

camera_rotate

(Laborprojekt)

Vorlesung: Labor Rechnersysteme

Erik Alexander Hennig

tae@erik-hennig.me

Technische Akademie Esslingen



Inhaltsverzeichnis

1 Aufgabenstellung	1
2 Implementierung	1
2.1 Parameter-Variation	2
2.2 Probleme	2
3 Performance	2
4 Fazit	2
5 Bibliographie	3

1 Aufgabenstellung

Ziel des Laborprojekts ist die eigenständige Implementierung einer Bildverarbeitung in Hardware. Als Rahmenbedingung gegeben ist die Verwendung des Kria KV260 Vision AI Starter Kits sowie die Nutzung der Xilinx Toolchain für High-Level-Synthese. Darüber hinaus ist die Wahl des Projekts offen. In diesem Fall wurde als selbstgestellte Aufgabe gewählt, ein aufgenommenes Kamera-Bild unter Zuhilfenahme eines Orientierungssensors so zu rotieren, dass es immer aufrecht angezeigt wird. Dabei soll die Rotation performant genug sein, um ein flüssiges Live-Bild anzusehen.

2 Implementierung

Um das Performance-Ziel zu erreichen, soll die Rotation des aufgenommenen Bilds in Hardware implementiert werden. Die übrige Logik wird nur in Software implementiert. Abbildung 1 zeigt die architekturelle Umsetzung des Projekts. Der Mikrocontroller-Baustein in der Mitte stellt den nicht-programmierbaren Teil des Kria-Boards dar. Dieser kommuniziert mit dem FPGA-Teil via AXI und AXI-Lite. Die Aufnahme des (verdrehten) Live-Bilds erfolgt mit einer per USB angeschlossenen Webcam. Zur Bestimmung der Kameraorientierung wird ein modernes Smartphone genutzt. Solche Geräte verfügen in der Regel alle über einen Orientierungssensor und sind leicht zu integrieren, da sie über zahlreiche Wege kommunizieren können. Das Smartphone wird mittels einiger Gummibänder mechanisch an die Webcam gekoppelt, sodass beide die gleiche räumliche Orientierung aufweisen. Zum Auslesen der aktuellen Orientierung implementiert das Programm auf dem Kria-Board in einem extra Thread einen minimalistischen Webserver. Der in der Website enthaltene Javascript-Code liest den Orientierungssensor des Smartphones aus und sendet ihn per HTTP zurück zum Webserver. Anhand der erhaltenen Orientierung kann dann das Kamerabild auf dem FPGA um die passende Gradzahl rotiert werden. Die Ausgabe des Bildes erfolgt per HDMI auf einem externen Display oder per SSH mittels X11 Display Forwarding auf einem weiteren PC.

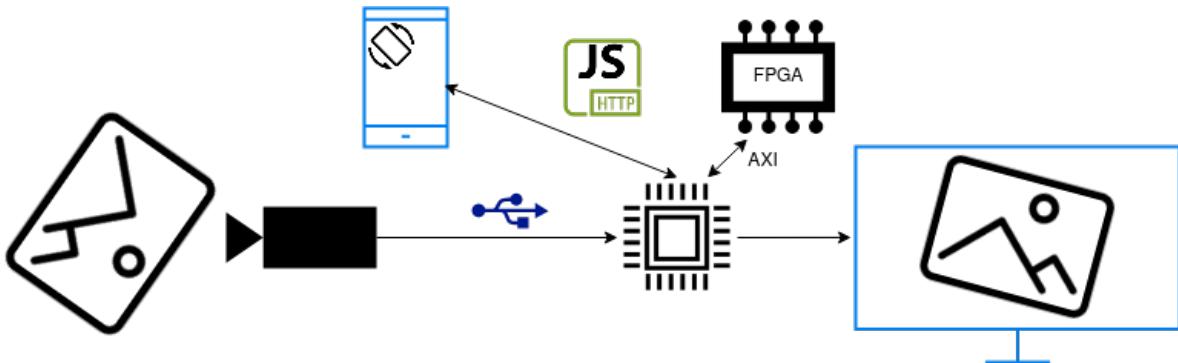


Abbildung 1: Kommunikationskanäle

Zur Umsetzung der Rotation in Hardware kommen folgende Funktionen der Vitis Vision Library auf den ersten Blick in Betracht:

- `remap` [1]: vertauscht Pixel anhand einer Relokationsmatrix
- `warpTransform` [2]: wendet eine affine Transformation auf das Eingangsbild an
- `rotate` [3]: rotiert das Eingangsbild um 90, 180 oder 270 Grad

Die ersten beiden Funktionen erlauben eine beliebige Rotationen. Allerdings sind sie nicht ganz so trivial zu verwenden wie `rotate`, das explizit nur für die Rotation implementiert wurde. Aus diesem Grund wurde `rotate` für die erste Implementierung genutzt und dann aus Zeitgründen auch nicht mehr ersetzt.

- Welcher Teil genau in Hardware?
- Welche HW-Funktion kommt in Betracht?
- Implementierung der Hw-Funktion

2.1 Parameter-Variation

2.2 Probleme

3 Performance

4 Fazit

- viel gelernt
- Doku könnte besser sein
- persönlich: hat Spaß gemacht
- performance? measure first? oder HW ist schnell

Die vollständige Implementierung des Projekts findet sich unter https://github.com/ede1998/camera_rotate [4].

5 Bibliographie

- [1] AMD Xilinx, „Vitis Vision Library API Reference: Remap“. [Online]. Verfügbar unter: https://xilinx.github.io/Vitis_Libraries/vision/2022.1/api-reference.html#remap
- [2] AMD Xilinx, „Vitis Vision Library API Reference: Remap“. [Online]. Verfügbar unter: https://xilinx.github.io/Vitis_Libraries/vision/2022.1/api-reference.html#warp-transform
- [3] AMD Xilinx, „Vitis Vision Library API Reference: Remap“. [Online]. Verfügbar unter: https://xilinx.github.io/Vitis_Libraries/vision/2022.1/api-reference.html#rotate
- [4] E. Hennig, „camera_rotate“. [Online]. Verfügbar unter: https://github.com/ede1998/camera_rotate