Conversion of RGBD Images to Textured Triangle Meshes with GPU

Dalton Banks, Collin Boots

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 important is being able to efficiently generate this environment from sensor data in real time. Like the human brain, the robot should also be able to perform these low level functions with only minimal intervention from higher cognitive functions.

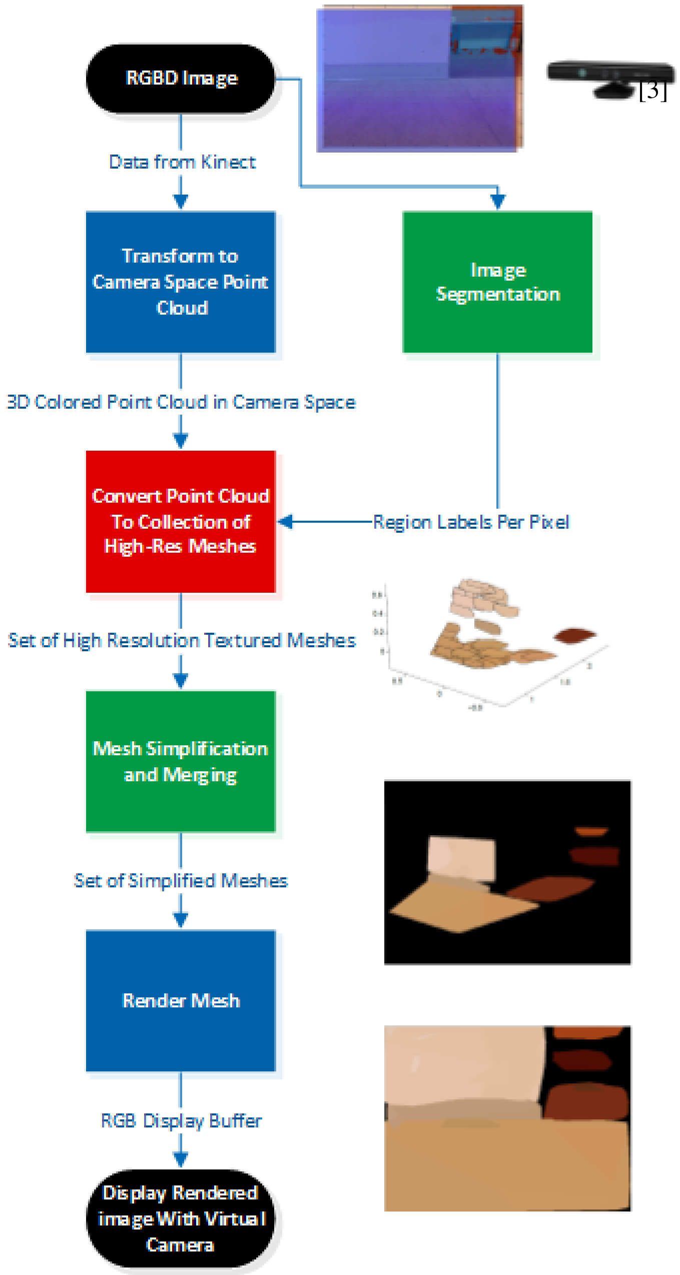
Previous work has demonstrated the diverse capabilities of depth cameras from generating highly accurate 3D surface models [2] to reliable 3D pose estimation [1, 3]. However, these efforts have collectively suffured from several limitations. Many algorithms attempt to store the generated environment as a RGB 3D point cloud (either pruned or complete), which is not easily adaptable to dynamic environments, requires exorbitant quantities of memory to store large environments, and provides no intuition to higher perception processes about distinct objects beyond a volumetric approximation. Other approaches have been able to store and merge the surface data more efficiently, but still regard the environement as a unified whole rather than discrete objects. These approaches may be sufficient for mapping static envirnoments, but provide only limited utility to more interactive robots and can actually hinder robots that reconfigure their surroundings (and hence invalidate part of their map).

Proposed Method

A new method for storing and manipulating environement maps is proposed using triangle meshes to store all geometries. By extracting meaningful geometry from the RGB-D in the form of triangle meshes, a large number of advantages can be realized.

1) The storage format of a trinagle mesh is very efficient, allowing for large and detailed maps to be stored in memory constrained systems like small mobile robots.
2) Mesh models provide a natural way of isolating distinct objects in the environment in a way that point clouds and surface maps cannot.
3) Meshes are easy to manipulate, modify, and manifest in real time using well established computer graphics technology and open the problem to the wealth of domain knowledge from the gaming graphics industry. By representing the robot's environment in a similar manner to a game world, future robots may be able to more easily leverage the gaming industry's experience in intelligent actor design.
4) By manipulating and moving meshes instead of erasing and redrawing portions of a discrete map or point cloud, map artifacts from more dynamic environements could be handled more gracefully and naturally.
5) Storing objects as distinct, flexible models provides a natural means of handling moving objects in lower level software, freeing higher level processes for reasonsing about object movement more abstractly.
6) Meshes provide an efficient and easy to process intuition of geometry to higher cognitive functions that may simply object recognition and manipulation tasks.
7) Meshes are very flexible in their ability to record complex geometries. By using a single storage format for all object types, many implementation efficiencies may be realized. Also, meshes provide a very straightforward tradeoff between simplicity and accuracy that can be situationally controlled.

Using an off the shelf RGB-D depth camera like the Kinect to accomplish all of these goals also brings with it all the benefits explored in previous work: hardware simplicity, minimal integration complexity, and low cost. This work will focus primarily on generation of high-accuracy triangle meshes for a variety of scene complexities and difficult edge cases such as touching objects. Time and technology permitting, some of the potential advantages of the approach listed above will be explored and demonstrated.



RGBD Image
Data from Kinect
Transform to Camera Space Point Cloud
Image Segmentation
3D Colored Point Cloud in Camera Space
Convert Point Cloud To Collection of High-Res Meshes
Region Labels Per Pixel
Set of High Resolution Textured Meshes
Mesh Simplification and Merging
Set of Simplified Meshes
Render Mesh
RGB Display Buffer
Display Rendered image With Virtual Camera

Fig. 1. Pipeline Overview

References

[1] Felix Endres et al. "An Evaluation of the RGB-D SLAM System". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* (2012).

[2] Richard A. Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking." In: http://redwood.berkeley.edu/bruno/3DFM/kinect-fusion-IEEE.pdf, 2011.

[3] Yuichi Taguchi et al. *Point-Plane SLAM for Hand-Held 3D Sensors*. Tech. rep. http://www.cc.gatech.edu/ yjian6/publication/taguchi2013icra.pdf: Georgia Institute of Technology, University of Michigan.