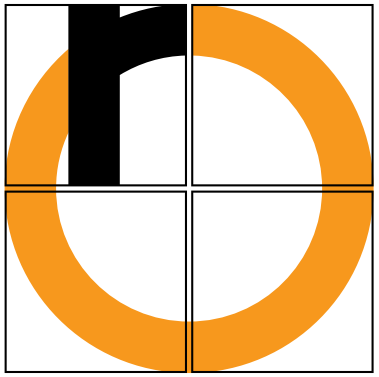
Hochschule **Rosenheim**
University of Applied Sciences



Bachelor-Thesis

Vergleich von Strategien zum dynamischen Auslagern großer 3D-Punktwolken für die schritthaltende Registrierung von Messdaten

Robert Hümmer

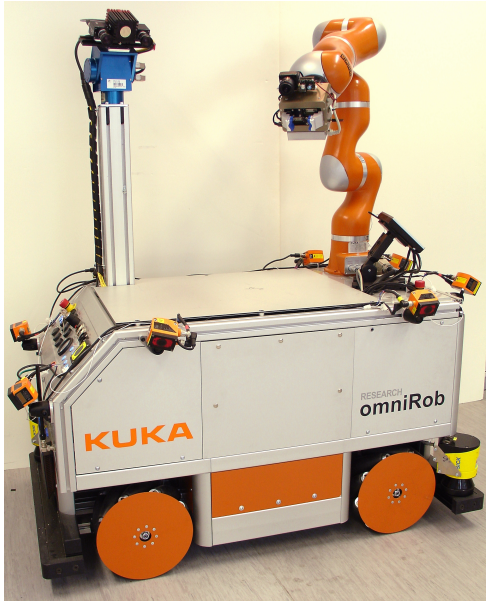
University of Applied Science Rosenheim

robert.huemmer@gmail.com

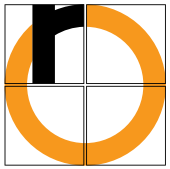
22.01.2017

Umfeld

- Roboter sollen:
  - Hindernisse vermeiden
  - Unbekannte Umgebung erkunden
  - Objekte greifen
- Wahrnehmung über Tiefenkameras
- Registrierung – Anwendungsfälle:
  - Lokalisation
  - mehrere Scans zusammenführen
- Lib3D-Framework (DLR-intern)



omniRob
[**Buschor2013**]



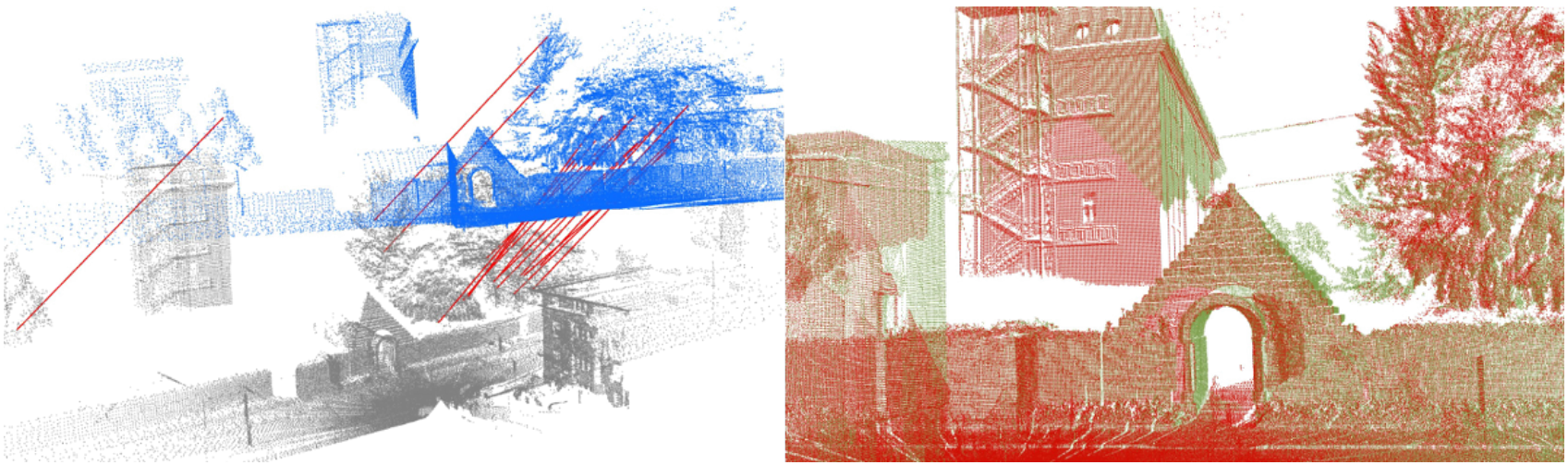
Überblick

1. Motivation
2. Lösungsansätze
3. Ziel der Arbeit
4. Caching-Strategien
5. Analyse
6. Konzepte
7. Ergebnisse
8. Zusammenfassung und Ausblick

Motivation

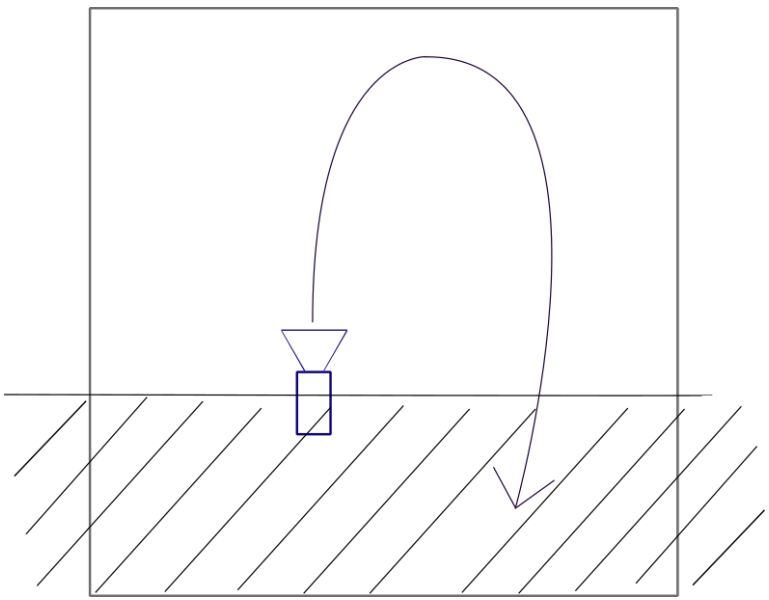
- Arbeitsspeicher ist teuer und begrenzt (Embedded Systemen)
- Vollständiges Halten der Daten im HSP nicht möglich

$\Rightarrow$ Sinnvolles Auslagern nötig

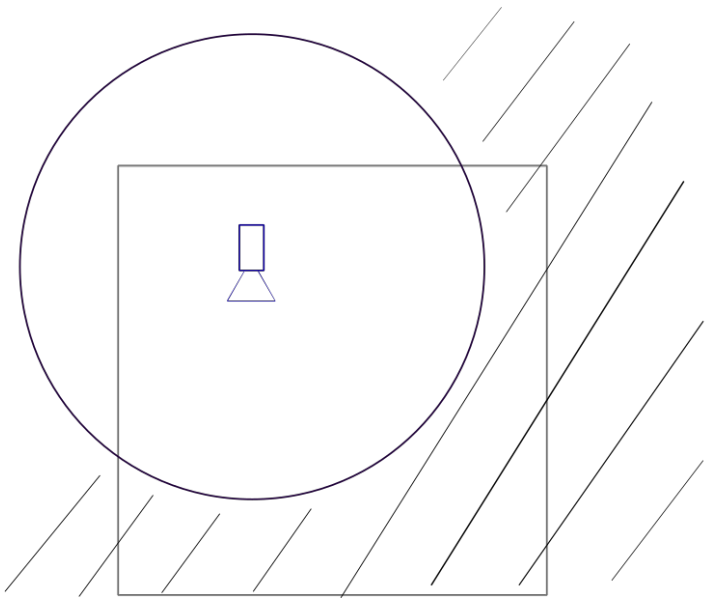


Beispiele großer Punktwolken (aus [**PCLReg**])

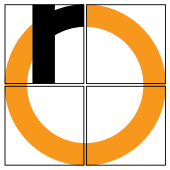
Lösungsansätze



(a) Auslagern alles hinter der Kamera liegend



(b) Auslagern alles außerhalb eines bestimmten Radius



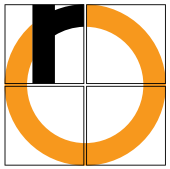
Lösungsansätze

Einsatz von

- Caching-Strategien

Anforderungen

- Eine gute Cache-Hit-Rate erzielen
- Laufzeit- sowie Speichereffizient (wenig Overhead)



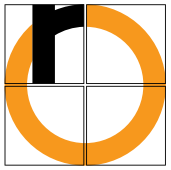
Ziel der Arbeit

Ziel:

- Sinnvolles und effizientes Auslagern mithilfe bereits existierender Caching-Strategien

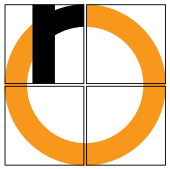
Dazu ist nötig:

- Einbringen der Strategien in den Registrierungs-Prozess (Lib3D-Framework)
- Entscheidung was genau auszulagern ist (siehe Extendable Octree)



Caching-Strategien

- Vermeidung einer Strategie, die eine offline Auswahl eines optimierbaren Parameters benötigt



Least Recently Used (LRU)

Verdrängt das Element, auf das in der Vergangenheit am seltensten zugegriffen wurde.

Vorteile

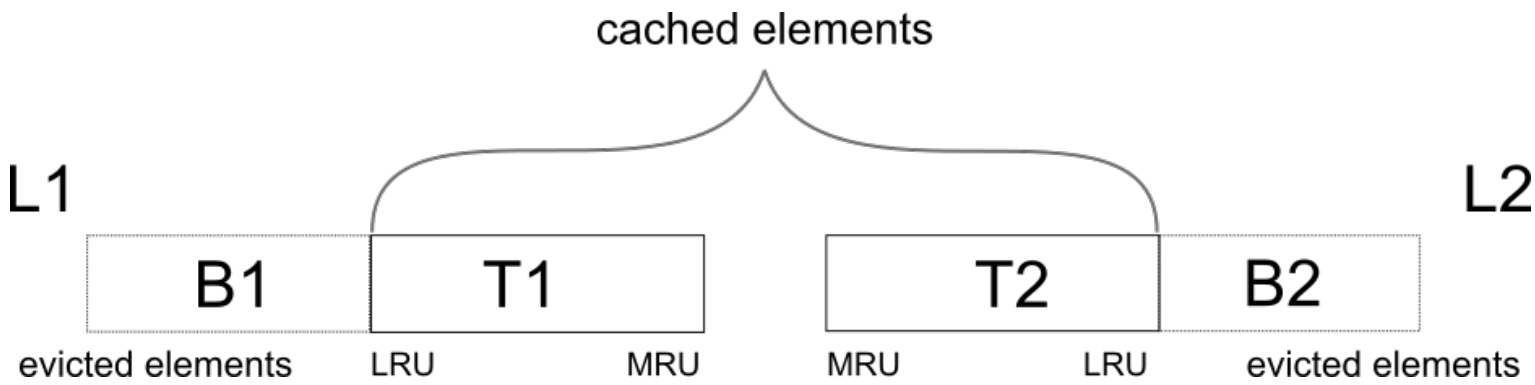
- Einfache Implementierung – nur eine Information nötig
- wenig Speicher-Overhead

Nachteile

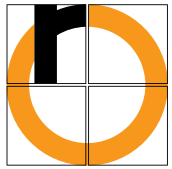
- Bezieht keine Zugriffs-Häufigkeiten mit ein

Adaptive Replacement Cache (ARC)

Datenstruktur:

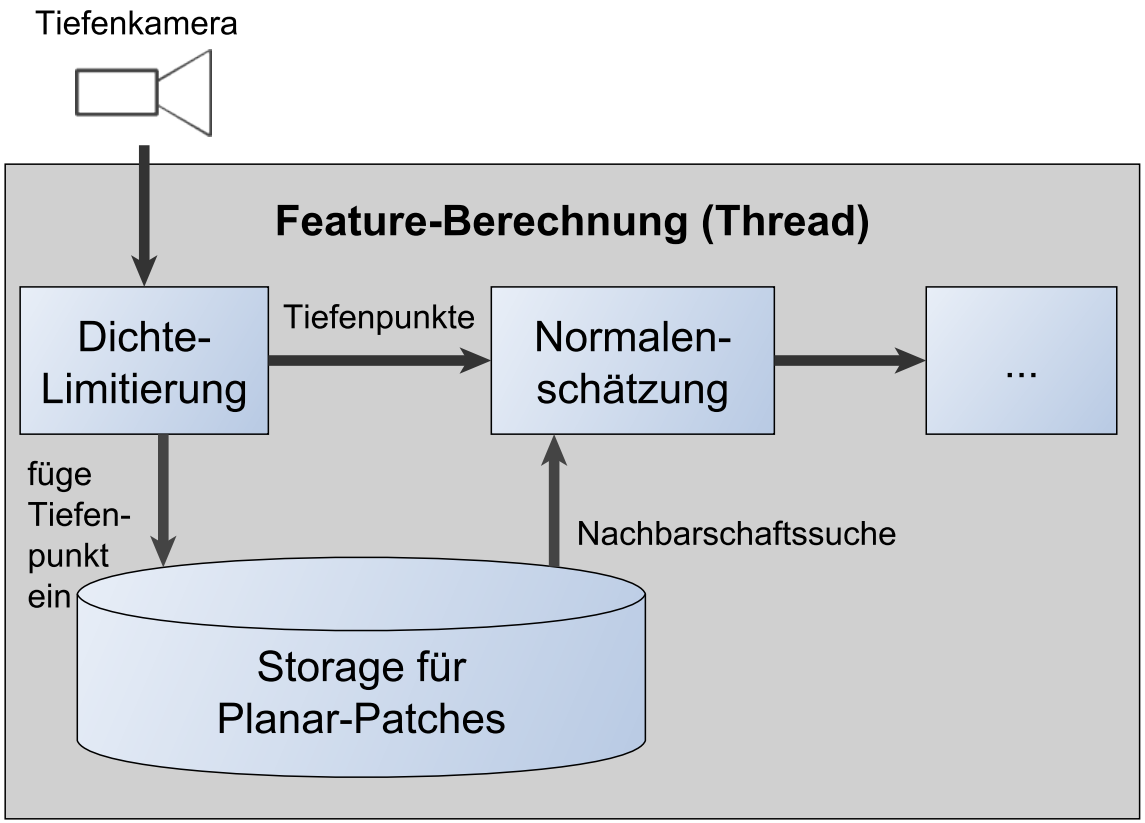


Schema der Datenstruktur von ARC (basierend auf [**megiddo2003arc**])



Überblick

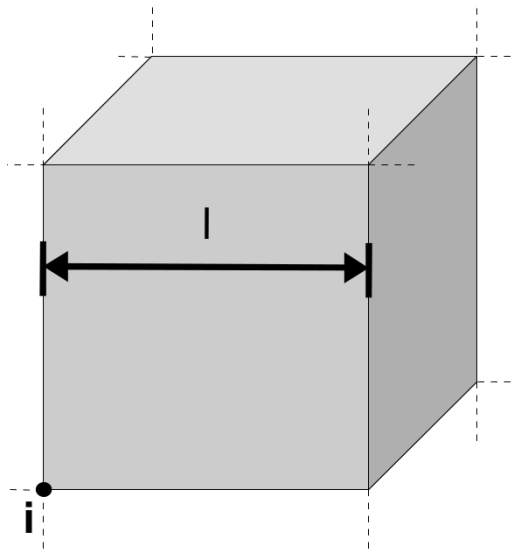
Prozess ohne Caching



Teil des streambasierten Prozesses der Feature-Berechnung

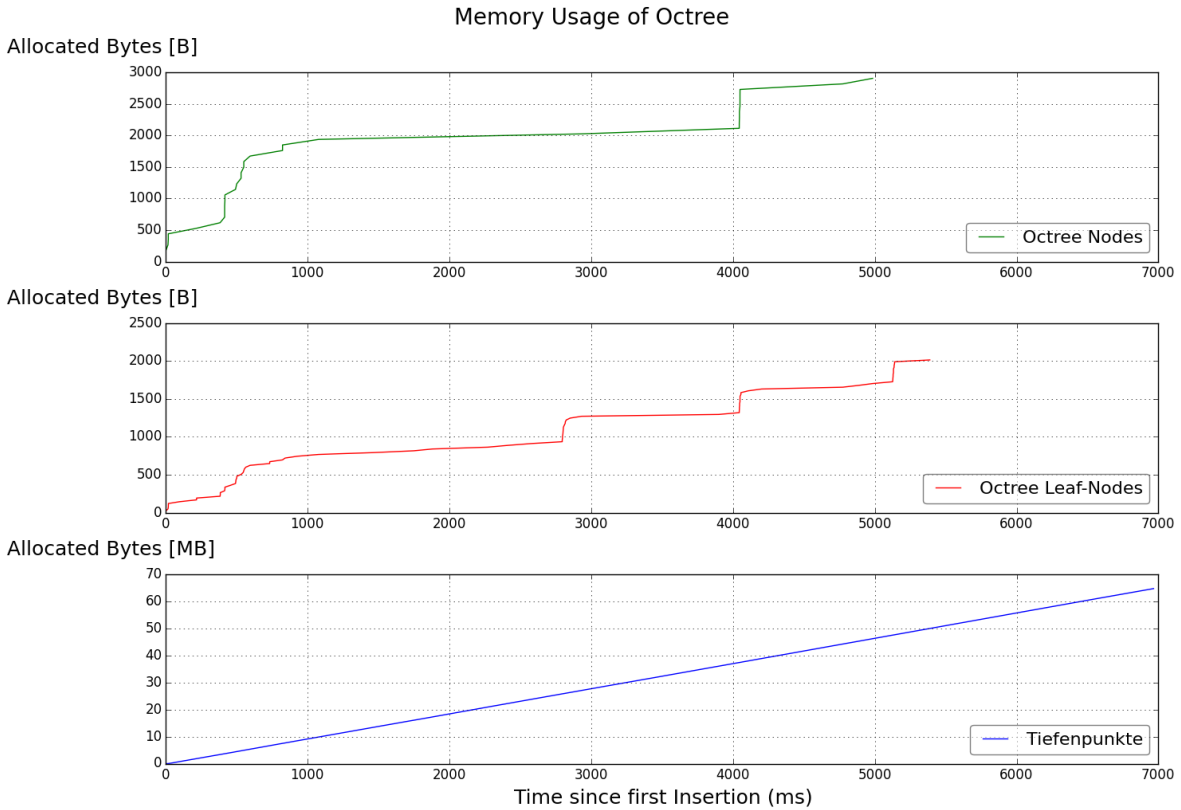
Extendable Octree

- Punkte werden in den Leaf-Nodes (Voxel) des Extendable Octrees abgelegt

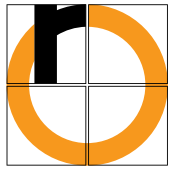


Definition eines Voxels (Abbildung aus [**Bodenmueller2009**])

Analyse des Speicherbedarfs

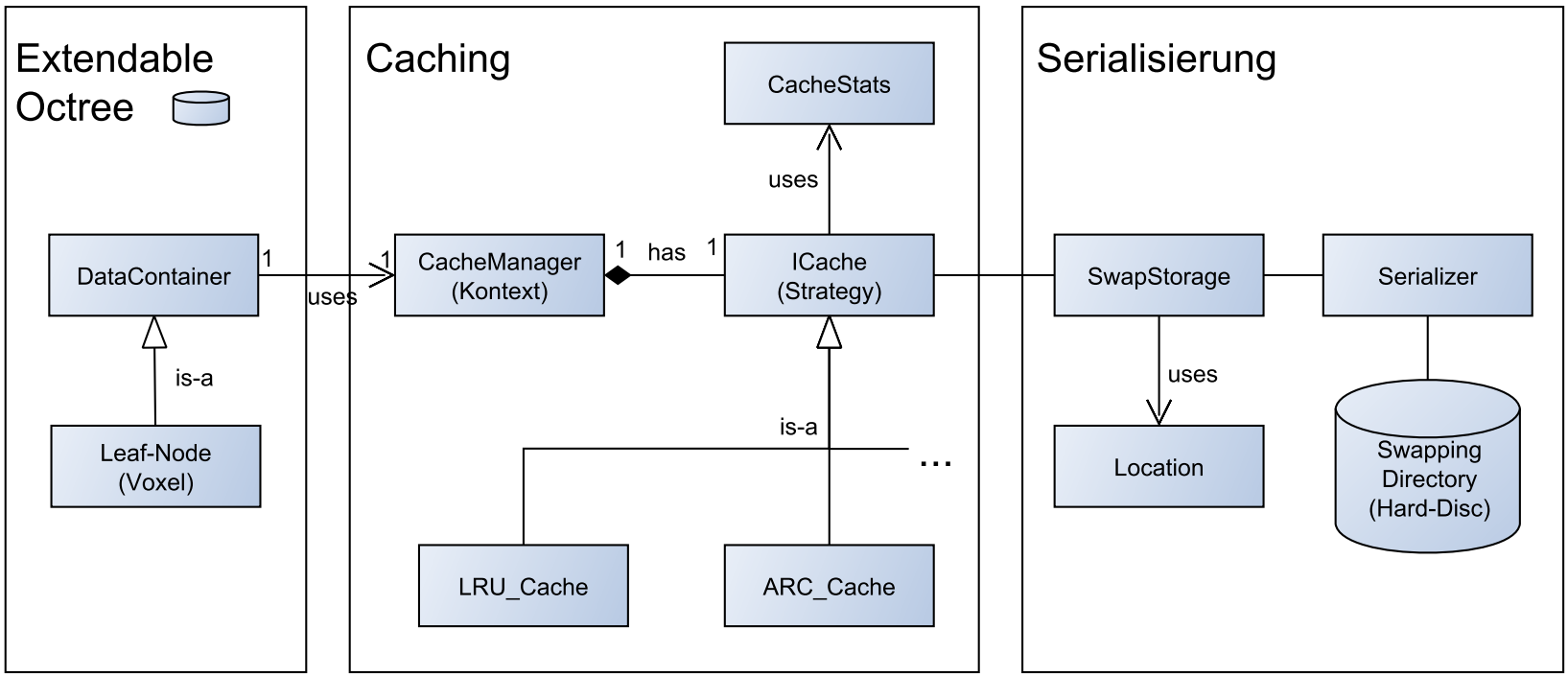


Speicherbedarf des Extendable Octree (ohne Caching)

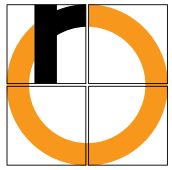


Überblick

Prozess mit Caching



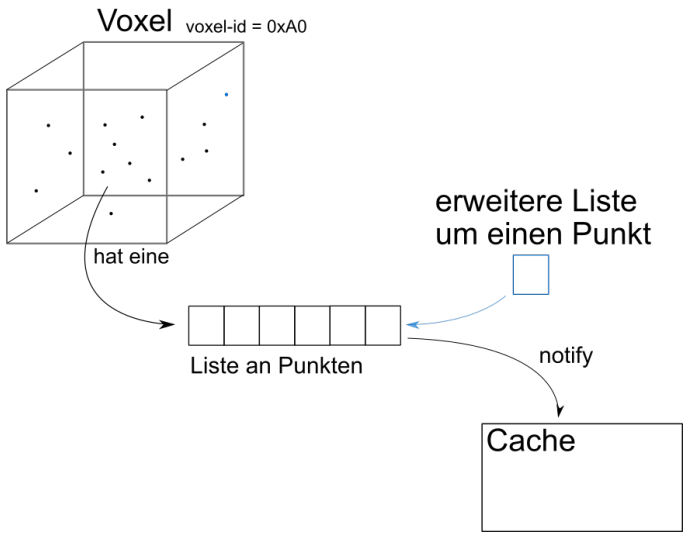
Erweiterung der Feature-Berechnung (Feature-Estimation)



Problematik dynamischer Elemente

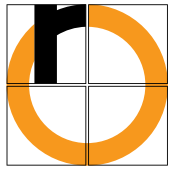
- Existierende Caching-Strategien basieren auf Elemente fester Größe
- In dieser Anwendung sind die gecachten Elemente jedoch Voxelinhalte, die sich dynamisch in ihrer Größe verändern

Update-Konzept



Schema zum Benachrichtigen des Caches bei Größenänderung

- STD: Pro Container-Typ ein zustandsloser Allokator
- EASTL: Pro Container-Instanz ein Allokator



Überblick

Ergebnisse

Vergleich der Hit-Rate

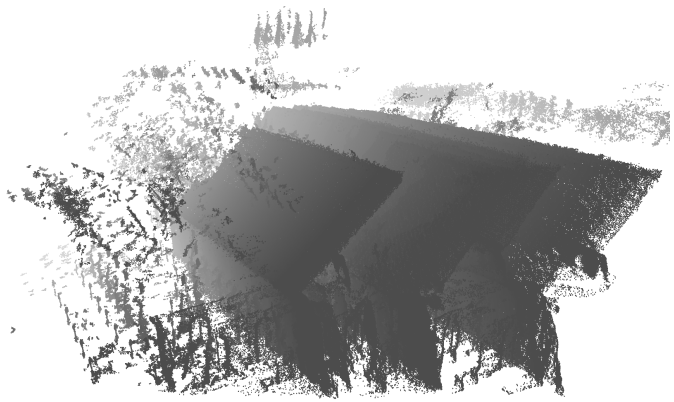
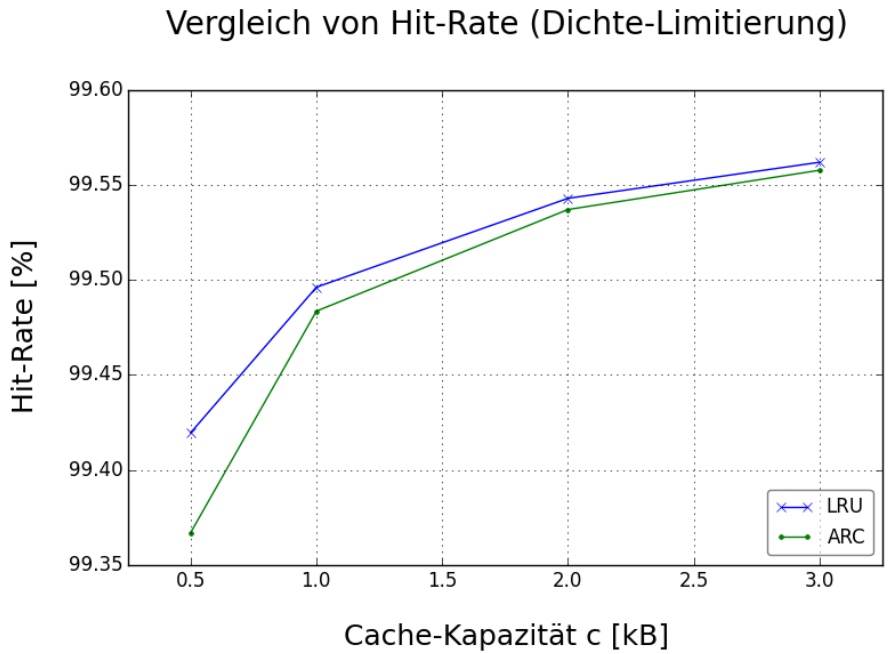


Illustration des Datensatzes *Unikirche* (3D-Punktwolke)

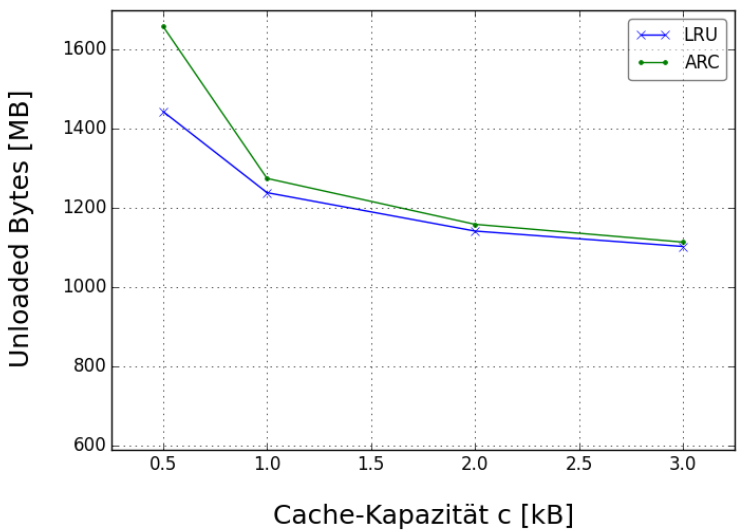


Zeigt, dass die Cache-Hit-Rate von LRU besser ist als die von ARC

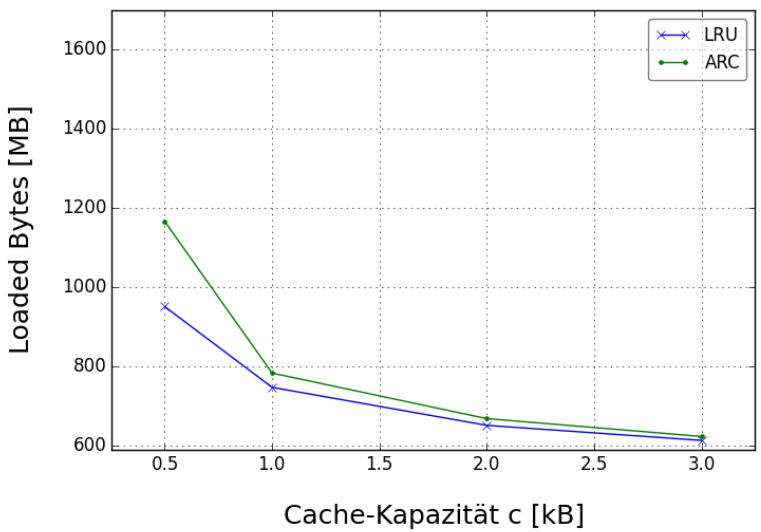
Ergebnisse

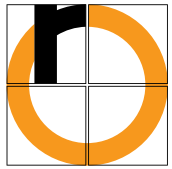
Vergleich der Swapping-Kosten

Vergleich der Kosten zum Entladen (Dichte-Limitierung)



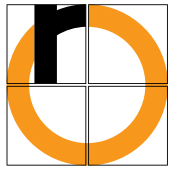
Vergleich der Kosten zum Laden (Dichte-Limitierung)





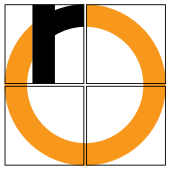
Überblick

1. Motivation
2. Lösungsansätze
3. Ziel der Arbeit
4. Caching-Strategien
5. Analyse
6. Konzepte
7. Ergebnisse
8. Zusammenfassung und Ausblick



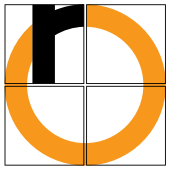
Zusammenfassung

- Auslagern der Voxelinhalte des Extendable Octrees (pro Voxelinhalt eine Binärdatei)
- Identifikation der Voxelinhalte über die Speicheradresse des Leaf-Nodes
- Update-Konzept zur Rückmeldung über den von Voxelinhalten angelegten Speicher
- ARC hat sich als schlechter herausgestellt
- Aufeinanderfolgende Zugriffe auf Voxelinhalte bei der Dichte-Limitierung sowie Feature-Berechnung sind räumlich und zeitlich nah beieinander



Ausblick

- Verwendung des Dateiformats *Hierarchical Data Format* (HDF)
- LIRS können noch untersucht werden (Metrik: IRR)



Referenzen

Ergebnisse

Testdaten

- Punktwolke einer Unikirche
- Voxel-Auflösung von 0.01
- Dichte-Limitationsradius von 0.000561231

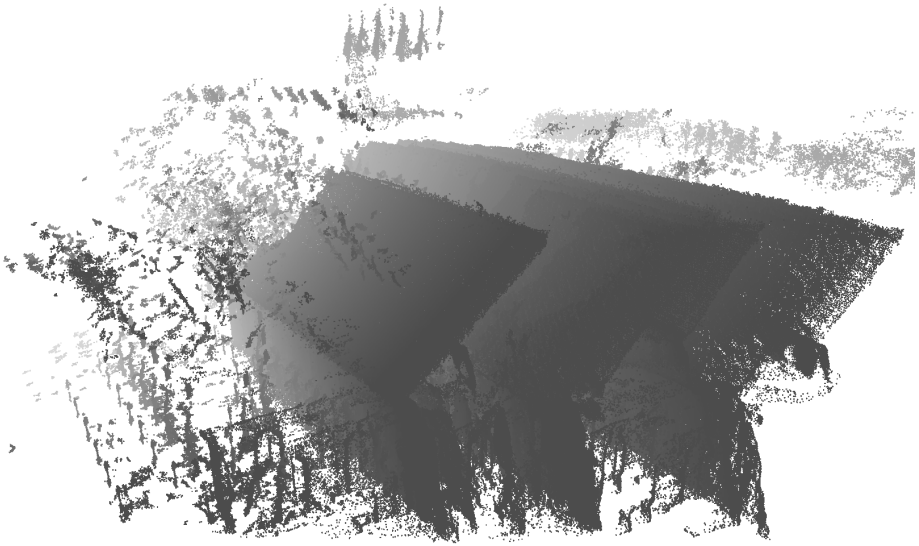


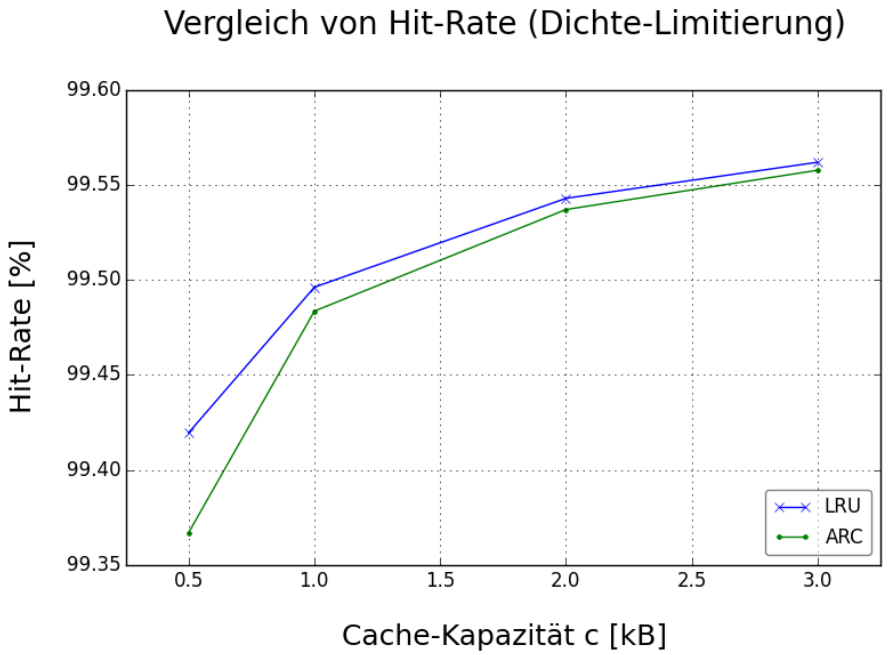
Illustration des Datensatzes *Unikirche*

Ergebnisse

Vergleich der Hit-Rate

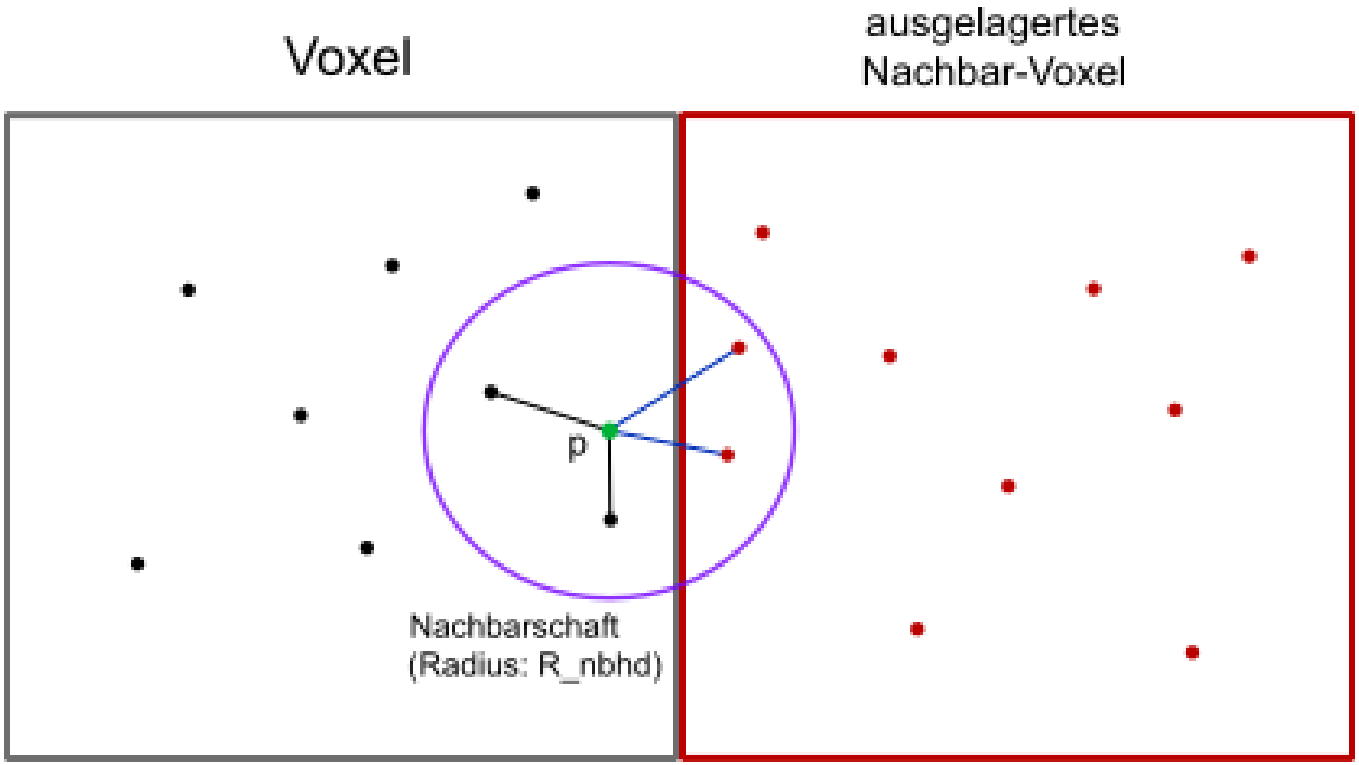
- Cache-Inserts (ohne unload):
  69.475 (entspricht gesamte Voxelanzahl)
- Cache-Gets:
  11.663.094

$\Rightarrow$ Gets pro Voxelinhalt durchschnittlich
≈ 168



Zeigt, dass die Cache-Hit-Rate von LRU besser ist als die von ARC

Problematik ungültiger Referenzen



Zeigt die Problematik ungültiger Referenzen