21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Parallelized Annealed Particle Filter for Real-time Marker-less Motion Tracking via Heterogeneous Computing

Yatao Bian, Xu Zhao, Jian Song, Yuncai Liu
Shanghai Jiao Tong University
{*bianyatao,zhaoxu,whomliu*}*@sjtu.edu.cn*

Abstract

We propose a parallelized Annealed Particle Filter method via heterogeneous computing (P-APF), to implement real-time marker-less motion tracking based on OpenCL framework. The overall computing procedure in P-APF is decomposed into several computational tasks with corresponding granularity. According to the degree of parallelism, the tasks are assigned to standard and attached processors respectively, to fully leverage heterogeneous computing ability. A task latency hidden strategy is used to further reduce time cost. Experiments on different human motion datasets demonstrate that P-APF can achieve real-time tracking performance without losing accuracy. With an average acceleration ratio of 106 compared to serial implementation, the time cost basically remains constant with the growth of particle number and view number in a limited range.

1. Introduction

3D marker-less motion tracking is one of the most significant yet challenging tasks of visual human motion analysis. The applications range from virtual realization, film animation to sports and medical treatment, etc. Current works [7, 8, 9] mainly concern conducting motion tracking accurately, while ignoring its computational efficiency for real-time application. Annealed Particle Filter (APF) [2] is capable of recovering full articulated body motion efficiently, however, as a particle filter method, its computational cost is prohibitively large for practical application due to the evaluation to likelihood function, which has to be performed once at every time step for every particle. It has been a severe bottleneck of APF. The emerging of heterogeneous computing [4] and heterogeneous programming framework such as Open Computing Language (OpenCL) [5] provides an efficient way to widen this bottleneck.

In this paper, we propose to embed APF implementation for human motion tracking into heterogeneous computing framework. The principal contributions of this paper lie in two aspects. (1) Develop parallelized annealed particle filter via heterogeneous computing to conduct real-time marker-less motion tracking. (2) Use task latency hidden strategy to overlap extra data transfer time and computational time. Performance of the proposed system is demonstrated by experiments conducted on different human motion datasets with different actions. Its performance was also tested in 2011 AMD China acceleration computing competition[1], in which we got the first place from more than 100 teams.

2. APF Based Human Motion Tracking

The articulated model of human body used in this paper (Figure 1(a)) consists of 10 truncated cones (green), representing limbs, torso and head, and 15 joints (red points). This model has 31 DOFs comprising the position and orientation of the torso and the relative joint angles between limbs. Figure 1(d) shows tracking result under this model.

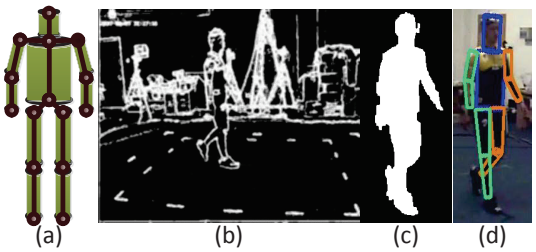

Figure 1. Human motion tracking (a) model (b) edge distance map (c) silhouette map (d) result

Tracking problem is formulated as one of estimating the posterior probability distribution $P_t^+ \equiv p(x_t|y_{1:t})$

[1] Website: http://accontest.eol.cn/.

for the state x_t of the human body at time t given a sequence of image observations $y_{1:t} \equiv (y_1, ..., y_t)$. In APF, distributions are represented by a set of 31D particle vectors with associated normalized weights $\{x_t^i, \pi_t^i\}_{i=1}^N$, which are propagated over time using temporal dynamics and assigned new weights according to likelihood function. Then, APF searches for peaks in the posterior distribution using simulated annealing. The negative log-likelihood of silhouette (Figure 1(c), generated by algorithm in [7]) is estimated by projecting a number of visible points on each limb into the silhouette image and then computing the mean square error (MSE) [3].

$$-\log p(y_t|x_t) \propto |\{\xi\}|^{-1} \sum\nolimits_{\xi} (1 - M(\xi))^2, \qquad (1)$$

where $\{\xi\}$ is the set of projected points and M is the silhouette map. Edge-based likelihood is computed in a similar way.

3. Parallelized Annealed Particle Filter

Heterogeneous architecture consists of standard processor such as traditional multi-core CPU and attached processors such as Graphic Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), which are dedicated stream accelerators containing hundreds of lightweight cores. Heterogeneous computing aims to combine standard processor's ability for general computing and attached processors' ability for high intensive computing to get better performance for most applications. In this section we will formulate P-APF via heterogeneous computing in detail.

3.1. Task Decomposition and Assignment

We firstly analyze conventional APF to find its parallel part, the algorithm is decomposed into several tasks with corresponding granularity as shown in Figure 2 (blue and green rectangles). We can then assign the general computing tasks to standard processor (host) and intensive computing tasks to attached processor (device). The criteria of each task's assignment to host or device is degree of parallelism (DOP), which is the number of independent parts that can be computed in parallel for each task. The first 5 tasks with highest DOP are shown in Table 1, which will be assigned to the device. Detailed description about each task's DOP is in the following.

1) 3D cones projection: project each 3D truncated cone to every 2D camera plane for every particle, this procedure is independent with each other, so its DOP is $N_p * N_{tc} * N_{view}$.

Table 1. The five tasks with highest DOP
N_{tc}: truncated cone number; N_p: particle number; N_{view}: view number

3D cones projection	Silhouette likelihood	Edge likelihood	Particle selection	Skeleton projection
$N_p * N_{tc} * N_{view}$	$N_p * N_{tc} * N_{view}$	$N_p * N_{tc} * N_{view} * 2$	$N_p * 3$	$N_{tc} * N_{view}$

2) Compute silhouette likelihood for particle $x_t^i (i = 1, ..., N_p)$ is reformulated as the following equation from equation (1):

$$-\log p(y_t|x_t^i) \propto \sum_{j=1}^{N_{view}} \sum_{k=1}^{N_{tc}} |\{\xi\}|^{-1} \sum_{\xi_{jk}} (1 - M_j(\xi_{jk}))^2, \qquad (2)$$

where M_j is the silhouette image from view j. ξ_{jk} is the set of projected points from truncated cone k to M_j. In this task, silhouette likelihood should be computed for every particle, so its DOP is $N_p * N_{tc} * N_{view}$.

3) Compute edge likelihood: similar to the analysis of 2), consider that ξ_{jk} is the set of projected points along the edges of truncated cone k and computing along two long sides of one specific cone is the same, DOP of computing edge likelihood is $N_p * N_{tc} * N_{view} * 2$.

4) Particle selection is the task to check for angles exceeding anatomical joint limits and for inter-penetrating limbs. We only check three pairs of limbs, so the DOP is $N_p * 3$.

5) Skeleton projection is the process to project each truncated cone of the optimal model configuration x_k to the 2D camera plane, its DOP is $N_{tc} * N_{view}$.

We represent the procedure of P-APF by a flow chart in Figure 2, the yellow rectangles is extra data transfer tasks between host and device.

3.2. Parallel Computing on Attached Processor

In OpenCL platform model, each device has thousands of processing elements within which computations on a device occur. For one specific task, parallel computing on device should be arranged effectively to maximize the use of attached processor's highly intensive computing ability. Here we just take silhouette likelihood computing for example, it has a DOP of $N_p * N_{tc} * N_{view}$. In practice, N_p can be several hundred, N_{tc} is 10 in our human skeleton model and N_{view} may take from 3 to 16. Considering factors such as data reduce using shared local memory and algorithm expansibility on particle number, we choose to use a two-dimensional workgroup whose indexes are view number and cone number, each workgroup is responsible

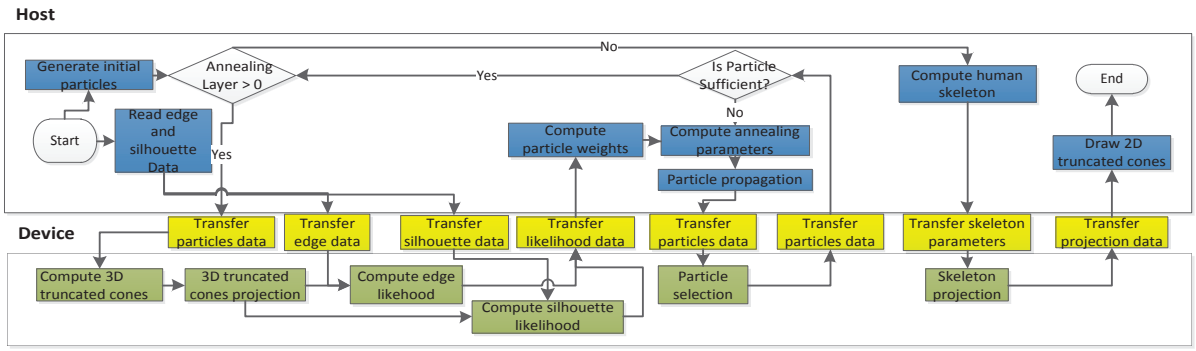

Figure 2. Flowchart of P-APF

for the computing of one specific particle, as shown in Figure 3.

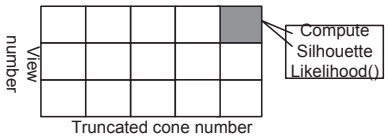

Figure 3. Workgroup index space

3.3. Hardware-independent Latency Hidden Strategy

It pays for distributing tasks on host and device, that is, some extra time overhead for data transfer between host and device has to be considered (yellow rectangles in Figure 2). For example, in order to compute weights for each particle, edge and silhouette data must be transferred to the device. Data transfer task between host and device is notably very time-consuming, sometimes it even costs half of the time after parallelization. Meanwhile, data transfer between host and hard disk or cameras is also time-consuming when we want to conduct real-time processing. To tackle these problems, we propose a task latency hidden strategy using task parallelization.

To achieve optimal task latency hidden, we seek to parallelize tasks on host, tasks on device and tasks for data transfer without ignoring the dependency relations among them. This can be formulated as a multi-thread task scheduling problem, which aims to assign the tasks to the threads so that the precedence relations are maintained and all of the tasks are completed in the shortest time.

To complete the formulation, we need to know the cost time C_i for each task T_i. However, C_i is not constant, it varies with the performance of standard processor and attached processor. As a result, for one specific machine we must run the algorithm first to get accurate task time cost C_i. Then one computational task can be represented by a task-time pair $\{T_i, C_i\}$. We get a task schedule heuristically using $N_{view} + 4$ threads: N_{view} threads for reading data from cameras, one main thread for tasks on host, two threads for transfer edge and silhouette data to device because they are independent with each other, one thread for reading result data of device.

4. Experiments

To demonstrate performance of P-APF for real-time human motion tracking, we implement the other two C++ versions of APF, one is the serial program (S-APF) and the other is parallelized APF using multi-threads on multi-core CPU (T-APF), which is different with that using heterogeneous computing since CPU cannot hold as many threads as attached processor (GPU). Intel Threading Building Blocks[2] (TBB) is used to build the program. A motion dataset[3] with ground truth motion made by Brown University in [6] is used to quantitatively compare tracking accuracy and time cost with different particle number N_p. To analyze time cost for different view number N_{view}, we use a more challenging motion dataset, PEAR, in which 16 calibrated cameras with resolution of 704*576 capture more kinds of actions including jumping, jogging and skipping. All the experiments were carried out on a standard PC with 4-core CPU and AMD HD 7970 GPU.

4.1. Tracking Accuracy Evaluation

Weighted error measurement in [1] is used for accuracy evaluation. Four trials of each experiment are

[2]Licensed under GPLv2 with the runtime exception
[3]http://www.cs.brown.edu/~ls/Software/

performed in which N_p was set to 200 and layer number 10. We plot the mean error at each frame and compute an average error and standard deviation over all 500 frames in Figure 4. It is apparent that three implementations have the same tracking error. The average error of about 77.3 millimeter is caused by low quality silhouette data used here.

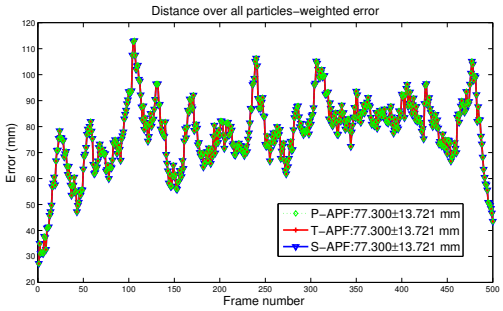

Figure 4. Performance of accurancy

4.2. Time Cost and Scalability

We compare time cost per frame T_f over N_p and N_{view} with annealed layer number set to 10. N_p or N_{view} can't be too less for successful tracking, so we set N_p ranging from 50 to 400 with 8 viewpoints and N_{view} from 3 to 12 with 200 particles. We did four trials with 100 frames for each configuration and plot mean T_f in Figure 5. Average T_f over all particle number and view number is also computed.

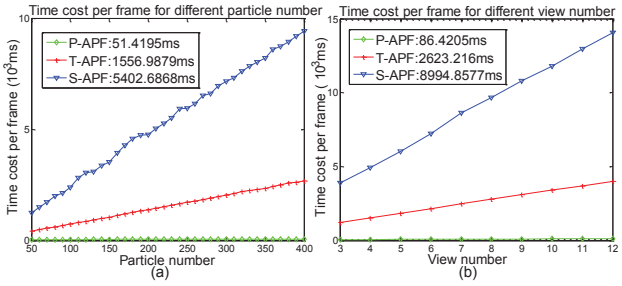

Figure 5. Time cost per frame

In Figure 5(a), with an average acceleration ratio of 106 (Tabel 2), T_f is 41 milliseconds for a common configuration of N_p 200, layer number 10, which achieves real-time processing. T_f of P-APF remains basically constant with increasing of N_p and N_{view} while it increases dramatically for the other two implementations, which is the so-called "weak scalability" of parallel program on heterogeneous architecture, that is, number of "light threads" on attached processor increases with number of particle or view increasing, so the overall time cost remains basically constant or little increment for extra data transfer overhead. This is an attractive property of P-APF since applications with high dimensional configuration need more particles for higher tracking accuracy, more views will contribute to accuracy as well. This property will not break down until the available intensive computing resources are used out. However, civil attached processor such as common GPU has enough resources for general intensive computing demand, as shown in Figure 5, it can hold a high particle number of 400.

Table 2. Acceleration ratio on some N_p

Particle number	100	150	200	250	300	350	400
Acceleration ratio	55	79	99	116	127	137	147

5. Conclusion and Future Work

In this paper we develop parallelized annealed particle filter via heterogeneous computing to conduct real-time marker-less motion tracking and use task latency hidden strategy to further reduce time cost. Experiments on different motion datasets demonstrated its real-time performance and scalability on particle and view number. For future work, we plan to apply P-APF to other high dimensional tracking problems.

References

[1] A. Balan, L. Sigal, and M. Black. A quantitative evaluation of video-based 3d person tracking. *VS-PETS*, pages 349–356, 2005.

[2] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. *CVPR*, 2:126–133, 2000.

[3] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *IJCV*, 61(2):185–205, 2005.

[4] B. Gaster, L. Howes, D. R. Kaeli, P. Mistry, and D. Schaa. *Heterogeneous Computing with OpenCL*. Elsevier Science, Burlington, 2011.

[5] A. Munshi. OpenCL Specification, Version 1.1, 2010.

[6] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard. Tracking loose-limbed people. *CVPR*, 1:421–428, 2004.

[7] J. Yan, Y. Li, E. Zheng, and Y. Liu. An accelerated human motion tracking system based on voxel reconstruction under complex environments. *ACCV*, pages 313–324, 2010.

[8] J. Yan, J. Song, and Y. Liu. Simultaneous 3-D human-motion tracking and voxel reconstruction. *OE*, 49(9):097201+, 2010.

[9] J. Yan, J. Song, L. Wang, and Y. Liu. Model-based 3d human motion tracking and voxel reconstruction from sparse views. *ICIP*, pages 3265–3268, 2010.