Using Gaussian Process Annealing Particle Filter for 3D Human Tracking

Leonid Raskin, Ehud Rivlin, Michael Rudzsky

Computer Science Department,Technion Israel Institute of Technology,
Technion City, Haifa, Israel
{*raskinl*, *ehudr*, *rudzsky*} @cs.technion.ac.il

Abstract. We present an approach for human body parts tracking in 3D with prelearned motion models using multiple cameras. Gaussian Process Annealing Particle Filter is proposed for tracking in order to reduce the dimensionality of the problem and to increase the tracker's stability and robustness. Comparing with a regular annealed particle filter based tracker, we show that our algorithm can track better for low frame rate videos. We also show that our algorithm is capable of recovering after a temporal target loose.

1 Introduction

Human body pose estimation and tracking is a challenging task for several reasons. First, the large dimensionality of the human 3D model complicates the examination of the entire subject and makes it harder to detect each body part separately. Secondly, the significantly different appearance of different people that stems from various clothing styles and illumination variations, adds to the already great variety of images of different individuals. Finally, the most challenging difficulty that has to be solved in order to achieve satisfactory results of pose understanding is the ambiguity caused by body.

This paper presents an approach to 3D articulated human body tracking, that enables reduction of the complexity of this model. We propose a novel algorithm, Gaussian Process Annealed Particle Filter (GPAPF) (see also Raskin et al. [11, 26]). In this algorithm we apply a nonlinear dimensionality reduction using Gaussian Process Dynamical Model (GPDM) (Lawrence [7], Wang et al. [19]) in order to create a low dimensional latent space. This space describes poses from a specific motion type. Later we use annealed particle filter proposed by Deutscher and Reid [3, 15] that operates in this laten space in order to generate particles.

The annealed particle filter has a good performance when applied on videos with a high frame rate (60 fps, as reported by Balan et al. [5]), but performance drops when the frame rate is lower (30fps). We show that our approach provides good results even for the low frame rate (30 fps and lower). An additional advantage of our tracking algorithm is the capability to recover after temporal loss of the target, which makes the tracker more robust.

2 Related Works

There are two main approaches for body pose estimation. The first one is the body detection and recognition, which is based one a single frame (Perona et al. [33], Ioffe et al. [30], Mori et al. [12]). The second approach is the body pose tracking which approximates body pose based on a sequence of frames (Sidenbladh et al. [27], Davidson et al. [14], Agarwal et al. [2, 1]). A variety of methods have been developed for tracking people from single views (Ramanan et al. [9]), as well as from multiple views (Deutscher et al. [3]).

One of the common approaches for tracking is using particle filtering methods. Particle Filtering uses multiple predictions, obtained by drawing samples of pose and location prior and then propagating them using the dynamic model, which are refined by comparing them with the local image data, calculating the likelihood (see, for example Isard and MacCormick [24] or Bregler and Malik [6]). The prior is typically quite diffused (because motion can be fast) but the likelihood function may be very peaky, containing multiple local maxima which are hard to account for in detail. For example, if an arm swings past an arm-like pole, the correct local maximum must be found to prevent the track from drifting (Sidenbladh et al. [17]). Annealed particle filter (Deutscher et al. [15]) or local searches are the ways to attack this difficulty. An alternative is to apply a strong model of dynamics (Mikolajcyk et al. [20]).

There exist several possible strategies for reducing the dimensionality of the configuration space. Firstly it is possible to restrict the range of movement of the subject. This approach has been pursued by Rohr [21]. The assumption is that the subject is performing a specific action. Agarwal and Triggs [2, 1] assume a constant angle of view of the subject. Because of the restricting assumptions the resulting trackers are not capable of tracking general human poses. Several works have been done in attempt to learn subspace models. For example, Ormoneit et al. [25] has used PCA on the cyclic motions. Another way to cope with high-dimensional data space is to learn low-dimensional latent variable models [4, 13]. However, methods like Isomap [10] and locally linear embedding (LLE) [31] do not provide a mapping between the latent space and the data space. Urtasun et al. [29, 28, 18] uses a form of probabilistic dimensionality reduction by Gaussian Process Dynamical Model (GPDM) (Lawrence [7], and Wang et al. [19]) and formulate the tracking as a nonlinear least-squares optimization problem.

We propose a tracking algorithm, which consists of two stages. We separate the body model state into two independent parts: the first one contains information about 3D location and orientation of the body and the second one describes the pose. We learn latent space that describes poses only. In the first one we generate particles in the latent space and transform them into the data space by using learned a priori mapping function. In the second stage we add rotation and translation parameters to obtain valid poses. Then we project the poses on the cameras in order to calculate the weighted function.

The article is organized as follows. In Section 3 and Section 4 we give a description of particle filtering and Gaussian fields. In Section 5, we describe our algorithm. Section 6 contains our experimental results and comparison to

annealed particle filter tracker. The conclusions and possible extension are given in Section 7.

3 Filtering

3.1 Particle filter

The particle filter algorithm was developed for tracking objects, using the Bayesian inference framework. In order to make an estimation of the tracked object parameter this algorithm suggests using the importance sampling. Importance sampling is a general technique for estimating the statistics of a random variable. The estimation is based on samples of this random variable generated from other distribution, called proposal distribution, which is easy to sample from.

Let us denote x_n as a hidden state vector and y_n be a measurement in time n. The algorithm builds an approximation of a maximum posterior estimate of the filtering distribution: $p(x_n|y_{1:n})$, where $y_{1:n} \equiv (y_1, ..., y_n)$ is the history of the observation. This distribution is represented by a set of pairs $\left\{x_n^{(i)}; \pi_n^{(i)}\right\}_{i=1}^{N_p}$, where $\pi_n^{(i)} \propto p\left(y_n|x_n^{(i)}\right)$. Using Bayes' rule, the filtering distribution can be calculated using two steps: Prediction step:

$$p(x_n|y_{1:n-1}) = \int p(x_n|x_{n-1})\, p(x_{n-1}|y_{1:n-1})\, dx_{n-1} \tag{1}$$

Filtering step:

$$p(x_n|y_{1:n}) \propto p(y_n|x_n)\, p(x_n|y_{1:n-1}) \tag{2}$$

Therefore, starting with a weighted set of samples $\left\{x_0^{(i)}; \pi_0^{(i)}\right\}_{i=1}^{N_p}$, the new sample set $\left\{x_n^{(i)}; \pi_n^{(i)}\right\}_{i=1}^{N_p}$ is generated according to the distribution, that may depend on the previous set $\left\{x_{n-1}^{(i)}; \pi_{n-1}^{(i)}\right\}_{i=1}^{N_p}$ and the new measurements y_n: $x_n^{(i)} \sim q\left(x_n^{(i)}|x_{n-1}^{(i)}, y_n\right), i = 1, ..., N_p$. The new weights are calculated using the following formula:

$$\pi_n^{(i)} = k\pi_n^{(i)} \frac{p\left(y_n|x_n^{(i)}\right) p\left(x_n^{(i)}|x_{n-1}^{(i)}\right)}{q\left(x_n^{(i)}|x_{n-1}^{(i)}, y_n\right)} \tag{3}$$

where

$$k = \left(\sum_{i=1}^{N_p} \pi_n^{(i)} \frac{p\left(y_n|x_n^{(i)}\right) p\left(x_n^{(i)}|x_{n-1}^{(i)}\right)}{q\left(x_n^{(i)}|x_{n-1}^{(i)}, y_n\right)}\right)^{-1} \tag{4}$$

and $q\left(x_n^{(i)}|x_{n-1}^{(i)}, y_n\right)$ is the proposal distribution.

The main problem is that the distribution $p(y_n|x_n)$ may be very peaky and far from being convex. For such $p(y_n|x_n)$ the algorithm usually detects several local maxima instead of choosing the global one (see Deutscher and Reid [15]). This usually happens for the high dimensional problems, like body part tracking. In this case a large number of samples have to be taken in order to find the global maxima, instead of choosing a local one. The other problem that arises is that the approximation of the $p(x_n|y_{1:n})$ for high dimensional spaces is a very computationally inefficient and hard task. Often a weighting function $w_n^i(y_n, x)$ can be constructed according to the likelihood function as it is in the condensation algorithm of Isard and Blake [23], such that it provides a good approximation of the $p(y_n|x_n)$, but is also relatively easy to calculate. Therefore, the problem becomes to find configuration x_k that maximizes the weighting function $w_n^i(y_n, x)$.

3.2 Annealed Particle Filter

The main idea is to use a set of weighting functions instead of using a single one. While a single weighting function may contain several local maxima, the weighting function in the set should be smoothed versions of it, and therefore contain a single maximum point, which can be detected using the regular annealed particle filter.

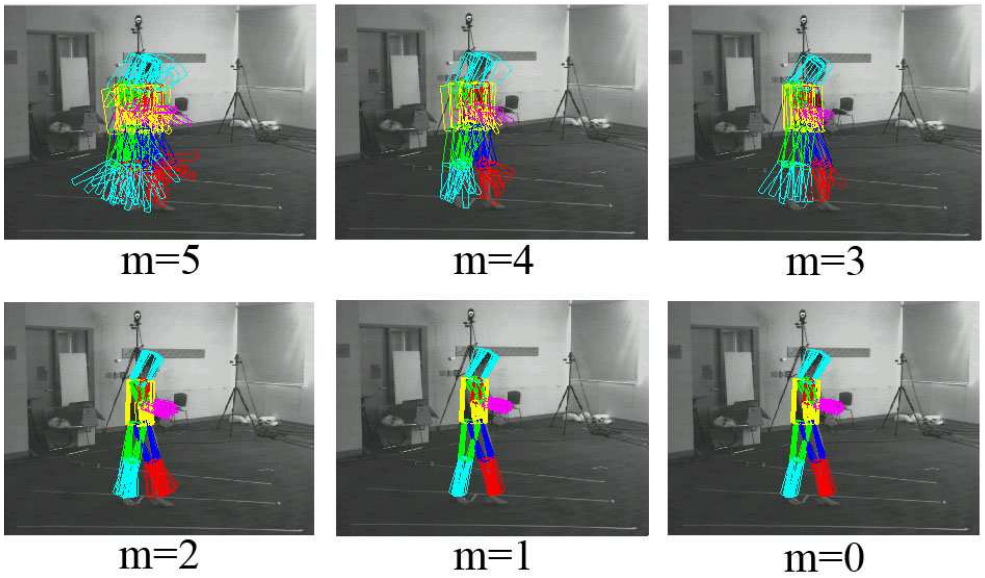

Fig. 1. Annealed particle filter illustration for M=5 . Initially the set contains many particles that represent very different poses and therefore can fall into local maximum. On the last layer all the particles are close to the global maximum, and therefore they represent the correct pose.

A series of $\{w_m(y_n, x)\}_{m=0}^{M}$ is used, where $w_{m-1}(y_n, x)$ differs only slightly from $w_m(y_n, x)$ and represents a smoothed version of it. The samples should be drawn from the $w_0(y_n, x)$ function, which might be peaky, and therefore a large number of particles needed to be used in order to find the global maxima. Therefore, $w_M(y_n, x)$ is designed to be a very smoothed version of $w_0(y_n, x)$. The usual method to achieve this is by using $w_m(y_n, x) = (w_0(y_n, x))^{\beta_m}$, where $1 = \beta_0 > ... > \beta_M$ and $w_0(y_n, x)$ is equal to the original weighting function. Therefore, each iteration of the annealed particle filter algorithm consists of M steps, in each of these the appropriate weighting function is used and a set of pairs is constructed $\left\{x_{n,m}^{(i)}; \pi_{n,m}^{(i)}\right\}_{i=1}^{N_p}$. Tracking is described in Algorithm 1.

Fig. 1 shows the illustration of the 5-layered annealing particle filter. Initially the set contain many particles that represent very different poses and therefore can fall into local maximum. On the last layer all the particles are close to the global maximum, and therefore they represent the correct pose.

Algorithm 1 : The annealed particle filter algorithm

Initialization: $\left\{x_{n,M}^{(i)}; \frac{1}{N}\right\}_{i=1}^{N_p}$

for each: frame n

for $m = M$ downto 0 **do**

1. Calculate the weights: $\pi_n^{(i)} = k\frac{w^m\left(y_n, x_{n,m}^{(i)}\right)p\left(x_{n,m}^{(i)}|x_{n,m-1}^{(i)}\right)}{q\left(x_{n,m}^{(i)}|x_{n,m-1}^{(i)}, y_n\right)}$, where $k = \left(\sum_{i=1}^{N_p}\frac{w^m\left(y_n|x_{n,m}^{(i)}\right)p\left(x_{n,m}^{(i)}|x_{n,m-1}^{(i)}\right)}{q\left(x_{n,m}^{(i)}|x_{n,m-1}^{(i)}, y_n\right)}\right)^{-1}$.

2. Draw N particles from the weighted set $\left\{x_{n,m}^{(i)}; \pi_{n,m}^{(i)}\right\}_{i=1}^{N_p}$ with replacement and with distribution $p\left(x = x_{n,m}^{(i)}\right) = \pi_{n,m}^{(i)}$.

3. Calculate $x_{n,m-1}^{(i)} \sim q\left(x_{n,m-1}^{(i)}|x_{n,m}^{(i)}, y_n\right) = x_{n,m}^{(i)} + n_m$, where n_m is a Gaussian noise $n_m\ N(0, P_m)$.

end for

- The optimal configuration can be calculated using the following formula: $x_n = \sum_{i=1}^{N_p}\pi_{n,0}^{(i)}x_{n,0}^{(i)}$.

- The unweighted particle set for the next observation is produced using $x_{n+1,M}^{(i)} = x_{n,0}^{(i)} + n_0$, where n_0 is a Gaussian noise $n_m\ N(0, P_0)$.

end for each

4 Gaussian Fields

The Gaussian Process Dynamical Model (GPDM) (Lawrence [7], Wang et al. [19]) represents a mapping from the latent space to the data: $y = f(x)$, where $x \in \mathbb{R}^d$ denotes a vector in a d-dimensional latent space and $y \in \mathbb{R}^D$ is a vector, that represents the corresponding data in a D-dimensional space. The

model that is used to derive the GPDM is a mapping with first-order Markov dynamics:

$$\begin{aligned} x_t &= \sum_i a_i \phi_i \left(x_{t-1}\right) + n_{x,t} \\ y_t &= \sum_j b_j \psi_j \left(x_t\right) + n_{y,t} \end{aligned} \tag{5}$$

where $n_{x,t}$ and $n_{y,t}$ are zero-mean Gaussian noise processes, $A = [a_1, a_2, ...]$ and $B = [b_1, b_2, ...]$ are weights and ϕ_j and ψ_j are basis functions.

For Bayesian perspective A and B should be marginalized out through model average with an isotropic Gaussian prior on B in closed form to yield:

$$P\left(Y|X,\overline{\beta}\right) = \frac{|W|^N}{\sqrt{(2\pi)^{ND}}|K_y|^D} e^{-\frac{1}{2}tr\left(K_y^{-1}YW^2Y^T\right)} \tag{6}$$

where W is a scaling diagonal matrix, Y is a matrix of training vectors, X contains corresponding latent vectors and K_y is the kernel matrix:

$$\left(K_y\right)_{i,j} = \beta_1 e^{-\frac{\beta_2}{2}\|x_i - x_j\|} + \frac{\delta_{x_i,x_j}}{\beta_3} \tag{7}$$

W is a scaling diagonal matrix. It is used to account for the different variances in different data elements. The hyper parameter β_1 represents the scale of the output function, β_2 represents the inverse of the Radial Basis Function (RBF) and β_3^{-1} represents the variance of $n_{y,t}$. For the dynamic mapping of the latent coordinates X the joint probability density over the latent coordinate system and the dynamics weights A are formed with an isotropic Gaussian prior over the A, it can be shown (see Wang et al. [19]) that

$$P\left(X|\overline{\alpha}\right) = \frac{P\left(x_1\right)}{\sqrt{(2\pi)^{(N-1)d}}|K_x|^d} e^{-\frac{1}{2}tr\left(K_x^{-1}X_{out}X_{out}^T\right)} \tag{8}$$

where $X_{out} = [x_2, ..., x_N]^T$, K_x is a kernel constructed from $[x_1, ..., x_{N-1}]^T$ and x_1 has an isotropic Gaussian prior. GPDM uses a "linear+RBF" kernel with parameter α_i :

$$\left(K_y\right)_{i,j} = \alpha_1 e^{-\frac{\alpha_2}{2}\|x_i - x_j\|} + \alpha_3 x_i^T x_j + \frac{\delta_{x_i,x_j}}{\alpha_4} \tag{9}$$

Following Wang et al. [19]

$$P\left(X,\overline{\alpha},\overline{\beta}|Y\right) \propto P\left(Y|X,\overline{\beta}\right) P\left(X|\overline{\alpha}\right) P\left(\overline{\alpha}\right) P\left(\overline{\beta}\right) \tag{10}$$

the latent positions and hyper parameters are found by maximizing this distribution or minimizing the negative log posterior:

$$\begin{aligned} \Lambda = \frac{d}{2} ln|K_x| + \frac{1}{2} tr\left(K_x^{-1} X_{out} X_{out}^T\right) + \sum_i ln\alpha_i - Nln|W| + \\ + \frac{D}{2} ln|K_y| + \frac{1}{2} tr\left(K_y^{-1} Y W^2 X^T\right) + \sum_i ln\beta_i \end{aligned} \tag{11}$$

5 GPAPF Filtering

5.1 The Model

In our work we use a model similar to the one proposed by Deutscher et al. [3] with some differences in the annealing schedule and weighting function. The body model is defined by a pair $M = \{L, \Gamma\}$, where L stands for the limbs lengths and Γ for the angles between the limbs and the global location of the body in 3D. The limbs parameters are constant, and represent the actual size of the tracked person. The angles represent the body pose and, therefore, are dynamic. The state is a vector of dimensionality 29: 3 DoF for the global 3D location, 3 DoF for the global rotation, 4 DoF for each leg, 4 DoF for the torso, 4 DoF for each arm and 3 DoF for the head (see Fig. 2). The whole tracking process estimates the angles in such a way that the resulting body pose will match the actual pose. This is done by maximizing the weighting function which is explained next.

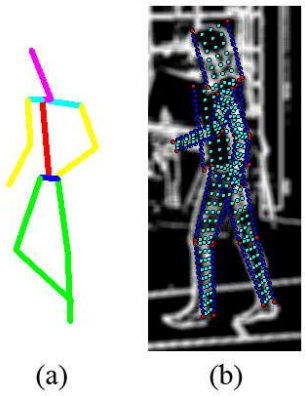

Fig. 2. The 3D body model (a) and the samples drawn for the weighting function calculation (b). On the right image the blue samples are used to evaluate the edge matching, the cyan points are used to calculate the foreground matching, the rectangles with the edges on the red points are used to calculate the part-based body histogram.

5.2 The Weighting Function

In order to evaluate how well the body pose matches the actual pose using the particle filter tracker we have to define a weighting function $w(\Gamma, Z)$, where Γ is the model's configuration (i.e. angles) and Z stands for visual content (the captured images). The weighting function that we use is a version of the one suggested by Deutscher et al. [15] with some modifications. We have experimented with 3 different features: edges, foreground silhouette and foreground histogram.

The first feature is the edge map. As Deutscher et al. [15] proposes this feature is the most important one, and provides a good outline for visible parts,

such as arms and legs. The other important property of this feature is that it is invariant to the color and lighting condition. The edge maps, in which each pixel is assigned a value dependent on its proximity to an edge, are calculated for each image plane. Each part is projected on the image plane and samples of the N_e hypothesized edges of human body model are drawn. A sum-squared difference function is calculated for these samples:

$$\Sigma^e\left(\Gamma, Z\right) = \frac{1}{N_{cv}} \frac{1}{N_e} \sum_{i=1}^{N_{cv}} \sum_{j=1}^{N_e} \left(1 - p_j^e\left(\Gamma, Z_i\right)\right)^2 \tag{12}$$

where N_{cv} is a number of camera views, and Z_i stands for the image from the i-th camera . The $p_j^e\left(\Gamma, Z_i\right)$ are the edge maps. Each part is projected on the image plane and samples of the N_e hypothesized edges are drawn.

However, the problem that occurs using this feature is that the occluded body parts will produce no edges. Even the visible parts, such as the arms, may not produce the edges, because of the color similarity between the part and the body. This will cause $p_j^e\left(\Gamma, Z_i\right)$ to be close to zero and thus will increase the squared difference function. Therefore, a good pose which represents well the visual context may be omitted. In order to overcome this problem for each combination of image plane and body part we calculate a coefficient, which indicates how well the part can be observed on this image. For each sample point on the model's edge we estimate the probability being covered by another body part. Let N_i be the number of hypothesized edges that are drawn for the part i. The total number of drawn sample points can be calculated using $N_e = \sum_{i=1}^{N_{bp}} N_i$, where N_{bp} is the total number of body parts in the model. The coefficient of part i for the image plane j can be calculated as following:

$$\lambda_{i,j} = \frac{1}{N_i} \sum_{k=1}^{N_i} \left(1 - p_k^{fg}\left(\Gamma_i, Z_j\right)\right)^2 \tag{13}$$

where Γ_i is the model configuration for part i and $p_k^{fg}\left(\Gamma_i, Z_j\right)$ is the value of the foreground pixel map of the sample k. If a body part is occluded by another one, then the value of $p_k^{fg}\left(\Gamma_i, Z_j\right)$ will be close to one and therefore the coefficient of this part for the specific camera will be low. We propose using the following function instead of sum-squared difference function as presented in (12):

$$\Sigma^e\left(\Gamma, Z\right) = \frac{1}{N_{cv}} \frac{1}{N_e} \sum_{i=1}^{N_{bp}} \sum_{j=1}^{N_{cv}} \lambda_{i,j} \overline{\Sigma}\left(\Gamma_i, Z_j\right) \tag{14}$$

where

$$\overline{\Sigma}\left(\Gamma_{bp}, Z_{cv}\right) = \sum_{k=1}^{N_i} \left(1 - p_k^e\left(\Gamma_{bp}, Z_{cv}\right)\right)^2 \tag{15}$$

The second feature is the silhouette obtained by subtracting the background from the image. The foreground pixel map is calculated for each image plane with

background pixels set to 0 and foreground set to 1 and sum squared difference function is computed:

$$\Sigma^{fg}(\Gamma, Z) = \frac{1}{N_{cv}} \frac{1}{N_e} \sum_{i=1}^{N_{cv}} \sum_{j=1}^{N_e} \left(1 - p_j^{fg}(\Gamma, Z_i)\right)^2 \tag{16}$$

where $p_j^{fg}(\Gamma, Z_i)$ is the value is the foreground pixel map values at the sample points.

The third feature is the foreground histogram. The reference histogram is calculated for each body part. It can be a grey level histogram or three separated histograms for color images, as shown in Fig. 3. Then, on each frame a normalized histogram is calculated for a hypothesized body part location and is compared to the referenced one. In order to compare the histograms we have used the squared Bhattacharya distance [32, 16], which provides a correlation measure between the model and the target candidates :

$$\Sigma^{h}(\Gamma, Z) = \frac{1}{N_{cv}} \frac{1}{N_{bp}} \sum_{i=1}^{N_{bp}} \sum_{j=1}^{N_{cv}} \left(1 - \rho^{part}(\Gamma_i, Z_j)\right) \tag{17}$$

where

$$\rho^{part}(\Gamma_{bp}, Z_{cv}) = \sum_{i=1}^{N_{bins}} \sqrt{p_i^{ref}(\Gamma_{bp}, Z_{cv})\, p_k^{hyp}(\Gamma_{bp}, Z_{cv})} \tag{18}$$

and $p_i^{ref}(\Gamma_{bp}, Z_{cv})$ is the value of bin i of the body part bp on the view cv in the reference histogram, and the $p_i^{hyp}(\Gamma_{bp}, Z_{cv})$ is the value of the corresponding bin on the current frame using the hypothesized body part location.

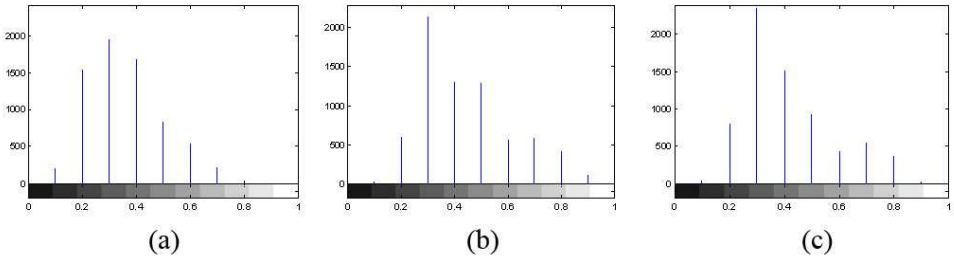

Fig. 3. The reference histograms of the torso: (a) red, (b) green and (c) blue colors of the reference selection.

The main drawback of that feature is that it is sensitive to changes in the lighting conditions. Therefore, the reference histogram has to be updated, using the weighted average from the recent history.

In order to calculate the total weighting function the features are combined together using the following formula:

$$w(\Gamma, Z) = e^{-\left(\Sigma^{e}(\Gamma,Z) + \Sigma^{fg}(\Gamma,Z) + \Sigma^{h}(\Gamma,Z)\right)} \tag{19}$$

As was stated above, the target of the tracking process is equal to maximizing the weighting function.

5.3 GPAPF Learning

The drawback in the particle filter tracker is that a high dimensionality of the state space causes an exponential increase in the number of particles that are needed to be generated in order to preserve the same density of particles. In our case, the data dimension is 29-D. In their work Balan et al. [5] show that the annealed particle filter is capable of tracking body parts with 125 particles using 60 fps video input. However, using a significantly lower frame rate (15 fps) causes the tracker to produce bad results and eventually to lose the target.

The other problem of the annealed particle filter tracker is that once a target is lost (i.e. the body pose was wrongly estimated, which can happen for the fast and not smooth movements) it is highly unlikely that the pose on the following frames will be estimated correctly.

In order to reduce the dimension of the space we introduce Gaussian Process Annealed Particle Filter (GPAPF). We use a set of poses in order to create a low dimensional latent space. The latent space is generated by applying non-linear dimension reduction on the previously observed poses of different motion types, such as walking, running, punching and kicking. We divide our state into two independent parts. The first part contains the global 3D body rotation and translation parameters and is independent of the actual pose. The second part contains only information regarding the pose (26 DoF). We use Gaussian Process Dynamical Model (GPDM) in order to reduce the dimensionality of the second part and to construct a latent space, as shown on Fig. 4. GPDM is able to capture properties of high dimensional motion data better than linear methods such as PCA. This method generates a mapping function from the low dimensional latent space to the full data space. This space has a significantly lower dimensionality (we have experimented with 2D or 3D). Unlike Urtasun et al. [18], whose latent state variables include translation and rotation information, our latent space includes solely pose information and is therefore rotation and translation invariant. This allows using the sequences of the latent coordinates in order to classify different motion types.

We use a 2-stage algorithm. In the first stage a set of new particles is generated of in the latent space. Then we apply the learned mapping function that transforms latent coordinates to the data space. As a result, after adding the translation and rotation information, we construct 31 dimensional vectors that describe a valid data state, which includes location and pose information, in the data space. In order to estimate how well the pose matches the images the likelihood function, as described in the previous section, is calculated.

The main difficulty in this approach is that the latent space is not uniformly distributed. Therefore we use the dynamic model, as proposed by Wang et al. [19], in order to achieve smoothed transitions between sequential poses in the latent space. However, there are still some irregularities and discontinuities. Moreover, while in a regular space the change in the angles is independent on the

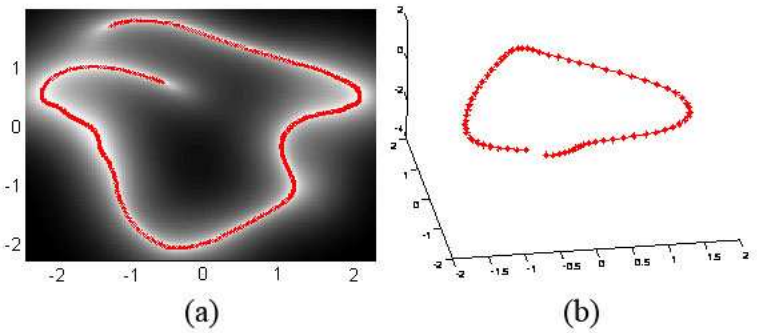

Fig. 4. The latent space that is learned from different poses during the walking sequence. (a) the 2D space; (b): the 3D space. On the image (a): the brighter pixels correspond to more precise mapping.

actual angle value, in a latent space this is not the case. Each pose has a certain probability to occur and thus the probability to be drawn as a hypothesis should be dependent on it. For each particle we can estimate the variance that can be used for generation of the new ones. In Fig. 4.(a) the lighter pixels represent lower variance, which depicts the regions of the latent space that produce more likely poses.

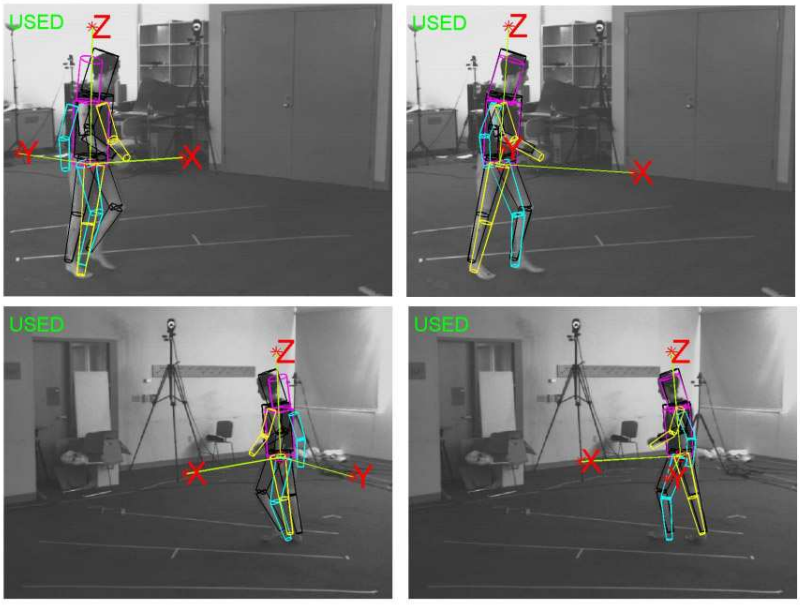

Fig. 5. Losing and finding the tracked target despite the miss-tracking on the previous frame. Top: camera 1, Bottom: camera 4.

Another advantage of this method is that the tracker is capable of recovering after several frames, from poor estimations. The reason for this is that particles

generated in the latent space are representing valid poses more authentically. Furthermore because of its low dimensionality the latent space can be covered with a relatively small number of particles. Therefore, most of possible poses will be tested with emphasis on the pose that is close to the one that was retrieved in the previous frame. So if the pose was estimated correctly the tracker will be able to choose the most suitable one from the tested poses. However, if the pose on the previous frame was miscalculated the tracker will still consider the poses that are quite different. As these poses are expected to get higher value of the weighting function the next layers of the annealing process will generate many particles using these different poses. As shown in Fig. 5 in this way the pose is likely to be estimated correctly, despite the miss-tracking on the previous frame.

In addition the generated poses are, in most cases, natural. The large variance in the data space causes the generation of unnatural poses by the CONDENSATION or by annealed particle filtering algorithms. In the introduced approach the poses that are produced by the latent space that correspond to points with low variance are usually natural as the whole latent space is constructed based on learning from a set of valid poses. **The unnatural poses correspond to the points with the large variance (black regions on Fig. 4.(a)) and, therefore, it is highly unlikely that it will be generated.** Therefore the effective number of the particles is higher, which enables more accurate tracking.

As shown in Fig. 4 is that the latent space is not continuous. Two sequential poses may appear not too close in the latent space; therefore there is a minimal number of particles that should be drawn in order to be able to perform the tracking.

The other drawback of this approach is that it requires more calculation than the regular annealed particle filter due to the transformation from the latent space into the data space. However, as it is mentioned above, if same number of particles is used, the amount of the effective poses is significantly higher in the GPAPF then in the original annealed particle filter. Therefore, we can reduce the number of the particles for the GPAPF tracker, and by this compensate for the additional calculations.

5.4 GPAPF Algorithm

As we have explained before we are using a 2-stage algorithm. The state consists of 2 statistically independent parts. The first one describes the body 3D location: the rotation and the translation (6 DoF). The second part describes the actual pose i.e. the latent coordinates of the corresponding point in the Gaussian Space (that was generated as we have explained in part 5.3). The second part usually has a very small DoF (as was mentioned before we have experimented with 2 and 3 dimensional latent spaces). The first stage is the generation of new particles. Then we apply the learned transform function that transforms latent coordinates to the data space (25 DoF). As the result, after adding the translation and rotation information, we construct a 31 dimensional vectors that describe a valid data state, which includes location and pose information, in the data space. Then

the state is projected to the cameras in order to estimate how well it fits the images.

Suppose we have M annealing layers. The state is defined as a pair $\Gamma = \{\Lambda, \Omega\}$, where Λ is the location information and Ω is the pose information. We also define ω as a latent coordinates corresponding to the data vector Ω: $\Omega = \wp(\omega)$, where $\wp$ is the mapping function learned by the GPDM. $\Lambda_{n,m}$, $\Omega_{n,m}$ and $\omega_{n,m}$ are the location, pose vector and corresponding latent coordinates on the frame n and annealing layer m. For each $1 \leq m \leq M-1$ $\Lambda_{n,m}$ and $\omega_{n,m}$ are generated by adding multi-dimensional Gaussian random variable to $\Lambda_{n,m+1}$ and $\omega_{n,m+1}$ respectively. Then $\Omega_{n,m}$ is calculated using $\omega_{n,m}$. Full body state $\Gamma_{n,m} = \{\Lambda_{n,m}, \Omega_{n,m}\}$ is projected to the cameras and the likelihood $\pi_{n,m}$ is calculated using likelihood function as explained in section 5.2 (see Algorithm 2).

In the original annealed particle filter algorithm the optimal configuration is achieved by calculating the weighted average of the particles in the last layer. However, as the latent space is not an Euclidian one, applying this method on ω will produce poor results. The other method is choosing the particle with the highest likelihood as the optimal configuration $\omega_n = \omega_{n,0}^{(i_{max})}$, where $i_{max} = \arg\min_i \left(\pi_{n,m}^{(i)}\right)$. However, this is unstable way to calculate the optimal pose, as in order to ensure that there exists a particle which represents the correct pose we have to use a large number of particles. Therefore, we propose to calculate the optimal configuration in the data space and then project it back to the latent space. At the first stage we apply the $\wp$ on all the particles to generate vectors in the data space. Then in the data space we calculate the average on these vectors and project it back to the latent space. It can be written as follows: $\omega_n = \wp^{-1}\left(\sum_{i=1}^{N} \pi_{n,0}^{(i)} \wp\left(\omega_{n,0}^{(i)}\right)\right)$.

5.5 Towards more precise tracking

The problem with such a 2-stage approach is that Gaussian field is not capable to describe all possible posses. As we have mentioned above, this approach resembles using probabilistic PCA in order to reduce the data dimensionality. However, for tracking issues we are interested to get the pose estimation as close as possible to the actual one. Therefore, we add an additional annealing layer as the last step. This stage consists from only one stage. We use data states, which were generated on the previous 2 staged annealing layer, described in previous section, in order to generate data states for the next layer. This is done with very low variances in all the dimensions, which practically are equal for all actions, as the purpose of this layer is to make only the slight changes in the final estimated pose. Thus it does not depend on the actual frame rate, contrary to original annealing particle tracker, where if the frame rate is changed one need to update the model parameters (the variances for each layer).

The final scheme of each step is shown in the Fig. 6 and described in Algorithm 3 . Suppose we have M annealing layers, as explained in section 5.4.

Algorithm 2 : The GPAPF algorithm

Initialization: $\left\{\Lambda_{n,M}^{(i)}; \omega_{n,M}^{(i)}; \frac{1}{N}\right\}_{i=1}^{N_p}$

for each: frame n

for $m = M$ downto 1 **do**

1. Calculate $\Omega_{n,M}^{(i)} = \wp\left(\omega_{n,M}^{(i)}\right)$ applying the prelearned by GPDM mapping function $\wp$ on the set of particles $\left\{\omega_{n,M}^{(i)}\right\}_{i=1}^{N_p}$.

2. Calculate the weights of each particle:
$\pi_n^{(i)} = k\frac{w^m\left(y_n, \Lambda_{n,m}^{(i)}, \omega_{n,m}^{(i)}\right) p\left(\Lambda_{n,m}^{(i)}, \omega_{n,m}^{(i)} | \Lambda_{n,m-1}^{(i)}, \omega_{n,m-1}^{(i)}\right)}{q\left(\Lambda_{n,m}^{(i)}, \omega_{n,m}^{(i)} | \Lambda_{n,m}^{(i)}, \omega_{n,m-1}^{(i)}, y_n\right)}$,
where $k = \left(\sum_{i=1}^{N_p} \frac{w^m\left(y_n, \Lambda_{n,m}^{(i)}, \omega_{n,m}^{(i)}\right) p\left(\Lambda_{n,m}^{(i)}, \omega_{n,m}^{(i)} | \Lambda_{n,m-1}^{(i)}, \omega_{n,m-1}^{(i)}\right)}{q\left(\Lambda_{n,m}^{(i)}, \omega_{n,m}^{(i)} | \Lambda_{n,m}^{(i)}, \omega_{n,m-1}^{(i)}, y_n\right)}\right)^{-1}$. Now the weighted set is constructed, which will be used to draw particles for the next layer.

3. Draw N particles from the weighted set $\left\{\Lambda_{n,m}^{(i)}; \omega_{n,m}^{(i)}; \pi_{n,m}^{(i)}\right\}_{i=1}^{N_p}$ with replacement and with distribution $p\left(\Lambda = \Lambda_{n,m}^{(i)}, \omega = \omega_{n,m}^{(i)}\right) = \pi_{n,m}^{(i)}$.

4. Calculate $\left\{\Lambda_{n,m-1}^{(i)}; \omega_{n,m-1}^{(i)}\right\} \sim q\left(\Lambda_{n,m-1}^{(i)}; \omega_{n,m-1}^{(i)} | \Lambda_{n,m}^{(i)}; \omega_{n,m}^{(i)}, y_n\right)$, which can be rewritten as $\Lambda_{n,m-1}^{(i)} \sim q\left(\Lambda_{n,m-1}^{(i)} | \Lambda_{n,m}^{(i)}, y_n\right) = \Lambda_{n,m}^{(i)} + n_m^{\Lambda}$ and $\omega_{n,m-1}^{(i)} \sim q\left(\omega_{n,m-1}^{(i)} | \omega_{n,m}^{(i)}, y_n\right) = \omega_{n,m}^{(i)} + n_m^{\omega}$, where n_m^{Λ} and n_m^{ω} are multivariate Gaussian random variables.

end for

- The optimal configuration can be calculated using the following formula:$\Lambda_n = \sum_{i=1}^{N_p} \pi_{n,1}^{(i)} \Lambda_{n,1}^{(i)}$ and $\omega_n = \omega_{n,1}^{(i_{max})}$, where $i_{max} = \arg\min_i \left(\pi_{n,1}^{(i)}\right)$.

- The unweighted particle set for the next observation is produced using $\Lambda_{n+1,M}^{(i)} = \Lambda_{n,1}^{(i)} + n_1^{\Lambda}$ and $\omega_{n+1,M}^{(i)} = \omega_{n,1}^{(i)} + n_1^{\omega}$, where n_1^{Λ} and n_1^{ω} are multivariate Gaussian random variables.

end for each

Then we add one more single-staged layer. In this last layer the $\Omega_{n,0}$ is calculated using only the $\Omega_{n,1}$ without calculating the $\omega_{n,0}$. We should also pay attention that the last layer has no influence of the quality of tracking in the following frames, as $\omega_{n,1}$ is used for the initialization of the next layer. Fig. 7 shows the difference between the version without the additional annealing layer and the results after adding it. We have used 5 2-staged annealing layers in both cases. For the second tracker we have added additional single staged layer. In the Fig. 8 shows the error graph that were produced by two trackers. The error was calculated, based on comparison of the tracker's output and the result of the MoCap system. The comparison was suggested by A. Balan [5]. This is done by calculating the 3D distance between the locations of the different joints that is estimated by the MoCap system and by the trackers results. The joints that are used are hips, knees, etc. The distances are summed and multiplied by the weight of the corresponding particle. Then the sum of the all weighted distances is calculated, which is used as an error measurement. We can see that the error, produced by GPAPF tracker without the additional layer (blue circles on the graph) is lower then the one produced by the original GPAPF algorithm with the additional annealing layer (red crosses on the graph) for the walking sequence taken at 30 fps. We can notice that the error is lower when we add the layer. However, as we have expected, the improvement is not dramatic. This is explained by the fact that the difference between the estimated pose using only the latent space annealing and the actual pose is not very big. That suggests that the latent space accurately represents the data space.

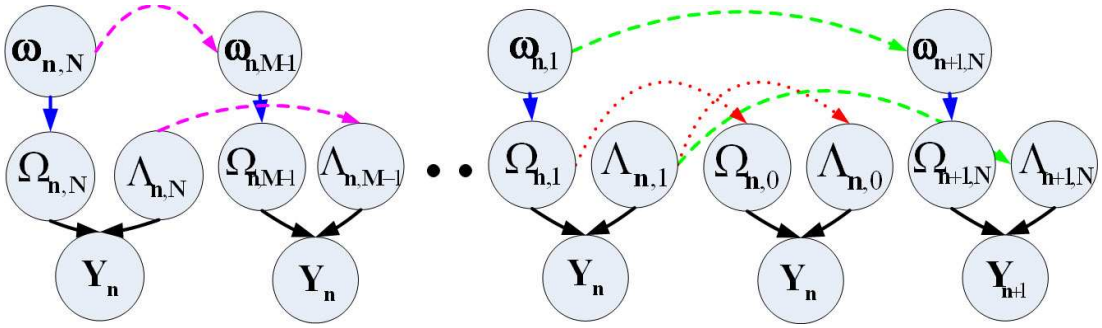

Fig. 6. GPAPF with additional annealing layer graphical model. The black solid arrows represent the dependencies between state and visual data; the blue arrows represent the dependencies between the latent space and the data space; dashed magenta arrows represent the dependencies between sequential annealing layers; the red arrows represent the dependencies of the additional annealing layer. The green arrows represent the dependency between sequential frames.

We can also notice that the improved GPAPF has less peaks on the error graph. The peaks stem from the fact that the *argmax* function, that has been used to find the optimal configuration, is very sensitive to the location of the best fitting particle. In the improved version, we calculate weighted average of all the

Algorithm 3 : The GPAPF algorithm with the additional layer

Initialization: $\left\{\Lambda_{n,M}^{(i)};\omega_{n,M}^{(i)};\frac{1}{N}\right\}_{i=1}^{N_p}$

for each: frame n

for $m = M$ downto 1 **do**

1. Calculate $\Omega_{n,M}^{(i)} = \wp\left(\omega_{n,M}^{(i)}\right)$ applying the prelearned by GPDM mapping function $\wp$ on the set of particles $\left\{\omega_{n,M}^{(i)}\right\}_{i=1}^{N_p}$.

2. Calculate the weights of each particle:
$\pi_n^{(i)} = k\frac{w^m\left(y_n,\Lambda_{n,m}^{(i)},\omega_{n,m}^{(i)}\right)p\left(\Lambda_{n,m}^{(i)},\omega_{n,m}^{(i)}|\Lambda_{n,m-1}^{(i)},\omega_{n,m-1}^{(i)}\right)}{q\left(\Lambda_{n,m}^{(i)},\omega_{n,m}^{(i)}|\Lambda_{n,m}^{(i)},\omega_{n,m-1}^{(i)},y_n\right)}$,
where $k = \left(\sum_{i=1}^{N_p}\frac{w^m\left(y_n,\Lambda_{n,m}^{(i)},\omega_{n,m}^{(i)}\right)p\left(\Lambda_{n,m}^{(i)},\omega_{n,m}^{(i)}|\Lambda_{n,m-1}^{(i)},\omega_{n,m-1}^{(i)}\right)}{q\left(\Lambda_{n,m}^{(i)},\omega_{n,m}^{(i)}|\Lambda_{n,m}^{(i)},\omega_{n,m-1}^{(i)},y_n\right)}\right)^{-1}$.

3. Draw N particles from the weighted set $\left\{\Lambda_{n,m}^{(i)};\omega_{n,m}^{(i)};\pi_{n,m}^{(i)}\right\}_{i=1}^{N_p}$ with replacement and with distribution $p\left(\Lambda = \Lambda_{n,m}^{(i)},\omega = \omega_{n,m}^{(i)}\right) = \pi_{n,m}^{(i)}$.

4. Calculate $\left\{\Lambda_{n,m-1}^{(i)};\omega_{n,m-1}^{(i)}\right\} \sim q\left(\Lambda_{n,m-1}^{(i)};\omega_{n,m-1}^{(i)}|\Lambda_{n,m}^{(i)};\omega_{n,m}^{(i)},y_n\right)$, which can be rewritten as $\Lambda_{n,m-1}^{(i)} \sim q\left(\Lambda_{n,m-1}^{(i)}|\Lambda_{n,m}^{(i)},y_n\right) = \Lambda_{n,m}^{(i)} + n_m^{\Lambda}$ and $\omega_{n,m-1}^{(i)} \sim q\left(\omega_{n,m-1}^{(i)}|\omega_{n,m}^{(i)},y_n\right) = \omega_{n,m}^{(i)} + n_m^{\omega}$, where n_m^{Λ} and n_m^{ω} are multivariate Gaussian random variables.

end for

- The optimal configuration can be calculated by the following steps:

1. Calculate $\left\{\Lambda_{n,m-1}^{(i)};\Omega_{n,m-1}^{(i)}\right\} \sim q\left(\Lambda_{n,m-1}^{(i)};\Omega_{n,m-1}^{(i)}|\Lambda_{n,m}^{(i)};\Omega_{n,m}^{(i)},y_n\right)$, which can be rewritten as $\Lambda_{n,m-1}^{(i)} \sim q\left(\Lambda_{n,m-1}^{(i)}|\Lambda_{n,m}^{(i)},y_n\right) = \Lambda_{n,m}^{(i)} + n_m^{\Lambda}$ and $\Omega_{n,m-1}^{(i)} \sim q\left(\Omega_{n,m-1}^{(i)}|\Omega_{n,m}^{(i)},y_n\right) = q\left(\Omega_{n,m-1}^{(i)}|\wp\left(\Omega_{n,m}^{(i)}\right),y_n\right) = \wp\left(\Omega_{n,m}^{(i)}\right) + n_m^{\Omega}$, where n_m^{Λ} and n_m^{Ω} are multivariate Gaussian random variables.

2. Draw N particles from the weighted set $\left\{\Lambda_{n,m}^{(i)};\Omega_{n,m}^{(i)};\pi_{n,m}^{(i)}\right\}_{i=1}^{N_p}$ with distribution $p\left(\Lambda = \Lambda_{n,m}^{(i)},\Omega = \Omega_{n,m}^{(i)}\right) = \pi_{n,m}^{(i)}$. Calculate the weight of each particle.

3. The optimal configuration is $\Lambda_n = \sum_{i=1}^{N_p}\pi_{n,0}^{(i)}\Lambda_{n,0}^{(i)}$ and $\Omega_n = \sum_{i=1}^{N_p}\pi_{n,0}^{(i)}\Omega_{n,0}^{(i)}$.

- The unweighted particle set for the next observation is produced using $\Lambda_{n+1,M}^{(i)} = \Lambda_{n,1}^{(i)} + n_0^{\Lambda}$ and $\omega_{n+1,M}^{(i)} = \omega_{n,1}^{(i)} + n_0^{\omega}$, where n_0^{Λ} and n_0^{ω} are multivariate Gaussian random variables.

end for each

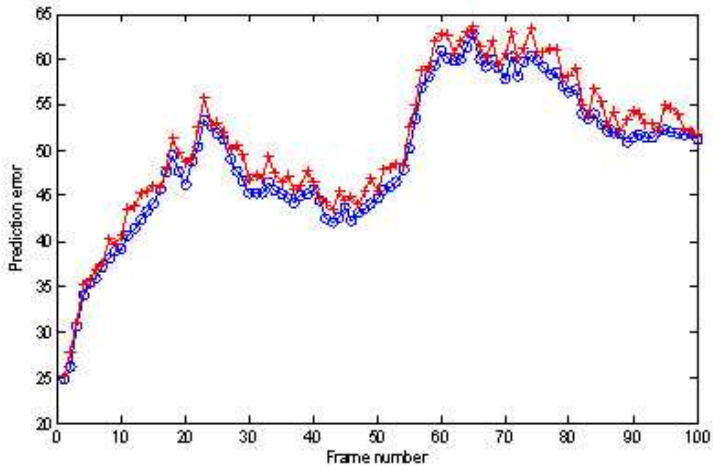

Fig. 7. The errors GPAPF tracer with additional annealing layer (blue circles) and without it (red crosses) for a walking sequence captured at 30 fps.

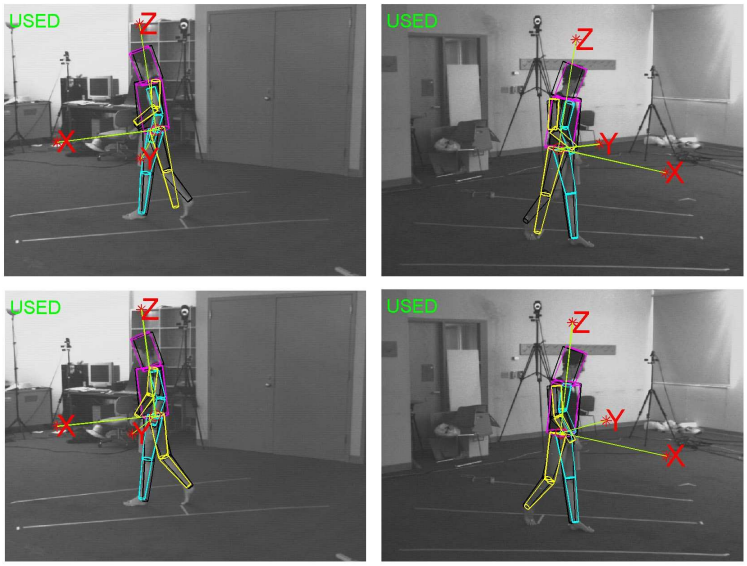

Fig. 8. GPAPF algorithm before (top) and after (bottom) adding additional annealed layer.

particles. As we have seen from our experiments, there are often many particles with the weight close to the optimal. Therefore, the result is less sensitive to the location of some particular particle. It depends on the whole set of them.

We have also tried to use the results, produced by the additional layer, in order to initialize the state in the next time step. This was done by applying the inverse function $\wp^{-1}$, suggested by Lawrence [8], on the particles that were generated in previous annealing layer. However, this approach did not produce any valuable improvement in the tracking results. As the inverse function is computationally heavy it caused significant increase the calculation time. Therefore, we decided not to experiment with it further.

6 Results

We have tested GPAPF tracking algorithm using HumanEva dataset [22]. The sequences contain different activities, such as walking, boxing etc. which were captured by 7 cameras; however we have used only 4 inputs in our evaluation. The sequences were captured using the MoCap system that provides the correct 3D locations of the body parts for evaluation of the results and comparison to other tracking algorithms.

The first sequence that we have used was a walk on a circle. The video was captured at frame rate 120 fps. We have tested the annealed particle filter based body tracker, implemented by A. Balan, and compared the results with the ones produced by the GPAPF tracker. The error was calculated, based on comparison of the tracker's output and the result of the MoCap system, using average distance between 3-D joints location, as explained in section 5.4. Fig. 10 shows the error graphs, produced by GPAPF tracker (blue circles) and by the annealed particle filter (red crosses) for the walking sequence taken at 30 fps. As can be seen, the GPAPF tracker produces more accurate estimation of the body location. Same results were achieved for 15 fps. Fig. 9 presents sample images with the actual pose estimation for this sequence. The poses are projected to the first and second cameras. The first 2 rows show the results of the GPAPF tracker. The third and forth rows show the results of the annealed particle filter.

We have experimented with 100 particles up to 2000 particles. For the 100 particles per layer using 5 annealed layers the computational cost was 30 sec per frame. Using the same number of particles and layers in the annealed particle filter algorithm takes 20 seconds per frame. However, the annealed particle filter algorithm was not capable of tracking the body pose with such a low number of particles for 30 fps and 15 fps videos. Therefore, we had to increase the number of particles used in the annealed particle filter to 500.

We have also tried to compare our results to the results of CONDENSATION algorithm. However, the results of

the condensation algorithm were either very poor or a very large number of particles needed to be used, which made this algorithm computationally not effective. Therefore we do not show the results of this somparisson.

The second sequence was captured in our lab. On that sequence we have filmed similar behavior, produced by a different actor. The frame rate was 15 fps. In case of walking, the learning was done on the first sequence data. The GPAPF tracker was able to track the person and produced results similar to the ones, which were produced for the original sequence.

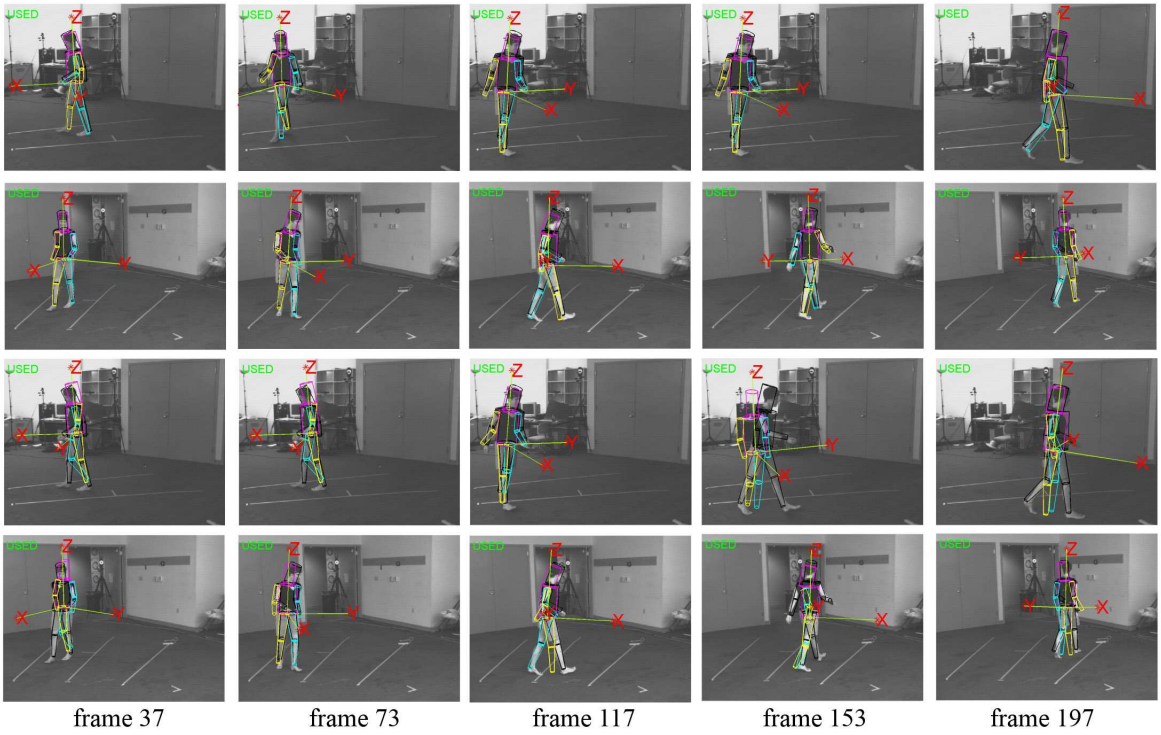

Fig. 9. Tracking results of annealed particle filter tracker and GPAPF tracker. Sample frames from the walking sequence. First row: GPAPF tracker, first camera. Second row: GPAPF tracker, second camera. Third row: annealed particle filter tracker, first camera. Forth row: annealed particle filter tracker, second camera.

We have also experimented with sequences containing different behavior, like leg movements, object lifting, clapping and boxing. We have manually marked some of the sequences in order to produce the needed training sets for GPDM. After the learning we have run the validation on the other sequences containing same behavior. As it is shown on the Fig. 11, the tracker successfully tracked these sequences. We have experimented with 100 going up to 2000 particles. For the 100 particles the computational cost was 30 sec/frame. The results that are

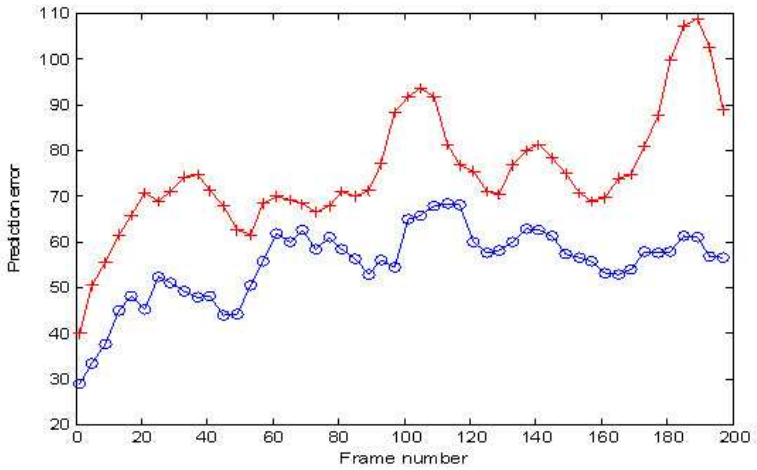

Fig. 10. The errors of the annealed tracker (red crosses) and GPAPF tracker (blue circles) for a walking sequence captured at 30 fps.

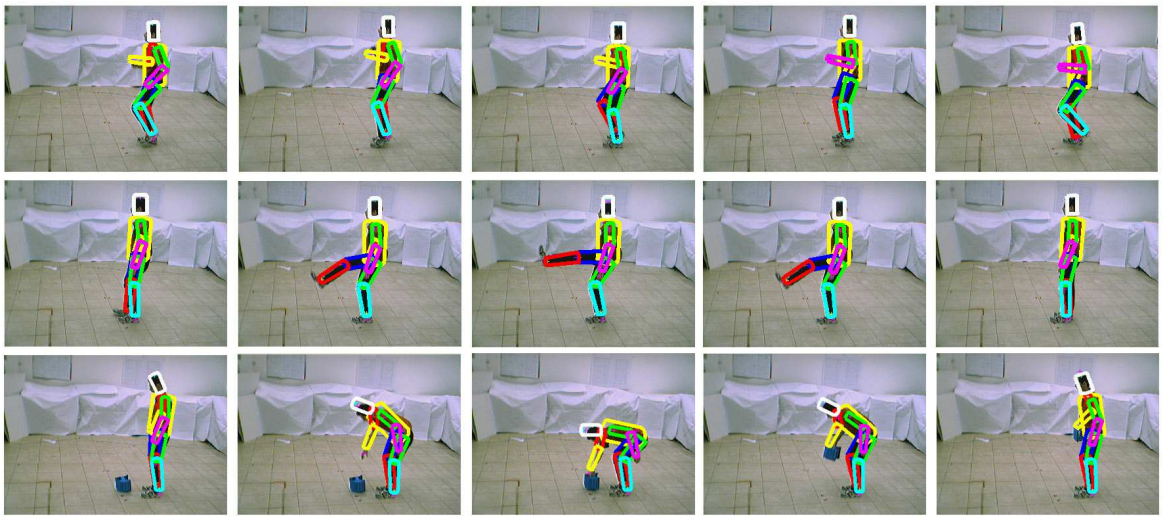

Fig. 11. Tracking results of annealed particle filter tracker and GPAPF tracker. Sample frames from the running, leg movements and object lifting sequences.

shown in the videos are done with 500 particles (2.5 min per frame). The code that we are using is written in Matlab with no optimization packages. Therefore the computational cost can be significantly reduced if moved to C libraries.

7 Conclusion and Future Work

We have presented an approach that uses GPDM in order to reduce the dimensionality and in this way to improve the ability of the annealed particle filter tracker to track the object even in a high dimensional space. We have also shown that using GPDM can increase the ability to recover from temporal target loss. We have also presented a method to approximate the possibility of self occlusion and we have suggested a way to adjust the weighed function for such cases, in order to be able to produce more accurate evaluation of a pose.

The main problem is that the learning and tracking are done for a specific action. The ability of the tracker to use a latent space in order to track a different motion type, has not been shown yet. A possible approach is to construct a common latent space for the poses from different actions. The difficulty with such approach may be the presence of a large number of gaps between the consecutive poses. In the future we plan to extend the approach in order to be able to track different activities, using same learned data.

The other challenging task is to track two or more people simultaneously. The main problem here is that in this case there is high possibility of occlusion. Furthermore, while for a single person each body part can be seen from at least one camera that is not the case for the crowded scenes.

References

1. Agarwal A. and Triggs B. 3d human pose from silhouettes by relevance vector regression. *Proc. CVPR*, 2:882–888, 2004.
2. Agarwal A. and Triggs B. Learning to track 3d human motion from silhouettes. *Proc. ICML*, page 916, 2004.
3. Deutscher J. Blake A. and Reid I. Articulated body motion capture by annealed particle filtering. *Proc. CVPR*, pages 2126–2133, 2000.
4. Elgammal A.M. and Lee C. Inferring 3d body pose from silhouettes using activity mani-fold learning. *Proc. CVPR*, 2:681–688, 2004.
5. A. Sigal L. Balan and Black. M. A quantitative evaluation of video-based 3d person tracking. *IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 349–356, 2005.
6. Bregler C. and Malik J. Tracking people with twists and exponential maps. *Proc. CVPR*, pages 8–15, 1998.
7. Lawrence N. D. Gaussian process latent variable models for visualization of high dimensional data. *Information Processing Systems (NIPS)*, 16:329–336, 2004.
8. Lawrence N. D. and Candela J. Q. Local distance preservation in the gp-lvm through back constraints. *Proc. International Conference on Machine Learning (ICML)*, pages 513–520, 2006.
9. Ramanan D. and Forsyth D. A. Automatic annotation of everyday movements. *Information Processing Systems (NIPS)*, 2003.

10. Tenenbaum J. B. de Silva and V. Langford. J.C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.
11. Raskin L. Rivlin E. and Rudzsky M. 3d human tracking with gaussian process annealed particle. *Proc. VISAPP*, pages 661–675, 2007.
12. Mori G. and Malik J. Estimating human body configurations using shape context matching. *Proc. ECCV*, 3:134–141, 2002.
13. Wang Q. Xu G. and Ai H. Learning object intrinsic structure for robust visual tracking. *Proc. CVPR*, 2:227–233, 2003.
14. Davison A. J. Deutscher J. and Reid I. D. Markerless motion capture of complex full-body movement for character animation. *In Eurographics Workshop on Computer Anima-tion and Simulation*, pages 3–14, 2001.
15. Deutscher J. and Reid I. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205, 2004.
16. Perez P. Hue C. Vermaak J. and Gangnet M. Color-based probabilistic tracking. *Proc. ECCV*, pages 661–675, 2002.
17. Sidenbladh H. Black M. J. and Sigal L. Implicit probabilistic models of human motion for synthesis and tracking. *Proc. ECCV*, 1:784–800, 2002.
18. Urtasun R. Fleet D. J. and Fua P. 3d people tracking with gaussian process dynamical models. *Proc. CVPR*, 1:238–245, 2006.
19. Wang J. Fleet D. J. and Hetzmann A. Gaussian process dynamical models. *Information Processing Systems (NIPS)*, pages 1441–1448, 2005.
20. Mikolajczyk K. Schmid K. and Zisserman. A. Human detection based on a probabilistic assembly of robust part detectors. *Proc. ECCV*, 1:69–82, 2003.
21. Rohr K. Human movement analysis based on explicit motion models. *Motion-Based Recognition*, 8:171–198, 1997.
22. Sigal L. and Black M.J. Humaneva: Cynchronized video and motion capture dataset for evaluation of articulated human motion. *Technical Report CS-06-08*, 2006.
23. Isard M. and Blake A. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
24. Isard M. and MacCormick J. Bramble: A bayesian multiple blob tracker. *Proc. ICCV*, pages 34–41, 2001.
25. Ormoneit D. Sidenbladh H. Black M. and Hastie T. Learning and tracking cyclic human motion. *Advances in Neural Information Processing Systems*, 13:894–900, 2001.
26. Raskin L. Rudzsky M. and Rivlin E. Gpapf: A combined approach for 3d body part tracking. *Proc. ICVS*, 2007.
27. Sidenbladh H. Black M. and Fleet D. Stochastic tracking of 3d human figures using 2d image motion. *Proc. ECCV*, 2:702–718, 2000.
28. Urtasun R. Fleet D.J. Hertzmann A. Fua P. Priors for people tracking from small training sets. *Proc. ICCV*, 1:403–410, 2005.
29. Urtasun R. and Fua P. 3d human body tracking using deterministic temporal motion models. *Proc. ECCV*, 3:92–106, 2004.
30. Ioffe S. and Forsyth. D. A. Human tracking with mixtures of trees. *Proc. ICCV*, 1:690–695, 2001.
31. Roweis S. T. and Saul L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
32. Comaniciu D. Ramesh V. and Meer P. Kernel-based object tracking. *IEEE Trans. PAMI*, 25(5):564–577, 2003.
33. Song Y. Feng X. and Perona P. Towards detection of human motion. *Proc. CVPR*, pages 810–817, 2000.