Textured Mesh Reconstruction of Indoor Environments Using RGB-D Camera

Collin Boots

A THESIS

in

Robotics

Presented to the Faculties of University of Pennsylvania in Partial

Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2014

Dr. Daniel D. Lee
Supervisor of Thesis

Dr. Camillo J. Taylor
Graduate Group Chairperson

Abstract

Your abstract goes here... ...

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

As robots continue to be incorporated into human environments, the need for intelligent and high-speed reasoning about the objects around them increases dramatically. At the simplest level, mobile robots need to create a map of their environment for navigation. At a higher level, some robots need to recognize distinct objects in their environment, track object movement, and have some intuitive sense of object geometry that is easily stored and processed. Even more importantly, robots must be able to efficiently generate adaptable models of their environment, or worldview, from sensor data in real time. Many different methods for representing the world have been proposed and implemented, and they will be discussed below in more detail. Like the human brain, the robot should also be able to perform these low level functions with only minimal intervention from higher cognitive functions. Such technology also has potential uses beyond robotics. Applications may include easily modeling indoor environments for interior design concepts, generating 3D tours or maps for various buildings, or low resolution rough mapping of archaeological excavations.

This thesis is intended to work towards such a system based on RGB-D cameras, triangle meshes, and the powerful parallel computing capability modern Graphics Processing Units (GPUs) offer. RGB-D cameras like Microsoft's Kinect provide a great low cost solution for capturing 3D environments. Triangle meshes are efficient to store, simple to manipulate and refine, and very versatile. Meshes have the added benefit of being well suited to GPU hardware (which was originally designed for just that purpose).

1.2 Related Work

A great variety of methods have been applied to RGB-D camera data in an effort to construct a coherent worldview. Each has advantages and disadvantages.

1.2.1 Worldview Storage Models

Point Cloud Models The simplest approach to storing RGB-D data is as a raw point cloud. Each point is stored in a self contained data structure containing at least the point's position and color information, e.g. $P_i = \{pos_x, pos_y, pos_z, red, green, blue\}$. This approach allows a complete record of the raw data to be stored very easily, but the size of the data stored will grow very quickly, and simply storing the data linearly results in very slow queries.

Nearest neighbor approximations have recently become a popular approach for speeding queries on large point clouds[1].

Voxel Space Models The most robust and impressive

Point Cloud Offline Processing

1.2.2 Real-Time Plane Extraction

1.2.3 Additional Resources

Although this thesis focuses on detecting and representing only planar elements, the envisioned worldview generating pipeline would need several additional components and capabilities to be successful. Additionally, many data processing technologies exist that can improve the quality of the input data through more sophisticated filtering.

RGB-D Filtering

3D SLAM

Mesh Processing and Modification

1.3 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 will review some basic precepts of parallel algorithm design, along with specific optimization considerations for programming GPUs with CUDA. Chapter 3 will provide an overview of the system design as well as break down the pipeline into sub-modules to be explored in much more detail in Chapter 4. Finally, Chapters 5 and 6 will provide performance analysis, conclusions, and areas of potential improvement for future work.

TODO: Double Check that this hasn't changed

Chapter 2

Parallel Programming Paradigms

2.1 Principles of Parallel Programming

2.2 Parallel Algorithm Building Blocks

2.3 Programming with CUDA

2.3.1 CUDA GPU Architecture

2.3.2 Optimizing CUDA Code

Chapter 3

Problem Formulation and Approach

3.1 Problem Specification

3.2 High Level System Design

3.3 Plane Detection and Meshing Pipeline Design

Chapter 4

Implementation

4.1 RGBD Framework

4.2 Preprocessing

4.3 Plane Segmentation

4.4 Mesh Generation

Chapter 5

Performance Analysis

Chapter 6

Conclusions and Future Work

Bibliography

[1] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.