Developing an Indoor Localization and Mapping System Using Microsoft Kinect and Raspberry Pi

Thomas Lavastida, Shuai Zheng, Wuyi Yu, Xin Li

Introduction

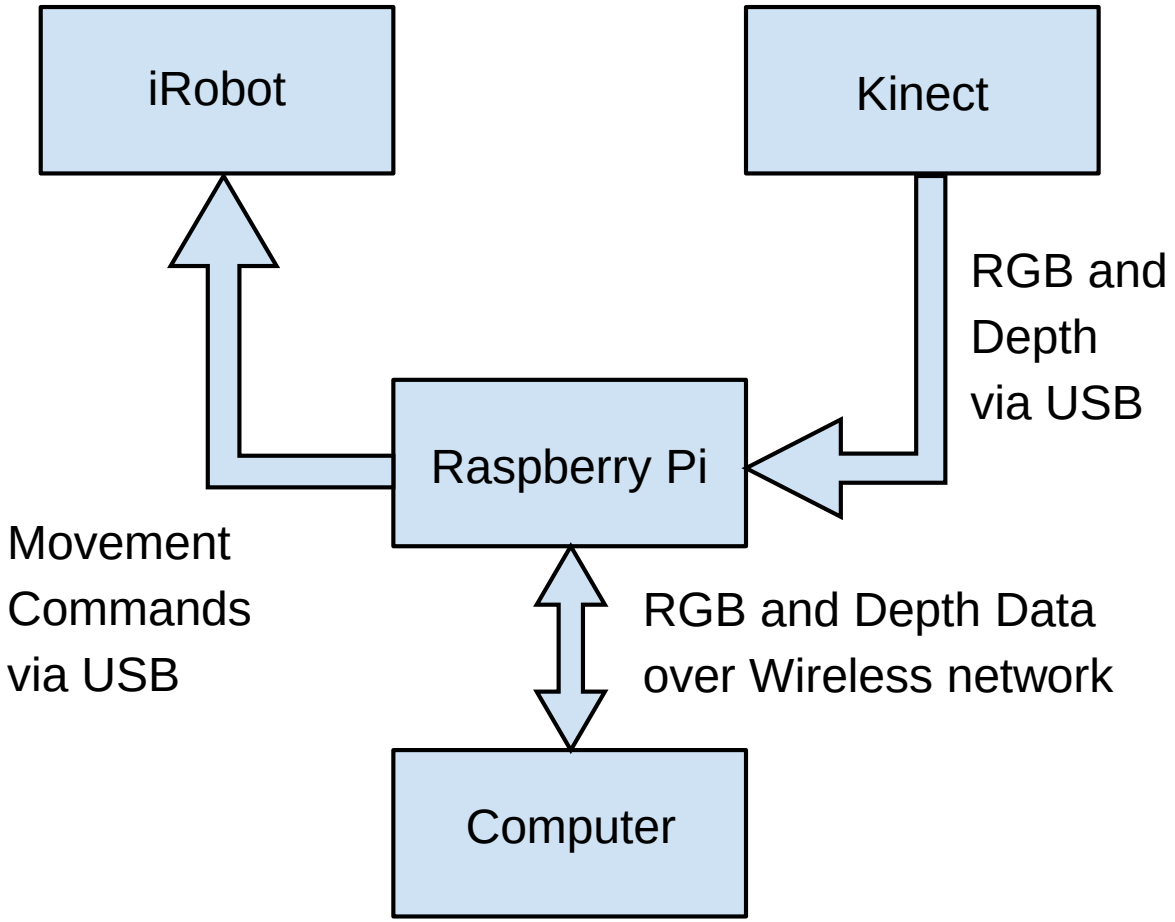
- Robot localization and mapping
  - Scan environment to produce accurate 3D reconstruction
  - Applications – autonomous navigation and unfamiliar environment reconstruction
- Goal
  - Develop a robot that can be used to evaluate localization and mapping techniques
- Requirements
  - System must be mobile
  - Must be able to collect data from Kinect Sensor
  - Efficient

Hardware

- Microsoft Kinect
  - RGB Camera and Infrared Depth Sensor
- Raspberry Pi
  - Inexpensive and small
  - GNU/Linux operating system
  - 700 MHz ARM processor
- iRobot Create 2
  - Programmable robot based on Roomba robotic vacuum cleaner

System Design

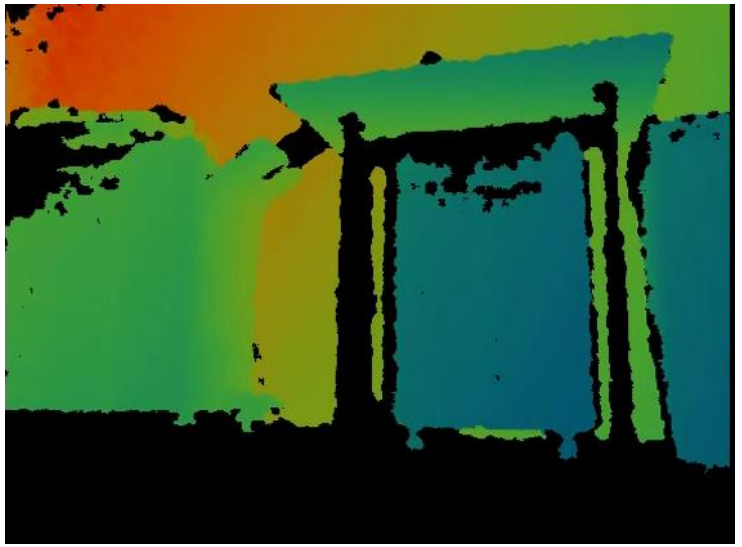
- Raspberry Pi is portable but not enough computation power for localization and mapping – use wireless network to send collected data to a more powerful computer
- Raspberry Pi acts as hub between other components in system
  - Collects data from Kinect
  - Sends commands to iRobot



Software

- Libraries/API's
  - Libfreenect – Open source driver and libraries to use Kinect on GNU/Linux operating systems
  - TCP/IP – Connection-based protocol for reliable communication over the internet
  - Zlib – Open source compression library
  - PySerial – Allows for communication over serial port using Python
- Programs
  - Server/Client programs for Kinect data transfer – C/C++
  - iRobot control program - Python

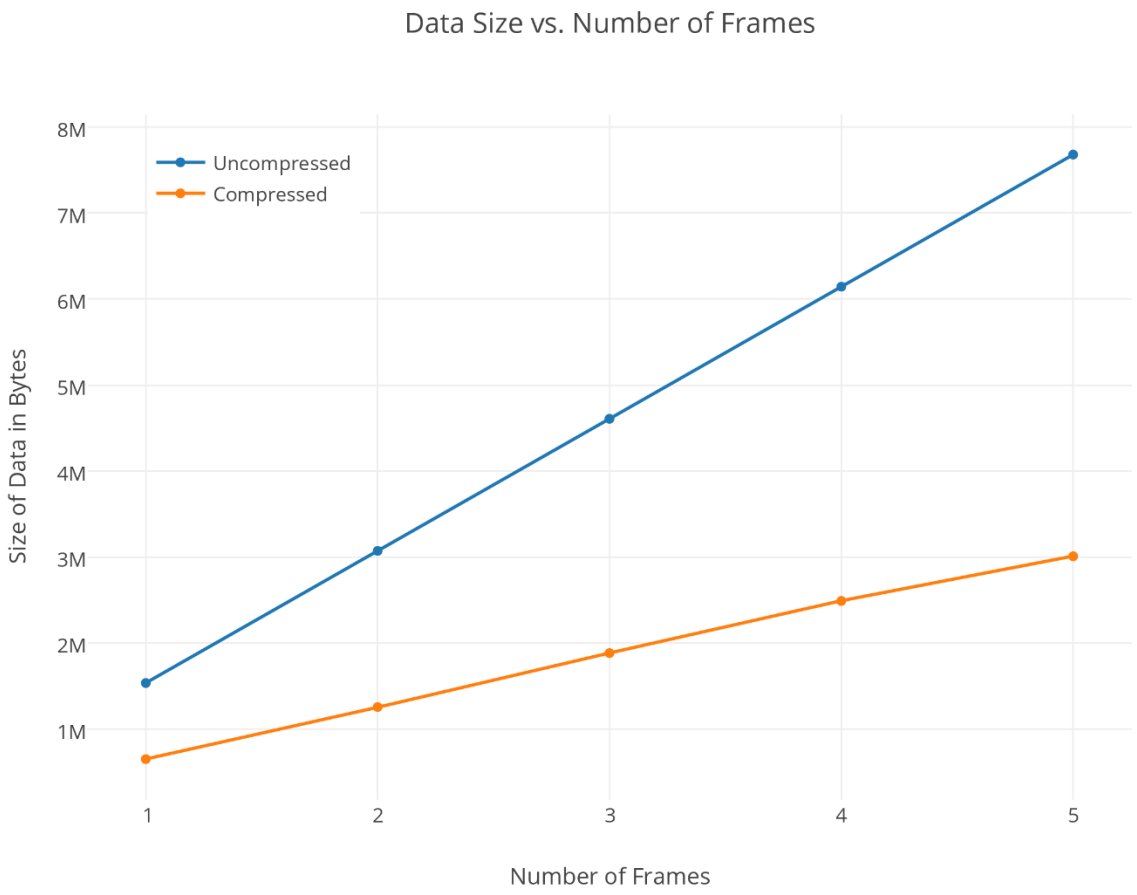
Results

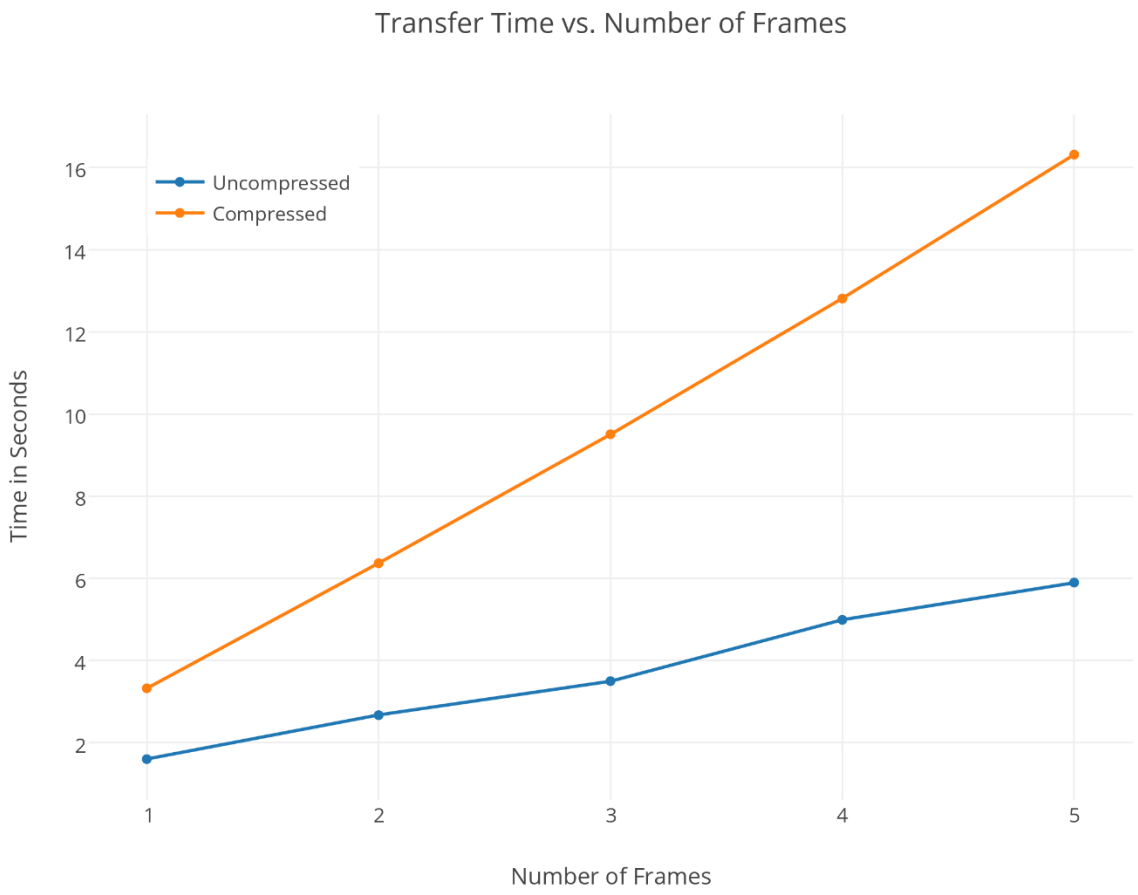


Performance Issues

- System can send one set of RGB and depth frames in 1.6 seconds
  - Very far way from ideal rate – Kinect produces data at 30 fps
  - Bottleneck in wireless transfer rate
- One set of RGB and depth frames use 1536000 Bytes (1.46 MB)
  - Attempt to improve time by reducing size of data to be transferred
  - Use data compression – Zlib provides functionality for this

Compression Test Results





Future Work

- Improve system so that it is more robust
  - Detect poor network conditions (heavy traffic) and adjust – store data that can't be immediately transferred in a temporary buffer
- Use the Raspberry Pi 2
  - 900 MHz quad-core processor vs. 700 MHz single core processor
- Implement pre-calculations for localization and mapping
  - Small computations may be done on the Pi before transferring data

Conclusions and Acknowledgements

- Accomplished basic task of transferring Kinect data over network
- Starting point for continued work
  - Possibility of continuing project in Senior Design class here at LSU

- Thanks to Dr. Li for providing guidance on this project
- Thanks to Shuai Zheng and Wuyi Yu for providing programming assistance

Questions?