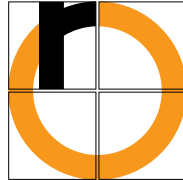




Hochschule Rosenheim
University of Applied Sciences



Bachelor-Thesis

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

Robert Hümmner

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 [Bus13]

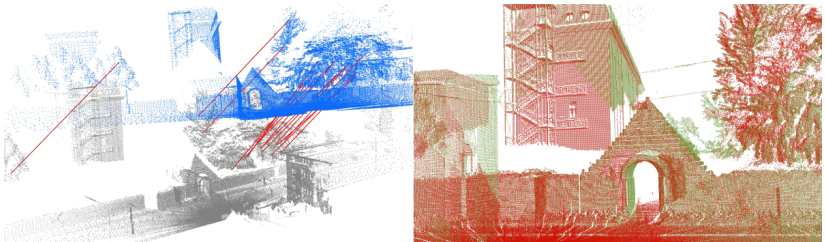


Überblick

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

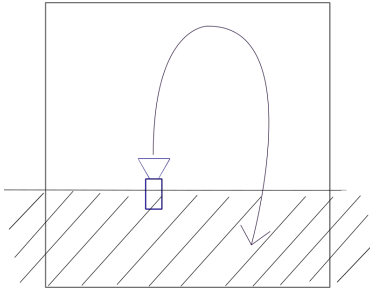
Motivation

- ▶ Arbeitsspeicher ist teuer und begrenzt (Embedded Systemen)
 - ▶ Vollständiges Halten der Daten im HSP nicht möglich
- ⇒ Sinnvolles Auslagern nötig

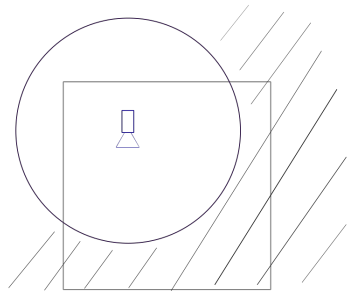


Beispiele großer Punktwolken (aus [PCL])

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

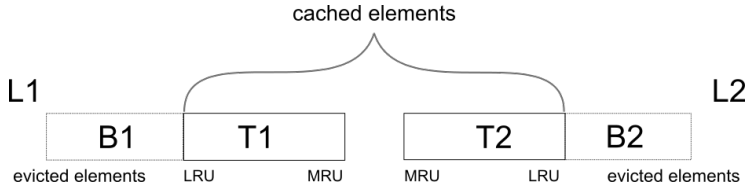
Nachteile

- ▶ Bezieht keine Zugriffs-Häufigkeiten mit ein



Adaptive Replacement Cache (ARC)

Datenstruktur:



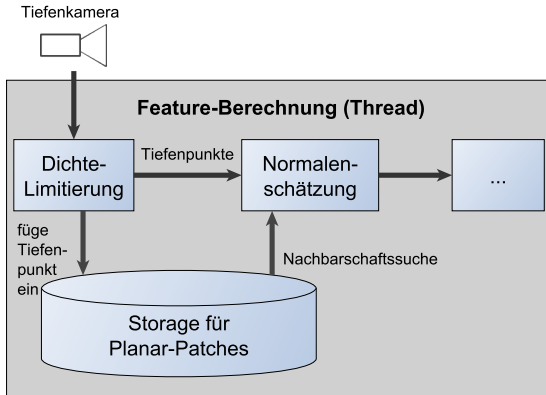
Schema der Datenstruktur von ARC (basierend auf [MM03])



Überblick

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

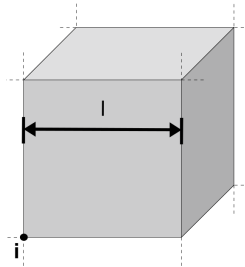
Prozess ohne Caching



Teil des streambasierten Prozesses der Feature-Berechnung

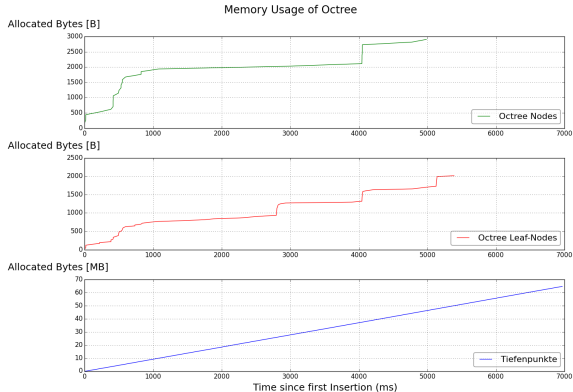
Extendable Octree

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



Definition eines Voxels (Abbildung aus [Bod09])

Analyse des Speicherbedarfs



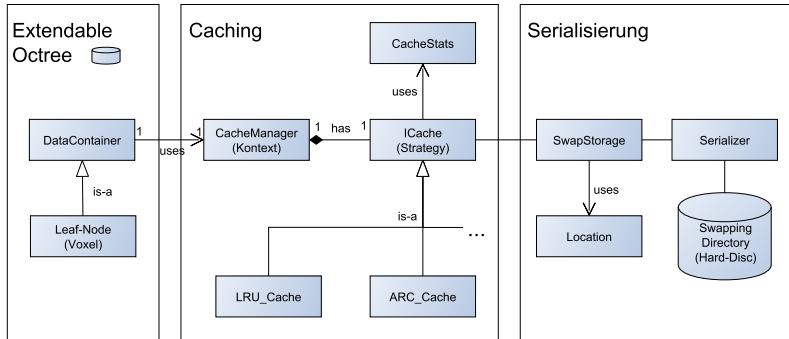
Speicherbedarf des Extendable Octree (ohne Caching)



Überblick

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

Prozess mit Caching



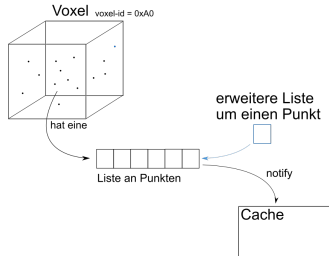
Erweiterung der Feature-Berechnung (Feature-Estimation)



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

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

Ergebnisse

Vergleich der Hit-Rate

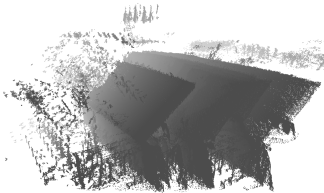
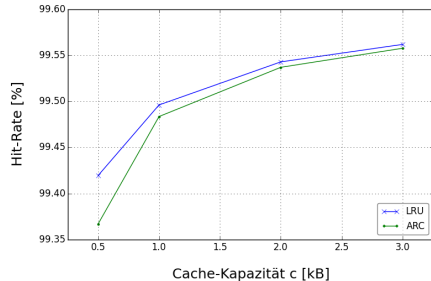


Illustration des Datensatzes
Unikirche (3D-Punktwolke)

Vergleich von Hit-Rate (Dichte-Limitierung)

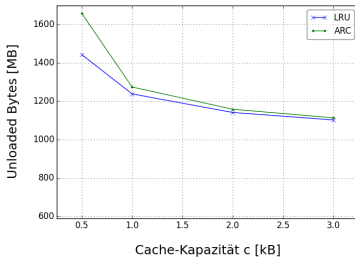


Zeigt, dass die Cache-Hit-Rate von LRU besser ist als die von ARC

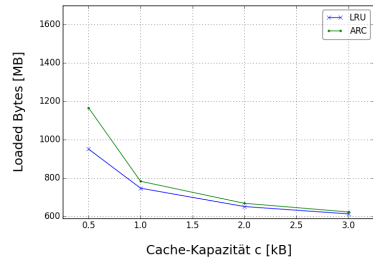
Ergebnisse

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

- [Bod09] Tim Bodenmüller. „Streaming Surface Reconstruction from Real Time 3D Measurements“. Dissertation. Munich: Technische Universität München, 2009.
- [Bus13] Christopher Peter Ulrich Buschor. „Next-Best-View-Planning auf Voxelkarten zur Exploration durch einen mit Tiefensensoren ausgestatteten mobilen Roboter“. Bachelorarbeit. Technische Universität München, Aug. 2013.
- [Che98] Ludmila Cherkasova. *Improving WWW proxies performance with greedy-dual-size-frequency caching policy*. Hewlett-Packard Laboratories, 1998.
- [KLW94] Ramakrishna Karedla, J Spencer Love und Bradley G Wherry. „Caching strategies to improve disk system performance“. In: *Computer* 27.3 (1994), S. 38–46.
- [MM03] Nimrod Megiddo und Dharmendra S Modha. „ARC: A Self Tuning, Low Overhead Replacement Cache.“. In: *FAST*. Bd. 3. 2003, S. 115–130.

Ergebnisse

Testdaten

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

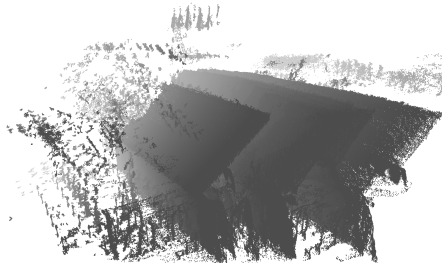


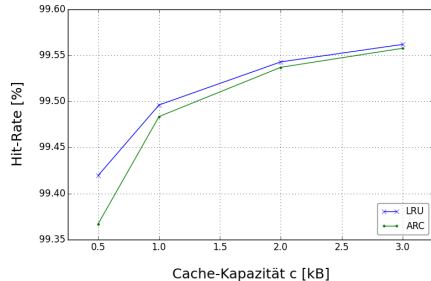
Illustration des Datensatzes *Unikirche*

Ergebnisse

Vergleich der Hit-Rate

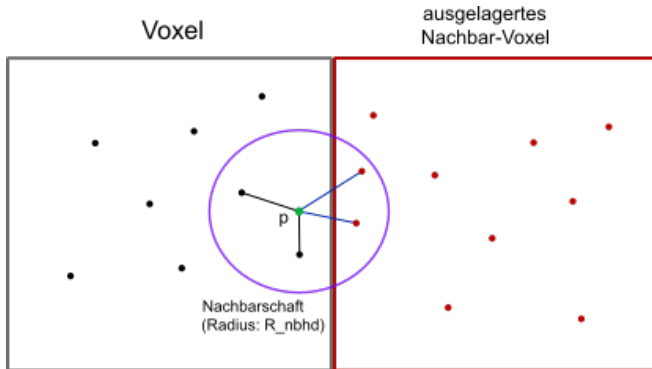
- Cache-Inserts (ohne unload):
69.475 (entspricht gesamte Voxelanzahl)
 - Cache-Gets:
11.663.094
- ⇒ Gets pro Voxelinhalt
durchschnittlich
 ≈ 168

Vergleich von Hit-Rate (Dichte-Limitierung)



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

Problematik ungültiger Referenzen



Zeigt die Problematik ungültiger Referenzen