, *FF* và *FO*. Các tham số không có trong bảng sẽ được giữ nguyên giá trị mặc định như mô tả tại [32].

B. *Phần cứng*

Quá trình kiểm thử, nghiên cứu sử dụng một nền tảng Robot di động đã được thương mại hóa là Pimouse Robot. Nền tảng Pimouse Robot sử dụng bộ vi xử lý máy tính nhúng Raspberry Pi 3 với hệ điều hành Ubuntu 18.04 LTS và ROS Melodic (Bảng II mô tả chi tiết về cấu hình Raspberry Pi 3 được sử dụng). Nền tảng bao gồm phần thân robot và cảm biến LiDAR Hokuyo được gắn trên đỉnh của robot PiMouse như trong Hình 2.

Pimouse robot được điều khiển bởi một máy tính chủ thông qua kết nối wifi giữa máy tính chủ và máy

tính nhúng Raspberry Pi 3. Hình 2 mô tả sơ đồ kết nối tổng quan của hệ thống.

IV. KIỂM THỬ VÀ ĐÁNH GIÁ KẾT QUẢ

A. *Điều kiện kiểm thử*

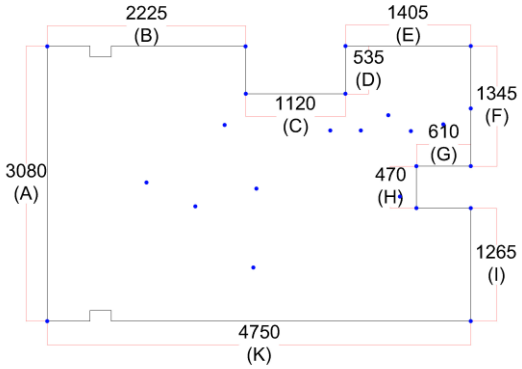
Thử nghiệm được tiến hành trong môi trường trong nhà kín với kích thước được mô tả theo bản vẽ kĩ thuật Hình 3, vật cản ở đây chính là các bức tường xung quanh phòng. Để đảm bảo các yếu tố bên ngoài như thời gian hoạt động, quãng đường di chuyển,… không ảnh hưởng tới kết quả kiểm thử, với mỗi lần thử nghiệm, robot sẽ được khởi động lại và di chuyển theo cùng một quỹ đạo được vạch ra trước với tốc độ di chuyển cố định. Bên cạnh đó tất cả các tham số khác đều được cố định theo giá trị mặc định, ngoại trừ tham số mà ảnh hưởng của nó đang được khảo sát. Bốn tham số *DTh*, *ATh*, *FF* và *FO* sẽ được tiến hành kiểm thử theo từng cặp nhằm phù hợp với mô tả về chức năng của chúng trong bảng I.

BẢNG I. CÁC THAM SỐ ĐƯỢC KHẢO SÁT

Tham số	Chức năng	Giá trị mặc định
map_update_ distance_thresh (*DTh*)	Ngưỡng để thực hiện cập nhật bản đồ (m, rad). Robot phải di chuyển xa đến mức này hoặc trải qua một góc theo thông số kể từ lần cập nhật cuối cùng.	0.4(m)
map_update_ angle_thresh (*ATh*)		0.9(rad)
update_ factor_free (*FF*)	Công cụ sửa đổi cập nhật bản đồ để cập nhật các ô trống (giá trị 0.0)/bị chiếm đóng (giá trị 1.0) trong phạm vi (0.0, 1.0)	0.4
update_ factor_occupied (*FO*)		0.9

BẢNG II. CẤU HÌNH RASPBERRY PI 3

Parameter	Configuration
Processor	Broadcom BCM2837 Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
RAM	1 GB LPDDR2 SDRAM
OS	Ubuntu 18.04
ROS	ROS Melodic
Storage	Micro SD 16Gb

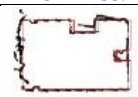
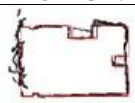
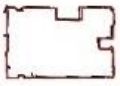
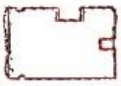
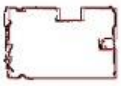
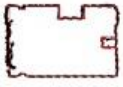


Hình 3: Bản vẽ kĩ thuật môi trường kiểm thử và các điểm đo sai số RMSE trên bản đồ

Trong thực nghiệm đánh sự ảnh hưởng của cặp tham số *DTh* và *ATh*, 25 cặp giá trị tổ hợp từ các giá trị 0.05, 0.1, 0.2, 0.4 và 0.9 sẽ lần lượt được cấu hình cho hệ thống. Robot sẽ được di chuyển đều với vận tốc 0.25 (m/s) trong thử nghiệm.

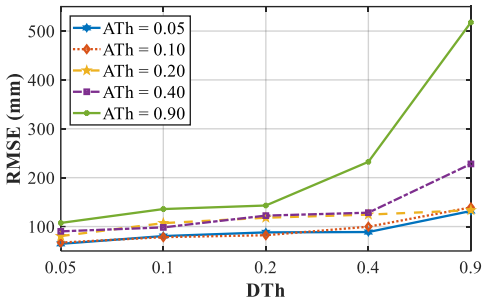
Một số lưu ý trong việc đánh giá sự ảnh hưởng của cặp tham số *FF*-*FO*: Tham số *FF* cần giữ một giá trị dưới 0.5 và tham số *FO* cần giữ giá trị trên 0.5. Nếu hai tham số này đều bằng 0.5, bản đồ thu được sẽ không có

BẢNG III. BẢN ĐỒ THU ĐƯỢC VÀ SAI SỐ RMSE TƯƠNG ỨNG VỚI BỘ THAM SỐ DTH VÀ ATH

DTh \ *ATh*	**0.05**	**0.1**	**0.2**	**0.4**	**0.9**
0.9	 RMSE = 131.80	 RMSE = 139.20	 RMSE = 133.46	 RMSE = 228.11	 RMSE = 517.74
0.4	 RMSE = 88.76	 RMSE = 99.66	 RMSE = 124.40	 RMSE = 128.36	 RMSE = 232.42
0.2	 RMSE = 88.05	 RMSE = 82.24	 RMSE = 118.06	 RMSE = 122.35	 RMSE = 143.04
0.1	 RMSE = 80.51	 RMSE = 78.28	 RMSE = 107.02	 RMSE = 98.26	 RMSE = 135.79
0.05	 RMSE = 64.63	 RMSE = 67.19	 RMSE = 80.51	 RMSE = 90.16	 RMSE = 107.38

[a.] Trong hình vẽ bản đồ, đường màu đỏ thể hiện bản đồ gốc (tham chiếu), đường màu đen thể hiện bản đồ thu được từ thuật toán

[b.] Đơn vị của sai số RMSE là (mm)



Hình 4: Sai số RMSE khi thay đổi DTh và ATh

sự thay đổi. Tuy nhiên, nếu cặp giá trị *FF-FO* cách xa nhau quá nhiều sẽ gây ra chênh lệch lớn trong việc cập nhật bản đồ dẫn tới bản đồ không chính xác.

Với lưu ý nêu trên, sự ảnh hưởng của cặp tham số *FF-FO* sẽ được đánh giá qua thực nghiệm với 04 cặp giá trị là 0.1-0.6, 0.2-0.7, 0.3-0.8 và 0.4-0.9. Ngoài ra, 03 vận tốc khác nhau là 0.30, 0.25 và 0.20 (m/s) sẽ được sử dụng cho khảo sát này.

Thước đo Root Mean Square Error (RMSE) được chúng tôi sử dụng nhằm đánh giá sai lệch thử nghiệm, so sánh dữ liệu vị trí thu thập được từ thuật toán Hector SLAM với dữ liệu vị trí thực của các điểm marker được lựa chọn từ trước, như mô tả trong Hình 3. RMSE (mm) được xác định theo phương trình sau:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} \frac{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}{N}} \quad (1)$$

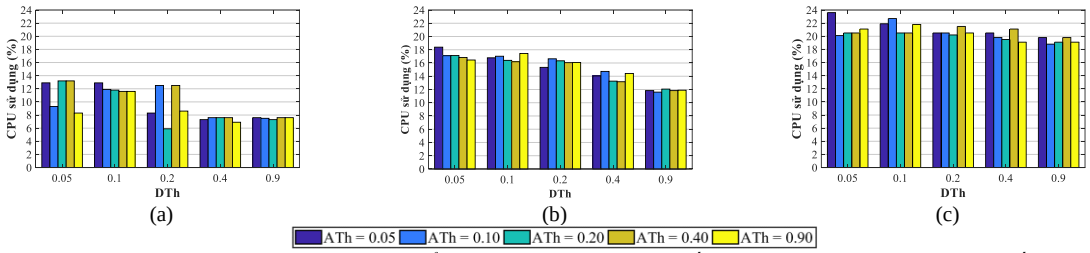
Trong đó:

- x_i và y_i là tọa độ điểm thứ i trong sơ đồ kỹ thuật;
- $\hat{x}_i$ và $\hat{y}_i$ là tọa độ điểm thứ i trong bản đồ thu được bởi thuật toán;
- N là số lượng các điểm được đánh giá.

B. *Kết quả kiểm thử*

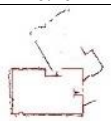
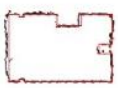
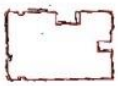
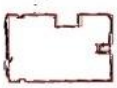
Kết quả khảo sát chất lượng bản đồ cũng như sai số RMSE tương ứng khi thay đổi giá trị cặp tham số *DTh* và *ATh* được mô tả trong Bảng III. Hình 4 cho một cái nhìn rõ hơn về sự thay đổi của sai số RMSE.

Qua Bảng III, có thể thấy với môi trường rộng và ít vật thể như môi trường kiểm thử mô tả trong bài báo, việc cập nhật bản đồ với tham số ngưỡng khoảng cách (*DTh*) quá lớn sẽ làm bản đồ xuất hiện nhiều sai lệch so



Hình 5: Mức độ sử dụng CPU khi thay đổi DTh và ATh (a) mức nhỏ nhất, (b) mức trung bình, (c) mức lớn nhất

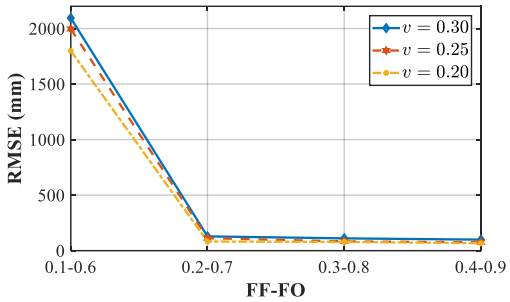
BẢNG IV. BẢN ĐỒ THU ĐƯỢC VÀ SAI SỐ RMSE TƯƠNG ỨNG VỚI BỘ THAM SỐ FF VÀ FO

FF	**FO**	***v* (m/s)**		
		0.30	**0.25**	**0.20**
0.1	**0.6**	 RMSE = 2094.14	 RMSE = 1995.95	 RMSE = 1799.07
0.2	**0.7**	 RMSE = 128.46	 RMSE = 112.48	 RMSE = 82.20
0.3	**0.8**	 RMSE = 110.58	 RMSE = 83.37	 RMSE = 78.66
0.4	**0.9**	 RMSE = 99.20	 RMSE = 75.51	 RMSE = 67.02

[a.] Trong hình vẽ bản đồ, đường màu đỏ thể hiện bản đồ gốc (tham chiếu), đường màu đen thể hiện bản đồ thu được từ thuật toán

[b.] Đơn vị của sai số RMSE là (mm)

với bản đồ kỹ thuật, điều tương tự cũng xảy ra khi



Hình 6: Sai số RMSE khi thay đổi FF và FO

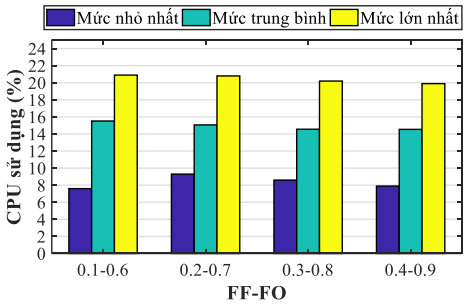
ngưỡng của góc quay (*ATh*) quá lớn. Kết quả mô tả tại Hình 4 cũng cho thấy, với giá trị *DTh* và *ATh* càng nhỏ, bản đồ thu được sẽ cho sai số RMSE càng nhỏ. Ngoại trừ trường hợp khi giá trị *DTh* = 0.1 sẽ có sự nhấp nhô khi *ATh* = 0.1 hay *ATh* = 0.4, về tổng thể nhận định trên vẫn đúng với kết quả tạo bởi 23 cặp tham số còn lại.

Lý do ở đây là do bản đồ thu được sẽ bị mất mát nhiều khoảng khi *DTh* lớn, dẫn tới sai số cao. Với *ATh* lớn, tầm nhìn LiDAR sẽ bị thay đổi nhanh dẫn tới bản đồ sẽ bị sai lệch lớn.

Hình 5 mô tả về sự ảnh hưởng của cặp tham số *DTh* và *ATh* tới lượng CPU chiếm dụng. Về tổng quan, xét với biểu đồ phần trăm sử dụng CPU trung bình, các giá trị *DTh* và *ATh* cao sẽ khiến bản bản đồ cập nhật ít liên tục hơn so với các ngưỡng cao thấp đó cần ít tài nguyên về CPU hơn. Tuy nhiên, mức độ sử dụng CPU cũng phụ thuộc vào dữ liệu mà LiDAR quan sát được nên có thể thấy sự nhấp nhô của 03 biểu đồ thu được. Sự nhấp nhô bị ảnh hưởng bởi *ATh* nhiều hơn so với *DTh*, bởi lượng dữ liệu LiDAR quan sát được khi robot quay một góc sẽ thay đổi nhiều hơn khi so với lượng thông tin thay đổi khi robot di chuyển thẳng theo một phương.

Như vậy về tổng thể sự thay đổi giá trị *DTh, ATh* có ảnh hưởng tới tiêu thụ tài nguyên bộ nhớ của phần cứng theo chiều tỉ lệ nghịch với chất lượng bản đồ thu được cũng như sai số RMSE. Với kết quả này, cặp tham số *DTh* và *ATh* nên được đặt ở khoảng từ 0.1 tới 0.2 để vừa có thể đảm bảo chất lượng bản đồ vừa đảm bảo mức chiếm dụng CPU đặc biệt với phần cứng hiệu năng thấp.

Kết quả khảo sát chất lượng bản đồ và sai số RMSE tương ứng khi thay đổi cặp tham số *FF-FO* được mô tả



Hình 7: Mức độ sử dụng CPU khi thay đổi FF và FO

trong Bảng IV. Hình 6 cho một cái nhìn rõ hơn về sự thay đổi của sai số RMSE.

Kết quả thu được từ Bảng IV cũng như Hình 6 cho thấy tốc độ di chuyển chậm hơn sẽ cho bản đồ xây dựng được chính xác hơn. Mặt khác có thể thấy, khi cặp tham số mức cập nhật ô trống-ô chiếm đóng (*FF-FO*) có giá trị nhỏ (0.1-0.6), sai số gây ra bởi tốc độ di chuyển sẽ có sự sai khác rất lớn. Tuy nhiên với mức cập nhật là 0.4-0.9 sự sai khác đó đã giảm đi khá nhiều. Độ chính xác của bản đồ cũng tăng dần khi tăng dần giá trị cặp tham số *FF-FO*.

Đối với thực nghiệm khảo sát mức chiếm dụng CPU của hệ thống, nghiên cứu sẽ chỉ tiến hành kiểm thử với trường hợp tốc độ di chuyển $v = 0.25$ (m/s). Hình 7 biểu diễn phần trăm sử dụng CPU của máy tính nhúng Raspberry Pi 3 khi thay đổi cặp tham số *FF* và *FO*. Tương tự như trường hợp thay đổi cặp tham số *DTh* và *ATh*, mức sử dụng CPU nhỏ nhất có sự nhấp nhô nhỏ. Đối với mức độ sử dụng CPU trung bình và lớn nhất, cả hai loại này đều có xu hướng giảm dần khi tăng dần giá trị của cặp tham số *FF-FO*. Tuy nhiên, mức độ sử dụng CPU trung bình giảm chậm hơn so với khi thay đổi cặp tham số *DTh* và *ATh*.

Khác với khảo sát tại cặp tham số *DTh* và *ATh*, kết quả khảo sát cặp tham số *FF* và *FO* cho thấy sự tỉ lệ thuận trong sự thay đổi về chất lượng bản đồ cũng như sai số RMSE khi so với mức CPU sử dụng. Cặp tham số *FF-FO* nhận giá trị càng cao sẽ cho bản đồ thu được có chất lượng càng tốt, sai số RMSE càng nhỏ và mức độ sử dụng CPU càng nhỏ. Tuy nhiên, sự khác biệt mà tốc độ di chuyển tác động tới cặp tham số này rất đáng lưu tâm, tốc độ di chuyển cao sẽ giúp tiết kiệm thời gian của quá trình xây dựng bản đồ, nhưng đổi lại sai số sẽ lớn nếu cặp giá trị *FF-FO* nhỏ. Để giảm bớt ảnh hưởng của tốc độ di chuyển của robot, giá trị *FF* nên đặt trong khoảng từ 0.3 tới 0.4 và giá trị *FO* nên đặt trong khoảng từ 0.8 tới 0.9.

V. KẾT LUẬN

Trong bài báo này, chúng tôi khảo sát ảnh hưởng của các tham số trong thuật toán Hector SLAM trên Pimouse Robot trong môi trường thực. Cụ thể, chúng tôi đưa ra hai thử nghiệm về đánh giá sự ảnh hưởng của tham số tới chất lượng bản đồ, sai số định vị và mức độ sử dụng CPU. Kết quả cho thấy sự tỉ lệ nghịch giữa sự biến đổi về chất lượng bản đồ, sai số định vị với mức độ sử dụng CPU khi khảo sát cặp tham số *DTh* và *ATh*, từ đó đưa ra gợi ý về việc lựa chọn cặp tham số phù hợp nhằm cân nhắc sự đánh đổi giữa chất lượng bản đồ và mức độ tiêu thụ bộ nhớ phần cứng của robot. Đối với khảo sát về cặp tham số FF và FO, nghiên cứu chỉ ra ảnh hưởng của tốc độ di chuyển tới chất lượng bản đồ thu được và độ chính xác về vị trí robot. Từ đó đưa ra lời khuyên về việc lựa chọn cặp tham số FF-FO phù hợp giúp giảm bớt sự ảnh hưởng của vận tốc tới quá trình xây dựng bản đồ và định vị của robot.

Trong tương lai, chúng tôi sẽ nghiên cứu các phương pháp định tuyến trong di chuyển như tối ưu hóa tuyến đường cho một robot độc lập, kết hợp nhiều robot di chuyển (robot bầy đàn) để giải quyết các vấn đề về SLAM.

LỜI CẢM ƠN

Nghiên cứu này được tài trợ bởi Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội theo đề tài mã số CN20.42.

TÀI LIỆU THAM KHẢO

[1] T. T. Mac, C. Y. Lin, N. G. Huan, L. D. Nhat, P. C. Hoang, and H. H. Hai, "Hybrid SLAM-based Exploration of a Mobile Robot for 3D Scenario Reconstruction and Autonomous Navigation," Acta Polytechnica Hungarica, vol. 18, no. 5, 2021.

[2] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," IEEE Trans. Robot., vol. 23, no. 1, pp. 34-46, 2007.

[3] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in 2010 IEEE/RSJ Int. Conf. Intell. Robot. Syst., pp. 22-29, Oct. 2010.

[4] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in 2016 IEEE Int. Conf. Robot. Autom. (ICRA), pp. 1271-1278, May. 2016.

[5] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation in 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, pp. 155–160, Nov. 2011.

[6] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in 2007 6th IEEE. ACM int. sympo. Mixed. augm. Real., pp. 225–234, Nov. 2007.

[7] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," IEEE Robot. Autom. Mag., vol. 13, no. 3, pp. 108–117, 2006.

[8] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in 2014 IEEE Int. Conf. Robot. Autom. (ICRA), pp. 2609–2616, May. 2014.

[9] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, 2015.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in 2011 IEEE Int. Conf. Comput. Vis. (ICCV), pp. 2564–2571, Nov. 2011.

[11] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM : Dense Tracking and Mapping in Real-Time," in 2011 IEEE Int. Conf. Comput. Vis., pp. 2320–2327, Nov. 2011.

[12] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct monocular SLAM," in Eur. Conf. Comput. Vis., vol. 8690 LNCS, no. PART 2, pp. 834–849, Sept. 2014.

[13] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in IEEE Int. Conf. Intell. Robot. Syst., vol. 2015-Decem, pp. 1935–1942, Sept. 2015.

[14] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO : Fast Semi-Direct Monocular Visual Odometry," in 2014 IEEE Int. Conf Robot. Autom. (ICRA), pp. 15–22, May. 2014.

[15] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," in 2014 IEEE/RSJ Int. Conf. Intell. Robot. Syst., pp. 2661–2666, Sept. 2014.

[16] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in Proc. IEEE Conf. Comput. Vis. Pattern Recognition (CVPR), pp. 6565–6574, 2017.

[17] A. Concha and J. Civera, "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence," in IEEE/RSJ Int. Conf. Intell. Robot. Syst., pp. 5686–5693, 2015.

[18] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 3, pp. 611–625, 2017.

[19] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-PTAM: Stereo Parallel Tracking and Mapping," Robot. Auton. Syst., vol. 93, pp. 27–42, 2017.

[20] R. W. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, "ROS: an open-source Robot Operating System," in ICRA Work. open source Softw., May. 2009.

[21] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," in 2013 IEEE Int. Symp. Safety, Secur. Rescue Robot. (SSRR), pp. 1-6, Oct. 2013.

[22] P. Sankalprajan, T. Sharma, H. D. Perur, and P. Sekhar Pagala, "Comparative analysis of ROS based 2D and 3D SLAM algorithms for autonomous ground vehicles," in 2020 Int. Conf. Emerg. Technol. (INCET), pp. 1–6, Jun. 2020.

[23] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," in 2017 14th Work. Positioning, Navig. Commun. (WPNC), pp. 1–6, Oct. 2017.

[24] Z. Xuexi, L. Guokun, F. Genping, X. Dongliang, and L. Shiliu, "SLAM algorithm analysis of mobile robot based on lidar," in 2019 Chinese Control Conf. CCC, pp. 4739–4745, Jul. 2019.

[25] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," in 2018 9th Int. Conf. Intell. Syst. (IS), pp. 400–407, Sept. 2018.

[26] M. Vlaminck, H. Luong, W. Goeman, and W. Philips, "3D scene reconstruction using Omnidirectional vision and LiDAR: A hybrid approach," Sensors, vol. 16, no. 11, 2016.

[27] C. Shi, K. Huang, Q. Yu, J. Xiao, H. Lu, and C. Xie, "Extrinsic Calibration and Odometry for Camera-LiDAR Systems," IEEE Access, vol. 7, pp. 120106–120116, 2019.

[28] N. Yu and B. Zhang, "An Improved Hector SLAM Algorithm based on Information Fusion for Mobile Robot," in 2018 5th IEEE Int. Conf. Cloud Comput. Intell. Syst. (CCIS), pp. 279–284, Nov. 2018.

[29] "Raspberry Pi Mouse," [Online]. Available: https://robots.ros.org/raspberrypimouse/.

[30] P. Thrilochan Sharma, P. Sankalprajan, A. J. Muppidi, and P. S. Pagala, "Analysis of Computational Need of 2D-SLAM Algorithms for Unmanned Ground Vehicle," in Proc. Int. Conf. Intell. Comput. Control Syst 2020, pp. 230–235, 2020.

[31] Spournias, A., Skandamis, T., Pappas, E., Antonopoulos, C., and Voros, N., "Enchancing SLAM method for mapping and tracking using a low cost laser scanner," in 2019 10th Int. Conf. Infor., Intell., Syst. Appli., pp. 1-4, Jul. 2019.

[32] S. Kohlbrecher, "Hector SLAM," [Online]. Available: http://wiki.ros.org/hector_slam.

[33] "Hokuyo," [Online]. Available: https://hokuyo-usa.com/products/lidar-obstacle-detection/urg-04lx-ug01