RIT-ADS (Autonomous Driving Sets)

Felix Blanco, Rodney Sanchez, Xavier Tarr

Advanced Robotics 784

Rochester Institute of Technology

Electrical Engineering Department: Spring 2019

***Abstract* – With only a limited number of high quality datasets, such as the KITTI Dataset, it is often times hard to perform “robust” algorithm tests for slam, odometry, and computer vision. By adding another high quality dataset with a similar format to that of the KITTI Dataset, algorithm developers can perform cross dataset verification and make their algorithms that much more robust. A comparable dataset was created to KITI’s dataset that included both indoor and outdoor components of RIT’s campus.**

***Index Terms – KITTI Dataset, HUSKY, Lidar, Image Processing, Stereo Imaging***

I. Introduction

The KITTI Dataset was created on a Volkswagen Passat containing a large number of sensors that collect synchronized data points at a 10 Hz frequency. The data from these sensors includes grayscale stereo sequences, color stereo sequences, 3D Velodyne point clouds, 3D GPS/IMU data, calibration data, and 3D object labels [1].

To add more variation between the KITTI Dataset and the RIT’s Dataset, the HUSKY robot by Clearpath was utilized. HUSKY currently has an onboard a SICK 2D Lidar sensor, SMART6 GPS module, OpenIMU sensor, and high resolution quadrature encoders with 78,000 pulses/m for near perfect odometry [12]. Two Flir Fle3 3.0 USB cameras were added to HUSKY. Each camera has a resolution of 1028x1024 pixels, 120 degree view, and 60 Hertz capture speed. Images from these cameras are combined to create stereo image sequences similar to that of KITTI’s dataset.

This paper is divided into four sections. Section II describes the problem statement. Section III describes the setup, how the experiment was conducted and what data was collected. Finally, Section IV presents the results with a small description on the possible avenues open for future exploration.

II. Problem Statement

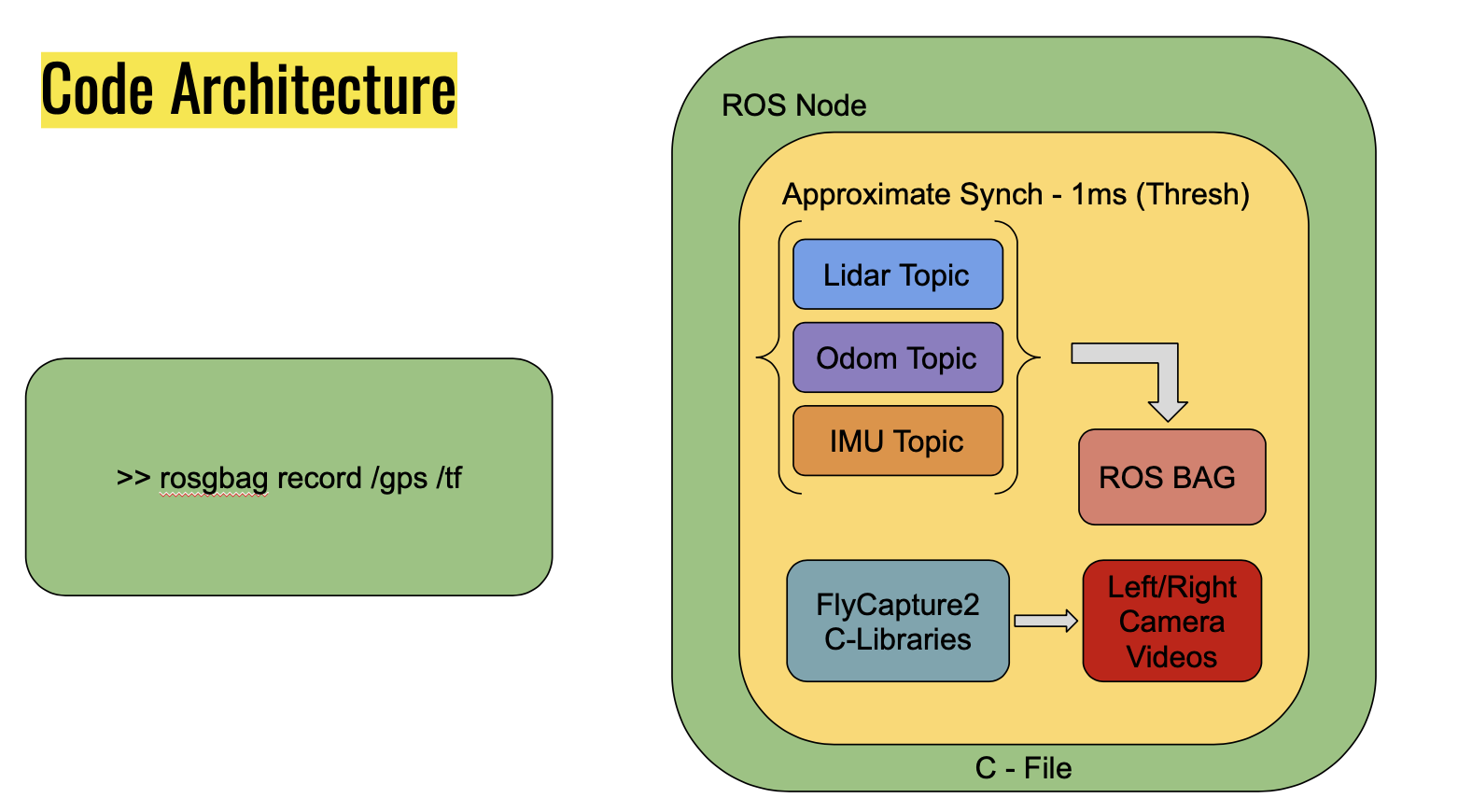
The KITTI dataset is used for developing and testing localization, and mapping algorithms geared towards autonomous 2dof vehicles. The KITTY dataset includes a multitude of sensors, all of which are widely used in autonomous vehicles today. However, the dataset is created on a specific platform, a Volkswagen passat B6. Most of the data gathered with this setup includes roads, and highways in optimally lit driving conditions. As a result there is no data for medium speed, close quarter environments, such as small roads, parking structures, or alleyways. Additionally environments that pose troublesome conditions to contemporary computer vision systems, such as windows, and glass structures are also ultimately left out. If vehicles are to become fully autonomous, they must be able to maneuver in these environments. By using HUSKY, close quarter environments such as college campus quads and pathways, can be virtually brought to algorithm developers. HUSKY and its multitude of ROS drivers, allow data to be saved in the easy to use, and compact “ROS bag” file type. The “ROS bag” file type helps by seamless integrating into the ROS ecosystem, that many robotics algorithm developers use today. New data will help autonomous vehicle algorithms become more robust in close quarter environments such as college campuses and parking structures. Eventually leading to smarter, and safer autonomous vehicles.

III. Setup

KITTI was created through the use of 3 cameras a lidar, GPS, autoencoder and a moving platform.[1]. Kitty set the rate of all sensors to capture at the same frequency between 10 and 100 hertzs.[1] Furthermore the location and representation of a 3D object in space will be recreated as the platform passes by.[6] Similarly, the auto and encoders and gps will be used in order to assure that error minimization can be kept between both systems. Husky will be the platform used for data capture and ROS will be used in for a control method by utilizing the rosbag operation[17] .The system will be operated in large part of the RIT campus going through landmarks such as the pi qudrant, Erdle Commons, and the MABL Lab. During the creation of the RIT dataset civilians will actively be removed from the environment, something not done in the KITTY dataset.

HUSKY is manually controlled by the dataset collector through the use of a simple Logitech game controller. The user simply moves the left joystick towards the desired direction of travel while holding down either the ‘A’(slow speed), or ‘X’(fast speed) buttons. As the joystick moves farther forward HUSKY’s speed increases, and the same is done in the reverse direction.

As mentioned before HUSKY is now equipped with two 1028x1024 Fle3 cameras, by Flir. As well as a 2D Lidar point clouds, GPS, IMU, and wheel encoders for odometry. Data from these sensors is gathered in a synchronized fashion.



**Code Overview**

Data Synchronization

HUSKY provides an easy to use ROS interface where it broadcasts 2D Lidar, GPS, IMU, and Odometry data. Piggybacking on top of the ROS ecosystem a simple C++ file is created that subscribes to 3 data topics, IMU, Odometry, and 2D Lidar. The ‘Approximate Synchronizer’ class of synchronization tools is used to synchronize all three message topics. This allows for a ROS callback function to be executed only when the timestamps of all three topics does not exceed 1 millisecond. Essentially meaning that the callback is called only if all incoming message timestamps are less than 1 millisecond apart.

In our ROS callback, our code simply snaps a picture of the environment from both Fle3 cameras using the FlyCapture2 C++ API libraries, and appends those pictures to an array of images for the current dataset. In addition the synchronized ROS messages are all saved to a bag file.

Stereo Camera Setup and Calibration

As stated before the FlyCapture2 C++ API is used to control camera setup and image capturing of the Fle3 cameras. The required libraries can be downloaded from <https://www.flir.com/products/flycapture-sdk> .

Calibration of the cameras themselves was done using the FlyCapture2 GUI. Making sure both cameras have the same exposure, brightness, and hue is critical for creating high quality stereo images.

Additionally, 100 images of a standard calibration checkerboard were collected indoors and OpenCV was used to create calibration files for each camera. These calibration file are used for removing lense distortion found on each camera. The file type for each calibration file is a “pickled” numpy array that holds the transformation necessary for removing lense distortion with OpenCV.

Two 3D printed physical mounts were created to secure the cameras in place, as to ensure the cameras do not rotate during data collection. Having very fixed cameras is extremely important for stereo imaging, because any rotation or motion of the cameras from there calibrated positions, will require recalibration.

Collecting GPS and HUSKY Transforms

Unfortunately, synchronizing HUSKY’s GPS ROS topics with all the other sensors data would have decreased our systems sample time from 10Hz to 5Hz. To work around this the ROS ecosystem came into play. By using the command line “rosbag record” command, we were able to capture GPS and HUSKY transform data asynchronously to that of the LIDAR, IMU, and Odometry data. This was a better method because GPS is often times slow and used only as a double check on odometry. Transform data is useful to have for visualizing Lidar data points in software such as rviz, but does not necessarily have to be synchronized.

Programs and Scripts

Two C++ programs were created for this project. The first of which is called “Dataset\_test”. This program is the main executable that is loaded into our ROS workspace. When run the executable starts a ROS node, subscribes to HUSKY’s Lidar, Odometry, and IMU topics, synchronizes them, and finally captures video footage of its environment. ROS topic messages are saved to a bag file, while images from both the left, and right cameras are saved as video files.

The next C++ file when compiled creates an executable called “MultiCamSaveToAvi”. This code was based on a FlyCapture2 SDK example for recording video from a single camera and saving it to disk as an AVI file. This program is used for calibrating the two cameras. It takes 100 pictures, with a keyboard interrupt in between each captured picture. Finally, it saves all 100 images as two AVI video files.

As a method of providing a baseline value for our datasets new stereo cameras, a python script was created to generate colored stereo disparity maps of the environment based on the indoor camera calibration files we gathered. The python file uses OpenCV’s Stereo Rectification, Stereo\_Create, and WLS post filtration classes to create depth maps of our datasets. This is provided in the dataset as well.

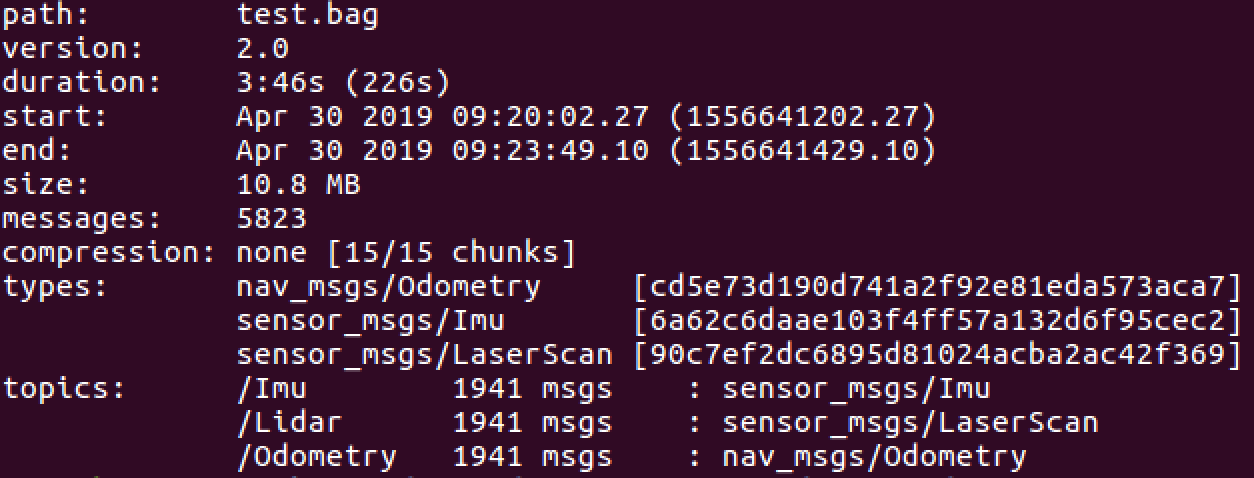
Finally, image stitching was considered as well [16]. Since we will have 2, 120 degree viewing angle cameras, the visuals between both will have to be combined. This paper serves as a solis reference on how we went about this.

IV. results

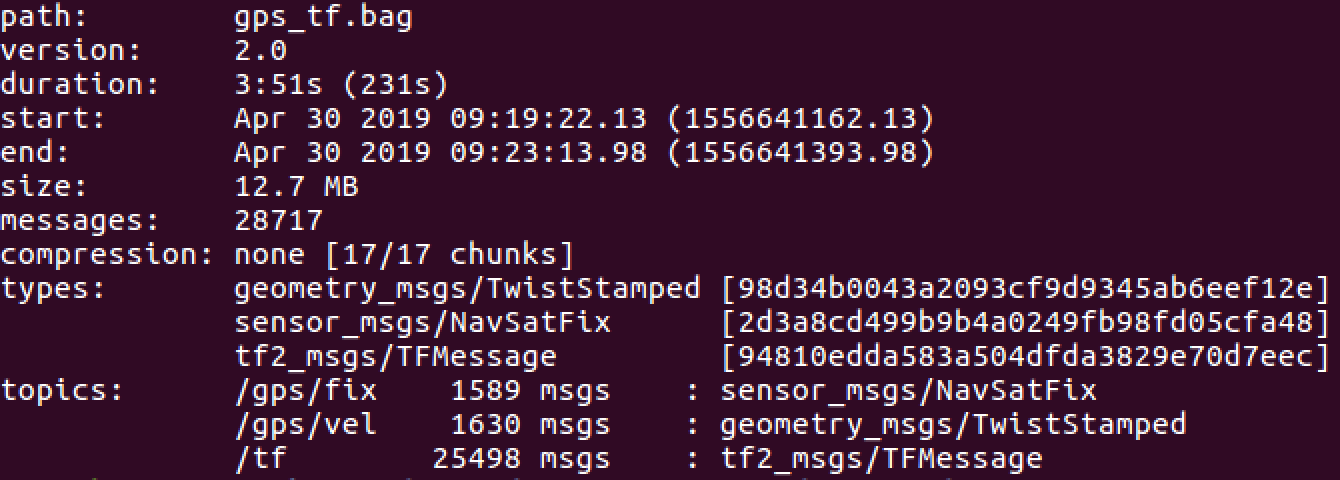
We have generated two high quality datasets of two portions of RIT’s campus. RIT’s PI Quad, as well as RIT’s Infinity Quad. We have also created an indoor dataset of our Robotics research lab, however, this dataset does not contain GPS, or transform data.

These datasets contain more than 1500 data points of each sensor type. For each dataset the same number of camera frames, as Lidar point clouds are found. In the case of the Infinity quad data set, 1941 lidar point clouds were taken and as a result 1941 stereo frames were taken. These frames were used to create 1941 depth frames of the dataset scene.

Infinity - Quad



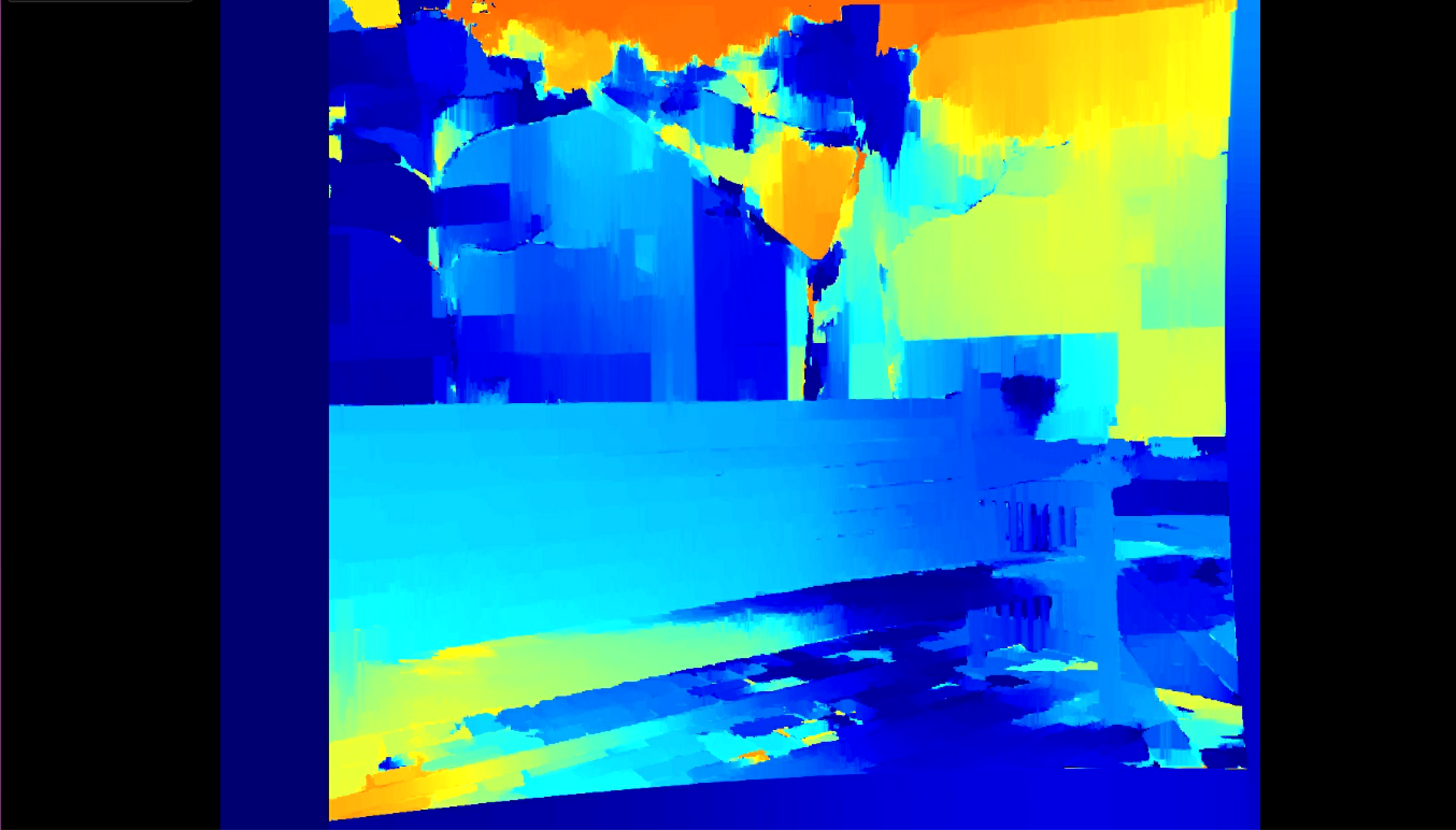
**Lidar, IMU, and Odometry Data Points**



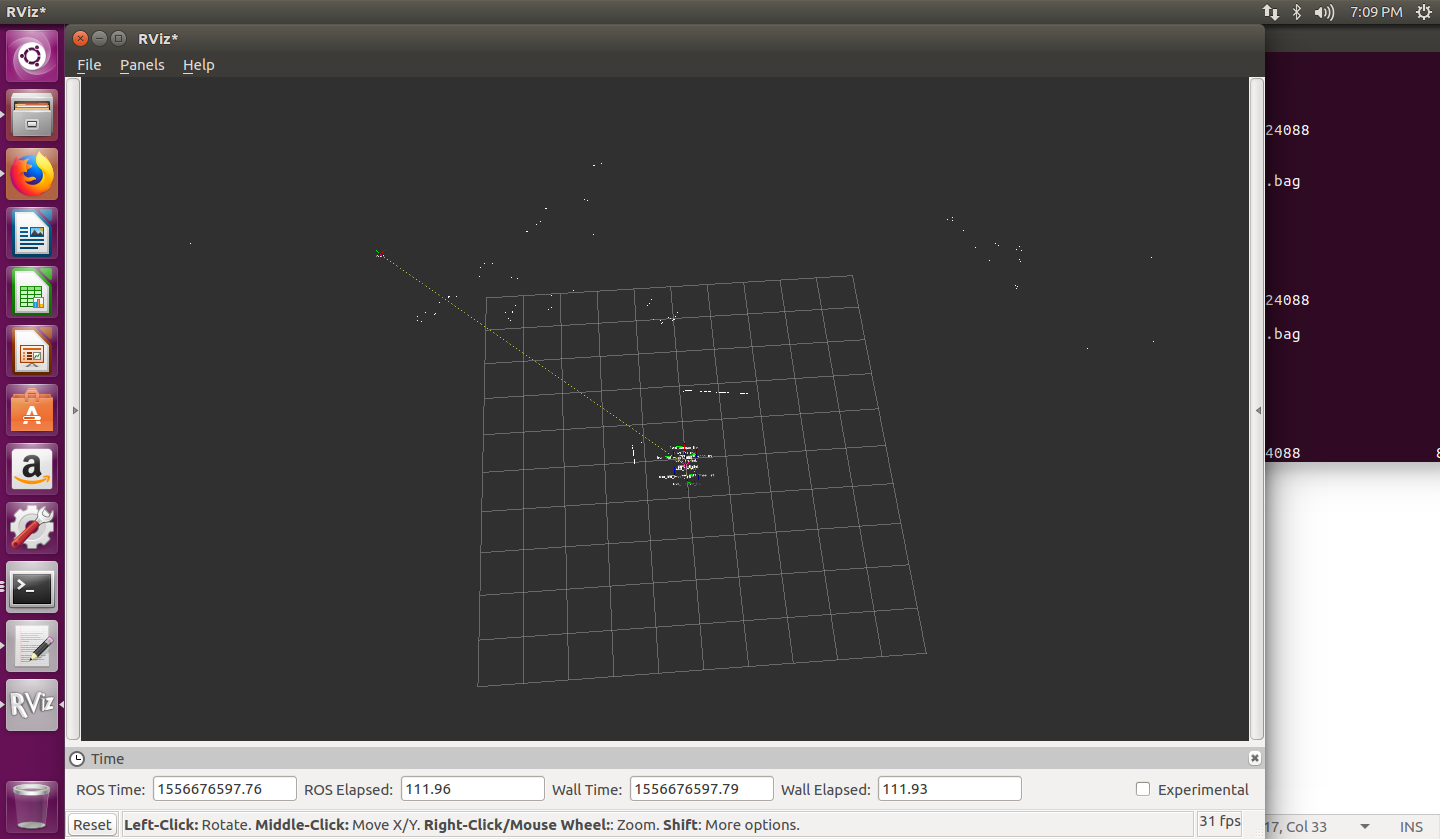
**GPS and Transform Data Points**



**Color Stereo Images**

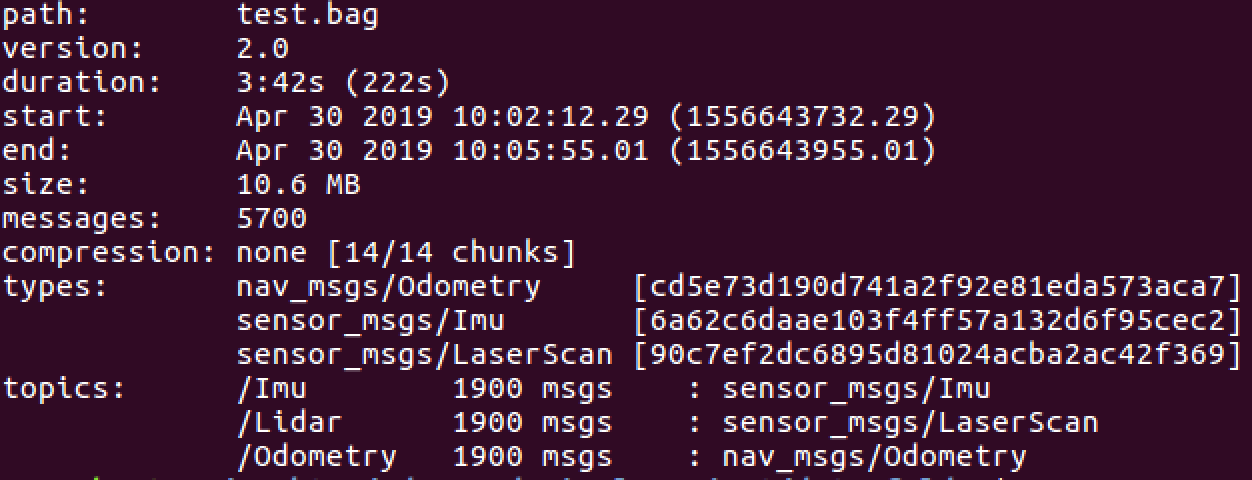


**Simple Baseline Stereo Image**

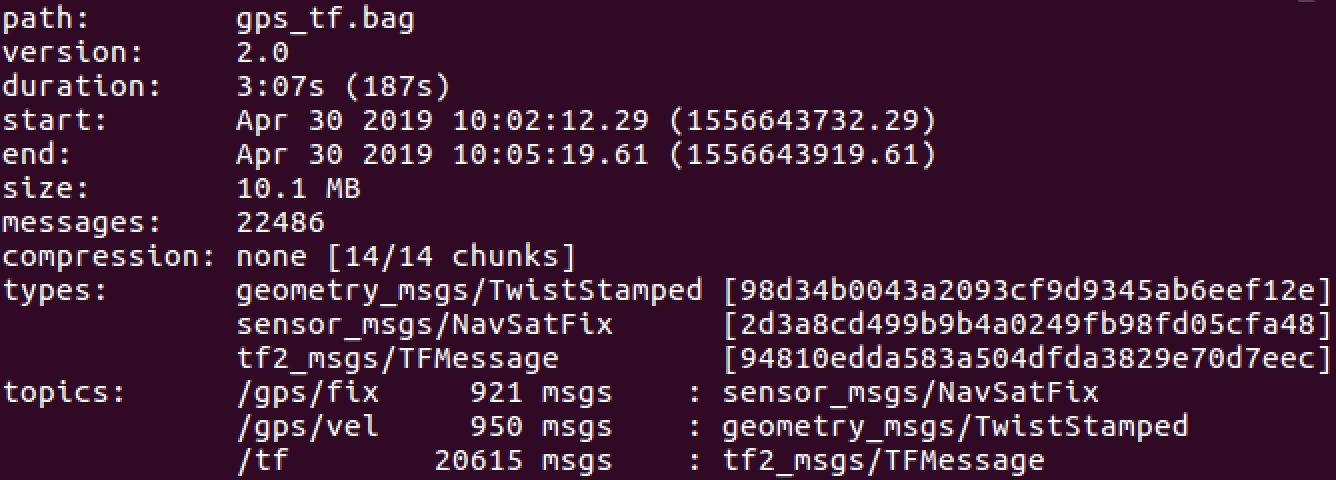


**Lidar Visualization in RVIZ**

Pi - Quad



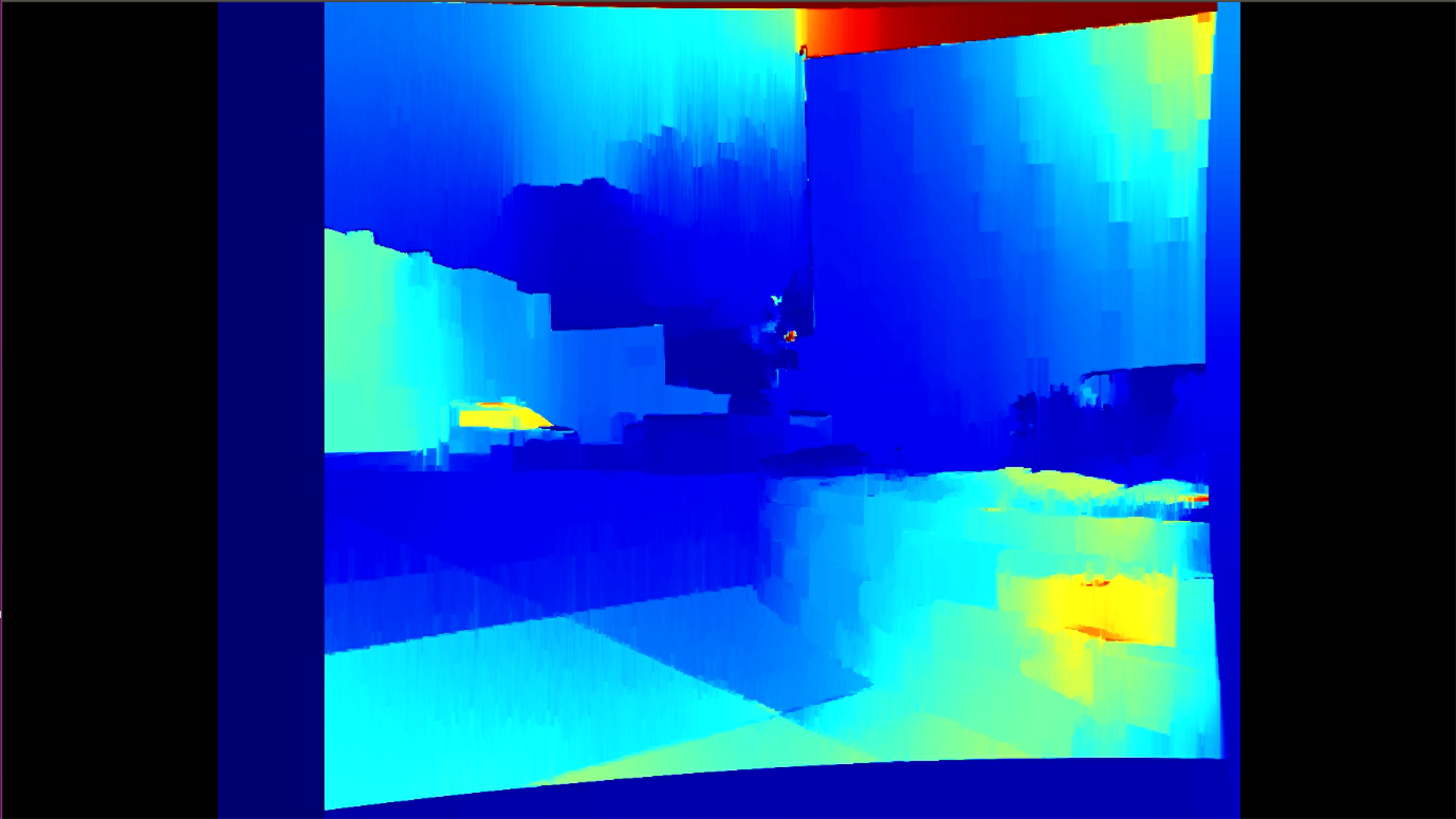
**Lidar, IMU, and Odometry Data Points**



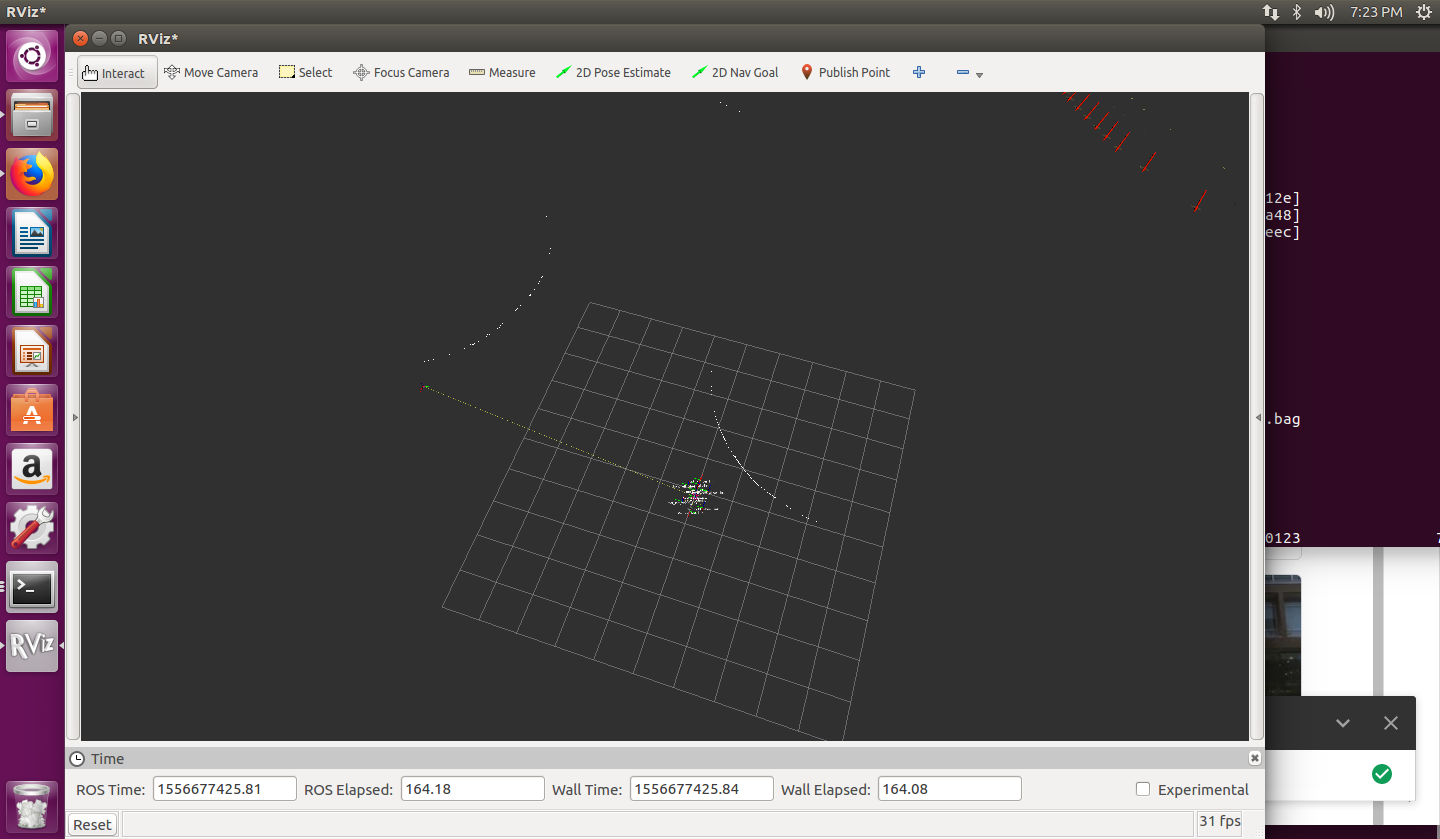
**GPS and Transform Data Points**



**Color Stereo Images**



**Simple Baseline Stereo Image**

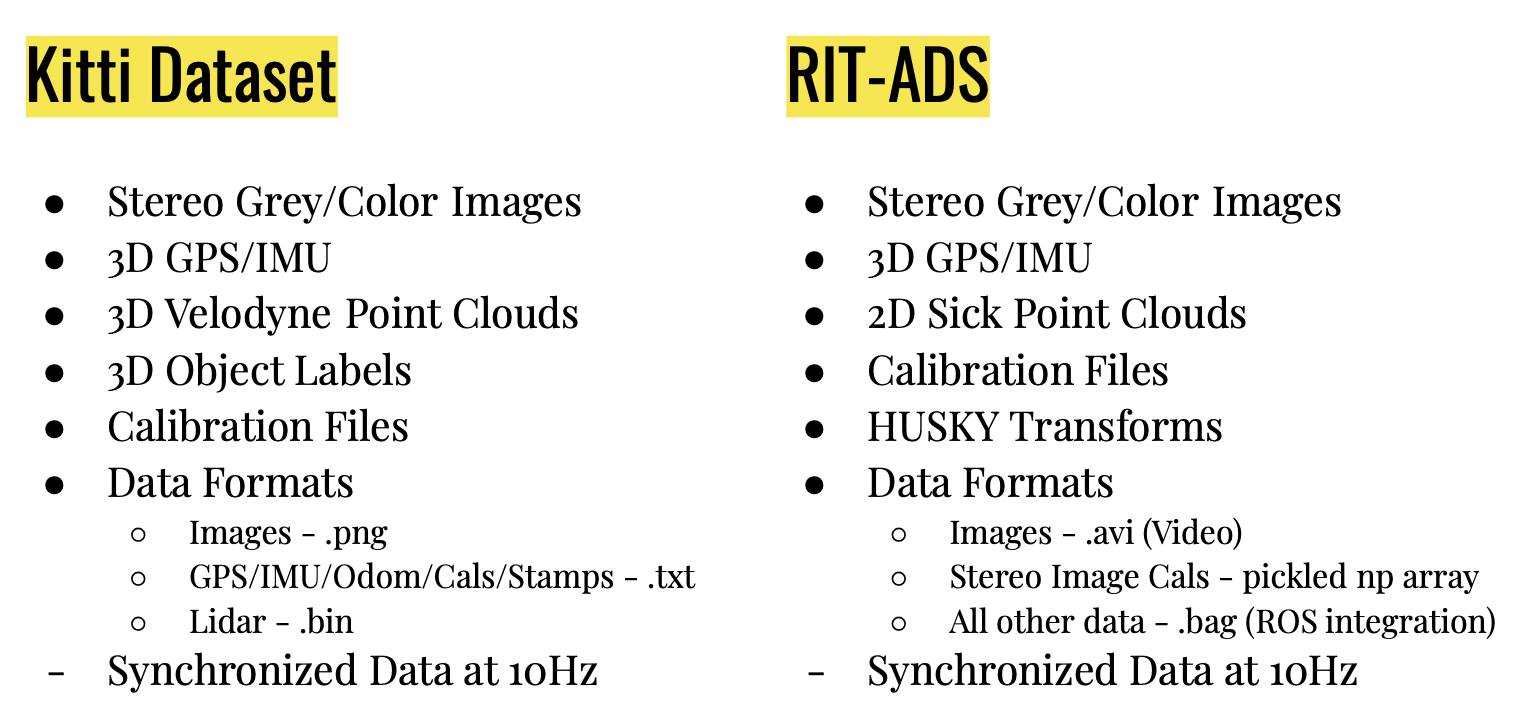


**Lidar Visualization in RVIZ**

As can be seen from the images above lidar, imu, gps, and stereo vision were all successfully synchronized and saved in easy to use file types. We created two datasets which in total contain 3841 lidar, imu, odometry, and stereo sequences. Both datasets were taken by maneuvering around small areas of campus that had medium to high activity. Lidar was very easy to visualize in RVIZ due to the saving of HUSKY transforms.

Stereo images were created using OpenCV tutorials on stereo imaging. A simple WLS filter was implemented to clean the stereo images to produce a more high quality image that did not contain any depth holes.

The frequency of synchronized data points matched that of the KITTI dataset, which was 10Hz. We were able to effectively take the HUSKY platform and create an easy to use data collection system out of it.



**Dataset Comparison**

Future work would include integrating a 3D Lidar onto HUSKY instead of a 2D Lidar. Additionally 3D object labels could be added in order to make the dataset even more robust.

Reflective surface datasets should also be created to help algorithm developers test using stereo imaging for depth perception on these troublesome areas.

Adding better mounts for our stereo cameras would be an ideal path for future work. Currently 3D printed parts are used for mounting the stereo cameras to HUSKY. These parts are susceptible to vibrations, and thus the cameras may possibly move during data collection. To mitigate this metallic parts should be machined for a more rigid stereo camera holder.

Stereo camera calibration is critical. Thus, a more robust stereo camera calibration must be done to provide more high quality stereo images.

Lastly, because Rochester is located in an area that sees large changes in weather, it would be ideal to have our data collection platform collect data during snowy, and rainy conditions. Collecting data during multiple seasons would help develop more robust algorithms. Autonomous vehicles will be used in a variety of weather conditions, thus data of these conditions is required to create the most optimal solutions.

References

1. Andreas Geiger, “Vision meets Robotics: The KITTI Dataset”.
2. G. Singh, “Acquiring semantics induced topology in urban environments”.
3. R. Paul, “FAB-MAP 3D: Topological mapping with spatial and visual appearance”.
4. C. Wojek, “Monocular visual scene understanding: Understanding multi-object traffic scenes”.
5. A. Geiger, “A generative model for 3d urban scene understanding from movable platforms”.
6. A. Geiger, “Joint 3d estimation of objects and scene layout”.
7. M. A. Brubaker, “Lost! leveraging the crowd for probabilistic visual self-localization”.
8. A. Geiger, “A toolbox for automatic calibration of range and camera sensors using a single shot”.
9. M. Goebl, “A real-time-capable hard- and software architecture for joint image and knowledge processing in cognitive automobiles”.
10. Xinyu Huang, “The ApolloScape Open Dataset for Autonomous Driving and its Application”.
11. Wilfried Elmenreich, “An Introduction to Sensor Fusion”.
12. David Portugal, “An autonomous all terrain robotic system for field demining missions”
13. Mary B. Alatise, “Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter”.
14. Mohamed M. Atia, “Integrated Indoor Navigation System for Ground Vehicles with Automatic 3D Alignment and Position Initialization”.
15. W.R.C.B.S. Welikala, “Multi Sensor Fusion for Position and Indoor Navigation”.
16. Chung-Ching Lin, “Adaptive As-Natural-As-Possible Image Stitching”.
17. Fan Yang, “Husky: Towards a More Efficient and Expressive Distributed Computing Framework”.
18. Jiang Dong, “Advances in Multi-Sensor Data Fusion: Algorithms and Applications”.
19. Moshe Kam, “Sensor Fusion for Mobile Robot Navigation”.
20. Yan Pei, “An Elementary Introduction to Kalman Filtering”.
21. <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html>