TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Technische Informatik, Professur für VLSI Entwurfssysteme, Diagnostik und Architektur

Untersuchung der Berechnung der 3D-Punktkorrelation auf hochparallelen Plattformen

Verteidigung Großer Beleg

Simon Willeke
simon.willeke@mailbox.tu-dresden.de

Dresden, 28.03.13

[1]

Gliederung

1 Aufgabenstellung

- „Zeitliche Punktkorrelation über statistische Grauwertsequenzen“
- Hochpräzisions-3D-Vermessung
- viele Bilder für eine Punktwolke erforderlich
  - hoher Rechenaufwand
  - Auswertung auf CPU dauert mehrere Sekunden

- Ziel ist Auswertung in Echtzeit:
  Punktwolkenfrequenz ~ Bildfrequenz

- Institut für angewandte Optik, Friedrich Schiller Universität Jena, Arbeitsgruppe 3D-Vermessung, Prof. Kowarschik

1 Aufgabenstellung

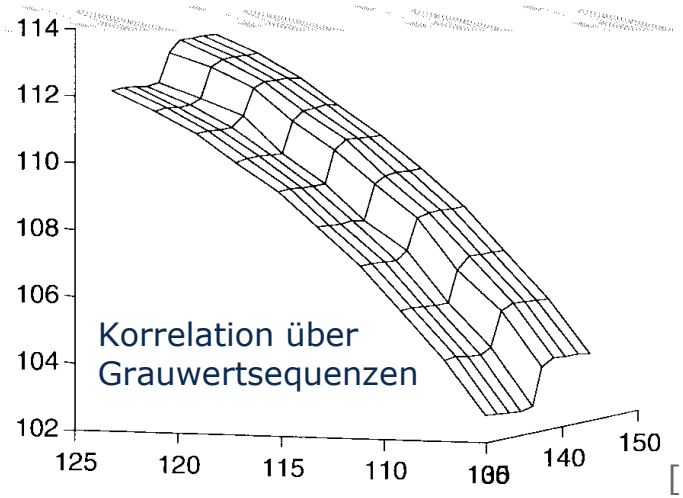
- meine Aufgabe:
  Auswahl einer hochparallelen Plattform für Berechnung

- gewählte Kandidaten:
  - FPGA – Kintex7 Board KC705
  - GPU – beliebige CUDA-fähige Nvidia Grafikkarte

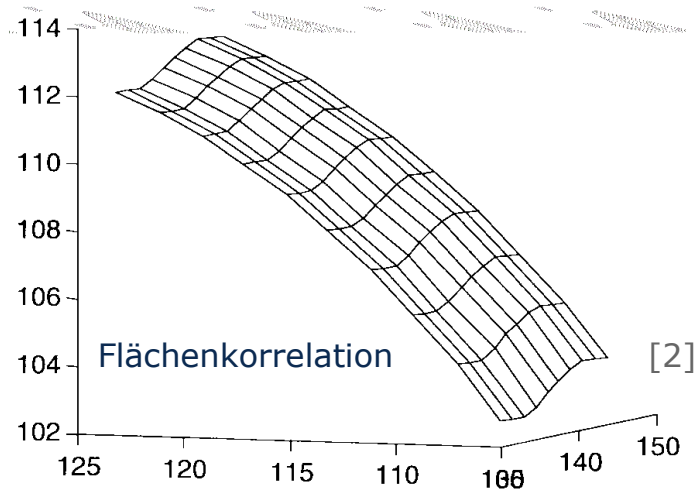
2 Grundlagen

Motivation

- viele Verfahren für 3D-Scanning
- Anwendungen:
  - Medizin
  - automatisierte Produktprüfung auf Produktionslinien
- besonders hohe Präzision gefordert
- pixelgenaue Tiefeninformation
  - keine Approximation von Objektkanten (Bild)



[2]

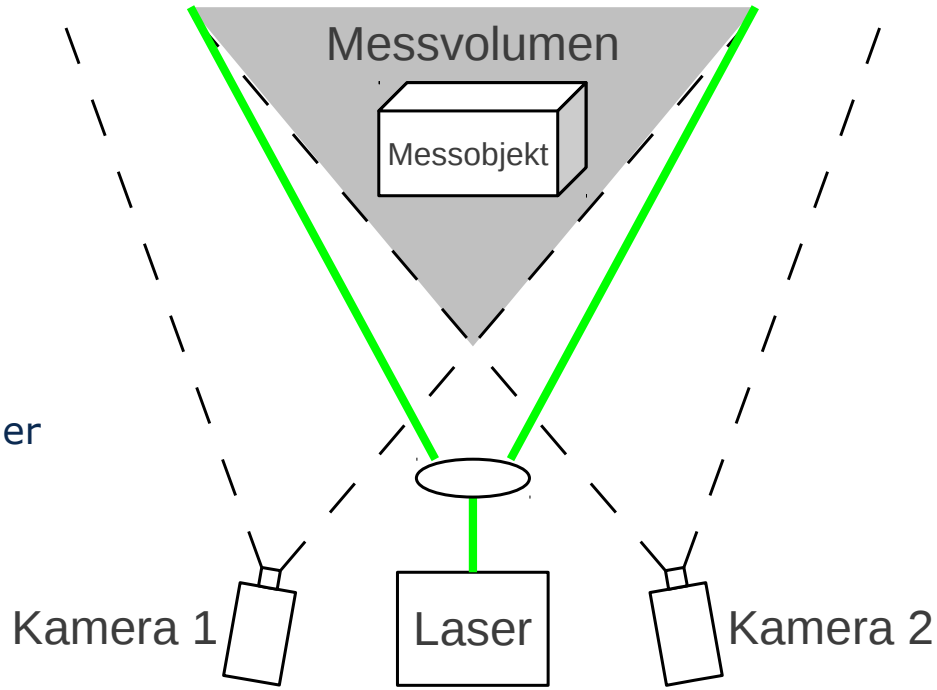


[2]

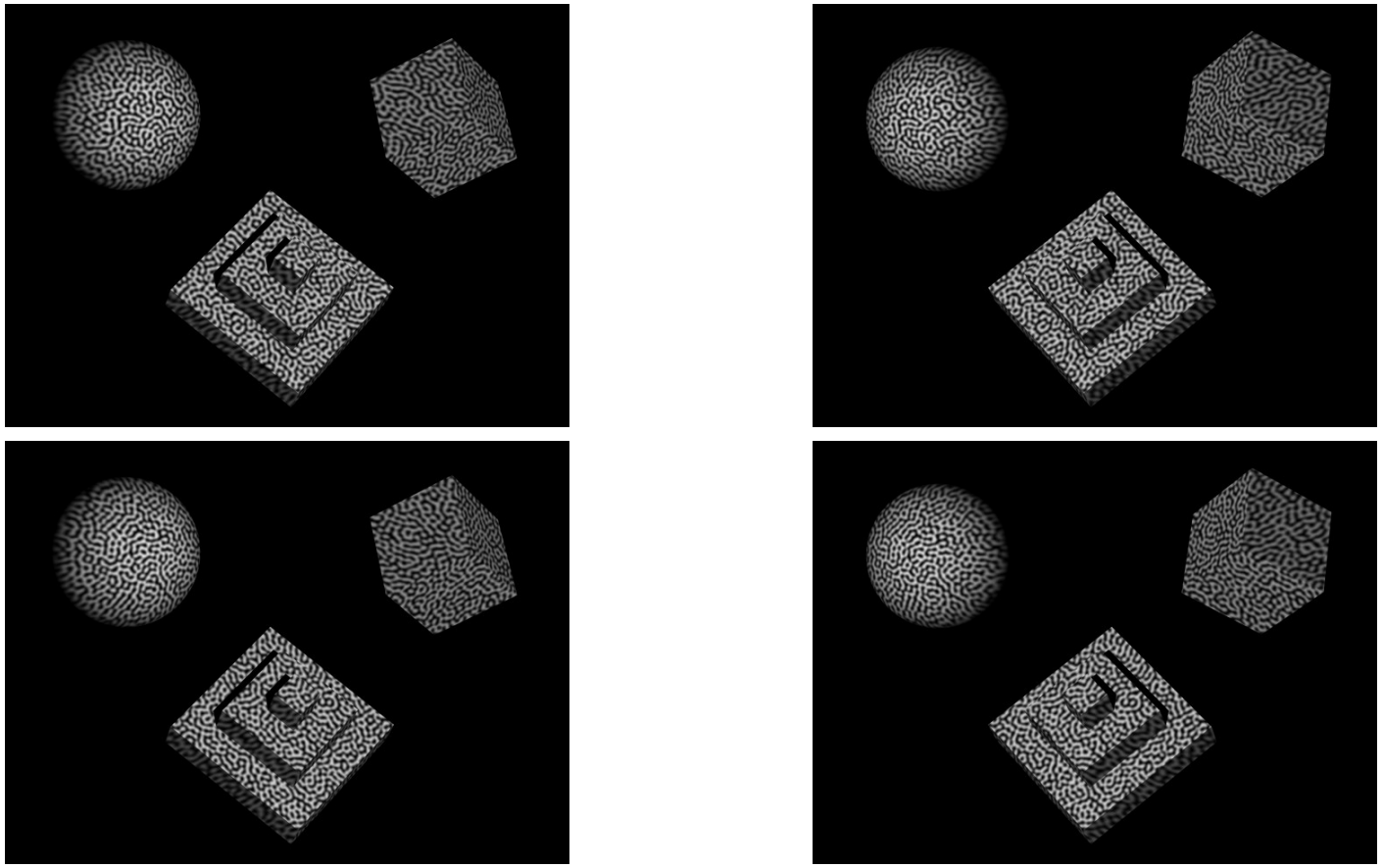
2 Grundlagen

Aufbau

- 2 Kameras (8bit Grauwerte)
- Projektion zeitveränderlicher statistischer Muster
  - Laser-Speckles (Bild)
  - Diaprojektor
- Aufnahme und Auswertung einer Sequenz von Stereobildern: „Stereobildsequenz“

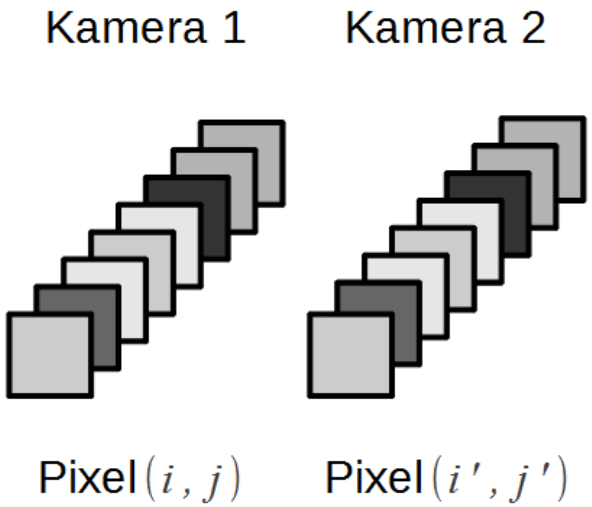


2 Grundlagen



2 Grundlagen

Basisalgorithmus

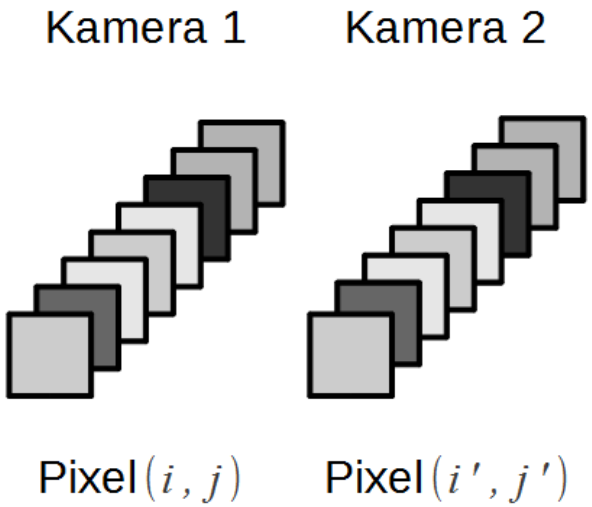


- Suche nach homologen Punkten
  - Pixel im linken und rechten Bild zeigen selben Objektpunkt
  - Berechnung mittels normalisierter Kreuzkorrelation ρ
  - ρ liegt zwischen -1,0 und 1,0
  - zwei Punkte sind homolog, wenn max(ρ) > 0,8
  - Objektpunkt nur von einer Kamera sichtbar → Schwellwert wird nicht erreicht
- Triangulation

$$\rho_{i,j,i',j'} = \frac{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right) \cdot \left(g_{i',j',t} - \overline{g_{i',j'}}\right)}{\sqrt{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right)^2} \cdot \sqrt{\sum_{t=1}^{N} \left(g_{i',j',t} - \overline{g_{i'.j'}}\right)^2}}$$

2 Grundlagen

Basisalgorithmus



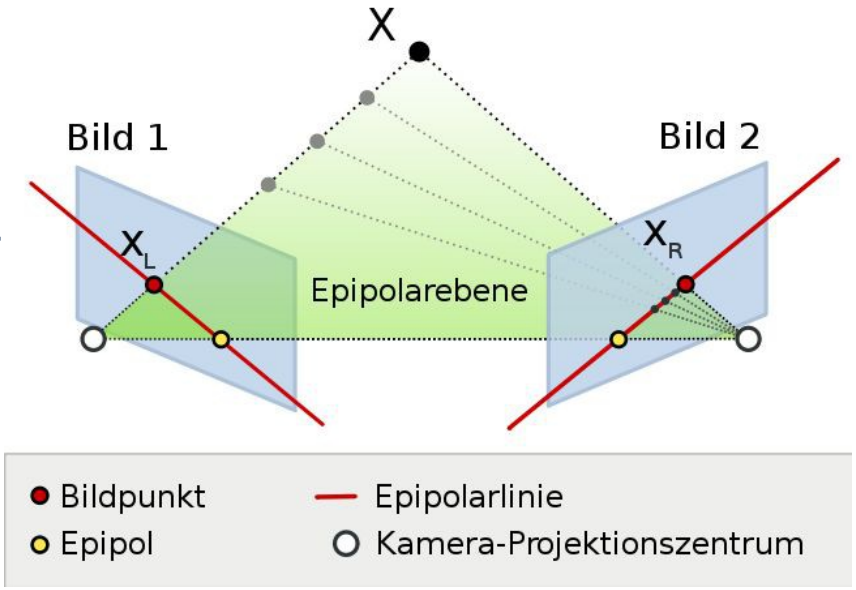
- Suche nach homologen Punkten
  - Pixel im linken und rechten Bild zeigen selben Objektpunkt
  - Berechnung mittels normalisierter Kreuzkorrelation ρ
  - ρ liegt zwischen -1,0 und 1,0
  - zwei Punkte sind homolog, wenn max(ρ) > 0,8
  - Objektpunkt nur von einer Kamera sichtbar → Schwellwert wird nicht erreicht
- ~~Triangulation~~

$$\rho_{i,j,i',j'} = \frac{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right) \cdot \left(g_{i',j',t} - \overline{g_{i',j'}}\right)}{\sqrt{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right)^2} \cdot \sqrt{\sum_{t=1}^{N} \left(g_{i',j',t} - \overline{g_{i'.j'}}\right)^2}}$$

2 Grundlagen

Verbesserung der Performance

- Punktsuche im ganzen Bild dauert sehr lange
- Beschränkung auf Epipolarlinien bei bekannten Kameraparametern
- **Rektifizierung** zur Verbesserung der *Cache Utilisation* und Pipelinebarkeit der Speicherzugriffe (Burst-Zugriff)



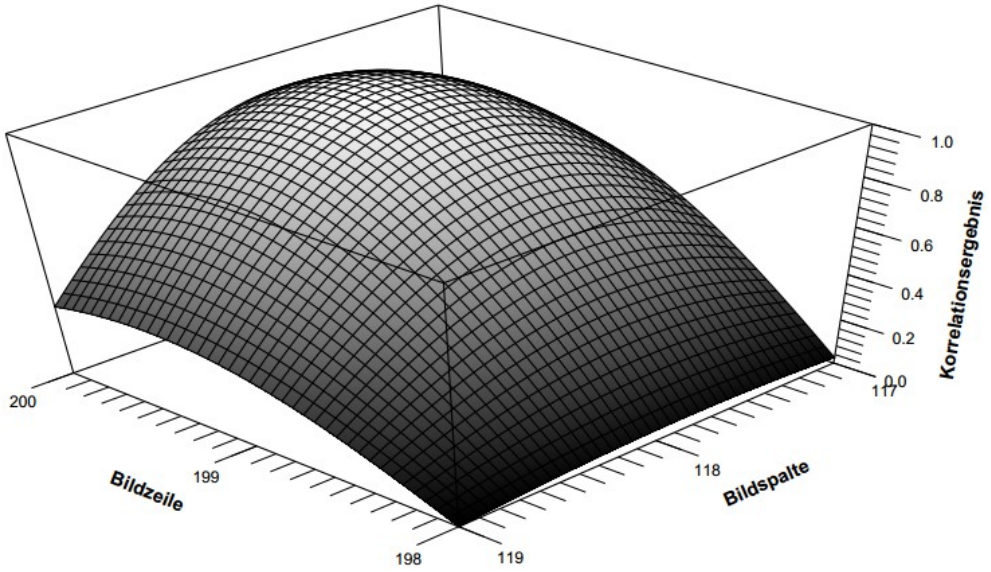
[3]

2 Grundlagen

Verbesserung der Präzision

- Punktzuordnung auf Pixelraster ungenau
- besser: Subpixelraster
  - Interpolation notwendig
- Ziel ist maximale Präzision
  - bikubische Interpolation

» **Subpixelrefinement**

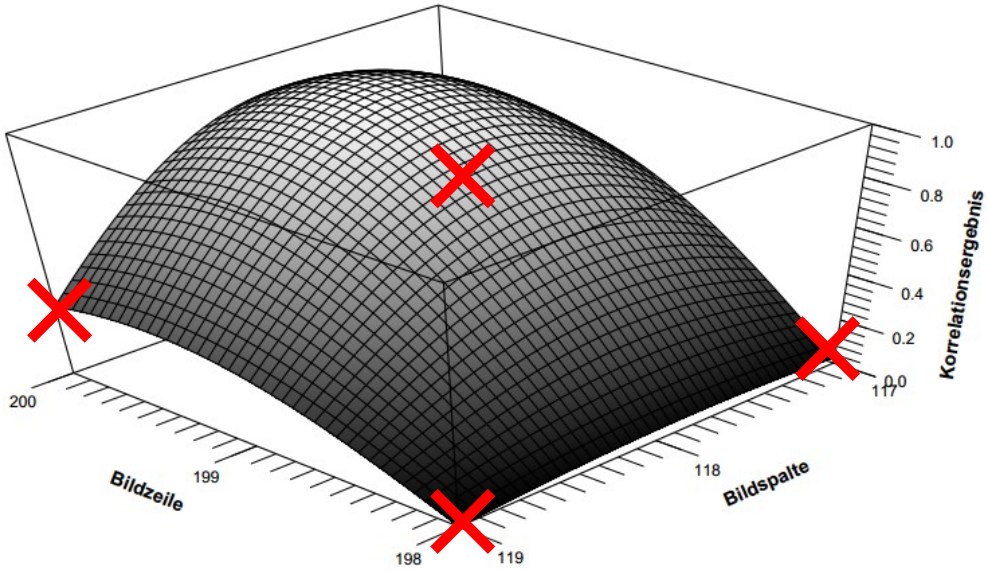


2 Grundlagen

Verbesserung der Präzision

- Punktzuordnung auf Pixelraster ungenau
- besser: Subpixelraster
  - Interpolation notwendig
- Ziel ist maximale Präzision
  - bikubische Interpolation

» **Subpixelrefinement**

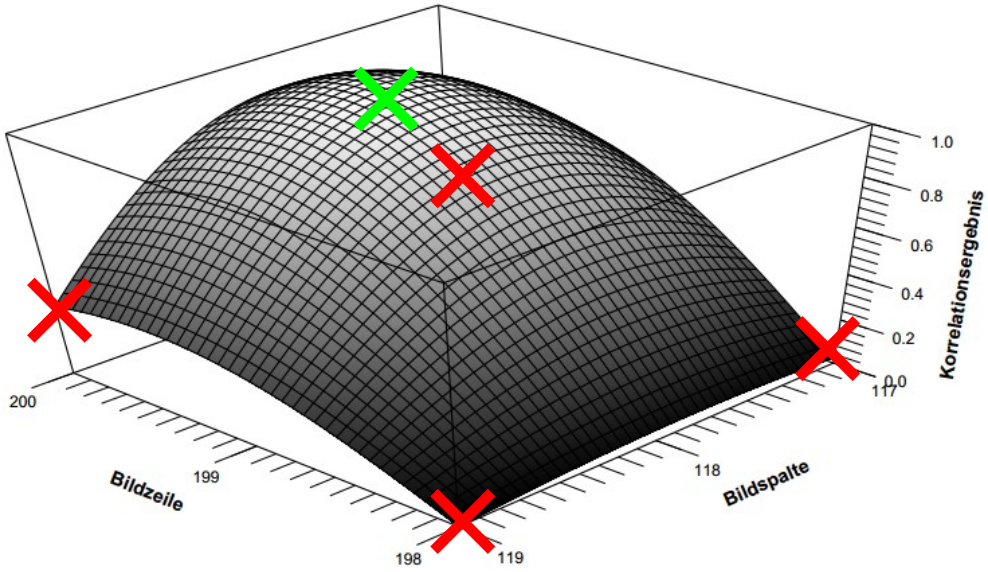


2 Grundlagen

Verbesserung der Präzision

- Punktzuordnung auf Pixelraster ungenau
- besser: Subpixelraster
  - Interpolation notwendig
- Ziel ist maximale Präzision
  - bikubische Interpolation

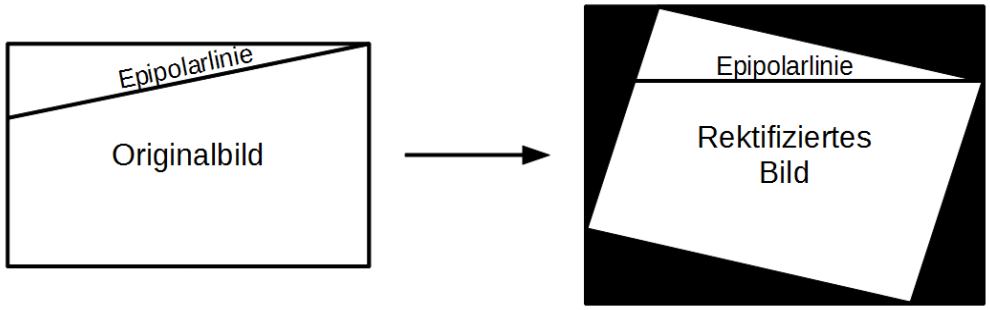
» **Subpixelrefinement**



2 Grundlagen

Zusammenfassung des Algorithmus

- **Rektifizierung**



- **Punktsuche**
  mit vorheriger Normalisierung

$$\rho_{i,j,i',j'} = \frac{\sum_{t=1}^{N} (g_{i,j,t} - \overline{g_{i,j}}) \cdot (g_{i',j',t} - \overline{g_{i',j'}})}{\sqrt{\sum_{t=1}^{N} (g_{i,j,t} - \overline{g_{i,j}})^2} \cdot \sqrt{\sum_{t=1}^{N} (g_{i',j',t} - \overline{g_{i',j'}})^2}}$$

- **Subpixelrefinement**

3 Untersuchung auf dem Grafikprozessor

Implementierung in CUDA

- Vollständige Implementierung auf Basis von *Correlation Expert* und *Shape Measurement Toolkit (SMTK)*
- CUDA als Plattform wegen:
  - verfügbarer Hardware an beiden Lehrstühlen
  - minimal besserer Performance gegenüber OpenCL
- Entwurf eines Kernels für jeden Teilalgorithmus
  - Partitionierung: ein Block per Zeile, ein Thread per Pixelsequenz

3 Untersuchung auf dem Grafikprozessor

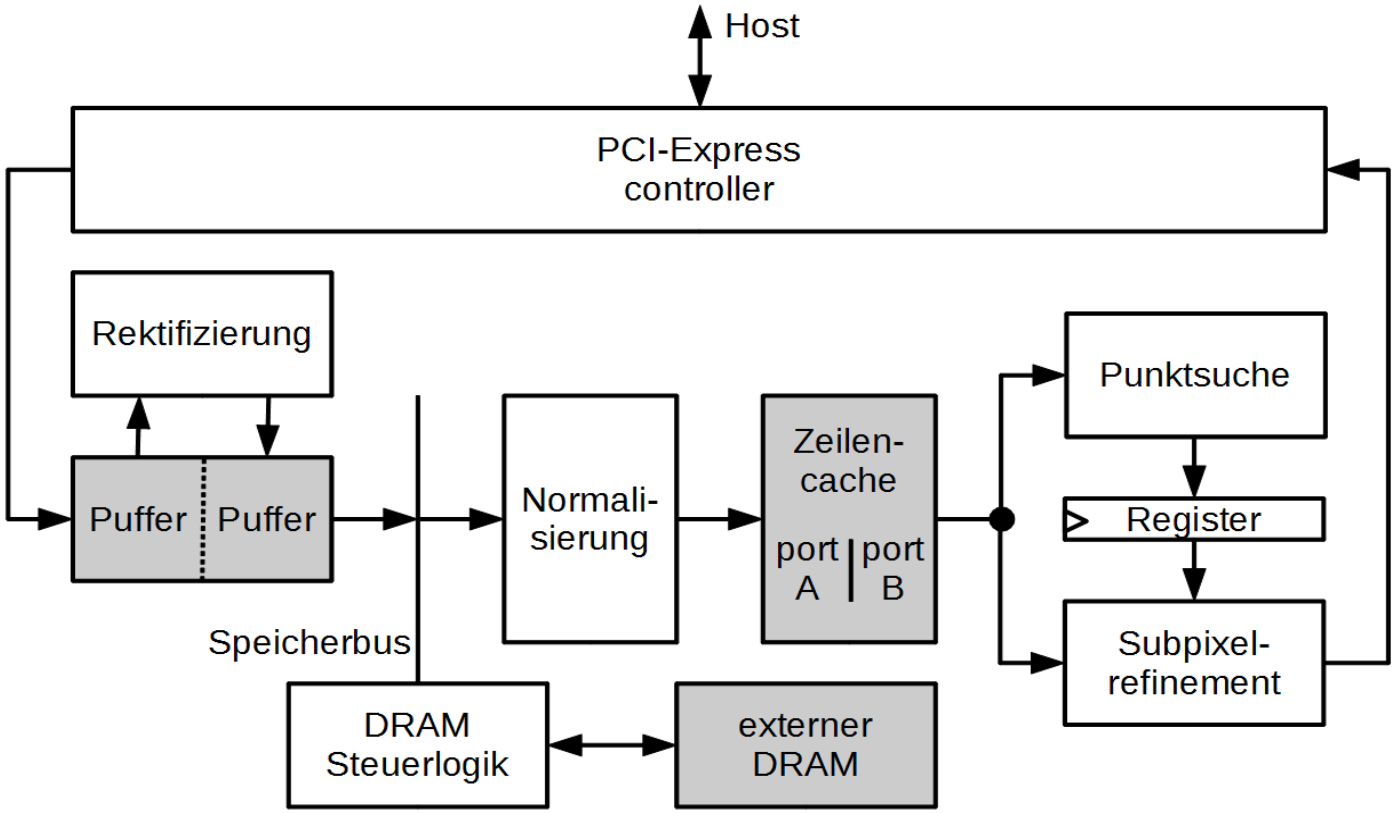
Implementierung in CUDA

- Probleme:
  - *Shared Memory* zu klein für komplette Bildsequenzzeile
  - Konstantenspeicher nicht benutzt
  - Handoptimierung aufgrund eingeschränkter Registerzahl
  - *Branch Divergence* bei Punktsuche und Subpixelrefinement
    - Overhad bis zu 77%
- Erfolge:
  - Rektifizierung und Normalisierung ohne *Branch Divergence*
  - gute Texturcache-Nutzung aufgrund 2D-Lokalität

- Optimierungsmöglichkeiten:
  - Algorithmen für Punktsuche und Subpixelrefinement vereinfachen
    - keine *Branch Divergence*, dafür überflüssige Berechnungen

4 Untersuchung auf dem FPGA

Blockdiagramm des Designs



4 Untersuchung auf dem FPGA

FPGA Entwurf

- keine VHDL Implementierung
- Entwurf der Teilalgorithmen auf Papier
- Abschätzung zu Ressourcenverbrauch und Geschwindigkeit

- kritische Ressourcen sind Multiplizierer und BlockRAMs

- Flaschenhals DRAM?
  - DDR3, 800MHz, 64bit Datenbus
  - theoretisch über 4400fps!
  - ca. 4200 Takte je Stereobildsequenzzeile

4 Untersuchung auf dem FPGA

FPGA Entwurf

- Punktsuche sehr langsam
  - sequentielle Suche dauert min. 22000 Takte je Zeile
  - DRAM nur 4200 Takte je Zeile
  - parallele Punktsuche
    - mehrfach je Zeile
    - in mehreren Zeilen → großer Zeilencache

- Subpixelrefinement
  - parallelisierbar, jede Einheit verarbeitet ein Punktpaar
  - ca. 1200 Multiplizierer für vollst. Pipeline notwendig

4 Untersuchung auf dem FPGA

FPGA Entwurf

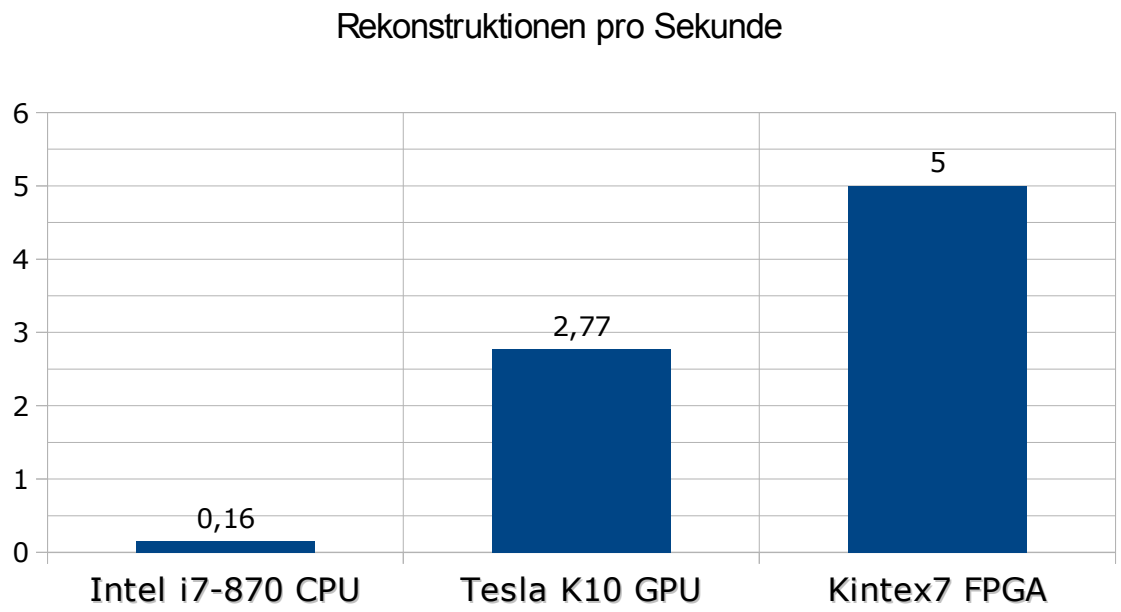
- Beispielentwürfe:

Entwurf	Geschwindigkeit	kritische Ressourcen	
		DSP Slices	BlockRAM 36k
<0.5, 2, 0.5>	5 Hz	690	272
<1, 4, 1>	11 Hz	1332	356
<4, 8, 4>	46 Hz	5184	583
<16, 64, 16>	184 Hz	20592	2891

- Entwurf: <Rektifizierungsmodule, Punktsuchemodule, Refinementmodule>
- Geschwindigkeit in 3D-Rekonstruktionen pro Sekunde

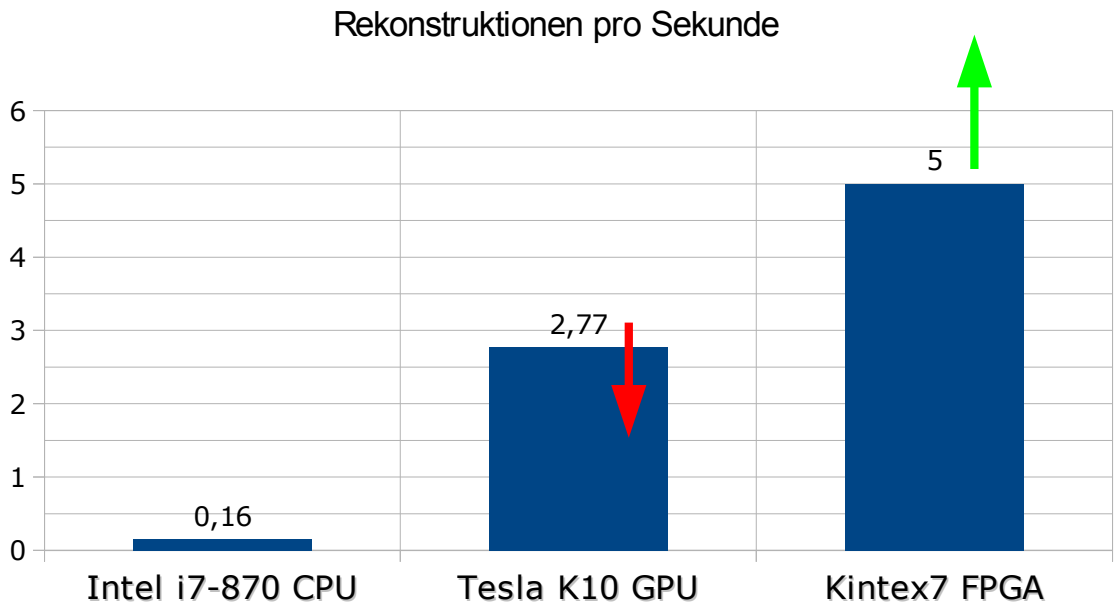
5 Ergebnis

Vergleich der Plattformen



5 Ergebnis

Vergleich der Plattformen



6 Ausblick

Maßgeschneiderte Algorithmen für den FPGA

- Interpolationsalgorithmen ohne Multiplikationen
- Refinementalgorithmen ohne Normalisierung
- Herausfiltern strukturloser Punkte

Kombination der Vorteile

- Teilalgorithmen ohne *Branch Divergence* auf GPU
- komplexe Algorithmen auf dem FPGA

oder

- hochinterpolieren der Bilder auf GPU
- Punktsuche ohne Subpixelrefinement auf FPGA

Quellen

[1] YouTube Kanal der Arbeitsgruppe 3D-Vermessung
http://www.youtube.com/user/IAOag3DVermessung

[2] P. Albrecht et al. (1998): „Improvement of the Spatial Resolution of an optical 3-D measurement Procedure“, IEEE Transactions on Instrumentation and Measurement, Vol. 47. No. 1, February 1998.

[3] www2.informatik.hu-berlin.de/cv/index.php?menu=teaching&submenu=overviewteaching&mode=overviewteaching

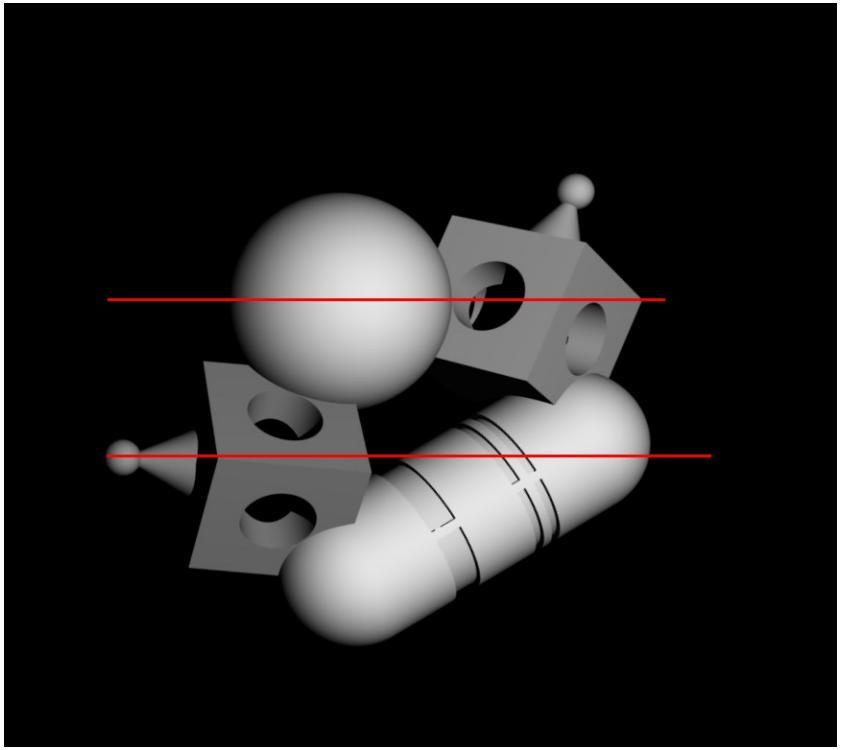
[4] T. Luhmann. Nahbereichsphotogrammetrie. Grundlagen, Methoden und Anwendun-gen. Herbert Wichmann Verlag, Heidelberg, 2003.

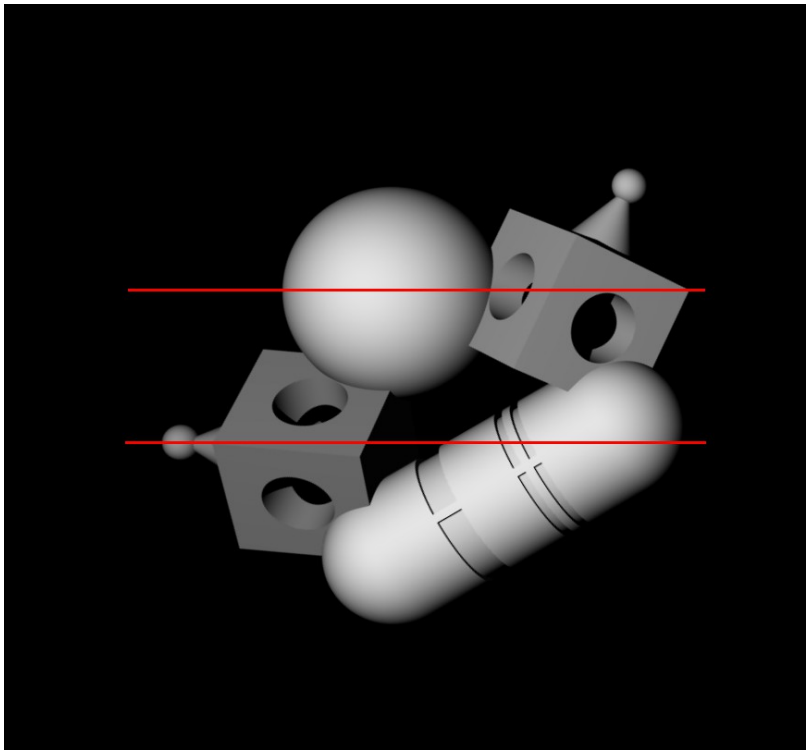
[5] J. Sanders, E. Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Longman, Amsterdam, 1 edition, Juli 2010.

[6] M. Schaffer, M. Grosse, R. Kowarschik. High-speed pattern projection for three-dimensional shape measurement using laser speckles. Applied Optics, 49(18): S 3622–3629, Juni 2010.

»Wissen schafft Brücken.«

Anhang





Anhang

hier Versuchsvideo