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

Zwischenvortrag Großer Beleg

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

Dresden, 20.06.12

Einleitungsvideo

Video aus [1]

Gliederung

1 Aufgabenstellung

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

- Hochpräzisionsvermessung
- viele Bilder für eine Punktwolke erforderlich
  - hoher Rechenaufwand
  - Auswertung auf CPU dauert mehrere Sekunden

- Ziel ist Auswertung in Echtzeit:
  Punktwolkenfrequenz ≈ Bildfrequenz

1 Aufgabenstellung

- meine Aufgabe:
  Auswahl einer hochparallelen Plattform für Berechnung

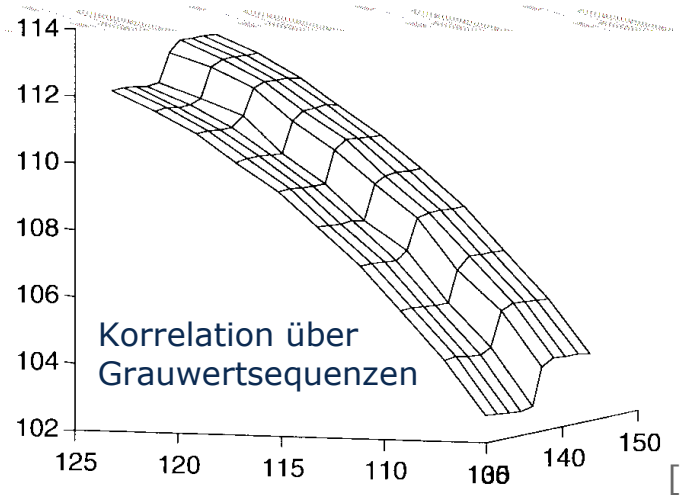
- gewählte Kandidaten:
  - FPGA – Kintex7 Board KC705
  - GPU – beliebige CUDA-fähige Nvidia Grafikkarte

- vollständige Implementierung in Diplomarbeit?

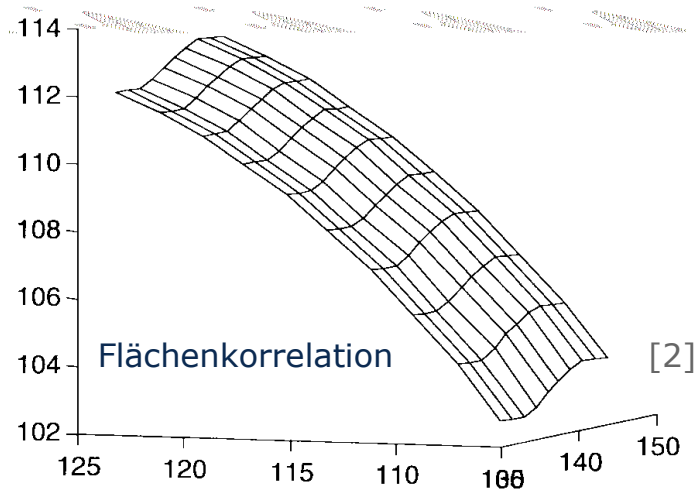
2 Grundlagen

Motivation

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



[2]

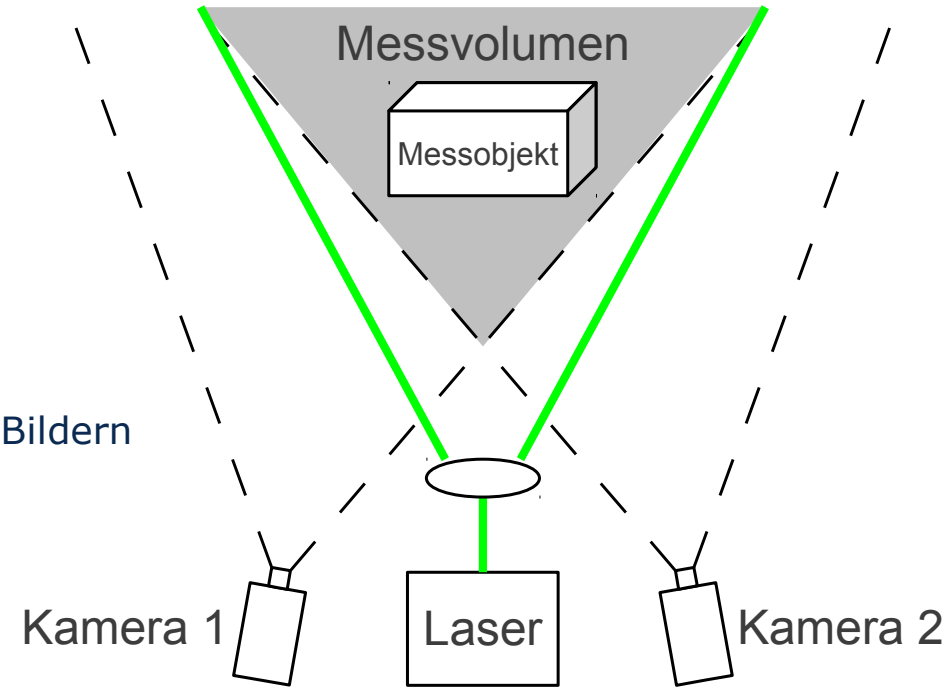


[2]

2 Grundlagen

Aufbau

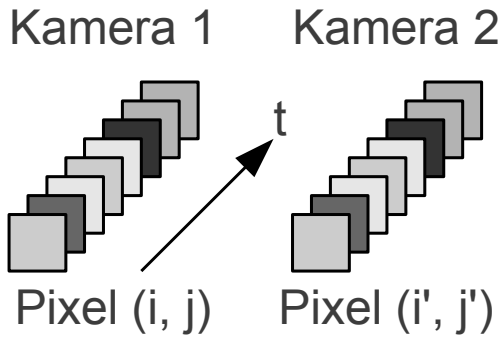
- 2 Kameras
- Projektion zeitveränderlicher statistischer Muster
  - Laser-Speckles (Bild)
  - Diaprojektor
- Ausertung einer Sequenz von Bildern



2 Grundlagen

Basisalgorithmus

- Suche nach homologen Pixeln
  - 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

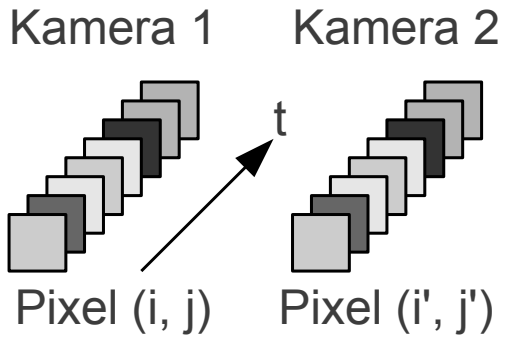


$$\rho_{i,j,i',j'} = \frac{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right) \cdot \left(g_{i',j',t} - \overline{g_{i',j'}}\right)}{\sqrt{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right)^2} \cdot \sqrt{\sum_{t=1}^{N} \left(g_{i',j',t} - \overline{g_{i'.j'}}\right)^2}}$$

2 Grundlagen

Basisalgorithmus

- Suche nach homologen Pixeln
  - 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~~

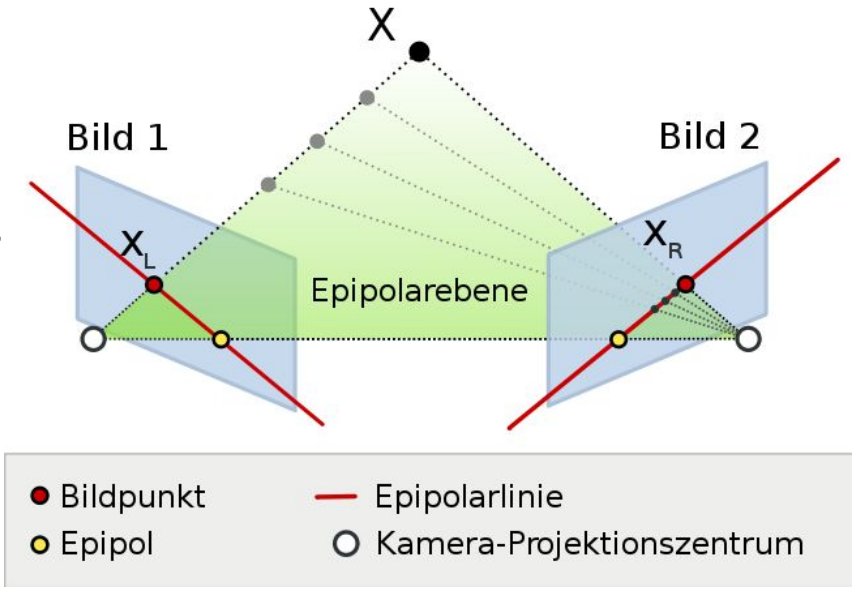


$$\rho_{i,j,i',j'} = \frac{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right) \cdot \left(g_{i',j',t} - \overline{g_{i',j'}}\right)}{\sqrt{\sum_{t=1}^{N} \left(g_{i,j,t} - \overline{g_{i,j}}\right)^2} \cdot \sqrt{\sum_{t=1}^{N} \left(g_{i',j',t} - \overline{g_{i'.j'}}\right)^2}}$$

2 Grundlagen

Verbesserung der Performance

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



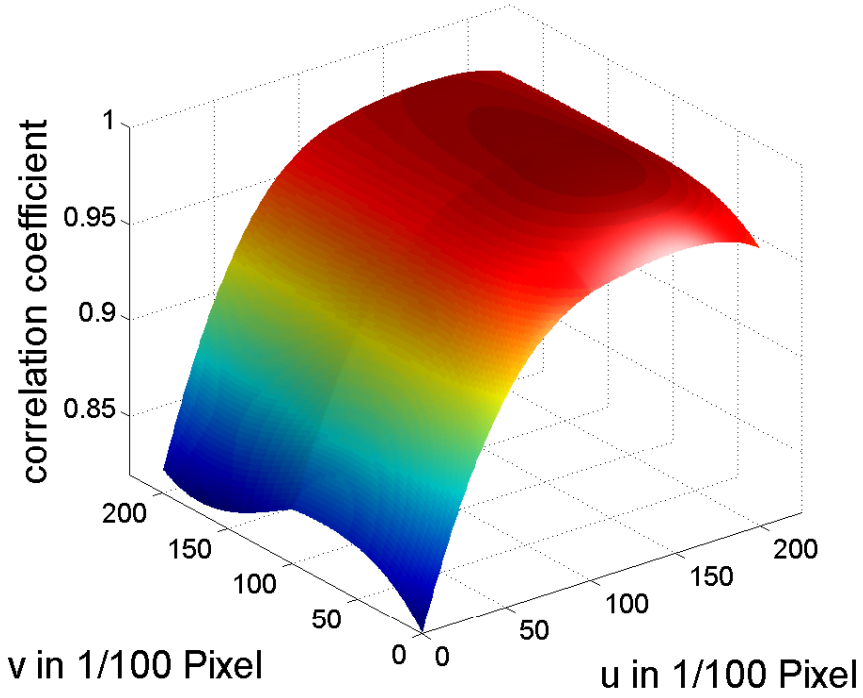
[4]

2 Grundlagen

Verbesserung der Präzision

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

» **Subpixelinterpolation**



[3]

2 Grundlagen

Zusammenfassung des Algorithmus

1) **Rektifizierung**

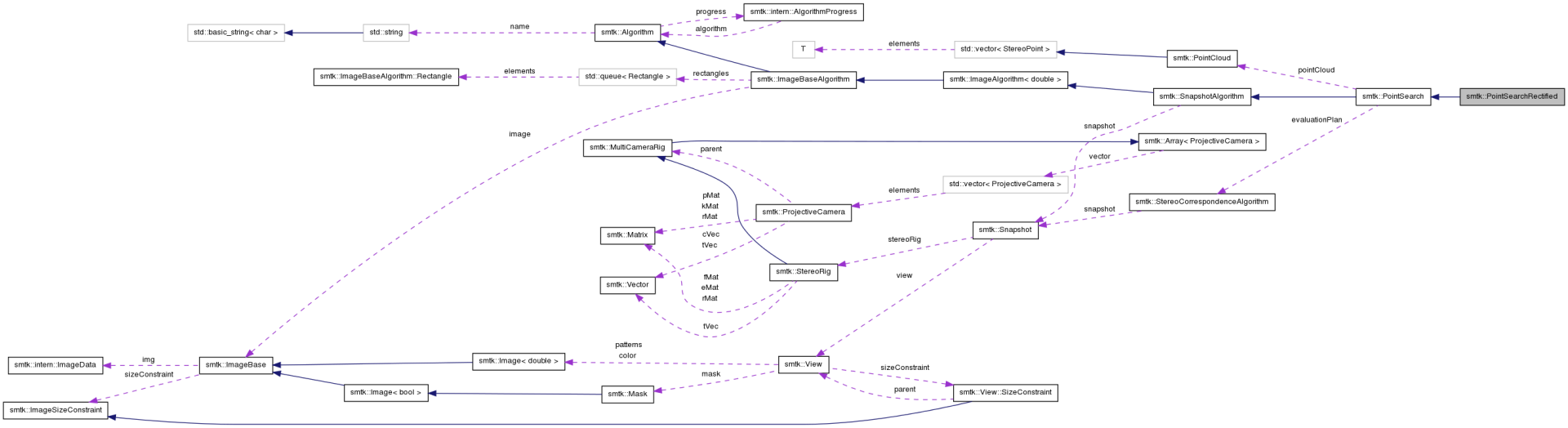
2) **Punktsuche**

3) **Subpixelinterpolation**

3 Vorgehensweise und Lösungsansätze

Shape Measurement ToolKit (SMTK)

- C++ Klassenbibliothek
- Grundlage für Auswertung auf CPU
- Portierung der benutzten Klassen zu CUDA



3 Vorgehensweise und Lösungsansätze

C++ Programm

```
#include <smtk/smtk.h>
using namespace smtk;

int main(int argc, char** argv) {
            Algorithm::threads = 4;
            ImageInterpolationMode = InterpolateBicubic;
            Snapshot snapshot;
            snapshot.loadPatternImages("data/camera-1_%03d.png", "data/camera-2_%03d.png",0,24);
            snapshot.stereoRig.read("data/stereorig.ini");

            snapshot.rectify();

            PointCloud pointcloud;
            CrossCorrelationPlan evaluator(0.8);
            PointSearchRectified pointsearch(snapshot, pointcloud, evaluator);
            ParaboloidFit refinement(snapshot, pointcloud, evaluator, 0.001, 1.0, 100);
            Reconstruction reconstruction(pointcloud, snapshot.stereoRig);

            pointsearch.run();

            refinement.run();

            reconstruction.run();
            pointcloud.write("data/pointcloud.pcf");
            TimerReport::print();
            return 0;
}
```

3 Vorgehensweise und Lösungsansätze

CUDA Umsetzung

- CUDA großteils kompatibel mit C++

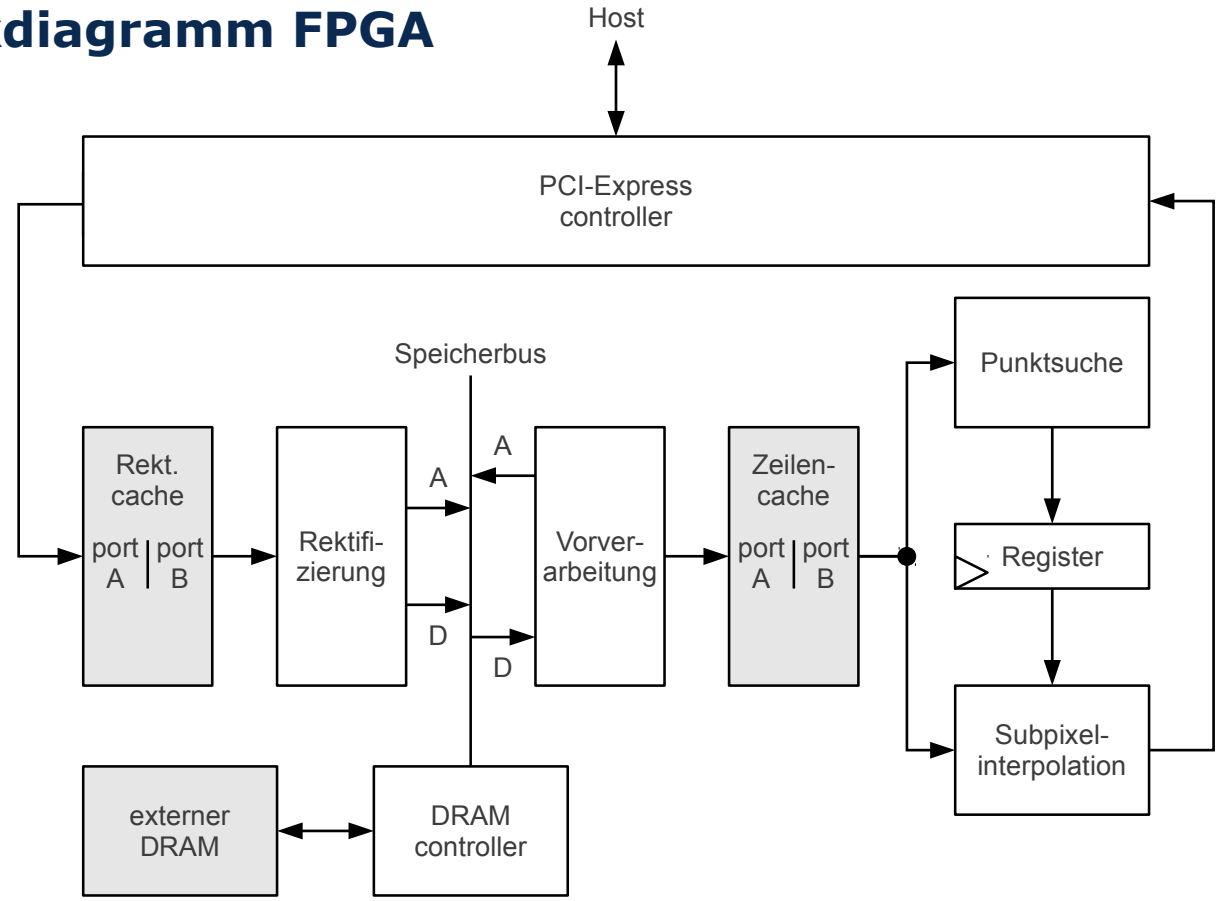
- Datentransfer Hauptspeicher↔Grafikkarte mit CUDA-Streams
- Optimierung der internen Speicherhierarchie vorerst vernachlässigbar

- cuBLAS Library für Rektifizierung
- Punktsuche könnte kritisch sein
- Subpixelinterpolation sehr gut parallelisierbar

  Flaschenhälse
  - sequentielle Punktsuche (?)
  - Datentransfer

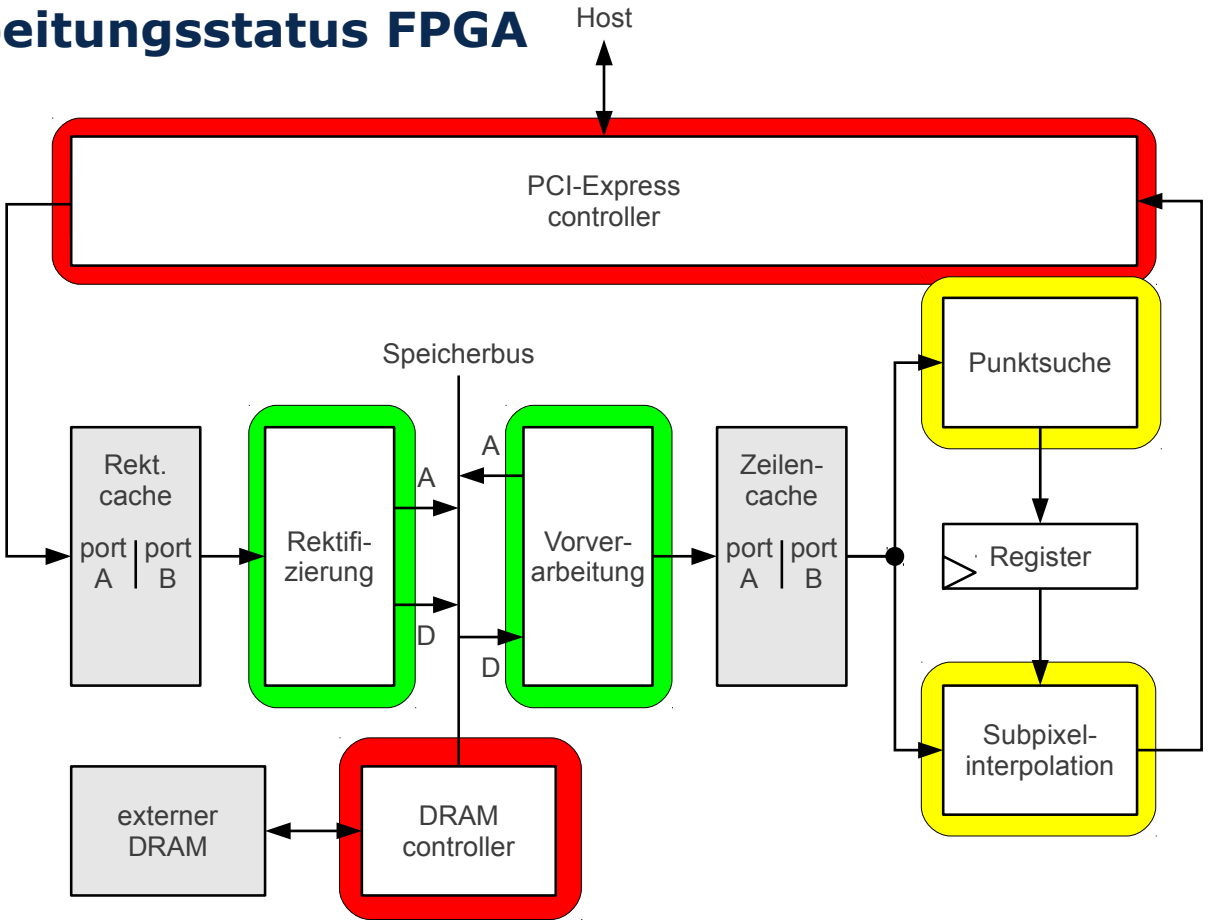
3 Vorgehensweise und Lösungsansätze

Blockdiagramm FPGA



3 Vorgehensweise und Lösungsansätze

Bearbeitungsstatus FPGA



3 Vorgehensweise und Lösungsansätze

FPGA Entwurf

- Rektifizierung
  - bearbeitet einen Pixel pro Takt
  - ca. 300fps bei 200MHz Takt und Auflösung 640x480
  - benötigt 36 Multiplikatoren (DSP Slices)

- Flaschenhals DRAM?
  - DDR3, 800MHz, 64bit Datenbus
  - theoretisch über 4700fps!
  - ca. 4200 Takte je Zeile

- BlockRAM
  - Bildcache für Rektifizierung - 342 KB je Bild
  - Zeilencache für Punktsuche - 66 KB je Zeile

3 Vorgehensweise und Lösungsansätze

FPGA Entwurf

- Punktsuche sehr langsam
  - sequentielle Suche dauert ca. 25000 Takte je Zeile
  - DRAM nur 4200 Takte je Zeile
  - parallele Punktsuche
    - in mehreren Zeilen → großer Zeilencache
    - zweifach je Zeile

- Subpixelinterpolation
  - parallelisierbar, jede Einheit verarbeitet einen Pixel
  - zwei Möglichkeiten:
    - nur eine Einheit (vollst. Pipeline)
    - mehrere iterativ arbeitende Einheiten
  - ca. 25 Multiplikatoren nötig

4 Diskussion

Fragen?

Vorschläge?

mögliche Probleme?

Alternativen?

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] A. Wiegmann et al. (2006): „Human face measurement by projecting bandlimited random patterns“, 21 August 2006 / Vol. 14, No. 17 / Optics Express 7698.

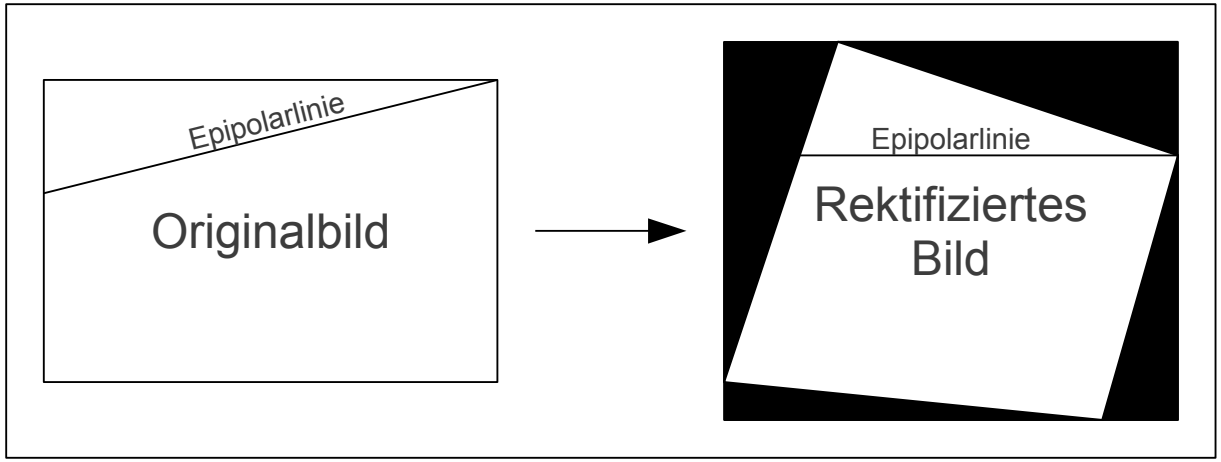
[4] www2.informatik.hu-berlin.de/cv/index.php?menu=teaching&submenu=overviewteaching&mode=overviewteaching

»Wissen schafft Brücken.«

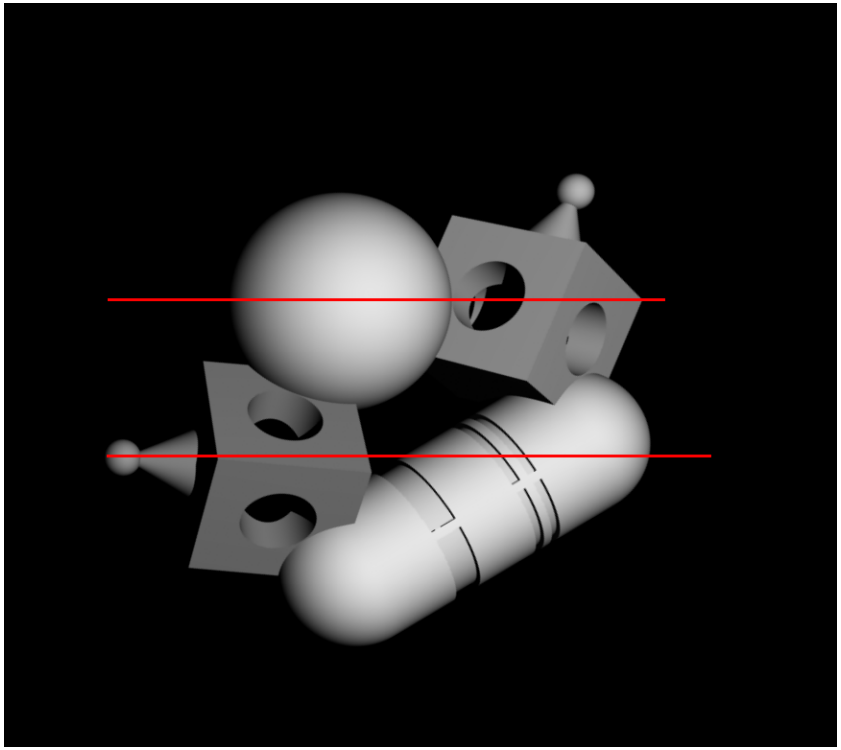
Anhang

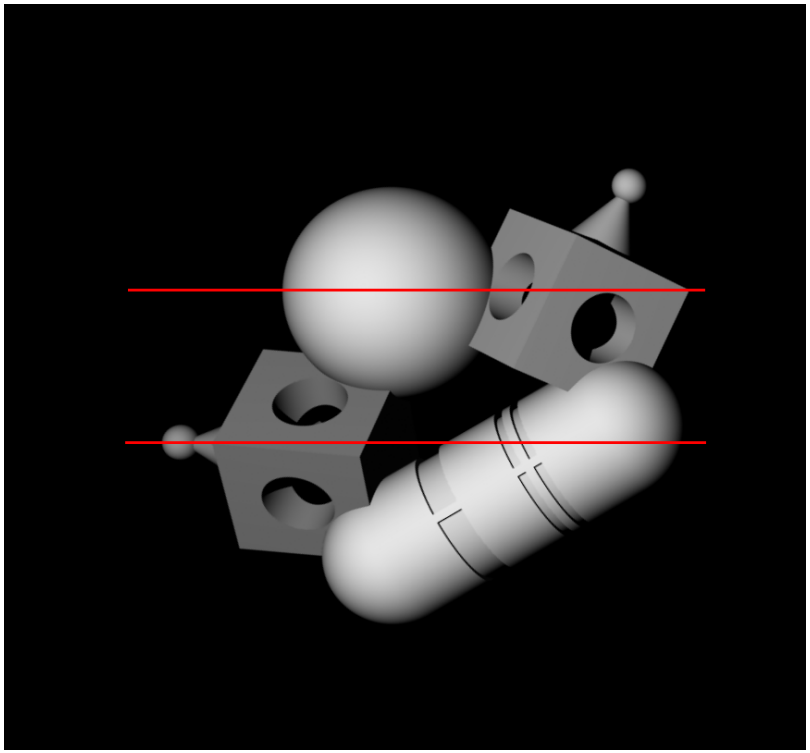
Video aus [1]

Anhang


Epipolarlinie
Originalbild
Epipolarlinie
Rektifiziertes Bild

Anhang





Anhang