Name: Aman Sangram Singh

Date: 02/06/2021

Underwater Localisation using a ROV set-up via VINS-MONO

by *Aman Sangram Singh*

Abstract

This paper provides a detailed description of the development of an underwater sensor vision data collection tool for mapping purposes and the use of VINS-MONO which is a visual inertial state estimator with a monocular structure having six degrees of freedom. After analysing various state estimation and mapping techniques it was decided to go with VINS approach in underwater conditions. One of the main jobs was to build a setup capable of collecting quality data in real time underwater testing. The setup consists of cameras and sensors, which gather appropriate data which would be used for analysing and mapping the underwater surface. The system consists of an acrylic enclosure with all the equipment including the power source fitted inside with the assistance of 3D printed parts. This setup is fitted over an underwater ROV using 3D printed structure supports which attach both together. The paper examines various design approaches for achieving the end goal of collecting good quality analysable data. The main blocker was to collect data in challenging environment like, poor lighting conditions and muddy water which reduces the camera and sensor range. Collected data includes the odometrical data, camera and IMU data. This is further analysed using VINS MONO for the purpose of underwater mapping.

Introduction

State estimation is one of the most fundamental bases for a vast range of applications, specially in robotics and AI which is one of the most rapidly growing field. Augmented or virtual reality (AR/VR), autonomous driving, drone mapping and robotic navigation are a few of the applications using state estimation principles [1]. The stereo camera setup still lacks in gathering the metric scale, which limits its usage in real work applications. But paring up the stereo camera with an inertial measurement unit IMU into a monocular visual inertial system (VINS) overrules the limitation and is capable of capturing metric scale along with roll and pitch data. The main aim of the project is to obtain a tool for autonomously or semi-autonomously constructing 3D maps of the complex underwater environment using the onboard sensors including spatial perception camera, IMU and laser sensors. This data is further analysed via VINS MONO for state estimation and map generation, which can further assist in monitoring coral reefs in sea, structural conditions in canals and so on. In the further sections the process of development of the physical system is explained. Two different arrangements were considered and tested. One which was successful was chosen for final test. The components, arrangement and explanation for each setup is explained in the next section. After that the sensors or camera used for this project are illustrated along with a brief explanation of the software setup used and the output data that is being generated which is required for further analysis. In the final part, analysis is shown using VINS ROS package and its outcomes are discussed. Output of the VINS package is shown in form of graphs and RVIZ representation. All of this is followed by the result an conclusion of the paper for using VINS in underwater scenario via a remotely operated vehicle setup.

Literature review

The use of highly sophisticated and efficient robotic systems such as ROVs, UAVs, drones and other autonomous robotic systems have drastically increased in recent years. Some of the most common commercial applications and uses for UAV Drones include aerial photography & Videography, real estate photography, mapping & surveying, asset inspection, payload carrying, agriculture, bird control, crop spraying, mining, aviation, disaster relief live streaming etc. The same trend of increased usage can be seen in case of underwater ROVs and unmanned vehicles, mainly for underwater mapping which has also seen drastic increase due to is necessity in worming of underwater autonomous or controlled robotics. In last two decades underwater ROVs and other autonomous or unmanned vehicles have found great use in ocean-seabed mapping. Some of the main application includes exploration of mineral resources, renewable energy, hydrography, marine pollution, research and development, oceanography or other multidisciplinary studies, fishery resource management, climate change environmental impacts, marine life, or reef health monitoring etc. [5]. Most of the applications using these systems work on state estimation principles to map their surrounding for navigation and tracking. The earlier systems being expensive and sensitive to sensor quality, did not quiet catch up with the increasing demand and usage[10]. Many solutions using different approach have come up in past years which are robust, versatile, cheaper, and more accurate. A technique involving large scale direct monocular SLAM (LSD-SLAM), proposed by Jakod E, Thomas S and Daniel C. It has the ability of large-scale consistent mapping of its environment. LSD-SLAM reconstructs the 3D environment in real-time with depth maps associated to pose graph of keyframes. Direct image alignment-based pose estimation with high accuracy is obtained. Essential steps involved are tracking, depth map estimation and map optimization [7]. ORB-SLAM proposed by Raul MA and JMM Montiel (2015) is a robust approach which uses the basic SLAM features for tracking, mapping loop closing and relocalization. The system is versatile for motion clutter allowing wide base-line loop-closing while relocalizationa and fully automated initialization. Applying "survival of the fittest" concept, only those keyframes and points are selected which results in better result and high robustness. It can work in both indoor and outdoor scenarios while attaining high accuracy of below 1cm for indoor setup and a few meters for outdoor condition, while generating a trackable compact map [9]. As the navigation system proposed by Jing-Hyuk J., Stuart W. and Salah S. (2006), which is a low-cost solution where guidance, navigation and control (GNC) is executed in a flight control system (FCS). Using a cost-efficient inertial measurement unit (IMU) and global positioning system (GPS) receiver together for the GNC. Controlling the UAV remotely via receivers and transmitters. The setup is capable of recognizing, transferring, and communicating data among multiple systems/UAVs for getting more accurate data [6]. In their article Helen M S, Sam A and Colin D (2019) talk about various seabed mapping techniques, while focusing on collection of bathymetry data for the ocean floor which comprise of roughly 71% of the total earth's surface. It starts with single and multi-beam echo sounders which used sound for measuring the depth, it was followed by satellite derived bathymetry (SDB) for shallow water. The most common method of light detection and ranging (LIDAR) which uses laser on an airborne platform, although it usage is again limited to shallow water [9]. In the Malta Cistern Mapping Project [8] an underwater robotic setup is used for exploring and mapping ancient cisterns on the island of Malta and Gozo. For this project, an underwater ROV was deployed with video and sonar equipment for gathering both video data and sonar range measurement. Due to unpredictability of underwater data collection issues caused by low illumination and other reasons the system deployed had the capability of six different mapping and localization techniques. These methods used are as follow:

- Sonar image mosaics using stationary sonar scans.
- Image mosaics using stationary sonar scans with Smart Tether position data.
- Simultaneous localization and mapping (SLAM) while the vehicle was in motion.
- SLAM using stationary sonar scans.
- Localization using previously created maps.
- SLAM while the vehicle was in motion with Smart Tether position data.

The result for the experiment reveals that an unexpected factor of low water level caused two of the chosen sites to be not fit for testing. The rest had limitations in gathering data for SLAM due to low mobility control. Although in motion SLAM worked well but the real time situation caused issued which were not there for pool testing. This shows the problems and unpredictable situations that might be faced in real time underwater testing[13].

Proposed by Aerial Robotics Group of HKUST [1], a monocular visual inertial system where in addition to a camera, an inertial measurement unit is used to get data. VINS-MONO is a robust and accurate state estimator where both camera and IMU data are integrated to reduce the noise and increase accuracy. This approach provides accuracy than the older techniques and provides better and more reliable result. Although being reliable and much accurate the new techniques still struggle when conditions are changed. When used for underwater applications, the struggle can clearly be seen as the data quality is compromised. In this paper an underwater ROV system is made with camera-IMU setup to check the feasibility of VINS Mono technique. The results will show the underwater usability of the algorithm with future scope. The system will help with tracking, navigation, and control of ROVs, unmanned vehicles and other robotic systems underwater. In reality, environment with 6 DOF causes issues with mobility of the submersibles, same as that faced by ariel drones. Along with that visual features might be easier to capture at one place and not so easy in the other, therefore when the visual data is of poor quality the inertial data is used for addressing the estimation issue. Various approach for making an independent underwater setup with data collecting sensors and onboard storage and computation capabilities. The development of an underwater system is not a trivial task but, with improvement in the manufacturing processes and availability of advances manufacturing techniques with modern material, the process had become much flexible as the engineer can take the design to its limits and be creative while keeping it feasible.[8][9] Techniques like 3D printing allows the design to be much reliable and innovative, this means that different sensors and components can be installed together without focusing too much on physical or arrangement compatibility. Various challenges should be kept in mind like buoyancy and mobility issues, but the most important one is keeping all the electronics and sensors away from water and making a watertight setup [4]. The success of the project will reveal the efficiency with which VINS can be used for pose estimation and mapping underwater.

The design approach or the research methodology for this project is the agile methodology. Agile methodology is an iterative and incremental approach in which the whole development process is broken into many steps and are referred to as mini projects. Each step is considered one at a time and goes through the whole development process, from concept through development, testing, and delivery.

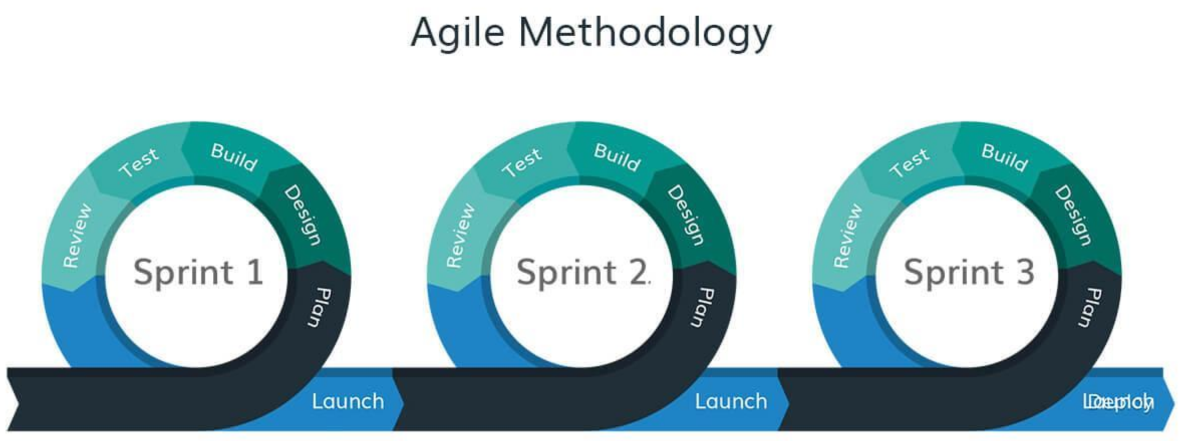


Fig 1: AGILE methodology flow, Image source- 9 reasons to choose Agile Methodology

The project was divided into short sprints for roughly two to three weeks each. This involved discussing the desired outcome for that particular component and planning the approach, this was followed by the designing part and the building part. The designing mostly consisted of the software like solid works, where different models were made and then assembled with the main ROV to visualise the practicality. After that, the build part consisted if 3D printing, removing the support material and then assembly. The last part was testing where all the components were installed together, and test were conducted for collecting data. This approach helped in breaking the project into small parts and assisted in focusing on each step one at a time.

System Approach

Base Model: BlueROV2

The base system used for the project is the BlurRobotics's BlueROV-2. It is a new generation mid-size, highly manoeuvrable and versatile remotely operated vehicle. Having four propellers and an onboard control system, it has easy controls and is high mobility. The external enclosure setup which is made in this project will be mounted on this via 3D printed support structures. The ROV uses Q-ground controller for navigation and tracking purposes.

Version1:

First version of the model setup included use of stereo cameras for gathering data which would be operated via onboard computing unit which in this case is Intel NUC. The underwater camera and sensor system which is managed by an onshore computing system can be used for collecting in the form of stereo videos. As the system consists of an onboard storage and computing system, the onshore controller will be communicating with the system via secure shell. Secure shell is a remote administration network protocol allowing the user to manage or control remote servers over internet, providing a mechanism for a remote user authentication and data transfer between client and host.

The approach used for the setup in this project was to mount the stereo cameras outside the main unit. Using equipment from Blue robotics, which is the provider of the BlueROV2, the 2-inch enclosers hold the two stereo cameras and the 6-inch enclosure holds the Intel NUC and the battery. FLIR stereo cameras are used, which capture RGB videos and images which can be further processed. The NUC has i5 5th gen intel chip with 4GB ram, enough to process and store the data captured. As the stereo cameras needs to be kept at about 12cm apart from each other for proper functioning, a 3D printed holding mount is used. This mount holds the two enclosures which holds the cameras at fixed distance from each other. An acrylic plate of 7mm thickness hold the camera mount at the top end of the 6-inch enclosure, making it a complete system with wires running externally, connecting cameras to the main computing system.

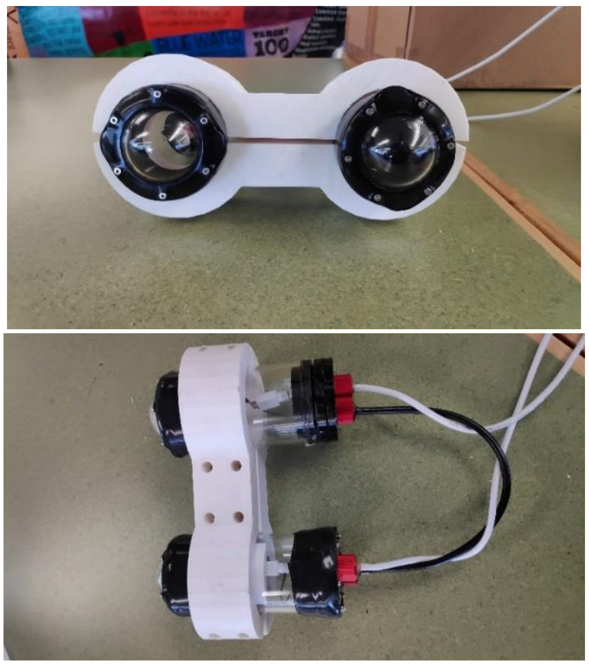


Fig 2- The stereo cameras mounted inside the 2inch enclosures, held together with the 3D printed mount.

Testing:

The data obtained from the stereo cameras contains data which is not up to a high usability when doing advanced analysis. Moreover, while using these cameras underwater reduces the range to a great extent. The 2-inch enclosures have a dome end where the cameras are places, this causes noise in the data even if there is a slight offset in the camera placement, as compared to a flat screen where data quality is much better. During the testing phase for this system, the 2-inch enclosures with stereo camera assembly was put together and dip in water before connecting to the whole system. The 3D printed mount which holds both the camera enclosures together, after submerging it for about 10 minutes at a depth of one meter, leaks were found in one of the enclosures. The whole enclosure was taken apart and reassembled with new sealing tape and tested again. Two consecutive testing still resulted in leaks in at least one of the enclosures. Considering the complex and fragile structure of having the wires outside the main enclosure and the leak issues, it was decided to go for a better approach with a single system with computing system integrated with the sensors and cameras.

Version2:

The new approach consists of developing a single unit which would consist of the sensors, computing system, storage, and the power source. Here new and advanced components are used for all the functions. For sensing part, the camera used is an AI camera, ZED2 which is a high-quality spatial perception camera capable of depth sensing, object detection, tracking etc. Laser sensor used in this is the Hokuyo UTM 30lX, which has a range of 30 meters with a 270-degree view angle. It is capable of working at higher speeds due to the wide range. The Zed2 camera needs NVIDIA packages to operate, thus NVIDIA NX is used which comfortably runs the camera and the sensor. The whole setup is placed inside an 8-inch enclosure (from Blue Robotics). The enclosure has an acrylic body with two aluminium heads on both sides. One covered with the metal plate for mounting wires and the other side is an acrylic plate which is transparent and perfect for the camera and laser. The arrangement has the camera facing the transparent plat mounted on a 3D printed platform which extends throughout the body of the enclosure dividing it into two parts, and has mounting points for the laser sensor, the NX in the top and the battery below. The total length of the completed system is 12.7 Inches which would sits over the ROV.

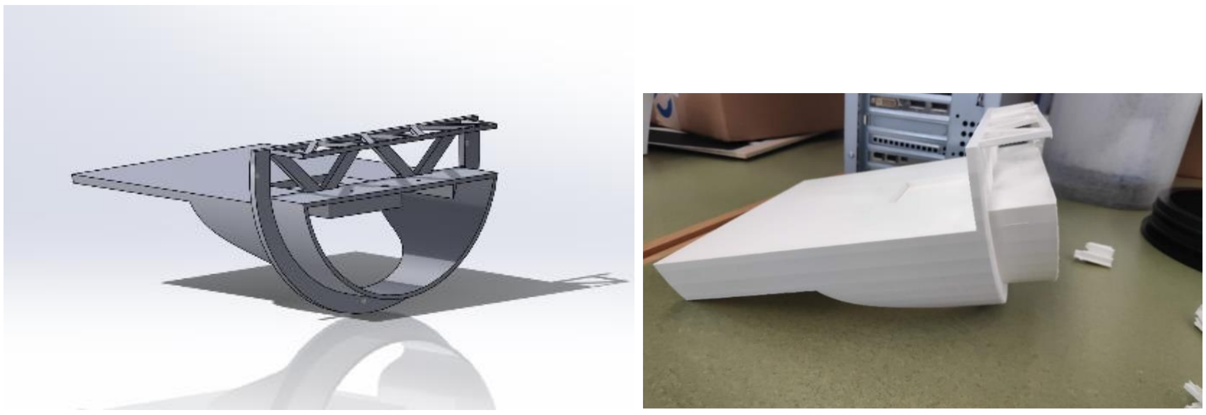


Fig 3- The CAD model and 3D printed part of the mount used for installing components inside the enclosure.

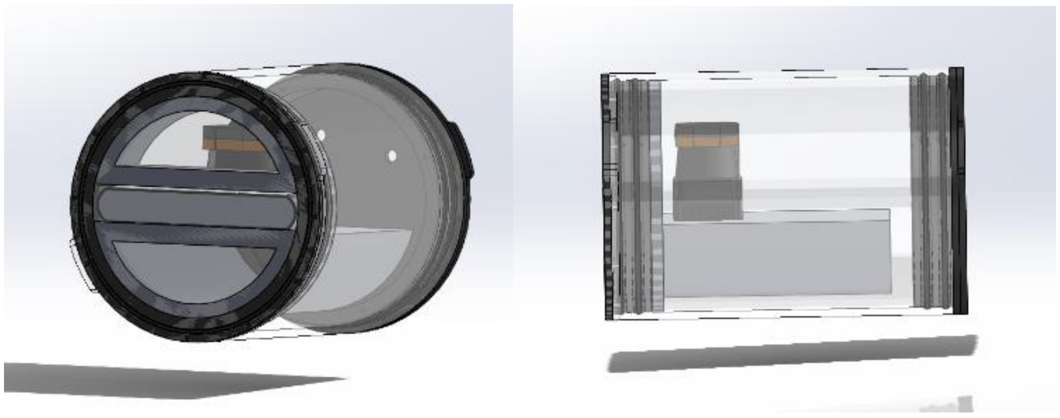


Fig 4- The CAD assembly of the camera and the laser sensor in the enclosure.

The ZED2 is a spatial perception camera which has many functionalities like depth sensing, measurements like IMU data, barometer and magnetometer data, object detection and tracking using localization and cloud connectivity. It can capture up to 2.2K video with 15fps at a resolution of 4416X1242. The stereoscopic camera uses two high-resolution sensors to detect vehicles and objects. It can therefore combine different viewpoints, with the help of an IMU and efficient motion sensors, to perform multiple operations like simultaneous, localisation & mapping, floor plane detection, skeleton tracking etc.

Fig 5- ZED2

Operating system used here is ubuntu where the camera is operated using ROS packages. The computing system is NVIDIA NX which comes with NVIDIA GPU required for operating the ZED2 camera.

ROS package: ZED-ROS-wrapper
The ZED ROS wrapper is the ROS package which provides access to the ZED stereo camera and other sensors which provide a verity of data, such as:

- Left and right rectified/unrectified images.
- Depth map
- Coloured 3D point cloud
- Visual odometry: Position and orientation of the camera
- Pose tracking: Position and orientation of the camera fixed and fused with IMU data.
- Spatial mapping: Fused 3d point cloud.
- Sensor data including accelerometer, gyroscope, barometer, magnetometer, internal temperature sensors.

The data published by the ZED camera node is on more than 40 topics from left & right camera, stereo pair, depth sensor, tracking & mapping, object detection and more. Following are some examples of the topics which are published:

Left Camera
rgb/image_rect_color: Color rectified image (left RGB image by default)
rgb/image_rect_gray: Grayscale rectified image (left RGB image by default)
rgb_raw/image_raw_color: Color unrectified image (left RGB image by default)

rgb_raw/image_raw_gray: Grayscale unrectified image (left RGB image by default)
rgb/camera_info: Color camera calibration data
rgb_raw/camera_info: Color unrectified camera calibration data
left/image_rect_color: Left camera color rectified image
left/image_rect_gray: Left camera grayscale rectified image
left_raw/image_raw_color: Left camera color unrectified image
left_raw/image_raw_gray: Left camera grayscale unrectified image
left/camera_info: Left camera calibration data
left_raw/camera_info: Left unrectified camera calibration data

Right Camera
right/image_rect_color: Color rectified right image
right_raw/image_raw_color: Color unrectified right image
right/image_rect_gray: Grayscale rectified right image
right_raw/image_raw_gray: Grayscale unrectified right image
right/camera_info: Right camera calibration data
right_raw/camera_info: Right unrectified camera calibration data

Stereo Pair
stereo/image_rect_color: stereo rectified pair images side-by-side
stereo_raw/image_raw_color: stereo unrectified pair images side-by-side

Depth and point cloud
depth/depth_registered: Depth map image registered on left image (32-bit float in meters by default)
depth/camera_info: Depth camera calibration data
point_cloud/cloud_registered: Registered color point cloud
confidence/confidence_map: Confidence image (floating point values to be used in your own algorithms)
disparity/disparity_image: Disparity image

Tracking
odom: Absolute 3D position and orientation relative to the Odometry frame (pure visual odometry for ZED, visual-inertial for ZED-M and ZED 2)
pose: Absolute 3D position and orientation relative to the Map frame (Sensor Fusion algorithm + SLAM + Loop closure)
pose_with_covariance: Camera pose referred to Map frame with covariance
path_odom: Sequence of camera odometry poses in Map frame
path_map: Sequence of camera poses in Map frame

Mapping
mapping/fused_cloud: Fused color point cloud

Sensor Data
imu/data: Accelerometer, gyroscope, and orientation data in Earth frame [only ZED-M and ZED 2]
imu/data_raw: Accelerometer and gyroscope data in Earth frame [only ZED-M and ZED 2]
imu/mag: Calibrated magnetometer data [only ZED 2]
atm_press: Atmospheric pressure data [only ZED 2]
temperature/imu: Temperature of the IMU sensor [only ZED 2]
temperature/left: Temperature of the left camera sensor [only ZED 2]
temperature/right: Temperature of the right camera sensor [only ZED 2]
left_cam_imu_transform: Transform from left camera sensor to IMU sensor position

Object Detection
objects: array of the detected/tracked objects for each camera frame [only ZED 2]
object_markers: array of markers of the detected/tracked objects to be rendered in Rviz [only ZED 2]

Diagnostic
/diagnostics: ROS diagnostic message for ZED cameras

As the system is controlled via secure network with the onshore computer over wifi, a strong signal is required as the Wi-Fi signal intensity decreases rapidly as the arrangement is dipped inside water. To solve the issue a Wi-Fi modem is placed inside the enclosure which would provide better signal quality when under water.

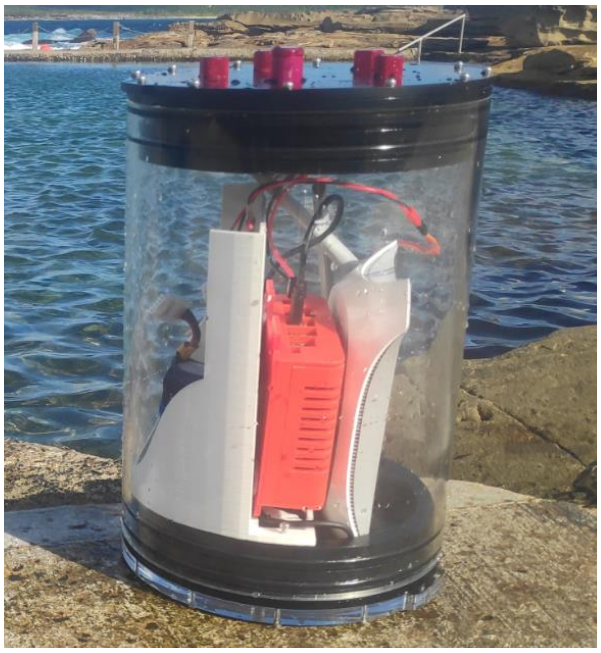


Fig 6- The complete setup of the enclosure with all the components.

The second step in this approach was to attach the enclosure to the BlueROV2 for better mobility of the whole system. The ROV will provide the fast movement underwater while keeping it stable. The Enclosure will provide its superior visual and odometry data collection capacity. Together the system can collect good quality data with minimal efforts. For this the design approach suggested that the enclosure should be placed on top of the ROV. This will help in maintaining the buoyancy and will not obstruct the water flow to the rotors. The mount that was designed for this was split into two parts, both connected to the side frame of the ROV and holding the enclosure on the top in centre directly above the main camera while resting on the main frame with generating any unwanted load which would compromise the sensitive components of the ROV.

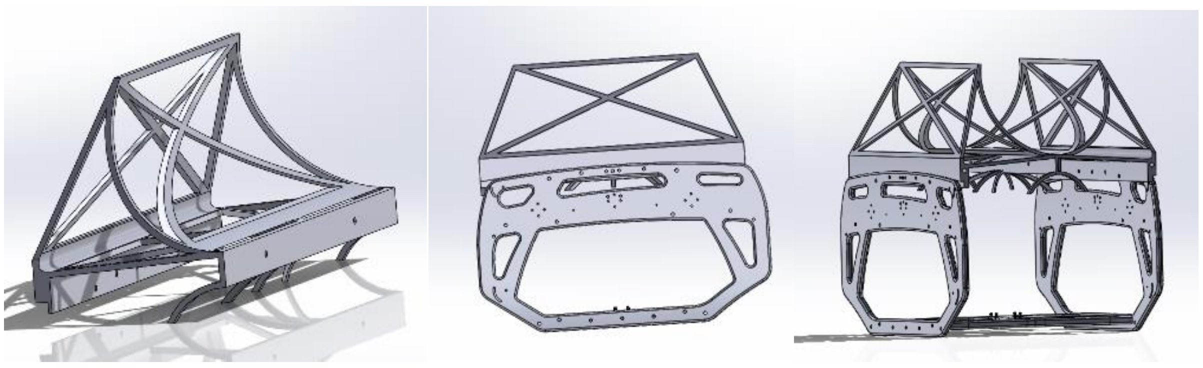


Fig 7- CAD model of the mount assembled on the ROV body.

Testing:

For the testing, all the components were installed together along with the battery, NX and the Wi-Fi modem. The connection to the onshore control system was made via secure network. The enclosure was dropped into the pool and was manually moved around with in a distance of 4-5 meters. The range was limited due to a weak local Wi-Fi network hosted via mobile. The test was successful as the camera was publishing all the required topics and they were being subscribed in the inshore system. Data recorded in form of rosbag consists of compressed image from the right camera.

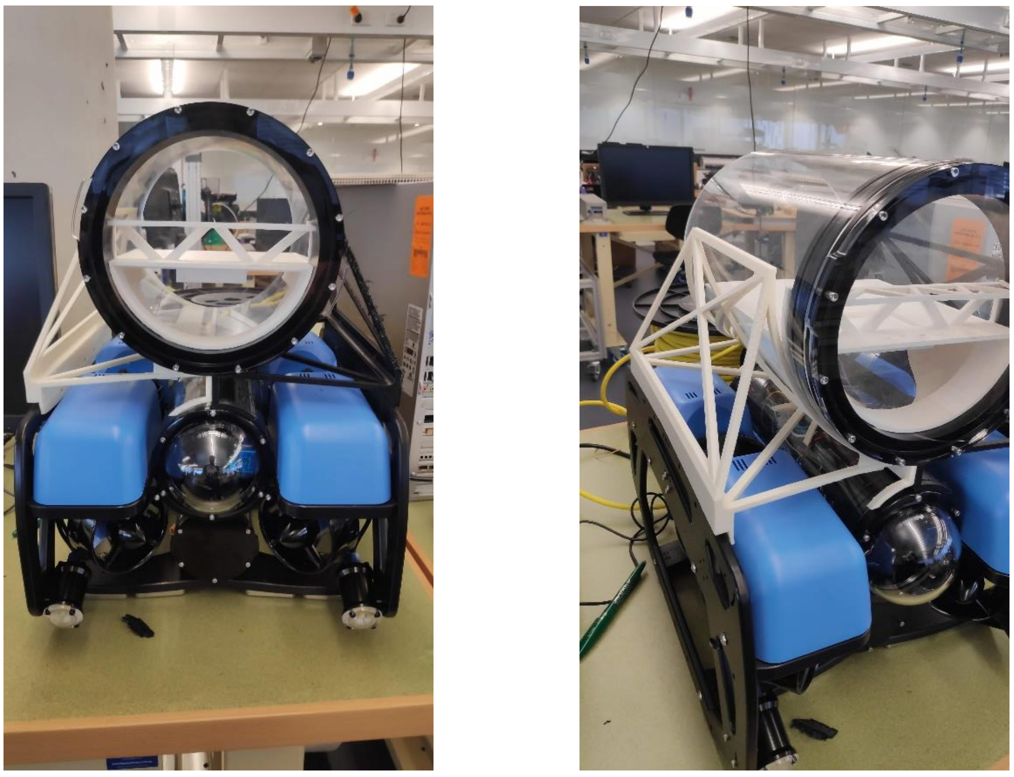


Fig 8- The ROV and enclosure mounted together via 3D printed mounts.

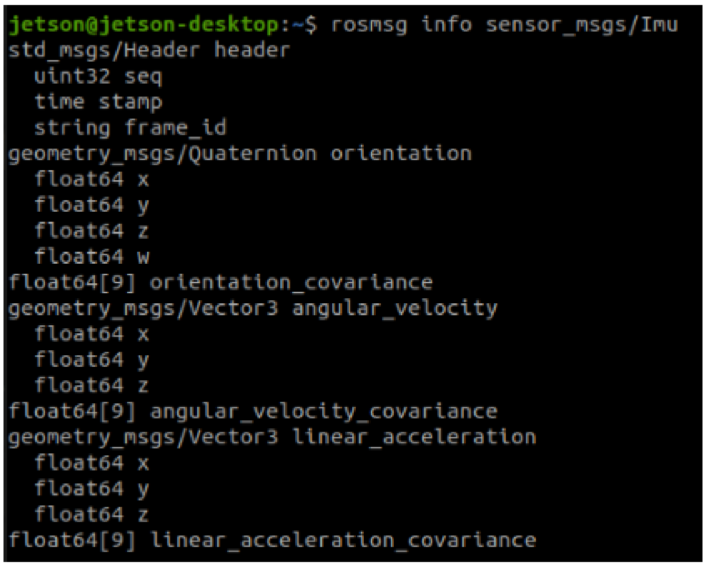
For the second test. It was decided that the ROV and the modem will also be used. The setup is mounted over the ROV system using 3D printed support structures which holds the 8-inch enclosure on top of the ROV horizontally in the direction of the main ROV camera, and the modem providing a better signal strength. The mounting structured are 300gm each which connects to the side member of the ROV and extends to the middle 4-inch enclosure, connecting to the structure mounted on the other side of the ROV. Both are screw together and to the ROV using 5mm screws and the external setup sits over the mounts which restricts its movement and holds it in place. The aim was to control the ROV as well as the enclosure unit from an onshore computer. But the testing in the end was done using only the enclosure. There was mobility issue with the testing location and there for the ROV was not used. The testing site was manually accessible, so the testing was done by moving around in the pool while carrying the setup, still being controlled by an onshore computing system.

Data collection

Data collected using the system had following parameters which VINS MONO package can utilise for analysis:

ZED2 – IMU raw data

Topic: /zed2/zed_node/imu/data_raw
Msg: sensor_msgs/IMU



```
jetson@jetson-desktop:~$ rosmsg info sensor_msgs/Imu
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Quaternion orientation
  float64 x
  float64 y
  float64 z
  float64 w
float64[9] orientation_covariance
geometry_msgs/Vector3 angular_velocity
  float64 x
  float64 y
  float64 z
float64[9] angular_velocity_covariance
geometry_msgs/Vector3 linear_acceleration
  float64 x
  float64 y
  float64 z
float64[9] linear_acceleration_covariance
```

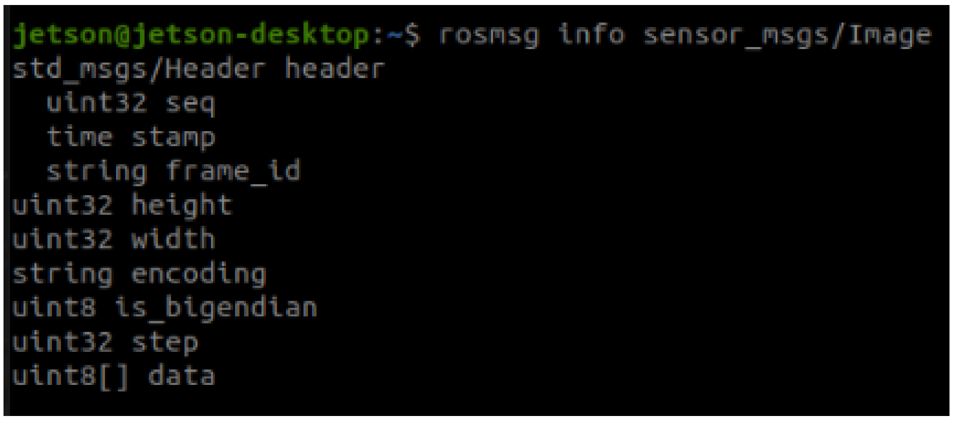
Fig 9- IMU message info

The inertial measurement unit (IMU) data provides data related to the movement of the body. It has sensors like accelerometer and gyroscope which gives the acceleration, velocity, and orientation of the body. The IMU message has a message type of quaternion orientation which provides velocity in x, y, z direction and w i.e., angular velocity or yaw in float64 data type.

It also gives angular velocity and linear acceleration in the x, y and z direction along with the linear acceleration covariance which is the difference between the actual reading and the predicted outcome. Data in angular velocity and linear acceleration is also in form of float64.

ZED2 – Left camera colour image

Topic: zed2/zed_node/left/image_rect_color
Msg: sensor_msgs/Image



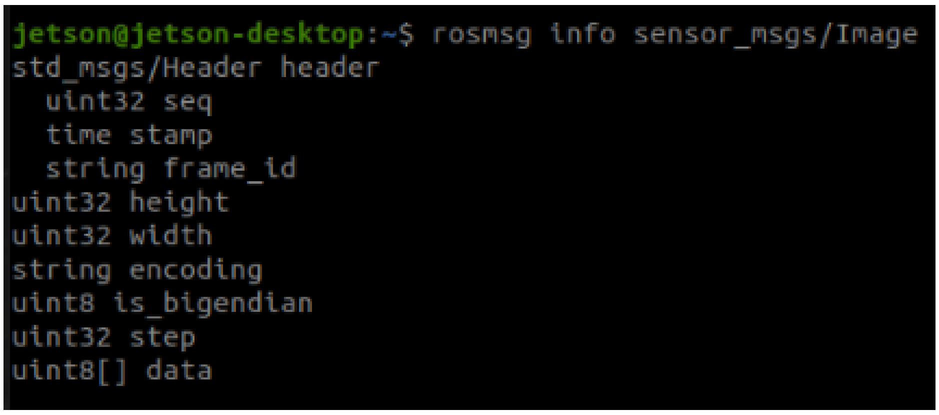
```
jetson@jetson-desktop:~$ rosmsg info sensor_msgs/Image
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

Fig 10- image message info

The ZED2 has a stereo camera which records video/image data from two cameras. The zed-ros-wrapper package publishes rgb colored images, raw images, greyscale images and rectified images. The sensor_msgs/Image message contains rectifies image which is published for both left and right cameras. A rectified image is an image which has gone through a process which is used for projecting different images on a common plane. This process is very helpful in computer stereo vision applications, as it solves the correspondence problem i.e. simplifies the work for finding matching points between different images. This message contains rectified image from the left camera.

ZED2 – Right camera colour image

Topic: zed2/zed_node/right/image_rect_color
Msg: sensor_msg/Image



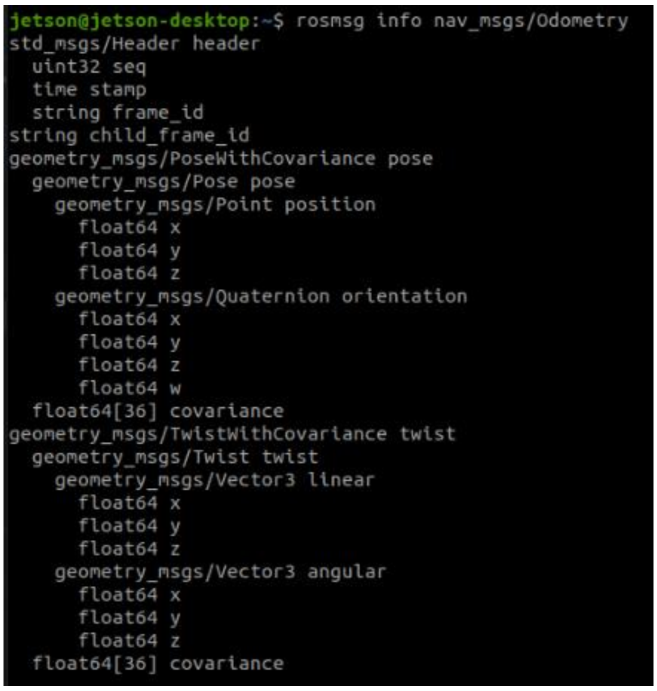
```
jetson@jetson-desktop:~$ rosmsg info sensor_msgs/Image
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

Fig 11- Image message info

This message is exactly like the one above. Only difference being, it contains a rectified image from the right camera.

ZED2 – Odometer data

Topic: zed2/zed_node/odom
Msg: nav_msgs/Odometry



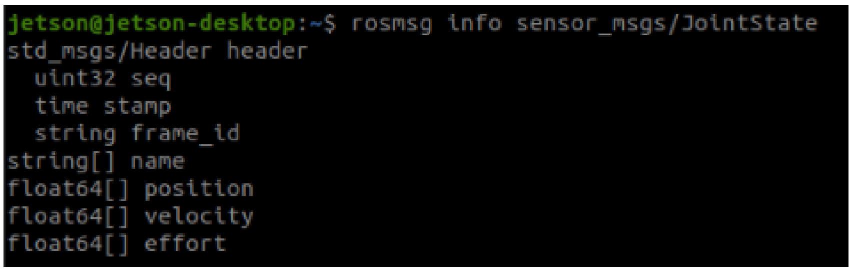
```
jetson@jetson-desktop:~$ rosmsg info nav_msgs/Odometry
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance
```

Fig 12- Odometry message info

Odometer topics provides velocity and position information of the associated movement in different directions. For ZED2 only visual odometry is applied. The message consists of information about the "Point position" and "Quaternion orientation" in float64 form in the x, y and z direction. This is a sub message to geometry message "Pose Pose", which is the sub message of "PositionCovariance Pose".

"TwistWithCovariance twist" message type in this message has a sub message "Vector3 linear" and "Vector3 angular". Both if these message types have data in x, y, and z in float64.

ZED2 – Joint states: /joint_states



```
jetson@jetson-desktop:~$ rosmsg info sensor_msgs/JointState
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string[] name
float64[] position
float64[] velocity
float64[] effort
```

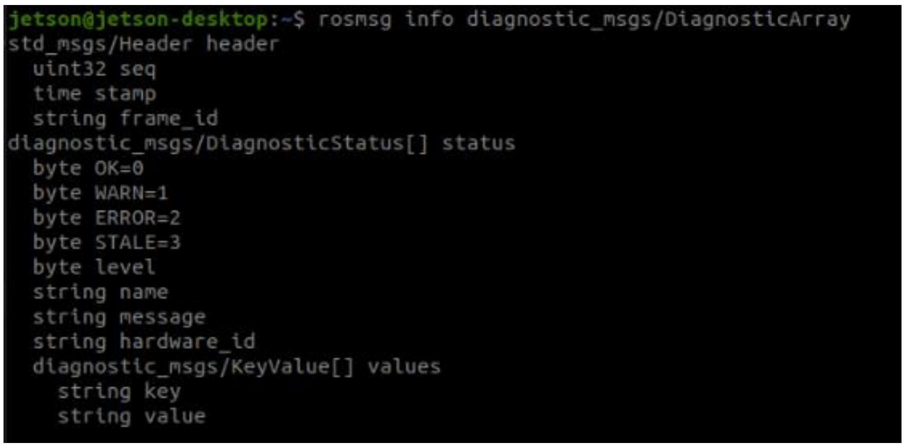
Fig 13- Joint state message info

The joint state message contains tool for publishing and setting values of a given joint state from URDF. URDF is unified robot description format, which is in XML form, representing robot model.

From parameter server it reads the robot_description parameter and finds out the non-fixed joints. Then it publishes JointState message, with all joints defined. The message information can be used in conjunction with robot_state_publisher node, for publishing transforms related to all joint states. The message stores the joint state data in float64 format.

ZED2 – Diagnostics

Topic: /diagnostics
Msg: diagnostic_msgs/DiagnisticArray



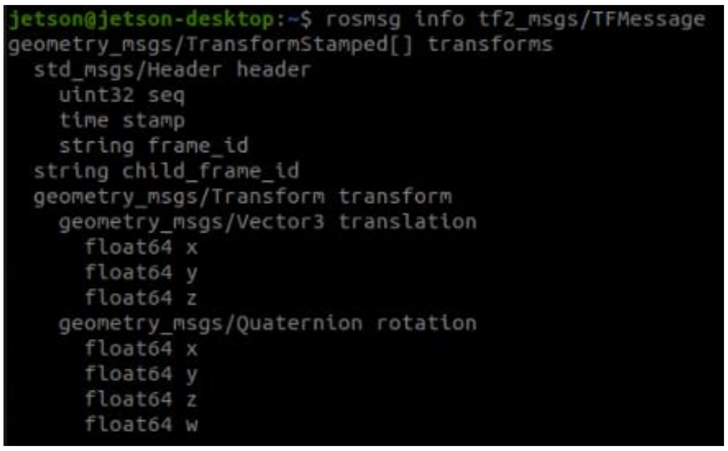
```
jetson@jetson-desktop:~$ rosmsg info diagnostic_msgs/DiagnosticArray
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
diagnostic_msgs/DiagnosticStatus[] status
  byte OK=0
  byte WARN=1
  byte ERROR=2
  byte STALE=3
  byte level
  string name
  string message
  string hardware_id
  diagnostic_msgs/KeyValue[] values
    string key
    string value
```

Fig 14- Diagnostic message info

Diagnostic message is a generic message type for robot operating system (ROS). It provides a standard interface for, runtime monitoring ROS system. These diagnostic messages are used by diagnostics Stack for providing libraries for ways to set and access the messages along with automated ways for processing the diagnostic data.

ZED2 – TF_static

Topic: /tf_static
Msg: tf2_msgs/TFMessage



```
jetson@jetson-desktop:~$ rosmsg info tf2_msgs/TFMessage
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion rotation
      float64 x
      float64 y
      float64 z
      float64 w
```

Fig 15- Time frame message info

For pose estimation and mapping purpose, a robotic system often has multiple 3D coordinate frames which change over time. There is one global frame, base frame, gripper frame, head frame and other local frames which align themselves with respect to the global frame for providing accurate results. tf message enables the use rot keep multiple coordinate frames in track by maintaining a relation between various coordinate frames in tree structure which are buffered over time.

The message has a message type of geometry_msgs/Transform transform, which contains vector3 translation in x, y, z and quaternion rotation for x, y, z and w. All the data in this message is of type float64.

VINS MONO- Data Analysis

VINS-Mono is a visual inertial state estimator with a monocular structure having six degrees of freedom. Proposed by Aerial Robotics Group of HKUST in 2017 [1], VINS-MONO is a robust and accurate state estimator. It can be applied over ROVs, MAVs, smartphones, and other custom setups. The main principle/working includes IMU pre integration, nonlinear optimization, estimator initialisation and visual inertial co-initialization. A monocular visual inertial system consists of a camera and an inertial measurement unit (IMU). Together both of these components form the minimum sensor requirement for a state estimator with six DOF. Although the camera and IMU setup bridges the gap between visual track losses and drastically improves the motion tracking along with being a versatile solution which is available on drones, mobile robots and devices, there are still several limitations. Coming to the rescue, VINS mono addresses the many issues using visual inertial odometry, global optimization, loop detection and relocalization. It provides on the go state initialization. Further the initialization module is also used for failure recovery.

The basic working of VINS algorithm includes tightly coupled sliding window non-linear optimization which provides a robust visual inertial odometry (VIO). The monocular visual inertial odometry (VIO) performs extrinsic camera IMU calibration along with finding accurate local pose, orientation estimation and velocity. It also performs IMU biases correction. VINS is a robust and versatile solution which can easily be used in a small-scale AR, medium size drones or full-size robotic systems using state estimation. As in this project where a medium size setup is used, which combines an underwater ROV along with an independent camera IMU setup installed in a single enclosure. The ROV overcomes the mobility issues faced by the independent enclosure and in return the independent enclosure overshadows the data collection limitations of the ROV by using state of the art stereo camera setup with inbuilt IMU and various other sensors which collect valuable and high-quality data. The vibration issues faced by the sensors and camera while mounted on the drone are addressed in this scenario, as the system is operating underwater where most of the heavy vibrations are absorbed by the surrounding water. But this comes at a price of poor visibility, as the camera range underwater decreased from around 35 meters to just 1 meter. Lighting provided by the ROV helps improve the situation along with IMU data which drastically improves the motion tracking by bridging the gap between visual track losses and overcomes the low lighting issues.

Working

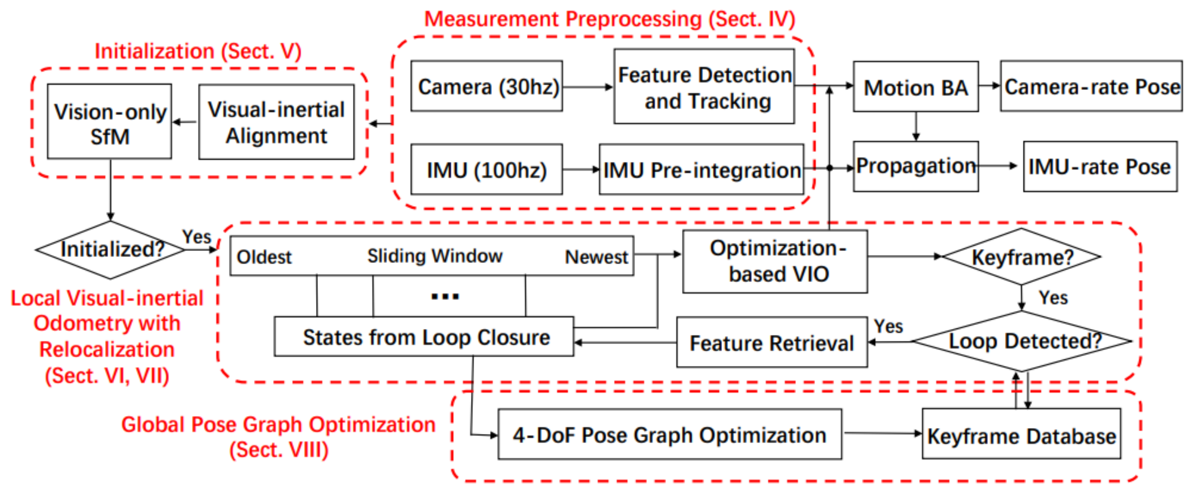


Fig 16- VINS-MONO flow diagram

The flow chart in the Fig shows the working of a section of the VINS-MONO algorithm. The diagram explains the structure for monocular visual inertial state estimation which consists of five sections. These sections include measurement pre-processing, initialization, visual inertial odometry, relocatization and pose graph optimization.

As in the figure, section VI is where measurement pre-processing happens, and feature extraction and tracking is done. Initialization process sits in section V, which provides most of the valuable data including pose, gravity vector, velocity, feature location and gyroscope bias, and bootstraps nonlinear optimization based visual inertial odometry (VIO). If initialized section VI which is visual inertial odometry (VIO) and relocalization modules which is section VII work together on the result/data from previous section. Pre-integrated measurements from the IMU, observed features and re detected loop closure features are tightly fused here. In the final step i.e., section VIII consisting of graph optimization module, uses results from geometrically verified relocalization and eliminates drift via global optimization. The visual inertial odometry (VIO), relocalization and pose graph optimization module operate simultaneously in multi thread setup with different operating rate for each module.

Measurement pre-processing is the first step in VINS, here the data collected from camera and IMU is pre-processed for further analysis. For the visual data, KLT spare optical flow algorithm is used for extracting features, along with this key frame selection also occurs here. Pre-integration is a technique which accurately summarizes all the measurements into one motion constraint. Initially proposed pre-integration describes rotation errors into Euler angles, while ignoring the bias in inertial measurements. In the VINS-MONO algorithm, the IMU bias correction is also introduced for increasing the accuracy. The algorithm is robust enough that even if gyroscope measurements have noise, it will still only pick up suboptimal keyframes. In parallel the IMU pre-integration is done, to improve the results IMU bias is introduced in the measurements. Posterior IMU bias correction is added to the result.

Estimator initialisation is the next step in this process. Here the main aim is vision aided inertial navigation. Motion estimation is to be done using the inertial and visual measurements. In VINS structure from motion (SfM) is used due to its initialisation property. IMU preintegration is aligned with the visual Sfm to trace actual scale measurements of gravity and velocity and also even bias. This part consists of two part, first uses sliding window vision only SfM and the second is visual inertial alignment. Visual-inertial alignment has gyroscope bias alignment which uses rotation from SfM and relative constraint from IMU pre-integration. The cost function can be represented as:

$$\min_{\delta b_w} \sum_{k\in\mathcal{B}} \left\| {\mathbf{q}_{b_{k+1}}^{c_0}}^{-1} \otimes \mathbf{q}_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} \right\|^2$$
$$\gamma_{b_{k+1}}^{b_k} \approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\delta\mathbf{b}_w \end{bmatrix},$$

Next is the initialisation of velocity, gravity and matric scale. Here the body frame velocities, for each frame are obtained in the window.

$$\min_{\mathcal{X}_I} \sum_{k\in\mathcal{B}} \left\| \hat{\mathbf{z}}_{b_{k+1}}^{b_k} - \mathbf{H}_{b_{k+1}}^{b_k}\mathcal{X}_I \right\|^2,$$

The last part is the gravity refinement. This leads to completion of the initialisation process. Next up comes the tightly coupled monocular VIO based on sliding window. This helps in making a highly robust and versatile state estimator. This step contains main calculation with maximum steps. All the steps here further extracts the key points from the data and improves the integration between visual and IMU measurements.

The last two steps are relocalization and global pose graph optimization. Relocalization eliminates drift by using a tightly coupled, relocalization module which can integrate the monocular VIO seamlessly. The main components of relocalization are loop detection and feature retrieval. Here the nonlinear cost function is further modified via additional loop terms as follow:

$$\begin{aligned}\min_{\mathcal{X}} \Bigg\{ \|\mathbf{r}_p - \mathbf{H}_p\mathcal{X}\|^2 + \sum_{k\in\mathcal{B}} \left\|\mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X})\right\|^2_{\mathbf{P}^{b_k}_{b_{k+1}}} \\ + \sum_{(l,j)\in\mathcal{C}} \rho(\left\|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_l, \mathcal{X})\right\|^2_{\mathbf{P}^{c_j}_l}) \\ + \sum_{(l,v)\in\mathcal{L}} \rho(\|\mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{v}_l, \mathcal{X}, \hat{\mathbf{q}}^w_v, \hat{\mathbf{p}}^w_v)\|^2_{\mathbf{P}^{c_v}_l} \Bigg\},\end{aligned}$$

Using the relocalization results from the previous step, graph optimisation adds keyframes into pose graph. Here pose graph management takes place where all keyframes are retained along with loop closure constraints. All other frames are removed.

To sum up, using camera-IMU monocular visual inertial system (VINS) provides following advantages:

- A versatile solution for initialization capable of bootstrapping the system/setup from an unknown initial state.
- Adventitious calibration and bias estimation for the camera-IMU system and optimization based tightly coupled monocular visual inertial odometry.
- Global pose graph-optimization with 4 degree of freedom and online loop detection.
- Tightly coupled relocalization and performance display in real-time scenarios for navigation and localization for drones, small scale robotic setups, ROVs etc.

Using the open-source PC version with ROS (robot operating system) integration The data collected from the setup is analysed.

Results:

The data recorded here is from Mohan Pool located in Maroubra, Sydney. Due to some physical and transportation limitations the ROV was not used for the data collection. The enclosure was moved around in the pool manually. The test was conducted during daytime therefore, the visual issues were not there, although with the use of ROV the conditions would become better because of the lights mounted in the front of the system, and the testing could also be done during the dark.

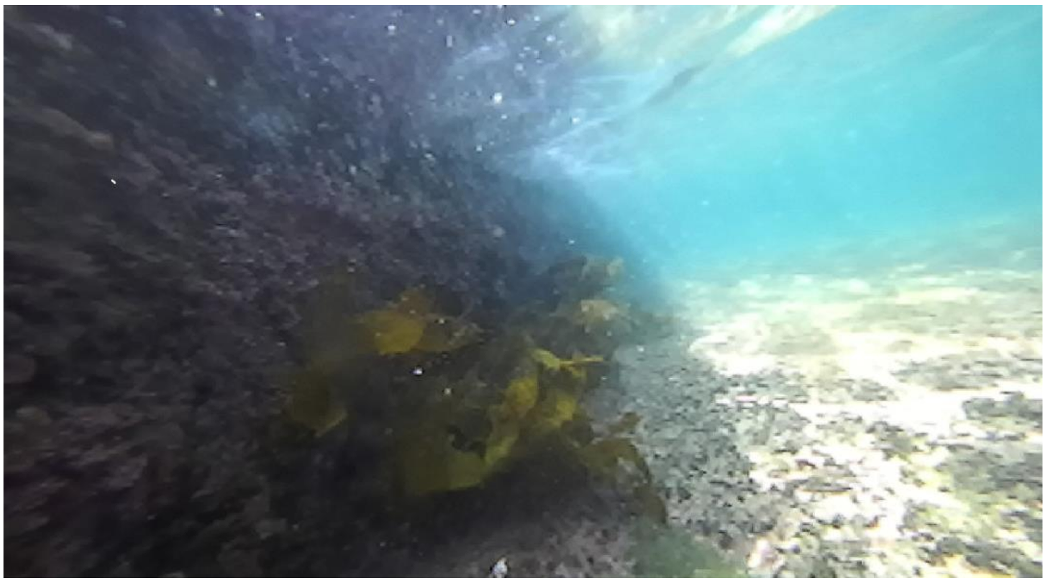


Fig 17– Stereo camera recorded RGB video

The other advantage which the ROV will provide is of stable movement, as the manual movement is not perfect and not in a straight line. The ROV has negligible effect of waves and thus can stay stable in moving water.

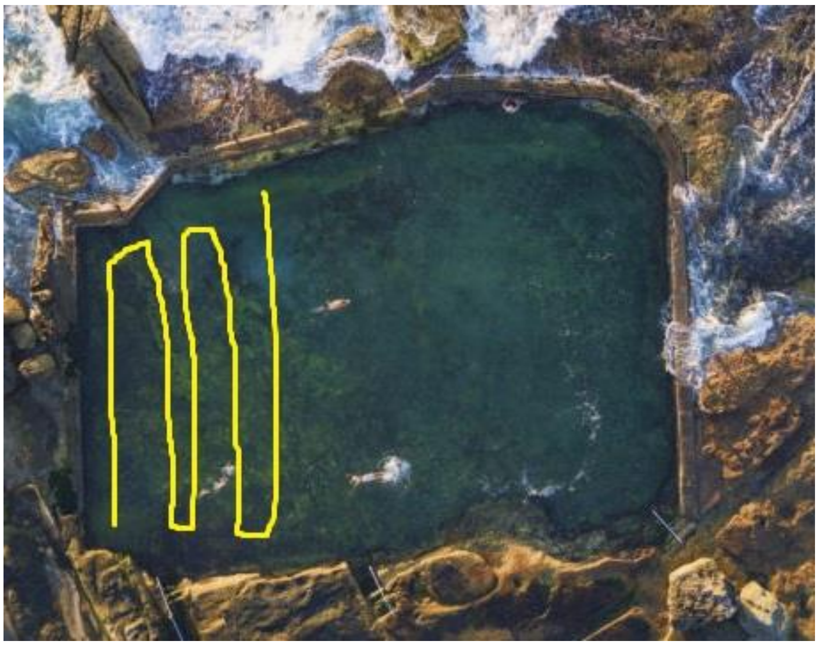


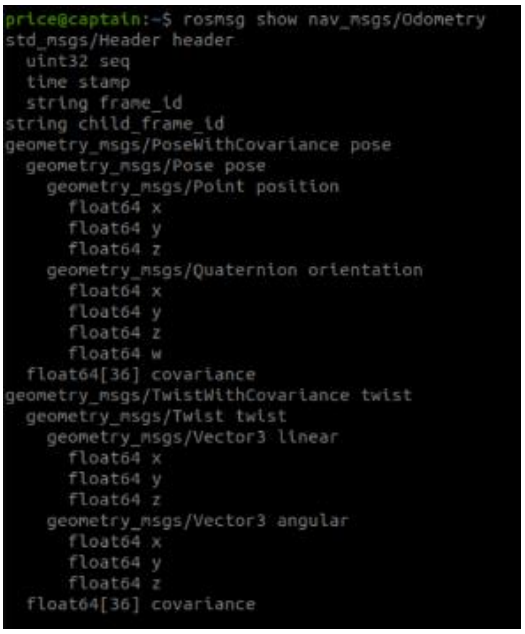
Fig 18- Route taken for the test

The recorded data was analysed using VINS and the results are discussed here. The test data recorded consists of the odometry, camera and IMU topics along with time frame and diagnostics which help in VINS analysis. The VINS results consist of the topics such as camera pose, image track, inertial measurement data, odometry data etc. Comparing the odometrical data for the ZED camera and the VINS analysis. We can see the superiority of the VINS module via the graphs where the odometer data is displayed. The messages published by the odometer and the path topic are as follow:

VINS – Odometer data

Topic: /vins_estimator/odometry

Msg: nav_msgs/Odometry



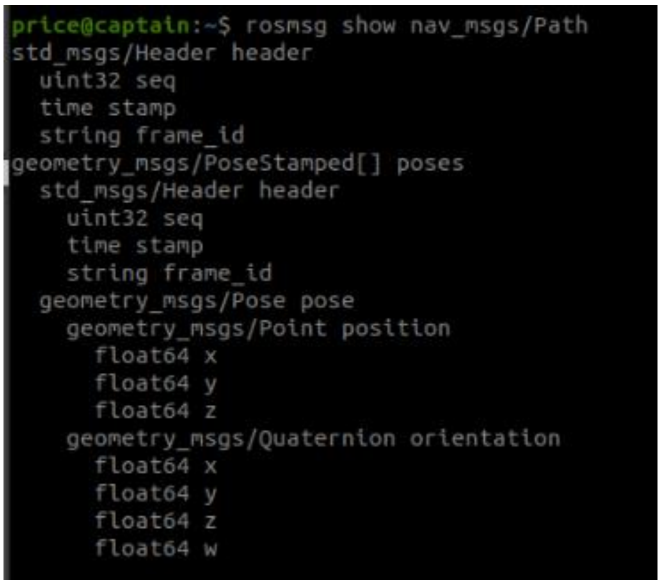
```
price@captain:~$ rosmsg show nav_msgs/Odometry
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance
```

Fig 19- VINS odometry message info

VINS – Path data

Topic: /vins_estimator/path

Msg: nav_msgs/path



```
price@captain:~$ rosmsg show nav_msgs/Path
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/PoseStamped[] poses
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
```

Fig 20- VINS path message info

Image below shows the features that are being extracted, using these features the algorithm tried to improve the pose estimation by managing the measure bias. The features extracted are used to match the same features when the loop is closed to increase accuracy and get better results. Closing the loop refers to following the same path so that the camera can gather same features/points again for bias correction. After detecting a loop the local sliding window connects the loop closure and retrieves the corresponding features. BRIEF description matching helps in founding correspondences.

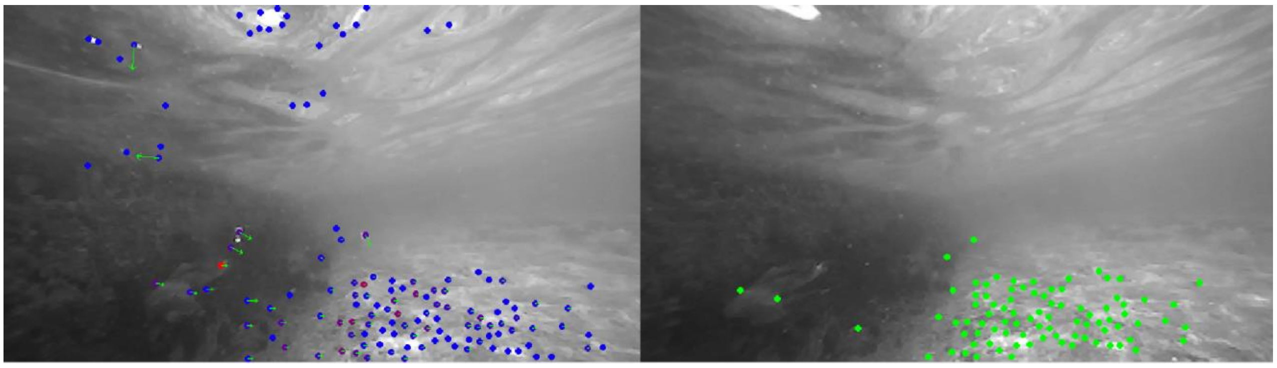


Fig 21- Feature detection

The relocalization is the process which aligns the current sliding window from the monocular visual inertial odometry. The sliding window is jointly optimized using the IMU and visual measurements. The nonlinear cost function used here is modified:

$$\min_{\mathcal{X}} \left\{ \|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}}(\hat{\mathbf{z}}^{b_k}_{b_{k+1}}, \mathcal{X}) \right\|^2_{\mathbf{P}^{b_k}_{b_{k+1}}} \right.$$
$$+ \sum_{(l,j) \in \mathcal{C}} \rho(\left\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^{c_j}_l, \mathcal{X}) \right\|^2_{\mathbf{P}^{c_j}_l})$$
$$\left. + \sum_{(l,v) \in \mathcal{L}} \rho(\| \mathbf{r}_{\mathcal{C}}(\hat{\mathbf{z}}^v_l, \mathcal{X}, \hat{\mathbf{q}}^w_v, \hat{\mathbf{p}}^w_v) \|^2_{\mathbf{P}^{c_v}_l} \right\},$$

The integration between the visual data and the IMU measurement is the one which gives the odometry data in the graphs below shows that comparison between the raw data from the ZED camera and the VINS-MONO. The testing was done by manually moving the system underwater by hands, thus due to the physical constraints the movement was not in proper parallel lines, instead got drifted while trying to move forward.

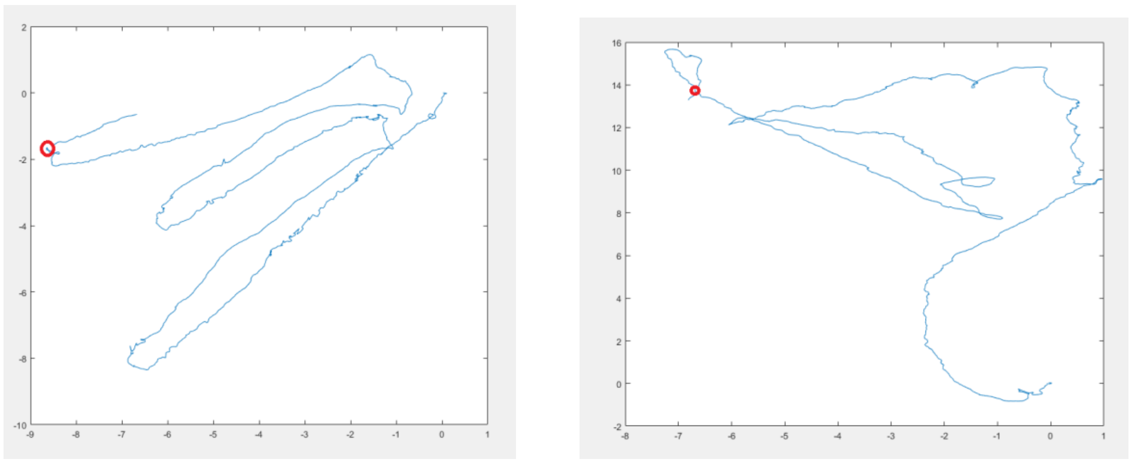


Fig 22- Odometry data from VINS and ZED in X-Y

The graph is the representation of ZED IMU data which is plotted in the X-Y direction. The starting point is close to the red dot on the graph. From the graphs it can clearly be seen that the odometry chard for the VINS in the X-Y axis has better accuracy. The path followed while testing is traced much closer to the real one. The bias correction in the VINS has improved the pose estimation and traced the path that is similar to actual path to a great extent.

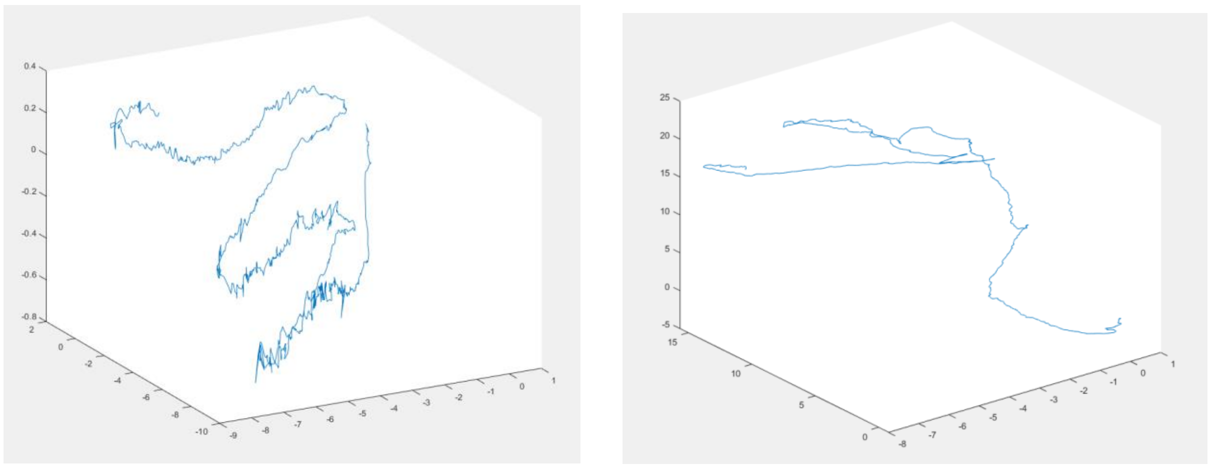


Fig 23- Odometry data from VINS and ZED in X-Y-Z

The graphs above are the representation of the odometer data for the ZED and the VINS-MONO in the X-Y-Z plane. The ZED graph shows distorted result with a lot of noise, which the VINS graph has less noise and is clear to read.

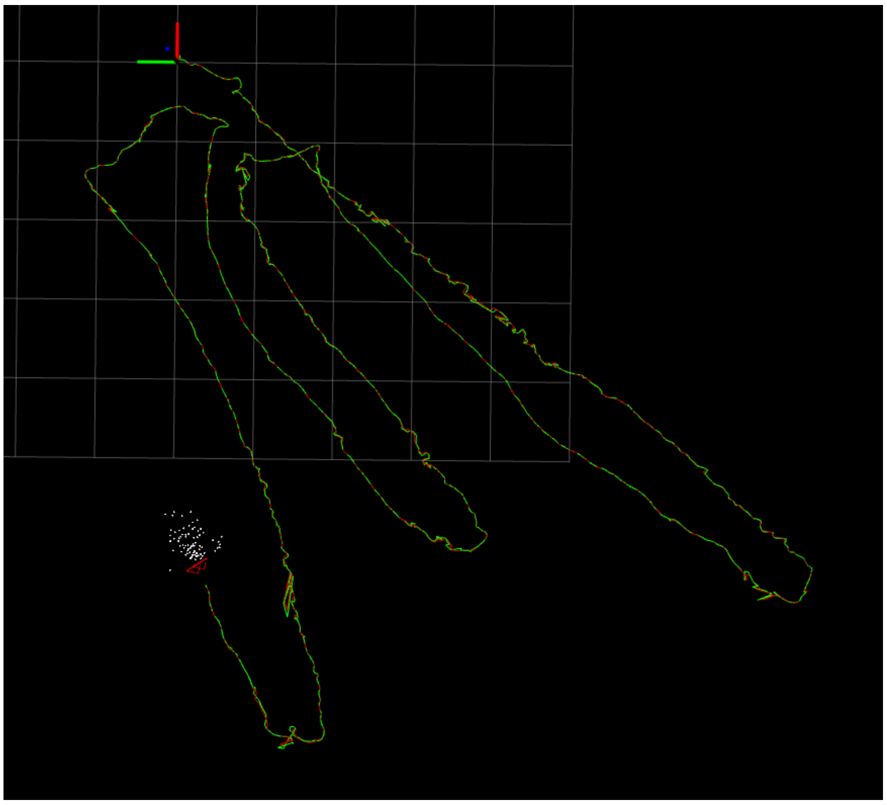


Fig 24- map merging in RVIZ for the recorded data

The results above are the red line is the ground truth and the green line represents the VINS_MONO. The plot is in X-Y axis and the Z axis being vertical to the plane as can be seen in the top left corner, represented by a blue dot.

Conclusion

In this paper a versatile and robust pose estimation solution was tested for underwater applications. The monocular visual inertial system (VINS) is the estimator algorithm which uses combination of the inertial measurement data along with the visual stereo camera data for increasing the accuracy. It performs IMU pre-integration, failure recovery, estimator initialization, relocalization, extrinsic calibration, tightly coupled visual inertial odometry and global optimization. The end result shows that the VINS module can be efficiently used for underwater applications. It not only provides accurate results but also make up for the poor visual data by integrating the IMU readings. The comparison shows the superiority of the VINS on the conventional models which showed difficulty in working when unfavourable environmental conditions were met. The easy of use for the VIN-MONO gives it an edge over others as a simple stereo camera and IMU setup can be used without extra effort.

The results prove that the VINS is robust and reliable enough that it can be used for experiments and other applications such as ocean floor mapping, coral reef monitoring, sea life study etc. The next step will be to perform experiment in the great barrier reef and the canal in Brisbane where the results will be valuable for future study.

References

[1] T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.

[2] A. Hogue, A. German, J. Zacher and M. Jenkin, "Underwater 3D Mapping: Experiences and Lessons learned," *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, 2006, pp. 24-24, doi: 10.1109/CRV.2006.80.

[3] Armagan Elibol, Nuno Gracias, and Rafael Garcia "Augmented State–Extended Kalman Filter Combined Framework for Topology Estimation in Large-Area Underwater Mapping",*Computer Vision and Robotics Group, University of Girona, 17071 Girona, Spain*

[4] P S Menandro & A C Bastos "Seabed Mapping: A Brief History from Meaningful Words",Marine Geosciences Lab (Labogeo), Departmento Oceanografia, Universidade Federal do Espírito Santo. *19 May 2020; Accepted: 7 July 2020; Published: 16 July 2020*

[5] Engel J., Schöps T., Cremers D. (2014) LSD-SLAM: Large-Scale Direct Monocular SLAM. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8690. Springer, Cham. https://doi.org/10.1007/978-3-319-10605-2_54

[6] C. White, D. Hiranandani, C. Olstad, K. Buhagiar, T. Gambin, and C. M. Clark, “The malta cistern mapping project: Underwater robot mapping and localization with ancient tunnel systems,” Journal of Field Robotics, 2010

[7] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.

[8] T. Qin, P. Li, and S. Shen, “Relocalization, global optimization and map merging for monocular visual-inertial SLAM,” in Proc. IEEE Int. Conf. Robot. Autom, Brisbane, Australia, 2018.

[9] M. Shinohara *et al*., "Development of a High-Resolution Underwater Gravity Measurement System Installed on an Autonomous Underwater Vehicle," in *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 12, pp. 1937-1941, Dec. 2018, doi: 10.1109/LGRS.2018.2863261.

[10] A. Filisetti, A. Marouchos, A. Martini, T. Martin and S. Collings, "Developments and applications of underwater LiDAR systems in support of marine science," *OCEANS 2018 MTS/IEEE Charleston*, 2018, pp. 1-10, doi: 10.1109/OCEANS.2018.8604547.

[11] Kim, J., Sukkarieh, S., and Wishart, S. Real-time navigation, guidance and control of a UAV using low-cost sensors. In Proceedings of the International Conference of Field and Service Robotics, Yamanashi, Japan, 2003, 95–100.

[12] A. Anwer, S. S. A. Ali and F. Mériaudeau, "Underwater online 3D mapping and scene reconstruction using low cost kinect RGB-D sensor," *2016 6th International Conference on Intelligent and Advanced Systems (ICIAS)*, 2016, pp. 1-6, doi: 10.1109/ICIAS.2016.7824132.

[13] Baek H, Jun BH, Noh MD. The Application of Sector-Scanning Sonar: Strategy for Efficient and Precise Sector-Scanning Using Freedom of Underwater Walking Robot in Shallow Water. Sensors (Basel). 2020 Jun 29;20(13):3654. doi: 10.3390/s20133654. PMID: 32610644; PMCID: PMC7374510.

[14] Kang, S.; Rong, Y.; Chou, W. Antidisturbance Control for AUV Trajectory Tracking Based on Fuzzy Adaptive Extended State Observer. *Sensors* **2020**, *20*, 7084. https://doi.org/10.3390/s20247084

[15] Kang S, Rong Y, Chou W. Antidisturbance Control for AUV Trajectory Tracking Based on Fuzzy Adaptive Extended State Observer. Sensors (Basel). 2020 Dec 10;20(24):7084. doi: 10.3390/s20247084. PMID: 33321909; PMCID: PMC7763566.