BRICS_3D: 3D Perception and Modeling Library

Sebastian Blumenthal
Outline

• Introduction
• 3D Perception
• 3D World Model
Introduction

- Context of work
  - BRICS (Best Practice in Robotics)
  - Goal: Improve development process
    - What are best practice algorithms for 3D perception and modeling?
Introduction

• What is 3D Perception and Modeling?
  – *3D Perception* is ability to perceive a 3D world
  – *3D Model* is a digital representation of 3D entities
  – *3D World Model* manages 3D data over a (long) period of time
Introduction

• What is 3D Perception and Modeling?
  – *3D Perception* is ability to perceive a 3D world
  – *3D Model* is a digital representation of 3D entities
  – *3D World Model* manages 3D data over a (long) period of time

• Why do we need it?
  – Robot needs a model to reason about world
Introduction

Mobile Manipulation use cases:

- **Environment reconstruction**
  - Retrieve a 3D environment from sensors, useful for *Mobile Manipulation* planners

- **Object recognition**
  - Identify objects (+ surface for contact) in a scene that are task relevant. E.g. a cup to be grasped
  - Retrieve a 6DoF pose
Common 3D perception and modeling data-types in robotics:

- Point
- Point cloud
- Triangle mesh (required for mobile manipulation)
Introduction

- Different approaches and implementations for 3D perception and modeling
- Approaches often not easily exchangeable
  - comparable
- To deduce best practice, comparisons of algorithms are needed
Introduction

• The problem is to *identify* best practice algorithms for 3D perception and modeling

• BRICS_3D: Goal is to make algorithms
  – measurable
  – comparable
  – reusable
Introduction

• The problem is to identify best practice algorithms for 3D perception and modeling

• BRICS_3D: Goal is to make algorithms
  – measurable
  – comparable
  – reusable

Which algorithm?
Introduction

- The problem is to *identify* best practice algorithms for 3D perception and modeling

- BRICS_3D: Goal is to make algorithms
  - measurable
  - comparable
  - reusable

  Which algorithm?
  Which algorithm implementation?
Introduction

1.) Exploration
2.) Harmonization
3.) Refactoring
4.) Integration
5.) Evaluation

- Problem domain
  - Algorithms
  - Libraries
  - Benchmarks

- Exploration
- Harmonization
- Refactoring
- Integration
- Evaluation

- Real robot
- Simulation
- Bar charts
3D Perception
Algorithm taxonomy

- Depth perception
- Filtering
- Feature extraction
- Registration
- Segmentation
- Classification
- Mesh generation
- Visualization
Depth Perception

- Process of acquiring 3D data
- 3D Sensors
  - 3D cameras
  - Stereo cameras
  - Tilting laser scanners
- Mostly hardware specific
- Serves as input for perception algorithms
Filtering

• **Process of modifying a data stream**
• Size reduction of input data
  – Octree is defacto standard [Nüchter, 2008]
• Noise reduction
• Region of Interest (ROI) Extraction (but not detection)

Octree size reduction
Feature Extraction

• **Process of detecting and/or describing distinctive points or regions**

• Local and global approaches
  - Local
    • Spin Images, Point Feature Histograms (PFH)
  - Global
    • Extended Gaussian Image

• Normal Estimation (required by many other algorithms)
Registration

- Process of merging captures from different viewpoints into one global, consistent coordinate frame
- Iterative Closest Point (ICP) + various improvements
  - [Besl and McKay, 1992], [Zhang, 1992]
  - E.g. more robustness with PFH or FPFH [Rusu et al., 2009]
- Normal Distributions Transform (NDT)
  - [Magnusson et al., 2007]
- Hough Scan Matched in 3D (HSM3D)
  - [Carpin and Censi, 2009]
Segmentation

- Process of spatial partitioning of point clouds into subsets
- Many algorithms base on estimated point normals
  - [Rusu et al., 2008]
- Fitting of geometric primitives
  - Planes, cylinders, etc.
  - [Weingarten et al., 2003]
- Euclidean Clustering
Classification

- Process of creating association between data and (typically) prior knowledge
- Approaches
  - k-Nearest Neighbours
  - Support Vector Machines
  - Neural Networks
Mesh generation

- **Process of transforming a 3D point cloud into a triangle mesh**
- Many approaches base on Delaunay triangulation
  - EigenCRUST [Kolluri et al., 2004]
  - ProFORMA [Q. Pan, et al., 2009]
  - Alpha-Shapes [Edelsbrunner and Mücke, 1992]
Visualization

- **Process of displaying(rendering) 3D data**
- Typically done on graphics adapter hardware
- Standardized interfaces: OpenGL, DirectX
Example

- Environment reconstruction use case

```
Depth perception
  Filtering
  Registration
  Segmentation
  Mesh generation
  Visualization

raw data
  point cloud
  triangle mesh
```
## Existing Libraries

### 3D perception and modeling libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Depth perception</th>
<th>Filtering</th>
<th>Registration</th>
<th>Segmentation</th>
<th>Mesh generation</th>
<th>Visualization</th>
<th>World Model</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DTK</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRPT</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>*</td>
<td>* = only 2D</td>
</tr>
<tr>
<td>IVT</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td>* = only 2D</td>
</tr>
<tr>
<td>FAIR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>* = only 2D</td>
</tr>
<tr>
<td>PCL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VTK</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITK</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meshlab</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGAL</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>*</td>
<td>* = uses geomview</td>
</tr>
<tr>
<td>Gmsh</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Qhull</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRICS_3D</td>
<td>(✓)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Complete Table: [http://www.best-of-robotics.org/2nd_researchcamp/Libraries](http://www.best-of-robotics.org/2nd_researchcamp/Libraries)
BRICS_3D Facts

- C++ Library
- License: BSD / LGPL
- SVN:
  https://svn.best-of-robotics.org/brics/BROCRE/BRICS_3D/trunk
- Doxygen documentation
  - $ doxygen BRICS_3D.doxyfile
  - $ firefox ./doc/html/index.htm
- Sample project (ROS): Tower of Hanoi
  http://youbot-store.com/libraries/20/software.aspx
#include <core/PointCloud3D.h>
#include <iostream>

int main(int argc, char **argv) {
    BRICS_3D::PointCloud3D* myPointCloud = new BRICS_3D::PointCloud3D();
    myPointCloud->addPoint(BRICS_3D::Point3D(0,0,0));
    myPointCloud->addPoint(BRICS_3D::Point3D(1,0,0));
    myPointCloud->addPoint(BRICS_3D::Point3D(0,1,0));
    myPointCloud->addPoint(BRICS_3D::Point3D(0,0,1));
    std::cout << *myPointCloud;
    delete myPointCloud;
}

Output:
0 0 0
1 0 0
0 1 0
0 0 1
BRICS_3D API (Excerpt)

- Filtering
  - Octree reduction, ROIs
    ```
    void filter(PointCloud3D* originalPointCloud, PointCloud3D* resultPointCloud);
    ```

- Feature Extraction
  - Normals
    ```
    void estimateNormals(PointCloud3D* pointCloud, NormalSet3D* estimatedNormals);
    ```

- Registration
  - ICP
    ```
    void match(PointCloud3D* model, PointCloud3D* data, IHomogeneousMatrix44* resultTransformation);
    ```
BRICS_3D API (Excerpt)

• Classification
  – k-Nearest Neighbour
    void setData(PointCloud3D* data);
    void findNearestNeighbors(Point3D* query, std::vector<int>* resultIndices, unsigned int k = 1)

• Mesh generation
  – 2.5D Delaunay Triangulation
    void generateMesh(PointCloud3D* pointCloud, ITriangleMesh* mesh);
BRICS_3D Processing example

PointCloud3D* pointCloud1 = new PointCloud3D();
PointCloud3D* pointCloud2 = new PointCloud3D();
pointCloud1->readFromTxtFile("foo.txt");
pointCloud2->readFromTxtFile("bar.txt");

/* reduce point cloud with Octree filter */
Octree* octreeFilter = new Octree();
octreeFilter->setVoxelSize(4.0);
PointCloud3D* reducedPointCloud = new PointCloud3D();
octreeFilter->filter(pointCloud1, reducedPointCloud);

/* perform registration via ICP */
IterativeClosestPointFactory* icpFactory = new IterativeClosestPointFactory();
IIterativeClosestPointPtr = icpFactory->createIterativeClosestPoint();
IHomogeneousMatrix44* resultTransformation = new HomogeneousMatrix44();
icp->match(pointCloud1, pointCloud2, resultTransformation);

/* create mesh */
ITriangleMesh* mesh = new TriangleMeshExplicit();
DelaunayTriangulationOSG* meshGenerator = new DelaunayTriangulationOSG();
meshGenerator->generateMesh(pointCloud1, mesh);
cout << "Number of generated triangles: " << mesh->getSize() << endl;

/* visualize the point cloud */
OSGPointCloudVisualizer* visualizer = new OSGPointCloudVisualizer();
visualizer->visualizePointCloud(reducedPointCloud); //blocking call
3D World Model
Existing World Models

• Grid-based representations
  – Occupancy grids, Octrees, Octomap, Collision Map

• Object-oriented representations
  – Local dynamic map (LDM), Three-Pillar-Information Model
Existing World Models

- Existing implementations are application specific
- No common WM implementation exists
Existing World Models

- Existing implementations are application specific
- No common WM implementation exists

- What about the 3D computer graphics domain?
  - Scenegraph
World Model Concept

● Our approach follows the scenegraph concept
  – Objects
  – Graph (DAG) of geometrical relations

● Leaves can contain
  – Raw data
  – Intermediate processing data
  – "high level objects"

● Inner graph nodes can be
  – Transforms
  – Groups
World Model Concept

[Diagram showing a tree structure with nodes labeled as Sensor (raw data), Downsampling PC, Scene Objects, Robot #1, Point Cloud, Table, Plate, Leg #1, Leg #2, (blue) Cylinder, (red) Box, (green) Box, etc., with transformation functions (TF) and time stamps (t1, t2, t3, t4).]
World Model API

● Update interface

```cpp
bool addNode(unsigned int parentId, unsigned int & assignedId, vector<Attribute> attributes);
bool addGroup(unsigned int parentId, unsigned int & assignedId, vector<Attribute> attributes);
bool addTransformNode(unsigned int parentId, unsigned int & assignedId, vector<Attribute> attributes, IHomogeneousMatrix44::IHomogeneousMatrix44Ptr transform, TimeStamp timeStamp);
bool addGeometricNode(unsigned int parentId, unsigned int & assignedId, vector<Attribute> attributes, Shape::ShapePtr shape, TimeStamp timeStamp);
bool setNodeAttributes(unsigned int id, vector<Attribute> newAttributes);
bool setTransform(unsigned int id, IHomogeneousMatrix44::IHomogeneousMatrix44Ptr transform, TimeStamp timeStamp);
bool deleteNode(unsigned int id);
bool addParent(unsigned int id, unsigned int parentId);
```
World Model API

• Query interface

```cpp
bool getNodes(vector<Attribute> attributes, vector<unsigned int>& ids);
bool getNodeAttributes(unsigned int id, vector<Attribute>& attributes);
bool getNodeParents(unsigned int id, vector<unsigned int>& parentIds);
bool getGroupChildren(unsigned int id, vector<unsigned int>& childIds);
bool getTransform(unsigned int id, TimeStamp timeStamp, IHomogeneousMatrix44::IHomogeneousMatrix44Ptr& transform);
bool getGeometry(unsigned int id, Shape::ShapePtr& shape, TimeStamp& timeStamp);
bool getTransformForNode (unsigned int id, TimeStamp timeStamp, IHomogeneousMatrix44::IHomogeneousMatrix44Ptr& transform);
```
World Model API

- High level interface

```cpp
void getSceneObjects(vector<Attribute> attributes, vector<SceneObject>& results);
void getCurrentTransform(unsigned int id, IHomogeneousMatrix44::IHomogeneousMatrix44Ptr transform);
void insertTransform(unsigned int id, IHomogeneousMatrix44::IHomogeneousMatrix44Ptr transform);
void addSceneObject(SceneObject newObject, unsigned int& assignedId);
```
Summary

- BRICS_3D developed according to a methodology how to identify best practice
- Most perception algorithms fall into a common taxonomy
  - Some API interfaces and examples of BRICS_3D presented
- BRICS_3D World model approach bases on the scenegraph concept
- Further documentation: doxygen
Thank you for your attention.
References


Appendix

- Impressions: *BRICS_3D* library

Octree size reduction

IPA data set loader

Iterative Closest Point

Delaunay triangulation