Humanoid Robotics

3D World Representations

Maren Bennewitz
Robots in 3D Environments

source: Honda
Motivation

- Robots live in the 3D world
- Collision avoidance, motion planning, and localization require accurate 3D world models
- Given: 3D point cloud data from known robot and sensor poses
- Question: How to represent the 3D structure of the environment?
Laser Scanning Principle

Range scanners measure the distance to the closest obstacle

Image courtesy: Sick
Other Range Sensors – Kinect
Example: Data Acquisition
Popular Representations of the 3D World

- Point clouds
- Voxel grids
- Height maps
- Surface maps
- Meshes
- Distance fields
- ...

Point Clouds

- Set of 3D data points in world frame
- Obtained, e.g., by a laser scanner or depth camera
Point Clouds

Pros

- No discretization of data
- Mapped area not limited

Cons

- Unbounded memory usage
- No constant time access for location queries
- No distinction between free or unknown space
Point Clouds and Efficient Location Queries

- Naïve implementation (list, array) has a linear complexity for location queries
- More effective solutions through kd-trees
- **kd-trees** operate in k-dimensions
- Space-partitioning data structure for organizing k-dimensional points
- Search/insert/delete in **logarithmic** time on average

[see exercise]
Example: \textit{kd-Tree (2-dim.)}

Binary space partitioning

3D Voxel Grids
3D Voxel Grids

Pros
- Volumetric representation
- Constant access time
- Probabilistic update possible

Cons
- Memory requirement: Complete grid is allocated in memory
- Extent of the map has to be known/guessed
- Discretization errors
2.5D Maps: Height Maps

Average over all points that fall into a 2D cell and consider this as the height value
2.5D Maps: Height Maps

Pros
- Memory efficient (2D)
- Constant time access

Cons
- No vertical objects
- Only one level is represented
Example:
Problem of Height Maps
Multi-Level Surface Maps (MLS)
MLS Map Representation

Each 2D cell stores a set of “patches” consisting of:

- The height mean $\mu$
- The height variance $\sigma$
- The depth value $d$

Note:

- A patch can have no depth (flat objects, e.g., floor)
- A cell can have one or many patches (vertical gap, e.g., bridges)
From Point Clouds to MLS Maps

- Determine the 2D cell for each 3D point
- Compute vertical intervals based on a **threshold**

- Determine for the vertical objects:
  - The **height** and its **variance**
  - The **depth** as the difference between the highest and the lowest measurement
Example: MLS Maps

Point cloud

Multi-level surface map
MLS Maps

Pros

- Can represent multiple surfaces per 2D cell

Cons

- No volumetric representation but a discretization in the vertical dimension
- Several tasks in a MLS map are not straightforward to realize
Octree-Based Representation

- Tree-based data structure
- Recursive subdivision of the space into octants
- Volumes allocated as needed
- “Smart” 3D grid
Octrees
Octrees

Pros
- Full 3D model
- Inherently multi-resolution
- Memory-efficient, volumes only allocated as needed
- Probabilistic update possible

Cons
- Efficient implementation can be tricky (memory allocation, update, map files, ...)

[see exercise]
Multi-Resolution Queries

\[ P(m_i) = \max_{j=1...8} P(m_{ij}) \text{ with } m_{ij} \in \text{children}(m_i) \]
OctoMap Framework

- Based on octrees
- Probabilistic, volumetric representation of occupancy including unknown
- Supports multi-resolution map queries
- Memory efficient
- Generates compact map files (maximum likelihood map as bit stream)
- Open source implementation as C++ library available at http://octomap.github.io/
Ray Casting for Map Updates

- Ray casting from sensor origin to end point in the map along the beam
- Mark last voxel as occupied, all other voxels on ray as free
- Measurements are integrated probabilistically given the robot’s pose (recursive binary Bayes’ filter)
Probabilistic Map Update

- Occupancy probability modeled as recursive binary Bayes’ filter

\[
p(m_i \mid z_{1:t}, x_{1:t})
= \left[ 1 + \frac{1 - p(m_i \mid z_t, x_t)}{p(m_i \mid z_t, x_t)} \frac{1 - p(m_i \mid z_{1:t-1}, x_{1:t-1})}{p(m_i \mid z_{1:t-1}, x_{1:t-1})} \frac{p(m_i)}{1 - p(m_i)} \right]^{-1}
\]

- Efficient update using log-odds notation

\[
l(m_i \mid z_{1:t}, x_{1:t})
= \underbrace{l(m_i \mid z_t, x_t)}_{\text{inverse sensor model}} + \underbrace{l(m_i \mid z_{1:t-1}, x_{1:t-1})}_{\text{recursive term}} - \underbrace{l(m_i)}_{\text{prior}}
\]
Video: Large Outdoor Area

Freiburg computer science campus

(292 x 167 x 28 m³, 20 cm resolution)
Online Mapping With Octomap