HAL
open science

Pose Estimation For A Partially Observable Human Body From RGB-D Cameras

Abdallah Dib, François Charpillet

To cite this version:

Abdallah Dib, François Charpillet. Pose Estimation For A Partially Observable Human Body From RGB-D Cameras. IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), Sep 2015, Hamburg, Germany. pp.8. hal-01203638

HAL Id: hal-01203638

https://inria.hal.science/hal-01203638

Submitted on 23 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Pose Estimation For A Partially Observable Human Body From RGB-D Cameras

Abdallah Dib† and François Charpillet†

***Abstract*— Human pose estimation in realistic world conditions raises multiple challenges such as foreground extraction, background update and occlusion by scene objects. Most of existing approaches were demonstrated in controlled environments.**

In this paper, we propose a framework to improve the performance of existing tracking methods to cope with these problems. To this end, a robust and scalable framework is provided composed of three main stages. In the first one, a probabilistic occupancy grid updated with a Hidden Markov Model used to maintain an up-to-date background and to extract moving persons. The second stage uses component labelling to identify and track persons in the scene. The last stage uses an hierarchical particle filter to estimate the body pose for each moving person. Occlusions are handled by querying the occupancy grid to identify hidden body parts so that they can be discarded from the pose estimation process.

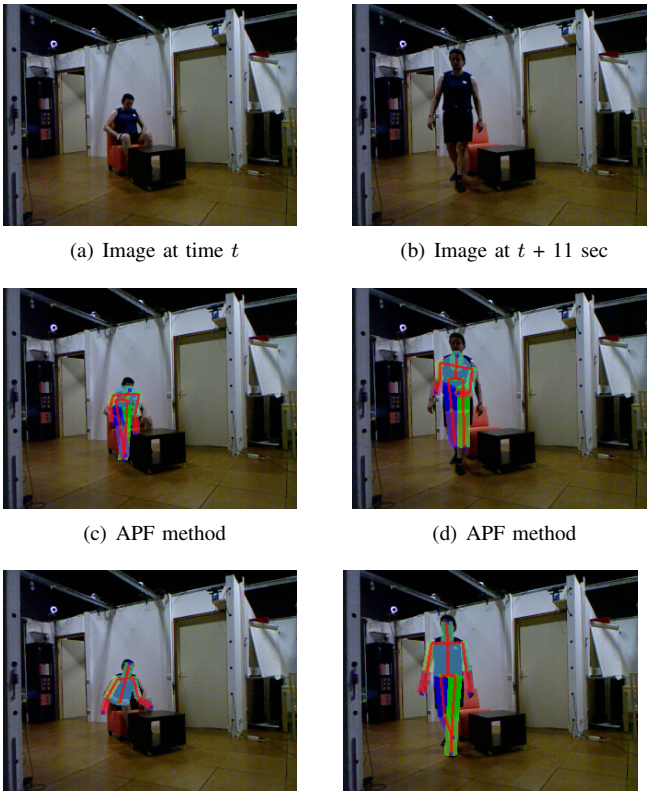
We provide a parallel implementation that runs on CPU and GPU at 4 frames per second. We also validate the approach on our own dataset that consists of synchronized motion capture with a single RGB-D camera data of a person performing actions in challenging situations with severe occlusions generated by scene objects. We make this dataset available online.

I. Introduction

Markerless human body pose estimation consists of tracking body parts by analyzing a sequence of input images from single or multiple cameras. It is still a fundamental challenge in robotics and computer vision. Its application domains are human-robot interaction, gait analysis, computer animation, etc.

For the human-robot interaction and assistance tasks, human pose estimation is a fundamental problem to solve in order for the robot to interact and assist the person. This must be done in daily life situations where occlusion with scene objects and background changes frequently occur. Additionally, applications such as activity recognition typically rely on a precise pose estimation. To this end, it is very important to provide a robust pose estimation method working in such conditions.

Different approaches for human pose estimation have been developed and demonstrated in controlled environments. What we propose in this paper, is an adaptation and improvement of these existing methods, that allow them to work in real-world conditions. For instance, in Figure 1, only the upper body part is visible of a subject sitting with a table in front of him. If we consider a straightforward implementation of a well known and widely used approach such as [1] or [2], it will certainly fail because it keep tracking hidden body parts which generates inconsistent hypotheses. This is illustrated in the Figure 1 where legs are invisible. In fact, the method tries to bend the legs to the torso to reduce the cost function which yields to a tracking failure when the legs are visible again. To cope with such situations, we propose a method that can identify and exclude hidden parts in order to keep a consistent pose estimation for only visible parts. In this sense, when these hidden parts are visible again, the tracking is not lost as shown in Figure 1.



(a) Image at time t (b) Image at t + 11 sec

(c) APF method (d) APF method

(e) OUR method (f) OUR method

Fig. 1. Comparison of our method with APF on a situation from sequence 10 of our dataset where a subject enters and exit an occlusion situation and only upper body part is visible. a) A person enters the occlusion situation. b) The person exits the occlusion space. c) and d) APF method fails. e) and f) Our method successfully discards hidden parts and produces accurate pose estimation.

Beside scene object occlusions, maintaining an up-to-date background and extracting persons from the scene is another challenge to tackle in order to improve the tracking in real-

*This work was supported by the LAR project

†Inria, Villers-lès-Nancy, 54600, France.

Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France.

CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France.
(abdallah.dib, francois.charpillet)@inria.fr

world scenarios.

To cope with all these problems, we propose a *model based* pose estimation method for depth cameras composed of three stages: The first one uses a probabilistic occupancy grid updated with a Hidden Markov Model (HMM) to maintain an up-to-date background and to extract moving persons from a dynamic scene. In this way, our method does not require any background initialization unlike some classic approaches that assume that there should be no moving objects in the initialization phase and that the scene is static.

In the next stage, we apply a component labelling pass on moving cells to label different subject in the scene. These subjects are then tracked over time.

The final stage consists of estimating the human pose for each subject in the scene. This stage detects hidden body parts and excludes them from the hierarchical particle filter.

The first contribution of this work is the introduction of a robust method for extracting moving persons from a dynamic scene without any need for an initialization phase. This is done with a three states Hidden Markov Model (HMM) applied for each cell of the occupancy grid to identify moving objects from static background. This is detailed in section III-A. The use of the occupancy grid makes the proposed framework easily scalable by having a unique world representation, which can be easily adapted to work with multiple static or moving cameras. Occupancy grids are widely used in the robotics field for localization and mapping, that is why our approach is attractive for the human-robot interaction community.

The second contribution of this work is the proposal of a method that can identify and eliminate hidden parts from the pose estimation so that the hierarchical particle filter only process visible parts. This is detailed in section III-C. The robustness of our method is evaluated with a set of experiments in challenging situations in section IV.

The last contribution of this paper, is the proposal of a rich dataset of synchronized RGB-D camera and motion capture data for a person performing daily life activities in a scene with severe object occlusion. This dataset is available online on the following link [3].

II. Related Work

Different approaches for human pose estimation have been developed in the literature. In the Moeslund et al [4] survey, the *pose estimation* problem can be solved by three categories of solutions: the *Model-free* technique aims to recover the pose without an explicit prior model; the *Indirect model* uses an a priori model as a reference for guiding the interpretation of observed data; while the *Direct model* uses an explicit 3D model representation of a human to recover the body pose and which is the most widely investigated approach.

Among the model-based approaches, stochastic tracking techniques such as Particle filtering [5] are promising and widely investigated in the literature because they do not require a complex analytical calculation and they use a set of particles to approximate the non-gaussian density distribution of state estimation. However, particle filtering in its original form requires a very high number of particles to approximate the density distribution. Two main strategies were developed that aim to reduce this complexity. The first one uses simulated annealing as introduced by [1] called APF. The second approach uses Portioned Sampling (PS) of the state space proposed by [2].

It has been shown in [6] that these approaches tend to fail in realistic world conditions with occlusions. For that, several methods were investigated to deal with such conditions. Leen et al. [7] proposed a particle filter approach with analytical inference to reduce the degrees of freedom and to recover after occlusion, but to work, an accurate analytical detection is needed which is not always available. Peursum et al. [6] proposed an action recognition module that assists the motion tracking algorithm during occlusion. As mentioned by the author of this work, this method gives rough, not accurate pose tracking results.

Bandouch et al. [8] proposed a combined approach between APF and PS that gives more accurate results. Scene objects that are candidates for occlusion are manually labelled before tracking which make their approach inappropriate for real applications.

Lee and Nevatia [9] proposed a three stages system for 3D pose estimation from monocular camera, that consists on extracting people blobs. An histogram appearance model is learned for each blob in order to detect occlusions between persons. The main disadvantage of this approach is the computation time, where each frame needs 5 minutes to be processed, additionally this method was not tested in challenging situations where most of the body parts are occluded by scene objects. Zhao and Nevatia [10] and Gammeter et al. [11], address the problem of multi-person pose estimation in street scenes. Following the same line, [12] proposes a 3D human pose estimation method for crowded scene guided by a 2D people detection module that uses Pictorial Structures [13]. In the last few years, there has been a significant approaches like [14] that uses Pictorial Structures for pose estimation.

Stoll et al. [15] and Liu et al.[16], handle occlusion generated by human-human interaction and scene object. A set-up of 12 cameras is used in these methods.

Recently, RGB-D sensors were introduced for human motion capture using machine learning approaches such as random forests [17], [18]. Other model-based approaches [19], [20] produce impressive, accurate tracking and achieve real-time performance.

The authors in [19], [20] does not compare their method in scenes with severe occlusion caused by scene objects.

Zhang et al. [21] uses multiple depth sensors to handle occlusions by fusing depth images into one single point cloud and a truncated signed distance as an observation function within a combined particle filter similar to the one proposed by [8]. The authors claims that using one camera set-up is highly sensitive to occlusions, and that by using multiple depth cameras they manage to reduce ambiguities. While their approach gives interesting results, it still does not deal

with more challenging conditions when some body parts are not seen by any camera as in Figure 1. We show in the next the capacity of our method to handle occlusions even with one single depth camera.

Finally, Rhodin et al. [22] proposed a pose estimation method for interacting humans in unconstrained environment from hand-held depth sensors. Skeletal position and camera poses are estimated simultaneously by solving a joint energy minimization problem.

This paper takes a different perspective. In fact, existing methods handle self-occlusions, scene object occlusions and humans interaction occlusions by trying to estimate the position of hidden parts by relying on the redundancy of information from multiple cameras which is not always a reliable solution because there is always some situations in real conditions where occlusions are present even with a very high number of cameras. For this, our method handle occlusion differently. It discards hidden body parts when they are not visible. as shown in Figure 1, when the person enters an occluded place, hidden parts are detected and excluded from the pose estimation process. When they are again visible, the search intervals for these hidden parts is increased in order to resume tracking for these body parts. We show the capacity of our method to handle severe occlusions even with a single monocular depth camera.

III. Method

In this section, we describe the different stages of our framework. As shown in Figure 2, the system receives depth images as input. The first stage consists of an occupancy grid that extracts moving and static cells in the scene. This stage is detailed in section III-A. In the next stage, a component labelling algorithm is run across moving cells to extract and track different persons in the scene which is discussed in section III-B. The final stage identify hidden body parts and apply a particle filter to estimate the human body pose on only visible parts. This is described in section III-C.

A. Occupancy Grid and HMM model

The role of this stage is to maintain an up-to-date background and extract moving subjects from the scene. This is done by using a 3D occupancy grid with a HMM.

Occupancy grid was first introduced by [23] for robot localization and navigation tasks. It consists on representing the world with a rigid structure composed of small cells (also called voxels or cubes) of a fixed size. A grid is defined by its size and resolution. Smaller resolution gives more accurate world representation but requires more computational power. Typical occupancy grids defines two states for each cell: Occupied or free. In order to distinguish between cells occupied by static and mobile objects, we added a new state for each cell. At the end, a cell is represented with three states:

- "O" The cell is occupied by a static object from the environment.
- "M" The cell is occupied by a moving object.
- "E" The cell is empty (not occupied).

For this, a three states HMM is used to model the cell state.

A HMM requires a state transition probability, an observation model and an initial state distribution.

Let c_t be a discrete variable that represents the occupancy state for each cell at time t and $p(c_t)$ represents the occupancy probability of a cell at time t.

The state transition probability $p(c_t/c_{t-1})$ represents the way the occupancy state of a cell changes between two time steps. As shown in Figure 4, α, β and γ represent the transition probabilities between each state. Therefore, the transition matrix that represents the dynamic of a cell state can be written as follow:

$$A = \begin{pmatrix} 1-\alpha & 0 & \alpha \\ 0 & 1-\gamma & \gamma \\ 0 & \beta & 1-\beta \end{pmatrix}$$

The observation model $p(z/c)$ represents the likelihood of the observation z from the depth sensor given the state of the cell. This observation z is obtained from the depth image of the RGB-D sensor.

Let l, be the distance of the voxel to the camera in its coordinate frame, d the sensor measurement that represents the distance of the pixel to the camera, and ε the difference between l and d. So that $\varepsilon = d - l$. There is three cases to distinguish:

- $\varepsilon = 0 \Rightarrow$ voxel is occupied.
- $\varepsilon < 0$ $(d < l) \Rightarrow$ voxel is not visible.
- $\varepsilon > 0$ $(d > l) \Rightarrow$ voxel is not occupied.

From these three cases, we derived the observation function $P(z/c_t) = f(\varepsilon)$ represented in Figure 3 to evaluate the occupancy probability for each cell.

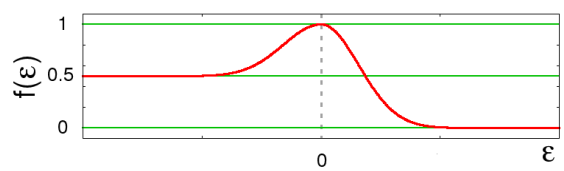


Fig. 3. The observation function used by the HMM.

Thus, the three observation functions are given as follow:

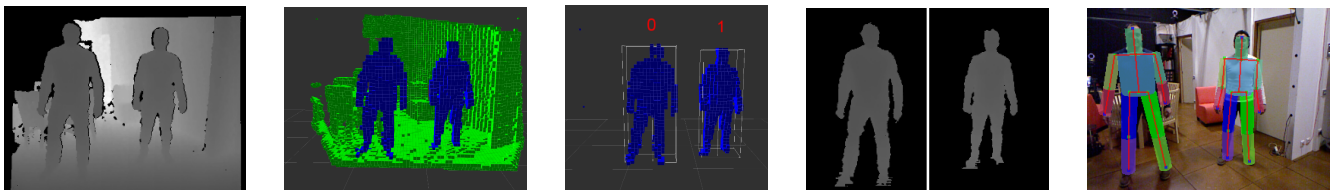
- $P(z/c_t = O) = f(\varepsilon)$
- $P(z/c_t = M) = f(\varepsilon)$
- $P(z/c_t = E) = 1 - f(\varepsilon)$

Having the same observation function for states "M" and "O" is known as the aliasing problem, and HMM is used to solve it.

The observation model at each time step t can be represented by the following matrix:

$$B_t = \begin{pmatrix} f(\varepsilon) & 0 & 0 \\ 0 & f(\varepsilon) & 0 \\ 0 & 0 & 1-f(\varepsilon) \end{pmatrix}$$

Finally, the posterior probability distribution for each cell $p(c_t/z_{1:t})$ is updated as follows:



(a) (b) (c) (d) (e)

Fig. 2. a) Input depth image. b) The first stage outputs moving (blue) and static (green) voxels in the scene. c) The second stage labels moving cells and track these labels over time. d) Depth image is extracted for each label by back projecting each label to each camera. e) The output of the pose estimation stage that extracts skeleton information for each subject in the scene.

$$Q_t = {Q_{t-1}}^T \times A \times B_t,$$

with Q_t equal to:

$$Q_t = \begin{pmatrix} p(c = O/z_{1:t}) \\ p(c = M/z_{1:t}) \\ p(c = E/z_{1:t}) \end{pmatrix},$$

Q_t is then normalized with the following normalization factor: $D = p(c = O/z_{1:t}) + p(c = M/z_{1:t}) + p(c = E/z_{1:t})$.

Cells are finally classified by choosing the maximum *a posteriori* (MAP) which is the most likely state of the HMM. Figure 2(b) shows the output of this stage where green color represents cells occupied with static objects (the background) and blue one represents cells occupied with moving one (the foreground). With the current HMM model, moving persons are extracted, but in the other hand, if a furniture has moved in the scene, it will be detected as moving also. In order to maintain an up-to-date background, moving objects like furnitures must become static again once they are not moving. In order to achieve this results, we added a new transition probability ψ between M and O states as illustrated in Figure 4. As shown in Figure 5, with the new HMM, after a certain time, the chair becomes static again. This is best illustrated in the attached video.

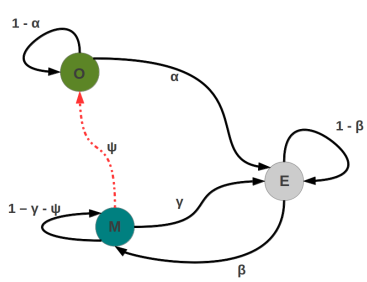
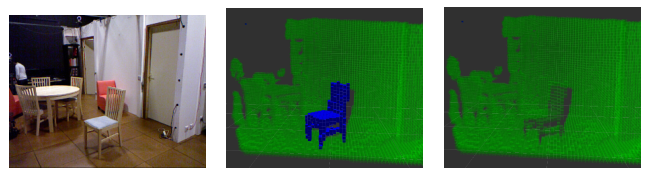


Fig. 4. The three state HMM used to identify the occupancy state of each cell in the grid.

At the same time, when traversing the grid, we label each cell as visible or not by the current point of view of the camera. This is done by projecting each cell to the camera depth image and comparing the value of the depth pixel with the depth of the cell. Based on this comparison we can determine the visibility state for each cell. This step is important for the third stage to determine whether a body part of the 3D model is visible or not, this will be discussed in the section III-C.



(a) (b) (c)

Fig. 5. a) Input image shows a chair that has been moved. b) The chair is identified as moving object. c) The same chair is static again after certain time. In this sense, our method maintain an up-to-date background.

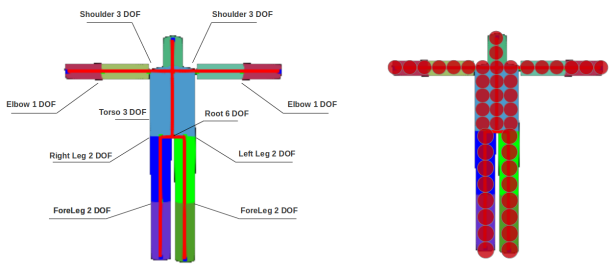
B. Component labelling

The role of this stage is to assemble moving cells returned by the first stage into one seperate entities and to track them over time. For this, a component labelling algorithm is run across moving voxels in order to label different persons in the scene. The algorithm starts by assigning a different label for each cell. Then for each label, it finds the neighbour that has the smallest label and assigns it to the current cell. This process is repeated until there is no changes in the assignment. At the end, cells that have the same label value are considered as a single moving person called label. For each label, the center of mass is calculated for each frame t. To track this labels over time, let $L_t = ({l_1}^t, {l_2}^t, ..{l_n}^t)$ be a set of n labels found at time t and L_{t+1} a set of m labels found at instant $t+1$. In order to track these labels, each label ${l_i}^{t+1}$ in L_{t+1} is matched to the closest one in L_t based on a predefined threshold euclidean distance between the center of mass of each label. Figure 2(c) shows the output of this stage, where the two subjects are identified separately.

C. 3D human pose estimation

In this section, we discuss the third stage of the system. It consists on estimating the 3D pose of moving persons in the scene. The body model used for the tracking consists of 10 rigid articulated segments. These parts are constrained to each others according to a skeleton as show in Figure 6(a). The body model consists of 25 degrees of freedom. The geometry of each body part is represented with a cylinder defined by its top, bottom radius and its length.

These set of articulated segments find a natural representation as a Dynamic Bayesian Network (DBN) which reduces the inference complexity. Particle filters uses a set of particles to estimate a posteriori probabilities in the DBN given a



(a) (b)

Fig. 6. a) The body model used for tracking. The Degrees Of Freedom (DOF) for each body parts is shown. Which gives a total of 25 DOF. b) Each body part is approximated with a set of spheres (See section III-D).

sequence of observations.

A typical particle filter relies on three essential steps: The first one is the re-sampling step that consists on drawing a new set of un-weighted particles from the original weighted set. The second step consists of moving these particles based on a motion model and the last step produces a new weighted set of particles that represents the pdf by assigning a weight for each particle based on an observation function. A drawback of particle filters is that a very high number of particles is needed to approximate the probability density function.

In order to reduce the complexity of the problem, Deutscher and Reid, [1] proposed the Annealed Particle Filter (APF) as a combination of particle filtering with simulated annealing. Instead of trying to approximate the probability density function which is clearly impossible in high dimensional space, APF strategy consists on focusing particles around the modes of the pdf. For each image, a multi-layered search policy is applied to help particles escape local maximum. This is done by heavily smoothing the weight function in the earlier layers in order to produce unconstrained particles so that they are not stuck in a local maximum. One limitation of APF, is that the performance decreases at higher dimensions.

Another approach proposed by MacCormick and Isard [2] called Partitioned Sampling (PS) that consists on representing the human body in a factored manner by considering the torso at the root of a tree structure composed by the different body members. This representation rely on a conditional independences assumptions. This assumption allows to drastically reduce the number of required particles to estimate the pose. The idea is that, early evaluation of the hypothesis about the torso, particles are re-sampled around the most likely hypothesis for the torso before exploring the leg positions and thus avoid wasting exploration effort on unlikely areas of the state space. On the other hand, PS suffers from inaccurate initial pose estimation especially for the torso which is the root of the tree.

It has been shown in [8] that the use of a combination of PS and APF highly improves the tracking quality and clearly outperforms both APF and PS. In our work, we use the same combination of APF and PS. This combination consists on using a multi-layered search for each partition of PS.

1) Motion Model: Let

$$S(t)^m = \{{x_t}^{(i),(m)}, {w_t}^{(i),(m)}\}^{i:1..N},$$

be a set of N weighted particles ${x_t}^{(i)}$ at instant t at layer m, with $m = M, ..., 0$. For each layer, particles are iteratively re-sampled and propagated m times for the same observation function using the following equation:

$${x_t}^{(i),(m-1)} = {x_t}^{(i),(m)} + B^{(m)},$$

where $B^{(m)}$ is a multi-variate Gaussian random variable with variance P_m and mean 0. For each layer, P_m is reduced by the annealed factor as described in [1].

2) Observation Function: The likelihood function is built using depth images only. This depth image is obtained for each label found by the component labelling stage. In fact each label consists of a set of voxels. These voxels are back-projected to the depth image of the camera and the corresponding pixels sampled from the depth image are used as observation function for the particle filter. Figure 2(d) shows the final depth image obtained for each label. The likelihood function is calculated from the difference between the depth of the observed image I_R and the synthesized one I_S obtained by rendering the current pose hypothesis ${x_t}^i$. This can be formulated as follow:

$$\sigma = \sum_{p_i \in N} \left[f(d_s(p^i), d_r(p^i)) \right]^2,$$

with:

$$f(d_s(p^i), d_r(p^i)) = \begin{cases} f_1 & \text{if } p_i \in S_\cap \\ D & \text{if } p_i \notin S_\cap \end{cases}$$

with f_1 a function equal to:

$$f_1 = \begin{cases} \frac{(d_s(p^i) - d_r(p^i))}{D} & \text{if } |d_s(p^i) - d_r(p^i)| \leq D \\ 1.0 & \text{if } |d_s(p^i) - d_r(p^i)| > D \end{cases}$$

Where N is the set of all pixels and p_i is the current pixel. $d_s(p^i)$ and $d_r(p^i)$ are the depth of the *ith* pixel of I_S and I_R respectively. $S_\cap$ is area of intersection between the outer appearance of the projected body part and the silhouette of the real image. D is a normalization factor found empirically and set to 100. At the end, the weight of a particle is then calculated as follow: $w(x_t^{(i)}) \propto \exp(-\sigma)$.

D. Occlusion Handling

In this section, we discuss how our method addresses occlusions from scene objects.

The particle filter stage outputs at time step t the optimal model configuration (See Figure 2(e)) which is equal to the weighted average of all particles:

$$\chi_t = \sum_{i=1}^{N} (w(x_t^{(i)}) * x_t^{(i)}),$$

Where χ_t is a set of K body part denoted as: $\{{P_t}^{(i)}\}^{i:1..K}$, with $K = 10$ is the number of body parts (See Figure 6(a)).

Each body part ${P_t}^{(i)}$ has a configuration c_t that holds the relative position of its underlying cylinder in the kinematic tree. For each body part we define two states: Visible (V) and Hidden (H).

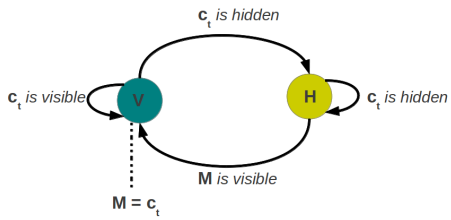


Fig. 7. The state machine used to identify if a body part is visible or not.

As shown in Figure 7, if ${P_t}^{(i)}$ is in a visible state (V), a memory M is used to save its current configuration c_t. ${P_t}^{(i)}$ becomes hidden if its configuration c_t is observed as hidden. In the hidden state, the last saved configuration in M is used to test if ${P_t}^{(i)}$ is visible again or not (The algorithm for detecting a body part is hidden or not will be detailed next). Finally, the visibility state for each body part is transmitted to its children in the kinematic tree. That is, if the left leg is hidden, left foreleg is also. If the torso is invisible the tracking stops. Next, Hidden parts found at time t are saved in a set ${S_h}^t$ that holds all hidden body parts. A new kinematic tree is generated from visible body parts only (that is from ${P_t}^{(i)} \notin {S_h}^t$) and sent to the particle filter. In this sense, our method only estimates the position for visible body parts.

The algorithm used to determine the visibility for each body part is as follow: At time step $t+1$, when a new image is received, the occupancy grid and labels positions are updated accordingly. χ_t found at time t, is moved to the the position of the center of mass of its corresponding label ${l_i}^{t+1}$. Each body part ${P_t}^{(i)}$ of χ_t represented by a cylinder is approximated by a set of n spheres as shown in Figure 6(b). Then each sphere, is projected to the occupancy grid to check if it lies within a hidden or visible cell. If more than the half of the n spheres are hidden, the body part is then considered hidden. The described method depends on the resolution of the grid, so in certain situation, it may identify visible parts as hidden. This is because the center of mass of a sphere may sometimes not project in the right cell (e.g, a person sitting on a chair, legs may be identified as hidden). We show in the experiments that this does not affect the tracking performance of our approach.

IV. Evaluation

In order to quantitatively evaluate the performance of our approach, we compared the 3D pose estimation of our system to ground truth data obtained from the Qualisys marker-based motion capture system that delivers accurate joint poses in 3D. Eight infra-red cameras are used to return the ground truth data. An RGB-D camera is synchronized and calibrated with respect to the Qualisys motion capture system. We build a rich dataset that consists of 12 sequences of 50 seconds each, for a person performing different actions (moving, sitting, walking, joking...) in a scene with obstacles. We put this dataset online with documentation on the following link [3]. Figure 8 shows some images from the dataset.

We run our experiments on a decent i7 intel processor quad core with a Nvidia Quadro K2000M programmable GPU. The first two stages are implemented on CPU and accelerated using Intel threading blocks. The particle filter stage is fully GPU accelerated using OpenGL shading language. The resolution of the occupancy grid is set to 5 cm. For the HMM, we used the following values for the transition matrix.

$$\alpha = 0.01, \beta = 0.1, \gamma = 0.4, \psi = e^{-38}.$$

For the particle filter, we use a combined particle filter as described in III-C with 200 particles and 5 annealed layers. We compare our method to APF with 200 particles and 10 annealed layers and to PS with 2000 particles. These values were chosen carefully by running each method with different number of particles on a sequence without any scene object occlusion (Seq12 in Table I). For each method, we chose the lowest number of particles and layers that give an average error close to 10 cm (See next paragraph). For the three methods, the particle filter is automatically initialized from the 3D position of the center of mass of its corresponding label.

For each sequence of T frames we compute the average performance and the standard deviation for each of the three methods using the following equations:

$$\mu = \frac{1}{T}\sum_{t=1}^{T}(D(X_t, \hat{X}_t), \sigma = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(D(X_t, \hat{X}_t) - \mu)^2},$$

where $D(X_t, \hat{X}_t)$ is the average absolute distance between the markers position $X_t = \{x_1, x_2, ...x_m\}^t$ and the estimated maker pose for each method $\hat{X}_t = \{\hat{x}_1, \hat{x}_2, ...\hat{x}_m\}^t$ equal to:

$$D(X_t, \hat{X}_t) = \sum_{m=1}^{M}\frac{\|x_m - \hat{x}_m\|}{M},$$

where M is the number of joints, $x_m \in \mathbb{R}^3$ is the marker position for joint m and $\hat{x}_m \in \mathbb{R}^3$ is the joint pose estimated by each method. Additionally, for each method, we measured the percentage of time the error for all joints $D(X_t, \hat{X}_t)$ does not exceeds 10 cm, 15 cm and 20 cm. We also measured the percentage of time the error exceeds 20 cm. Table I reports the results for the three methods for all sequences in the dataset where the best results are marked in bold.

Please note that the results displayed in the table are the average of three runs for each method.

A. Tracking performance

The results in Table I show that our method highly increase tracking stability for all sequences and outperforms APF and PS. In fact, for all sequences expect for sequence 3, our method reports the lowest average error and standard deviation. For sequence 3, while the average error of APF (15.15 cm) is slightly better than our method (16.13 cm), the percentage of time the error was less than 10 and 15 cm in our method is higher than APF. Sequence 1 and 10, are

the only sequences where the subject is sitting on a couch with a table in front of him, in these sequences, APF and PS have a very high average error and very low percentage of error less than 20 cm, while in contrast, our method clearly outperforms these two methods and reports a lower average error and higher percentage of error less than 20 cm on these two sequences. For the percentage of error higher than 20 cm (the sixth column in the table) , our method gives the lowest percentage that does not exceeds 24% on all sequences, and on sequence 11 and 12 it is less than 3%. For the percentage of error less than 20 cm, our method reports a percentage always higher than 76% for all sequences and outperforms APF and PS. For the percentage of error less than 15 cm, our method also reports the best percentage over APF and PS expect for sequence 6 where the percentage is very close to PS. As mentioned before, Sequence 12 is the only sequence that does not contain scene object occlusion, where we can see that the three methods gives very good results. While our method reports the best percentage for the error less than 10 cm. This percentage is still low for the three methods which is not surprising because the 3D model used for pose estimation is composed of cylinders which has an inaccurate outer appearance to approximate the real shape of a human.

A last conclusion that we can extract from this table is that PS gives the worst results which is expected because PS performs badly in occluded situation. APF gives clearly better results than PS. Our method that uses a combination of APF and PS with occlusion handling reports the best results on all sequences and visually succeeds on all the dataset. This is mentioned in the last column of the table. The results of this column are obtained by visually looking at the videos to check whenever legs are inverted, or if torso or legs are misplaced. APF completely lost the tracking on sequence 1, 4 and 10 after the first occlusion, while PS failed on sequence 1 to 10 also after the first occlusion. This tracking failure is explained by the fact that when the subject is in an occlusion situation, APF and PS continue to generate inconsistent hypotheses, so that when the subject is again visible, the tracking is generally lost(see Figure 1). The fact that APF gives better results than PS is explained by the fact that the use of multi layered search helps APF recovers in some situations after occlusion but not always. More than 10 annealed layers are required to improve APF performance. While in contrast our method that uses less annealed layers than APF succeeds on all sequences.

sequence	Error Approach	<10cm [%]	<15cm [%]	<20cm [%]	>20cm [%]	Avg Err [cm]	Std Dev [cm]	Status
Seq1	APF	4.95	9.41	11.72	88.28	37.40	15.05	FAIL
	PS	20.13	42.41	53.47	46.53	24.38	15.65	FAIL
	OUR	**28.22**	**58.25**	**76.07**	**23.93**	**17.25**	**10.86**	
Seq2	APF	9.64	42.02	76.15	23.85	17.17	6.63	
	PS	7.90	27.01	49.92	50.08	25.92	15.86	FAIL
	OUR	**12.01**	**49.13**	**80.41**	**19.59**	**16.45**	**6.17**	
Seq3	APF	30.23	60.29	**77.97**	**22.03**	**15.15**	**7.66**	
	PS	8.84	26.53	42.93	57.07	26.07	13.16	FAIL
	OUR	**36.17**	**60.93**	76.53	23.47	16.13	9.96	
Seq4	APF	22.63	50.29	65.96	34.04	18.19	9.70	FAIL
	PS	3.68	9.86	10.44	89.56	39.67	21.97	FAIL
	OUR	**29.79**	**60.54**	**81.82**	**18.18**	**14.24**	**6.04**	
Seq5	APF	14.70	50.00	**86.82**	**13.18**	14.85	**4.53**	
	PS	13.18	45.27	82.77	17.23	15.63	4.96	FAIL
	OUR	**19.59**	**51.52**	86.32	13.68	**14.77**	4.91	
Seq6	APF	22.40	51.91	83.16	16.84	15.53	6.54	
	PS	22.05	**53.30**	**84.20**	**15.80**	15.28	**6.11**	FAIL
	OUR	**28.20**	52.08	83.74	16.26	**15.05**	6.48	
Seq7	APF	12.94	50.81	78.32	21.68	16.12	6.18	
	PS	5.50	21.84	49.03	50.97	22.27	8.77	FAIL
	OUR	**23.46**	**53.88**	**80.91**	**19.09**	**14.85**	**5.17**	
Seq8	APF	23.76	83.15	87.00	13.00	14.43	9.89	
	PS	25.36	75.92	77.21	22.79	16.96	11.97	FAIL
	OUR	**42.22**	**86.20**	**88.60**	**11.40**	**12.20**	**5.88**	
Seq9	APF	21.32	57.69	67.27	32.73	21.27	16.48	
	PS	16.53	52.07	64.46	35.54	22.87	15.73	FAIL
	OUR	**27.27**	**65.29**	**77.85**	**22.15**	**17.28**	**12.28**	
Seq10	APF	13.43	22.16	34.72	65.28	30.30	16.14	FAIL
	PS	3.7	9.41	14.66	85.34	35.48	13.57	FAIL
	OUR	**39.35**	**69.75**	**82.72**	**17.28**	**15.28**	**10.61**	
Seq11	APF	47.62	80.30	87.68	12.32	12.57	6.96	
	PS	46.96	85.71	95.73	4.27	11.15	3.96	
	OUR	**52.22**	**86.37**	**97.87**	**2.13**	**10.85**	**3.68**	
Seq12	APF	36.73	84.26	96.55	3.45	11.61	3.56	
	PS	31.48	83.21	94.30	5.70	12.01	3.63	
	OURS	**45.28**	**87.71**	**98.50**	**1.50**	**10.87**	**3.14**	

TABLE I

PERFORMANCE COMPARISON BETWEEN OUR METHOD, APF AND PS.

B. Tracking Speed

Our implementation takes advantage of CPU and GPU power. The first two stages are accelerated on CPU using Intel threading blocks, while the particle filter stage runs on another thread and uses the GPU to generate hypothesis and evaluate them. We provide a pipeline implementation, in fact, while the first two stages process the image at time t, the particle filter processes the frame at $t-1$. In that way we guarantee that both CPU and GPU are always busy. Our implementation runs at 4 frames per second with the above set-up (5 layers with 200 particles). Most of the time is spent in generating and evaluating hypotheses for the particle filter. In the current implementation, hypotheses are generated sequentially on GPU (using OpenGL frame buffer objects for offscreen rendering) and evaluated using GLSL Shaders. This can be accelerated using OpenGL instanced rendering extension that allows to render multiple hypotheses with a single draw call. We choosed OpenGL over CUDA, because with CUDA we still need OpenGL as rendering engine to generate particles, and switching context between CUDA and OpenGL is costly.

V. Conclusion

In this paper, we proposed a framework to improve tracking performance for existing particle filter methods (APF and PS). We show that the use of combination of PS and APF with the ability to detect and discard hidden body parts highly improve the tracking performance and stability. We also tackled the background initialization and update problem that frequently occurs in realistic conditions. For this, we proposed a method that uses a probabilistic occupancy grid updated with a HMM in order to maintain an up-to-date background and extract moving persons from the scene. We conducted a series of experiments on our own dataset that contains ground truth data and we showed the robustness of our approach to cope with severe occlusion situations with a single monocular depth camera. Compared to APF and PS, our method highly increases tracking stability and outperforms these two methods in challenging situations. The use of an occupancy grid in the first stage of the framework makes our framework scalable in the sense that it can easily integrates multiple depth sensors. Finally, We put our dataset

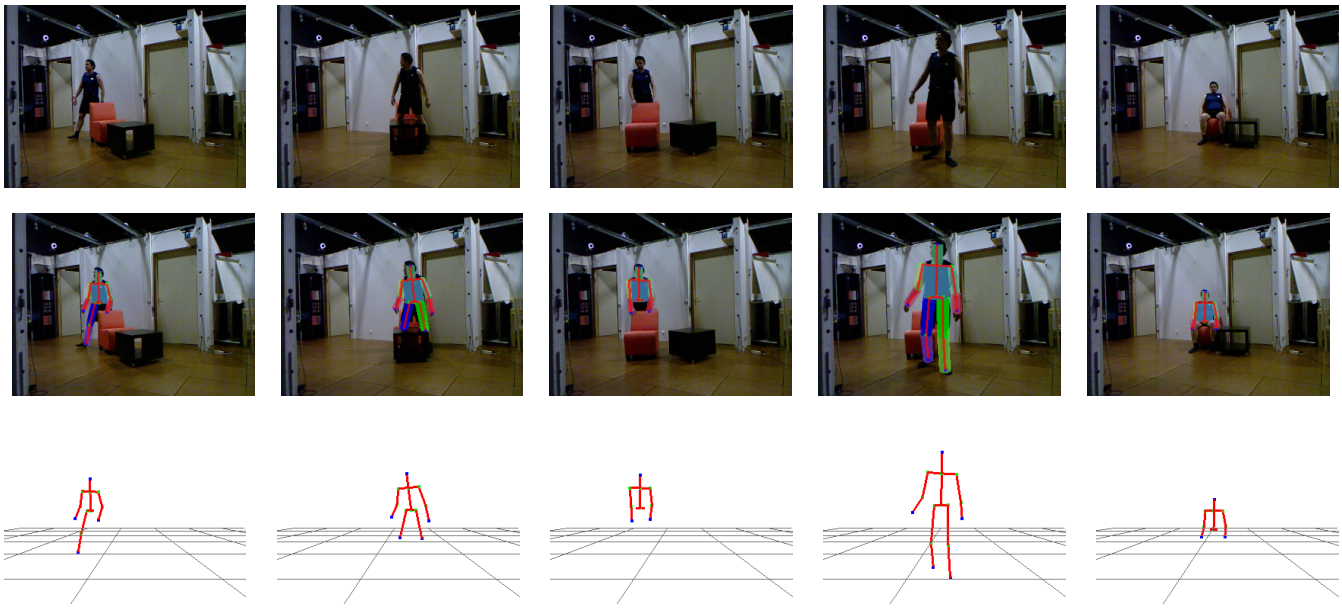


Fig. 8. First row: Screen shots from our dataset that shows a subject performing different type of actions in a scene with occlusions. Second and third row: Shows the estimated human pose for each image in the first row using our method. As shown in the figure, Our method successfully discards hidden parts from the pose estimation and outputs accurate estimation even in severe occlusions (i.e, a person behind a coach).

freely online so everyone can use it to evaluate his approach.

References

[1] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 126–133 vol.2.

[2] J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ser. ECCV '00. London, UK, UK: Springer-Verlag, 2000, pp. 3–19. [Online]. Available: http://dl.acm.org/citation.cfm?id=645314.649442

[3] "Synchronized rgbd and motion capture dataset," Feb. 2015, https://team.inria.fr/larsen/software.

[4] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Underst.*, vol. 104, no. 2, pp. 90–126, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2006.08.002

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

[6] P. Peursum, S. Venkatesh, and G. West, "Tracking-as-recognition for articulated full-body human motion analysis," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–8.

[7] M. W. Lee, I. Cohen, and S. K. Jung, "Particle filter with analytical inference for human body tracking," in *Workshop on Motion and Video Computing*, 2002, pp. –.

[8] J. Bandouch, F. Engstler, and M. Beetz, "Evaluation of Hierarchical Sampling Strategies in 3D Human Pose Estimation," in *Proceedings of the 19th British Machine Vision Conference (BMVC)*, 2008.

[9] M. W. Lee and R. Nevatia, "Human pose tracking in monocular sequence using multilevel structured models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 1, pp. 27–38, Jan 2009.

[10] T. Zhao and R. Nevatia, "Tracking multiple humans in complex situations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1208–1221, Sept 2004.

[11] S. Gammeter, A. Ess, T. Jggli, K. Schindler, B. Leibe, L. V. Gool, E. Zrich, and R. Aachen, "Gool, articulated multibody tracking under egomotion," in *in ECCV*, 2008.

[12] M. Andriluka and S. R. B. Schiele, "Monocular 3d pose estimation and tracking by detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[13] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *Int. J. Comput. Vision*, vol. 61, no. 1, pp. 55–79, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000042934.15159.49

[14] S. Amin, M. Andriluka, M. Rohrbach, and B. Schiele, "Multi-view pictorial structures for 3d human pose estimation," in *British Machine Vision Conference (BMVC)*, September 2013.

[15] C. Stoll, N. Hasler, J. Gall, H.-P. Seidel, and C. Theobalt, "Fast articulated motion tracking using a sums of gaussians body model," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 951–958. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126338

[16] Y. Liu, J. Gall, C. Stoll, Q. Dai, H.-P. Seidel, and C. Theobalt, "Markerless motion capture of multiple characters using multiview image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2720–2735, 2013.

[17] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *CVPR*. IEEE, June 2011. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=145347

[18] J. Lallemand, O. Pauly, L. Schwarz, D. Tan, and S. Ilic, "Multi-task forest for human pose estimation in depth images," in *3D Vision - 3DV 2013, 2013 International Conference on*, June 2013, pp. 271–278.

[19] D. K. S. T. Varun Ganapathi, Christian Plagemann, "Real-time human pose tracking from range data," in *ECCV*, 2012.

[20] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 188:1–188:12, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366207

[21] L. Zhang, J. Sturm, D. Cremers, and D. Lee, "Real-time human motion tracking using multiple depth cameras," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[22] J. T. K. V. H.-P. S. C. T. Helge Rhodin, Kwang In Kim, "Interactive motion mapping for real-time character control," in *EUROGRAPHICS 2014*, 2014.

[23] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, June 1989. [Online]. Available: http://dx.doi.org/10.1109/2.30720