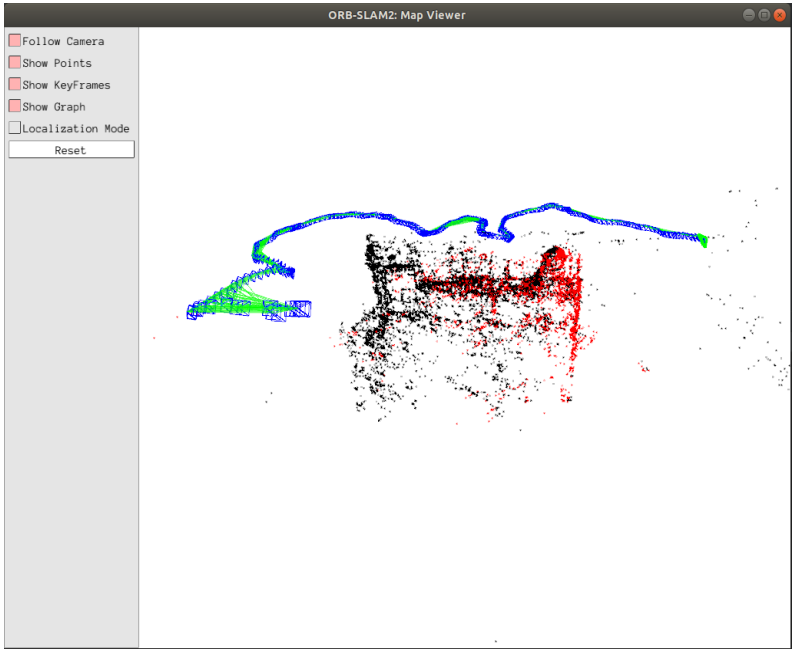
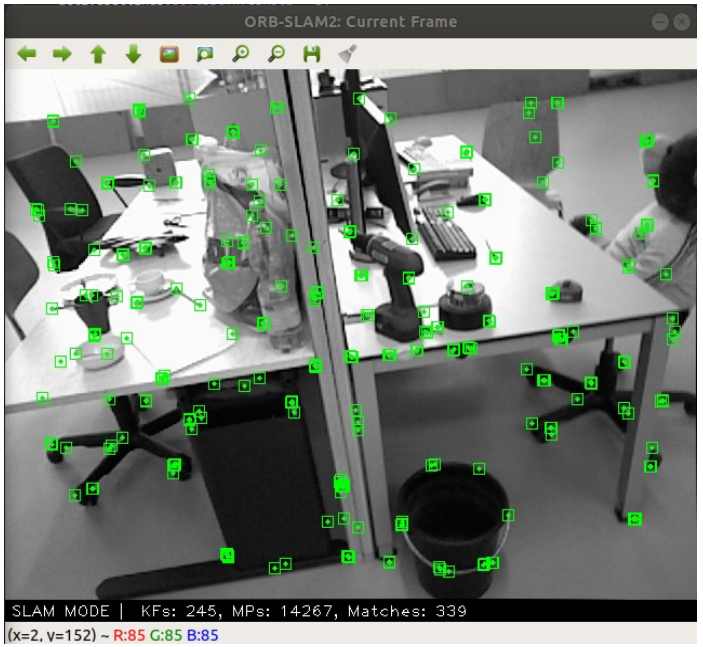
Optimizing Feature Detection and Keypoint Distribution for real-time SLAM

Bachelorarbeit
Ralph Gelnar
19.07.2019

SLAM:
simultaneous **l**ocalization **a**nd **m**apping

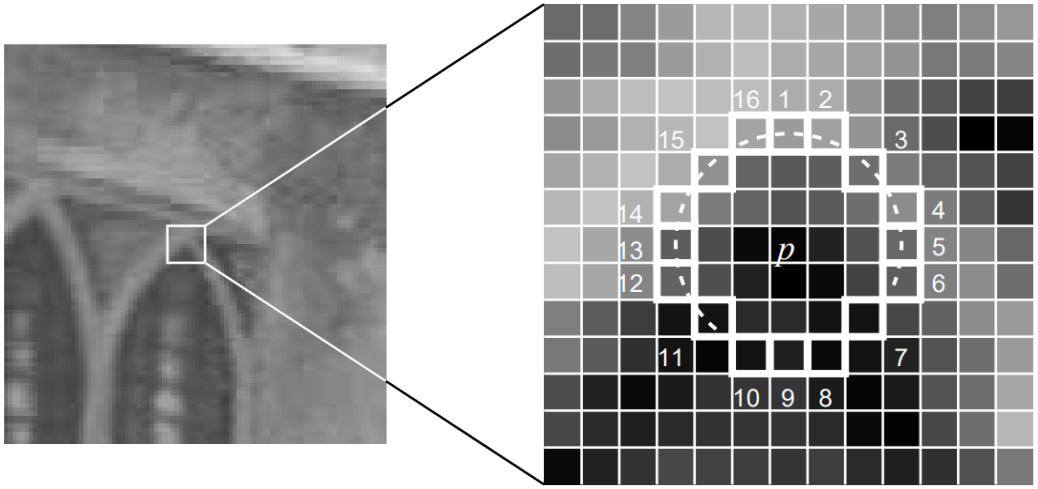
- Problem: aus einer Serie von Sensor-Inputs soll die Position des Agenten und eine Karte der relevanten Umgebung berechnet werden
- Hier: Input ist (Stereo-)Video, depth map
- Pro Video-Frame werden Features (Ecken) erkannt, aus den Matches Frame-zu-Frame und deren Veränderung dann die Position der Kamera rekonstruiert
- Genutztes System, in das meine Feature-Detection integriert wurde: ORB-SLAM2





Feature Detection

ORB: Oriented FAST and rotated BRIEF



FAST: Pixel ist Ecke falls ein ~Halbkreis von kontinuierlichen Pixeln existiert mit $I_x > I_p + threshold$ oder $I_x < I_p - threshold$

- Skalierungsinvarianz durch Konstruktion einer Skalierungspyramide und Detektion pro Level
- Rotationsinvarianz durch Winkelberechnung der Umgebung (per Intensitätszentroid) und Anwendung der errechneten Winkel vor Berechnung der Deskriptoren
- Matching durch Vergleich von Bitvektoren, die Vergleiche von verschiedenen Intensitäten in der (geglätteten) Umgebung eines erkannten Keypunktes beschreiben

Feature Distribution

- Mit typischen Thresholds (20/7) werden pro Megapixel ca. 10.000 – 20.000 Keypunkte erkannt
- Optimale Anzahl fürs Matching (je nach Videoauflösung): ca. 1000 Keypunkte

  → Welche Keypunkte sollen behalten werden?

Während FAST wird pro erkanntem Keypunkt eine Cornerness-Score berechnet:

$$max\left(min\left(I_p - I_k, \ldots, I_p - I_{k+9}, threshold\right), max\left(I_p - I_k, \ldots, I_p - I_{k+9}, -threshold\right)\right), k \in \{0, 2, \ldots, 16\}$$

→ verschiedene Ansätze diese Punktzahl zur Auswahl der N geeignetesten Features zu nutzen:

- Naiver Ansatz
- Bucketing
- Quadtree
- KD-Tree ANMS
- Range-Tree ANMS
- Suppression via Square Covering (SSC)
- SSC-Variante: Soft SSC
- Reverse Grid Supression

Unabhängig von der gewählten Methode werden die Features pro Ebene der Skalierungspyramide verteilt

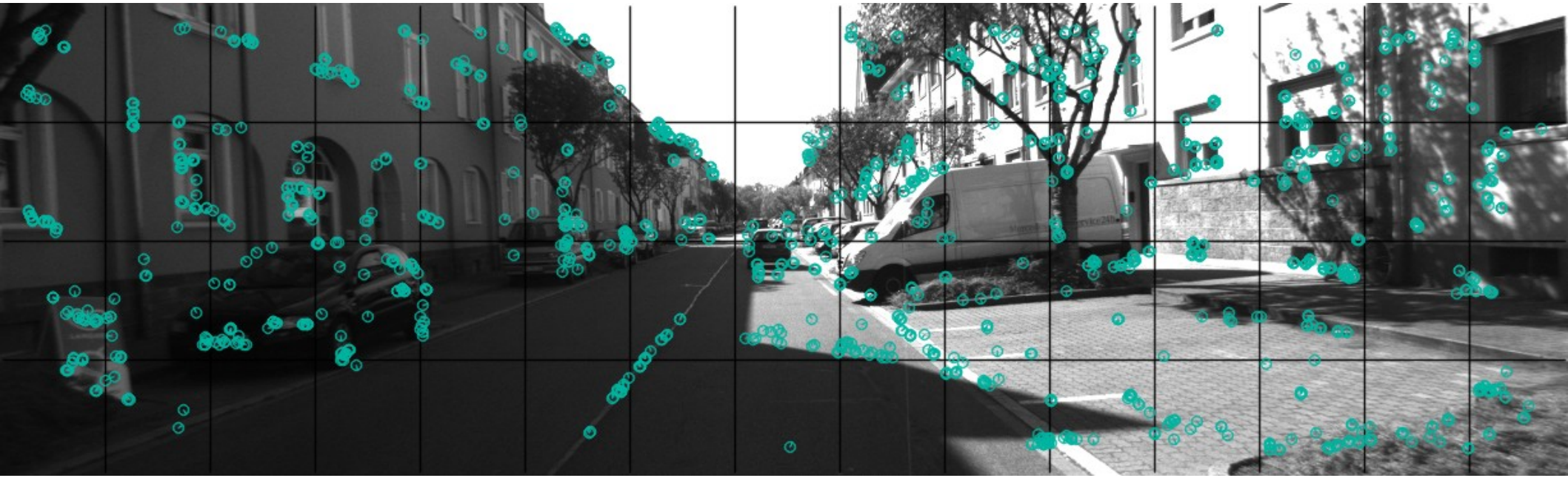
Naiver Ansatz: Top N

Die nach Punktzahl besten N Keypunkte werden behalten

Problem: Unabhängig von der Bilddomäne starke Häufung von Features an markanten Stellen
→viel reduntante Information, große Bereiche des Bildes nicht abgedeckt

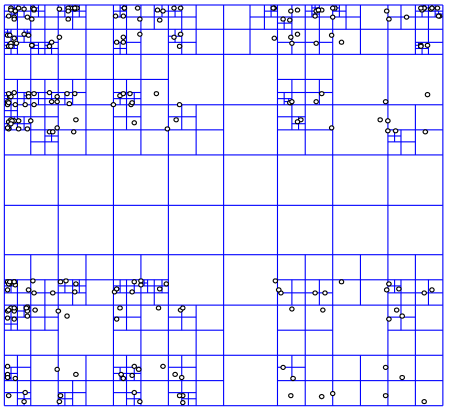
Bucketing

- Einteilung des Bildes in Zellen der Größe x*x mit x = min(80, height, width)
- Pro Zelle werden N / Zellen Features behalten
- Sehr einfach, liefert trotzdem gute Ergebnisse



Quadtree

- Jeder Knoten wird weitergeteilt bis entweder die Anzahl der Knoten = N ist oder jeder Knoten nur einen Keypunkt enthält
- Von ORB_SLAM2 verwendete Methode



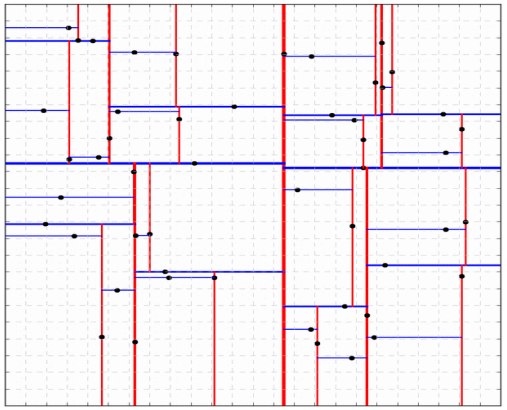
Adaptive non-maximal Suppression-Ansätze

Bester Keypunkt aus dem Input wird gewählt, in ROI (je nach Methode unterschiedlich gewählt) um Keypunkt herum werden andere Features ausgeschlossen:

```
Input: Gefundene Features P
Sortiere P nach Corner-Score
While !(N-ε < |P| < N+ε):
    Markiere alle p_i in P als legal
    Output = ∅
    Für alle p_i in P:
        Falls p_i  legal:
            Füge p_i zu Output hinzu
            Markiere alle p_j in ROI um p_i als illegal
    Update ROI
Return Output
```

KD-Tree ANMS

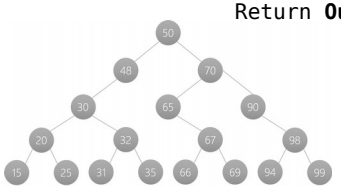
- Um die Punkte in der ROI um den aktuellen Keypunkt zu finden wird ein KD-Tree konstruiert
- Keypunkte werden per Radiussuche um den aktuellen Punkt ausgeschlossen
- Anpassung des Suchradius bis N-ε < P < N+ε



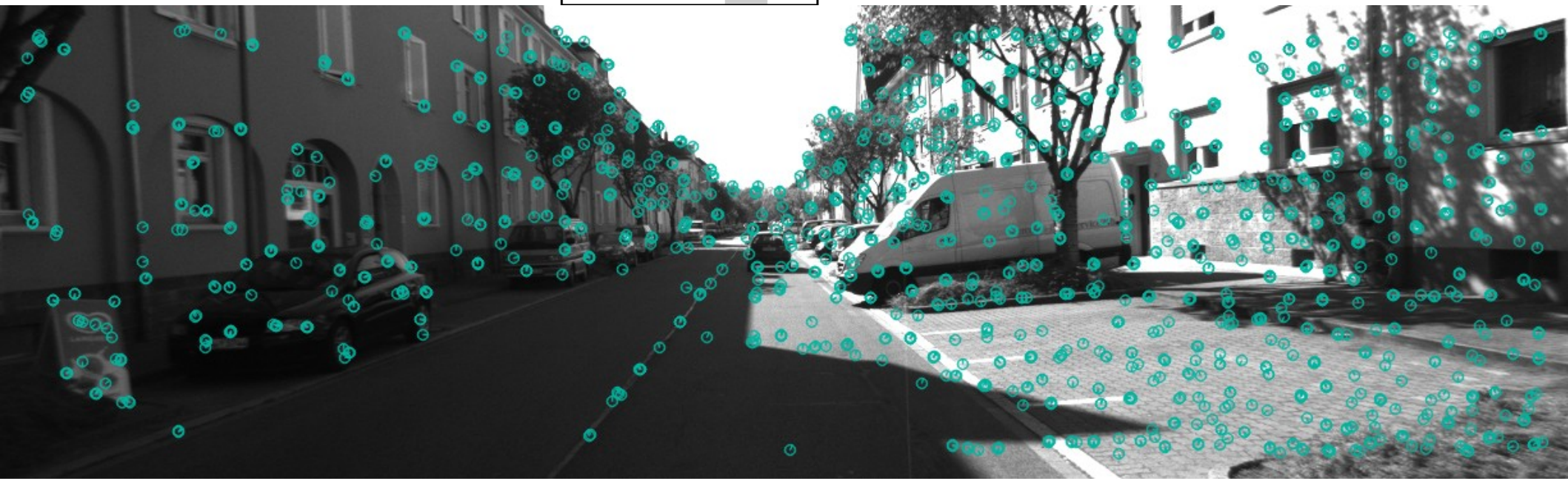
```
Input: Gefundene Features P
Sortiere P nach Corner-Score
While !(N-ε < |P| < N+ε):
    Markiere alle p_i in P als legal
    Output = ø
    Für alle p_i in P:
        Falls p_i  legal:
            Füge p_i zu Output hinzu
            Markiere alle p_j in ROI um p_i als illegal
    Update ROI
Return Output
```

Range-Tree ANMS

- Diesmal wird in der ROI mit Hilfe eines vorher konstruierten Range-Trees gesucht
- In einem Range Tree ist in jedem Knoten pro Dimension ein binärer Suchbaum gespeichert
- Anpassung der Suchbreite bis N-ε < P < N+ε




```
Input: Gefundene Features P
Sortiere P nach Corner-Score
While !(N-ε < |P| < N+ε):
    Markiere alle p_i in P als legal
    Output = ø
    Für alle p_i in P:
        Falls p_i  legal:
            Füge p_i zu Output hinzu
            Markiere alle p_j in ROI um p_i als illegal
    Update ROI
Return Output
```



Suppression via Square Covering (SSC)

- Keypunkte werden in Zellen der Breite w sortiert, Ausgeschlossen wird in allen Zellen, die mit 2w um Den hinzugefügten Keypunkt herum erreicht werden

- Gridauflösung wird über w angepasst, bis gewünschte Anzahl an Features erreicht ist

- Vorteil: Keine teuer zu konstruierende Datenstruktur notwendig

```
Input: Gefundene Features P
Sortiere P nach Corner-Score
While !(N-ε < |P| < N+ε):
    Markiere alle p_i in P als legal
    Output = ∅
    Für alle p_i in P:
        Falls p_i  legal:
            Füge p_i zu Output hinzu
            Markiere alle p_j in ROI um p_i als illegal
    Update ROI
Return Output
```

Soft SSC

- Variante von SSC: anstatt Zellen sofort komplett auszuschließen, wird die Corner-Score von hinzugefügten Punkten in den Zellen gespeichert
- Solange $score_{pi} < score_{Zelle} + threshold$, wird p_i zum Output hinzugefügt

```
Input: Gefundene Features P
Sortiere P nach Corner-Score
While !(N-ε < |P| < N+ε):
    Markiere alle p_i in P als legal
    Output = ∅
    Für alle p_i in P:
        Falls p_i  legal:
            Füge p_i zu Output hinzu
            Markiere alle p_j in ROI um p_i als illegal
    Update ROI
Return Output
```

Soft SSC mit threshold = 5

Reverse Grid Suppression

- Ziel: Vermeidung der Iteration über Größe der ROI
- Aufteilung in Zellen wie in SSC
- Ein Durchlauf durch Keypunkte P, mit niedrig gewählter Zellenbreite w

```
Sortiere P nach score
Für alle p_i in P:
    Suche in w nach besserem p_j
    if ∄p_j mit score(p_j) > score(p_i):
        Output = Output ∪ p_i
    if |Output| ≥ N return Output
```

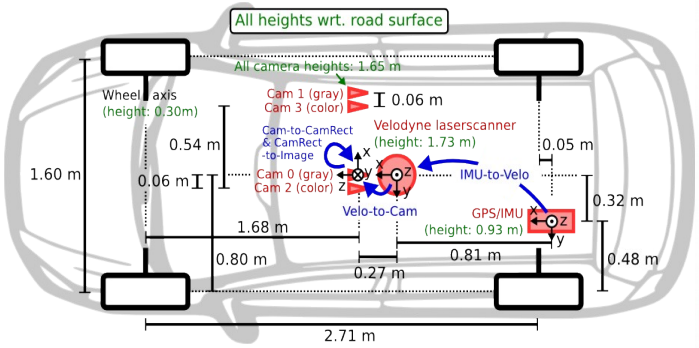
Evaluation

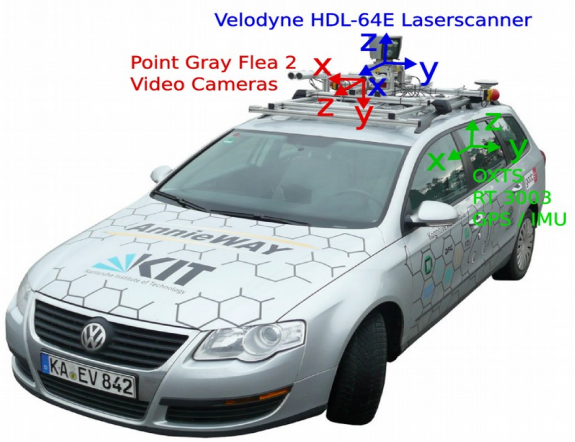
Messung der Qualität der Ergebnisse erfolgt durch Nutzung von Datensätzen mit Ground-Truth Information und Messung des relativen Fehlers der berechneten Kamera-Trajektorie

Genutzte Datensätze:

- KITTI:

  Stereo-Bilder,
  10FPS-Aufnahmen von
  Dorf/Stadt-Straßen



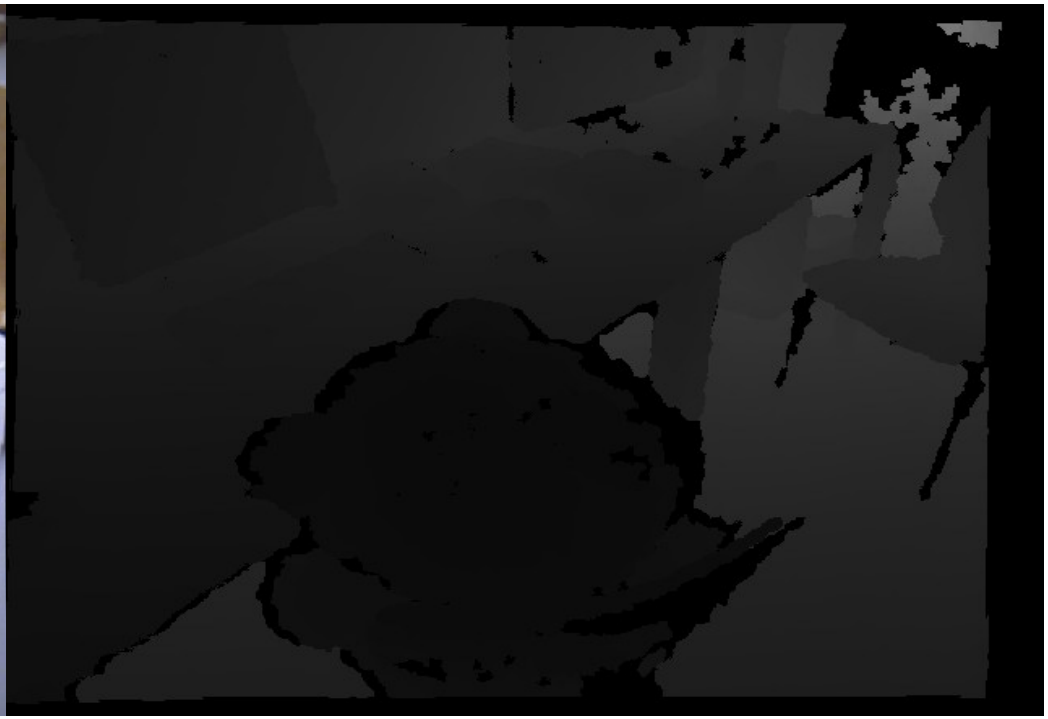


Genutzte Datensätze:

TUM:

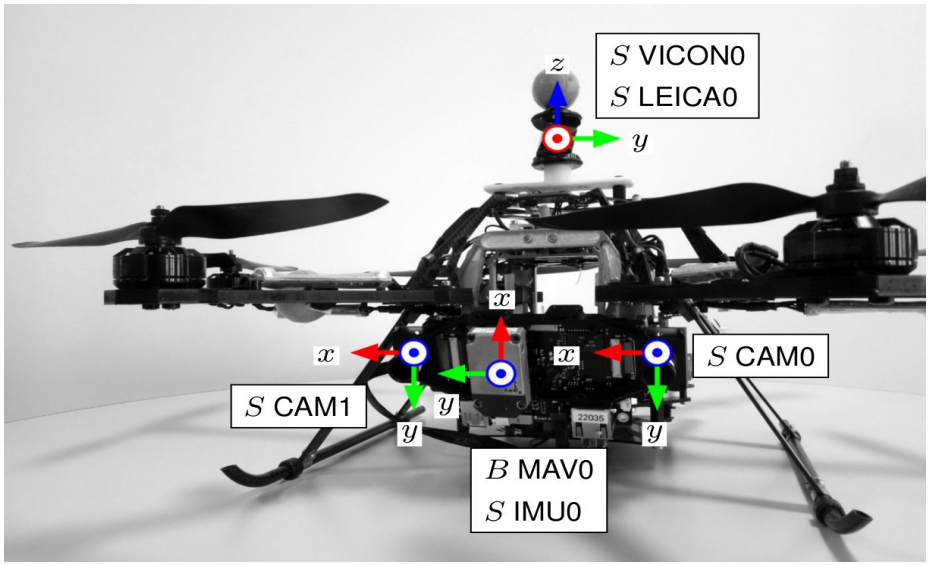

- 30FPS monokulare Aufnahmen von recht begrenzten Innenräumen
- Handheld/Ferngesteuerter Roboter


- Depth Map

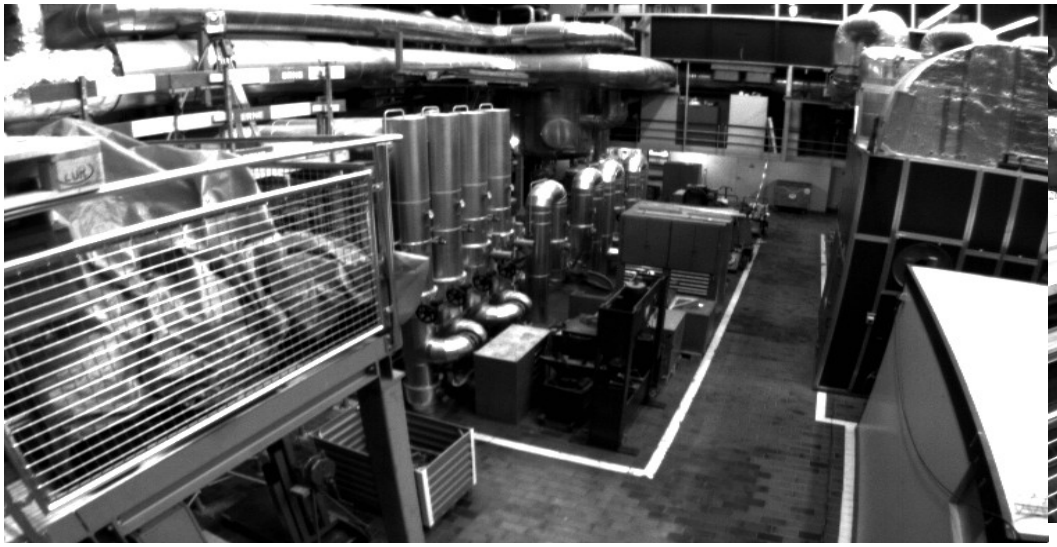



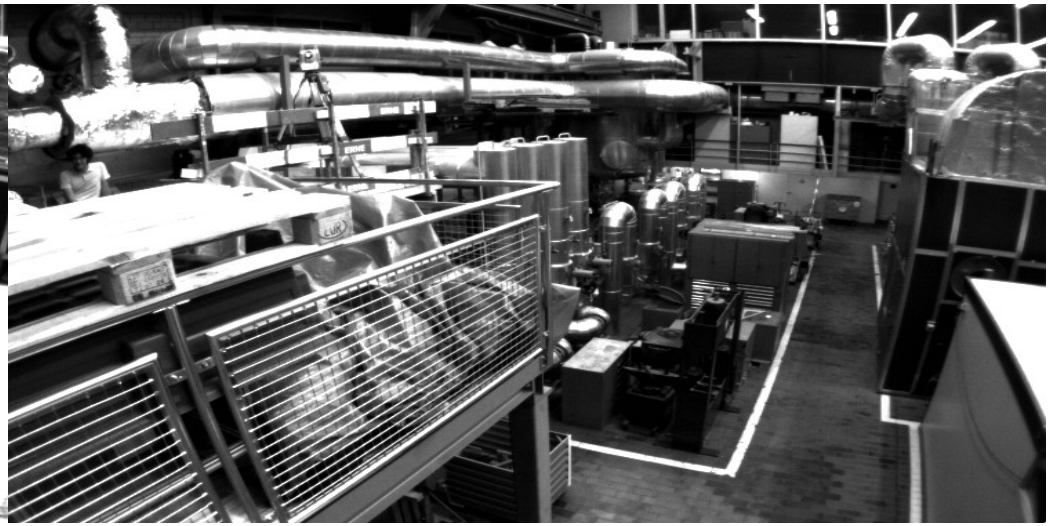
Genutzte Datensätze:

EuRoC MAV:



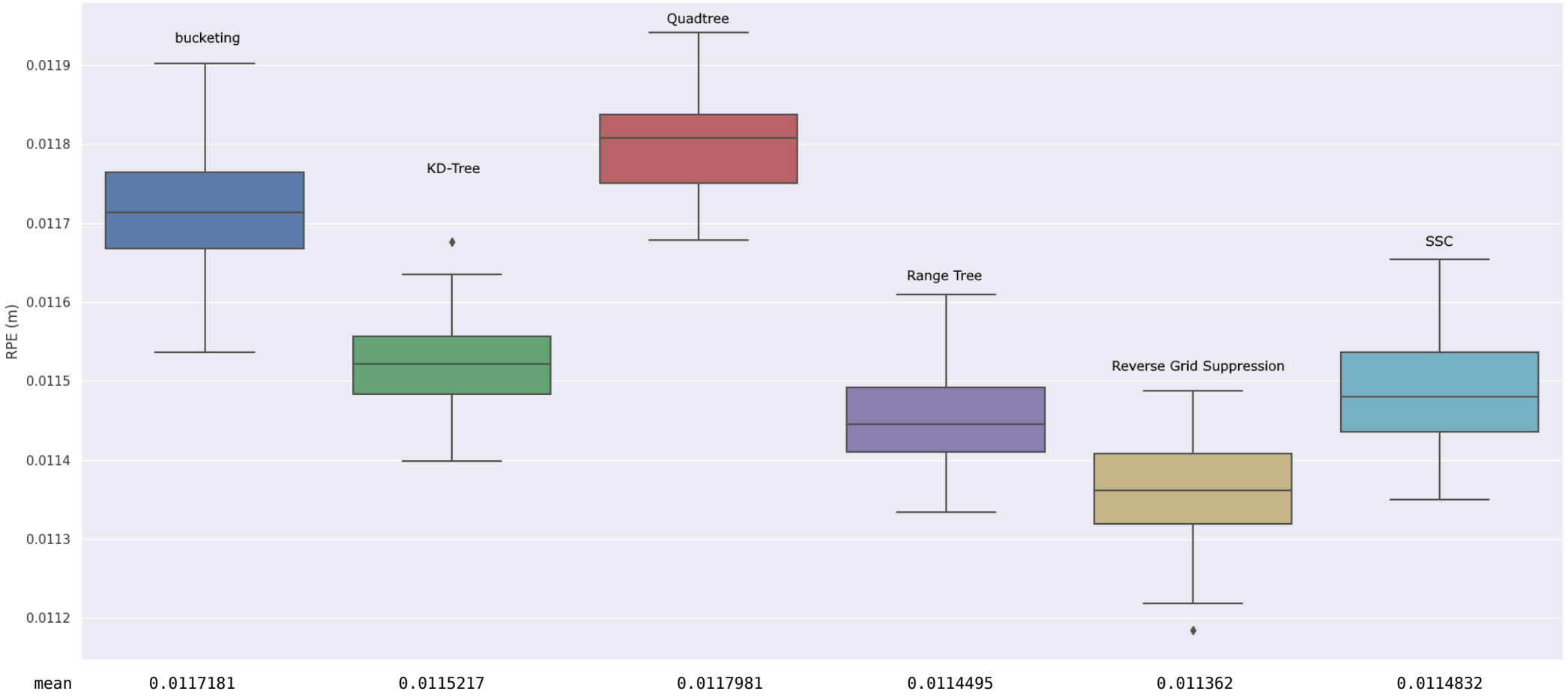
- 30 FPS, mit Drone aufgenommen
- Kleine und große Innenräume (Zimmer, Maschinenhalle)





Ergebnisse

KITTI Sequenz 7, relativer Fehler pro Frame in Metern über 100 Iterationen:



	bucketing	KD-Tree	Quadtree	Range Tree	Reverse Grid Suppression	SSC
mean	0.0117181	0.0115217	0.0117981	0.0114495	0.011362	0.0114832

TODO: Mit Soft SSC Updaten

Ergebnisse

RGBD TUM, freiburg3_long_office_household, relativer Fehler pro Frame in Metern über 100 Iterationen:

TODO

Ergebnisse

EuRoC, Machine_Hall_03, relativer Fehler pro Frame in Metern über 100 Iterationen:

TODO


The End