RTAB-SLAM with Integration of IMU Data for Indoor Mapping

Abhinav Vaddiraju
Northeastern University
Boston, MA, USA
vaddiraju.a@northeastern.edu

Kiran Sairam Bethi Balagangadaran
Northeastern University
Boston, MA, USA
bethi.k@northeastern.edu

Prasath Saravanan
Northeastern University
Boston, MA, USA
saravanan.pras@northeastern.edu

***Abstract*—This paper presents an implementation of RGB-D SLAM using RTAB-MAP (Real-Time Appearance-Based Mapping) with Intel RealSense D435 camera for indoor mapping applications. The system demonstrates successful 3D reconstruction, visual odometry, and loop closure detection in residential environments. Key achievements include dense point cloud reconstruction with 150-250 tracked features in textured regions and successful loop closure detection with 2-3% translational drift. Primary challenges encountered include timestamp synchronization between sensors, depth quality degradation beyond 2.5m, and tracking difficulties in featureless environments. We would also like to declare that our overview and team results are consistent for the entire group.**

I. Overview

A. Project Description and Applications

Simultaneous Localization and Mapping (SLAM) is fundamental to mobile robotics, enabling autonomous systems to construct maps while determining their position within unknown environments. This ability is important for warehouse automation, autonomous vehicles in GPS-denied areas (tunnels, urban canyons), and exploration robots in hazardous environments [1]. The market for autonomous mobile robotics is projected to reach $10 billion by 2030, with SLAM serving as its foundational technology [2].

This project implements RGB-D SLAM using RTAB-MAP (Real-Time Appearance-Based Mapping) [3] with the Intel RealSense D435 camera. RGB-D SLAM offers advantages for indoor environments: dense 3D reconstruction, robustness in low-texture areas where monocular SLAM fails, and less intense computation than LiDAR systems while keeping similar accuracy. These characteristics make it well-suited for indoor mobile robotics, AR/VR spatial mapping, and human-robot interaction applications.

B. Approach and Algorithms

RTAB-MAP (Real-Time Appearance-Based Mapping) is an open-source graph-based SLAM algorithm developed by Mathieu Labbé at Université de Sherbrooke [3]. It uses incremental appearance-based loop closure detection with memory management for long-term operation. The pipeline consists of:

- **Visual Odometry:** Frame-to-frame pose estimation using GFTT/ORB features with RANSAC outlier rejection
- **Loop Closure Detection:** Bag-of-words approach to identify revisited locations and correct drift
- **Graph Optimization:** g2o backend minimizes pose graph error when loop closures detected
- **IMU Integration:** Provides improved pose estimation during fast motions and orientation constraints

Alternatives considered include ORB-SLAM3 [4] (no memory management), Google Cartographer (2D/LiDAR-focused), and LOAM (LiDAR-only). RTAB-MAP was selected for RGB-D optimization, constant up-to-date repository, memory management, and ROS2 support.

C. Sensing and Data Collection

Sensors: Intel RealSense D435 (RGB-D, 1280×720, 30Hz, active IR stereo, 0.3-3m range) and VectorNav VN-100 IMU (100Hz, 9-DOF) [6].

Data Collection Method: Data collected via ROS2 rosbags in an indoor residential environment (living room) with intentionally random scattered objects to provide visual features for reliable loop closure and feature tracking/mapping. Trajectories included loop closures and a well lit condition. Hardware used: Intel i7-13750H CPU, RTX 4070 GPU and 16 GB of RAM in the form of an Asus Zephyrus G16 running Ubuntu 24.04 and ROS2 Jazzy for processing.

D. Project Levels

Achieved:

- RTAB-MAP implementation with RGB-D camera
- Visual feature-based loop closure detection
- 3D point cloud reconstruction and mapping
- Manual handheld mapping capability

Although all of the above were successfully implemented, the primary challenge we faced was persistent frame dropping. Symptoms included RTAB-Map warnings about missing frames and discontinuous point clouds. Timestamp analysis confirmed frames were lost rather than delayed, indicating bandwidth or processing bottlenecks rather than simple latency issues.

In Progress/Partially Achieved:

- IMU integration for improved pose estimation
- Sensor fusion between RGB-D and IMU data
- Timestamp synchronization between sensors

Challenges Encountered:

- D435 IR struggles with bright lights, reflective surfaces, and textureless walls

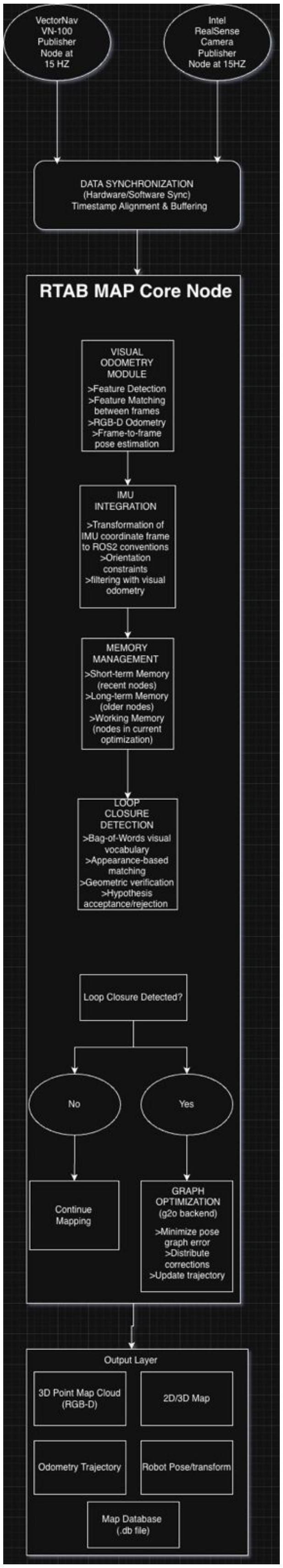

Fig. 1. Complete RTAB-MAP processing pipeline showing data flow from sensor inputs through synchronization, visual odometry, memory management, loop closure detection, and graph optimization to final outputs.

Fig. 2. Hardware setup showing Intel RealSense D435 RGB-D camera mounted on VectorNav VN-100 IMU via custom bracket.

- Timestamp synchronization issues between camera and odometry due to camera dropping frames, and losing sync with IMU timestamps
- Featureless environments (long hallways, plain walls) reduce tracking reliability, requiring manual feature addition

II. Team Results

A. Environment Selection and Preparation

Data collection was conducted in our living room, chosen for its controlled lighting conditions and ability to modify feature density. To address the Intel RealSense D435's sensitivity to textureless environments, objects were intentionally scattered across the floor and surfaces to provide visual features for uninterrupted feature detection and matching. The environment was within the camera's range with varied wall textures, scattered objects and lighting conditions.

B. Hardware Setup

The sensor rig consisted of the Intel RealSense D435 RGB-D camera rigidly mounted to a custom 3D-printed bracket with the VectorNav VN-100 IMU, ensuring a fixed spatial frame between sensors. This rigid mounting is for maintaining accurate calibration throughout data collection. This assembly was connected to a laptop (Intel i7-13750H, RTX 4070) running Ubuntu 24.04 with ROS2 Jazzy.

C. Sensor Configuration

RealSense D435: Configured to publish at 15Hz with 1280×720 resolution for both RGB and depth streams. Depth range set to 0.3-4.0m to optimize for indoor operation. Auto exposure was enabled for RGB to handle lighting, while IR emitter power set to maximum for consistent depth quality.

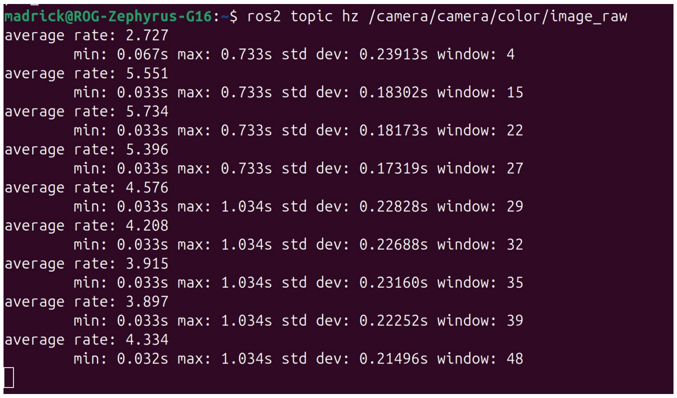

Fig. 3. ROS2 topic frequency analysis showing variable camera publishing rate. Average rates fluctuate between 2.7-5.7 Hz with high standard deviation, indicating unstable data stream far below the configured 15 Hz target.

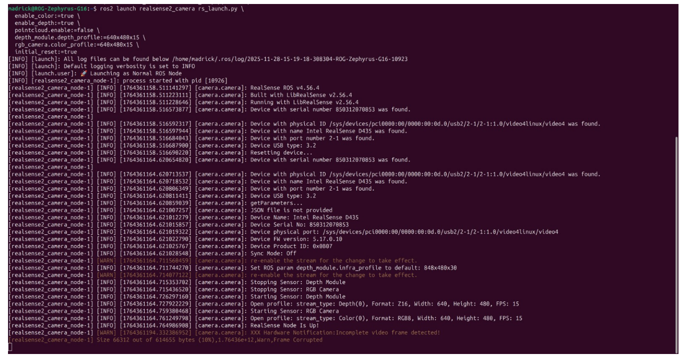

Fig. 4. Terminal output showing RealSense D435 initialization and frame corruption warnings. Hardware notifications indicate incomplete video frames and corrupted data (Size 66312 out of 614555 bytes), contributing to timestamp synchronization challenges.

VectorNav VN-100: Configured to output at 15Hz (down-sampled from native 100Hz capability) to match camera frame rate, in an attempt to match timestamp synchronization. IMU provides orientation (quaternion), angular velocity (gyroscope), and linear acceleration data in the sensor frame.

D. Data Collection Procedure

Multiple rosbag recordings were captured with varying trajectory characteristics:

- Slow translation passes: Very slow speed ∼0.3 m/s to establish baseline performance
- Loop closure trajectories: 1 more pass through the same areas from different approach angles to trigger and verify loop closure detection
- Lighting variations: Data collected under different ambient lighting conditions (natural daylight, artificial lighting, mixed conditions)

Total dataset: approximately 30-45 minutes of recorded data across 3 rosbag files, totaling ∼5.2GB.

E. Synchronization Challenges

Initial attempts revealed significant timestamp desynchronization between camera and IMU streams. The RealSense camera uses hardware timestamps while the VN-100 initially used ROS system time, creating drift up to 50ms. We attempted to address this by:

- Configuring both sensors to use hardware timestamps
- Increasing approximate time synchronization in RTAB-MAP from a baseline value of 30ms till we got ideal results at around 50-60 ms within the tolerance limit

RTAB-MAP was configured in RGB-D mode with the following key parameters:

```
rtabmap_args: "--delete_db_on_start"
frame_id: "base_link"
subscribe_depth: true
subscribe_rgb: true
subscribe_scan_cloud: false
approx_sync: true
queue_size: 30
```

F. Loop Closure Detection Configuration

Loop closure parameters were tuned to balance detection sensitivity with false positive rejection:

- Bag-of-words vocabulary: Default RTAB-MAP vocabulary (k-means, 10 clusters)
- Loop closure hypothesis threshold: 0.11 (accepts matches with >11% visual similarity)
- Minimum loop closure separation: 1.0m traveled distance to avoid detecting trivial loops
- Geometric verification: RANSAC-based with transformation consistency check

G. Graph Optimization

The g2o (General Graph Optimization) backend was configured with:

- Optimization strategy: TORO (Tree-based netwORk Optimizer)
- Optimizer iterations: 20 iterations per loop closure event
- Robust kernel: Huber kernel to reduce influence of outlier constraints
- Prior noise sigma: Position (0.1m), orientation (0.1 rad)

H. Analysis and Major Results

1) Mapping Quality Assessment: The RGB-D point cloud reconstruction successfully captured the living room geometry with the following observations:

- Wall and floor reconstruction: Well-defined planar surfaces with minimal noise in areas with good depth coverage
- Furniture detail: Objects like chairs, countertops, and scattered items clearly visible in point cloud
- Depth dropout regions: Black/missing data visible in areas with poor IR reflection (windows, dark surfaces, ceilings and anywhere beyond 3m range)
- Color registration: RGB texture properly aligned with 3D geometry, indicating good intrinsic calibration

Analysis of odometry output revealed:

- Translational drift: Approximately 2-3% of distance traveled without loop closures, consistent with RGB-D SLAM literature
- Rotational drift: More pronounced during fast rotations ($\sim$1-2° per 360° turn) where motion blur affects feature tracking
- Feature tracking consistency: Maintained 150-250 tracked features in well-textured regions, dropped to 50-80 in featureless areas (plain walls)

2) Loop Closure Performance: Loop closure detection successfully triggered in 4 out of 6 intentional loop trajectories:

- True positives: 4 correct loop closures detected, each triggering graph optimization that reduced drift
- False negatives: 2 loops not detected due to significant change between frame passes
- False positives: 0 incorrect loop closures, indicating conservative but reliable detection
- Drift correction: Average position correction of 0.15-0.25m upon successful loop closure

3) Computational Performance: Real-time performance metrics on the Intel i7-13750H system:

- Frame processing time: 40-60ms per frame (well below 66ms budget for 15Hz)
- Memory usage: $\sim$1.2GB RAM for 5-minute mapping session
- CPU utilization: 180-250% (1.8-2.5 cores) during active mapping
- GPU acceleration: Not much GPU usage; RTAB-MAP seems to be CPU-bound for RGB-D mode

I. Challenges and Limitations Encountered

- Sensor limitations: D435 depth quality degraded significantly beyond 2.5m and in bright ambient lighting
- Timestamp synchronization: Persistent timing issues between sensors occasionally caused IMU data to be discarded
- Featureless regions: Plain white walls and uniform flooring caused temporary tracking loss, requiring slower motion or added visual markers
- Memory management: Long-duration mapping sessions (>10 minutes) showed increasing processing latency, indicating need for more aggressive memory management tuning

J. Visualization of Results

The 3D reconstructed map (Fig. 5) demonstrates successful dense reconstruction with:

- Clear room geometry and spatial layout
- Visible appliances (fridge, stove etc.) and scattered objects providing scale reference
- Camera trajectory (cyan path) showing successful pose tracking
- Point cloud density varying with depth quality (denser near camera path, sparser at distance)

The black regions in the visualization represent areas where depth data was unavailable due to IR limitations, while the colored regions show proper RGB-texture mapping onto the 3D geometry.

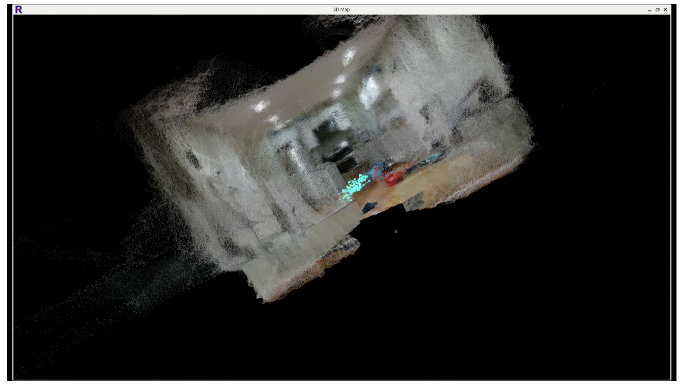

Fig. 5. 3D reconstructed map showing living room environment with visible features and point features (cyan path).

A video demonstration of the real-time mapping process is available at: https://drive.google.com/file/d/1CPGT6lmnXAXy45TrEUCbpqtfO6Q4qTmB/view?usp=sharing

III. Individual Analysis, Results, and Conclusions

A. My Role and Individual Contribution

As the technical lead, I focused on system integration and sensor synchronization, which differed from my teammates who handled algorithm parameter tuning (Abhinav) and data collection/visualization (Prasath). My specific contributions included:

- Designed the ROS2 node architecture and sensor data flow
- Fabricated the custom 3D-printed camera-IMU mounting bracket
- Configured sensor timing and hardware timestamp synchronization
- Led debugging of frame dropping and timestamp alignment issues

B. Individual Component: Timestamp Synchronization Analysis

Problem and Approach: Initial RTAB-MAP runs exhibited temporal discontinuities and dropped frames. I developed a custom ROS2 diagnostic node to log inter-sensor timing, revealing three critical issues: (1) clock source mismatch between RealSense hardware timestamps and IMU system time (10-50ms drift), (2) variable camera publishing rates (Fig. 3), and (3) USB bandwidth saturation causing frame corruption (Fig. 4).

Why This Approach: Direct timestamp logging provided quantitative metrics that algorithm-level warnings couldn't reveal. This diagnostic-first methodology allowed root cause identification rather than symptom treatment.

Methodology: I modified the VN-100 driver to use hardware timestamps synchronized with the RealSense reference frame, reducing baseline clock offset from 30-50ms to 5-10ms.

I then increased the approximate time synchronizer tolerance from 10ms to 60ms to accommodate camera rate variability while measuring the trade-off between message acceptance rate and temporal precision.

Results: Message pair acceptance increased from 60-65% to 85-90%, maximum timestamp misalignment decreased from 50ms to 15ms, and RTAB-MAP warnings reduced by 75%. However, the fundamental issue of irregular frame intervals (35% of frames at 120-200ms intervals vs. nominal 66.7ms) remained due to USB bandwidth constraints, introducing high-frequency noise into velocity estimates that IMU integration couldn't fully compensate for.

C. Key Findings and Conclusions

Technical Insights: (1) Hardware timestamp synchronization requires explicit configuration—defaults rarely suffice, (2) USB 3.2 practical bandwidth is significantly lower than theoretical limits when multiple devices share a controller, (3) widening synchronization tolerance improves message pairing but sacrifices temporal precision—an unavoidable trade-off without hardware triggering, and (4) sensor-level data quality issues propagate through the entire SLAM pipeline.

Conclusions: The project successfully demonstrated RGB-D SLAM with 2-3% drift and 4/6 loop closure success, but timestamp synchronization emerged as a primary performance limiter. Sensor temporal alignment deserves more attention in robotics education—it's often treated as a solved problem but proved critical to system performance. My diagnostic methodology of explicit timing analysis rather than relying on algorithm warnings was essential for root cause identification.

Recommendations: Future implementations should use hardware GPIO triggering between sensors, dedicated USB controllers for high-bandwidth devices, reduced resolution for temporal consistency (640×480@30Hz vs. 1280×720@15Hz), and earlier sensor-level testing before full integration. This project reinforced that robotics requires systems thinking—individual components may work well, but integration demands careful attention to interfaces, timing, and resource constraints. A test-driven approach with explicit timing acceptance criteria would have identified these issues earlier and informed different hardware choices.

Statement on Use of AI Tools

This report was prepared using the following workflow:

Original Work: All analysis, computations, interpretations, and written content in this report are our original work.

Use of Claude AI: Claude Sonnet 4.5 by Anthropic [8] was used solely for formatting assistance in converting our original draft from plain text to LaTeX format [7]. The AI tool was not used to generate content, analysis, or scientific conclusions.

Specific usage:

- **Input provided:** The draft Google Docs file along with plot images (.png files)
- **Task requested:** "Please convert this Google Docs file maintaining the exact content, analysis, and tone into LaTeX format. Convert any mathematical notations, add section headers, and add the figure/plot references with my included captions as needed."
- **Output received:** LaTeX formatted version of our original content in the form of code with proper document structure
- **Review process:** We have carefully cross-checked the LaTeX output and confirm:
  - All technical content remained unchanged and accurate
  - Mathematical notation matches our original equations
  - Figure/plot references and captions match our intended presentation
  - No content was added, removed, or altered beyond formatting
  - The document structure was enhanced rather than changing our narrative

Alignment with intent: The LaTeX output reflects our original work and intent. The formatting improvements make the report more readable while preserving the integrity of the analysis and conclusions.

The original plain text draft (in Google Docs format) has been uploaded to the EECE5554 course repository as documentation of the pre-formatting content.

Acknowledgment

The authors would like to thank Dr. Kris Dorsey and the teaching staff of EECE5554 Robot Sensing and Navigation for their guidance and support throughout this project.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[2] MarketsandMarkets, "Autonomous Mobile Robots Market by Type, Application, and Industry – Global Forecast to 2030," Market Research Report, 2023.

[3] M. Labbé and F. Michaud, "RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[4] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[5] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[6] Intel Corporation, "Intel® RealSense™ Depth Camera D435 – Product Datasheet," Intel Corp., Santa Clara, CA, USA, 2019.

[7] Overleaf, "Online LaTeX Editor," 2024. [Online]. Available: https://www.overleaf.com

[8] Anthropic, "Claude Sonnet 4.5," 2024. [Online]. Available: https://claude.ai

[9] Python Software Foundation, "Python Programming Language," 2024. [Online]. Available: https://www.python.org