Fast GPU-based KLT feature point tracking using CUDA

Hannes Fassold, *Jakub Rosner, Peter Schallauer and Werner Bailer*

JOANNEUM RESEARCH
Forschungsgesellschaft mbH

Institute of Information Systems

Hannes Fassold

Steyrergasse 17
8010 GRAZ, AUSTRIA

Phone +43 316 876-1119
Fax +43 316 876-1191

hannes.fassold@joanneum.at

iis@joanneum.at
www.joanneum.at/iis

Introduction

- Automatic selection and tracking of feature points is a basic task for many CV algorithms (e.g. structure from motion, object tracking)
- Very popular due to good performance: KLT algorithm [Kanade, Lucas and Tomasi]
- Optimized OpenCV implementation available, but is not realtime capable (> 30 fps) for High Definition (HD) video
- GPUs provide a lot of computation power and can be used for general purpose compution (CUDA)
- Goal: Port KLT feature point tracker to CUDA to achieve realtime performance for HD video

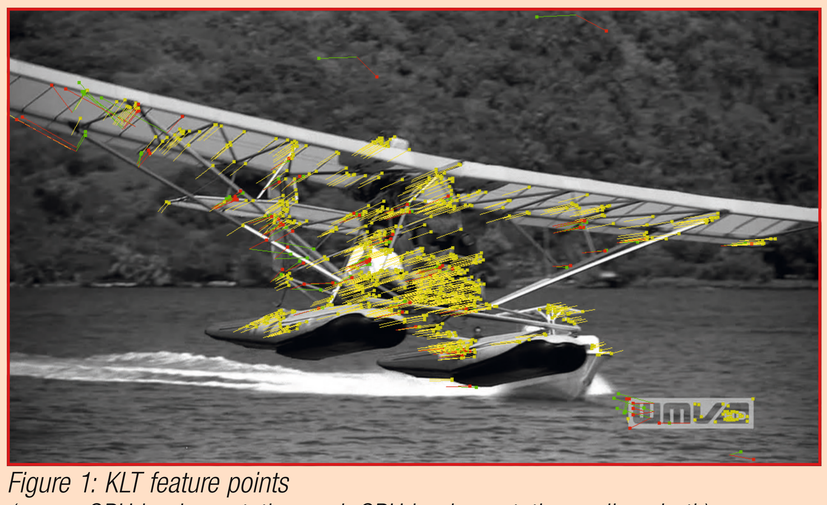

Figure 1: KLT feature points (green: GPU implementation, red: CPU implementation, yellow: both)

Feature point selection

- Algorithm
  1. Calculate cornerness for each pixel
  2. Enforce min. quality (5–10 % of max. cornerness) and do non-maxima suppression
  3. Enforce minimum distance between all feature points
- First two steps are done on the GPU
- Step 3 is inherently serial, therefore done on CPU
- Alternative method for step 3 was implemented (uses a mask image, much faster for many feature points)

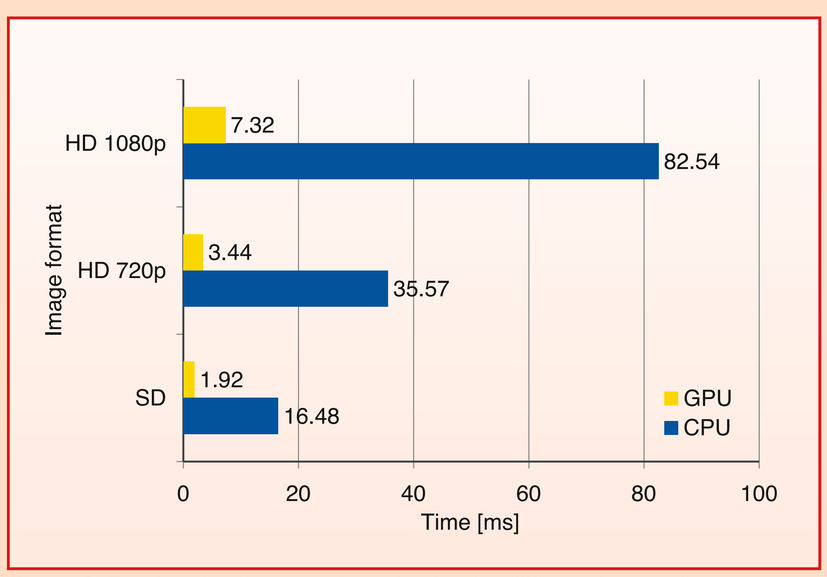

Figure 2: Runtime for feature point selection for different image sizes (without minimum distance enforcement)

Feature point tracking

- Algorithm
  1. Calculate image pyramids
  2. For each feature on every pyramid level (from coarse to fine), calculate the optical flow iteratively, using Gauss-Newton method
- All steps are done on the GPU (each feature point is one thread)
- For a low number of feature points (< 1000) modern GPUs are under-utilized

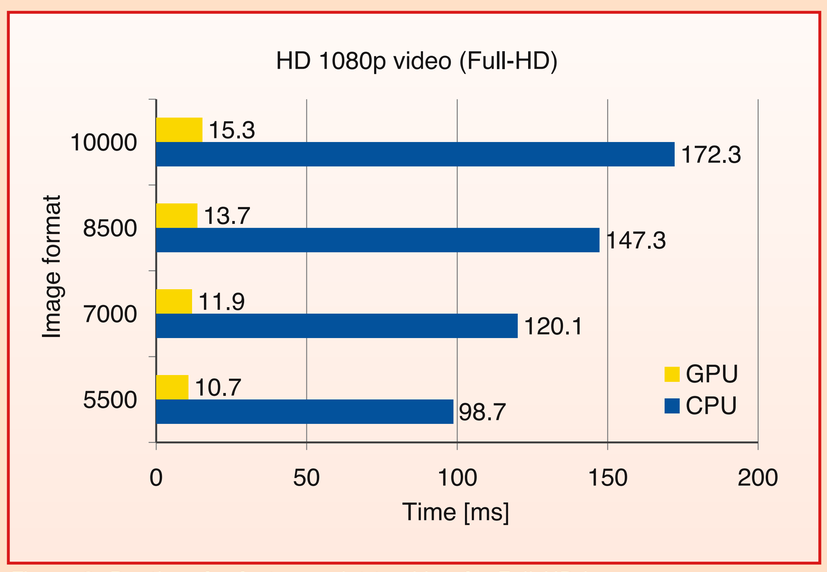

Figure 3: Runtime for feature point tracking for Full-HD images for different numbers of feature points

Evaluation

- Tracker quality (subjective assessment)
  - ➜ Most of the feature points and their optical flow are the same for CPU and GPU implementation
  - ➜ Differences mainly for feature points which seem to be not correctly tracked by both implementations
- Tracker runtime
  - ➜ CPU (OpenCV) routine uses IPP & OpenMP internally, runs on Intel Xeon QuadCore 2.4 Ghz
  - ➜ GPU routine runs on Geforce GX280
  - ➜ Achieves overall **5–10 times speedup**
  - ➜ Higher speedup for larger images and more feature points

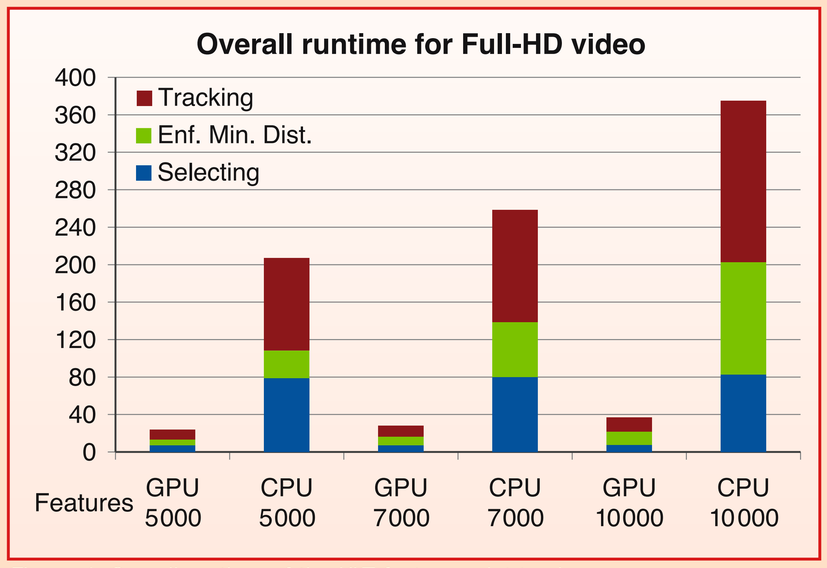

Figure 4: Overall runtime of the KLT feature point tracker for Full-HD (1920 x 1080) video

Conclusion

- Goal of HD realtime tracking achieved
- Key for successfully porting an algorithm to the GPU
  - ➜ Parallelization of the algorithm into a large number of loosely coupled threads possible
  - ➜ Knowledge of the GPU architecture (e.g the different memory types and their properties)

This work was partially supported by the European Commission

under the contracts

FP7-215475, “2020 3D Media – Spatial Sound and Vision”, www.20203dmedia.eu

FP7-231161, “PrestoPRIME”, www.prestoprime.org

and

FP7-216465, “SCOVIS – Self-Configurable Cognitive Video Supervision”, www.scovis.eu