15618 project: GPU object tracking

Denis Merigoux (dmerigou) & Ilaï Deutel (ideutel)

Tuesday, April 25^{th}

Abstract

We are going to implement an optimized object tracker on NVIDIA GPUs. The goal is to perform real-time object tracking. Here is the link to our GitHub repository.

1 Changes from initial proposal

After having reviewed [2] which presented the SCM algorithm we wanted to implement, we realized that the code provided by the authors wasn't a good starting point (Matlab 2009 code running only on a Windows XP virtual machine). Instead, we chose to take as a starting point the code for the KCF tracker [1] for which a sequential implementation exist in OpenCV. Our new goal is to parallelize this tracker to make it run on the GPU, using OpenCV's CUDA framework. Our work could have more potential and unsefulness as part of the OpenCV framework.

2 Work completed so far

The first task was to setup a build system, which is not easy given the intricacy of OpenCV, the various options needed to run the tracker or having CUDA support, and the lack of root privileges on the GHC machines. We had to developer various build scripts and try several configurations before reaching an optimal build workflow.

Then we proceeded to implemented a correctness test and a profiling analysis of the tracker code, to monitor our progress when parallelizing the code. We test the tracker on a 141-frames fullHD video, and take the average computing time over all frames for the profiling (see figure 1). The analysis of this profiling led us to focus our first efforts into parallelizing the Discrete Fourier transform used by KCF, task still under development since the beginning of the week.

3 Updated goals

3.1 Primary goals

- Having a correct sequential implementation of the algorithm using C++. **Done.**
- Analyse the workload and determine which bottlenecks we need to parallelize in CUDA. **Done.**
- Parallelize, verify correctness, compute speedup compared to sequential implementation.

Phase	Baseline implemtation time (ms)
Initalization	308.471
Average frame time	111.353
Detection	47.48
Extract and pre-process the patches	1.055
Non-compressed custom descriptors	0.193
Compressed descriptors	7.355
Compressed custom descriptors	2.774
Compress features and KRSL	9.372
Merge all features	1.558
Compute the gaussian kernel	20.087
Compute the FFT	1.748
Calculate filter response	3.101
Extract maximum response	0.236
Extracting patches	14.602
Update bounding box	0
Non-compressed descriptors	1.02
Non-compressed custom descriptors	0.196
Compressed descriptors	7.301
Compressed custom descriptors	3.058
Update training data	3.027
Feature compression	25.118
Update projection matrix	20.164
Compress	4.249
Merge all features	0.705
Least Squares Regression	22.638
Initialization	0
Calculate alphas	18.57
Compute FFT	1.758
Add a small value	0.378
New alphaf	1.095
Update RLS Model	0.837

Figure 1: Breakdown of times for each phase of the KCF tracker's procedure to update the bouding box for each new frame.

- Analyse the result of our implementation on the Need for speed dataset and compute the equivalent framerate at which our algorithm is capable of tracking the object.

The main grading criteria would be the framerate reached by our parallel algorithm.

3.2 Additional goals

- Find or implement an initialization front-end to trace bounding boxes around objects to track.
- Depending on the performance of the algorithm, use our program to actually track objects in real time from a camera feed using the initialization front-end.

3.3 Parallelism competition

We may present a graph showing plotting the average time taken to update one frame by the sequential algorithm and our parallelized algorithm, against the resolution of the video (larger resolutions should yield more opportunity for parallelism).

4 Revised schedule

Week	Task
04/10 – 04/16	Make a short bibliography about object tracking, build OpenCV and all its dependencies on the GHC machines
04/17 – 04/23	Analyse performance by timing each step to find the bottlenecks
04/24 – 04/30	Implement the parallelized version of the program
05/01 – 05/06	Finish optimizing the parallel version and run the benchmarks
05/07 – 05/12	If time left, complete the additional goals.

References

[1] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *CoRR*, abs/1404.7584, 2014.

[2] W. Zhong, H. Lu, and M. H. Yang. Robust object tracking via sparsity-based collaborative model. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1838–1845, June 2012.