Real-time RGB-D Mapping and 3-D Modeling on the GPU using the Random Ball Cover Data Structure

1st IEEE Workshop on Consumer Depth Cameras for Computer Vision
in conjunction with ICCV 2011, Barcelona, Spain

2011-11-12

D. Neumann¹, F. Lugauer¹, S. Bauer¹, J. Wasza¹, J. Hornegger¹,²
¹ Pattern Recognition Lab, Dept. of Computer Science
² Erlangen Graduate School in Advanced Optical Technologies (SAOT)
Friedrich-Alexander-University Erlangen-Nuremberg
Reconstruction Framework
Methods
Performance Analysis
Qualitative Results
Summary & Outlook
Reconstruction Framework
Generation of 3-D Scene Model

- Working principle
  - Given a global scene model
  - Successively register the latest point cloud to the previous one
  - Merge the global scene with the aligned data
Reconstruction Framework

- **CPU**
  - RGB-D Frame Grabber
  - Transformation Estimation
  - 3-D Scene Model

- **GPU**
  - 2-D/3-D Transformation
  - 3-D Point Cloud
  - Landmark Extraction
  - RBC Data Structure Generation
  - RBC Nearest Neighbor Search
  - Photometric ICP
    - Point Cloud Transformation

**Data Preprocessing**
- Start with RGB-D Frame Grabber
- Transform to 2-D/3-D Transformation
- Generate 3-D Point Cloud
- Extract Landmark
- Generate RBC Data Structure
- Search RBC Nearest Neighbor
- Estimate Transformation
- Apply Photometric ICP for convergence

**Output**
- 3-D Scene Model
Methods

Iterative Closest Point Algorithm [1]

Iterative Closest Point (ICP) Algorithm

\[
(R, t) = \arg \min_{R, t} \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \| (Rm_g + t) - f_g \|_2^2
\]

\[
f^m = \arg \min_{f \in \mathcal{F}} \| f_g - m_g \|_2^2 + \alpha \| f_p - m_p \|_2^2
\]

\[\rightarrow\text{photogeometric ICP}\]

rotation
moving point set
geometric information
Euclidean distance

\[
\begin{array}{c}
\tilde{R} \\
\mathcal{M} = \{ m \} \\
f_g, m_g \\
\| \cdot \|_2 \\
\alpha
\end{array}
\]

translation
fixed point set
photometric information
weighting factor \( \geq 0 \)
Methods

Random Ball Cover \[^{2,3}\] – efficient NN search


Nearest Neighbor (NN) Search

- Tree-based approaches
  - \textit{kd} tree
  - PCA tree
  - Ball tree
  - …

- Hardware acceleration techniques
  - Brute-force (BF) on GPU
  - … outperforms \textit{kd} tree on CPU \cite{4}

Random Ball Cover (RBC)

- Acceleration structure for efficient NN search
- Originally intended for high-dimensional search spaces
- Exploits parallel architecture of modern GPUs
  - Construction of data structure uses BF primitive
  - Dataset queries use BF primitive

- Two approaches
  - Exact RBC: slower, exact
  - One-shot RBC: faster, not exact
RBC Construction

BF search to find closest representative
RBC Query – Step 1

- Query Point
- Representative
- NN List

⇒ BF search among representatives
RBC Query – Step 2

BF search among NN list of closest representative
Performance Analysis
Runtime Partitioning
(regarding one ICP iteration)

- **NN search**
- **Transformation Estimation**
- **Transformation**

![Graphs showing relative runtime for different landmark counts.](image)

**Brute-force**

**One-shot RBC**
Absolute Runtime: Comparison

(NVIDIA GeForce GTX 460, Intel Core 2 Quad Q9550)

- NN query scales quadratically
- ICP iteration scales quadratically
- $0.62 \text{ ms (times 30 iterations) + set-up time} < 20 \text{ ms}$
- $0.99 \text{ ms per iteration}$
- Outperformed (up to factor of 2)
Error Evaluation
(approximative NN search)

![Graph showing mean error vs. number of representatives for different numbers of LMs (512, 1024, 2048, 4096). The graph indicates typical configuration: error < 5mm, and exactly one data point per representative.]

- All data points governed by a single representative
- BF
Qualitative Results
On-the-fly Reconstruction
Summary & Outlook
Summary & Outlook

- **Fast** ICP using the RBC data structure
  - Many algorithms rely on (real-time) ICP for surface alignment
  - RBC in low-dimensional spaces
  - Promising results: ICP runtimes < 20 ms

- **Outlook**
  - Improvements on reconstruction framework
    - Loop closure
    - SLAM
  - Application to other algorithms → feature matching
Thank you for your attention.

For further information (including source code requests) please contact the authors. The implementation is based on the “Range Imaging Toolkit” (RITK).

>> http://www5.cs.fau.de/ritk