Particle Filter on GPUs for Real-Time Tracking

Antonio S. Montemayor* ESCET-URJC
Juan José Pantrigo ESCET-URJC
Ángel Sánchez ESCET-URJC
Felipe Fernández FI-UPM

1 Introduction

Efficient object tracking is required by many Computer Vision application areas like surveillance or robotics. It deals with state-space variables estimation of interesting features in image sequences and their future prediction. Probabilistic algorithms has been widely applied to tracking. These methods take advantage of knowledge about previous states of the system reducing the computational cost of an exhaustive search over the whole image. In this framework, posterior probability density function (pdf) of the state is estimated in two stages: prediction and update. General particle filters are based on discrete representations of probability densities and can be applied to any state-space model [Arulampalam et al. 2002]. Discrete particles j of a set $(X_t,\Pi_t) = \{(x_t^0,\pi_t^0)...(x_t^N,\pi_t^N)\}$ in time step t, contains information about one possible state of the system x_t^j and its importance weight π_t^j. In a practical approach, particle weights computation is the most expensive stage of the particle filter algorithm, and it has to be executed at each time step for every particle [Deutscher et al. 2000].

Consumer graphics processing units (GPU) have become inexpensive and programmable stream processors. Their programmable capabilities have drastically grown and this has helped the development of applications far beyond rendering purposes.

In this work, we have designed and implemented a preliminary real-time particle filter algorithm that makes use of a GPU to execute the algorithm's main performance bottleneck. Our strategy uses a texture multiplication for reducing the computational efforts generated by a sequential evaluation. This work presents some similarities with the work of [Oh and Jung 2004] applied to GPU implementation of neural networks.

2 Exposition

Figure 1 outlines an iteration of the particle filter algorithm. The performance of the filter has been tested on a rolling ball sequence. The actual frame of the sequence is loaded into main memory and N samples are taken using (x,y) coordinates stored in each particle. In the first iteration this sampling is randomly generated from a uniform pdf. In a measurement process N square windows are captured from the image using the coordinates given by the previous sampling stage. These windows are arranged and loaded in a first texture unit (`Tex0`). In another one (`Tex1`) a tracked shape template is loaded. This template consists of N repeated instances with the same size and spatial distribution of the target shape.

Particle weights computation is based on a template matching approach although we have previously sampled the original image at discrete locations instead of performing an exhaustive search. In order to improve weights computation a fragment program is created to carry out an effective hardware accelerated texture multiplication. The overlapping of both textures results the likelihood of a template given a measure. As we have chosen square regions for each measurement a mipmap reduction is accomplished to get an estimation of the particle weights π_t^j in each pixel position.

*e-mail: a.sanz@escet.urjc.es

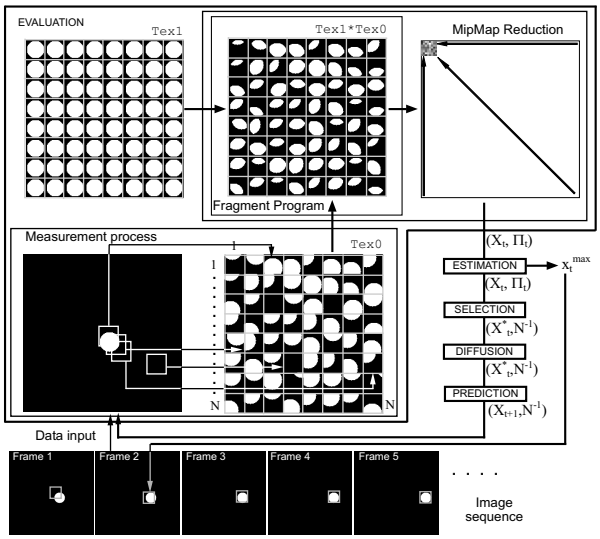

Figure 1: Hybrid GPU/CPU particle filter scheme.

Next, we get the rendering results to proceed with the following stages of the particle filter. The particle with the maximum weight x_t^{max} is selected as best candidate for the state of the system in the iteration. A new particle set (X_t^*,N^{-1}) at time t is created by selecting individuals from X_t with probabilities π_t^j. We use a roulette wheel as selection strategy. Since particles with larger weight values can be chosen several times, a Gaussian diffusion stage is applied to avoid loss of diversity. Finally, particle set at time $t+1$, (X_{t+1},N^{-1}), is predicted by using an appropriate motion model.

3 Conclusion

A preliminary real-time particle filter that exploits the intrinsic parallelism of GPUs architecture has been implemented. As future works we propose comparisons between GPU and optimized CPU implementations. Also it would be very interesting to study related realistic applications of this GPU particle filter framework such as articulated motion and multiple object tracking.

References

ARULAMPALAM, S., MASKELL, S., GORDON, N., AND CLAPP, T. 2002. A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing 50*, 2, 174–188.

DEUTSCHER, J., BLAKE, A., AND REID, I. 2000. Articulated body motion capture by annealed particle filtering. In *Proc. of the IEEE Conf. on CVPR*, vol. 2, 126–133.

OH, K.-S., AND JUNG, K. 2004. Gpu implementation of neural networks. *Pattern Recognition 37*, 1311–1314.