High Speed Articulated Object Tracking using GPUs: A Particle Filter Approach

Raúl Cabido, David Concha, Juan José Pantrigo, Antonio S. Montemayor
Departamento de Ciencias de la Computación
Universidad Rey Juan Carlos, Móstoles, Madrid, Spain
Email: {raul.cabido, david.concha, antonio.sanz, juanjose.pantrigo}@urjc.es

Abstract—This paper presents a novel application of the GPU processing power to a very computationally demanding articulated human body tracking problem in a view-based approach. This work includes some optimizations at the algorithmic level as well as some tricks at the implementation level using OpenGL and shader programming. An underlying particle filter framework is combined with a novel particle weight computation, where heterogeneous templates are considered for distribution mode recovering. Also, a form of elitism is taking into account to prevent flickering when the best candidate of the particle population is chosen. Impressive performance up to 317-713 frames per second is guaranteed for common configurations of about 1024-256 particles and 320×240 video resolutions.

I. INTRODUCTION

Graphics Processing Units (GPUs) are consumer and high performance hardware that can be programmed for obtaining solutions to general purpose problems. Ideally, these problems should present a high arithmetic intensity and low communication among data. Image processing and computer vision are very promising fields for exploiting hardware platforms like FPGAs or GPUs. Many and independent pixel operations easily offer a data-parallel framework for exploiting hardware resources, vector operations and high memory bandwidth.

Many applications areas have introduced GPUs into their platform lists. One of the first works on using GPUs for accelerating image processing tasks was presented in [18], where the authors demonstrated an improvement of about 30% over optimized CPU solutions in segmentation problems by direct pixel manipulations with a graphics card. For direct pixel operations, the GPU is very efficient and offers very high throughput because it is basically designed for this purpose in its computer-generated graphics context. With the advent of high level abstraction layers, like high level shading languages and, especially, Nvidia C for CUDA (Nvidia CUDA architecture) and OpenCL proposals, GPU solutions have been extended to more general domain problems, including evolutionary computation, numerical optimization and simulation, clustering problems, etc.

Visual tracking consists of locating or determining the configuration of a known (moving, deforming) object at each frame of a video sequence [12]. This is a relevant problem in computer vision and it has been focused using different methodologies. Depending on the kind of object or the kind of scene, the problem can lead to blob tracking, multiple object tracking, articulated object tracking, deformable object tracking or any combination of these [19].

In this work, we present a novel application of the GPU computational model to a complex 2D visual tracking problem, such as the articulated human body tracking, over the OpenGL environment and shading techniques instead of higher level abstraction strategies. For visual tracking purposes the particle filter takes a large number of hypotheses about the system state and evaluate them independently. We demonstrate how the adaptation of a particle filter to the GPU platform may lead to a considerable performance improvement that can be used as an underlying framework for further algorithmic optimizations. Additionally, we present two improvements over the algorithm adaptation to the GPU: geometric templates and elitism.

A. Particle filtering for visual tracking purposes

One of the most popular approaches to visual tracking problems in recent years is the Particle Filter (PF) proposed in [7]. PF is based on discrete representations (called particles) of the probability density function (pdf) which describes the evolution of a given system [2]. To achieve this goal, PF combines adaptation and prediction strategies. Isard and Blake adapted this algorithm to be applied to visual tracking in the middle 90’s [8] and later, they proposed the CONDENSATION (CONdititional DENsity propagaTION) algorithm [9]. This proposal makes possible the contour tracking in video sequences and, nowadays, constitutes the basis of most tracking algorithms based on probabilistic principles.

As multiple estimations are simultaneously considered, PFs are specially well designed to be performed in a data-parallel fashion. In Montemayor et al. [13] the authors modeled a PF evaluation on GPU that was applied to visual tracking purposes. Only the particle weight computation was accelerated. However, all its algorithmic stages were accelerated using Shader Model 3.0 capabilities in [14] and a deeper study with extended local search processes for multiple object tracking problems was presented in [1].

B. Articulated Human Tracking

There are several papers in the literature which addresses the problem of articulated motion tracking using particle filter algorithms. Partitioned Sampling (PS) [11] is a statistical approach to tackle hierarchical search problems. PS consists of dividing the state space into two or more partitions, and
sequentially applying the dynamic model for each partition followed by a weighted resampling stage. Deutscher [5][6] developed an algorithm referred to as Annealed Particle Filter (APF) for tracking people. This is one of the first PF-based algorithms successfully applied to a multi-dimensional visual tracking problem. APF achieves a performance of about 1/60 frames per second (fps) on a 3D full-body tracking problem with 29-34 degrees of freedom (DOFs).

There are several algorithms based on the hybridization of PF and genetic algorithms [3][6]. In [3], a genetic algorithm particle filter is used to track a human hand but performance results were not provided.

In [17] a detection and tracking scheme is proposed to track the face, arms and forearms of a user. This method achieves a performance of 10 fps on a 6 DOFs tracking problem. In [16] two articulated tracking approaches were presented: the decentralized articulated object tracking method (DAOT) and the hierarchical articulated object tracking method (HAOT). Experimental results in a 3.2 GHz Pentium 4 and C++ programming language demonstrated a performance of 9-10.2 fps for DAOT and 7.4-8.3 fps for HAOT on real-world videos (12 DOFs problems). The same work also reports 26 fps for a 5 DOFs problem and 15 fps for a 8 DOFs problem. They also compare their algorithms against [10] in which a joint particle filter (JPF) performs at 13-14 fps with 300 particles on a 5 DOFs problem and 0.3-0.4 fps with 2500 particles on a 8 DOFs problem in the same platform.

Pantrigo et al. [15] propose an algorithm which hybridizes particle filters (PF) and the scatter search (SS) metaheuristic, called scatter search particle filter (SSPF) for 2D articulated object tracking problem. Authors reported processing rates of about 1/46 fps on a 7 DOFs tracking problem.

As it can be seen in the methods proposed in the literature, the high number of DOFs in articulated motion tracking feature spaces results in a large particle requirement [4]. The evaluation of a weighting measure for each hypothesis makes articulated object tracking a computationally very demanding task [4].

II. OUR PROPOSAL: ACCELERATED PARTICLE FILTER FOR ARTICULATED OBJECT TRACKING

Our proposal integrates the application of an accelerated particle filter using the GPU computational power to a very demanding articulated human motion tracking problem, in which some optimizations have been performed over the algorithm to produce a more robust and reliable tracking.

A simplified vision of the particle filter for visual tracking purposes begins with a population of estimates, which are a set of possible states of the system configuration. This configuration depends on the geometrical model that we use for the articulated structure.

A. Geometrical model

Figure 1 shows an upper body of a person that is going to be tracked over a sequence. For such a task we model the limbs and body parts with six segments connected with joints, parameterized by eight variables $s = (x, y, \alpha, \beta_1, \beta_2, \gamma_1, \gamma_2, \kappa)$ as showed in Fig. 2 which define the 8 DOFs problem. These variables form the state space of the system, where different selections of them make different states of particles. Basically, we keep track of a reference position $(x, y)$ and the relative orientation $\alpha$ of the trunk, the different angles $(\beta_1, \beta_2)$ for the arms with respect to the trunk, $(\gamma_1, \gamma_2)$ for the forearms with respect to their respective arms and $\kappa$ for the head orientation with respect to the trunk.

B. Particle Filter Adaptation to the GPU

The particle filter is composed of several algorithmic stages: an initialization at time 0, and evaluation, selection, diffusion and prediction stages at every time step $t$. After the evaluation stage, the best estimator is considered as the state of the system at time $t$. However, first of all, we need a preprocessed frame in order to get measurements from the image.

In order to extract the foreground objects from the image we upload the video frame at time step $t$ and an offline grabbed reference background image to video memory using two textures. Then, we perform a background subtraction fragment shader on every pixel. The result is binarized using a threshold and is kept in a segmentation texture.

The geometrical model for a particular pose (like the one shown in Fig. 2) can be packed into a square tile taking into account a data association with the 8 parameters which describe the pose (see Fig. 3). Doing so for many different poses we can evaluate them using the PF framework assigning weights
to each descriptor. A large texture (typically 1024 × 1024) containing a number of poses is then created by texturing from the segmentation texture a collection of individual Quad primitives, each one including a considered pose (for example a tile of 64 × 64 pixels). An example of this measurement texture is shown in Fig. 4.

Note the binary behavior of the texture after the background subtraction shader and the packed poses. The size of this measurement texture determines the number of particles that we are able to process on the GPU. Then, for a measurement texture of size 1024×1024 and tiles of 64×64 we can organize up to 256 particles (a 16 × 16 grid of tiles) in one color channel, if we would use a 2048 × 2048 container we would encode 1024 particles. These particles (states) are maintained in two RGBA particle textures (8 values corresponding to \( s = (x, y, \alpha, \beta_1, \beta_2, \gamma_1, \gamma_2, \kappa) \) variables).

Each particle is evaluated in parallel counting the number of labeled pixels (foreground pixels) that are included in its corresponding tile (see again Fig. 4). This value is used for weighting the particle. The parallel implementation of this evaluation is done by a sum-reduction process of the whole texture to the size of the number of particles (for example, from 1024 × 1024 to 16 × 16, that is dividing the original size of the texture, 1024 × 1024, by the dimension of the tile, 64 × 64, which results 256 values). Each value of the resulting texture is the weight of the corresponding particle of the previous particle textures.

Once the particle set has been evaluated we extract the estimator of the system at time \( t \) as the state with the largest weight in that time step. Figure 5 shows all the particles’ state after the weight computation stage (shown in blue) and the best one (in black). Then, we replace every state of the set with the best one in order to get a new, and improved, particle set. To avoid repetition, we apply a diffusion process to create a diversification of the population. This diffusion moves the states randomly through the state space adding a random offset (\( \Delta x, \Delta y, \Delta \alpha, \Delta \beta_1, \Delta \beta_2, \Delta \gamma_1, \Delta \gamma_2, \Delta \kappa \)) to each variable of the state. If we had a known motion model we could apply it to every state, if not, the diffusion process finishes the algorithmic stages of the PF. However, an entirely GPU accelerated particle filter needs a feedback stage to move the final particle set (the a priori estimation at time \( t+1 \)) to the beginning of the rendering pipeline for the new time step. Each estimator \( s_i \) described by its state \( (x_i, y_i, \alpha_i, \beta_{1i}, \beta_{2i}, \gamma_{1i}, \gamma_{2i}, \kappa_i) \) should create the new pose for its evaluation. We perform this feedback with the aid of rendering to a texcoord array (RTTA).

A typical problem in this straightforward model of evaluating poses is that a particle with maximum weight may arise by collapsing all the segments into the body, and thus containing segmented pixels which increase its weight (see Fig. 6). Some authors [15][5] have proposed the use of an edge measurement model in addition to the blob structure created by the background subtraction in order to penalize limbs collapsing to a low edge density area (trunk). In this work, we propose the use of templates in order to privilege geometrical shapes of every limb. More about these templates will be explained in the next subsection.

Another inconvenience of the particle filter is the shaky estimation along the time because of the probabilistic nature of the filter and the fact that more than one position of a segment may contain the same number of labeled pixels than another and then get the maximum. To avoid this problem, we use a form of elitism.
C. Algorithmic improvements: Geometric templates and elitism

In order to track a limb with the simple heuristic of counting the number of labeled pixels that our connected segments include from the blob image (segmented texture), we need a refinement method to avoid failure because of the collapse of the segments into the trunk in case of open arms. This problem arises because the majority of labeled pixels resides inside the trunk and the states get more weight in that kind of configuration. To avoid this problem we propose the use of a weighting mask as a template in the evaluation stage for every pose. This template enhances or penalizes geometrical features of different limbs if the measurement does or does not properly match the template. Figure 7.a shows the tile of measurements where we encode the head (upper left), trunk (bottom left), arms (upper right) and forearms (bottom right) of a pose. Figure 7.b shows a proposed weighting mask for an entire pose, where red color regions are 1 and green and yellow are negative values. In this example only the head and forearms are transformed with the geometric template. The trunk and the arms are not filtered assuming that they will occupy the entire region of the segment. Multiplying as an element-wise product this weighting mask by the pose tile we can enhance limbs of the pose if they match inside the red regions of the mask, otherwise the mask will penalize labeled pixels that coincide with negative values of the template. Figure 7.c shows a simulation of the 3D profile of the geometrical template associated to a foreground. Positive values are in the middle while negative values are located at the extremes along the principal axis. A large texture (of equal size of the measurement texture) with the repeated pattern of the geometrical template is used at the evaluation stage in order to weight every pose encoded in the measurement texture.

Elitism is known in the genetic algorithm terminology as a way to privilege a solution, remove it from subsequent processes (combination, mutation, etc.) and promote it directly to next iterations. In the particle filter framework we use elitism to privilege the state of the best estimator at time $t - 1$ if its weight at time $t$ prevails over the rest. Then, that estimator survives a number of frames if there is no better candidate. The direct consequence of this is a smooth motion on the estimation and a more reliable tracking. In terms of computational cost, this strategy is almost free.

III. EXPERIMENTAL RESULTS

In Table I we show the performance of the proposed method using different GPUs and particle filter configurations. In particular, three experiments using different number of particles are shown (256, 1024 and 4096 particles) for $320 \times 240$ and $640 \times 480$ video resolutions. We can clearly note how well the scaling of the particle filter is while decreasing the number of particles. For the evaluation of 4096 particles we need at least two $2048 \times 2048$ RGBA float textures, then, the need of large onboard memory makes the experiments unavailable to some platforms (those GPUs with 256MB onboard). Apart from the impressive performance of the latest GPUs up to 700 processed frames per second, we can see how a 2005 graphics processor can deliver more than 200 fps for the computationally expensive articulated object tracking problem. Performance results without the optimization strategies (geometric templates and elitism) are quite similar to those with the complete process. In fact, both of them lie in the error margins of the observed ratios shown in Table I.

Figure 8 shows the tracking performance of three different methods: particle filter (left), particle filter with the use of geometric templates in the PF evaluation stage (middle) and the proposed particle filter with the use of geometric templates and elitism (right). We can clearly note how the stand-alone particle filter collapses before reaching frame #30 and it is not able to recover the tracking. However, the other two methods performs well along the whole sequence. This performance is basically caused by the use of geometric templates, as they
can contain the tracking of the borders of the limbs with those negative weights showed in Fig. 7.b and avoid the model to collapse into the trunk. The performance of the elitism is more difficult to distinguish between both sequences.

Figure 9 shows an experiment where an image is repeated 10 times simulating a sequence with a static object. The particle filter estimation without the elitism proposal is shown on the left side of the figure, while the particle filter with elitism is shown on the right side. On one side, we can see how the particle filter without elitism offers a shaky estimation along the 10 static frames. On the other side, elitism based tracking gets an extremely stable estimation with this simple example, and the same candidate survives as the best one along the whole sequence.

IV. CONCLUSIONS AND FUTURE WORKS

The particle filter framework is widely used as an approach to some estimation problems. For visual tracking purposes it takes a large number of hypotheses and it is a recognized computationally expensive method, especially in multidimensional problems such as articulated human motion tracking. As far as authors know, described methods in the literature for the articulated human motion tracking never offered a throughput of more than 20 frames per second of performance. In this work we have demonstrated the great benefit of a scalable many-core architecture such as the GPU. It is important to remark that present and future consumer architectures are
evolving to multi and many-core processors. So we would insist on developing highly parallelizable methods whenever possible, as in near future only these kind of techniques will continue evolving along with the architecture. Particle filter is a well data-parallel method that will scale its performance as the architecture moves forward. In this work we took care of choosing parallel strategies to improve the particle filter estimation. Instead of mean-shift or iterative local search methods we have proposed a template weighting stage in order to find the mode of the distribution allowing a better tracking of the limbs. This template refinement is embedded into the particle filter weighting stage ensuring its parallel nature. Additionally, we have proposed a very simple elitism step to refine the shaky trajectory of a particle filter tracking process, instead of trajectory analysis using historic estimations, which are very prone to iterative processes. Execution time results are impressive, and the system offers up to 700 frames per second of quite robust tracking for the evaluation of 256 particles with a state of six variables constrained by the human body joints in $320 \times 240$ video resolutions.

However, the PF tracking presented in this work may not support occlusion events and it even could also collapse if the limbs enters in the trunk projection area. As a future work, we would include edge measurements in addition to the back-ground subtracted blob structure used in this work. There are well known edge detection methods which fit into the parallel ground subtracted blob structure used in this work. There are methods we have proposed a template weighting stage in order to find the mode of the distribution allowing a better tracking of the limbs. This template refinement is embedded into the particle filter weighting stage ensuring its parallel nature. Additionally, we have proposed a very simple elitism step to refine the shaky trajectory of a particle filter tracking process, instead of trajectory analysis using historic estimations, which are very prone to iterative processes. Execution time results are impressive, and the system offers up to 700 frames per second of quite robust tracking for the evaluation of 256 particles with a state of six variables constrained by the human body joints in $320 \times 240$ video resolutions.

However, the PF tracking presented in this work may not support occlusion events and it even could also collapse if the limbs enters in the trunk projection area. As a future work, we would include edge measurements in addition to the back-ground subtracted blob structure used in this work. There are well known edge detection methods which fit into the parallel model, such as those based on 2D convolutions. Another future work is the RGBA vectorization of the preliminary information (PF states and measurement texture) which would increment the maximum number of particles by a factor of $\times 4$.

**ACKNOWLEDGMENT**

The authors would like to thank the Spanish Ministry of Education and Science that has been supported this research by CICYT TIN2008-06890-C02-02, URJC and CAM by URJC-CM-2008-CET-3625, and the Nvidia Professor Partnership Program.

**REFERENCES**


