Massively-Parallel Proximity Queries for Point Clouds

Max Kaluschke

October 6, 2014

Motivation

Neuer Datentyp

- Neuer Datentyp: Point Clouds
- traditionell Triangle-Mesh
  - viel-erforschter Bereich
  - Surface Reconstruction sehr aufwändig
- immer frische Daten
  - Kinect hat ca. 30 FPS
  - kein Preprocessing möglich
- Umfassen sehr viele Punkte
  - Auflösung einer einzelnen Kinect Aufnahme: ca. 300.000 Punkte

Motivation

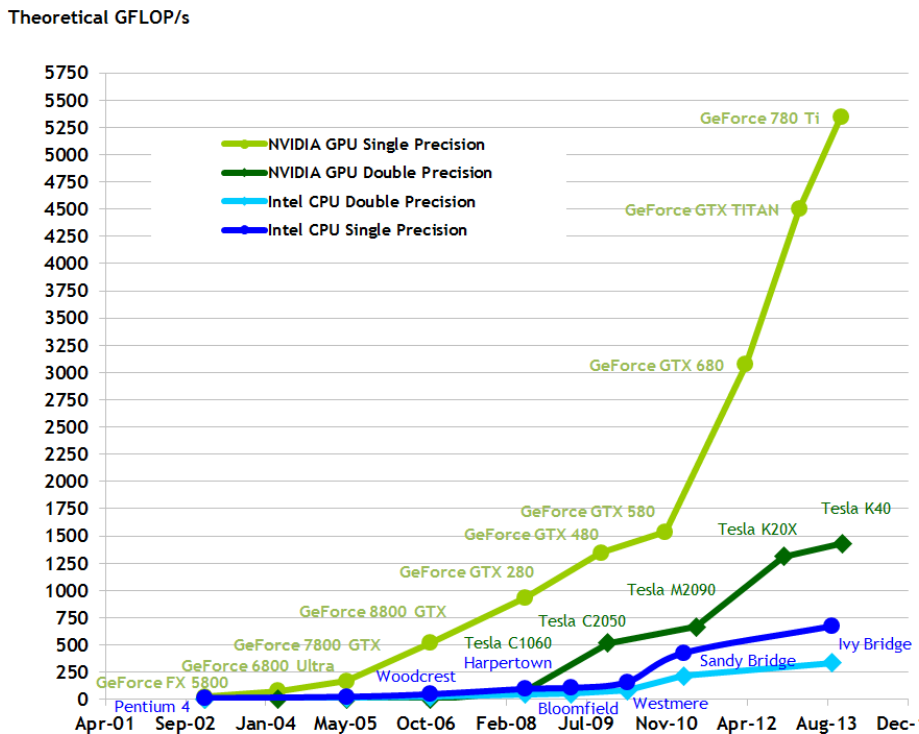
Kollisionen vermeiden

- Möglichst schnelle Berechnung

Motivation

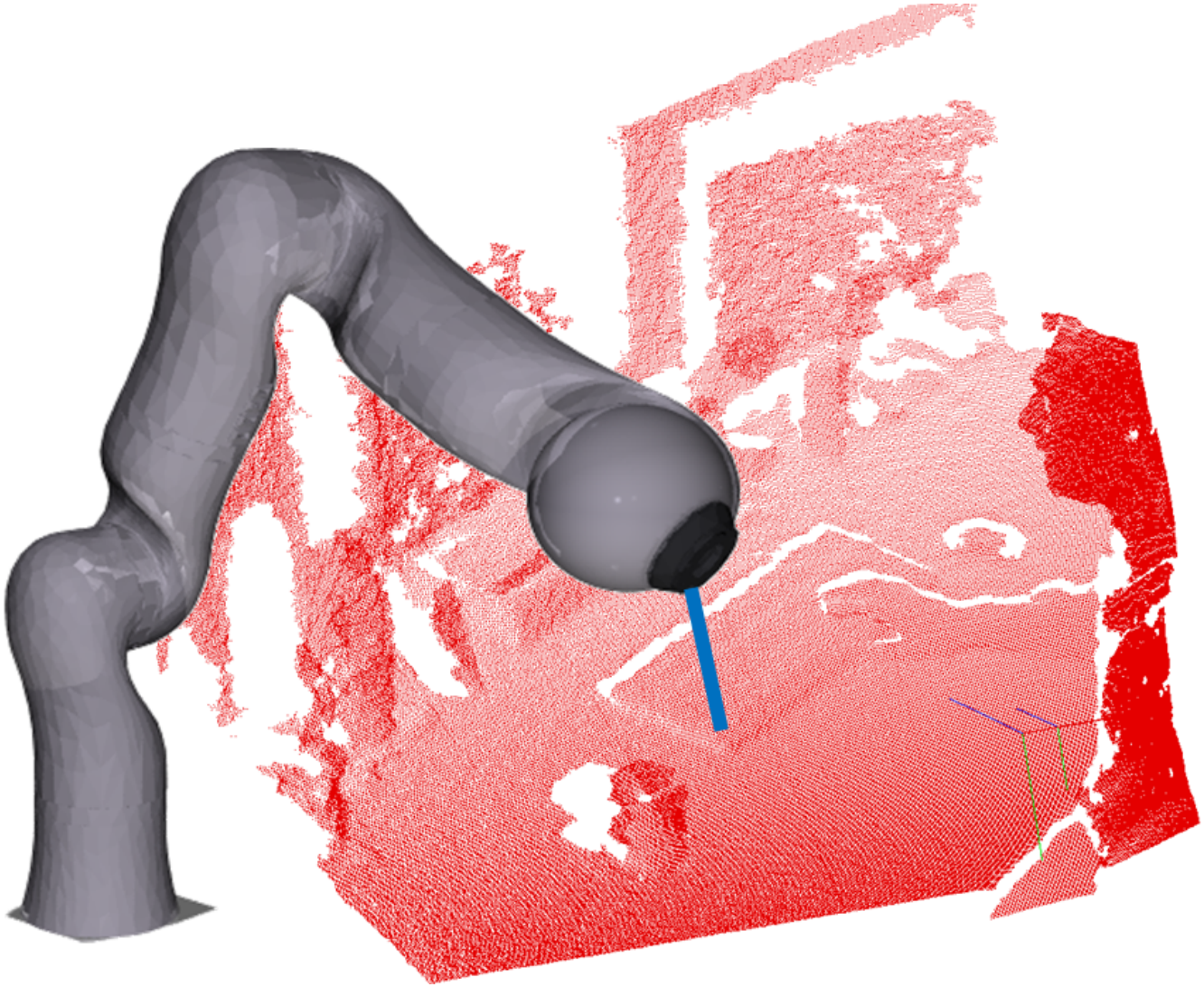
Vorteile der GPU

- CPU Fortschritt stagniert
  - früher: Transistoren werden immer kleiner → mehr Transistoren passen auf die selbe Fläche
  - in den letzten Jahren: mehr Cache, mehr Parallelisierung
- GPUs sind dagegen ausgelegt für parallele Arbeit
  - GPUs übertreffen CPUs in theoretischer Leistung:



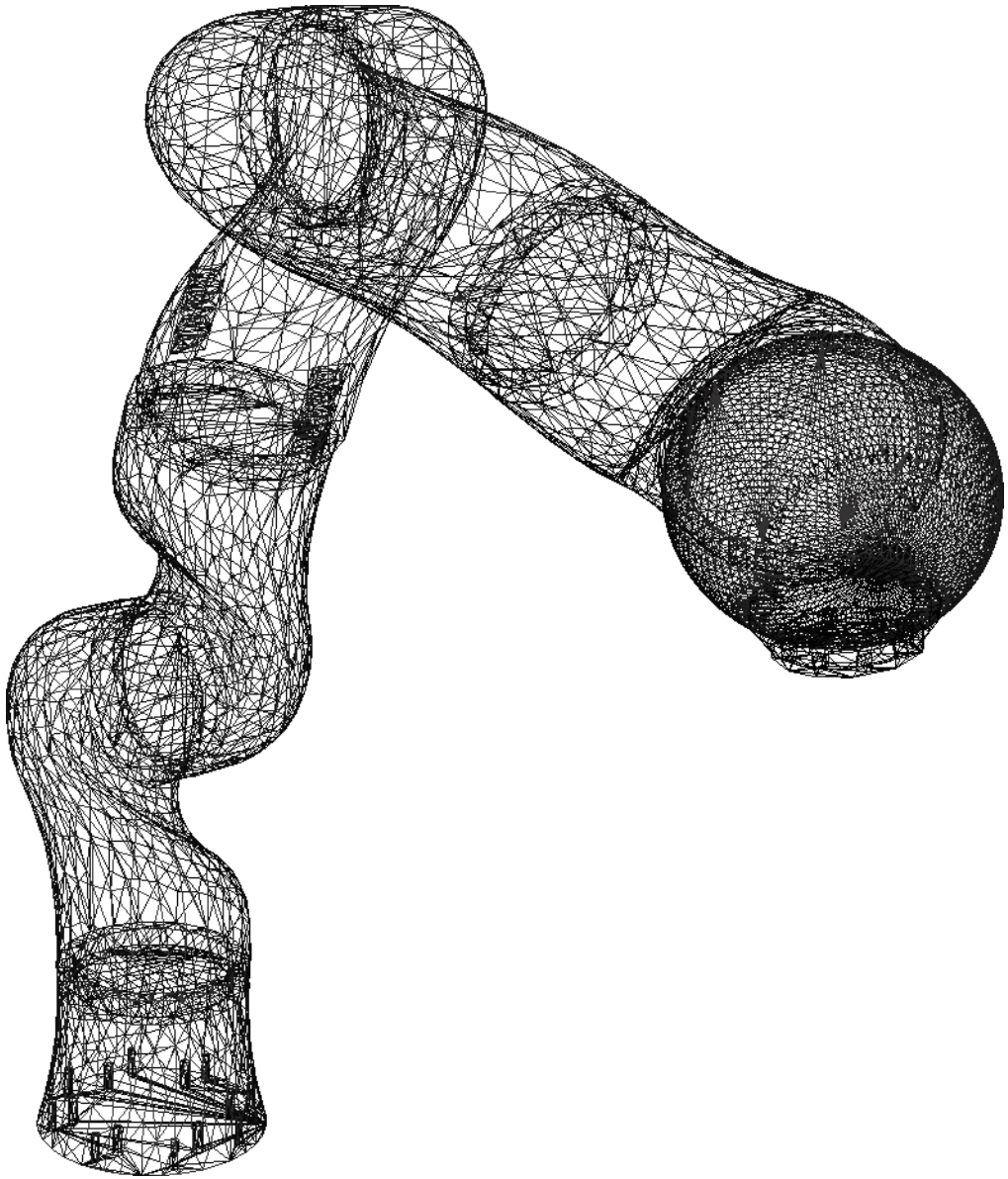
Das konkrete Setup

- Zusammenarbeit mit KUKA

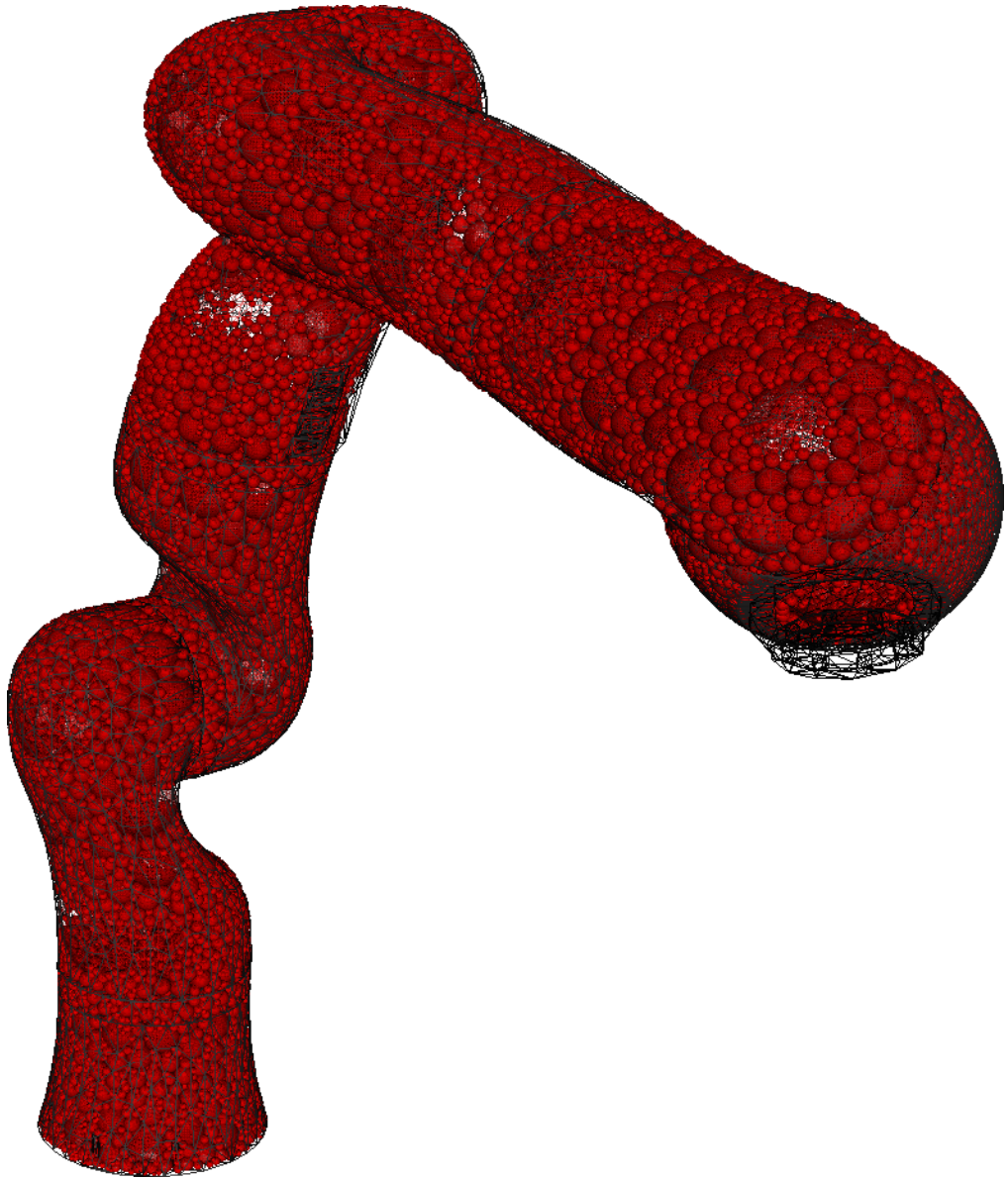


Der Roboter

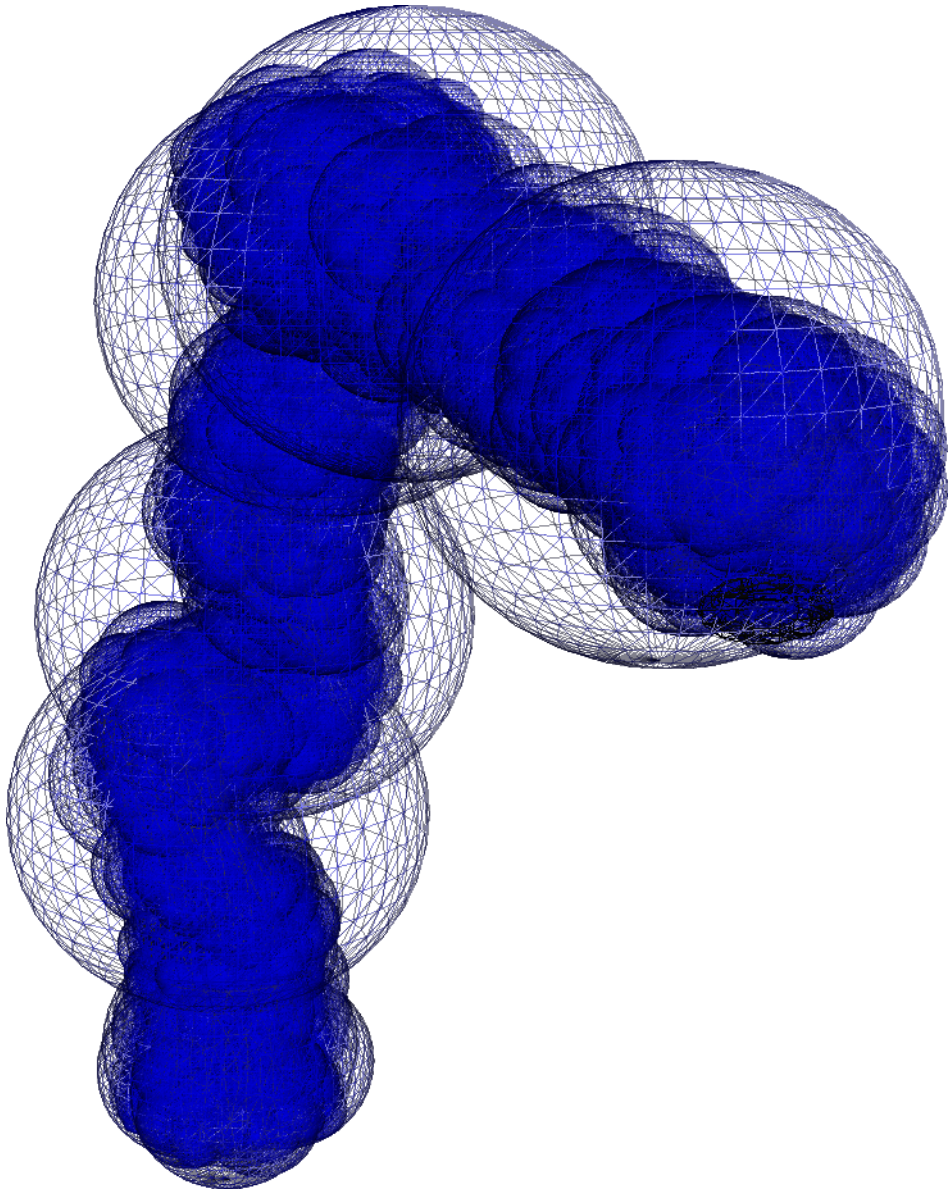
Der Roboter



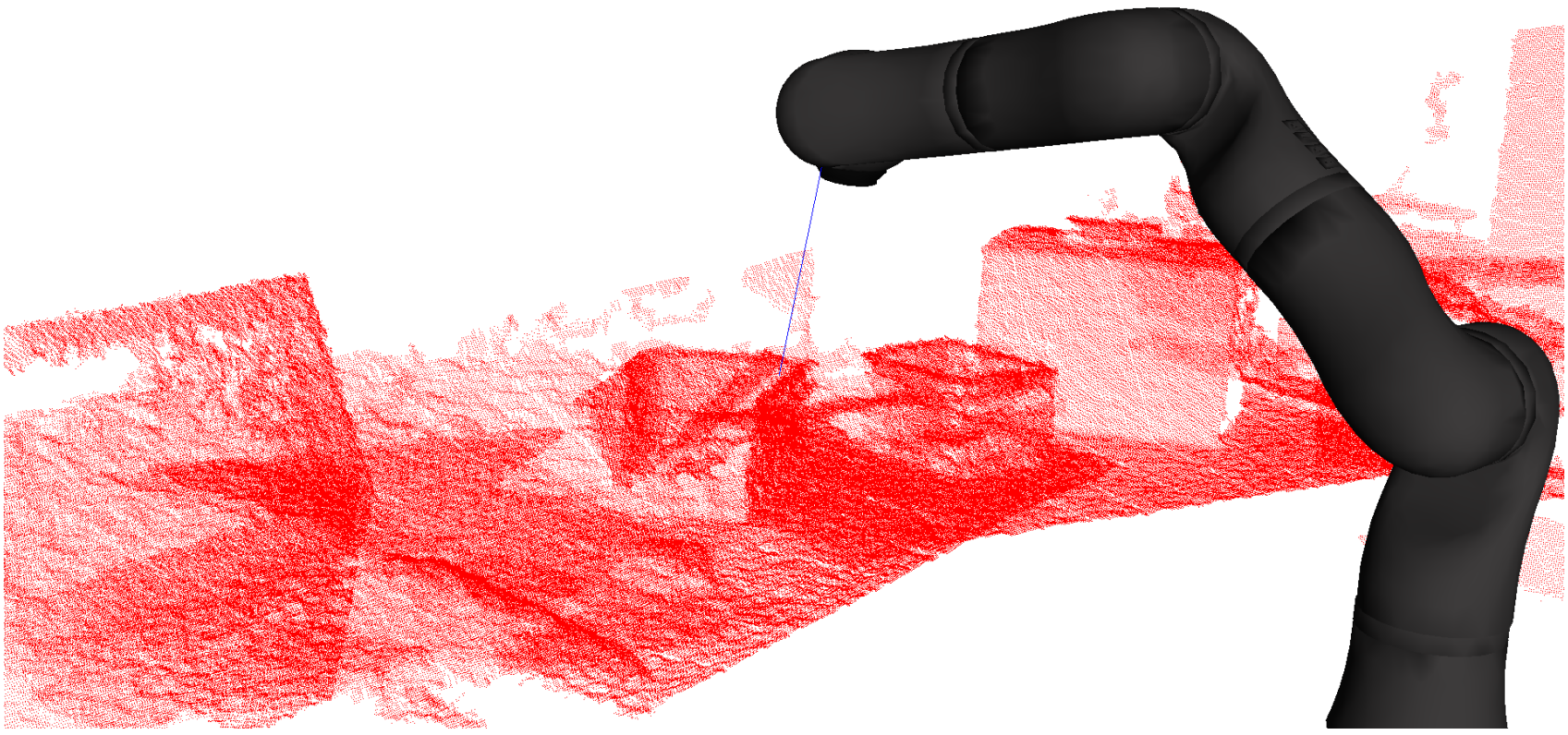
Der Roboter



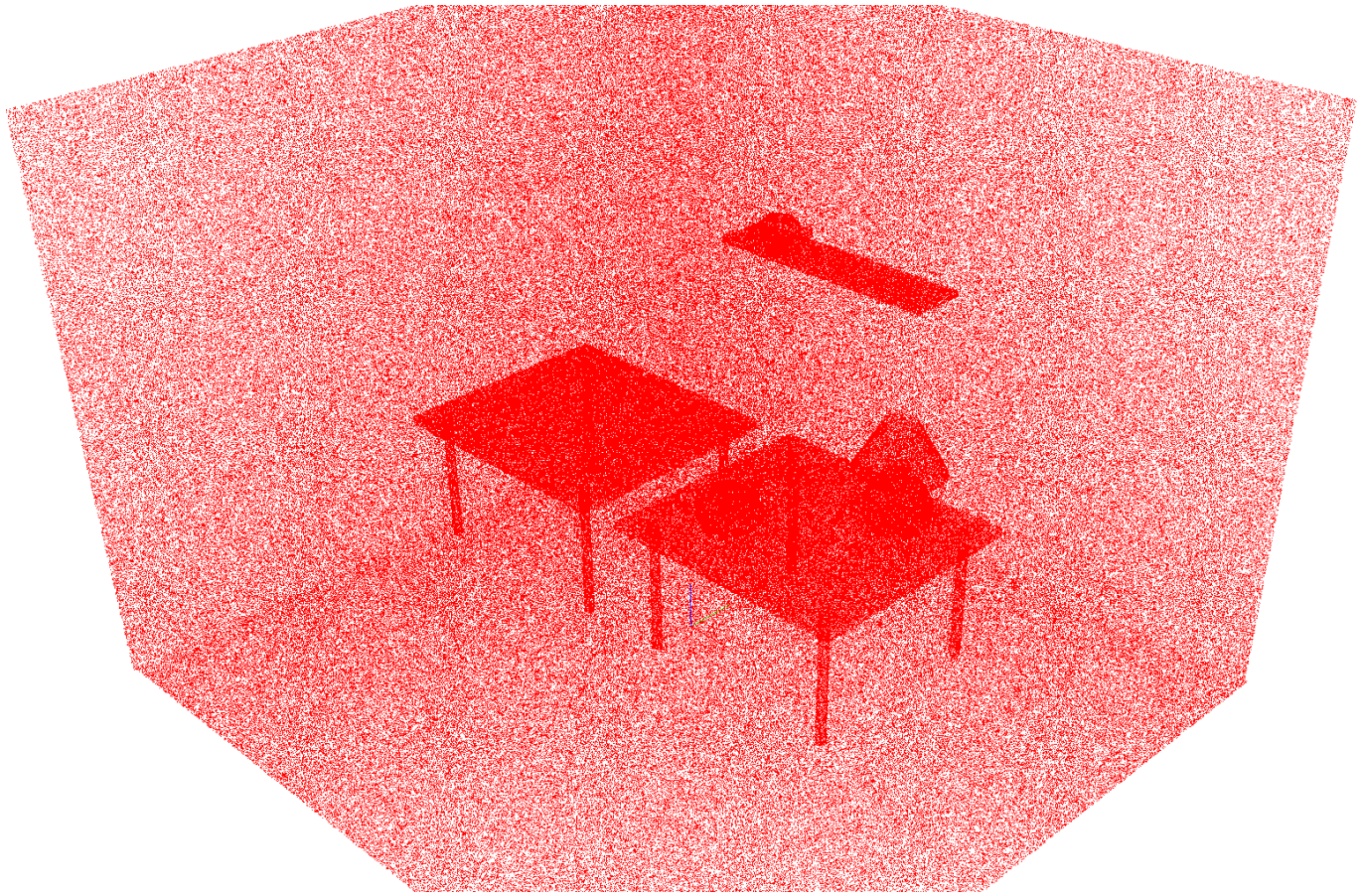
Der Roboter



Die Umgebung - Kinect Aufnahmen



Die Umgebung - von 3D-Mesh approximiert



Sequentieller Algorithmus

- Man iteriert über jeden Roboterteil und dann jeweils jeden Punkt aus der Punktwolke:
  - dabei macht man einen Abstandstest zwischen dem Punkt und dem Baum des Roboterteils
  - das Minimum aller Abstände ist unser kleinster Abstand

Algorithm 3.1: getMinDist(Set of ISTs R, point cloud P)

```
1: minDistance = maximum float
2: forall the ISTs r ∈ R do
3:     forall the Points p ∈ P do
4:         distance = traverseIST( Root sphere of r, p, minDistance )
5:         if distance < minDistance then
6:             minDistance = distance
```

Sequentieller Algorithmus

Traversierung der ISTs

- Rekursive Definition
- normale BVH-Traversierung
  - Innere Knoten führen zu Rekursion wenn ein kleinerer Abstand möglich ist
  - bei Blättern wird der Abstand zurückgegeben

Algorithm 3.2: traverseIST(Sphere s, point p, minDistance)

```
1: if s is Leaf then
2:     return d
3: forall the Children s_c of s do
4:     d = distance( s_c, p )
5:     if d < minDistance then
6:         traverseIST( s_c, p, d )
```

Paralleler Algorithmus

- sehr ähnlich zum sequentiellen Algorithmus
- globale Variable für die untere Schranke
  - globale Variablen können von jedem Thread gelesen und beschrieben werden
  - wird idR vermieden, da der Zugriff extrem langsam ist

Algorithm 3.3: getMinDist(Set of ISTs R, point cloud P)

```
1: globalMinDistance[1] = maximum float
2: In parallel forall the ISTs r ∈ R do
3:     In parallel forall the Points p ∈ P do
4:         traverseIST( Root sphere of r, p )
```

Paralleler Algorithmus

Traversierung

- Race-Condition bei Blättern
  - gelöst durch atomic Integer Operator (float als int interpretieren)
  - Ordnung bleibt erhalten

Algorithm 3.4: traverseIST(Sphere s, point p, distance d)

```
1: if s is Leaf then
2:     atomicMin( d, globalMinDistance )
3: forall the Children s_c of s do
4:     d = distance( s_c, p )
5:     if d < minGlobalDistance then
6:         traverseIST( s_c, p, d )
```

Implementierung

Rekursion

- Wird seit CUDA 3.1 unterstützt
- gilt im Allgemeinen als zu langsam, durch die zusätzliche Divergence
- Iterative Traversierung implementiert:

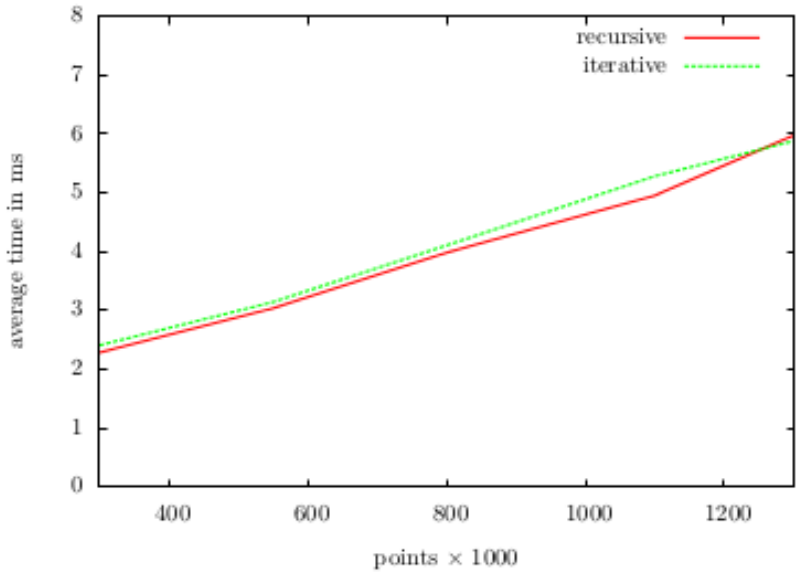
Algorithm 4.1: traverseIST(Tree t, Point p)

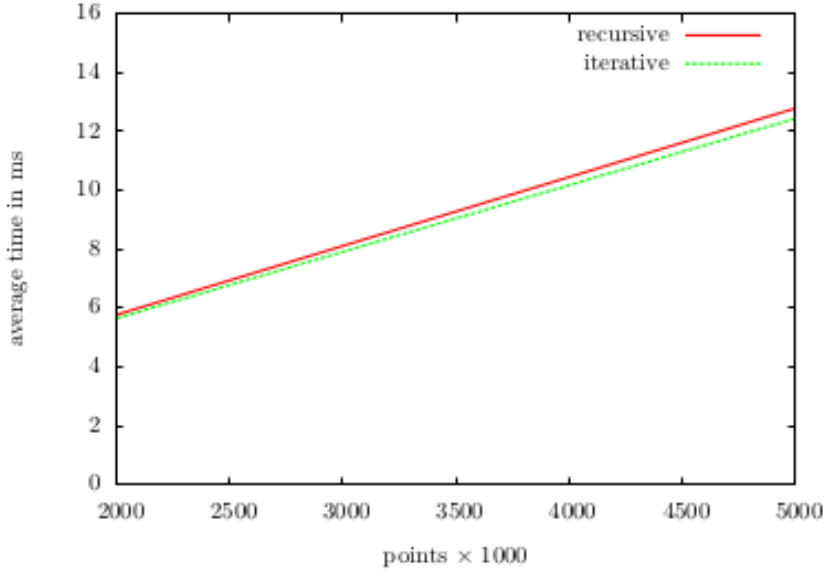
```
1:  Stack stack
2:  stack.push( root of t )
3:  while stack not empty do
4:      Sphere s = stack.pop()
5:      forall the Children s_i of s do
6:          d = distance( s, p )
7:          if d < globalMinDistance then
8:              if s_i is leaf then
9:                  atomicMin( d, globalMinDistance )
10:             else
11:                 stack.push( s_i )
```

Implementierung - Rekursion

Ergenis

- keine Verbesserung!

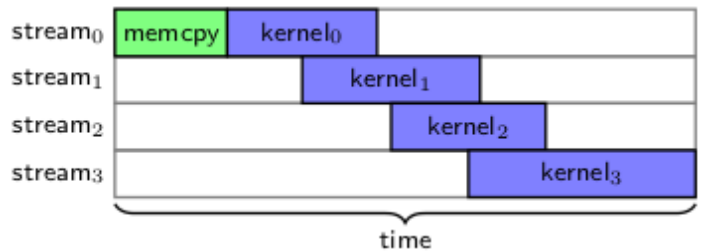




Implementierung

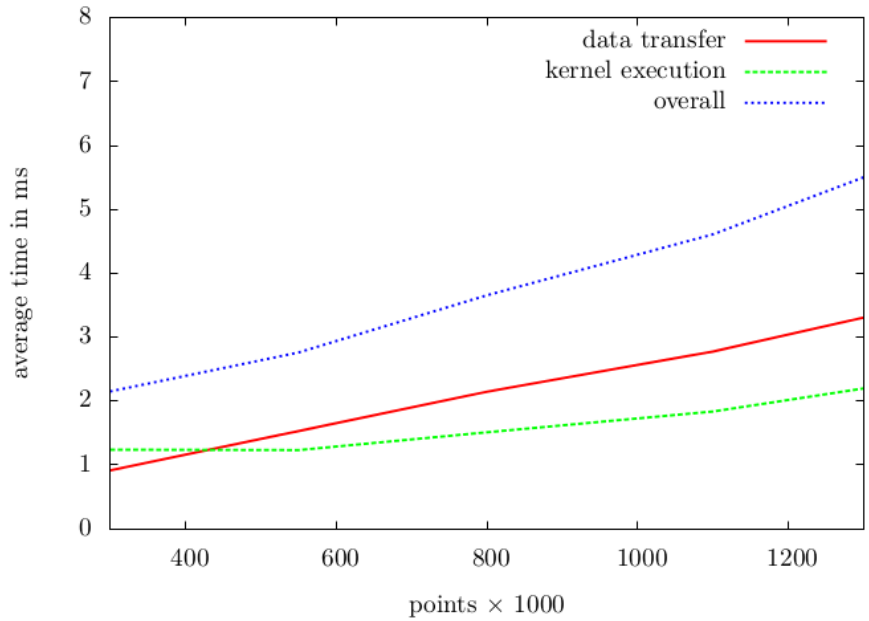
Streams

- Datenabhängigkeiten sind mit einem Stream nicht korrekt dargestellt
- jeder Kernel hängt nur von einem Datentransfer ab



Implementierung - Streams

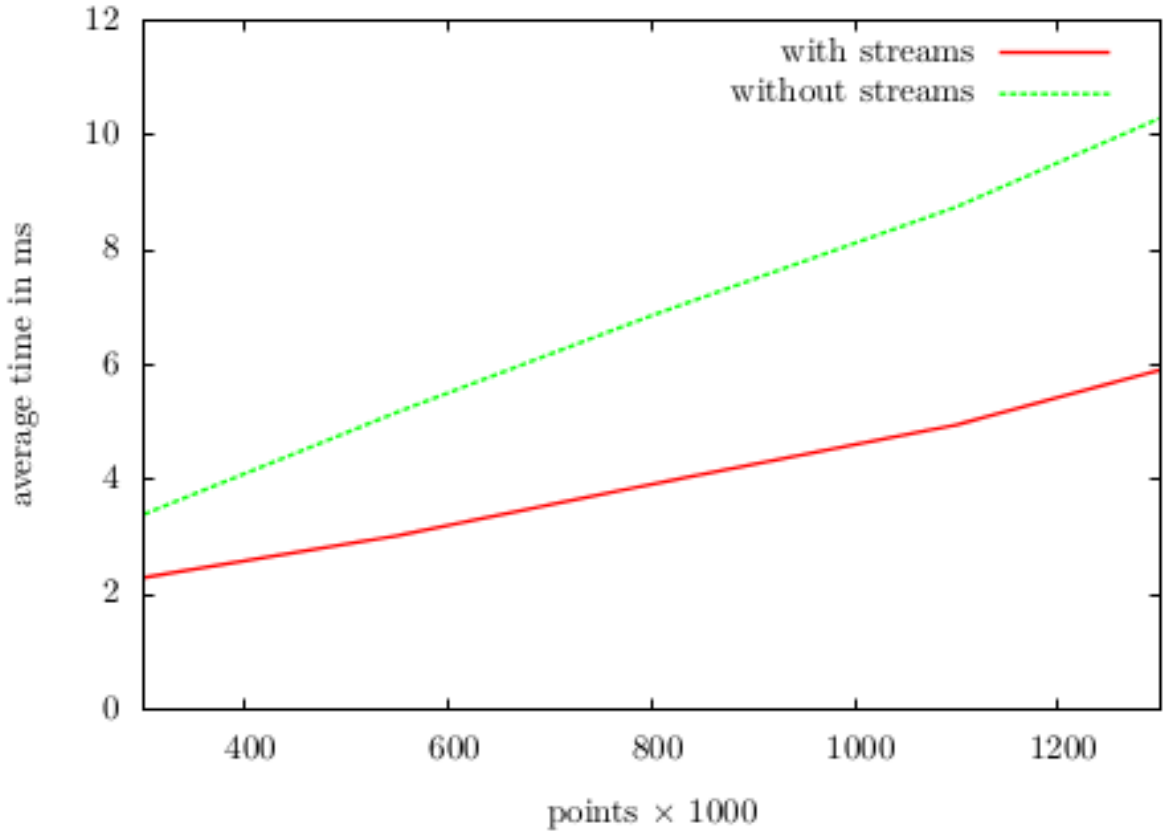
- Auch nützlich wenn Punktwolken für bestimmte Robotoerteile vorgefiltert wurden
  - zB durch einen Octree
- Grafikkarten können gleichzeitig Daten transferrieren und Instruktionen ausführen
  - gleichzeitig zu jedem Kernel wird der Transfer für den nächsten Kernel durchgeführt
- Bottleneck: Datentransfer



Implementierung - Streams

Ergebnis

- Speedup von fast 2



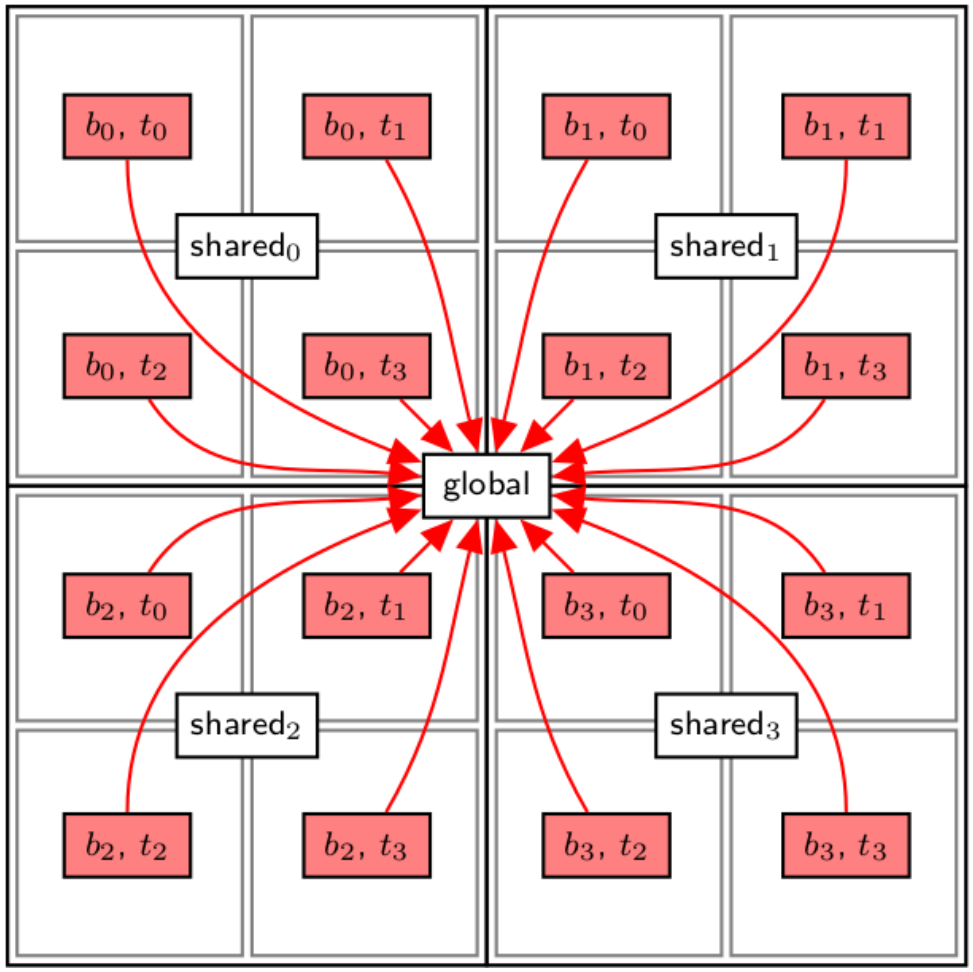
Implementierung

Shared Memory

- Idee: Keine globale Variable benutzen für die untere Schranke
  - Lesen und Schreiben extrem langsam bei globalem Speicher
- Shared Memory: Speicher den sich sog. Blöcke teilen
  - extrem schnell, praktisch kostenlos
  - ca. 100 mal schneller als globaler Speicher

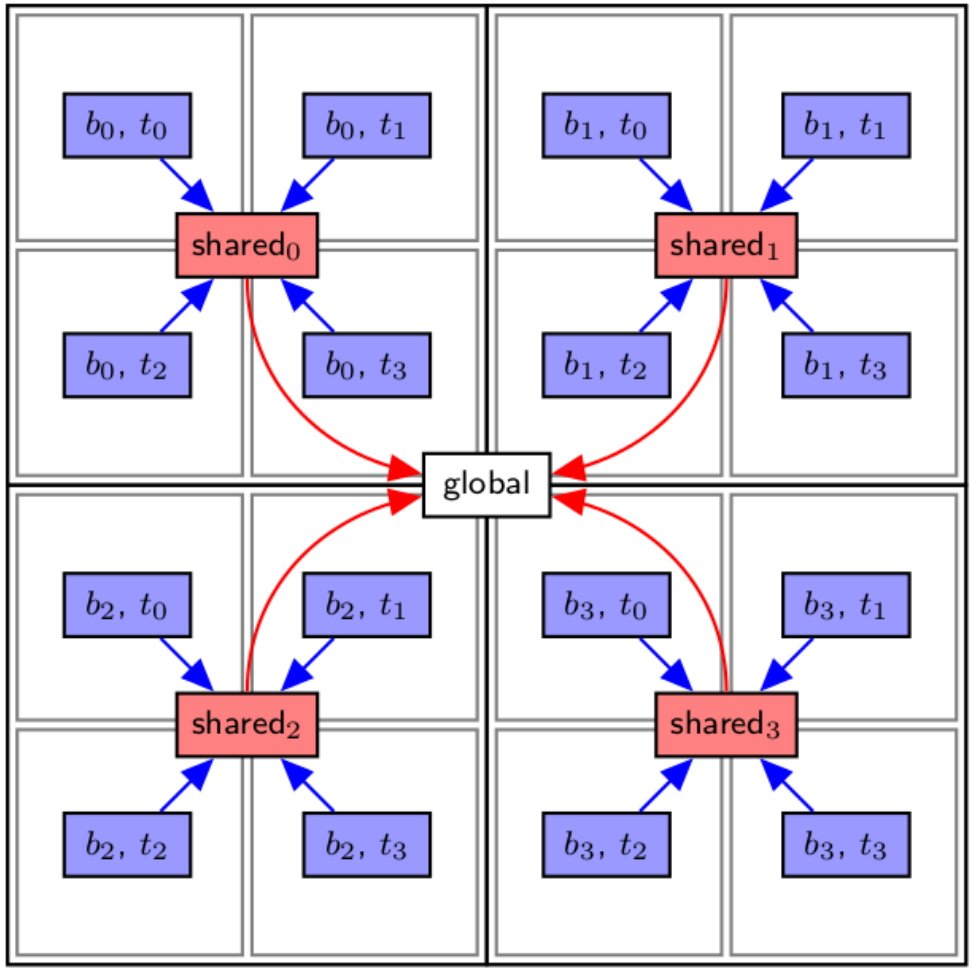
Implementierung - Shared Memory

Speicherzugriffe mit globalem Speicher



Implementierung - Shared Memory

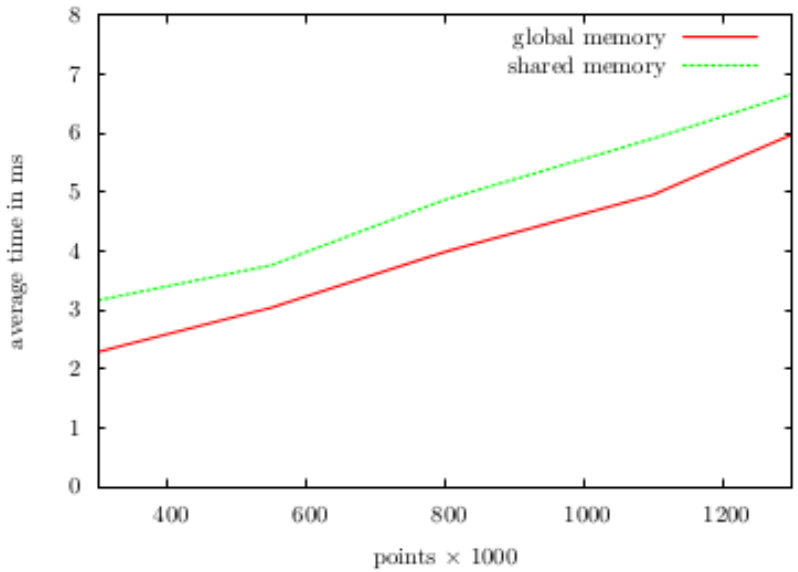
Speicherzugriffe mit Shared Memory

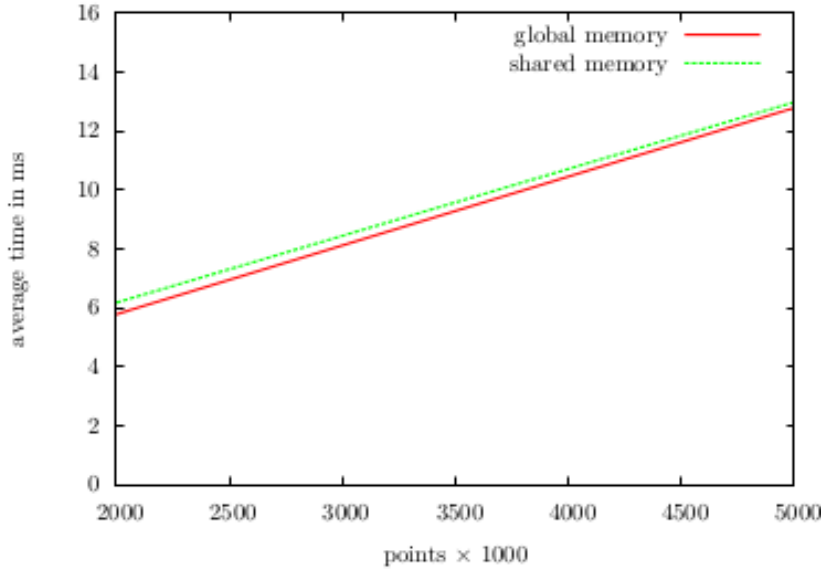


Implementierung - Shared Memory

Ergebnis

- Langsamer!



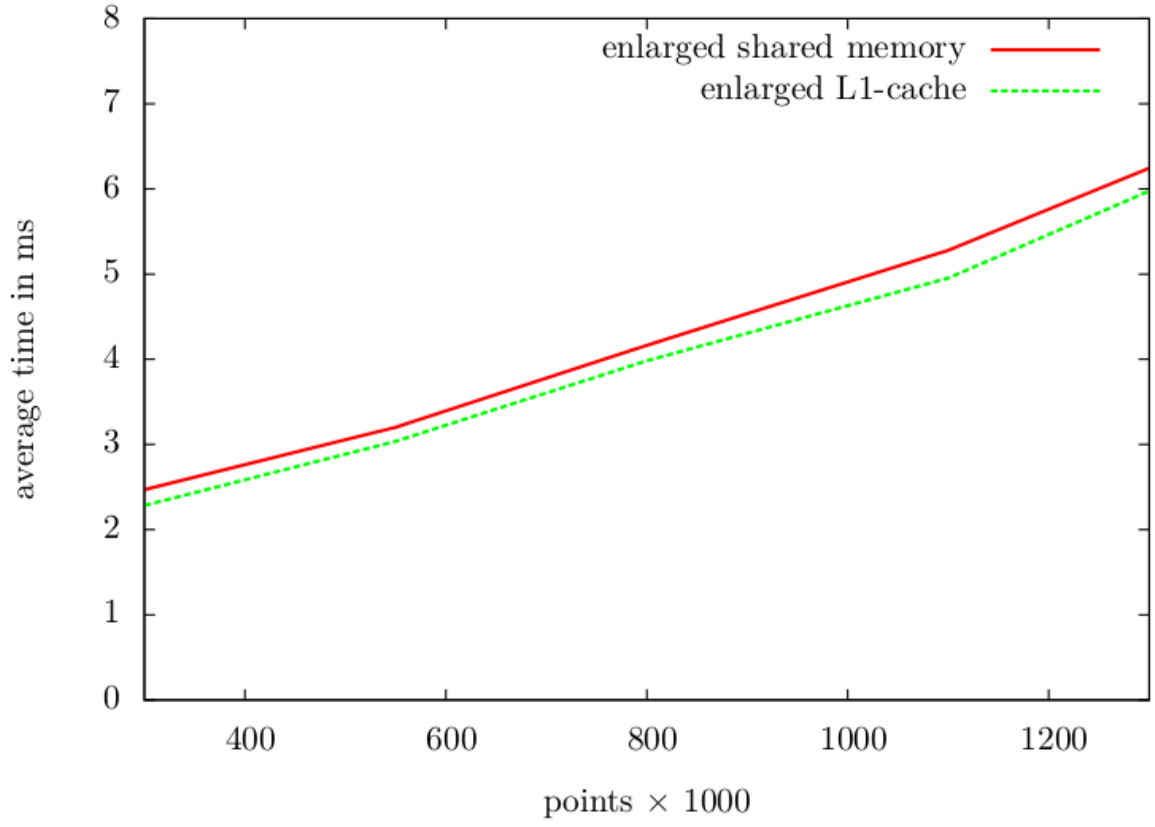


- Erklärung: Wahrscheinlich gibt es zu wenig Schreibzugriffe auf den untere Schranke, es lohnt sich mehr effektiver zu cullen
- möglicherweise Vorteilhaft wenn man die ISTs anders aufbaut
  - weniger inner Knoten, größerer Verzweigungsgrad

Implementierung

Cache

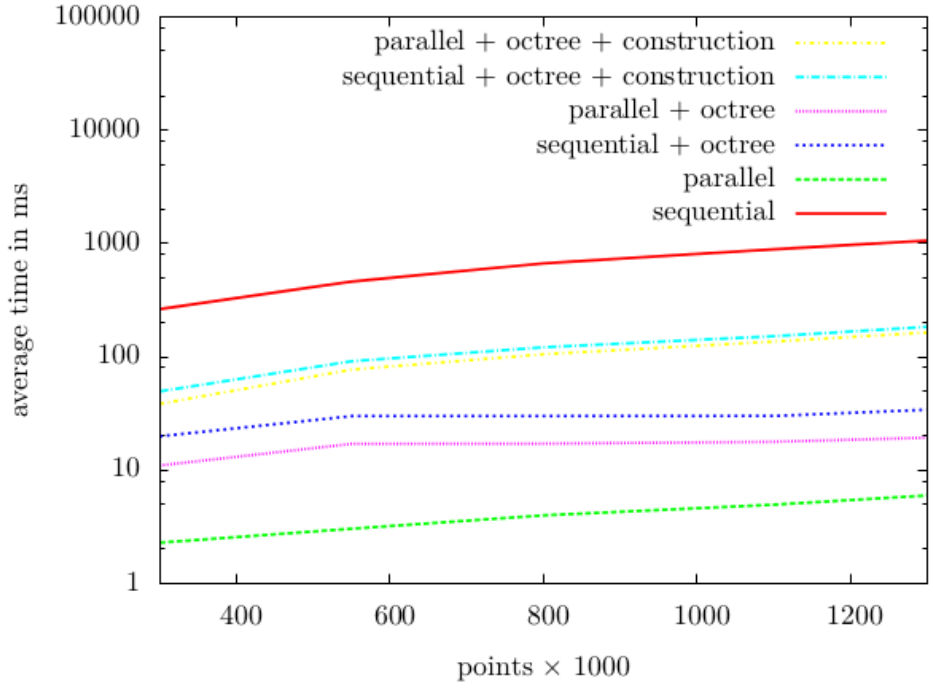
- Shared Memory wird für allgemeinen Cache verwendet
  - verschnellert manche Lesezugriffe auf globale untere Schranke



Results

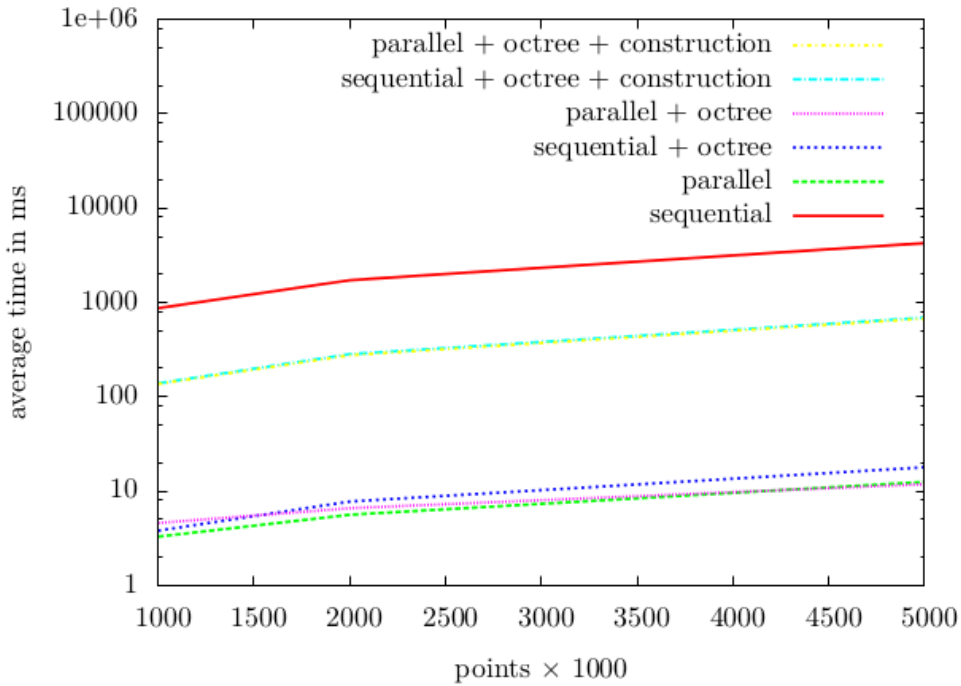
Allgemeine Messungen

- Bei Punktwolken die durch Kinect aufgenommen wurden:



Results - Allgemeine Messungen

- Bei künstlichen Punktwolken die an ein 3D-Mesh angenähert wurden:



Conclusion

- Paralleler Algorithmus ist in günstigen Fällen um eine Größenordnung schneller
  - noch mehr wenn man den Aufbau von BVHs einberechnet
- Zeiten sind Realtime fähig
- benutzt keine Preprocessing
  - wie zB den Aufbau eines Octrees
- einfach zu implementieren

Future Work

- für Haptisches Feedback müssen Anfragen in 1ms bearbeitet werden
- Mögliche Verbesserung: angepasste BVH für Punktwolken
  - BVH kann grob filtern, da viele Punkte verzeihbar sind
  - dafür müssten Anfragen und Aufbau extrem schnell sein
  - Auf der GPU, damit der Bottleneck Datentransfer nur einmal auftritt