3D Indoor Mapping Using ROS and Microsoft Kinect Sensor

Chirag Shah and Srijal Poojari

***Abstract*—This project deals with the exploring the ROS framework for development of a robotic system with various sensors and actuators in order to understand the underlying concepts and to create a robot/quadcoptor capable of forming a 3D map of a given environment using a depth camera (Microsoft Kinect).**

***Index Terms*—ROS, Robot Operating System, 3D-Mapping, Microsoft Kinect Sensor**

I. Introduction

AUTONOMOUS robots operating in real life settings must be able to navigate in large, unstructured, dynamic and unknown spaces. To do so, they must build a map of their operating environment in order to localize itself in it, a problem known as Simultaneous localization and mapping (SLAM).

mds
October 30, 2017

II. Major Components of Our System

A. Robot Operating System

The Robot Operating System (ROS) is a framework for writing robot software. It is a set of utilities and libraries for implementing different kinds of functionality on robots. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. It is not a programming language.

A ROS system is comprised of a number of independent nodes, each of which communicates with the other nodes using a publish/subscribe messaging model. For example, a particular sensors driver might be implemented as a node, which publishes sensor data in a stream of messages. These messages could be consumed by any number of other nodes. Note that nodes in ROS do not have to be on the same system or even systems of the same architecture. This makes ROS really flexible and adaptable to the needs of the user. ROS is open source, maintained by many people. ROS starts with the ROS Master. The Master allows all other nodes to find and talk to each other.

1) roscore: roscore is a service that provides connection information to nodes so that they can transmit messages to one another. Every node connects to roscore at startup to register details of the message streams it publishes and the streams to which it wishes to subscribe. When a new node appears, roscore provides it with the information that it needs to form a direct peer-to-peer connection with other nodes publishing and subscribing to the same message topics. Every ROS system needs a running roscore, since without it, nodes cannot find other nodes. With knowledge of the location of roscore on the network, nodes register themselves at startup with roscore and then query roscore to find other nodes and data streams by name. Each ROS node tells roscore which messages it provides and which it would like to subscribe to. roscore then provides the addresses of the relevant message producers and consumers.

2) Nodes: A node is a process that performs computation. Nodes are combined together into a graph and communicate with one another using streaming topics, RPC services, and the Parameter Server. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. For example, one node controls a laser range-finder, one Node controls the robot's wheel motors, one node performs localization, one node performs path planning, one node provide a graphical view of the system, and so on.

The use of nodes in ROS provides several benefits to the overall system. There is additional fault tolerance as crashes are isolated to individual nodes. Code complexity is reduced in comparison to monolithic systems. Implementation details are also well hidden as the nodes expose a minimal API to the rest of the graph and alternate implementations, even in other programming languages, can easily be substituted. A ROS node is written with the use of a ROS client library, such as roscpp or rospy.

3) Topics: A topic is a name for a stream of messages with a defined type. Topics implement a publish/subscribe communication mechanism, one of the more common ways to exchange data in a distributed system. Before nodes start to transmit data over topics, they must first announce, or advertise, both the topic name and the types of messages that are going to be sent. Then they can start to send, or publish, the actual data on the topic. Nodes that want to receive messages on a topic can subscribe to that topic by making a request to roscore. After subscribing, all messages on the topic are delivered to the node that made the request. In ROS, all messages on the same topic must be of the same data type.

B. Microsoft Kinect

Microsoft Kinect is a RGB-D camera. Kinect uses an RGB camera with depth sensor and infrared projector with a monochrome CMOS sensor. The Kinect functions by covering the room with a constant, predetermined pattern of infrared dots. The monochrome CMOS sensor is placed at an offset

relative to the IR transmitter, and the difference between the observed and expected IR dot positions is used to calculate the depth at each pixel. This gives a depth image of the surroundings. This depth image is super imposed with the RGB image to obtain a RGB-D image.

C. RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D 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 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 environnements are always respected. RTAB-Map can be used alone with a hand-held Kinect or stereo camera for 6DoF RGB-D mapping, or on a robot equipped with a laser rangefinder for 3DoF mapping.

D. Raspberry Pi

A roscore runs on the Raspberry Pi. This roscore is common to the laptop which is on the same network. Thus all the map/sensor data is available to the laptop over the WiFi network. The kinect sensor is connected to the Raspberry Pi. Mapping is performed on the Raspberry Pi while the map is visualized on the latptop.

III. Results

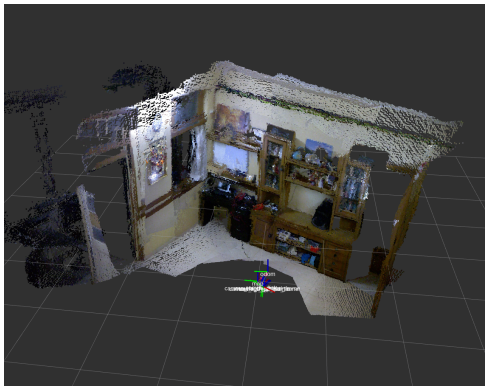

Fig. 1. Mapping results with laptop

IV. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

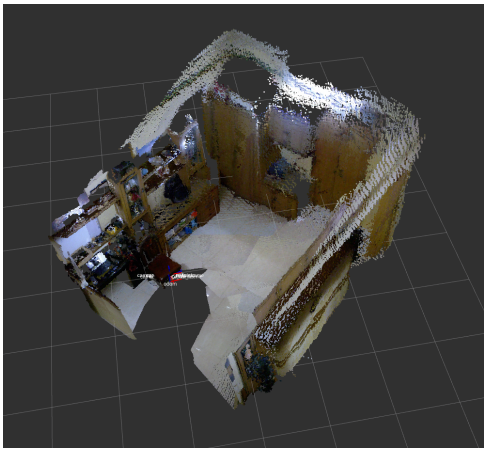

Fig. 2. Mapping results with laptop

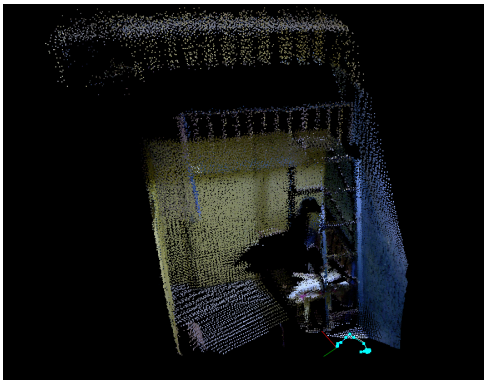

Fig. 3. Wireless Mapping with Raspberry Pi

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

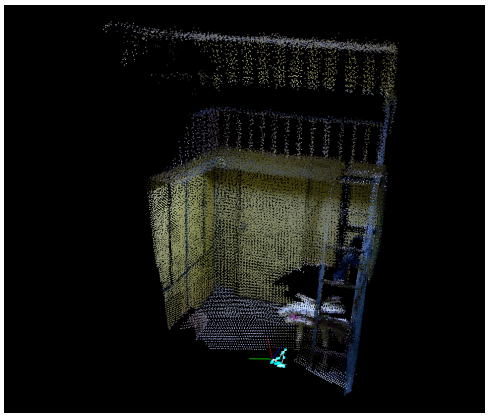

Fig. 4. Wireless Mapping with Raspberry Pi