Making sense of 3D data

Nico Blodow
blodow@cs.tum.edu
Intelligent Autonomous Systems, TUM, Germany

June 14, 2012
Motivation

Central question in many 3D perception applications:
How can we – at all times – know what is going on around us?
Motivation

Central question in many 3D perception applications:
How can we – at all times – know what is going on around us?

Focus of my work:
Dynamic Scene Perception and Spatio-temporal Memory for Robot Manipulation
Motivation

In service robotics especially, we have little to no control over the environment:

Wide range of objects:

- textured, non-textured
- 3D objects, flat objects (cutlery, paper...)
- indifferentiable objects (12 equal cups)
- state of objects (my cup, empty/full milk carton...)
- clutter, occlusions
Motivation

In service robotics especially, we have little to no control over the environment:

Wide range of object locations:

- table top
- containers (cupboards, drawers...)
- fridge
Motivation

In service robotics especially, we have little to no control over the environment:

Other problems:

- humans interfere with task / objects
- large universe of objects
- ever changing universe of objects
- lighting
- ...
Motivation

Many approaches:
- environment mapping, room / furniture classification
- table extents and positions, object catalog, container contents
- object detection, reconstruction and classification
- object identity resolution, tracking, etc.

Key Challenges
- data throughput
- dynamic environments
- humans
- hard constraints on processing times

This means: we need fast as well as general algorithms
Outline

1. GPU-Accelerated depth image processing
   pcl::cuda
   Kinect
   Results

2. Point Cloud Compression
   Octree
   Octree-based PC Compression
   Detail Component Compression

3. Unstructured Information Management Architecture
   Next Best View
   Room and furniture mapping
Past

Current strategies for optimizations:

- Downsampling = much less data
- Spatial locators / tree structures
- Ignoring some problems (online, humans)
- Reordering points for cache optimization
- "framedropping" / Using slow scanners – problem: Kinect

While these are all good and valid strategies (We can reach processing speeds in the range of seconds) our target is < 30 ms

Kinect produces VGA × 5 bytes @ 30Hz = 44MB/s!

➤ GPGPU programming
**pcl::cuda**

- focus on **real-time** point cloud processing

- implemented in **thrust** — CUDA template library similar to STL

- biggest problem: data transfer between Host (=CPU) and Device (=GPU)

- therefore: all algorithms should be implemented on the GPU to minimize performance hits

- **Input data:** Kinect Bayer image + depth image
  (or of course everything from **pcl::io**)
pcl::cuda

- **pcl::cuda::io**
deals with IO, projection of depth data to 3D, GPU memory transfer methods, Kinect "dediscretization", subcloud extraction etc.

- **pcl::cuda::nn**
neighborhood search; depth-image-based neighborhood search

- **pcl::cuda::features**
contains infrastructure for feature estimation, several implementations for normal estimation

- **pcl::cuda::sampleconsensus**
deals with robust estimation techniques and models. RANSAC and (novel, parallel) MultiSAC estimators, novel optimized plane estimator
pcl::gpu

- itseez’s reimplementation in “pure” CUDA
- **kinfu** — Kinect Fusion reimplementation
- **features** — normals, spin images, PFH, FPFH, VFH etc.
- **octree** search structures
Improving Kinect Data

e.g. Wall (top down view)
Improving Kinect Data

Kinect data discretized in disparity
Improving Kinect Data

Normal estimation (and all other feature computations) will have errors
If we knew the true geometry, we could compute whether the measured (red) point could have been sampled from that surface (purple point)
Improving Kinect Data

We don’t know the model (except for e.g. in RANSAC), but we can assume it to be smooth.
Improving Kinect Data

same parameters:
MultiSAC plane estimation

- Replace 3-point sample for plane estimation with 1 point + (smooth/oversmooth) normal
- leads to lower nr. of iterations $k = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$

1. Create batch of plane hypothesis on GPU by sampling 1 point each
2. Iterate (CPU) over $k$ plane hypotheses, compute inliers on GPU
3. After accepting model, each model created from an inlier can be invalidated easily
4. Compare plane equations of accepted model with all other valid models, only recompute inliers when necessary
Performance on NVIDIA GTX 560

- CUDA yields remarkable speedup for highly parallel tasks
  Example:
  - `openni_camera` driver in ROS: 70% CPU usage
  - `OpenNIGrabber` in PCL: 30% CPU usage
  - **Our Solution**: 3% CPU usage, 3% GPU usage.

<table>
<thead>
<tr>
<th>Task</th>
<th>CPU(OpenMP)</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disparity to Cloud + smoothing</td>
<td>25 – 35 ms</td>
<td>2 – 2.5 ms</td>
</tr>
<tr>
<td>Normal Estimation</td>
<td>250 – 1000 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>Fast Normal Estimation</td>
<td>2.5 – 3.5 ms</td>
<td>&lt; 0.15 ms</td>
</tr>
<tr>
<td>Surface Orientation Segmentation</td>
<td>1 s</td>
<td>≈ 100 ms</td>
</tr>
<tr>
<td>Multiple Plane Estimation</td>
<td>&gt; 10 s(^1)</td>
<td>50 – 200 ms</td>
</tr>
</tbody>
</table>

\(^1\) possibly much longer
Using the semantic map in perception
Harnessing OpenGL + CUDA interoperability

Using semantic maps for real time semantic segmentation

Normal space, depth image and mask from sensor’s point of view (< 1ms)

semantic map (normal space), distances between Kinect data and semantic map, distances filtered (∼ 1ms)
Outline

1. GPU-Accelerated depth image processing
   pcl::cuda
   Kinect
   Results

2. Point Cloud Compression
   Octree
   Octree-based PC Compression
   Detail Component Compression

3. Unstructured Information Management Architecture
   Next Best View
   Room and furniture mapping
Goals/Motivation:

- Efficient for real-time processing
- General compression approach for unstructured point clouds (varying size, resolution, density, point ordering)
- Exploit spatial sparseness of point clouds
- Exploit temporal redundancies in point cloud streams
- Keep introduced coding distortion below sensor noise

(Work with Julius Kammerl)
Hierarchical tree data structures can efficiently describe sparse 3D information.

Focus on real-time compression favors octree-based point cloud compression approach.

Octree structures enable fast spatial decomposition.
Octree-based Encoding

- Root node describes a cubic bounding box which encapsulates all points
- Child nodes recursively subdivide point space
- Nodes have up to eight children ⇒ Byte encoding
- Point encoding by serializing high-resolution octree structures!
Temporal Encoding

Temporarily adjacent point clouds often strongly correlate:

Differentially encode octree structures using XOR:

- Gain: reduced entropy of the serialized binary data!
- Compression using fast range coder (fixed-point version of arithmetic entropy coder)
Results

- Experimental results of octree-based point cloud compression
Results

- Data rate comparison between regular octree compression (gray) and differential octree compression (black) for 1 $mm^3$ resolution
Demo - change detection

Applications:

- 3D (not just 2.5D!) video streaming
- Real-time spatial change detection based on XOR comparison of octree structure
Detail encoding

- Challenge: With increased octree resolution, complexity grows exponentially
- Solution: Limit octree resolution and encode point detail coefficients
- Enables trade-off between complexity and compression performance
- Also applicable to point components (color, normals, etc.)
Compression Pipeline

Encoding Pipeline:
- Point Cloud
- Octree Structure
- Point Component Encoding
- Position Detail Coefficients
- Binary Serialization
- Entropy Encoding
- Compressed PC

Decoding Pipeline:
- Compressed PC
- Entropy Decoding
- Octree Structure
- Point Component Decoding
- Point Details
- Point Cloud
Results

- Experimental results for point detail encoding at octree resolution of $9 \text{mm}^3$.
- Enables fast real-time encoding with high point precision
- Constant run-time with Octree + Point Detail Coding
- *(Byproduct: octree based search operations, downsampling, point density analysis, change detection, occupancy maps)*
Demo - point cloud compression

- Point cloud compression demo
Outline

1. GPU-Accelerated depth image processing
   pcl::cuda
   Kinect
   Results

2. Point Cloud Compression
   Octree
   Octree-based PC Compression
   Detail Component Compression

3. Unstructured Information Management Architecture
   Next Best View
   Room and furniture mapping
Plugging everything together
Dealing with multimodal data

Robust robot systems have to solve really hard tasks

• harness various state-of-the-art methods from different fields

• data sources *(images, point clouds, web stores...)* and data formats can be very specialized

• UIMA (IBM Watson) is designed to deal with **unstructured** information created by different **experts**

• → can this be transferred from the text domain to 2D/3D data and ontological knowledge?
• Unstructured information processing and management
• Common analysis structure (CAS) holds original raw data, can hold multiple Subjects of Analysis (SOFAs), multiple Views, Annotations
• **Collection Readers**: Generate CAS with initial document (web scrapers, sensor streams, ontologies etc.)
• **Analysis Engines**: (primitive or aggregate) can enrich the CAS with Annotations on the different Views/SOFAs

• **Annotations** are stored in Index Repositories for access and search

• all data structures adhere to a language-agnostic hierarchical type system
• Analysis Engines (= ensembles of experts) is controlled by **flow controllers** → (task/scene/resource adaptation...)  
• CAS Consumers process the annotated CAS for storage or reuse (e.g. interface to KnowRob!)  
• MongoDB database collects long term memory, object models, environment models etc.
• Collection Engines can be spawned from Java, C++, command line from a CPE Descriptor
• CPM responsible for CAS Management, failure recovery, scale out
### Type System Definition

The following types (classes) are defined in this analysis engine descriptor. The grayed out items are imported or merged from other descriptors, and cannot be edited here. (To edit them, remove the gray overlay.)

<table>
<thead>
<tr>
<th>Type Name or Feature Name</th>
<th>SuperType or Range</th>
<th>Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ias.uima.core.Identifiable</td>
<td>uima.cas.TOP</td>
<td></td>
</tr>
<tr>
<td>ias.uima.cv.Mat</td>
<td>uima.cas.TOP</td>
<td></td>
</tr>
<tr>
<td>ias.uima.goggles.response</td>
<td>uima.cas.TOP</td>
<td></td>
</tr>
<tr>
<td>ias.uima.ros.std_msgs.Header</td>
<td>uima.cas.TOP</td>
<td></td>
</tr>
<tr>
<td>ias.uima.scene.Annotation</td>
<td>ias.uima.core.Identifiable</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.BarcodeAnnotation</td>
<td>ias.uima.scene.Annotation</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.Cluster</td>
<td>ias.uima.core.Identifiable</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.ClusterPoints</td>
<td>ias.uima.core.Identifiable</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.PclFeatureAnnotation</td>
<td>ias.uima.scene.Annotation</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.ReferenceClusterPoints</td>
<td>ias.uima.scene.ClusterPoints</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.Scene</td>
<td>ias.uima.core.Identifiable</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.scene.StandaloneClusterPoints</td>
<td>ias.uima.scene.ClusterPoints</td>
<td>uima.cas.TOP</td>
</tr>
<tr>
<td>ias.uima.spatial.PointCloud</td>
<td>uima.cas.TOP</td>
<td></td>
</tr>
<tr>
<td>ias.uima.spatial.PointIndices</td>
<td>uima.cas.TOP</td>
<td></td>
</tr>
</tbody>
</table>

- subtypes of Identifiable: storable in MongoDB
- Scene consist of Clusters taken at a certain time
- Clusters contain a point cloud and a list of Annotations
- "semantic" type hierarchy
• subtypes of Identifiable: storable in MongoDB
• Scene consist of Clusters taken at a certain time
• Clusters contain a point cloud and a list of Annotations
• "semantic" type hierarchy
Example: Robot Google Goggles

- 3D sensors can effectively segment object hypotheses (but resolution too low)
- leverage Google Goggles’ multimodal image analysis
- corresponding (higher-res) camera image region uploaded to Goggles
- Response: List of things (of various types) detected in object image
Example: Robot Google Goggles
Example: Robot Google Goggles

Segmentation / Clustering: Kinect-based Tabletop Object Detection

Product Link

Text + Translation

User Submitted
Logo / Brand
Similar Image (+ original site)

(+ Barcodes, Landmarks, etc)
Current IAS-UIMA Capabilities

- German-Deli scraping
- Kinect/PCD File input
- ROS → UIMA interface
- GPU-based normal estimation, clustering
- per-cluster feature estimation (all PCL features)
- per-cluster Goggles annotation
- storage in MongoDB database
Outlook and Future Work

- Further integration of UIMA and previous perception systems (COP)
- Entity resolution framework (going from clusters/classification to object instances)
- Integration of / interfacing to KnowRob, CRAM
- Leverage Machine Learning for control flow, ensemble methods
Peer-reviewed Conference Publications

(1/2)


Peer-reviewed Conference Publications (2/2)


Journal and Workshop Articles

Journal Articles


Workshop Papers


Thank you

Thanks for your attention!
Autonomous Exploration and Mapping
3D Mapping of indoor environments
Next Best View given “window voxels”

voxel $v_i$ visibility from sensor position $s$

possible sensor positions to see voxel $v$

“window voxels” are between free and unknown space. Stacked Costmaps with visibility kernel yield good scanning poses.
Point Cloud Interpretation - Floor, Ceiling
Point Cloud Interpretation - Walls, Vertical and Horizontal Planes
Point Cloud Interpretation - Fixtures, Doors, Drawers

Region growing from handles using median intensity and median average distance (work with Dejan Pangercic, Zoltan Marton)
Door and Drawer Hypotheses Validation through Interaction

Work by Thomas Ruehr.
Door and Drawer Hypotheses Validation through Interaction 2
Final (Manually Augmented) Map

Full Video: http://www.youtube.com/watch?v=T15ycSmNOFY