Massively-Parallel Proximity Queries for Point Clouds

Max Kaluschke\textsuperscript{1}, Uwe Zimmermann\textsuperscript{2}, Marinus Danzer\textsuperscript{2}, Gabriel Zachmann\textsuperscript{1} and Rene Weller\textsuperscript{1}

\textsuperscript{1) University of Bremen} cgvr.cs.uni-bremen.de
\textsuperscript{2) KUKA Laboratories} kuka-labs.com
Motivation
Motivation

KINOPTIK

Introduction

Our Approach

Improvements

Conclusions
Previous Works

Point cloud collision detection [Klein and Zachmann, 2004]
Real-time collision detection and distance computation on point cloud sensor data [Pan et. Al., 2013]
Precondition

- Polygonal object representation: Inner Sphere Trees (ISTs)

- Point cloud captured in real-time via Kinect
**Basic Algorithm**

\[ \text{minDist} = \infty \]

**Parallel Algorithm**

\[ \text{getDistance}( \text{Root} \text{IST}, p, \text{minDist}) \]

\[ \text{getDistance}( \text{Sphere } s, \text{Point } p, d ) \]

forall the Children \( sc \) of \( s \) do

\[ d = \text{distance}( sc, p ) \]

if \( d < \text{minDist} \) then

\[ \text{getDistance}( sc, p, d ) \]

\[ \text{minDist} = \min( d, \text{minDist} ) \]

**Parallel** $\text{IST} \in \text{Root}$:

$\text{parallel}$ $\text{point } p \in \text{Point Cloud}$:

**Introduction**

**Our Approach**

**Improvements**

**Conclusions**
Test Scenarios

- Implemented in CUDA (5.5 & 6.0)
- Geforce GTX 780, 2GByte Memory
- Pre-recorded and artificial point clouds with up tp 5M points
Results: Parallel vs. Sequential

![Graph showing comparison of parallel and sequential time in ms vs. points x 1000]
Improvement(?) 1: Recursive vs. Iterative

![Diagram showing comparison between recursive and iterative methods with time in ms on the y-axis and points x 1000 on the x-axis. The recursive method is represented in red and the iterative method in green.]
Improvement(?) 2: Global vs. Shared Memory

Introduction

Our Approach

Improvements

Conclusions
Improvement(?) 3: Pre-Filtering
Improvement(?) 3: Pre-Filtering
Improvement(?) 3: Pre-Filtering

The diagram shows the time in ms for different points x 1000. The graph is divided into three parts: data transfer, kernel execution, and total time. The data transfer is represented by the red area, kernel execution by the green area, and the total time by the blue area. As the number of points increases, the total time also increases, indicating an improvement in efficiency.
Improvement(?) 3: Pre-Filtering

The graph compares the time in ms for Octree-Pre-Filtering and No Octree-Pre-Filtering as a function of the number of points per million. The green line for Octree-Pre-Filtering shows a lower and more stable time compared to the red line for No Octree-Pre-Filtering, indicating a significant improvement in performance.
Improvement(?) 3: Pre-Filtering

Introduction                                      Our Approach                                Improvements                                 Conclusions
Improvement(?) 3: Pre-Filtering
Improvement(?) 3: Pre-Filtering
Conclusions and Future Work

- First algorithm to compute distances between CAD objects and point clouds in real-time
  - < 10 msec for 5M points
- Easy to implement, robust
- Sometimes, easier is better

- Faster pre-filtering
- Other applications
  - VR
  - Haptics
  - Path-planning
Thank You!

DFG Grant SFB/TR 8 I08 [DextrousSpace], GZ TRR 8/3-2013