 DOI: 10.31276/VJST.66(10DB).02-09

Ứng dụng SLAM và tính toán song song GPU trên Jetson Orin cho ước lượng vị trí máy ảnh 3D thời gian thực

Nguyễn Ngọc Tú*, Trần Xuân Thịnh

Trung tâm Quang điện tử, Viện Ứng dụng Công nghệ, C6, phường Thanh Xuân Bắc, quận Thanh Xuân, Hà Nội, Việt Nam

Ngày nhận bài 29/7/2024; ngày chuyển phản biện 1/8/2024; ngày nhận phản biện 8/8/2024; ngày chấp nhận đăng 27/8/2024

Tóm tắt:

Ước lượng vị trí của máy ảnh trong không gian 3D có nhiều ứng dụng quan trọng trong các hệ thống quét 3D và robot di động. Trong bài báo này, nhóm tác giả nghiên cứu phương pháp ước lượng vị trí của máy ảnh trong không gian 3D bằng thuật toán SLAM và sử dụng sức mạnh tính toán của GPU trên máy tính nhúng Jetson Orin nhằm phát triển các ứng dụng định vị trong thời gian thực. Bằng cách triển khai các thuật toán visual odometry trên nền tảng CUDA, kết quả của thực nghiệm đã đạt được tốc độ xử lý nhanh hơn đáng kể và độ chính xác cao hơn so với các phương pháp dựa trên CPU và cho thấy sự cải thiện hiệu suất đáng kể trong các kịch bản thời gian thực, cũng như tiềm năng ứng dụng trong robot tự hành và công nghệ quét 3D. Hệ thống xử lý hiệu quả việc trích xuất và so khớp đặc trưng sử dụng các mô tả nhị phân, tối ưu hóa quy trình thông qua tính toán song song trên GPU. Nghiên cứu cung cấp đánh giá chi tiết về độ chính xác và tốc độ của hệ thống, nêu bật những lợi thế của việc sử dụng GPU trên hệ nhúng trong ước lượng vị trí của máy ảnh 3D giúp cho các nghiên cứu thiết kế, chế tạo các thiết bị quét 3D cầm tay ngoài trời nhỏ gọn mà vẫn đáp ứng được thời gian thực.

Từ khóa: **máy ảnh thời gian thực, ORB-SLAM GPU, robot, SLAM, SLAM 3D, visual odometry, visual SLAM.**

Chỉ số phân loại: **1.1, 1.2, 2.2**

Real-time 3D camera pose estimation using SLAM and GPU parallel computing on Jetson Orin

Ngoc Tu Nguyen*, Xuan Thinh Tran

Center for Optoelectronics, National Center for Technological Progress, C6, Thanh Xuan Bac Ward, Thanh Xuan District, Hanoi, Vietnam

Received 29 July 2024; revised 8 August 2024; accepted 27 August 2024

Abstract:

Estimating the position of a camera in 3D space has numerous important applications in 3D scanning systems and mobile robotics. In this article, the authors investigate a method for estimating the camera's position in 3D space using the simultaneous localisation and mapping (SLAM) algorithm, leveraging the computational power of the graphics processing unit (GPU) on the Jetson Orin embedded computer to develop real-time localisation applications. By implementing visual odometry algorithms on the compute unified architecture (CUDA) platform, experimental results demonstrated a significantly faster processing speed and higher accuracy compared to CPU-based methods, showing remarkable performance improvements in real-time scenarios. This underscores the potential for application in autonomous robotics and 3D scanning technology. The system efficiently processes feature extraction and matching using binary descriptors, optimising the workflow through parallel computations on the GPU. The research provides a detailed assessment of the accuracy and speed of the system, highlighting the advantages of utilising GPUs on embedded systems for 3D camera position estimation, thereby facilitating the design and fabrication of compact outdoor handheld 3D scanning devices that can still meet real-time requirements.

Keywords: **ORB-SLAM GPU, real-time camera, robot, SLAM, SLAM 3D, visual odometry, visual SLAM.**

Classification numbers: **1.1, 1.2, 2.2**

Tác giả liên hệ: Email: ngoctu@cfoc.vn

1. Đặt vấn đề

Ước lượng vị trí của máy ảnh trong không gian 3D là một yếu tố then chốt trong nhiều ứng dụng hiện đại như robot tự hành, thực tế ảo và quét 3D. Độ chính xác và tốc độ của quá trình này ảnh hưởng trực tiếp đến hiệu suất của các hệ thống. Trong các nghiên cứu trước đây, các thuật toán ước lượng vị trí [1] và lập bản đồ đồng thời (SLAM) [2, 3] dựa trên CPU gặp hạn chế về tốc độ xử lý, đặc biệt khi xử lý các dữ liệu ảnh lớn và phức tạp. Với sự phát triển của công nghệ GPU, khả năng tính toán song song mạnh mẽ của nó mang lại tiềm năng lớn để cải thiện tốc độ và độ chính xác của các thuật toán này.

Trong nghiên cứu này, tập trung vào việc sử dụng GPU để tăng tốc độ và cải thiện độ chính xác của các thuật toán ước lượng vị trí máy ảnh trong không gian 3D. Đặc biệt, sử dụng các thuật toán Visual Odometry, triển khai trên nền tảng CUDA để tận dụng khả năng tính toán song song của GPU.

Một trong những thách thức lớn của các hệ thống SLAM là xử lý khối lượng lớn dữ liệu hình ảnh một cách hiệu quả. Các nghiên cứu trước đây đã chỉ ra rằng, việc sử dụng GPU trong xử lý ảnh có thể mang lại hiệu quả cao hơn nhiều so với CPU. Ví dụ, ORB-SLAM3 sử dụng các descriptor nhị phân như ORB (Oriented FAST and Rotated BRIEF) và các kỹ thuật như FAST đã giúp giảm thời gian tính toán mà vẫn duy trì độ chính xác cao [4]. Sự kết hợp giữa các thuật toán này và khả năng xử lý mạnh mẽ của GPU mang lại khả năng thực hiện các phép tính phức tạp trong thời gian thực, mở ra nhiều ứng dụng tiềm năng trong các lĩnh vực như robot tự hành, thực tế ảo, và các hệ thống nhận dạng hình ảnh tiên tiến.

Việc sử dụng GPU không chỉ tăng tốc độ xử lý mà còn mở ra cơ hội để triển khai các thuật toán phức tạp hơn mà trước đây không khả thi trên CPU do hạn chế về tài nguyên và thời gian tính toán. Trong môi trường quét 3D không gian lớn, việc ước lượng vị trí chính xác và kịp thời là cực kỳ quan trọng để đảm bảo độ tin cậy của dữ liệu 3D khi ghép nối các mảng không gian lại thành một mô hình hoàn chỉnh. Khả năng xử lý thời gian thực của GPU giúp ứng dụng quét 3D có thể phản ứng nhanh chóng với các thay đổi trong môi trường, từ đó cải thiện hiệu suất và độ tin cậy của hệ thống.

Trong bài báo này, các tác giả trình bày về phương pháp ứng dụng GPU cho thuật toán ước lượng vị trí, bao gồm các bước thu thập dữ liệu, triển khai thuật toán, và các kết quả thử nghiệm. Bên cạnh đó, cũng thảo luận về các thách thức và hạn chế của phương pháp này, cũng như những ứng dụng tiềm năng trong tương lai. Nghiên cứu này không chỉ đánh giá hiệu suất của các hệ thống hiện có mà còn mở ra hướng đi mới cho việc ứng dụng công nghệ GPU trong lĩnh vực ước lượng vị trí và lập bản đồ 3D.

2. Phương pháp nghiên cứu

2.1. Mô hình quán tính - trực quan máy ảnh và IMU

Theo nghiên cứu do R.M. Artal và cs (2015) [5], đầu vào cho mô hình quán tính - trực quan là luồng các phép đo IMU và khung hình của máy ảnh. Dựa trên mô hình lỗ nhỏ của máy ảnh thông thường có chức năng chiếu π: $R^3 \rightarrow \Omega$ biến đổi các điểm 3D $X_c \in R^3$ trong hệ tham chiếu camera C, thành các điểm 2D trên mặt phẳng ảnh $X_c \in \Omega \subset R^2$:

$$\pi(\boldsymbol{X}_c) = \begin{bmatrix} f_u \frac{X_c}{Z_c} + c_u \\ f_v \frac{Y_c}{Z_c} + c_u \end{bmatrix}, \boldsymbol{X}_c = [\boldsymbol{X}_c \quad \boldsymbol{Y}_c \quad \boldsymbol{Z}_c]^T \qquad (1)$$

trong đó, $[f_u f_v]^T$ là tiêu cự và $[c_u c_v]^T$ là tâm điểm trong thông số nội của camera đã được hiệu chuẩn. Chức năng chiếu này không xem xét đến độ méo do ống kính máy ảnh tạo ra. Khi trích xuất các điểm chính trên hình ảnh sẽ hủy biến dạng tọa độ của chúng để chúng có thể khớp với các điểm được chiếu bằng cách sử dụng (1).

IMU, có tham chiếu được ký hiệu là B, đo gia tốc $\boldsymbol{a}_B$ và vận tốc góc ω_B của cảm biến trong những khoảng thời gian đều đặn Δt, thường ở hàng trăm Hz. Cả hai phép đo đều bị ảnh hưởng, ngoài nhiễu cảm biến, do độ lệch b_a và b_g thay đổi chậm của gia tốc kế và con quay hồi chuyển tương ứng. Hơn nữa, gia tốc kế chịu tác dụng của trọng lực g_w và người ta cần trừ đi tác dụng của nó để tính chuyển động. Sự phát triển riêng biệt của định hướng IMU $\boldsymbol{R}_{wB} \in SO$ 10, vị trí $w\boldsymbol{P}_B$ và vận tốc wV_B, trong hệ tham chiếu thế giới W, có thể được tính như sau:

$$\begin{aligned} \boldsymbol{R}_{WB}^{k+1} &= \boldsymbol{R}_{WB}^{k} Exp\left(\left(\omega_B^k - \boldsymbol{b}_g^k\right)\Delta t\right) \\ wV_B^{k+1} &= wV_B^k + \boldsymbol{g}_w \Delta t + \boldsymbol{R}_{WB}^k(\boldsymbol{a}_B^k - \boldsymbol{b}_a^k)\Delta t \\ w\boldsymbol{P}_B^{k+1} &= w\boldsymbol{P}_B^k + w\,V_B^k \Delta t + \frac{1}{2}\boldsymbol{g}_w \Delta t^2 + \frac{1}{2}\boldsymbol{R}_{WB}^k(\boldsymbol{a}_B^k - \boldsymbol{b}_a^k)\Delta t^2 \end{aligned} \qquad (2)$$

Chuyển động giữa hai khung hình chính liên tiếp có thể được xác định theo sự tích hợp trước ΔR, Δv và Δp từ tất cả các phép đo ở giữa, sử dụng quá trình tích hợp IMU gần đây được mô tả:

$$\begin{aligned} \boldsymbol{R}_{WB}^{i+1} &= \boldsymbol{R}_{WB}^{i} \Delta \boldsymbol{R}_{i,i+1} Exp\left(\left(J_{\Delta R}^g \boldsymbol{b}_g^i\right)\right) \\ wV_B^{i+1} &= wV_B^i + \boldsymbol{g}_w \Delta t_{i,i+1} + \boldsymbol{R}_{WB}^i\left(\Delta V_{i,i+1} + J_{\Delta v}^g \boldsymbol{b}_g^i + J_{\Delta v}^a \boldsymbol{b}_a^i\right) \\ w\boldsymbol{P}_B^{i+1} &= w\boldsymbol{P}_B^i + w\,V_B^i \Delta t_{i,i+1} + \frac{1}{2}\boldsymbol{g}_w \Delta t_{i,i+1}^2 + \frac{1}{2}\boldsymbol{R}_{WB}^i\left(\Delta \boldsymbol{P}_{i,i+1} + J_{\Delta v}^g \boldsymbol{b}_g^i + J_{\Delta v}^a \boldsymbol{b}_a^i\right) \end{aligned} \qquad (3)$$

trong đó, Jacobians $J_{(\cdot)}^a$ và $J_{(\cdot)}^g$ giải thích xấp xỉ bậc một về tác động của việc thay đổi độ lệch mà không cần tính toán lại một cách rõ ràng các giá trị tiền tích hợp. Cả tiền tích hợp và Jacobian đều có thể được tính toán lặp đi lặp lại một cách hiệu quả khi các phép đo IMU xuất hiện.

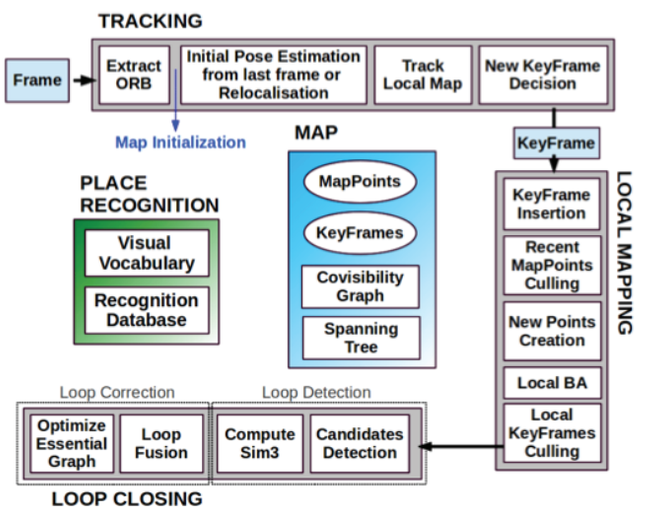
Camera và IMU được cố định với nhau, tạo thành hệ thống và phép biến đổi $T_{CB} = [R_{CB} \mid {}_C p_B]$ giữa các hệ quy chiếu của chúng đã biết từ quá trình hiệu chuẩn.

2.2. Thuật toán ước lượng vị trí trong SLAM

Thuật toán ước lượng vị trí là một phần quan trọng trong các hệ thống SLAM (Simultaneous Localization and Mapping), giúp xác định vị trí của máy ảnh trong không gian 3D. Một trong những thuật toán phổ biến nhất cho nhiệm vụ này là ORB-SLAM3 [2], sử dụng các đặc trưng ORB (Oriented FAST and Rotated BRIEF) [6] để phát hiện và mô tả các điểm đặc trưng trên hình ảnh. ORB-SLAM3 hoạt động dựa trên việc trích xuất các điểm đặc trưng từ các khung hình liên tiếp, sau đó so sánh và kết hợp chúng để ước lượng sự di chuyển của máy ảnh. Quá trình này bao gồm ba bước chính: phát hiện điểm đặc trưng, theo dõi và kết hợp các điểm này, và tối ưu hóa vị trí và bản đồ môi trường. ORB-SLAM3 sử dụng các descriptor nhị phân [4], giúp giảm thiểu thời gian tính toán và tăng cường hiệu suất, đặc biệt khi triển khai trên nền tảng GPU. Điều này cho phép hệ thống xử lý và cập nhật vị trí của máy ảnh trong thời gian thực, đáp ứng yêu cầu cao về độ chính xác và tốc độ trong các ứng dụng như robot tự hành và thực tế ảo. Sự kết hợp giữa các thuật toán phát hiện điểm đặc trưng hiệu quả và khả năng xử lý mạnh mẽ của GPU đã tạo nên một bước tiến lớn trong việc cải thiện hiệu suất và độ tin cậy của các hệ thống ước lượng vị trí.

2.2.1. Cấu trúc của chương trình ORB-SLAM

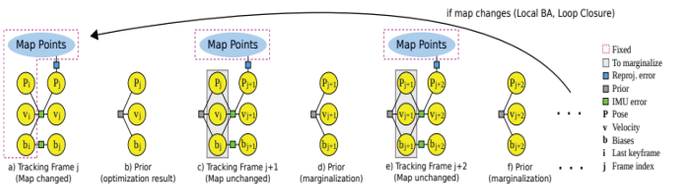
Hình 1 là tổng quan của ORB-SLAM, sử dụng các kỹ thuật tiên tiến để theo dõi và tái tạo bản đồ môi trường. Thư viện này được thiết kế để hoạt động hiệu quả trong nhiều điều kiện khác nhau, từ các môi trường trong nhà đến ngoài trời, và trong các tình huống di chuyển nhanh chóng (GalvezTRO12) [4].



Hình 1. Tổng quan của một hệ thống ORB-SLAM [5].

Các hệ thống SLAM truyền thống thường dựa vào các thuật toán như FAST và BRIEF để trích xuất và mô tả các đặc trưng của ảnh. Ví dụ, các nghiên cứu của D.G. López và cs (2012) [4] đã chỉ ra rằng, sử dụng các túi từ nhị phân giúp nhận diện vị trí nhanh chóng trong các chuỗi hình ảnh, từ đó tăng hiệu suất của hệ thống [6] (GalvezTRO12). Phương pháp này không chỉ tăng tốc độ mà còn giảm thiểu sai sót trong việc nhận diện vị trí, điều này đặc biệt quan trọng trong các ứng dụng yêu cầu độ chính xác cao và thời gian thực như robot tự hành và thực tế ảo.

i) Theo dõi (Tracking): Tính năng theo dõi quán tính trực quan chịu trách nhiệm theo dõi tư thế cảm biến, vận tốc và độ lệch IMU ở tốc độ khung hình. Điều này cho phép dự đoán tư thế của máy ảnh rất đáng tin cậy, thay vì sử dụng mô hình chuyển động đặc biệt như trong monocular system ban đầu. Sau khi dự đoán được tư thế của máy ảnh, các điểm trên bản đồ cục bộ sẽ được chiếu và khớp với các điểm chính trên khung. Sau đó, tối ưu hóa khung j hiện tại bằng cách giảm thiểu lỗi chiếu lại đặc trưng của tất cả các điểm trùng khớp và thời gian lỗi IMU. Việc tối ưu hóa này khác nhau tùy thuộc vào bản đồ có được cập nhật hay không bởi Luồng Local Mapping hoặc Loop Closing (hình 2).



Hình 2. Sự phát triển của việc tối ưu hóa trong chuỗi theo dõi [5].

ii) Bản đồ hóa cục bộ (Local Mapping): Chuỗi bản đồ hóa cục bộ thực hiện BA (gói điều chỉnh) cục bộ sau khi chèn khóa khung hình mới. Nó tối ưu hóa N khóa khung hình cuối (cửa sổ cục bộ) và tất cả các điểm mà N khung hình chính đó nhìn thấy. Tất cả các khung hình chính khác chia sẻ quan sát về các điểm cục bộ (tức là được kết nối trong biểu đồ khả năng hiển thị với bất kỳ khung hình chính cục bộ nào), nhưng không có trong cửa sổ cục bộ, đóng góp vào tổng chi phí nhưng được cố định trong quá trình tối ưu hóa (cửa sổ cố định). Khung hình chính N+1 luôn được bao gồm trong cửa sổ cố định vì nó hạn chế trạng thái IMU.

iii) Đóng vòng lặp (Loop Closing): Luồng đóng vòng lặp nhằm mục đích giảm độ trôi tích lũy trong quá trình thăm dò, khi quay trở lại khu vực đã được lập bản đồ. Mô-đun nhận dạng địa điểm khớp khung hình chính gần đây với khung hình chính trong quá khứ. Sự trùng khớp này được xác thực bằng cách tính toán một phép biến đổi rigid body (cơ thể cứng) để căn chỉnh các điểm khớp giữa các khung hình chính. Cuối cùng, việc tối ưu hóa được thực hiện để giảm sai số tích lũy trong quỹ đạo. Việc tối ưu hóa này có thể rất tốn kém trong các bản đồ lớn, do đó chiến lược là thực hiện tối ưu hóa biểu đồ tư thế, giúp giảm độ phức tạp

vì cấu trúc bị bỏ qua và thể hiện sự hội tụ tốt. Ngược lại với ORB-SLAM ban đầu, thực hiện tối ưu hóa biểu đồ tư thế trên 6 bậc tự do (DoF) thay vì 7 DoF, vì có thể quan sát được tỷ lệ. Biểu đồ tư thế này bỏ qua thông tin IMU, không tối ưu hóa vận tốc hoặc độ lệch IMU. Do đó, điều chỉnh vận tốc bằng cách xoay chúng theo hướng đã sửa của khung hình chính liên quan. Mặc dù điều này không tối ưu nhưng độ lệch và vận tốc phải chính xác cục bộ để tiếp tục sử dụng thông tin IMU ngay sau khi tối ưu hóa sơ đồ. Sau đó, thực hiện BA (gói điều chỉnh) đầy đủ trong một luồng song song nhằm tối ưu hóa tất cả các trạng thái, bao gồm cả vận tốc và độ lệch.

2.2.2. Thuật toán phát hiện keypoints và mô tả đặc trưng

i) Thuật toán FAST: Là một phương pháp phát hiện đặc trưng hiệu quả và nhanh chóng trong lĩnh vực xử lý ảnh và thị giác máy tính, nổi bật với khả năng xác định các điểm đặc trưng một cách chính xác và tốc độ cao [7] hơn so với thuật toán SIFT [8] và SUSFT [9]. Thuật toán này bắt đầu bằng cách chọn một điểm trung tâm trong hình ảnh và so sánh cường độ của điểm này với cường độ của các điểm ảnh nằm trên một vòng tròn có bán kính 3 điểm ảnh xung quanh điểm trung tâm.

Nhờ vào tính đơn giản và hiệu quả của thuật toán, FAST trở thành một công cụ mạnh mẽ trong nhiều ứng dụng thị giác máy tính, bao gồm theo dõi đối tượng, nhận dạng hình ảnh, và đặc biệt là trong các hệ thống SLAM dùng trong robot tự hành. Trong các hệ thống SLAM, việc phát hiện nhanh chóng và chính xác các điểm đặc trưng là yếu tố quan trọng để xây dựng bản đồ và định vị robot trong môi trường. Thuật toán FAST cũng thường được kết hợp với các thuật toán mô tả đặc trưng khác như BRIEF hoặc ORB để tạo ra các vector đặc trưng mô tả chính xác và hiệu quả cho các điểm đặc trưng được phát hiện.

ii) Mô tả bằng BRIEF: Là một trong những phương pháp quan trọng và phổ biến trong lĩnh vực thị giác máy tính, được sử dụng để biểu diễn các điểm đặc trưng trên hình ảnh dưới dạng chuỗi bit nhị phân [10]. Đây là một công cụ mạnh mẽ và đơn giản, đặc biệt phù hợp cho các ứng dụng yêu cầu tính nhanh và độ tin cậy cao như nhận diện đối tượng, theo dõi vật thể và xây dựng bản đồ 3D trong hệ thống SLAM.

Quá trình tạo ra mô tả đặc trưng bằng BRIEF bắt đầu bằng việc chọn các điểm ảnh quan tâm từ hình ảnh, thường thông qua các thuật toán phát hiện điểm nổi bật như FAST hoặc ORB. Sau khi các điểm ảnh quan tâm được xác định, một vùng xung quanh mỗi điểm ảnh này với kích thước (ví dụ, pixel) sẽ được chọn để phân tích.

Trong vùng này, một tập hợp các cặp điểm ngẫu nhiên được chọn ra. Mỗi cặp điểm này được đại diện bởi hai tọa độ (x_i, y_i) và (x_i', y_i'). Số lượng các cặp điểm thường được xác định trước và là một tham số của phương pháp, chẳng hạn n=256 cặp. Với mỗi cặp điểm, độ sáng của các điểm ảnh tại hai tọa độ này sẽ được so sánh. Nếu độ sáng tại tọa độ (x_i, y_i) nhỏ hơn độ sáng tại tọa độ (x_i', y_i'), kết quả so sánh sẽ là 1, ngược lại sẽ là 0. Kết quả của các phép so sánh này tạo thành một vector nhị phân với độ dài nnn bit, ví dụ 256 bit, biểu diễn mô tả đặc trưng cho điểm ảnh quan tâm.

Các chuỗi bit mô tả được tạo ra từ BRIEF không chỉ thực hiện nhanh chóng mà còn đảm bảo độ chính xác và tin cậy, thích hợp cho việc nhận diện các điểm đặc trưng trong các điều kiện khác nhau của hình ảnh, bao gồm cả sự biến đổi hình học và điều kiện chiếu sáng.

iii) Mô tả nhị phân bằng ORB: Là một phương pháp quan trọng trong thị giác máy tính, được sử dụng để biểu diễn các điểm đặc trưng trên hình ảnh dưới dạng chuỗi bit nhị phân [6]. ORB kết hợp giữa hai thành phần chính là thuật toán FAST và BRIEF, để cung cấp một cách tiếp cận hiệu quả và nhanh chóng cho việc nhận diện và mô tả các điểm đặc trưng.

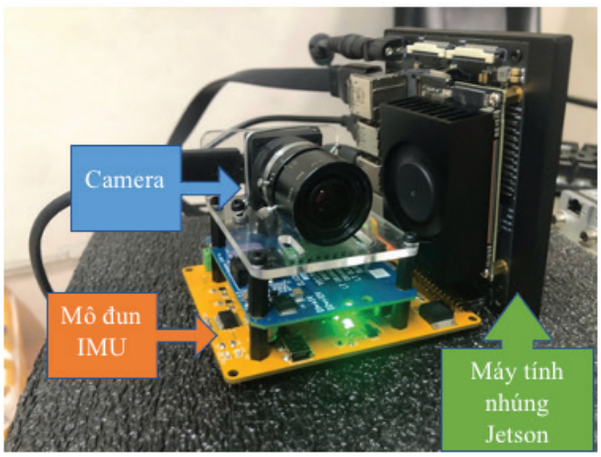
Thuật toán FAST được sử dụng để nhanh chóng phát hiện các điểm đặc trưng trên hình ảnh, dựa trên việc so sánh độ sáng của các điểm ảnh trong một vùng xung quanh. Sau khi các điểm đặc trưng được phát hiện, ORB sử dụng thuật toán BRIEF để tạo ra mô tả nhị phân cho từng điểm đặc trưng. Quá trình này bao gồm việc so sánh từng cặp điểm ảnh lân cận và ghi lại kết quả dưới dạng chuỗi bit, biểu diễn các đặc điểm quan trọng của điểm đặc trưng dựa trên sự khác biệt giữa các giá trị điểm ảnh.

Mô tả nhị phân bằng ORB nổi bật với tính nhanh chóng và độc lập với mô hình, giúp cho thuật toán có thể áp dụng hiệu quả trong các hệ thống thời gian thực và trong các điều kiện biến đổi hình học của hình ảnh. Đồng thời, ORB cũng mang lại độ tin cậy cao và khả năng hoạt động ổn định trong các ứng dụng như nhận diện đối tượng, theo dõi vật thể và trong hệ thống xây dựng bản đồ 3D như SLAM.

3. Thí nghiệm và cài đặt

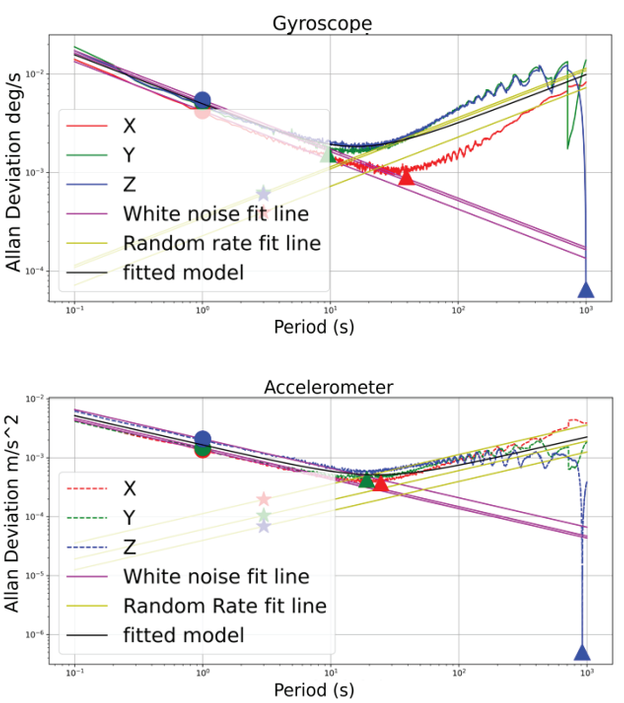
3.1. Hiệu chuẩn máy ảnh và IMU

Hệ thực nghiệm cấu hình gồm: 1 camera USB 3.0, global shutter, độ phân giải 1920x1200 điểm ảnh, 1 cảm biến IMU VN-100s, cùng với một máy tính Jetson ORIN NX 16G, hệ điều hành Ubuntu 20.04 và ROS Noetic (hình 3). Để hiệu chỉnh camera và IMU sử dụng công cụ Kalibr do J. Rehder và cs (2016) [11] phát triển. Quá trình cài đặt bao gồm việc cài đặt các phụ thuộc cần thiết, clone repository của Kalibr từ GitHub và build Kalibr cũng như cấu hình môi trường làm việc.

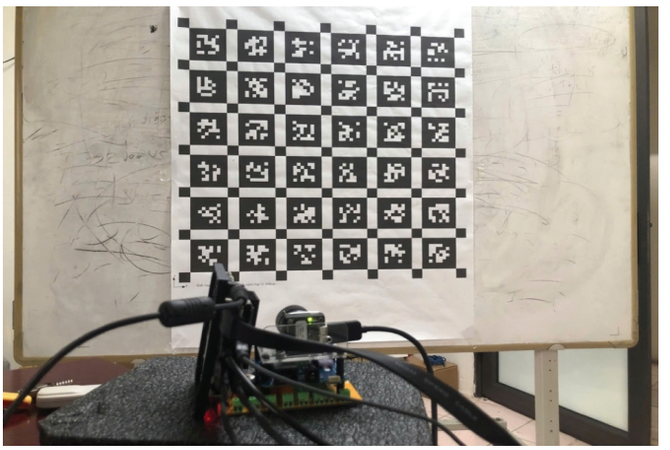


Hình 3. Hệ thí nghiệm phần cứng.

Tiến hành thu thập dữ liệu từ camera (sử dụng gói công cụ camera_aravis: *roslaunch camera_aravis camera_aravis.launch*) và IMU (sử dụng gói công cụ imu_vn_100:*roslaunch imu_vn_100 vn_100_cont.launch*). Đảm bảo rằng cả camera và IMU đều đã được kết nối và ROS master đã được khởi động, sử dụng công cụ rosbag để ghi lại dữ liệu (*rosbag record-O camera-imu-data.bag/imu/imu/Basler/Basler/image_raw*). Dữ liệu từ camera được thu thập bằng cách ghi lại topic */Blasler/Basler/image_raw* trong khi dữ liệu từ IMU (hình 4) được ghi lại từ topic */imu/imu.* Để đảm bảo tính đồng bộ, thực hiện ghi lại dữ liệu đồng thời từ cả hai thiết bị bằng cách ghi lại cả hai topic cùng lúc.

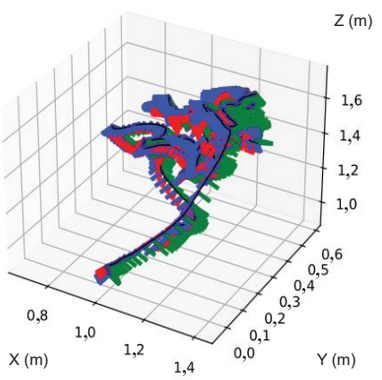


Hình 4. Biểu đồ nhiễu nền Gyroscope và Accelerometer của IMU VN-100S.

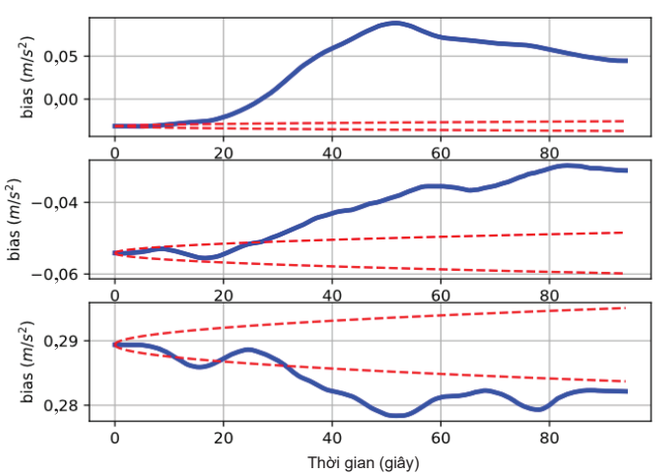


Hình 5. Hiệu chỉnh với kabilr dùng aprilgrid với thông số: Rows: 6, Cols: 6, Size: 0,098 (m), Spacing 0,0294 (m).

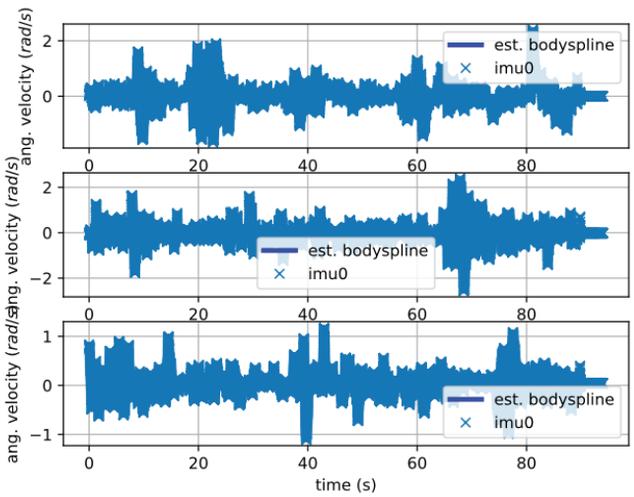
Cấu hình cho Kalibr chứa thông tin về các tham số của camera và IMU và thông số mẫu aprilgrid (hình 5). Quá trình hiệu chỉnh được thực hiện bằng cách sử dụng Kalibr với các file cấu hình và dữ liệu đã thu thập (*rosrun kalibr kalibr_calibrate_imu_camera –cam cam_calc.yaml–target aprilgrid_6x6.yaml–imu imu_vn100s.yaml–bag camera-imu-data.bag*). Kalibr xuất ra file kết quả hiệu chỉnh (hình 6-11).



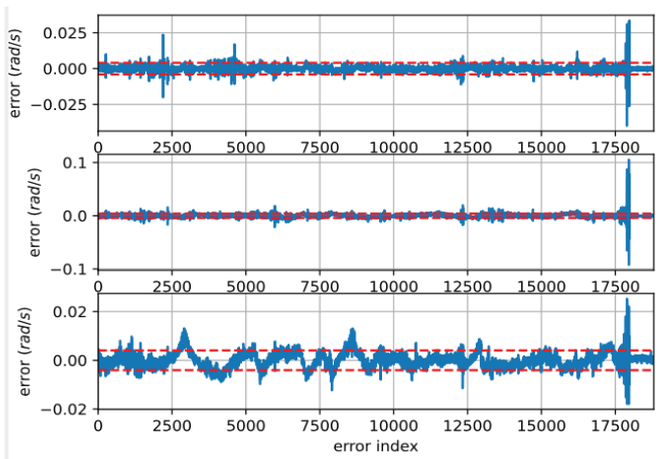
Hình 6. Tư thế ước tính của camera và IMU trong không gian 3 chiều (X, Y, Z).



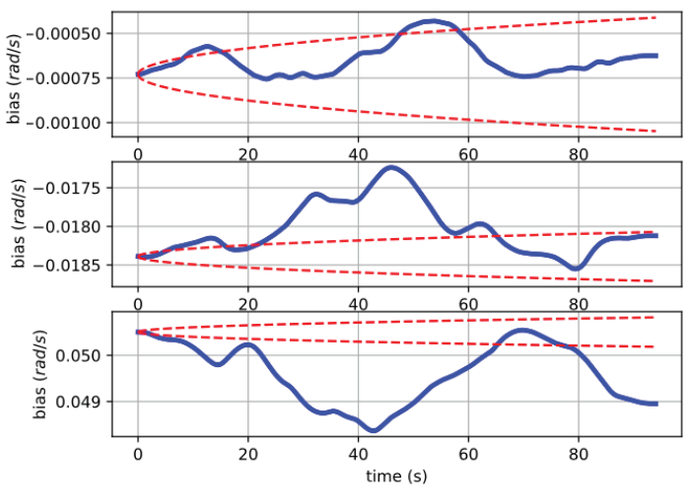
Hình 7. Độ lệch gia tốc ước tính của các trục X, Y, Z theo thời gian (IMU frame).



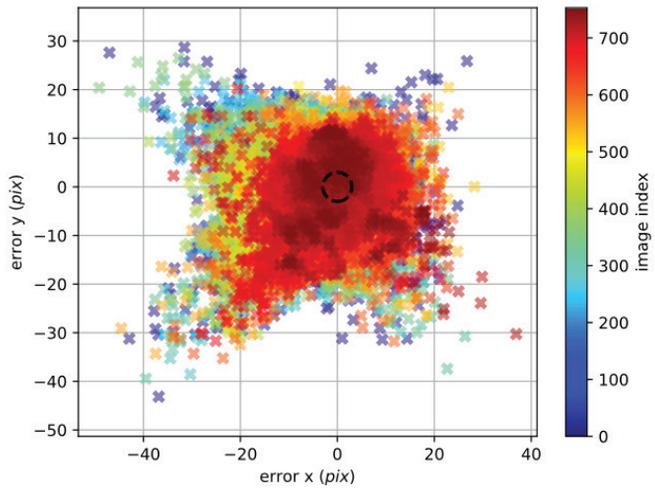
Hình 8. So sánh vận tốc góc dự đoán và đo được (body frame).



Hình 9. Lỗi vận tốc góc của IMU.



Hình 10. Ước tính độ lệch con quay hồi chuyển (khung IMU).



Hình 11. Biểu đồ lỗi phép chiếu các điểm trên mẫu aprilgrid của camera.

Cuối cùng, sử dụng tệp kết quả hiệu chỉnh để cấu hình hệ thống ORB-SLAM3. Điều này đảm bảo rằng, dữ liệu từ camera và IMU đã được hiệu chỉnh trước khi sử dụng trong quá trình xây dựng bản đồ 3D. Quá trình này giúp đảm bảo tính chính xác và hiệu quả của hệ thống trong việc tạo ra bản đồ 3D từ dữ liệu thu thập được. Bảng 1 thông số hiệu chuẩn của camera để thực hiện khử méo ảnh và xác định tọa độ điểm 3D của các Keypoints, bảng 2 là thông số ma trận chuyển vị giữa camera và IMU để ước lượng tư thế của camera, bảng 3 là thông số đặc tính của cảm biến IMU nhằm loại bỏ nhiễu nền.

Bảng 1. Bảng thông số hiệu chuẩn của camera.

Thông số	Fx	Fy	Cx	Cy	k1	k2	P1	P2
Giá trị	1361,6673	1364,9869	952,5862	592,1588	-0,2407	0,17603	0,0004	-0,0002

Bảng 2. Bảng thông số ma trận chuyển vị giữa camera và IMU.

Thông số	R			T
Giá trị	-0,01271982	-0,03139764	0,99942603	0,12018051
	0,99983075	0,01288734	0,01312984	-0,03879424
	-0,01329219	-0,25934957	0,0312284	-0,04984265
	0	0	0	1

Bảng 3. Bảng thông số xác định đặc tính của IMU VN-100S.

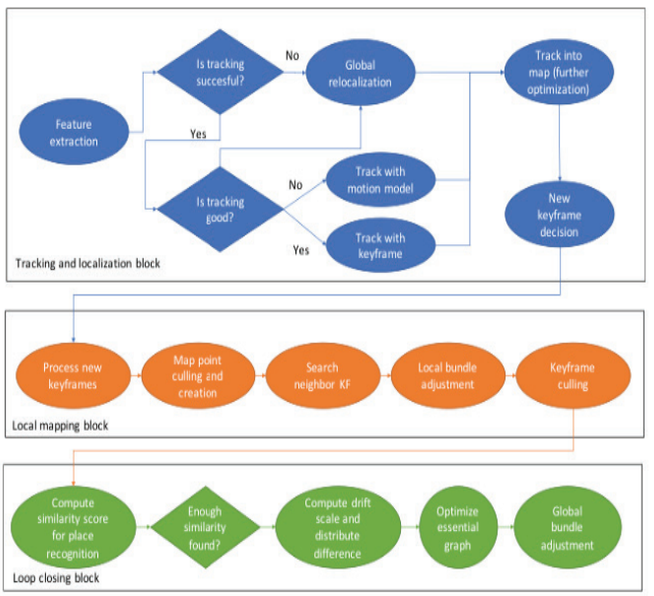
Thông số	NoiseGyro	NoiseAcc	GyroWalk	AccWalk	Frequency (Hz)
Giá trị	9,53685e-05	0,002086	1,0934e-05	0,000195764	400

3.2. Thực nghiệm

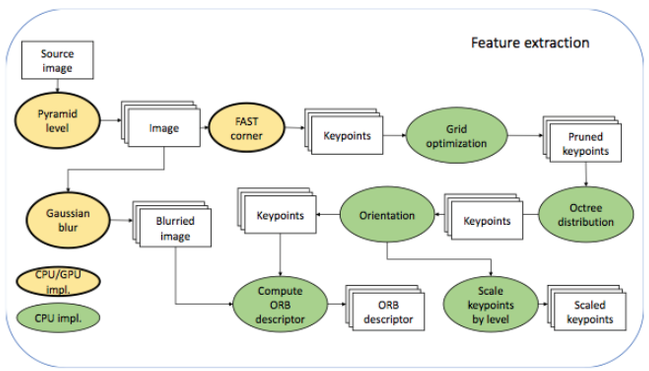
Sơ đồ các khối chương trình được thực hiện tương tự như trong ORB-SLAM theo hình 12, 13. Trong đó, khối chức năng màu vàng (sơ đồ khối hình 13) có thể được thực thi trên cả GPU và CPU, pyramid level thực hiện tạo các mức độ phân giải (cấu hình 8 độ phân giải giảm mẫu) trước khi sử dụng để phát hiện điểm đặc trưng từ ảnh (FAST Corner) và bộ lọc Gaussian Blur có tác dụng làm mờ hình

ảnh để giảm thiểu các chi tiết không cần thiết và nhiễu trong dữ liệu hình ảnh trước khi thực hiện các bước xử lý tiếp theo, như phát hiện và mô tả các điểm đặc trưng với ORB.

Tiến hành thực nghiệm với dữ liệu ảnh từ máy ảnh, IMU với lần lượt sử dụng các phương pháp tính toán Keypoints FAST và trích xuất đặc trưng ORB (sử dụng GPU và CPU theo hình 14 và khối các chức năng ORB-SLAM2 hình 15) và các thông số theo bảng 1, 2, 3, 4 thực thi để so khớp và tìm điểm tương đồng nhằm ước lượng vị trí của hệ máy ảnh-IMU trong không gian 3D (*rosrun ORB_SLAM3 Mono_Inertial ./Vocabulary/ORBvoc.txt ./Examples/ROS/ORB_SLAM3/cam_imu.yaml*).



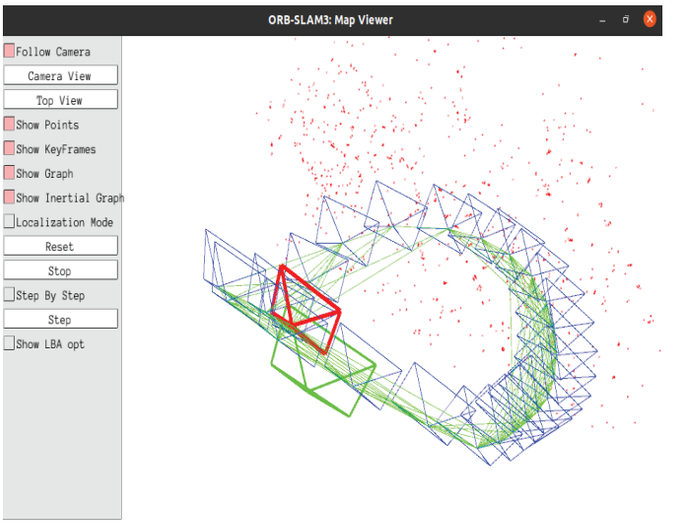
Hình 12. Các khối chức năng trong chương trình ORB-SLAM2 [12].



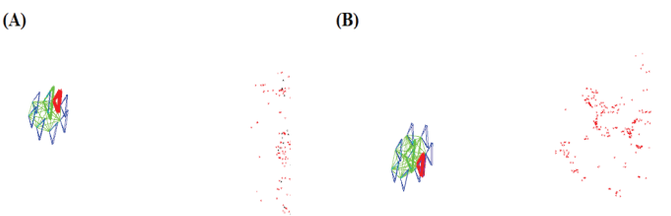
Hình 13. Sơ đồ khối xử lý cho 2 trường hợp dùng CPU và GPU [9] **cho tính toán keypoints và trích xuất đặc trưng.**

Bảng 4. Cài đặt thông số cho ORB-SLAM.

Thông số	nFeatures	scaleFactor	nLevels	iniThFAST	minThFAST
Giá trị	2000	1,2	8	20	7



Hình 14. Ước lượng vị trí của máy ảnh trong không gian 3D.



Hình 15. Vị trí của camera quỹ đạo hình tròn bán kính 0,5 m trên GPU (A) và CPU (B).

4. Kết quả và bàn luận

Hình 16 thể hiện kết quả được tính toán xử lý trên GPU. Mỗi khung hình được thực hiện theo các bước tính toán Keypoints bằng phương pháp FAST, tính toán mô tả đặc trưng ORB, ước lượng tư thế máy ảnh với dữ liệu thu được từ IMU. Sau khi tìm được các điểm tương đồng tiến hành xác định vị trí của hệ máy ảnh-IMU trong không gian 3D: các điểm màu đỏ là các điểm tương đồng giữa các khung hình, các ô màu xanh nước biển (blue) thể hiện tư thế của máy ảnh trong không gian 3D, các đường màu đỏ là thể hiện quỹ đạo di chuyển của máy ảnh trong không gian 3D (hình 14).

Kết quả thực nghiệm khi sử dụng 2 chế độ CPU và GPU cho thấy: khi sử dụng GPU, tốc độ xử lý của hệ thống (độ phân giải 1920x1200) đạt trung bình 31,5 FPS nhanh hơn so với 9,3 FPS trên CPU.

Để đánh giá sai số ước lượng ví trí của camera, thực hiện di chuyển camera theo quỹ đạo hình tròn, bán kính 0,5 m, xác định tâm đường tròn và ước lượng đường kính trung bình từ tập dữ liệu tọa độ điểm của camera trong không gian (hình 15). Hình 15A là quỹ vị trí của camera xử lý trên GPU với sai số trung bình là 0,017 m và hình 15B là quỹ đạo vị trí thực hiện cùng tập dữ liệu thực thi trên CPU với sai số trung bình 0,039 m.

Hình 16. Các điểm tương đồng (màu xanh) được phát hiện giữa 2 khung hình liên tiếp sau khi tính toán các Keypoint bằng FAST và so khớp đặc trưng ORB.

Từ thực nghiệm đánh giá hiệu suất của thuật toán SLAM trên CPU và GPU, chúng ta có thể thấy, GPU cho hiệu suất cao hơn cũng như đem lại độ chính xác hơn trên Jetson Orin NX. Kết quả này hoàn toàn có thể đáp ứng yêu cầu xử lý thời gian thực cho ước lượng vị trí của máy ảnh, với chi phí thấp và phát triển cho nhiều ứng dụng.

5. Kết luận

Từ nghiên cứu về ước lượng vị trí máy ảnh trong không gian 3D thông qua thuật toán OLB-SLAM3 sử dụng GPU trên nền tảng Jetson Orin, cho thấy những tiến bộ đáng kể trong cả tốc độ xử lý và độ chính xác vị trí so với thực hiện trên CPU. Việc áp dụng sức mạnh tính toán của GPU đã cho phép triển khai các thuật toán Visual Odometry một cách hiệu quả, ưu điểm của OLB-SLAM3 trên GPU gồm: i) Tốc độ xử lý cao: Sử dụng GPU giúp cải thiện tốc độ tính toán thông qua xử lý song song, rất cần thiết cho các ứng dụng thời gian thực; ii) Độ chính xác được cải thiện: Thực nghiệm chỉ ra rằng, phương pháp này đạt được độ chính xác cao hơn trong việc ước lượng vị trí so với các phương pháp dựa trên CPU; iii) Hệ thống có khả năng đáp ứng tốt trong các tình huống phức tạp, cho phép các kịch bản đa dạng trong không gian 3D. Nhược điểm gồm: i) Việc tối ưu hóa thuật toán cho GPU có thể tốn nhiều thời gian và công sức hơn so với các phương pháp dựa trên CPU; ii) Mặc dù Jetson Orin mạnh mẽ, nhưng vẫn có giới hạn về tài nguyên so với các cụm GPU chuyên dụng, có thể ảnh hưởng đến khả năng mở rộng của hệ thống. Để cải thiện hiệu suất của OLB-SLAM3, các hướng nghiên cứu có thể tập trung vào: i) Tối ưu hóa thuật toán: Nghiên cứu các phương pháp tối ưu hóa hơn nữa cho thuật toán SLAM, nhằm giảm thiểu độ trễ trong quá trình xử lý và cải thiện độ chính xác; ii) Mở rộng tính khả thi: Tìm kiếm các kiến trúc GPU khác, hoặc thậm chí là các giải pháp kết hợp đa dạng hơn, chẳng hạn như sử dụng máy học để cải thiện khả năng nhận diện và so khớp đặc trưng.

LỜI CẢM ƠN

Nghiên cứu này được thực hiện nhờ sự tài trợ từ đề tài cấp Bộ Khoa học và Công nghệ: "Nghiên cứu thiết kế, chế tạo thiết bị Lidar 3D cầm tay quét hiện trường". Các tác giả xin chân thành cảm ơn.

TÀI LIỆU THAM KHẢO

[1] S. Jung, S. Park, K.T. Lee (2021), "Pose tracking of moving sensor using monocular camera and IMU sensor", *KSII Transactions on Internet and Information Systems*, **15(8)**, pp.3011-3024, DOI: 10.3837/tiis.2021.08.017.

[2] C. Campos, R. Elvira, J.J.G. Rodríguez, et al. (2021), "ORB-SLAM3: An accurate open-source library for visual visual-inertial and multimap SLAM", *IEEE Transactions on Robotics*, **37(6)**, pp.1874-1890, DOI: 10.1109/TRO.2021.3075644.

[3] A. Aguiar, F. Santos, A.J. Sousa, et al. (2019), "FAST-FUSION: An improved accuracy omnidirectional visual odometry system with sensor fusion and GPU optimization for embedded low cost hardware", *Applied Sciences*, **9(24)**, DOI: 10.3390/app9245516.

[4] D.G. López, J.D. Tardós (2012), "Bags of binary words for fast place recognition in image sequences", *IEEE Transactions on Robotics*, **28(5)**, pp.1188-1197, DOI: 10.1109/TRO.2012.2197158.

[5] R.M. Artal, J.M.M. Montiel, J.D. Tardós (2015), "ORB-SLAM: A versatile and accurate monocular SLAM system", *IEEE Transactions on Robotics*, **31(5)**, pp.1147-1163, DOI: 10.1109/TRO.2015.2463671.

[6] E. Rublee, V. Rabaud, K. Konolige, et al. (2011), "ORB: An efficient alternative to SIFT or SURF", *2011 International Conference on Computer Vision*, pp.2564-2571, DOI: 10.1109/ICCV.2011.6126544.

[7] E. Rosten, T. Drummond (2006), "Machine learning for high-speed corner detection", *Computer Vision - ECCV 2006*, Springer, **3951**, DOI: 10.1007/11744023_34.

[8] D. Lowe (2004), "Distinctive image features from scale-invariant keypoints", *International Journal on Computer Vision (IJCV)*, **60**, pp.91-110, DOI: 10.1023/B:VISI.0000029664.99615.94.

[9] H. Bay, T. Tuytelaars, L.V. Gool (2006), "SURF: Speeded up robust features", *Computer Vision - ECCV 2006*, Springer, **3951**, DOI: 10.1007/11744023_32.

[10] M. Calonder, V. Lepetit, C. Strecha, et al. (2010), "BRIEF: Binary robust independent elementary features", *Computer Vision - ECCV 2010*, Springer, pp.778-792, DOI: 10.1007/978-3-642-15561-1_56.

[11] J. Rehder, J. Nikolic, T. Schneider, et al. (2016), "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes", *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp.4304-4311, DOI: 10.1109/ICRA.2016.7487628.

[12] S. Aldegheri, N. Bombieri, D.D. Bloisi, et al. (2016), "Data flow ORB-SLAM for real-time performance on embedded GPU boards", *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.5370-5375, DOI: 10.1109/IROS40897.2019.8967814.