Realtime Onboard Dense RGB-D Mapping on UAVs

Ayush Gupta
ayushgup@iitk.ac.in

***Abstract*— Dense RGB-D map of environments have applications in aerial drone navigation, robotic manipulation, reconnaissance operations and semantic mapping. This mapping process is quite computationally expensive, with the main bottleneck being the generation of point clouds from the visual inputs. We have made use of both Monocular Cameras and RGB-D cameras for evaluating the feasibility and accuracy of the systems, while noting the processing speed of the entire system, accounting for the less computational power present on an aerial drone due to its scarce size and energy resources.**

I. Introduction

In recent times, 3D reconstruction and mapping using unmanned aerial vehicles (UAV) have gained a lot of interest in the robotics community. In comparison to ground vehicles, UAV are more versatile in their ability to reach inaccessible places more quickly and efficiently. With onboard cameras, the UAVs can take aerial images during flight, which can be used to process 3D maps of the entire environment as it progresses. This mapping procedure can then be used for localization, and even extended to the Simultaneous Mapping and Localization (SLAM) Problem. Being capable of 3D SLAM means that a drone can easily capture a detailed understanding of the world around it, which points to new opportunities in autonomous navigation that can be made available to any developer.

The following sections are organized as follows, where we first present related works in the fields of 3D Reconstruction using optical sensors, and the general techniques and work flow of visual mapping and SLAM problems. Next, we present the hardware configurations, followed by the various methods we evaluated for their consistency and speed. Then we present the real time on-board results, which are finally followed by the conclusion, limitations and current status of work.

II. Literature Review

There are numerous publications relating to 3D reconstruction and mapping, with the majority of them focusing on using depth cameras for better per-pixel depth information. Due to hardware constraints, we first focused on using a monocular camera to create depth point clouds, and then further use them to create entire maps across a larger area.

Matia Pizzoli, et al. presented REMODE, where they estimated dense and accurate depth maps from a single moving camera. Probabilistic depth measurement on a per-pixel basis and it used a Bayesian estimation for rejecting erroneous estimations using the computer uncertainty. A real time implementation was provided by them, which has a CUDA-based implementation running at 30 Hz on a laptop computer.

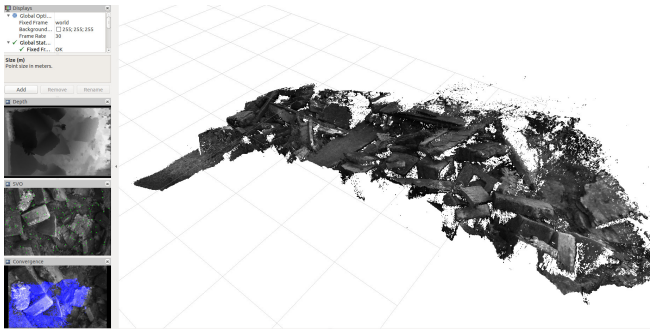


Fig. 1: Surface elevation mapping using REMODE

We also tried using two Monocular cameras in a stereo pair, and tried generating disparity maps after calibrating them. For this, we reviewed the basics of epipolar geometry, triangulation techniques, feature matching and disparity map generation. ETH-Z provided an open-source calibration tool, Kalibr which provided with multiple-camera calibration in addition to IMU-camera calibration. Stereo-block matching technique is generally used for creating disparity maps from a pair of images. The frames from the two cameras should be time-synchronized, which provides consistency to the feature matching technique, especially if the camera is not stationary.

Our final conclusion was to use RTAB-Map (Real-Time Appearance-Based Mapping). This is a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the maps graph, then a graph optimizer

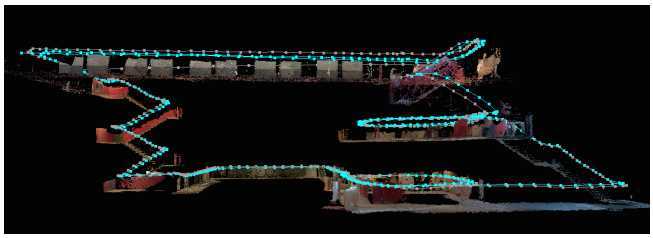


Fig. 2: Multi-level 3D RGB mapping using RTAB-Map

Fig. 3: RTAB-Map with failed loop closure

minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environments are always respected.

The recent increase in deep learning methods for problems in computer vision has resulted in development of many networks which are able to predict depth from monocular cameras as well as estimate the optical flow from a video feed.

III. Considerations and aims

Since our goal to map the 3D environment is quite difficult, in terms of computation power as well as result, we tried to solve the problem with a focus on the processing required by the algorithm, as well as the final resolution and detail of the map which could be achieved.

We aimed to achieve as high frame rate as possible with the entire system running on the onboard computer. The mapping procedure can be slow, since we can then make any controller make its way through the environment by giving it a path. Also, we tried to make sure there are not a lot of errors due to back projection from the camera, or having a lot of scale drift at different distances which would make ruin the map for us.

IV. Hardware Configuration

Most of the techniques have been tested on an 8th Gen Intel Core i7-8550U processor, with 8 GB of DDR4 RAM. The laptop has an entry-level Nvidia 940MX GPU, with CUDA and CUDNN installed on a Ubuntu 16.04 Xenial setup. Robot Operating System (ROS) was used as a middleware, and many open-source and publically availbable codes were tried and tested on this laptop. We used a pair of Logitech C930e webcams, with the framerates fixed at 30 FPS or 15 FPS.

Later in the project, we recieved a Stereolabs ZED Camera, which was a RGB-D camera which can capture depth at resolutions up to 2.2k and frame rate up to 120 FPS, with the computation being done on the Graphical Processing Unit of the computer to which the camera is attached.

Fig. 4: Stereolabs Zed Camera

For onboard implementations, we had been provided with a Nvidia Jetson TX2, built around an NVIDIA Pascal-family GPU and loaded with 8GB of memory and 59.7GB/s of memory bandwidth. It was mounted on a ConnectTech Astro Carrier, replacing the bulkier Nvidia TX2 Developer Kit, while providing with a reasonable number of serial ports in a small size. We also had access to a Pixhawk Cube which was used to record the odometry messages generated using its inbuilt IMU.

V. Implementations and Findings

From the initial phase of the project, we did a lot of literature review, and implemented these techniques on real data captured from the Helicopter Lab. Here, we present the methods and implementations in a chronological fashion, since the start of the project.

In the initial phase of the project, we first tried to calibrate the two separate cameras as a stereo camera. This mainly consisted of finding the intrinsic parameters of the camera, as well as the baseline shift while considering the cameras together. The intrinsic parameters were first found out from the internet, which resulted in a huge amount of

re-projection error. Then, we used the Kalibr toolset for this calibration. We created rosbag files infront of a AprilTags setup, and the Kalibr toolkit itself uses many rounds of iterations for decreasing the value of the re-projection error. We had assumed the Pinhole model for the camera, as it easily works for most off-the-shelf webcams.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The unknown parameters are fx and fy (camera focal lengths) and (cx,cy) which are the optical centers expressed in pixels coordinates.

These intrinsic parameters were used to rectify the images, and this was further passed to the stereo_proc pipeline to create a disparity map using the stereo-block matching techniques. This algorithm requires manual tuning of parameters which control how much it should difference in disparity is necessary for the algorithm to consider it as a pixel of different depth. This adjustment was very difficult, and we tried out many different open-source ROS packages in order to achieve better disparity map generation. Researching further, we found out that the video stream from the two cameras should be frame wise time synchronized, or else the disparity map generation fails due to poor feature matching performance. Better cameras allow for external triggering of the webcams, to restart the stream, but these webcams did not support the feature. We tried creating a driver for this using OpenCV, but it caused a major drop in the frame-rate and still the images were not quite synchronized due to different machine clocks.

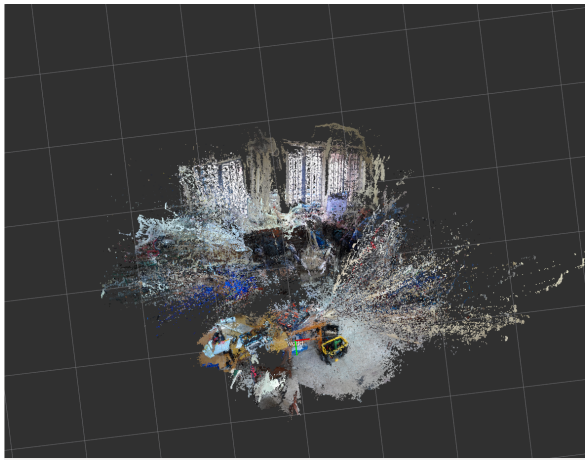


Fig. 5: Indoor 3D Mapping using REMODE and ORB-SLAM

Since we could not create a usable depth-map from this method, we switched to using monocular camera for this depth mapping. We created a pipeline for this using ORB-SLAM and REMODE. The open-source REMODE repository does not support RGB-point cloud generation, and thus we modified both ORB-SLAM and REMODE for handling RGB-pointclouds. We used the intrinsic parameters calculated earlier for this camera, and performed this implementation.

The results were great for an indoor room, where we could view an object from multiple perspectives and since REMODE is a probabilistic framework, we could compute with more certainty the depth of each particle in the object. The way REMODE works, is that for an area of samples in the image, it estimates the uncertainty and make sure it converges to a particular value. When we tried this out for a corridor in the Helicopter Lab, it failed. We made use of ORB-SLAM which was working fine, provided the camera motion were slow enough, and we avoided patches of clear white walls, which tricked the feature mapping present in ORB-SLAM. The attached loop closure for the complete round of the lab is attached here. Correspondingly, the 3D map of the entire area is also attached. You can see that in areas where there were not enough features or different camera angles, the REMODE algorithm failed to compute depth map, and there are voids left in the map.

Finally, we turned to using the ZED camera. This is a RGB-D camera which uses can give us per-pixel depth information along with the RGB information as well. This can work at 1280x760 pixels and generate a depth map at approximately 30 FPS on a CUDA enabled GPU. We applied ORB-SLAM on the input from the ZED camera, and passed this to Octomap, which basically combined thse values from this to create a 3D map of the entire space. This worked well enough, but did not have a lot of visual appeal.

Finally, we used the library RTAB-Map to do this. It was a SLAM library which was capable of performing loop closures, as well as post processing operations such as mesh smoothing and eliminating false odometry information.

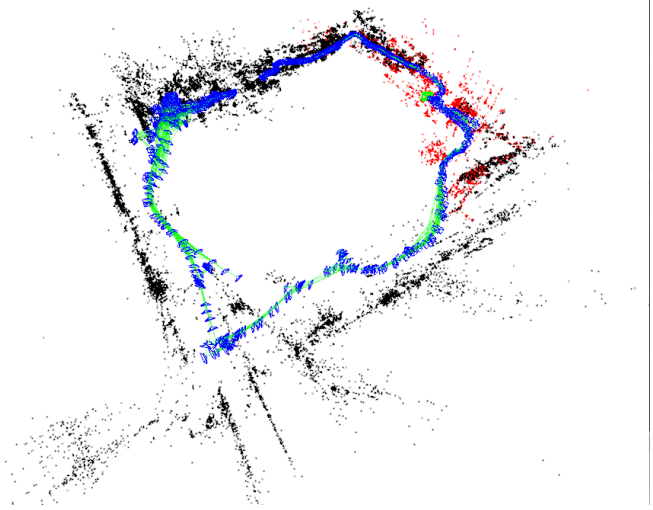
VI. Realtime onboard performance using RTAB-Map

This section needs to be completed once we perform this on a real drone.

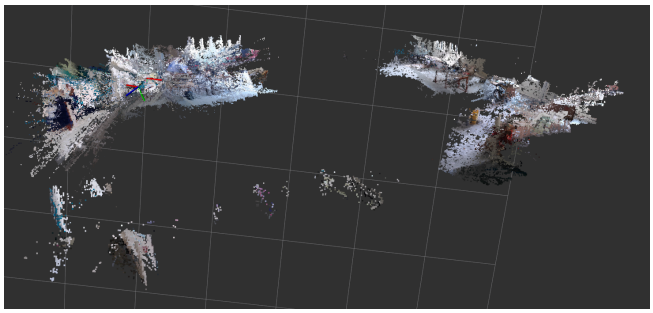
VII. Further scope

This mapping procedure can be used to autonomous exploration of unknown environments, where the drone can be attitude stabilized using an external controller. The drone can also be made capable of performing guided way point navigation with inputs provided from an external operator.

Progressing with these implementations, one can perform 3D SLAM of entire buildings and areas which can then



(a) ORB-SLAM performing loop closure



(b) REMODE fails with feature-less walls

Fig. 6: REMODE & ORB-SLAM tested in Helicopter Lab

be used to perform navigation and even find objects using advanced 3D mesh segmentation techniques.

REFERENCES

[1] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014.

[2] Armin Hornung et al. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: *Autonomous Robots* (2013). Software available at http://octomap.github.com. DOI: 10.1007/s10514-012-9321-0. URL: http://octomap.github.com.

[3] M. Labbfffdfffd and F. Michaud. "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation". In: *IEEE Transactions on Robotics* 29.3 (June 2013), pp. 734–745. ISSN: 1552-3098. DOI: 10.1109/TRO.2013.2242375.

[4] M. Labbfffdfffd and F. Michaud. "Online global loop closure detection for large-scale multi-session graph-based SLAM". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 2661–2666. DOI: 10.1109/IROS.2014.6942926.

[5] Raúl Mur-Artal and Juan D. Tardós. "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[6] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014.