UNIVERSITY of DAYTON

VISION LAB
Center of Excellence

AIR FORCE RESEARCH LABORATORY

Mobile Sensor Lab

Brandon Hampshire and Vijayan K. Asari

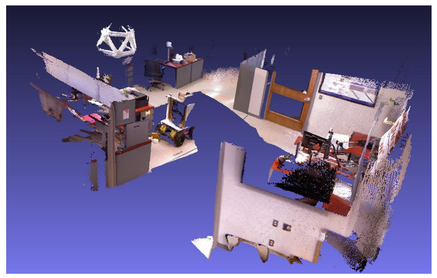
Introduction

The Vision Lab has many projects that involves data collection from various sensors. These sensors can vary by platform and programming language. In order to centralize data collection, the Robot Operating System (ROS) is applied to the Clearpath Husky robot. The Husky is made for multi-terrain transport and can be modified to carry multiple sensors. The Robot Operating System is not an operating system but is a network centralized library. ROS provides background services and different language libraries that allow sensors and languages to communicate to a common location. In addition, the libraries can let the user to receive data from the central network thus allowing multiple cross language platform communication. An application of these components can assist in data collection for environment 3D reconstruction. Environment 3D reconstruction requires depth imagery, RGB imagery, and orientation of the camera. The Husky provides estimated orientation to ROS and ROS supports the Microsoft Kinect. With open source coding, the Husky can be coded to collect the necessary data for 3D reconstruction.

Hardware

ROS and Data Collection

In order to collect the necessary data, the robot is controlled by a computer with a specialized Linux Ubuntu OS with ROS built in for controlling the Husky. Installing necessary drivers and scripts for the Kinect are available in the documentation site for ROS. With the environment set, the only challenge is to create a data recording code from open sourcing the code used to 3D recontruct an environment shown below.



The original code reads from a single file in this format below:

```
/**
* Format is:
* int64_t: Orientation/timeStamp
* int32_t: depthSize
* int32_t: imageSize
* depthSize * unsigned char: depth_compress_buf
* imageSize * unsigned char: encodedImage->data.ptr
*/
```

The new code writes from a single file for each frame in this format below:

```
snprintf(filename,sizeof(filename),"rosDataLogImage%d.lkg",frameCount);
FILE * logFile = fopen(filename,"wb+");
fwrite(&timeStamp, sizeof(uint64_t), 1,logFile);
fwrite(&depthSize, sizeof(int32_t), 1, logFile);
fwrite(&imageSize, sizeof(int32_t), 1, logFile);
 fwrite(depth_compress_buf, depthSize, 1,logFile);
fwrite(imageBuffer, imageSize, 1,logFile);
fwrite(&currentOrientationAngle,sizeof(float), 1,orientationLogFile);
```

What were the issues? Optimization!

The nature that ROS code retrieves and sends data is limited by cycles. The rate that the code was able to record error free data was limited to 10 frames per a second. Recording the data into a single file just like the original code was causing severe time delays and hook ups late into recording. By changing the code to record individual files, the rate of recording became consistent and only required changing the original code to read in the files into a single file.

The following changes were made to the callback functions for receiving data from ROS in order to make the system record 25 frames per a second.

- The data comes as a vector so instead of reading into array, they were assigned pointers
- Set the cycle rate of the code to 25 Hz

Applications

The applications for ROS and the Mobile Sensor Lab are only limited by the vast hardware support of ROS.

UNIVERSITY of DAYTON

Vision Lab
Kettering Laboratories
300 College Park
Dayton, OH 45469-0232

Center of Excellence for Computer Vision and
Wide Area Surveillance Research
visionlab.udayton.edu

Dr. Vijayan K. Asari
Director
(937) 229-4504
vasari1@udayton.edu