15618 project: Multi-objects GPU tracking

Denis Merigoux (dmerigou) & Ilaï Deutel (ideutel)

Monday, April 10th

1 Summary

We are going to implement an optimized object tracker on NVIDIA GPUs using the SCM algorithm. The goal is to perform real-time object tracking.

2 Background

There are four main parts in a tracking algorithm (according to [1]):

- Object initialization: choosing what regions of an initial image corresponds to the objects to track.
- Appearance modelling: choosing how to represent visiually and statistically the features of the objects to track compared to the background.
- Motion estimation: determining what parts of the image have moved compared to the previous one.
- Object localization: localize the tracked objects on the new image.

The appearance modelling and motion estimation phase consists in statistical learning and optimization. We will base the appearance modelling part of our implementation on the SCM algorithm [2], which has been implemented in Matlab only by its authors. Although the paper only concerns the appearance modelling phase, it is designed to work with a particle filter for the motion estimation phase.

The SCM algorithm is described as follows by its authors: "We develop a sparsity-based discriminative classifier (SDC) and a sparsity-based generative model (SGM). In the SDC module, we introduce an effective method to compute the confidence value that assigns more weights to the foreground than the background. In the SGM module, we propose a novel histogram-based method that takes the spatial information of each patch into consideration with an occlusion handing scheme".

3 The challenge

The main dependency is temporal: you need to examine the frames in order, each one after the other. However, the algorithm present some points of synchronization that prevent total parallelization. The size of the working data is not that big (we only work one one image at a time) but the complexity of the computations in the algorithm is relatively low, so the dominant factor between data access and computation is unclear.

The challenge is then to optimize the parallelization rate of the algorithm, find out the bottlenecks and try to overcome them. In particular, we need to identify which is the slowest part of the four; if for instance motion estimation is slower than appearance modeling then we would spend more time optimizing it. Moreover, the goal is to perform the computation in real time with the best framerate possible (with a decent resolution).

4 Ressources

We will use the Matlab SCM code as a starting point (to translate to C++ and CUDA), and use OpenCV functions for the parts of the algorithm that we don't want to optimize.

We will use the GPUs of the GHC machines to run our program. If we make enough progress, we could try to test our program on the NSH camera with the Tegra X1 attached to it.

5 Goals and deliverables

5.1 Primary goals

- Having a correct sequential implementation of the algorithm using C++ (correctness compared with the Matlab SCM program).
- Analyse the workload and determine which bottlenecks we need to parallelize in CUDA.
- Parallelize, verify correctness, compute speedup compared to sequential implementation.
- 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 reach by our parallel algorithm.

5.2 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.

5.3 Platform choice

C++, CUDA, GPU and the OpenCV framework are sensible choices for image processing applications.

5.4 Schedule

Week	Task
04/10 – 04/16	Make a short bibliography about object tracking, understand the Matlab starter code and start implementing the sequential program
04/17 – 04/23	Finish the sequential program and 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] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony R. Dick, and Anton van den Hengel. A survey of appearance models in visual object tracking. *CoRR*, abs/1303.4803, 2013.

[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.