Monocular Visual-Inertial Odometry using Incremental Smoothing and Mapping (ISAM2)

1st Atharva Anil Patwe
University of Girona
Girona, Spain
patweatharva@gmail.com

2nd Loc Thanh Pham
University of Girona
Girona, Spain
u1992429@campus.udg.edu

***Abstract*—Visual perception is crucial for robotics applications, requiring a robust system for localization using multiple sensers for accurate state estimation. Visual-Inertial Odometry (VIO), combining a camera and Inertial Measurement Units (IMUs), offers an efficient and accurate alternative to GPS and lidar for state estimation. Cameras provide rich visual data but suffer from limitations like low output rate and prone to motion blur, while IMUs offer high-frequency measurements but accumulate drift over time. VIO leverages the complementary strengths of both sensors.This paper discusses a VIO framework implemented on a Turtlebot platform using ROS and GTSAM for incremental smoothing and mapping. The approach involves a tightly coupled VIO pipeline, integrating camera and IMU data through Smart Projection Pose Factors for efficient state estimation. The methodology includes an image processing pipeline for feature detection and matching, and a graph optimization process for pose estimation.Initial testing in a simulation environment demonstrated the effectiveness of the VIO implementation, with future improvements suggested for buffer management and sensor calibration.**

***Index Terms*—Monocular Visual Inertial Odometry, ISAM2, GTSAM, Turtlebot, ROS**

I. Introduction

Visual Perception is one of the most crucial components of a robotics application. A robust system for localization reliant on multiple sensing modalities is required to accomplish accurate state estimation for robotics - a framework that integrates information from various visual and/or inertial sensors with differences in reliability (noise, covariance) under various conditions. For real-world scenarios, this must be done efficiently with continuous online updates. Visual Odometry or Visual SLAM tries to address this problem by combining one or more visual sensors or Cameras. Monocular Visual-Inertial, odometry (VIO) however, is the process of estimating the state (pose and velocity) of an agent by using only the input of one camera plus one or more Inertial Measurement Units (IMUs) attached to it. VIO is the only viable alternative to GPS and lidar-based odometry to achieve accurate state estimation. Since both cameras and IMUs are very cheap, these sensor types are ubiquitous in all today's robots.

In principle, camera is a device that accumulates photons during the exposure time of the sensor to get a flat 2D image of the environment. Therfore, Naturally they are very precise in slow motion cases and provide rich information which can be then utilized in various perception tasks. However,

Fig. 1: Visual Inertial Sensor adapted from [1]

cameras have a limited output rate (usually in the range of ∼30Hz to 100Hz). Additionally, they face challenges with scale ambiguity in a monocular setup and lack robustness in scenes with low texture, high-speed motions (due to motion blur), and High Dynamic Range (HDR) conditions, which can result in overexposure or underexposure of the image.

Inertial Measurement Units however, are proprioceptive sensor measuring the angular velocity and the external acceleration acting upon it. IMU sensors are independent of scene which make them unaffected by the aforementioned challenges faced by visual sensors. IMUs have a very high sampling frequency usually in the order of ∼300 to 1000Hz moreover they do not suffer from scale ambiguity. However, they have poor signal-noise ratio at low accelerations and low angular velocities. Additionally, due to the presence of sensor biases, the motion estimated from an IMU alone tends to accumulate drift quickly. The advantages and disadvantages of both the sensors when combined compliment each other and thus can provide accurate and robust state estimation in different situations.

In a typical Visual Inertial Odometry (VIO) setup, as shown in Figure 2, the camera(s) and IMU(s) are securely connected together. These sensors provide visual and inertial data at different rates. In VIO, the environment is represented by a set of 3D points (Landmarks) Wp that the camera projects into 2D image coordinates (Features/Keypoints) u:

$$u = \textbf{project}(T_{CW} \cdot Wp). \tag{1}$$

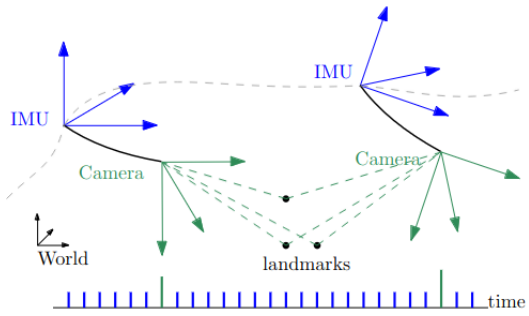


Fig. 2: Right: Camera frame in (green) and IMU frame in (blue) taking measurements along a trajectory (Adapted from [14])

The IMU captures angular velocity ω and external acceleration a:

$$\omega = I\omega + b_g + n_g, \quad a = R_{IW}(Wa - Wg) + b_a + n_a, \quad (2)$$

where $I\omega$ is the angular velocity of the IMU in the IMU frame, Wa is the acceleration of the IMU in the world frame, and Wg is the gravity in the world frame. The terms b and n represent biases and noise, respectively (See [2]). It's also important to note that for low-cost IMUs, this model can be overly simplified, and additional errors from scale factors and axis misalignment may also need to be considered as proposed in [3].

Essentially, VIO solves for the Pose of the sensor relative to world frame using camera and IMU measurements given by equations 1 and 2.

$$\mathbf{X}_i = \left[T^i_{WI},\ \mathbf{v}^i_{WI},\ \mathbf{b}^i_a,\ \mathbf{b}^i_g\right], \quad i = 1, 2, 3, \ldots, N$$

In this context, T^i_{WI} represents the 6-DoF (Degrees of Freedom) pose of the IMU, which includes its position and orientation. $\mathbf{v}^i_{WI}$ is the velocity of the IMU. The terms $\mathbf{b}^i_a$ and $\mathbf{b}^i_g$ are the biases of the accelerometer and gyroscope, respectively.

Unlike visual-only odometry, which only estimates the pose, VIO also requires the estimation of velocity and biases to make use of the IMU data. The biases are crucial because they allow us to calculate the true angular velocity and acceleration from the raw sensor data. Additionally, knowing the velocity is necessary for integrating the acceleration to determine the position.

This work is structured as follows: section II elaborates on state of the art techniques developed to solve VIO problem. This section provides understanding of different approaches for state estimation problem in VIO. Section III explains the methodology developed for implementation of VIO on a turtlebot platform using ROS. Furthermore, the results obtained during the implementation of this work are reported in section IV. Finally, this paper concludes with the section VI by explaining future direction for further development.

II. Related Works

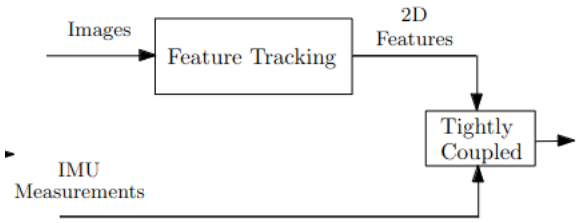


Fig. 3: Tightly Coupled VIO Pipeline

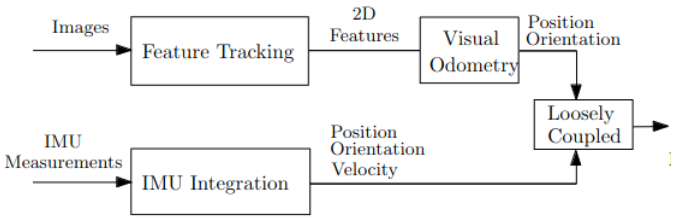


Fig. 4: Loosly Coupled VIO Pipeline

VIO can be implemented in different ways depending upon the way information is fused as well the number of camera poses involved in the estimation process. There are two way in which the visual and inertial information can be fused: *Tightly Coupled and Loosely coupled* (Adapted from [4]). The Loosely coupled VIO, processes visual and inertial motions separately and then fuse the resulting motion to get final output as shown in figure 4. On the other hand, tightly coupled VIO approach computes final output by directly combining the visual and inertial component which is depicted in figure 3. Usually, tightly coupled approach provide more accurate results because of their ability to predict and track the 2D features and correct the drift. At the same time, loosely coupled do not handle information fusion thus making them prone to drifting with vision only estimator.

State of the art VIO techniques can be categorized in three major types: *Filtering, Fixed Lag Smoothing and Full Smoothing*

A. Filtering

In the approach the inference process is restricted to the latest state of the system thus making it computationally efficient. The computational complexity of the Extended Kalman Filter grows quadratically with the number of estimated landmakrs thus only a small number of robust landmarks are tracked to use in real-time operations (Refer to [5]). To avoid limit of tracked landmarks, an alternative approach is to adopt a structureless model in which the features are marginalised out of the state vector as proposed in [6].

The drawback of this approach is that the filter can not utilize all of the visual information available until all of the

measurement are obtained (Adapted from [6]). Additionally, there are lionization errors making the filter inconsistent (See [7]).

B. Fixed-lag Smoothing

In this approach, the states falling within a specific time window are estimated and while rest are marginalized out ([8] [9] [10]). These approaches are generally more accurate than filtering techniques. This is because, Fixed-lag smoothing relinearizes part of past measurements as the estimation is updated. Moreover, this approach is more robust to outliers by using explicit outlier rejection methods. However, since fixed-lag smoothers marginalize the past they still suffer from inconsistency and linearization errors as shown in [11]. Fixed-lag smoothers are computationally more expensive than filtering approach since all the states within the time window are estimated.

C. Full Smoothing

This is an extension of the Fixed-lag Smoothing where instead of a confined time window, all the history of the states is solved using nonlinear optimization process [12] [13]. This approach provides most accurate results as compared to afore-mentioned methods as it can update the linearization point of the complete history as the state estimation evolves. VIO is a highly non linear problem to solve and thus keeping all the history makes this approach complex and computationally expensive. The optimization problem is approximately cubic with respect to the dimension of the states, real-time operation quickly becomes infeasible as the trajectory and the map grow over time as explained in [14]. The common approach to tackle this problem is to either keep selected keyframes or optimizing the graph in parallel architecture.

There has been a breakthrough development of Incremental Smoothing and Mapping proposed in first iSAM in [15] and later revised iSAM2 in [16]. Filtering approach can deal with multiple asynchronous sensors with different sampling frequency. This is because of the modular nature in which usually the IMU having higher frequency is used in Prediction model and Camera sensor with lower frequency is used in measurement model. However, in Smoothing methods the measurements from the sensor with higher frequency has to be combined together, termed as preintegration, in between two measurements from the sensor with low frequency. In VIO, IMU measurements are preintegrated in between two camera frames are received and the optimization process runs at the frequency of camera frames.

III. Methodology

SLAM or in our case, VIO can be posed as a Pose graph optimization problem containing Variables(or Keyframes) and Factors. Variables are the sequence of states in the system that need to be estimated, such as robot poses or landmark positions. Factors encode the probabilistic relationships or constraints between these variables (See Fig. 5), derived from sensor measurements or other observations. Observations in this model are the grounded realities that have a prior or a unary factor that make them rigid constraints e.g. Prior Pose factor of the robot.

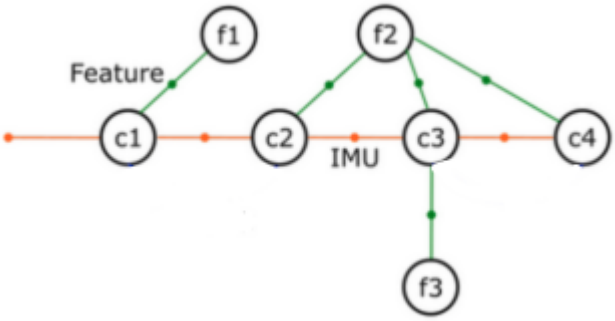


Fig. 5: Factor Graph for VIO Problem: The camera variables (c1, c2, c3, c4) represent the camera at a sequence of time points. A prior is placed on c1 to indicate that the pose of c1 is a known constant (the robot starts at c1). The feature variables (f1, f2, f3) represent the corners/distinct features found in the camera image by the image processing library. The factors represent the relationship between the variables.

In this work we are solving the problem of VIO with incremental Smoothing and Mapping with Smart Projection Pose Factors from Georgia Tech Smoothing and Mapping (GTSAM) library. As explained in the earlier section, iSAM2 provides an efficient back end for full-smoothing online factor graph optimization.

Rather than storing the graph in factor graph form, iSAM2 stores the graph as a Bayes Tree. The process by which a factor graph can be converted into a bayes tree is shown in Figure 6 as proposed in [16].This process, known as graph elimination, is similar to the factorization of a matrix. The matrix representation of the Bayes tree, called the square root information matrix, is much sparser than the matrix form of the factor graph. This sparsity allows for more efficient computation. Additionally, as shown in Figure 6b, updating the Bayes tree only requires updating some nodes, resulting in a sparser square root information matrix that needs less memory and computation. Finally, variable reordering can be done during the update, helping the tree maintain an optimal structure with smaller cliques, which leads to more efficient computations (Adapted from [16]) .

Smart Projection Pose Factors (SPPFs) are efficient way of managing landmark factors in a nonlinear factor graph. When optimizing the graph this type of factor offers more advantages than traditional *generic Projection* factors. SPPFs handles the case of missing measurements as well as the case of degeneracy while triangulating. Moreover, SPPFs do not optimize for the landmark state making it an ideal candidate of the VIO application and also making it a computationally efficient.

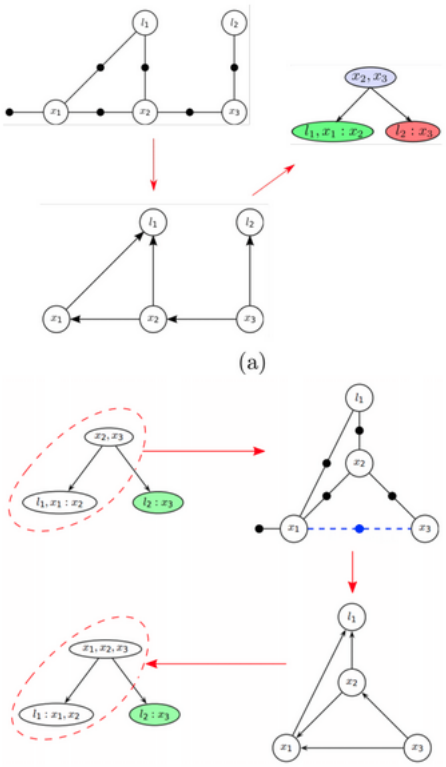


Fig. 6: (a) Factor graph is eliminated into a chordal Bayes Net, then cliques are collapsed into leaves to form a Bayes tree. (b) When a factor is added, only the nodes leading up to the root of the Bayes tree must be updated [16]

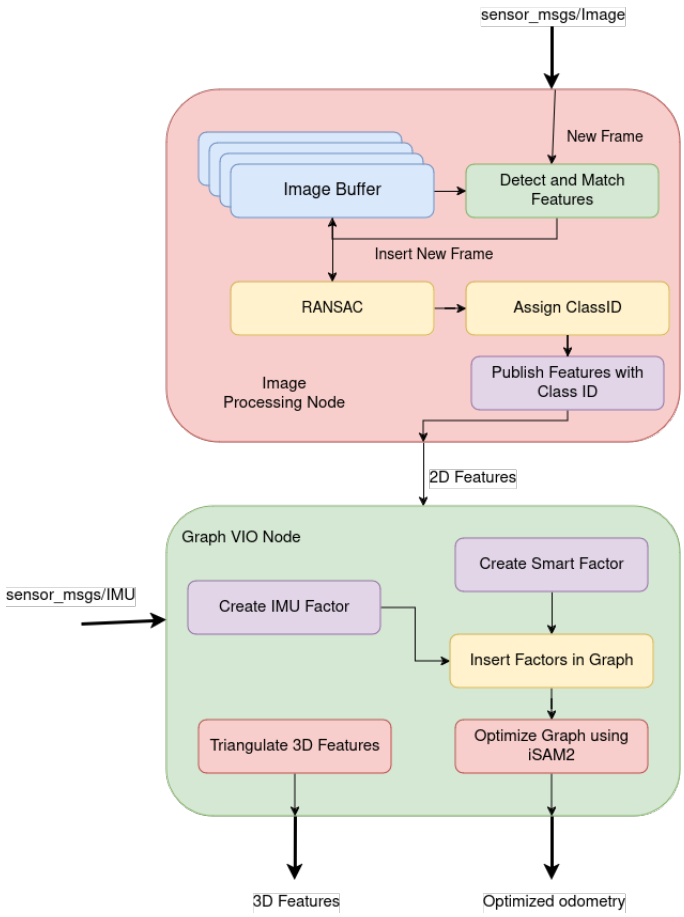


Fig. 7: Turtlebot VIO Package Architecture

A. Implementation

For implementation of the VIO using GTSAM, a ROS package was developed containing two ROS nodes: *Image Processing Node and Graph VIO Handler Node*. The package was created using the stonefish simulator on a turtlebot2 platform. Since the turtlebot is equipped with wheel encoders, an Extended Kalman Filter based odometry node based on wheel encoders was also created. When IMU is absent on the robot, the wheel odometry factor can be added instead of IMU preintegration. The overall architecture of the turtlebot VIO package is depicted in figure 7.

1) Image Processing Node: The landmarks shown in the figure 5 are features observed in the environment from the camera. For adding the SPPFs in the graph it is required to identify the features uniquely. Thus it is important to detect robust, distinctive and unique features. To perform this task an Image processing pipeline was developed in the Image Processing Node.

Since the scope of this work only deals with odometry and not full SLAM, an image buffer of customizable length was developed to store previous frames. This buffer holds a custom data type containing the frame ID, features detected with their assigned class IDs, their descriptors and the matching map to all the previous buffer frames.

Algorithm 1 Image Callback Pipeline Pseudo code

Start (Receive `sensor_msgs/Image` message)
Convert `sensor_msgs/Image` to OpenCV type
Convert to Gray scale
Remove Noise (Applying Gaussian blur)
Detect features and Compute descriptors
if first image **then**
    Add it to the image buffer
else
    for all frames in the buffer **do**
        Match features with the frames
        if matches not found **then**
            continue
        else
            Apply Lowe's ratio test to get good matches
        end if
    end for
    Insert new frame with good matches into the buffer
end if
Apply RANSAC
update Keypoint ClassIDs
publish2D Features

An image processing pipeline was developed as shown in Algorithm 1. Whenever a new frame is received, the frame is preprocessed by converting into OpenCV type followed by converting into gray scale and noise removal. From the processed image, features are detected and their descriptors are computed. These descriptors are then matched with all the frames present in the buffer. The good matches obtained after filtering with Lowe's ratio are then frame datatype and inserted into the buffer.When a new frame is inserted, RANSAC is utilized to remove outliers using homographic model. The good features matchings are then assigned with the same class ID. Finally features with their class ID for the frame are published.

2) Graph VIO Node: The graph VIO node subscribes to the keypoints topic and once received creates a Smart Projection Pose Factor. In GTSAM, for every landmark or feature there should only one instance of SPPF and thus can contain only one noise model. On the other hand, the *generic projection factor* is flexible in that sense where each landmark can have multiple instances of the factor and thus can contain multiple noise models. This drawback is handled by maintaining a map of feature ID and respective instance of SPPF. Whenever a new 2D feature Position is obtained the ID of the feature is checked in the map and if it exists the measurements are added to the respective instance of the SPPF.

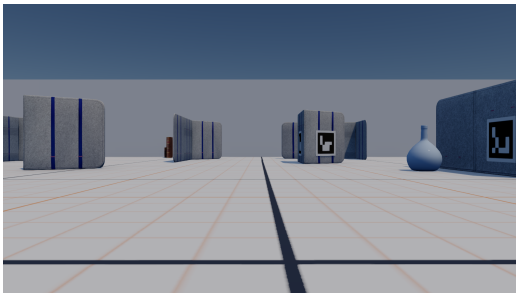
The IMU factor is created by using the GTSAM's *combinedImuFactor*. The values of the parameters for the IMU like Accelerometer and gyroscope standard deviations, their random walks, etc. are estimated using *Allan Variance ROS* Package prior to the implementation. Moreover, the camera intrinsics are calibrated with appropriate distortion model. The combined IMU-Camera Calibration is done using the tool ***Kalibr*** [17] Once all the factors are added to the graph, the graph is optimized using iSAM2 and last pose is published. SPPFs do not optimize the state of the features but the features can be triangulated to visualize. The triangulated features are published in the camera frame to visualize in Rviz.

IV. Results and Discussion

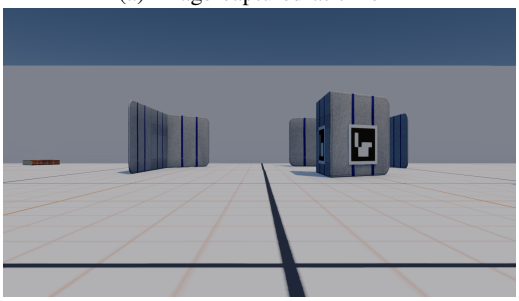
The `Turtlebot_VIO` Package was developed in ROS Noetic on an Ubuntu 20.04 machine. It was initially tested in the Stonefish simulator before being deployed onto the Turtlebot2 platform for further experimental validation.

A. Image processing

In first step, the robot moved in stonefish simulation environment and the image processing node was executed. As explained in the section III-A, in the image processing node the incoming `sensor_msgs/Image` msg shown in figures 8a and 8b is pre-processed by converting into gray scale and then noise is removed by applying gaussian blur as depicted in 9a and 9b. In the simulation environment the images are not feature rich as well as the patterns are repetitive thus features are not robust still the algorithm was able to find distinctive features as seen in 10. The RANSAC finds the outliers by finding the homograpic transformation between these images.

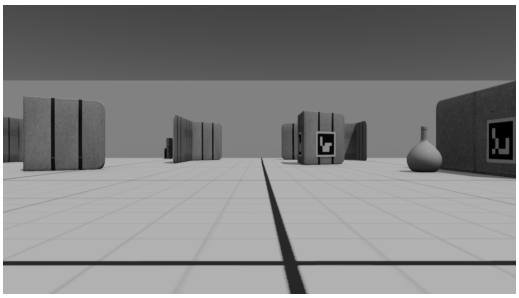


(a) Image captured at time k

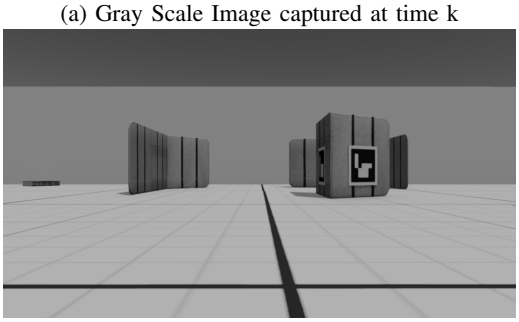


(b) Image captured at time k+1

Fig. 8: Image Captured at subsequent time step



(a) Gray Scale Image captured at time k



(b) Gray Scale Image captured at time k+1

Fig. 9: Image Captured at subsequent time step

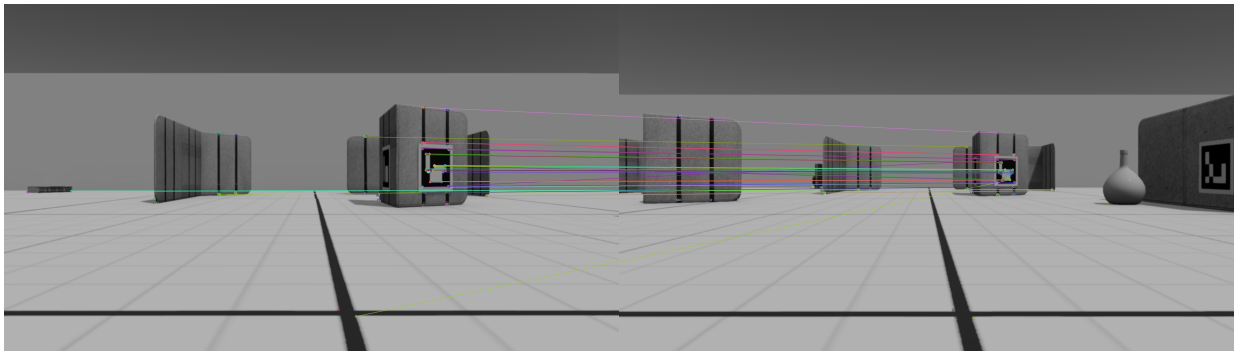


Fig. 10: Feature Matching for the Images captured at time step k and k+1

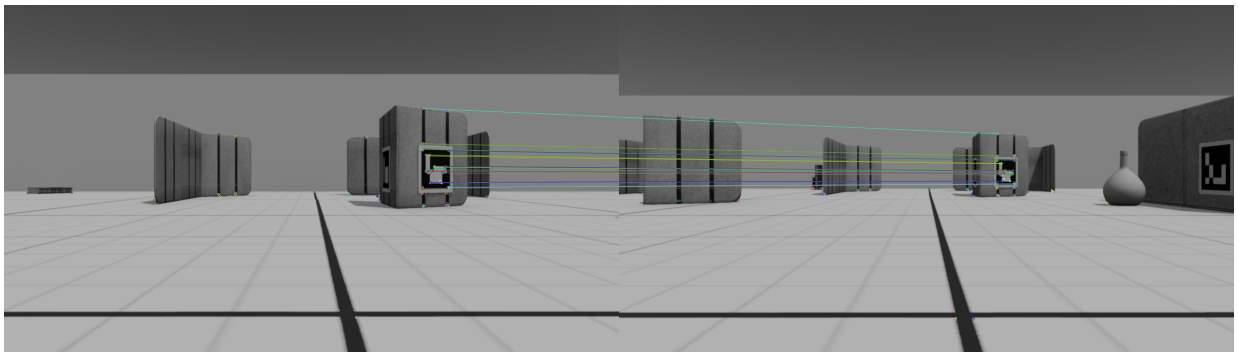


Fig. 11: RANSAC Outlier rejection for the Images captured at time step k and k+1

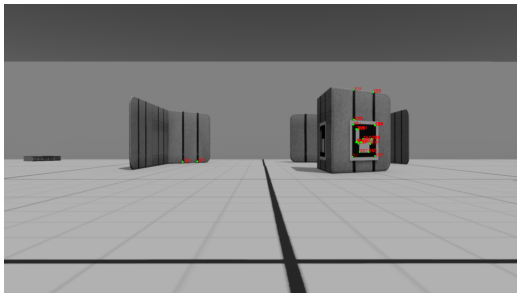


Fig. 12: classID Assigned to the features after matching and outlier rejection

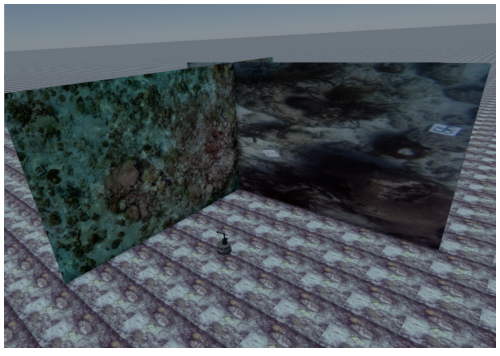


Fig. 13: Environment Created in Stonefish simulator

It can be observed in the figure 11 most of the outliers are removed. Once unique and robust matches are obtained, the keypoints are assigned Class IDs from the previous matches (See Figure 12).

As this environment is not feature rich, another environment is created with posters in the background and turtlebot was moved observing these posters as shown in figure 13. In simulation testing, the VIO was found to have good results when compared with ground truth as evident from the plot 14. As it takes time to process the incoming frame, the VIO lags behind the ground truth which can be improved by increasing the frequency of image capture.

V. Future Improvements

In this work an image buffer was introduced which can be used to store the previous frame to match features. Since this buffer has a limited length, it does not make sense to keep all the keyframes of the graph. The efficiency of the VIO can be improved by marginalizing the keyframes that are discarded from the buffer. Moreover, the efficiency of the image buffer itself can be improved by using clever storing techniques e.g. keeping only necessary frame data required for matching. The Camera and IMU spatio-temporal calibration (Independent and Combined) can be improved by performing more extensive calibration process. Estimation of IMU noise is a critical factor in the VIO, thus by gathering more IMU data the allan variance can give better estimation of the noise parameters.

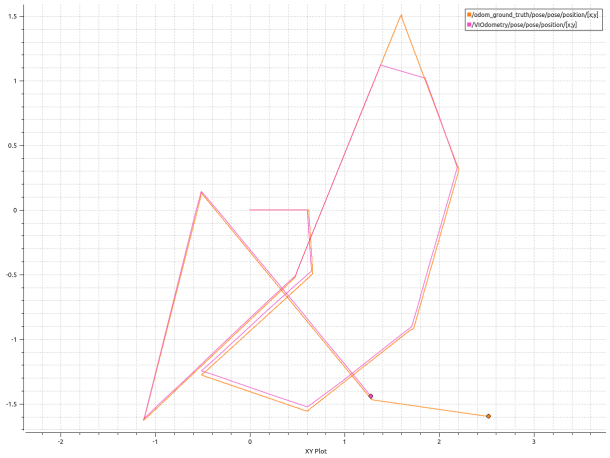
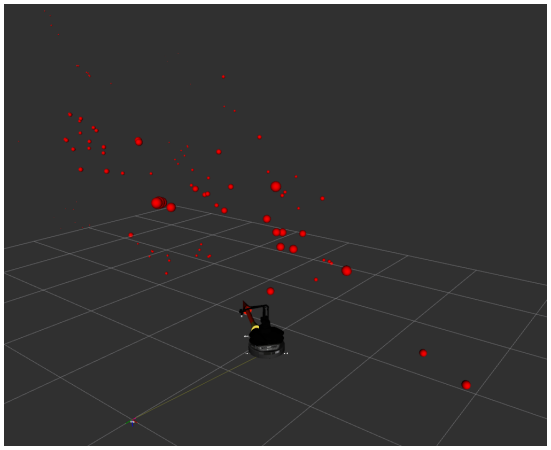
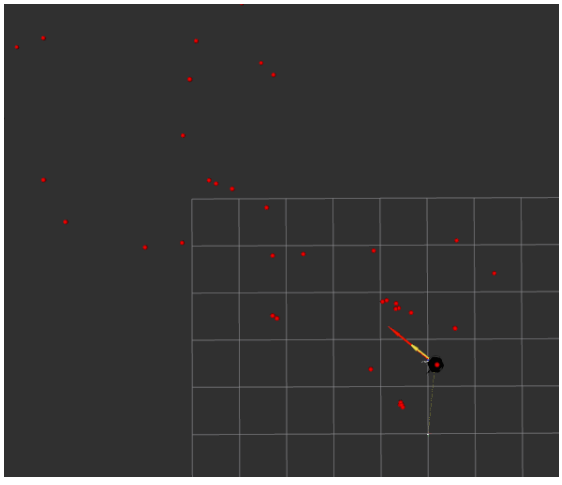


Fig. 14: Simulation Test: Ground Truth (in Orange) vs. Visual Inertial Odometry (in Pink) with Image Capture rate of 0.5 seconds



(a) Triangulated 3D Feature Landmarks in red by SPPFs in Camera Frame



(b) Triangulated 3D Feature Landmarks in red by SPPFs Top View in Camera Frame

Fig. 15: Triangulated 3D Feature Landmarks by SPPFs in Camera Frame, Red Arrow representing Ground Truth Current Pose and Yellow arrow is Current Pose estimated by VIO

VI. Conclusion

In this paper, we demonstrated the implementation of a Visual Inertial Odometry ROS package using Georgia Tech Smoothing and Mapping C++ Library. The paper also elaborates on the choices made during the development of the package e.g the choice of the SPPF along with the IMU factor. By tightly coupling camera and IMU data, the package leverages their complementary strengths, achieving accurate and efficient localization. The simulation testing validate the approach by comparing against the ground truth. Future work will focus on refining buffer management and sensor calibration to further enhance system performance.

References

[1] J. Nikolic et al., "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 431-437, doi: 10.1109/ICRA.2014.6906892.

[2] P. Furgale, J. Rehder and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 2013, pp. 1280-1286, doi: 10.1109/IROS.2013.6696514.

[3] Rehder, Jörn et al. "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes." 2016 IEEE International Conference on Robotics and Automation (ICRA) (2016): 4304-4311.

[4] Corke, Peter et al. "An Introduction to Inertial and Visual Sensing an Introduction to Inertial and Visual Sensing." (2007).

[5] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052-1067, June 2007, doi: 10.1109/TPAMI.2007.1049. keywords: Cameras;Simultaneous localization and

[6] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," Proceedings 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 2007, pp. 3565-3572, doi: 10.1109/ROBOT.2007.364024.

[7] K. Tsotsos, A. Chiuso and S. Soatto, "Robust inference for visual-inertial sensor fusion," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015, pp. 5203-5210, doi: 10.1109/ICRA.2015.7139924.

[8] A. I. Mourikis and S. I. Roumeliotis, "A dual-layer estimator architecture for long-term localization," 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Anchorage, AK, USA, 2008, pp. 1-8, doi: 10.1109/CVPRW.2008.4563131.

[9] Sibley, Gabe et al. "Sliding window filter with application to planetary landing." Journal of Field Robotics 27 (2010): n. pag.

[10] Leutenegger, Stefan et al. "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization." Robotics: Science and Systems (2013).

[11] Hesch JA, Kottas DG, Bowman SL, Roumeliotis SI. Camera-IMU-based localization: Observability analysis and consistency improvement. The International Journal of Robotics Research. 2014;33(1):182-201. doi:10.1177/0278364913509675

[12] S. . -H. Jung and C. J. Taylor, "Camera trajectory estimation using inertial sensor measurements and structure from motion results," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. II-II, doi: 10.1109/CVPR.2001.991037.

[13] Patron-Perez, A., Lovegrove, S. & Sibley, G. A Spline-Based Trajectory Representation for Sensor Fusion and Rolling Shutter Cameras. Int J Comput Vis 113, 208–219 (2015). https://doi.org/10.1007/s11263-015-0811-3

[14] Scaramuzza, D., Zhang, Z. (2020). Aerial Robots, Visual-Inertial Odometry of. In: Ang, M., Khatib, O., Siciliano, B. (eds) Encyclopedia of Robotics. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-41610-1_71-1

[15] M. Kaess, A. Ranganathan and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," in IEEE Transactions on Robotics, vol. 24, no. 6, pp. 1365-1378, Dec. 2008, doi: 10.1109/TRO.2008.2006706.

[16] Kaess M, Johannsson H, Roberts R, Ila V, Leonard JJ, Dellaert F. iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research. 2012;31(2):216-235. doi:10.1177/0278364911430419

[17] Paul Furgale, Joern Rehder, Roland Siegwart (2013). Unified Temporal and Spatial Calibration for Multi-Sensor Systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.

APPENDIX

Author	Contribution
Atharva Anil Patwe	- Literature Review - Image Processing Node - Graph SLAM Node - Construction of Environment - Testing - Report
Loc Thanh Pham	- Literature Review - EKF Odometry Node - IMU Preintegration - Testing - Report

TABLE I: Contribution of each author

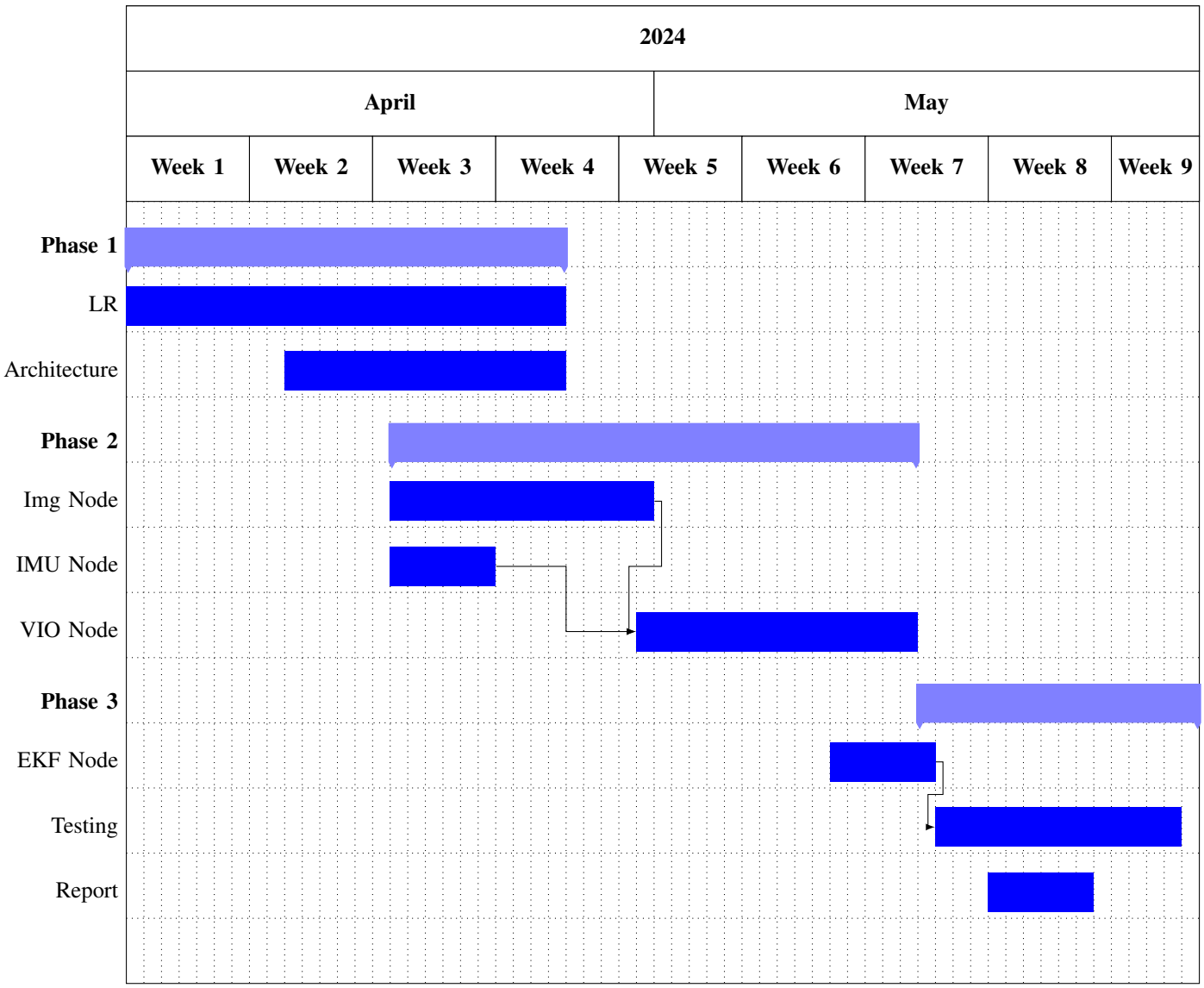


Fig. 16: Gantt Chart For the Hands On Project