Real-Time Particle Filtering with Heuristics for 3D Motion Capture by Monocular Vision

David Antonio Gómez Jáuregui, Patrick Horain, Manoj Kumar Rajagopal, Senanayak Sesh Kumar Karri

Institut TELECOM, TELECOM SudParis
9 rue Charles Fourier, 91011 Evry Cedex, France
David.Gomez@Telecom-SudParis.eu
Patrick.Horain@Telecom-SudParis.eu
Manoj_Kumar.Rajagopal@Telecom-SudParis.eu
Senanayak.Karri@Telecom-SudParis.eu

Abstract—**Particle filtering is known as a robust approach for motion tracking by vision, at the cost of heavy computation in the high dimensional pose space. In this work, we describe a number of heuristics that we demonstrate to jointly improve robustness and real-time for motion capture. 3D human motion capture by monocular vision without markers can be achieved in real-time by registering a 3D articulated model on a video. First, we search the high-dimensional space of 3D poses by generating new hypotheses (or particles) with equivalent 2D projection by kinematic flipping. Second, we use a semi-deterministic particle prediction based on local optimization. Third, we deterministically resample the probability distribution for a more efficient selection of particles. Particles (or poses) are evaluated using a match cost function and penalized with a Gaussian probability pose distribution learned off-line. In order to achieve real-time, measurement step is parallelized on GPU using the OpenCL API. We present experimental results demonstrating robust real-time 3D motion capture with a consumer computer and webcam.**

I. Introduction

Interest for human motion capture has been growing in recent years since many target applications have emerged (animation of virtual characters, video surveillance, gesture-based human-computer interaction, interaction with virtual environments, games, etc.). Vision-based motion capture can be a practical solution since it requires only a video stream that can conveniently be captured *e.g.* with a webcam. We focus on 3D human motion capture in real-time without markers. It is still a challenge because of the ambiguities resulting of the lack of depth information, of possible partial occlusion of the body limbs, of the high number of degrees of freedom and of the varying cloth of a person.

3D human motion capture by monocular vision in real-time without using markers has been achieved by optimally registering a 3D articulated body model on a video [1]. This involves searching in a high dimensional space (over 20 degrees of freedom for body poses) where multiple local minimums may occur. Particle filtering algorithm [2], has been widely used for human body tracking. It allows to propagate multiple hypotheses as particles, so possibly dealing with observation ambiguities. However classical particle filter require an impractically large number of particles for sampling the high dimensional state space; a small number of particles would usually imply frequent tracking failures that can hardly be recovered because of sample depletion in the state space.

In this work, we propose a real-time particle filter algorithm for human motion capture. We focus on capturing the motion of the upper part of the human body. First, we search the high-dimensional space of 3D poses by generating new hypotheses (or particles) with equivalent 2D projection by kinematic flipping [3]. Second, we use a deterministic particle prediction based on local optimization. Third, we deterministically resample the probability distribution for a more efficient selection of particles. The pose distribution is learned off-line from some motion corpus, and then used to penalize unlikely 3D poses (particles). Finally, real-time computation with a high number of particles is achieved with a parallel implementation on GPU of the particle evaluation.

This paper is structured as follows. In the next section, we discuss the state-of-the-art for motion capture by computer vision with focus on approaches based on particle filtering. In section III, we address particle filter acceleration on GPU. In section IV, we describe our method with particle filtering for real-time 3D motion capture from monocular video sequences and our GPU implementation to parallelize particles measurement step (section VI). Then, in section VII, we present experimental results. Finally, we conclude in section VIII.

II. Previous work on 3D human motion capture

Many methods for human motion capture from monocular images have been proposed in the literature [4]. Model-based approaches use a 3D body model to find the 3D pose that best matches the image descriptors [1] (*e.g.* color regions, edges, silhouettes). Instead, model-free approaches learn a function that maps the image features directly to the 3D pose space [5].

Model-based approaches have been widely used for 3D human motion capture. Here, the body pose is estimated by maximizing some matching criterion between the model and

MMSP'10, October 4-6, 2010, Saint-Malo, France.
978-1-4244-8112-5/10/$26.00 ©2010 IEEE.

the input image [1] or by detecting body-parts and then assembling them into the full pose with limb proximity constraints [6]. Robustly tracking the human pose over frames has been addressed in several works by modifying the original particle filter algorithm [2]. As one of the first such contributions, Deutscher *et al.* [7] proposed to reduce the number of particles using a multi-layered search to migrate gradually the particle set toward the global maximum, at the cost of multiple iterations per frame. Sminchisescu *et al.* [8] proposed an efficient search technique that inflates each sample posterior covariance along uncertainty directions (covariance sampling) and refine each sample using continuous optimization. However, most of these approaches involve a high-computational cost making it inappropriate for real-time.

Some other papers describe quasi real-time (2-6 Hz) by partitioning the high-dimensional pose space into several subspaces to alleviate the computation complexity. Gang Hua *et al.* [9] represent each individual limb with its own motion and reinforce their spatial coherence with a Markov network. Motion posteriors of each body part are approximated by Bayesian inference and a set of low dimensional particle filters for each body part interact and collaborate with each another. Noriega *et al.* [6] proposed independent particle filtering and likelihood evaluation for each limb, while taking into account interactions between limbs through belief propagation. Although these approaches allow reducing the computational cost, the self-occlusion may be a problem since unobserved body parts may give rise to motion mistrackings [4].

III. Previous work on GPU particle filtering approaches

Modern consumer personal computers contain extremely powerful graphics processing units (GPU). These GPUs are designed to perform a limited number of operations on very large amounts of data in parallel. In order to achieve real-time in particle filter approaches, some works have taken advantage of the growing computational power of graphic cards (GPU). Lenz *et al.* [10] proposed a GPU-accelerated particle filter algorithm to track skin-colored objects in real-time. They implemented the high computation steps (segmentation, hypotheses computation and likelihood evaluation) on graphics hardware, through OpenGL shader language, while minimizing the data transfer between CPU and GPU. Montemayor *et al.* [11] alleviated the computational cost of a 2D object tracking particle filter system by parallelizing the weight computation step. They exploited GPU higher memory bandwidth texture by creating a large square texture composed of a collection of small textured quads. Each quad contained a sub-image of a possible state of a system (particle) and is compared to the object model texture in parallel with the use of a fragment program. Lozano *et al.* [12] also parallelized the weight computation step as [11] in a face tracking particle filter approach, they take advantage of CUDA multiprocessor architecture to process in just one kernel, the full weighting operation. The kernel is invoked and executed in M blocks, each with N threads (one block per particle and one thread per feature point per block). Each thread from each block computes the matching error of a particle/feature pair and place the result in a different position of the blocks shared memory. Threads are synchronized and the particle weight is placed in global device memory. Hendeby *et al.* [13] proposed the first complete parallel particle filter on a GPU. Every step of the algorithm (measurement, resample and time update) is implemented on GPU fragment shaders outperforming the computation speed of a CPU implementation when number of particles is large (above 10^3 particles).

IV. Particle Filtering for 3D Human Motion Capture

We consider 3D motion capture without markers by registering a 3D articulated model of the upper human body on monocular video sequences. Registration consists in searching for the best match between primitives extracted from the 3D model and the captured image with respect to the model position and joint angles. This registration process can be achieved iteratively by some local optimization method (*e.g.* Downhill Simplex [14]) under biomechanical constraints [15]. Unfortunately, cluttered observations and ambiguities in monocular images may trap the registration process into some incorrect local minima. More robust pose tracking can be achieved by propagating multiple hypotheses (samples) over time and estimating the probability funtion over the state space.

Human motion cannot be predicted in the general case. Therefore, the prediction step in classical particle filtering relies on stochastic diffusion, at the cost of a large number of particles and heavy computation. Instead, we propose to reduce computation by heuristically guiding particles towards the local optimums. Rather that randomly resampling the posterior density, we select the best particles and we generate further particles for states with similar (ambiguous) appearance by kinematic flipping [3]. In the following subsections we respectively present the standard particle filter approach and our proposed modifications to each step of the standard particle filter that combine deterministic and random sampling and computes analytically alternative samples with the same model projection (kinematic jumps) [3].

A. Standard Particle Filter Algorithm (PFS)

Particle filtering is a Sequential Monte Carlo (SMC) method that recursively approximates the posterior density $p(X_t|Z_t)$ over the state x_t of a dynamical system using a large set of random candidate poses, named particles. These particles are propagated over time using *Importance Sampling* mechanism [2]. Particle filter is based on the assumption that the posterior $p(X_t|Z_t)$ can be represented by sets S_t of weighted particles $\langle x_t^{(i)}, w_t^{(i)} \rangle$. Each particle $x_t^{(i)}$ represents a state, and each $w_t^{(i)}$ is a normalized numerical factor called importance weights which is an observation likelihood $w = p(z_t|x_t)$, where z_t is an observation measurement. The algorithm consists of 3 steps:

- **Selection**. Select a new particle set $\{\tilde{x}_t^{(i)}\}_{i=1}^N$ with N particles from the weighted set $\{\langle x_{t-1}^{(i)}, w_{t-1}^{(i)}\rangle\}_{i=1}^N$. The probability of selecting a particle $x_{t-1}^{(i)}$ is proportional to its weight $w_{t-1}^{(i)}$.
- **Prediction**. Generate $\{x_t^{(i)}\}_{i=1}^N$ from the particle set $\{\tilde{x}_{t-1}^{(i)}\}_{i=1}^N$ according to a stochastic diffusion model $x_t^{(i)} = \tilde{x}_{t-1}^{(i)} + \eta$, where η is a vector of standard normal random variables. In this way, the new set $\{x_t^{(i)}\}_{i=1}^N$ accounts for the uncertainty in the behavior of the tracked object.
- **Measurement**. Evaluate the new particles $\{x_t^{(i)}\}_{i=1}^N$ based on observations $w_t^{(i)} \propto p(z_t|x_t^{(i)})$. Then, normalize the weighted particle set $\langle x_t^{(i)}, w_t^{(i)}\rangle$ so that $\Sigma_{i=1}^N w_t^{(i)} = 1$ and compute the cumulative probabilities. The resulted weighted particles represents the new posterior density of the system current state after measurement.

Each time that N samples have been obtained, the current state can be estimated by computing the mean state of $\{x_t^{(i)}\}_{i=1}^N$ or by selecting the particle with the maximum weight.

B. Selection by Parent-Children Heuristic (HSPC)

In the standard algorithm, this step would allow low-weight particles to be selected, although at a low probability. In order to save particles and computation, we heuristically enforce each parent particles to give birth to number of children (resampled) particles proportional to their weights, based on a proportional representation electoral scheme. Namely, each parent particle $x_{t-1}^{(i)}$ gives rise to $n_t^{(i)} = N * (w_{t-1}^{(i)}/\Sigma_{i=1}^N w_{t-1}^{(i)})$ children particles $\{\tilde{x}_t^{(i)}\}_{i=1}^{n_t}$, where N is the preset total number of particles, $w_{t-1}^{(i)}$ is the weight of each particle. In this way, some low weight particles may have no child, while particle with heavy weight give birth to a family of particle. The group of children generated from a parent particle are its exact copy to generate a new set $\{\tilde{x}_t^{(i)}\}_{i=1}^N$.

C. Prediction heuristics

We propose two methods to deterministically generate new hypotheses. In the first method, particles are heuristically guided towards optimums with limited local optimization. The use of optimization within the Particle Filtering is by now a relatively standard approach (e.g., Wang *et al.* [16], and Choo *et al.* [17]). In the second method, we compute analytically 3D poses that correspond to alternative local minimums associated with the same set of projected joint centers [3].

1) Prediction with Optimization and Random Variability (HPOR): We use limited local optimization in the particle filter prediction step to guide the particles towards local minimums (peaks of the posterior) of a matching cost function (e.g. equation 1 from section IV-D) in the state space. We used the downhill simplex optimization algorithm that requires only function evaluations, and not calculation of derivatives. In the N_D-dimensional pose space (20 parameters in our case), we construct an initial simplex with N_D + 1 vertices. The algorithm iteratively approaches a local optimum by four operations: reflection, expansion, one-dimensional contraction, and multiple contraction [14]. From the selection step proposed in sub-section IV-B, if the number of children $n_t^{(i)}$ from a parent particle $x_{t-1}^{(i)}$ is larger that N_D + 1, we build a new simplex where each vertex corresponds to a new particle state $x_t^{(i)}$. We make N_I = $n_t^{(i)}$ - N_D iterations (reflection, expansion, contraction) in order to guide these particles toward local optimums. However, because of the highly unpredictable human motion, it is important to conserve the random variability in the prediction. Thus, if the number of children $n_t^{(i)}$ is smaller that N_D + 1, we diffuse these children $x_t^{(i)}$ stochastically using random Gaussian noise.

2) Kinematic Flipping based Sampling (HKFS): Multiple 3D model poses may give the same ambiguous 2D image projection by forward/backward flipping of articulation joints. Unfortunately, in monocular images, we do not have enough information to disambiguate such poses. Selecting a wrong pose in the registration process leads to mistrackings. We have implemented the approach proposed in [3] to compute analytically new alternative poses that are consistent with the current image projection. This approach allows to find rapidly new "good" local minimums without searching exhaustively in the high-dimensional space.

We build kinematic trees for the kinematic sub-chains of the limb to be tracked (upper-limb in our case). An interpretation tree is generated by traversing each segment from the limb attachment to its end (e.g. from the shoulder joint to the wrist). For each upper-limb segment (arm and forearm), we construct an imaginary 3D sphere centered in the parent joint position (e.g. shoulder joint) with radius equal to the length of the segment limb. The imaginary 3D sphere is traversed at 2 points by the camera ray of sight so giving rise to one alternative 3D position for the joint with same image projection. Once the kinematic tree has been constructed, we calculate the joint angles that rotate each arm segment in order to reach each alternative 3D joint position computed in the last step. The arm kinematic sub-chains consists in the 3 DOF for the shoulder ($R_z^s R_x^s R_y^s$) and 1 DOF for the elbow (R_x^e). The joint angles are calculated analytically by representing each segment rotation by an axis-angle (or rotation vector). From this representation, we compute the Euler joint angles (through a rotation matrix) using the method proposed in [18]. We assign each sub-chain arm configuration to a different pose. Then, we combine the arm kinematic sub-chains in order to generate 15 new hypotheses (particles $x_t^{(i)}$) for each set of projected joints. However, only some particles can be permissible due to the biomechanical constraints implemented in our human body model. In our algorithm proposed, we find new kinematic flipping hypotheses for the best particle generated with the local optimization prediction described in IV-C1.

D. Measurement step by matching regions

Our likelihood function consists mainly in a matching correspondence measure between features extracted from the captured image and the 3D model. First, we detect the human silhouette using a robust background subtraction algorithm that compares different features (gradient orientation and chrominance correlation of YUV) of each captured image with the same features of a reference image of the background. The human silhouette (foreground) is segmented in three class of regions (face, arms and clothes). Face is detected with Adaboost face detector [19]. Then, a skin color sample (for arms detection) is taken from the face region and a clothes sample is taken under the face. We model each sample using a simple gaussian model in a HSV color space. For each image, we project the 3D model according to the pose described in the vector of parameters. The 3D model is projected by rendering the head, arms and clothes regions. The matching between the 3D model projection and the segmented image is evaluated using a non-overlapping ratio [15]:

$$F(q)=\prod_{c=1}^{m}\left(\frac{|A_c\cup B_c(q)|-|A_c\cap B_c(q)|}{|A_c\cup B_c(q)|}\right)^{\frac{1}{m}}\quad. \tag{1}$$

where q is the vector of parameters describing a candidate 3D pose, *m* is the number of classes, A_c is the set of pixels with the *c* region class in the segmented image, $B_c(q)$ is the set of pixels with the *c* region class in the projection of the 3D model and $|X|$ represent the number of pixels in *X*. The non-overlapping ratio is computed by adding the segmented image to the 3D model projection thus obtaining a third image. Then, the histogram of this third image is calculated in order to obtain the number of pixels that belong to the union and intersection of each region class. The weight of each particle is defined by the following region-based likelihood function:

$$p(z_t|x_t^{(i)})\propto \exp\left(-\frac{F(q)^2}{2\sigma_F^2}\right). \tag{2}$$

where *F(q)* is the non-overlapping ratio function as described by equation (1), σ_F^2 is the range of standard deviation (obtained experimentally) that express the variability of the non-overlapping ratio on a full video sequence.

Integrating Gaussian prior pose probability: We have learnt a prior Gaussian model of the probability of poses of some motion databases [20], [21]. We reduce the parameter space dimensionality through Principal Component Analysys (PCA). Here, dimensionality reduction is meant to ignore those pose features that may be captured but where not in the learning corpuses. So, pose probability is evaluated only with respect to those main features selected with PCA. We found that more that 95% of the variability could be captured with 2 eigen vectors itself. This probability is used as a penalty for particles in order to avoid less likely poses.

V. The steps of our Real-Time Particle Filter with Heuristics Algorithm

Input: The set of particles (poses) $\{\langle x_{t-1}^{(i)}, w_{t-1}^{(i)}\rangle\}_{i=1}^{N}$ that approximates the previous posterior $p(X_{t-1}|Z_{t-1})$.

1) For each particle x_{t-1}, generate $n_t^{(i)}$ children $\{x_t^{(i)}\}_{i=1}^{n_t}$ (as described in subsection IV-B).
2) For each group of children $\{x_t^{(i)}\}_{i=1}^{n_t}$
   a) If $n_t^{(i)} > N - dimensions$ + 1: Build a new simplex and iterate to guide the particles toward local optimums (as described in subsection IV-C1). Replace the poses resulted from each iteration in every child of the group. Then, find new Kinematic Flipping poses (as described in subsection IV-C2) for the best child resulted from the local optimisation.
   b) If $n_t^{(i)} \leq N - dimensions$ + 1: Diffuse each child of this group with random gaussian noise.
3) Evaluate all particles $\{x_t^{(i)}\}_{i=1}^{N}$ in parallel (section VI) and assign the weights $\{w_t^{(i)}\}_{i=1}^{N}$. Penalize each weight with prior Gaussian probability (μ). $w_t^{(i)} = w_t^{(i)} * \mu$ (section IV-D).
4) Normalize weights ($\Sigma_{i=1}^{N} w_t^{(i)} = 1$). The set of particles $\{x_t^{(i)}, w_t^{(i)}\rangle\}_{i=1}^{N}$ represent the new posterior density $p(X_t|Z_t)$.

VI. Implementing Particle Filter on GPU

Particles evaluation can benefit from Graphics Processing Units (GPUs) now available in most personal computers. GPUs were originally designed for rendering graphics using a pipeline and then have become programmable for general purpose computing. Furthermore, particles evaluation can be implemented efficiently in a batch using the massively parallel architecture of GPUs. The evaluation of candidate poses (particles) involves projecting the 3D body parts labels on the image plane, which can be done efficiently using OpenGL flat rendering onto a Pixel Buffer Object (PBO) . The segmented input image labels are also mapped into different bit-planes of this image by OpenGL blending. Then we derive the non-overlapping ratio from the histogram of the image thus formed, as described in section IV-D.

The evaluation of candidate poses (particles) involves projecting the 3D body parts labels on the image plane, which can be done efficiently using OpenGL flat rendering onto a Pixel Buffer Object (PBO) . The segmented input image labels are also mapped into different bit-planes of this image by OpenGL blending. Then we derive the non-overlapping ratio from the histogram of the image thus formed, as described in section IV-D.

While efficient histogram calculation on GPU has been a challenge with the earlier versions [22], GPUs now offer two useful features that make it easier. Scatter allows a computation thread to write to any location in the output array in parallel to the other threads, while atomic writes use locks to control access to while writing it [23]. Furthermore, the recent OpenCL API can leverage the power of GPUs and multicore CPUs and has complete access to the rendered pixel buffer [24].

GPUs are massively parallel devices that are the more efficient when the number of threads invoked is huge, so a single particle evaluation cannot take advantage of the GPU parallel architectures. Therefore, we concatenated all the image buffers

TABLE I

Computation time with respect to the number of particle evaluations (histogram computation). Image size: 160 x 120

Number of Particles	**CPU (Native C++)**	**CPU (OpenCL)**	**GPU-1 (OpenCL)**	**GPU-2 (OpenCL)**
100	27.14 ms	12.24 ms	1.50 ms	1.37 ms
300	40.31 ms	19.75 ms	2.73 ms	2.50 ms
500	75.91 ms	28.32 ms	6.85 ms	6.53 ms
700	110.34 ms	50.42 ms	20.84 ms	16.84 ms

formed by adding the rendered buffer and the segmented buffer to calculate the corresponding histograms in parallel. Our implementation involves two levels of parallelism, the one on the images associated with the particles, the other on the pixels of each of these images. The speed up achieved by using an Nvidia GPU of the Tesla generation is presented in Table I [1].

VII. Performance experiments

In order to analyze the robustness of our particle filter approach, we considered the residual values of each evaluation function (equation 1) and the discrepancy between the estimated joint angle parameters and the ground truth (provided by the motion databases [21], [20]). In each experiment, we vary the number of particle evaluations that can be processed in real-time on recent consumer computers (Table I). Our goal is to reproduce human gestures with a 3D avatar. Therefore, real-time is established at 10 Hz or more in order to have a visually smooth animation. We used video sequences from CMU Motion Capture database [21] and GRETA motion database [20] since ground truth (joint rotation angles) is provided with each video sequence. In all sequences, we drop the frame size to 160 x 120 in order to process less number of pixels.

A. Our 3D human body model

We use a human upper-body model defined as a polygon mesh and rendered with OpenGL. It has 3 global position parameters and 20 joint angles (chest, neck, arms, forearms and hands). Each particle or state $x_t^{(i)}$ is a different human pose represented by a vector of 20 joint angle parameters. Registering the 3D upper-body human model on each video sequence requires calibrating the body model to make it similar to the actor captured in the video. This is done by adjusting the 20 joint angles of the 3D model taking as reference the human pose in the first image and the dimensions (length, width and height) of each body part of the 3D human model [1].

B. Tracking results

We computed, for each video sequence, the mean residual value (non-overlapping ratio) and the root mean square error with respect to the ground truth. Lower mean residual values and lower root mean square error represent visually a more robust (less number of mistrackings) and accurate motion tracking. We evaluated the performance of each heuristic separately by using only one heuristic at a time within the Standard Particle Filter algorithm (PFS). In Table II and Table III, we show the results obtained from a video sequence from GRETA database [20]. We appreciate the robustness (Table II) and accuracy (Table III) contribution obtained by each heuristic proposed and our Real-Time Particle Filter with Heuristics algorithm (PFRTH) that integrates all the heuristics proposed (algorithm described in section V).

TABLE II

Mean residual values (non-overlapping ratio) obtained by each heuristic proposed with respect to the number of particles. (Acronyms were introduced in section: IV)

Heuristic	**100 P**	**300 P**	**500 P**	**700 P**	**2000 P**
PFS	2.39	2.34	2.32	2.31	2.28
HSPC	2.32	2.29	2.27	2.26	2.25
HPOR	2.32	2.26	2.24	2.23	2.21
HKFS	2.37	2.32	2.31	2.30	2.27
PFRTH	2.31	2.25	2.23	2.22	2.19

TABLE III

Root Mean Square Error for the Right elbow angle obtained by each heuristic proposed with respect to the number of particles. (Acronyms were introduced in section: IV).

Heuristic	**100 P**	**300 P**	**500 P**	**700 P**	**2000 P**
PFS	29.6	23.5	21.3	21.1	20.8
HSPC	29.6	22.1	20.4	20.3	20.0
HPOR	30.1	24.3	23.8	22.2	21.4
HKFS	29.0	21.8	21.0	20.1	19.5
PFRTH	24.4	21.4	20.2	19.3	18.1

From Table II, we can observe that effectively HSPC improves robustness (lower mean residual values) with respect to the PFS algorithm since it does not allow low-weight particles to be propagated at the cost of reducing the accuracy of the 3D poses (larger root mean square error values in Table III) since it removes the random variability of the original algorithm (PFS). HPOR is the heuristic that provides the highest robustness (Table II) but the greatest lost of accuracy (largest root mean square error values in Table III). The reason is that HPOR moves blindly the particles toward different peaks (local minimums) of the posterior probability. Therefore, HPOR can guide the particles toward incorrect peaks since the same observation can give more that one pose solution in monocular images. HKFS finds analytically new peaks of high probability (poses that belong to the same observation) improving the accuracy of the motion tracking in monocular images (lowest root mean square error values in Table III). However, HKFS does not lead correctly the particles inside the peaks of high probability causing a lower robustness that HPOR heuristic. In the proposed algorithm (PFRTH), we combined the advantages of each heuristic proposed. Thus, a smaller number of particles is needed in order to achieve a more robust and accurate motion tracking (Table II and Table III). Figure 1 presents some qualitative (visual) results of our

[1] Experiments were run on an Intel Core 2 Quad CPU Q9000 @ 2.0 Ghz, a GPU Nvidia GTX 280M with 240 CUDA cores @ 1296Mhz (GPU-1) and a GPU ATIRadeon HD 5870 with 800 Stream Processors @ 850Mhz (GPU-2) .

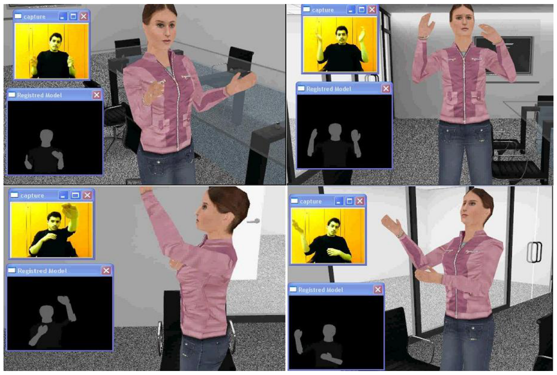

Fig. 1. 3D human motion capture by real-time monocular vision to animate a 3D avatar in a collaborative virtual environment. General human motion (including occlusions and relative motion in depth) is acquired with our Real-Time Particle Filter with Heuristics method proposed (700 particles were used). Joint angle pose parameters (in MPEG-4 BAP format) are sent to the 3D avatar in real-time.

3D motion tracking by monocular (with a webcam) vision in real-time. Particles were evaluated in parallel using the GPU implementation described in section VI.

VIII. Conclusions

We have presented a real-time particle filter algorithm for 3D human motion capture by monocular vision. Our method consists in a Particle Filter approach where particles are directed towards the optimal pose by heuristically integrating a fast local optimization algorithm. 3D ambiguities in monocular images are handled with kinematic flipping by adding particles later discriminated when tracking. Particles to be propagated in time are selected deterministically based on a proportional electoral scheme. Evaluation of particles is made with a function likelihood that consists in a region matching (face, arms and clothes) between the model projection and captured image. The function likelihood is penalized by integrating a Gaussian pose probability learned off-line from a motion database. In order to achieve real-time computation, evaluation of particles is accelerated by parallel GPU computing.

Our experiments on different video sequences showed that search in high-dimensional space was improved achieving a higher robustness and accuracy in monocular 3D motion tracking with relatively small number of particles. Our future works aims at reducing the high-dimensionality of the 3D pose space into a low-dimensional latent space in order to use a smaller number of particles to achieve a more robust tracking.

Acknowledgment

We thank Professor C. Pelachaud for kindly providing the GRETA Embodied Conversational Agent Application and the corpus of communication gestures with variants [20].

We thank I-Maginer for their networked 3D platform.

References

[1] D. A. G. Jáuregui and P. Horain, "Region-based vs. edge-based registration for 3d motion capture by real time monoscopic vision," in *Proceedings of MIRAGE 2009*, INRIA Rocquencourt, France, A. Gagalowicz and W. Philips (Eds.), LNCS 5496, 2009, pp. 344–355.

[2] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *IJCV : International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[3] C. Sminchisescu and B. Triggs, "Kinematic jump processes for monocular 3d human tracking," in *In International Conference on Computer Vision and Pattern Recognition*, Madison, WI, 2003, pp. 69–76.

[4] R. W. Poppe, "Vision-based human motion analysis: An overview," in *Computer Vision and Image Understanding*, vol. 108, 2007, pp. 4–18.

[5] A. Agarwal and B. Triggs, "Recovering 3d human pose from monocular image," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, 2006, pp. 44–58.

[6] P. Noriega and O. Bernier, "Multicues 3d monocular upper body tracking using constrained belief propagation," in *British Machine Vision Conference*, vol. 2, Warwick, United Kingdom, September 2007, pp. 10–13.

[7] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 126–133.

[8] C. Sminchisescu and B. Triggs, "Covariance scaled sampling for monocular 3d body tracking," in *In International Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.

[9] G. Hua and Y. Wu, "A decentralized probabilistic approach to articulated body tracking," *Journal of Computer Vision and Image Understanding*, vol. 108, no. 3, pp. 272–283, 2007.

[10] C. Lenz, G. Panin, and A. Knoll, "A gpu-accelerated particle filter with pixel-level likelihood," in *In International Workshop on Vision, Modeling and Visualization (VMV)*, Konstanz, Germany, October 2008.

[11] A. S. Montemayor, J. J. Pantrigo, R. Cabido, and B. Payne, "Bandwidth-improved gpu particle filter for visual tracking," in *In proceedings of the Ibero-American Symposium on Computer Graphics - SIACG (2006)*, Santiago de Compostela, Spain, 2006.

[12] O. M. Lozano and K. Otsuka, "Real-time visual tracker by stream processing," *Journal of Signal Processing Systems*, vol. 57, no. 2, pp. 285–295, November 2009.

[13] G. Hendeby, J. D. Hol, R. Karlsson, and F. Gustafsson, "A graphics processing unit implementation of the particle filter," in *Proceedings of European Signal Processing Conference*, Poznan, Poland, September 2007.

[14] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 208–313, 1965.

[15] J. M. Soares, P. Horain, A. Bideau, and M. Nguyen, "Acquisition 3d du geste par vision monoscopique en temps réel et téléprésence," in *Actes de l'atelier Acquisition du geste humain par vision artificielle et applications*, Toulouse, 2004, pp. 23–27.

[16] P. Wang and J. M. Rehg, "A modular approach to the analysis and evaluation of particle filters for figure tracking," in *In Conference on Computer Vision and Pattern Recognition*, vol. 1, New York, June 2006, pp. 790–797.

[17] K. Choo and D. J. Fleet, "People tracking using hybrid monte carlo filtering," in *In Proc. International Conference of Computer Vision*, vol. 2, 2001, pp. 321–328.

[18] K. Shoemake, *Graphics Gems IV*, P. Heckbert, Ed. Academic Press, 1994.

[19] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. 511–518.

[20] Z. Li, P. Horain, A. M. Pez, and C. Pelachaud, "Statistical gesture models for 3d motion capture from a library of gestures with variants," in *in Post-proceedings of the International Gesture Workshop (GW2009 external)*, vol. 5934. Bielefeld University: LNAI series, Springer-Verlag, February 2009.

[21] CMU graphics lab motion capture database. [Online]. Available: http://mocap.cs.cmu.edu

[22] T. Scheuermann and J. Hensley, "Efficient histogram generation using scattering on gpus," in *SI3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2007, pp. 33–37.

[23] M. Okun and A. Barak, "Atomic writes for data integrity and consistency in shared storage devices for clusters," *Future Gener. Comput. Syst.*, vol. 20, no. 4, pp. 539–547, 2004.

[24] Opencl - the open standard for parallel programming of heterogeneous systems. [Online]. Available: http://www.khronos.org/opencl/