An Exploration in Odometry Methods for RTAB-Map

Francesco Marrato
Department of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada
Email: 15fram@queensu.ca

***Abstract*—This paper explores how different forms of input odometry effect the quality of Simultaneous Localization and Mapping (SLAM) produced by Real-Time Appearance Based Mapping (RTAB-Map). An experiment was conducted using a Husky robotic platform and a PlayStation stereo camera controlled through the Robotic Operating System (ROS) in an indoor test field. Visual odometry, wheel odometry, and a hybrid visual-wheel odometry were compared quantitatively on their measurement of total distance traveled and their accuracy of measurements between objects in the environment. The experiment concluded that wheel odometry is the most accurate method when measuring total distance traveled while visual odometry has the highest accuracy when measuring between object in the environment. Qualitatively, hybrid odometry demonstrated frequent ground plane shifting and a high point variance in its generated point clouds. Additional investigations are planned using unstructured outdoor environments and a quadrupedal robotic platform.**

***Keywords*—** Odometry, SLAM, ROS, RTAB-Map

April 26, 2022

I. Introduction

It is vital for most mobile robotic platforms to be able to localize themselves within their environment. Interpretations of the robot's environment can be provided in many forms. Environments can consist of simple location tags positioned around a room or be as complicated as a full scale 3D model. Simultaneous Localization and Mapping (SLAM) is an established technique where a robot's interpretation of its environment is occurring along side its attempts to localize itself within that environment. Various forms of odometry have been used to provide SLAM algorithms with the measurements required to estimate their movements. Gathering odometry through wheel encoders has historically been a simple method, while interpreting odometry through visual input is a capability of more recent SLAM. The identification of the most robust odometry technique for SLAM would contribute to robots having a more accurate understanding of their surroundings.

Comparisons of odometry and SLAM techniques across different software packages have previously been undertaken in [1]. An analysis of the odometry path estimated by Real-Time Appearance Based Mapping (RTAB-Map) and ORB-SLAM2 was compared against ground truth Vicon tracking data. Accessible open source software has also reduced the time required by a single researcher to implement and analyze different approaches to components of SLAM [2].

The proposed research was to compare input odometry methods for the popular RTAB-Map SLAM ROS software package. An experiment was carried out using a modified PlayStation virtual reality camera mounted to a Husky Unmanned Ground Vehicle (UGV). The experiment was comprised of a repeatable motion path for the platform with unique objects positioned at known distances. The analysis quantitatively compared the total estimated distance traveled by the platform and the accuracy of the 3D map created by the SLAM software.

The contribution of this paper is to identify characteristics of each odometry method. By understanding the capabilities of each method, a more informed decision can be made for future applications of RTAB-Map. The results show that wheel odometry should be used if accurate travel distance is of greater concern, while visual odometry should be used if accurate environment measurement is of greater concern. The over-estimation of total distance travelled using visual odometry has also been replicated from [1].

The remainder of this paper begins with Section II which discusses related works, including existing software packages used in this research. Section III reviews the three different odometry methods and how they were implemented for this experiment. Section IV and V examine the experiments conducted and results obtained. Finally, Section VI shows conclusions drawn from the experiments and outlines future research.

II. Related Works

This section explored approaches to 3D mapping using SLAM. Existing software packages used in this research are also outlined with their importance and influences.

A. Real-Time Appearance-Based Mapping

RTAB-Map is a multi platform 3D SLAM software suit that is popular within the Robot Operating System (ROS) community. RTAB-Map was developed in [2]. The popularity of the software suit comes from its ease of use, frequent software updates, open source nature, and its ability to work with a variety of 3D sensors, across multiple platforms.

RTAB-Map recreates 3D representations of the surrounding environment (in the form of point clouds) using stereo

RGB/IR, RGB-D, or Lidar. RTAB-Map requires basic inputs in the form of time synchronized rectified images and odometry messages. RTAB-Map began as an appearance-based loop closure detection system with an emphasis on memory management for large-scale online operation. The software has grown to include SLAM and localization capabilities. The paper [2] outlines how RTAB-Map was developed to support a variety of robotic applications without laboring on identifying the single best method. The attitude of this project has allowed it to incorporate most popular keypoint detection algorithms such as ORB, SIFT, and SURF in which the preferred can be selected. The version of RTAB-Map covered in [2] was compared both quantitatively and qualitatively on popular real world datasets such as KITTI and TUM RGB-D (against their respective ground truth data).

B. ORB SLAM

ORB-SLAM2, the second rendition of ORB-SLAM, is a stereo/RGB-D implementation of the feature-based visual SLAM software package developed in [3]. ORB-SLAM2 was the first open-source SLAM system to cover the use of monocular, stereo, and RGB-D cameras that included loop closing, re-localization, and map reuse. This system implements the popular Oriented FAST and Rotated BRIEF (ORB) feature detection algorithm which has been open source since its creation, avoiding the potential patent infringement brought by using SIFT or SURF feature detection. ORB-SLAM2 achieved a higher accuracy when using RGB-D SLAM by using Bundle Adjustment (BA) over previous state-of-the-art methods based on photometric and depth error minimization. Using BA allowed a lightweight globally consistent sparse reconstruction to be constructed by CPU only computation. The goal of [3] was to achieve long-term globally consistent point clouds over densely populated reconstructions.

C. VSLAM Benchmarks

Previous analyses have been conducted to benchmark the capabilities of ORB-SLAM2 and RTAB-Map. In [1], researchers outlined an experiment where both VSLAM methods attempted to capture the same environment while following an identical pre-planned path. A motorized wheel chair was outfitted with an Intel RealSense camera and ROS control to act as the mobile platform for each run of the experiment. The benchmark included ground truth odometry by using a Vicon motion capture system which was then compared against the visual odometry of the VSLAM methods through a total distance traveled analysis. A qualitative analysis included the shortcomings of the visual odometry methods, highlighting that a loss of position within the map was a common issue. The conclusions made by [1] summarized that the sparse maps created by ORB-SLAM2 demonstrated near-zero drift odometry while RTAB-Map had a consistent overshoot of total distance traveled.

D. ROS Support

ROS is a set of system tools and libraries which aim to accelerate the development of robotic platforms and their application. ROS was introduced in 2007 in [4] as a method of interlinking varying robotic hardware and their respective software support. Researchers at Stanford University identified a key issue with robotics research as the depth of knowledge required (in both software and hardware aspects) to combine robotic components. The required breadth of expertise was well beyond the capabilities of a single researcher, requiring an understanding of driver-level software to higher level abstract reasoning. ROS has since seen widespread adoption in academia, being supported by popular robotics sales companies, including Clearpath Robotics [5]. The open source nature and inter-language capabilities of ROS has made the advancement of robotics significantly easier.

ROS has allowed for intermediate steps in larger robotics projects to be completed in hours instead of days. Providing documented tools to be downloaded and implemented quickly leaves less time spent on common processes. ROS provides access to an entire image processing pipeline for computer vision. Camera calibration, a necessary but trivial step in stereo imaging can now be completed using a device agnostic GUI through [6]. This package is community maintained with thorough tutorials, even providing the user with a printable calibration checkerboard. Stereo rectification, the preceding step to depth/disparity mapping can be handled by the stereo image processing package, which takes the calibration matrices of the camera and publishes rectified left/right images ready to be used in further steps of any computer vision processes.

III. Odometry Methods

A unique ROS launch file was developed for each odometry method. For each experiment, two launch files were used, one to bring up control of the Husky UGV and another to connect a specific odometry method with RTAB-Map.

A. Visual

Visual odometry is a feature that comes pre-packaged with RTAB-Map. To implement visual odometry with the UGV, a transform from its base to the cameras fixed position was established. This ensured the camera would be positioned at the correct height in RTAB-Map. The launch file also specified the use of visual odometry so that RTAB-Map would pre-compute odometry messages.

B. Wheel

Wheel odometry included an identical transform to what was used with visual odometry. The launch file specified the use of odometry messages generated from the wheel encoders of the UGV. It is important to note that the wheel odometry messages were made of pre-processed data. The UGV ROS software, filters the wheel odometry messages through an Extended Kalman Filter (EKF). The EKF takes partial pose inputs from the left and right wheel encoders, estimating the 3D pose of the robot.

C. Hybrid

Hybrid odometry takes the setup of the previous two methods one step further. A visual odometry node is spawned separately from the initialization of RTAB-Map. The visual odometry messages are then combined with the filtered wheel odometry messages using an Unscented Kalmann Filter (UKF). The UKF combines the wheel odometry translation in X and Y with the visual odometry translation in Z and orientation in roll, pitch, and yaw. The estimated 3D position is then provided as the input odometry topic used by RTAB-Map.

A UKF was chosen over an EKF to better handle the UGV's nonlinearities. In this application, the additional computational burder of the UKF was deemed negligible. There are only a maximum of 12 inputs to the UKF at any time, which is well within the computing capacity of the system. The UKF was implemented using the UKF localization node which comes as part of the robot localization ROS package in [7].

IV. Experimentation

A. Hardware

All development and experimentation was performed using a Husky UGV supplied by Ingenuity Labs and purchased through Clearpath Robotics. The Husky UGV is a medium sized robotic development platform with four fixed wheels. The platform has no suspension or steering, changing orientation through tank style control. The platform was fitted with wheel encoders which were used to acquire wheel odometry. The platform gains most of its traction from being heavy, weighing 50kg, and occupying a space less than a meter cubed.

A modified Play Station Virtual Reality (PSVR) camera was the single source of video input fed into RTAB-Map to create the 3D point clouds. Hardware modifications included a USB dongle to convert the proprietary connector to a standard USB 3.0 and a 3D printed mount for connecting the camera to the robotic platform. The PSVR camera consists of two RGB sensors placed at a constant/known distance from one another. Combined, the sensors capture an image of 3200x840 pixels which is then split into left and right frames. Hardware triggering for the image sensors is handled by the cameras firmware. An overview of the hardware system is shown in Fig. 1.

B. Software

All development and experimentation was conducted using Ubuntu 18.04 and ROS Melodic. ROS Melodic was selected as it was the newest supported version of ROS for the Husky UGV at the time. Most packages used were from the standard ROS Melodic build with a few exceptions. Husky-ROS communication was handled through software distributed by Ingenuity Labs and PSVR to ROS communication was handled through a library previous developed as a pet project [8]. Other ROS software packages include the stereo image processing package for stereo image rectification, the camera calibration package (self explanatory) [6], the robot localization package [7] for combining odometry data through EKFs and UKFs, and finally RTAB-Map which combines the rectified images with the input odometry to create 3D point clouds.

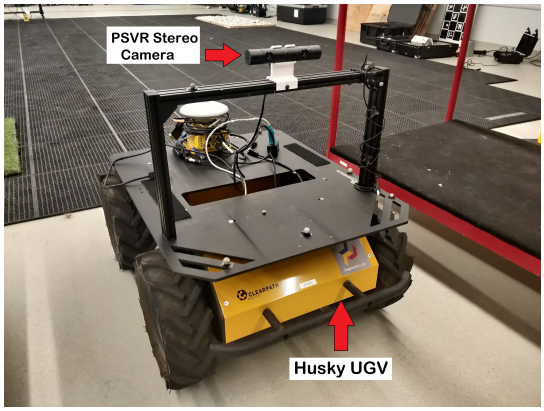

Fig. 1. Husky UGV outfitted with a modified PSVR stereo camera

C. Summary of Experiment

An experiment was organized to compare the three methods for input odometry. All experiments were conducted indoors at Ingenuity Labs (Queen's University). The experiments took place on a square test field 5.85 meters in size. The size of the grid square was dictated by the standard size of a single tile square (0.450 meters) and the maximum number of squares (13) that could be occupied by this experiment. The experiment area was littered with complex and non-duplicate objects to give the visual odometry the best possible chance for keypoint detection. Other considerations for the visual odometry included evening out the lighting by closing window shades, shutting off lights in adjacent rooms, removing heavily reflective objects, and ensuring the experiment would not be interrupted by individuals walking through the scene.

Each corner of the square test field included a sign post and a large unique road sign. The road signs were positioned at the edges of the square so accurate measurements between the signs could be established. In the center of the field, a small ramp was positioned. The center ramp would be used to evaluate the odometry method's ability to handle changes in elevation and orientation. The ramp would be traversed twice, once along its intended direction and another across its edge, requiring the vehicle to climb over the ramp as if it were a small obstacle.

The vehicle would begin centered between the two corners with its wheels facing towards the center ramp. The consistency of the starting position was regulated by having the vehicles wheels positioned against a set of starting blocks. The vehicle would begin each run by driving forward to a second set of wheel blocks where the body of the vehicle would be tilted in the roll and pitch directions. These orientation changes were to allow the visual odometry to zero itself and identify the initial horizon line. Orientation changes were performed across all runs to keep consistency.

The experiment consisted of three runs for each method of odometry. For consistency, all runs were conducted on the

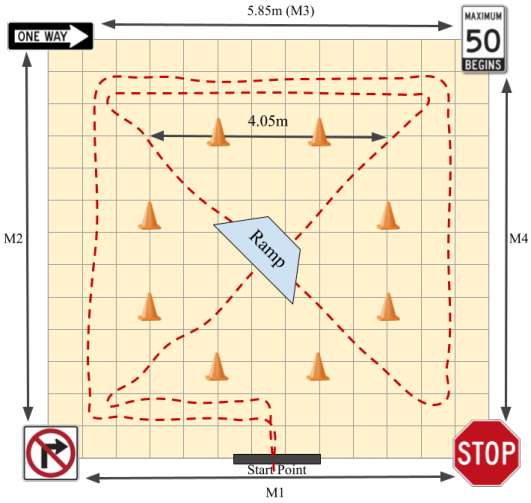

Fig. 2. Outline of the test field, including measurement labels (M#), path traveled (red), and unique obstacles.

same day with no chance for changes to the test field. The vehicle's batteries were swapped for a fully charged battery pack at the beginning of each odometry method. The vehicle followed the same path for each run, driven through tele-operation using a joystick controller. Driving characteristics were kept consistent by only driving in straight lines and turning in place. A full outline of the test field and the path travelled by the vehicle is included in Fig. 2 with a video summarising a single test run viewable here.

V. Results and Analysis

A. Data Pre-Processing

Data pre-processing was a crucial step before the data could be analyzed. All data processing was performed with the help of CloudCompare, an open source and trusted tool for working with point cloud data. The data pre-processing steps are outlined in Fig. 3 and were carried out as a five step process. The raw data was first cross sectioned to remove points beyond the bounds of the test field, including excess points above the tops of the traffic signs and points below the floor. Step one created a subsection containing only the area associated with the test field. Next, a Statistical Outlier Removal (SOR) filter was applied with a mean distance estimation consisting of ten points. The SOR filter works to clean the point cloud, removing outliers and excess points. Step three consisted of placing each point along a scalar field based off its red value in the RGB colour space. The red color channel was selected since the orange buckets used as footings for the traffic signs could easily be isolated. The points were then filtered based on their scalar value (Red value) to exclude any point with a value less than 180 (chosen from inspecting individual points belonging to the orange buckets). With the floor of the test field removed and the orange buckets isolated, the point cloud could be split into four quadrants, isolating each individual sign. With each sign isolated, measurements could be established between combinations of two of the four point clouds.

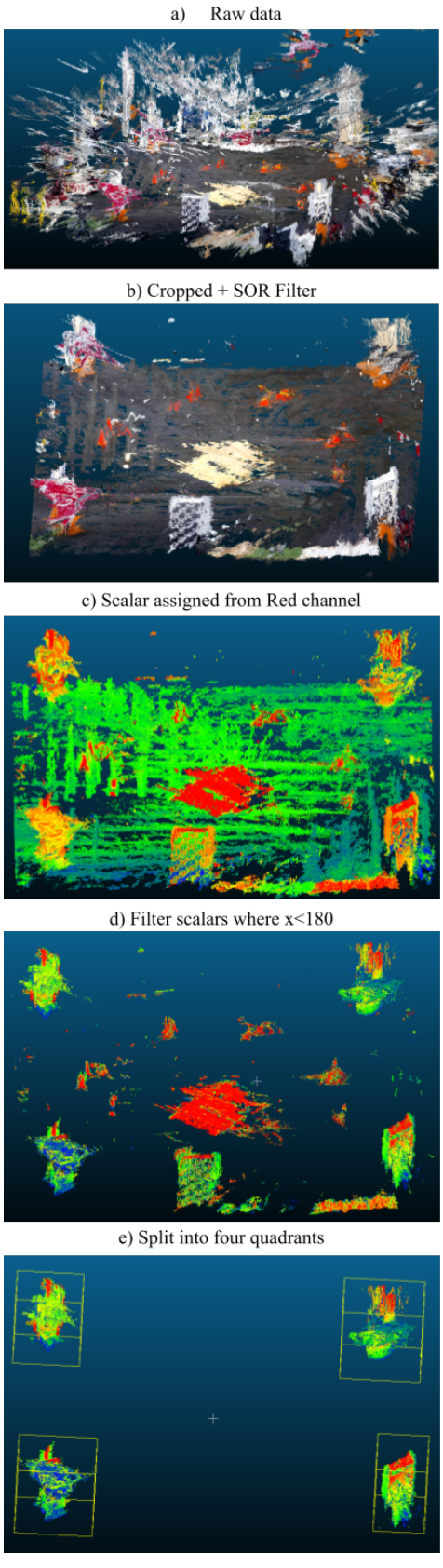

Fig. 3. Five step data pre-processing steps.

B. Evaluation of Total Distance Measured by Odometry

As outlined in Section IV-C, each odometry method underwent three runs on an identical test field. RTAB-Map has the functionality to export both the point cloud and the estimated path taken by the robotics platform. By knowing the layout of the test field and the total estimated distance of the cameras odometry from RTAB-Map, an accuracy analysis was performed. This analysis highlighted if each odometry method was within the correct range to be considered accurate. Unlike [1], the experiment did not include the use of a Vicon motion tracking system to gather ground truth odometry (see Section VI-A for an extension on this). This limitation was primarily influenced by the technical knowledge and time requirement needed to gather accurate data from the Vicon system in Ingenuity Labs. An alternative was devised which used a maximum and minimum range for total distance traveled by the robotic platform. The range was established by measuring the interior and exterior walls of the test field and extrapolating the maximum and minimum distance required to complete a test run. It was established that the robotic platform should travel no more than a distance of 45.8 meters and no less than a distance of 31.7 meters (a range of 14.1 meters). The robotic platform would move within a "doughnut" established by the exterior bounds of the test field and the interior bounds marked with orange cones. This experiment was best summarized through Fig. 2. The odometry data was exported from RTAB-Map in the KITTI txt format, storing each camera pose as a list of 12 float values. The data was processed using EVO, a python package for evaluating odometry and SLAM data.

The odometry distance measured by each run are shown in Table I. Of the three odometry methods, visual odometry was the only method to fall outside of the established bounds of the test field. Calculated from the average, visual odometry over shot the maximum travel distance by 28.6 meters and the minimum travel distance by a staggering 42.7 meters. The occurrence of overshoot in visual odometry was also noted in [1] where the results demonstrated a total distance measure of almost double the ground truth distance. Both wheel and hybrid odometry measured a total distance traveled that fell within the established range (33.5 and 35.1 meters respectively). It was expected that wheel odometry and hybrid odometry would have similar results. The combination of odometry techniques was primarily to observe whether the 2D odometry gathered from the platforms wheel encoders could be extended to 3D by introducing visual odometry. The 2D to 3D extension would be most evident when the robotic platform traveled over the ramp, having a minimal impact on the results of this specific experiment.

C. Evaluation of Measurement Accuracy Between Known Test Field Positions

The test field in Fig. 2 contains four traffic signs positioned at a distance of 5.85 meters from one another (excluding diagonal measurements). Using the physical measurement of the test field as a ground truth, an analysis was undertaken where the ground truth was compared against measurements taken from each odometries 3D point cloud. Themes could then be extracted relating to each methods accuracy, standard deviation, and trends.

TABLE I
TOTAL DISTANCE TRAVELLED ESTIMATED BY EACH ODOMETRY METHOD (METERS)

	Visual Odometry	Wheel Odometry	Hybrid Odometry
Run 1	67.8	34.0	36.7
Run 2	87.9	34.4	33.5
Run 3	67.5	32.1	35.0
Average	74.4	33.5	35.1
In Range	FALSE	TRUE	TRUE

With the point cloud data processed (outlined in Section V-A), minimum, maximum, and mean distances were calculated for each of the four measurements shown in Fig. 2. The ground truth distance of 5.85 meters was measured no less than 24 times per odometry method. Cloud-to-cloud measurements were performed by CloudCompare. The results are summarized in Table II and Table III. All three odometry methods underestimated the distance between the traffic signs, with visual odometry being the closest with an average measurement of 5.37 meters. Wheel odometry was second in accuracy with an average measurement of 5.03 meters followed by hybrid odometry with an average of 4.66 meters. Wheel and hybrid odometry both demonstrated a failure case where the point cloud was unable to be pre-processed properly because it was too dirty.

The standard deviation for each odometry method was calculated (Table III) with the normals plotted in Fig. 4. Fig. 4 includes a marker for the ground truth measurement, helping to visualize which odometry method has a standard deviation which overlaps with the ground truth measurement. All three odometry methods overlap the ground truth measurement in varying degrees. Visual odometry had the greatest overlap with the ground truth falling into the second standard deviation. The ground truth measurement was within the third standard deviation of both wheel and hybrid odometry methods. Wheel odometry had the smallest variance at 0.253, followed by visual and hybrid odometry at 0.299, and 0.494 respectively. The large variance of hybrid odometry is in alignment with expectations. Assuming the measurement noise of the visual and wheel odometry are both white (zero mean and constant variance), then the combination of the two in the hybrid odometry method should be the sum of the two's individual standard deviation. The sum of standard deviations of the visual and wheel odometry is 0.552, while the standard deviation of the hybrid odometry is 0.495.

The larger standard deviation of the hybrid odometry method is also observed in the point clouds. Fig. 5 a) (hybrid odometry point cloud) shows traffic signs with a higher variance in their points compared to Fig. 5 b) (wheel odometry point clouds). Hybrid odometry achieved the shortest minimum distance across all measurements while visual odometry achieved the longest maximum distance across all measurements. This observation is highlighted in Table II.

TABLE II

DISTANCES MEASURED BETWEEN ROAD SIGNS (M1-M4). ALL MEASUREMENTS IN METERS. HIGHEST MAX (ORANGE) AND LOWEST MIN (BLUE).

Cloud to Cloud Road Sign Measurements												
	M1			M2			M3			M4		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Visual 1	4.59	5.77	5.24	4.35	5.68	5.14	4.23	5.68	4.98	4.74	5.58	5.17
Visual 2	4.62	5.85	5.38	4.70	6.11	5.28	4.17	5.87	4.99	4.97	6.23	5.34
Visual 3	4.79	6.26	5.70	5.17	6.03	5.65	4.61	5.65	5.65	5.08	6.61	6.00
Wheel 1	FAILURE CASE											
Wheel 2	4.59	5.65	5.26	4.81	5.82	5.46	4.43	5.44	4.94	4.26	5.58	4.85
Wheel 3	4.13	5.18	4.65	4.72	5.55	5.28	4.41	5.43	4.91	4.24	5.10	4.91
Hybrid 1	3.55	5.14	4.36	4.90	5.93	5.30	4.52	5.36	5.00	4.11	5.84	5.17
Hybrid 2	4.56	5.40	4.93	3.61	4.51	4.24	3.82	5.25	4.56	3.23	4.23	3.75
Hybrid 3	FAILURE CASE											

TABLE III

MEAN AND STANDARD DEVIATION OF EACH ODOMETRY METHOD. ALL MEASUREMENTS IN METERS.

	Visual	Wheel	Hybrid
Mean	5.377	5.033	4.664
STD	0.299	0.253	0.494

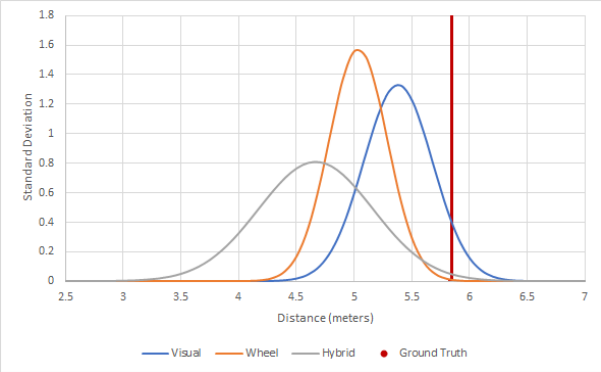

Fig. 4. Standard deviation of each odometry method with ground truth reference.

Qualitatively, it was noted that the traffic signs in the hybrid odometry point clouds had a denser population of points closer to the minimum measurement.

D. Qualitative Analysis of Raw Point Clouds

The visual quality of the point clouds was evaluated to identify repeated characteristics that could be associated with specific odometry methods. For wheel and hybrid odometry methods, a point cloud characteristic was identified. The raw point clouds were evaluated to emphasize the presence of any defects. Visual odometry demonstrated no discernible characteristics, becoming a reliable reference when identifying qualities of the other odometry methods.

Ground plane shift was observed as a common issue with the hybrid odometry method. An example of ground plane shift is provided in Fig. 6. Portions of the point cloud would be duplicated at an alternate axis to the main ground plane. Ground plane shift can cause major sections of the point cloud to become buried in garbage data. The presumed cause of ground plane shift is RTAB-Map falsely identifying new key

a) Hybrid Odometry

b) Wheel Odometry

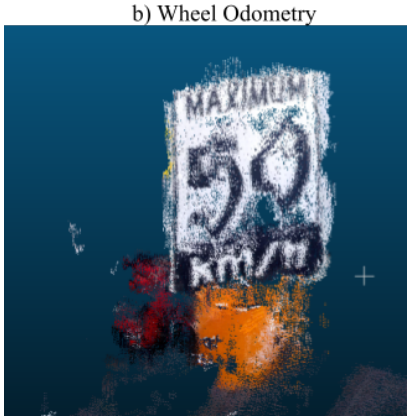

Fig. 5. Point cloud variance of hybrid and wheel odometry.

points when the robotic vehicle changes its orientation quickly. Ground plane shift is most evident on either sides of the central ramp in the test field, an area where the vehicle would undergo jerking in the pitch axis.

The wheel odometry method demonstrated the tightest point groupings and object readability. Fig. 5 shows that objects in the test field appear objectively less obscured while using the wheel odometry method. The characteristics of the test field gave an advantage to the wheel odometry method as it is primarily a flat plane. With no ability to measure role, pitch, and yaw, the quality of the wheel odometry point clouds would

Fig. 6. Example of ground plane shift (right side of image) using hybrid odometry.

suffer greatly if uneven terrain or rolling hills were introduced.

VI. Conclusion

This research paper explored the use of three methods of collecting vehicle odometry to be used by RTAB-Map. The goal was to isolate characteristics of each odometry method to help identify which could provide the cleanest point cloud and most accurate odometry measurements. An experiment was carried out to measure accuracy of the odometry path and the accuracy of the point cloud map created by RTAB-Map. From this experiment, the following conclusions were made. Quantitatively, visual odometry produced the most accurate measurements, while wheel odometry produced measurements with higher precision. The hybrid odometry method demonstrated repeatable defects in the form of ground plane shifting, although the defects did not have an effect on the measurement results because of a thorough data pre-processing step. This research validated the over estimation of total distance traveled when measured through visual odometry that was originally identified in [1]. The hybrid combination of visual and wheel odometry through a UKF did not demonstrate a direct improvement over either of the individual methods. Overall, this research concludes that (for an indoor setting with flat terrain and unique environment objects) wheel odometry should be used if accurate travel distance is of greater concern, while visual odometry should be used if accurate environment measurement is of greater concern.

A. Future Works

This research project was an extensive learning opportunity for SLAM odometry techniques and the application of real robotic vehicles. The experiments were carried out to the best ability of the researcher within the time allotted. From the experiments that were carried out, there are several questions that should be researched further. A reevaluation of the presented experiment should take place which includes ground truth data collected from a Vicon motion tracking system. The odometry data should be time stamped (on top of being pose stamped) so the odometry paths can be evaluated quantitatively. Pose stamped data created odometry readings of varying size. Odometry data across multiple runs couldn't be aligned with one another, leaving only total distance to be analyzed. Performing the experiment in an outdoor test field with varying elevation and rough terrain would help to confirm or deny the results presented by this research.

Furthermore, the information gained from this research project will be directly applied to research planned for future thesis work. The odometry and processing techniques will be re-applied in the following months to a new quadrupedal robotic platform. The goal of future research will be to use the point cloud maps in combination with a semantic segmentation deep learning network to segregate between traversable and non-traversable terrain. Pre-filtering and cleaning of the point cloud data will be the focus of the near future.

Acknowledgment

The researcher would like to acknowledge the contributions of Ingenuity Labs for providing robotic hardware and lab access. Many thanks to Professor Joshua Marshall who supported this research through his course "Autonomous Vehicle Control 'I&' Navigation". Further thanks to Johann von Tiesenhausen and Heshan Fernando who acted as the initial knowledge bass and supervised the developments in the project.

References

[1] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J.-Y. Ertaud, "Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map," in *2019 Eighth International Conference on Emerging Security Technologies (EST)*, 2019, pp. 1–6.

[2] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LabbÉ and michaud," *Journal of Field Robotics*, vol. 36, 10 2018.

[3] R. Mur-Artal and J. D. Tardos, "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. [Online]. Available: https://doi.org/10.1109%2Ftro.2017.2705103

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[5] T. B. Mike Purvis, "Husky," https://github.com/husky/husky/tree/melodic-devel, 2013.

[6] S. M. Vincent Rabaud, Kentaro Wada, "Ros image pipeline," https://github.com/ros-perception/image_pipeline/tree/melodic, 2013.

[7] I. Charles River Analytics, "Ros robot localization," https://github.com/cra-ros-pkg/robot_localization/tree/melodic-devel, 2013.

[8] F. Marrato, "Ps4 stereo ros slam," https://github.com/TankyFranky/PS4_Stereo_ROS_SLAM, 2022.