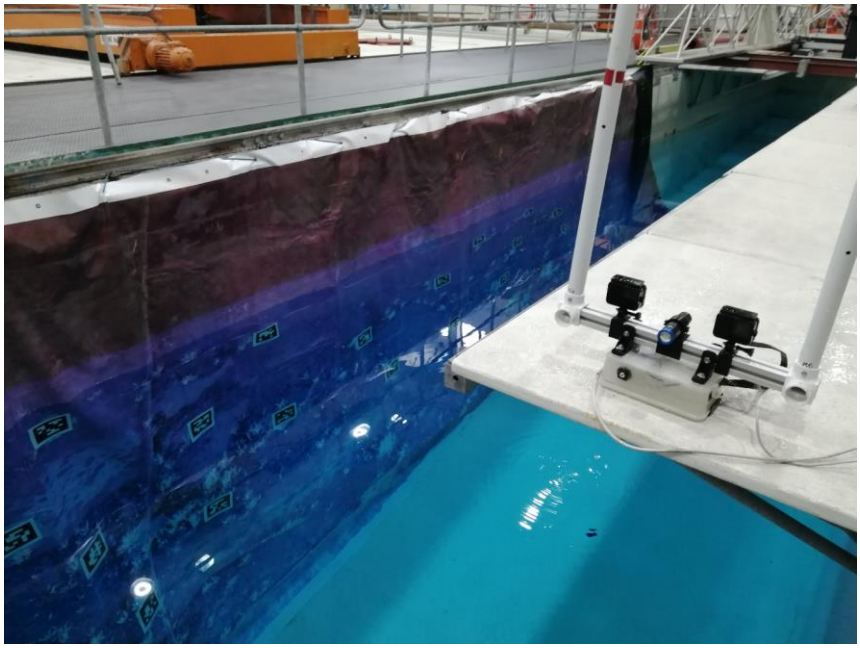
ARU Ship Hull Vinyl Dataset

Methods

Data Collection Method

An inside sweet water tank at IMT in Cape Town was used to take a controlled underwater dataset using a prototype sensor rig. The target scene is a life-sized photomosaic of the side of a commercial ship hull that IMT had previously taken when the ship was last in dry dock. The photomosaic is printed onto a large vinyl tarpaulin and hung along the side of the tank. The tank used for data collection can be seen below.

Because the tank platform is 1m from the surface of the water, two 3m PVC pipes were used to manoeuvre the rig through the water, seen below. The rig is moved horizontally along the side of the platform facing the vinyl surface. Marks were placed on the PVC pipes as a visual aid to keep the rig at a relatively constant depth when moving horizontally. Once at the end of the vinyl, the rig is lowered directly downward to the next marked depth and moved horizontally back along the platform, creating a square ‘C’ shape. The data collection starts at 0.5m depth and returns at 1m depth.



Ground Truth / Datum for Comparison

The need for a datum for comparison has been considered. AprilTags were laminated and placed on the vinyl prior to data collection for the purpose of creating a datum trajectory, as seen in the above and below images. AprilTags are designed to be highly recognisable and detectable in images, often used in underwater SLAM applications for creating a “ground truth” or datum trajectory [3]. The AprilTags can be used as very robust landmarks in the SLAM factor graph formulation to create a more accurate trajectory for comparison as well as providing highly recognisable landmarks for definite loop closures [3]. When the AprilTags are utilised in this way in the SLAM pipeline, the resulting trajectory will have higher accuracy than when these additional constraints are not included.

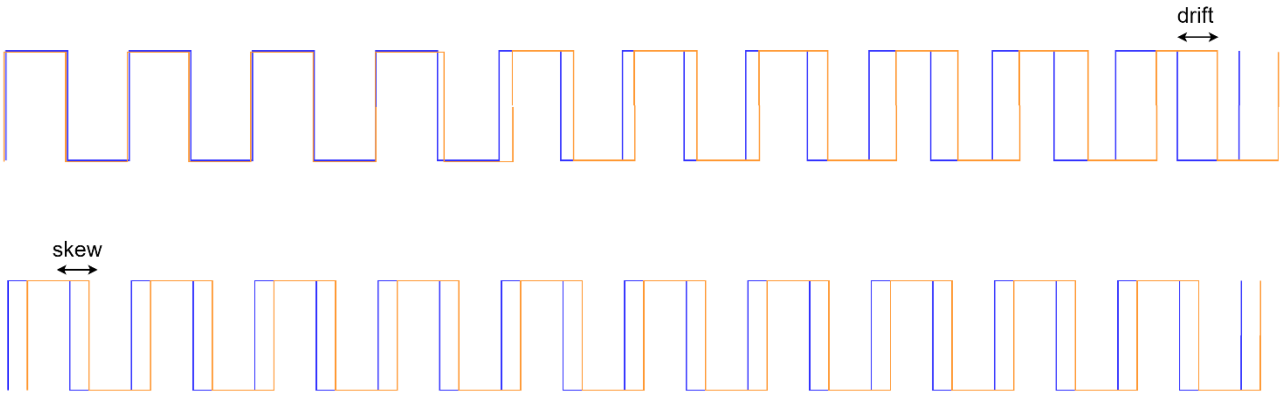
Calibration

A number of calibrations are performed for the dataset: camera intrinsics, stereo extrinsics and stereo-inertial. All final calibrations are included in the dataset meta data as yaml files as well as the original calibration data to enable users to perform their own calibrations if necessary.

Monocular camera intrinsic calibrations are performed in-air in the African Robotics Unit (ARU) laboratory for left and right cameras. These calibrations are computed using the MatLab app, 'cameraCalibrator'. The Pinax model, a method from undistorting underwater images [4], is used for image undistortion in conjunction with the monocular calibrations. The water temperature and salinity, used to find the water index of refraction for the Pinax method, are measured on the day of data collection using a commercial conductivity, salinity and temperature meter. A stereo calibration dataset is taken on the day of data collection using a checkerboard to find the stereo camera extrinsics, which is evaluated using MatLab's app 'stereoCameraCalibrator'. The final calibration performed is between the stereo cameras and IMU, to find the extrinsic relationship between these sensors. The stereo-inertial calibration ROS library, Kalibr [2] is used to perform this calibration. An AprilTag board is used as the calibration target, taken in the ARU laboratory. Kalibr requires the accelerometer and gyroscope noise density and random bias walk and so the IMU noise parameters were found by collecting data with the IMU stationary over 9 hours and plotting the Allan Variance in MatLab with code provided by a colleague, Michael Noyce. With his permission, Michael's MatLab code is included in the dataset resources.

Synchronization

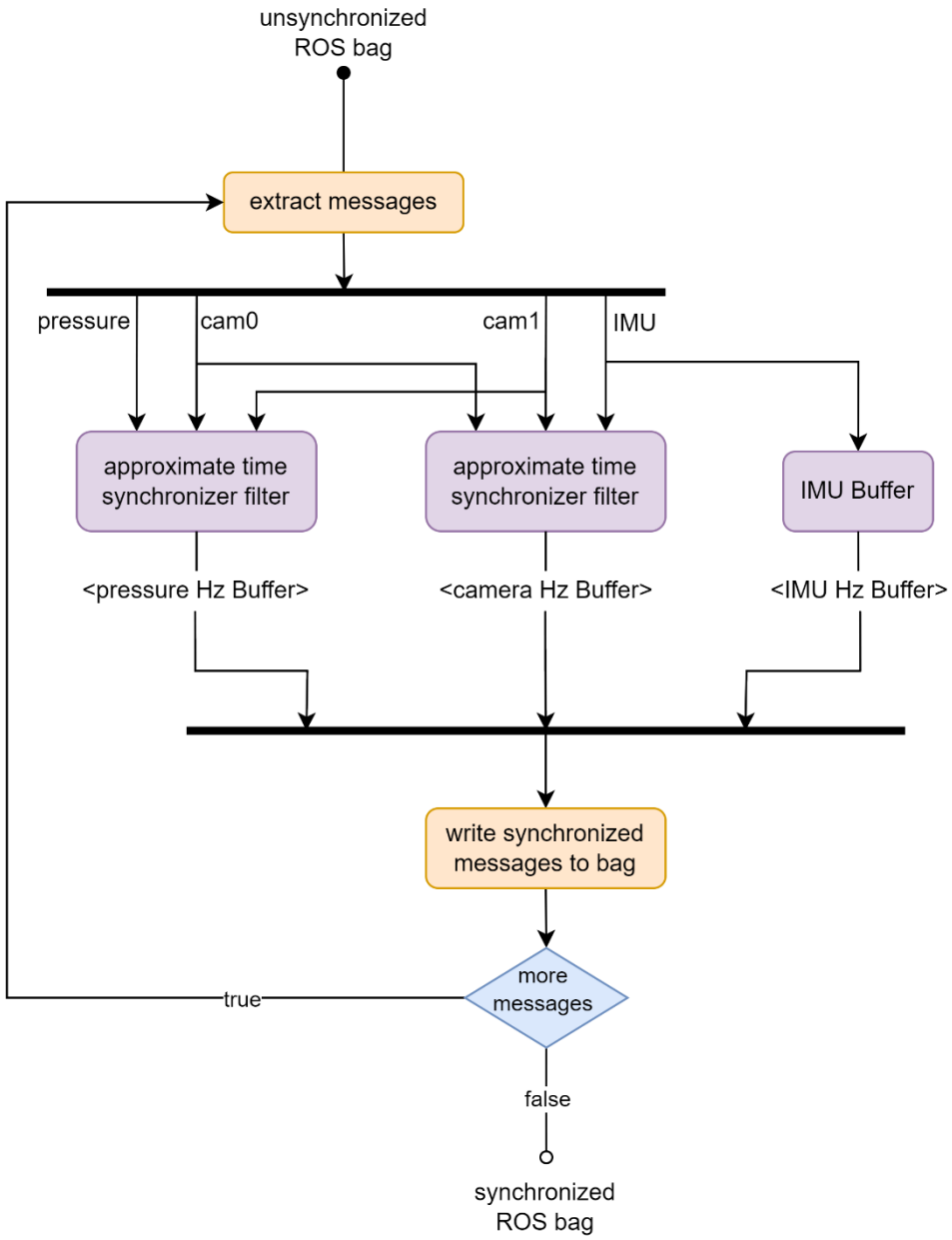
Good synchronization is imperative for visual odometry and SLAM pipelines. If the time step between two stereo camera streams is too large, the disparity value between matched features will be incorrect, leading to larger errors in scene depth and odometry estimations. If there is no possibility of hardware-level synchronization, we can synchronize data in software. One way of doing this is by using timestamps that have been allocated to sensor data messages by a central processor that has a single clock signal. If timestamps are allocated to data streams from decentralized processors or chips, their clock signals could be out of synch because of clock drift and skew, adding a layer of complexity to the synchronization. The diagram below illustrates the clock drift and skew problem when using decentralized clocks.



For this reason, ROS bags are used for centralized data collection of multiple sensors, allowing the use of ROS timestamps for data synchronization. A ROS synchronization package was written in C++ for the synchronization of stereo cameras, a pressure sensor and an IMU and is available on this project's GitHub [1].

The ‘Approximate Time Synchronizer’ filter is used from the ROS 'message_filters' package to synchronize messages. Because it is approximate time, it does not expect the timestamps to be exact, instead, it does a best-match search over the message header timestamps to find sets of synchronized messages across the input topics. When synchronizing sensors with different sampling rates, this filter outputs synchronized messages at the lowest sensor frequency, so to preserve the higher frequencies, a series of buffers are implemented with the Approximate Time Synchronizer filters. The output is a synchronized ROS bag with original sensor frequencies preserved.

There are three options for sensor configuration in the package: stereo camera, stereo-inertial and stereo-inertial-pressure. The diagram below shows the operation of the stereo-inertial-pressure code.



The package was tested on ROS Melodic on Ubuntu 18.04 and ROS Noetic on Ubuntu 20.04. The stereo synchronization has been tested on our own datasets collected in the African Robotics Unit (ARU) research lab at the University of Cape Town as well as on two datasets from the Autonomous Field Robotics Lab (AFRL) from the University of South Carolina, while the stereo-inertial and stereo-inertial-pressure synchronization has only been tested on datasets taken in the ARU. During evaluation, each message was saved with its original timestamp instead of saving all synched messages with the timestamp of the left camera and MATLAB was used to find the timestamp differences of synched messages.

The results of the data synchronization tests are shown in the table below, showing the maximum and average difference in timestamps (milliseconds) in relation to the left camera for the right camera, IMU and pressure sensor messages.

	Right Camera	**IMU**	**Pressure Sensor**
AFRL [ms]	Max: 987 Avg: 6.94	-	-
ARU [ms]	Max: 33 Avg: 0.09	Max: 40 Avg: 5.69	Max: 32 Avg: 5.58

Table showing the maximum and average difference in milliseconds in sensor timestamps after synchronization. Differences are given in relation to the left camera timestamps.

Resources

[1] Adriennewinter, GitHub - adriennewinter/Marine_Vision: African Robotics Unit - MSc project at the Univer-
sity of Cape Town. [Online]. Available: https://github.com/adriennewinter/Marine%5C_Vision.

[2] Ethz-Asl, Home. [Online]. Available: https://github.com/ethz-asl/kalibr/wiki.

[3] E. Westman and M. Kaess, “Underwater apriltag slam and calibration for high precision robot localization”, in tech. rep, Carnegie Mellon University Pittsburgh, PA, 2018.

[4] T. Łuczy ́nski, M. Pfingsthorn and A. Birk, “The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings”, Ocean Engineering, vol. 133, pp. 9–22, 2017.