Skeleton body pose tracking from efficient three-dimensional motion estimation and volumetric reconstruction

Zheng, Zhang; Seah, Hock Soon

2012

Zhang, Z., & Seah, H. S. (2012). Skeleton body pose tracking from efficient three-dimensional motion estimation and volumetric reconstruction. *Applied Optics*, 51(23), 5686-5697.

<https://hdl.handle.net/10356/79884>

<https://doi.org/10.1364/AO.51.005686>

© 2012 Optical Society of America. This paper was published in *Applied Optics* and is made available as an electronic reprint (preprint) with permission of Optical Society of America. The paper can be found at the following official DOI: [<http://dx.doi.org/10.1364/AO.51.005686>]. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper is prohibited and is subject to penalties under law.

Downloaded on 26 Apr 2023 15:03:16 SGT

Skeleton body pose tracking from efficient three-dimensional motion estimation and volumetric reconstruction

Zheng Zhang* and Hock Soon Seah

School of Computer Engineering, Nanyang Technological University, Singapore

*Corresponding author: zhang_zheng@pmail.ntu.edu.sg

Received 1 June 2012; revised 13 July 2012; accepted 15 July 2012;
posted 16 July 2012 (Doc. ID 169753); published 7 August 2012

We address the problem of body pose tracking in a scenario of multiple camera setup with the aim of recovering body motion robustly and accurately. The tracking is performed on three-dimensional (3D) space using 3D data, including colored volume and 3D optical flow, which are reconstructed at each time step. We introduce strategies to compute multiple camera-based 3D optical flow and have attained efficient and robust 3D motion estimation. Body pose estimation starts with a prediction using 3D optical flow and then is changed to a lower-dimensional global optimization problem. Our method utilizes a voxel subject-specific body model, exploits multiple 3D image cues, and incorporates physical constraints into a stochastic particle-based search initialized from the deterministic prediction and stochastic sampling. It leads to a robust 3D pose tracker. Experiments on publicly available sequences show the robustness and accuracy of our approach. © 2012 Optical Society of America

OCIS codes: 100.4999, 110.4155, 110.4153.

1. Introduction

Estimation and tracking of three-dimensional (3D) articulated body motion from image sequences is an active and important problem in the domain of computer vision. It has wide potential applications, such as in human computer interaction, biomechanics, computer animation, surveillance, and sport analysis. Its task is to recover 3D skeleton body pose parameters that describe the movements of one or multiple human bodies over time. Typically, the targeted body movement is limited to the movement of large body parts (body trunk, head, and limbs), and ignores the movement of fingers, toes, or other body motions. Generally, at least 30 degrees of freedom (DOFs) are needed to describe the human movement with accuracy. Searching in such a high-dimensional state space is challenging and often computationally expensive. In addition, human body motion is often

highly complex and nonlinear time varying. Other factors, such as noisy image observations, varied body shapes, and clothing, add to the difficulties. To make the problem tractable, it is common to make assumptions, for example, about the camera setup or the subject's movement. In this work, we tackle the scenarios where multiple cameras are assumed available and fixed around a 3D volume space within which the subject performs movements.

A variety of model-based approaches have been proposed to estimate body pose from multiple camera images [1–6]. These approaches exploit an implicitly known parametric 3D body model, and estimate the body pose either in a deterministic optimization [2,6] or stochastic state estimation [1] framework. For the former, the best posture is found by aligning the body model to a set of extracted two-dimensional (2D) or 3D image observations, optimizing some form of model-image matching metrics in the parameter space via optimization strategies [2,4,6]. For the latter, the body pose is seen as evolving over time according to an underlying motion dynamic model,

1559-128X/12/235686-12\$15.00/0
© 2012 Optical Society of America

and the task is to maintain a time-evolving probability distribution of the posture state via inference on the probability model. Since the underlying probabilistic models are difficult to obtain, an approximate inference is necessary and it often involves search over the pose space to find the peaks of posterior or likelihood using sampling techniques [1,7–9].

The high dimensionality of the pose space makes it difficult to search all accurate DOFs simultaneously, requiring sophisticated search strategies that must pay attention to the following aspects mentioned above. (1) For a model-based search, it ought to well fit the shape of the subject since rough modeling may lead to large tracking errors. (2) The image observations always have highly nonlinear correlation with the body posture, and their extraction suffers from various noises. The use of a single image cue is vulnerable and may easily meet failures. (3) Because of ambiguities and nonlinearities, the parameter-space cost function is highly nonlinear and has multiple local optima, so some form of global search incorporated with constraints is required.

In this paper, we present a model-based approach and solve pose estimation in an optimization-based framework. Our pose search method considers each of the three aspects mentioned above. We use a voxel-based subject-specific body model (Section 2), which can provide sufficient support for accurate pose recovery. We exploit the idea of multicue integration, using a robust correspondence metric that combines volume, appearance, and scene flow (Section 3). The computing of scene flow is integrated with the volumetric reconstruction procedure, and uses background subtraction for tracking of good 2D feature points, resulting in efficient and robust 3D motion estimation. We propose to implement the scene flow computing on a GPU, which results in very efficient scene flow computation. We are not aware of any previous work that recovers the multicamera scene flow in such a fast manner and uses it for 3D body pose tracking.

Our model-based pose search procedure follows a strategy involving prediction and estimation (Section 4). The estimation of scene flow in our method avoids repeated 3D-to-2D projection computing, and, thus, is a procedure independent of the tracked body model. At the end, the recovered scene flow

represents the actual 3D motion of body shape points, which provides strong information for pose prediction. The scene-flow-based prediction produces a hypothesis of which partial parameters are well recovered, resulting in a lower-dimensional search space. A stochastic particle-based optimization algorithm is used for global optimization in the lower-dimensional state space. The search combines the result of pose prediction with the previous estimate to generate a set of hypotheses that balances the deterministic prediction and stochastic sampling, leading to a robust 3D pose tracker. Our method enforces physical constraints and joint limits into a robust matching metric, which fuses together 3D volume and color information. This makes our approach capable of dealing with complex and ambiguous situations.

Figure 1 gives an overview of our approach. The rest of this paper is organized as follows. In Section 2, we describe the proposed subject-specific body model, and then, in Section 3, we introduce the method of 3D data reconstruction. Section 4 describes the pose search and tracking approach. Finally, experiments are given in Section 5, which is followed by the conclusion in the last section.

2. Articulated Body Model

The 3D articulated body model of our approach is a subject-specific voxel body model that is created from the initial volume [10]. This body model consists of a kinematical skeleton and a shape model. It has 10 body parts connected by joints, which form five open kinematic chains with a total of 36 degrees of freedom to describe the human movements with high accuracy. In pose tracking, we want to explicitly compute the positions of model points with a given posture hypothesis. Each body part is assigned a local coordinate system. Let $\mathbf{q}$ be a point on a body part that is associated to n joints (e.g., as shown in Fig. 2, $n = 4$ where the point is on the right forearm) with posture parameters $(\phi, \theta_1, \dots, \theta_{n-1})$, where ϕ is the six global DOFs of the root, θ_i ($i = 1, \dots, n - 1$) denotes the rotations of joint i (we assume joint i is the parent of joint $i + 1$ in the kinematic chain). $\hat{\mathbf{q}}_i$ denotes homogeneous coordinates in the local coordinate system. We can compute its homogeneous coordinates $\hat{\mathbf{q}}_{iw}$ in the world coordinate system by

Fig. 1. (Color online) Approach overview.

Fig. 2. (Color online) Voxel-based subject-specific body model. (a) The body model comprises a skeleton model and a colored voxel-based shape model. (b) Another representation that we often use for more clear result visualization. (c) The model has five open kinematic chains and 36 degrees of freedom.

$$\hat{\mathbf{q}}_w = M(\phi) \sum_i^{n-1} M(\theta_{n-i}) \hat{\mathbf{q}}_i, \quad (1)$$

where $M(\phi)$ is the 4×4 homogeneous matrix associated with free motion of the root, and $M(\theta_i)$ ($i = 1, \dots, n-1$) denotes the 4×4 homogeneous matrix of joint i .

3. 3D Data Reconstruction

Our goal is to recover the full body skeleton motion of a performer with N_{cam} (e.g., $N_{\text{cam}} = 8$) synchronized and calibrated video cameras. Our method directly performs pose estimation and tracking in 3D space, by using 3D reconstruction data, such as textured volume and scene flow.

A. Colored Volumetric Reconstruction

A visual hull is a geometric entity created by shape-from-silhouette 3D reconstruction techniques [11]. It represents the maximum volume or best approximation that is consistent with a set of the object's silhouettes. For our goal, we choose to construct a visual hull in a volumetric framework [4,12,13]. Compared to polyhedral-based methods [14–16], the volumetric approaches are relatively robust since they do not rely on the silhouette contours. In addition, the accuracy of the reconstruction can be easily limited to the desired accuracy. The use of dedicated hardware or software acceleration techniques has produced real-time reconstruction [13,17].

Our volumetric reconstruction starts by dividing the 3D space of interest into a total of $N \times N \times N$ voxels: $V = \{v_i | i = 1, \dots, N^3\}$, $v_i \in \{0, 1\}$, where N denotes voxel resolution; all voxels $v_i = 0$ represent the 3D object and the others $v_i = 1$ represent background. Initially, V is set with all $v_i = 0$. In contrast to the method of [13], where occupancy evaluation of voxel is performed by using a sparse pixel occupancy test, we project the center of each voxel to the silhouette images, and evaluate voxel occupancy by performing only a point-silhouette-intersection test for the N_{cam} projection pixels. We find the reconstruction using this coarse testing strategy is sufficient for our pose estimation task. The projection on every camera

plane of each voxel center point is precomputed and stored in a lookup table for computational speeding up. As the surface voxels are sufficient for the pose estimation application, inside voxels are removed to save much CPU rendering time. For the task of appearance modeling, we assign a color to each surface voxel, by mixing the colors of all projection pixels where the corresponding camera can see the voxel. This requires determining the set of cameras $C_{\text{vis}}(v_i)$ that can see voxel v_i . For avoiding multiple passes and increasing efficiency, each pixel of every camera plane is given a depth buffer, which is used to store the depth of the potential nearest voxel. The finding of the nearest voxel is combined with the pass of a voxel's occupancy determination. With such depth buffer, the set $C_{\text{vis}}(v_i)$ can be quickly found together with the pass of removing inside voxels. The process of voxel coloring linearly traverses all surface voxels, and assigns v_i the average color of corresponding projection pixel on every camera $c_j \in C_{\text{vis}}(v_i)$ ($j = 1, \dots, |C_{\text{vis}}(v_i)|$).

B. 3D Scene Flow Estimation

1. Scene Flow

The motion of objects in a 3D space is an important characteristic of dynamic scenes. By analogy with optical flow, which is a 2D motion field in the image plane, the 3D motion field of a 3D scene is called *scene flow* [18]. Most existing methods for scene flow estimation combine stereo and motion in one framework, known as motion–stereo approaches, which result in a view-dependent description of scene flow, also called “disparity flow” in this case [19]. Not much work has been done to estimate scene flow from multiple camera setups. In the work of [18], an algorithm for recovering 3D scene flow from multiple camera views is presented. Another recent approach solves the problem directly in the 3D scene domain, and to track 3D scene points or surface elements instead of 2D image points [20].

2D optical flow is commonly used as a feature in motion-based segmentation and tracking applications (e.g., [21]). Surprisingly, visual-motion research on tracking applications have rarely exploited scene flow information. The work of [22] combines scene flow with a silhouette-based body pose estimation method. The scene flow in this method is computed between the images of the next time step and the temporary images generated by rendering the body model estimated via a silhouette-based estimation. It does not describe the real 3D motion field of the subject's body. Moreover, the rendering of temporary images requires extra computational cost. It is reported that the computation of scene flow takes more than 45 s per frame. We utilize scene flows that represent the real 3D motion of the human body and introduce a method to compute it efficiently.

Scene flow can be recovered from multiple 2D optical flows by solving a linear system of equations [18]. Let $\mathbf{p}_i = (x_i, y_i, z_i)^t$ denote the 3D position of a surface voxel v_i and $\mathbf{u}_j = (u_j, v_j)^t$ denote its 2D image

projection on the j th camera image plane. If the camera is not moving and can see the voxel v_i , the differential relationships between $\mathbf{p}_i$ and $\mathbf{u}_j$ can be represented by a 2×3 Jacobian matrix $\mathbf{J}_{ji} = \frac{\partial \mathbf{u}_j}{\partial \mathbf{p}_i}$:

$$\frac{d\mathbf{u}_j}{dt} = \frac{\partial \mathbf{u}_j}{\partial \mathbf{p}_i} \frac{d\mathbf{p}_i}{dt}, \quad (2)$$

$\frac{d\mathbf{u}_j}{dt}$ is the optical flow of the projection point on the j th camera plane and $\frac{d\mathbf{p}_i}{dt}$ is the scene flow of the voxel. Equation (2) indicates that any optical flow $\frac{d\mathbf{u}_j}{dt}$ is the projection of the scene flow $\frac{d\mathbf{p}_i}{dt}$. Since one optical flow provides two linear constraints, if at least two optical flows of the multiple cameras are available, the scene flow can be recovered. Hence, assuming the voxel v_i is visible from n_i ($n_i \geq 2$) cameras, i.e., $|C_{vis}(v_i)| = n_i$, we can solve the $\frac{d\mathbf{p}_i}{dt}$ by finding the least-squares solution of a linear system of equations:

$$\mathbf{B} \frac{d\mathbf{p}_i}{dt} = \mathbf{U}. \quad (3)$$

2. Our Computation

In our framework, the computing of scene flow is integrated with the 3D visual hull reconstruction, thanks to the volumetric reconstruction method. We assume all the eight cameras are kept static and have been calibrated. Since the cameras are static and the position of every voxel is known, the mapping of each voxel v_i in the 3D space of interest, i.e., $v_i \in V$, between its 2D projection $\mathbf{u}_j$ is not changed. Hence, we can compute and store the Jacobian matrices $\mathbf{J}_{ji}$ for each voxel in a lookup table and only need to compute it once at startup. The visibility of a surface voxel (i.e., $C_{vis}(v_i)$) has already been computed in the process of voxel coloring. At the end, the matrix $\mathbf{B}$ for each surface voxel v_i is available. Computing the optical flows for each voxel in turn is computationally expensive and not necessary. Instead, we first determine the set of 2D feature points on every camera image and then compute the sparse optical flows for each image. Thus, we need to compute only the optical flow N_{cam} times. The 2D feature points on one camera image in fact are the projections of the visible surface voxels of this camera, and are available thanks to the process of voxel coloring. At each time step t , optical flow computing is performed on the segmented images with foreground at t and $t + 1$. The use of segmented foreground images restricts the finding of feature points' correspondences within the silhouettes, leading to robust motion estimation results. We adopt the pyramidal Lucas–Kanade (LK) algorithm [23]. After estimation of all optical flows, we solve the linear equation system [Eq. (3)] for each surface voxel v_i such that $|C_{vis}(v_i)| \geq 3$ by using the singular value decomposition (SVD) algorithm. Thus the scene flow computing

consists of two main steps: optical flow estimation and SVD computing. Owing to the use of foreground segmentation and a multicamera setup, the computation of scene flow in our approach is robust and reliable. Since the optical flow computation at one pixel is independent from the others, and the SVD computing is also individual-voxel dependent, the parallelization of the scene flow computation and the use of a GPU for acceleration are feasible. We have implemented the scene flow computation on a GPU and attained a speed of 10 frames per second (Subsection 5.A). To the best of our knowledge, our method is the first to compute the multicamera scene flow in such an efficient manner and use it for body pose tracking. Figure 3 gives one example of scene flow recovered by our method.

4. Pose Estimation and Tracking

A. Pose Prediction

We can evaluate the quality of scene flow by comparing the points formed by “flowing” the voxels of the last frame with the current voxel reconstruction. Specifically, let $V_t = \{v_i | i = 1, \dots, n_t\}$ denote the reconstructed voxel point set at time t , $\mathbf{p}_i$ be the position of voxel v_i , and $S_t = \{\mathbf{s}_i | i = 1, \dots, n_t\}$ denote the set of the estimated scene flow vector at t . Note that we assign a 0 estimate to those voxels where no valid scene flow is recovered to ensure that every voxel has a scene flow correspondence. We can obtain a flowed voxel set $V_t^* = \{v_i^* | i = 1, \dots, n_t\}$, where the position of each flowed voxel v_i^* is $\mathbf{p}_i^* = \mathbf{p}_i + \mathbf{s}_i$. We now define a *flow-reconstruction* error e_{fr} to measure the quality of scene flow estimation as

$$e_{fr} = \frac{1}{n_t^*} \sum_{\forall \mathbf{s}_i \neq 0} \|\mathbf{p}_i^* - \mathcal{F}(\mathbf{p}_i^*, V_{t+1})\|, \quad (4)$$

where n_t^* is the number of valid scene flow in S_t , and $\mathcal{F}(\mathbf{p}_i^*, V_{t+1})$ is a function that returns the position of the closest voxel in V_{t+1} for the flowed voxel v_i^* . The finding of closest voxel correspondence and the computing of Euler distance between the correspondence pair can utilize the 3D distance transform (DT) and

Fig. 3. (Color online) View-independent 3D scene flow of one frame in a “Wheel” sequence. For visualization, each scene flow is represented by one 3D arrow in blue (small motion) to red (large motion), of which the length and direction are the magnitude and direction of the scene flow vector, respectively. Green denotes that no valid scene flow is recovered at that voxel.

Fig. 4. (Color online) (a) Flow-reconstruction error of each frame on the “Wheel” sequence. (b) One frame example: the reconstructed voxel points are shown in green, and the points obtained by “flowing” these voxels at the last frame are shown in red.

closest feature transform (CFT) of the binary volume data [10]. Figure 4 plots the flow-reconstruction error of each frame for one multi-image sequence and gives an example of the flowed and reconstructed voxel sets. It is seen that the average flow-reconstruction error is less than 1.0 cm, which means that the 3D motion estimation in our framework is fairly robust and reliable and provides a good prediction of the body movement. This is why we can utilize scene flow estimation for pose prediction.

We assume that the body pose vector tracked at time t is $\mathbf{x}_t$ and the body model fits well with the voxel data V_t . The goal of pose prediction is to estimate a middle pose $\mathbf{x}_{t+1}^*$ such that the body model in the new pose would fit well with the flowed voxel set V_t^* . Let $\mathbf{q}_i^k(\mathbf{x}_t)$ denote an arbitrary model point on body part k ($k \in \{1, \dots, 10\}$); its closest voxel correspondence at time t is found as v_i^k , it is therewith associated with the scene flow $\mathbf{s}_i^k$, and we obtain an error f_i^k for each such correspondence: $f_i^k = (\mathbf{q}_i^k(\mathbf{x}_t^*) - \mathbf{q}_i^k(\mathbf{x}_t)) - \mathbf{s}_i^k$. The optimal pose $\mathbf{x}_{t+1}^*$ can be found by minimizing the sum of error from all body parts. Instead of optimizing all pose parameters at the same time, we can exploit a hierarchical optimization strategy to avoid high dimensional optimization. The whole body pose $\mathbf{x}$ is divided into seven subposes, $\mathbf{x} = (\theta_1, \dots, \theta_7)$, of which θ_1 covers the three global rotations and three global translations of the torso, θ_2 and θ_3 are the rotations of the chest and head joint, respectively, and θ_4 – θ_7 each covers the six DOFs of one limb. We obtain an objective function $F(\theta_j)$ for each subpose θ_j :

$$F(\theta_j) = \sum_{k \in B_j} \sum_i \|\mathbf{q}_i^k(\theta_j) - \mathbf{q}_i^k(\mathbf{x}_t) - \mathbf{s}_i^k\|^2, \quad (5)$$

where $B_j \subset \{1, \dots, 10\}$ denotes the relevant body parts of which the motion is affected by the subpose θ_j . For example, for the limb of left arm, B_j contains the body parts of the left upper arm and forearm. Note that for subposes θ_1 and θ_2 , only the torso is considered, as the torso can provide a sufficient set of correspondences. Because F is parameterized only

by θ_j , we can calculate the gradient $\nabla F(\theta_j)$ easily and use the Levenberg–Marquardt (LM) [24] algorithm to optimize it efficiently.

B. Search Space Reduction

The prediction using scene flow estimated at t gives an estimate of the pose at $t + 1$. Although the predictive result may not well match the reconstructed volume and appearance at the current time, often the pose parameters associated to the body parts that do not move fast are well recovered. To detect the body parts where the prediction is not sufficiently good, we compute an error E_k individually for each body part. The error is defined by

$$E_k = \frac{1}{n_k} \sum_i \|\mathbf{q}_i^k - \mathcal{F}(\mathbf{q}_i^k, V_{t+1})\|^2, \quad (6)$$

where $\{\mathbf{q}_i^k | i = 1, \dots, n_k\}$ is the set of all model points on the body part k . If the error of any body part k is less than a predefined threshold E_{thr} , the associated pose parameters estimated from the pose prediction is taken as the final estimate for time $t + 1$. Otherwise, the body part’s pose parameters are not sufficiently well estimated and need to be optimized in the procedure of global optimization (Subsection 4.C). In addition, if there are other body parts in the kinematic chain that are affected by the motion of this body part, the pose parameters of those preceding body parts also need to be optimized in the phase of global optimization.

C. Global Optimization

1. Objective Function

With the above evaluation and detection process, the high-dimensional search space $\mathbb{R}^d$ is changed to a lower one $\mathbb{R}^m$ by keeping the parameters of the detected body parts fixed. The goal is to find the optimal pose vector $\hat{\mathbf{x}}_{t+1}$ from the volume and appearance cues. To achieve this, we need to define an objective function $E(\hat{\mathbf{x}}_{t+1})$ that measures how accurately the body model in the pose configuration $\hat{\mathbf{x}}_{t+1}$ matches the volume and appearance data.

The first part of the objective function is measuring the fitting of the body model with the 3D volume shape V_{t+1} . Assuming $\{\mathbf{q}_i | i = 1, \dots, N_m\}$ denotes the set of all points on the body model, we find a voxel correspondence $\mathbf{p}_i$ for each $\mathbf{q}_i$: $\mathbf{p}_i \leftarrow \mathcal{F}(\mathbf{q}_i, V_{t+1})$, where $\mathcal{F}(\mathbf{q}_i, V_{t+1})$ is the closest correspondence function that returns the closest voxel in the set V_{t+1} to the point $\mathbf{q}_i$. The error E_v that describes the similarity of the body model and the volume is defined as

$$E_v(\hat{\mathbf{x}}_{t+1}) = \frac{1}{N_m} \sum_{i=1}^{N_m} \|\mathbf{q}_i(\hat{\mathbf{x}}_{t+1}) - \mathcal{F}(\mathbf{q}_i(\hat{\mathbf{x}}_{t+1}), V_{t+1})\|^2. \quad (7)$$

Minimization of the above error results in solutions that may not be physically feasible. We introduce a penalty value λ_i to each correspondence

for incorporating soft physical constraints. A higher penalty value is given for model points that lie in the same voxel bin with others or are associated to the same feature voxel with other model points. Assuming the sample point $\mathbf{q}_i$ lies in a voxel bin B and is associated to a closest feature voxel F_v , the value of λ_i is set according to the rules below:

$$\lambda_i = \begin{cases} \delta_\infty & \mathbf{q}_i \notin V_{\text{vol}}, \text{ else} \\ \delta_1 & B \rightarrow \text{others, else} \\ \delta_2 & F_v \rightarrow \text{others, else} \\ 1 & \end{cases} \quad (8)$$

where $\delta_\infty > \delta_1 > \delta_2 > 1$. In the rules, if the point $\mathbf{q}_i$ lies outside the volume space (i.e., $\mathbf{q}_i \notin V_{\text{vol}}$ and the volume space V_{vol} is the 3D cube space of interest where the subject is performing), a very large penalty value δ_∞ is used; if B has been taken by other sampling point(s) (i.e., $B \rightarrow \text{others}$), $\lambda_i = \delta_1$; else, if F_v has been associated to other sampling point(s) (i.e., $F_v \rightarrow \text{others}$), $\lambda_i = \delta_2$; otherwise, $\lambda_i = 1$. The optimal values of δ_1 and δ_2 are set through experiments. Equation (7) then becomes

$$\hat{E}_v(\hat{\mathbf{x}}_{t+1}) = \frac{1}{N_m} \sum_{i=1}^{N_m} \lambda_i \|\mathbf{q}_i(\hat{\mathbf{x}}_{t+1}) - \mathcal{F}(\mathbf{q}_i(\hat{\mathbf{x}}_{t+1}), V_{t+1})\|^2. \quad (9)$$

The second part of the objective function defines the appearance similarity between the body model and the color voxel data. Our voxel body model is comprised of color voxels, which define an appearance model for pose tracking. We define an error of appearance based on the similarity of the color distribution of the body model and the reconstructed voxels. The mixed colors assigned to one voxel involve two color channels, the H and S channel of the HSI color space. We adopt the histogram formulation to represent the density of the color distribution. For a body part b , assuming the corresponding closest feature voxel set for the set of body model point is found, for each color channel c , the two point sets separately correspond to a normalized K bin (here we set $K = 64$) histogram $q_c = \{q_{c,i}\}_{i=1,\dots,64}$ (with $\sum_i^K q_{c,i} = 1$) and $p_c(\mathbf{x}_t) = \{p_{c,i}\}_{i=1,\dots,64}$ (with $\sum_{i=1}^K p_{c,i} = 1$). By using the Bhattacharya distance to measure the histogram deviation, the error of appearance can be defined as

$$E_a(\hat{\mathbf{x}}_{t+1}) = \frac{1}{2} \sum_b w_b \sum_{c=1}^2 \left(1 - \sum_{i=1}^K \sqrt{p_{c,i} q_{c,i}(\hat{\mathbf{x}}_{t+1})} \right), \quad (10)$$

where w_b balances the influence of the different body parts.

The third part of the objective function is a smoothness term that penalizes high acceleration in the parameter space. It is defined as

$$E_s(\hat{\mathbf{x}}_{t+1}) = \left\| \frac{1}{2} (\mathbf{x}_{t-1} + \hat{\mathbf{x}}_{t+1}) - \mathbf{x}_t \right\|^2. \quad (11)$$

Combining the three terms of error using a weighted formulation, the total objective function is

$$E(\hat{\mathbf{x}}_{t+1}) = \hat{E}_v(\hat{\mathbf{x}}_{t+1}) + w_a E_a(\hat{\mathbf{x}}_{t+1}) + w_s E_s(\hat{\mathbf{x}}_{t+1}). \quad (12)$$

2. Hybrid Particle-Based Search

To minimize the objective function defined by Eq. (12), one strategy is to use local optimization starting from a pose initialized from a previous estimate, and perform an iterated closest point (ICP)-like optimization. However, when the previous estimate and the prediction are far away from the true pose, a global search is required to find the optimal solution and to recover from tracking failure. We exploit a particle-based stochastic search technique [25] for this global optimization.

The adopted particle-based stochastic search method is a hybrid algorithm derived from particle swarm optimization (PSO) and differential evolution (DE) [25]. The search ability of this algorithm comes from the interaction and the mutation between a swarm of N_{swa} particles evolved in an evolutionary computation framework. At the beginning, each individual particle is initialized with a random position $\mathbf{X}_i$ defined in the state space $\mathbf{X}_i \in \mathbb{R}^D$. Every particle has a self-best position $\mathbf{P}_i$ that it has met so far, and the entire swarm has a global-best position $\mathbf{P}_g$ defined as the best one among all $\mathbf{P}_i$. In every iteration, each particle's position is updated according to its own experience, the whole swarm's best position, and the differential mutation information. The equation of position update on each k th dimension is defined as

$$X_{i,k} = Q_{i,k}^* + N(0, 1) |P_{m,k} - P_{g,k}| + \alpha (X_{j,k} - X_{k,k}). \quad (13)$$

The first two terms on the right of the Eq. (13) represent a Gaussian sample [$N(0, 1)$ denotes a random generated from the standard Gaussian distribution] generated around a local attractor $\mathbf{Q}_i$ with a standard deviation $|\mathbf{P}_m - \mathbf{P}_g|$, where $\mathbf{Q}_i^*$ is the median between the local best and global best, and $\mathbf{P}_m$ is the mean of all particles' self-best. The use of all self-best information to compute the standard deviation gives a sampling covariance of global level. It enhances the particle diversity and the exploration ability of a particle. The third term, e.g., $\alpha (X_{j,k} - X_{k,k})$, in Eq. (13) is a mutation term borrowed from the DE algorithm. The DE mutation helps particles exchange information with each other, making it possible for exploring the state space more effectively. For particle i , two different particles j and k are randomly selected from the population such that i, j , and k are mutually different. α is a real factor in $[0, 1]$. A larger α value

means larger differential value, which enables particles to have a global exploration ability; while a smaller α allows particles to have a stronger exploitation ability.

3. Adaption to Pose Tracking

We adapted the algorithm to our optimization scheme with the aim of finding the optimal solution at an appropriate computational cost. In our problem, each particle individual represents a position $\hat{\mathbf{x}}$ in the state space $\mathbf{R}^m$. At the first frame, particles are normally sampled around the initial pose with a predefined diffusion covariance vector. With the convergence of the estimate at t ($t > 1$), particles are attracted to the area around the global optimal, resulting in the loss of particle diversity. Since the initial particles' positions and the particles' diversity largely affect the search ability of the algorithm, the particles need to be regenerated for each new frame. The initial particles' positions are generated from two pose hypotheses, the prediction from scene flow $\mathbf{x}_{t+1}^*$ and the previous pose estimate $\mathbf{x}_t$. This combines the information both from temporal propagation and prediction. Specifically, we sample ρN_{swa} particles around the prediction pose by a Gaussian distribution

$$\mathbf{X}_i \sim N(\mathbf{x}_{t+1}^*, \Sigma_{\text{pred}}), \quad (14)$$

and normally sample $(1 - \rho)N_{\text{swa}}$ particles around the previous estimate according to

$$\mathbf{X}_i \sim N(\mathbf{x}_t, \Sigma_{\text{prev}}). \quad (15)$$

Σ_{pred} and Σ_{prev} are the covariance matrices of the two Gaussian distributions; we set $\Sigma_{\text{pred}} = \kappa \Sigma_{\text{prev}}$, where κ is a constant factor usually set to 0.5. The diagonal elements of Σ_{prev} are set to the maximum absolute interframe angular or translation differences learned from the training data. The parameter ρ controls the number of particles sampled around the two pose hypotheses. We determine its value by considering the flow-reconstruction error, $\rho = 1 - \frac{e_{\text{fr}}}{e_{\text{const}}}$, where e_{const} is a predefined constant upper-bound error. In this way, lower flow-reconstruction error means higher quality of scene flow estimation, which leads to generating more particles around the prediction.

In the g th iteration of the search, we set the parameter α in the algorithm via an annealing form $\alpha = 0.7 - \frac{g}{3g_{\text{max}}}$, where g_{max} is the maximum iteration number. Using such a differential term with annealing allows particles to explore a wider area of the state space since it not only randomly samples around the median point $\mathbf{Q}^*$, but also exploits the information of random individual differences of the population. On each objective error evaluation, a subset of body model points is randomly chosen for the computing. The stochastic sampling of a point set has an advantage over the use of the whole point set or the deterministic sampling of a subset. It not

only enhances efficiency but also increases robustness, which also has been demonstrated by previous work [4].

D. Failure Detection and Recovery

Similar to all methods dealing with pose estimation in an optimization framework, our tracker may meet tracking failure, especially in situations with very complex and fast motions. Although we have combined the prediction with robust scene flow and low-dimensional global search using volume and appearance data, local minima may result in incorrect estimation or low accuracy in partial pose parameters, such as the rotations of one or two limbs, as shown in Fig. 5.

One typical failure happens when the limbs are very close to the torso. In this situation, the limbs may be easily attracted to touch or protrude into the torso and be trapped in an incorrect position, as shown in Fig. 5(a). Although we have incorporated constraints in the particle-based search, the self-intersection problem cannot be avoided completely. To cope with this failure, we perform collision detection for each limb. If one limb, for example, the right upper arm, is detected as protruding into the torso, the associated pose parameters (e.g., for the right upper arm, the associated parameters include the rotations of both right shoulder and right elbow) are reinitialized and reoptimized. For initialization, a larger sampling covariance is given and the previous several estimates are used for prediction, hoping that the tracker can climb out of the failure. Another typical failure that may be met is that the parameters of forelimbs are not of sufficient accuracy [Fig. 5(b)]. Most often, only one or two forelimbs are found with a relatively low accuracy estimation, but they do not intersect with other body parts. They can be detected in a similar way as that of detecting the limbs with a poor scene-flow-based prediction. After labeling these limbs, we use the LM local optimization to align them by minimizing the volume and appearance-based objective error. The optimization

Fig. 5. (Color online) Typical tracking failures. The reconstructed volume is shown by the green point cloud, and the body model is at the estimated posture configuration. Note that only the skeleton of the body model is shown for comparison. (a) The left forearm is protruding into the torso. (b) The pose estimation loses accuracy in the parameters of the left forearm.

Fig. 6. (Color online) (a) Volumetric reconstruction time for different voxel resolutions. The ratio of (b) invalid scene flows and (c) flow-reconstruction errors with voxel resolutions $N = 100$ and $N = 128$.

can be performed like an ICP iterative algorithm to ensure sufficient accuracy is reached.

5. Experimental Results

A. 3D Data Reconstruction

Voxel resolution is a critical factor of the entire process speed. We evaluate the algorithm on the public multicamera sequences [6], which are recorded with eight cameras at a resolution of 1004×1004 pixels at 40 frames per second. For efficient 3D data reconstruction and tracking, we scale these images to half-size. We found the half-size resolution (i.e., 502×502 pixels) is a good compromise between efficiency and accuracy. Figure 6 shows the timings of volumetric reconstruction on the “Wheel” sequence of [6] with voxel resolution ranges from 100 to 200. It is seen that, without estimating scene flow, the volumetric reconstruction can reach 15 frames per second at $N = 100$. The reconstruction speed, however, rapidly decreases to about 2 frames per second at $N = 200$. Voxel resolution also affects the performance of scene flow estimation. We test scene flow estimation on the 200-frame “Wheel” sequence, where we set the LK window size $w = 70$ and pyramidal level $l = 3$. The average numbers of surface voxels at $N = 100$ and $N = 128$ are about 2600 and 4400, respectively. Figure 6(b) plots the ratio of invalid scene flows for these two cases. It shows that the ratio of voxels having invalid scene flow estimation, i.e., 0 estimation, at $N = 128$ is much larger than that of $N = 100$. For the two cases, the average number of valid scene flows over the sequence is found almost equal, i.e., 2300 for $N = 100$ and 2500 for $N = 128$. This observation is consistent with the scene flow computation in our framework. The increase of voxel resolution leads to more surface voxels and smaller voxel size. This would certainly cause the increase of voxels that have their projections overlapping each other, causing more voxels that cannot be seen by the cameras. Without considering these 0 flows, the two cases have similar performance on scene flow reconstruction, as demonstrated by Fig. 6(c), which plots the flow-reconstruction error of each frame. In our experiments, we choose to set

$N = 100$ with each voxel 2.5 cm^3 . We found that this voxel resolution best balances efficiency and accuracy.

Although the computing of scene flow in our approach benefits from the procedure of volumetric reconstruction, it is not sufficiently efficient. For our experimental image data, we generally set the LK window size $w = 70$ and pyramidal level $l = 3$ (Fig. 7). The other relevant data and timings are shown in Table 1. It is seen that scene flow computing takes more than 2 s for one frame, and the most time-consuming step is the computing of optical flow. To increase the efficiency, we exploit graphics hardware for accelerating the computation of scene flow. Currently, GPUs are found on most personal computers and often exceed the capabilities of the CPU. The GPU provides a streaming, data-parallel arithmetic architecture and can allow remarkable performance gains when compared to the CPU for computationally intensive applications. We optimize and implement the computing of optical flow and SVD on a

Fig. 7. (Color online) Scene flow computation at different LK window sizes w with pyramidal level $l = 3$. (a) $w = 40$ and (b) $w = 70$. In this example frame, the motion of the left calf is very large. It is seen that setting $w = 40$ results in many noisy scene flows, while with $w = 70$ the reconstruction is much more robust.

Table 1. Algorithm Parameters and Timings of GPU and CPU Versions of Scene Flow Computation on the “Wheel” “Public Multi-Image Sequence [6]

Voxel resolution		100	
Image resolution		502 × 502	
Surface voxel number		2600	
Feature point number per image		2000	
LK pyramidal level		3	
LK window size (pixel)		50	70
Optical flow time (ms)	CPU	1455	2366
	GPU	71	93
SVD time for all voxels (ms)	CPU	139	140
	GPU	9	9

GPU by using compute unified device architecture (CUDA) [26], which results in speedup of more than 20 times compared with the computation on a CPU, as shown in Table 1. Because of limitation of space, the details of CUDA implementation are not given in this paper. The implementation of LK optical flow on a GPU can also be found in previous work [27]. The GPU-accelerated scene flow can run at frame rates of nearly 10 frames per second in our system. A demonstration of our efficient GPU-accelerated scene flow and volumetric reconstruction can be found at <http://youtu.be/mtR44XbexFY>.

B. Tracking Performance

We first tested our pose tracking algorithm on three of the multiple camera sequences of 6: “Dance,” “Wheel,” and “Handstand.” At initialization of each sequence, we acquire a subject-specific articulated body model by using our body model acquisition approach [see Section 2, or refer to [10] for detail]. The tracking may not start from the given first frame but from one where the reconstructed volume has few noise voxels and is suitable for body model acquisition.

Our pose tracking method mainly involves three steps: pose prediction from scene flow, global optimization in a lower-dimensional state space, and limb failure detection and recovery. The parameters in our method mainly include threshold E_{thr} (Subsection 4.B), constraint penalty constants δ_∞ , δ_1 and δ_2 [Eq. (8)], weights w_a and w_s [Eq. (12)], and the number of particles N_{swa} and maximum iterations g_{max} . In the majority of our experiments, we set a small value for E_{thr} , e.g., 0.8 cm, to ensure that only prediction results that are sufficiently reliable can be taken as the final estimate. The actual penalty constants we use are $\delta_\infty = 100$, $\delta_1 = 8$, and $\delta_2 = 5$, but we find the method is not overly sensitive to these parameters in the experiments. w_a and w_s balance the effect of the objective function terms in Eq. (12) and are set to $w_a = 0.4$ and $w_s = 0.02$ for most of our experiments. N_{swa} and g_{max} are two critical parameters we need to determine. It is found that search space reduction via the prediction stage usually leads to a lower-dimensional space of 9–21 DOF. For global optimization, we typically run the tracking with $N_{swa} = 50$ –100 particles and $g_{max} =$

Fig. 8. (Color online) (a) Reconstructed scene flow and the result of the tracker using scene flow. (b) Result obtained by hierarchical optimization without using scene flow information.

40–60 iterations. Thanks to the use of a 3D DT map, which significantly increases the efficiency of objective error evaluation, the average time of the global optimization is about 1.2–2.5 s per frame. Failure detection, in fact, can be regarded as a pose refinement stage, while the particles of the labeled body parts are rediffused and forced to converge again via the hybrid evolutionary search procedure. In most failure situations, only one or two forelimbs are not well aligned. The entire tracking speed of the system can reach near to 2 frames per second.

The prediction using scene flow is an important asset of our tracking method. We demonstrate its benefit by running the tracker with and without using scene flow. When scene flow is not used, the tracker uses the same stochastic optimization algorithm of the global optimization but in a hierarchical search framework, as done in our previous work [10]. We use seven hierarchical steps for the hierarchical search: torso→chest→head→four limbs. To ensure optimal estimation, suboptimization on the chest and head is performed with 25 particles and 30 iterations; others are given 30 particles and 40 iterations. The global optimization stage of the tracking with scene flow prediction is given 50 particles and 60 iterations. The results at one time instance are shown in Fig. 8. It is seen that the tracking without using scene flow loses accuracy at the frame where the legs are close to each other. The tracker with scene flow survives, because prediction from scene

Fig. 9. (Color online) Average joint position errors of the three methods on each sequence.

Fig. 10. (Color online) Example tracking results. Each sample gives the volume data (green), scene flow estimation, and the tracked body pose where only the skeleton of the body model is shown. Top two rows: “Dance” sequence, where every 40th frame between frame 196 and frame 440 are shown. Middle row: “Wheel” sequence (frames 70, 95, 110, 143, 166). Bottom row: “Handstand” sequence (frames 75, 90, 103, 127, 264, 390).

flow guides the sampling of particles, and reduces the chances of being trapped by local minima.

There is no ground truth data of the body pose for these eight-camera sequences. To quantitatively evaluate the reliability and accuracy of the tracking, we manually obtain the joint positions at each frame of every sequence by using a convenient user intervention tool with visual feedback. An error metric is defined to evaluate the quality of an estimated configuration. Assuming $\{m_i(\mathbf{x})|i = 1, \dots, M\}$ are the positions of targeted joints at the posture configuration of $\mathbf{x}$, the error between an estimated pose $\mathbf{x}^*$ and the ground truth pose $\mathbf{x}_0$ is defined by $\frac{1}{M} \sum_i^M \|m_i(\mathbf{x}^*) - m_i(\mathbf{x}_0)\|$. Figure 9 plots the average absolute joint errors in our tracking method, the hierarchical PSO method [10], and ICP local optimization algorithm. In the ICP algorithm, the pose estimation is performed iteratively where the estimation of each iteration is used to find the set of correspondences for the next iteration until the search converges to a local minimum. It is seen that our method clearly outperforms the local optimization (ICP) and the hierarchical PSO algorithm. Figure 10 gives some examples of the tracking results obtained by our method.

We also tested our method on the HumanEva-II dataset [28], which contains two sequences captured

by four cameras at a resolution of 656×490 pixels and 60 Hz. Ground truth data is available for these sequences. To extract silhouettes, we use a color-based background subtraction algorithm [29], which is shown capable of robustly removing shadows. The body model is created at the first frame. Figure 11 plots the first 50 frames’ relative joint position errors over the “S2” sequence, and gives one example of the tracked body model. Since the dataset is captured by only four cameras, the reconstructed volume is of

Fig. 11. (Color online) (a) Relative joint position error of HumanEva-II sequence S2 (frames 1–50). (b) One example estimated scene flow and the tracked result.

poor quality. This results in a body model having many erroneous voxels. Moreover, the camera configuration of the HumanEva-II dataset cannot carve erroneous voxels and may often generate a strangely shaped reconstruction. These factors significantly affect tracking accuracy and robustness. As any other volume based pose recovery method, the improvement of background subtraction quality and the increase of camera number would lead to much better tracking performance.

6. Conclusion

We address the problem of recovering 3D articulated body pose in the setting of multiple cameras. Our final goal is to build a markerless tracking framework that can track body movements as accurately and efficiently as the marker-based commercial motion capture systems. In this work, we present an approach that performs pose tracking directly in 3D space on the sequences of reconstructed volume and the 3D motion field. It is found that 2D optical flow is widely used in the application of object tracking. However, rarely has work utilized 3D scene flow for the pose tracking task. We propose computing scene flow by integrating it with the 3D volumetric reconstruction and accelerate it by a GPU. To this end, we have obtained fast and robust view-independent scene flow estimation, which we have not found in any previous work. To attain robust and accurate tracking performance, a prediction hypothesis is first computed from reliable scene flow estimation, and then global optimization is carried out to find the optimal pose in a lower-dimensional state space. We adapt a stochastic particle-based search algorithm into the global optimization, and introduce a robust metric that exploits multiple 3D cues and physical constraints. A method of tracking failure detection and recovery is introduced in order to make the tracker more robust and reliable. We have shown the effectiveness of our approach by testing it on several multiple camera sequences. For future work, we may analyze the effect of voxel noise and enhance tracking robustness in order to make it adaptable to general capture environment.

References

1. J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2000), Vol. 2, pp. 126–133.
2. C. Theobalt, J. Carranza, M. A. Magnor, and H. P. Seidel, "Enhancing silhouette-based human motion capture with 3D motion fields," in *IEEE Pacific Conference on Computer Graphics and Applications* (IEEE, 2003), pp. 185–193.
3. I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, "Human body model acquisition and tracking using voxel data," *Int. J. Comput. Vis.* **53**, 199–223 (2003).
4. R. Kehl, M. Bray, and L. V. Gool, "Full body tracking from multiple views using stochastic sampling," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2005), Vol. 2, pp. 129–136.
5. L. Mundermann, S. Corazza, and T. P. Andriacchi, "Accurately measuring human movement using articulated ICP with soft-joint constraints and a repository of articulated models," in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2007), pp. 1–6.
6. J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1746–1753.
7. J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *European Conference on Computer Vision*, Vol. 1843 of Lecture Notes in Computer Science (Springer, 2000), pp. 3–19.
8. C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Discriminative density propagation for 3D human motion estimation," in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2005), Vol. 1, pp. 390–397.
9. M. Bray, E. Koller-Meier, and L. V. Gool, "Smart particle filtering for high-dimensional tracking," *Comput. Vis. Image Understanding* **106**, 116–129 (2007).
10. Z. Zhang, H. S. Seah, C. K. Quah, and J. Sun, "A markerless motion capture system with automatic subject-specific body model acquisition and robust pose tracking from 3D data," in *IEEE International Conference on Image Processing* (IEEE, 2011), pp. 525–528.
11. A. Laurentini, "The visual hull: a new tool for contour-based image understanding," in *7th Scandinavian Conference on Image Analysis* (Springer, 1991), pp. 993–1002.
12. R. Szeliski, "Rapid octree construction from image sequences," *CVGIP Image Understanding* **58**, 23–32 (1993).
13. G. K. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, "A real time system for robust 3D voxel reconstruction of human motions," in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2000), Vol. 2, pp. 714–720.
14. W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (ACM, 2000), pp. 369–374.
15. W. Matusik, C. Buehler, and L. McMillan, "Polyhedral visual hulls for real-time rendering," in *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (Springer, 2001), pp. 115–126.
16. S. Lazebnik, E. Boyer, and J. Ponce, "On computing exact visual hull of solids bounded by smooth surfaces," in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2001), Vol. 1, pp. 1156–1161.
17. A. Ladikos, S. Benhimane, and N. Navab, "Efficient visual hull computation for real-time 3D reconstruction using CUDA," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (IEEE, 2008), pp. 1–8.
18. S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," *IEEE Trans. Pattern Anal. Machine Intell.* **27**, 475–480 (2005).
19. M. Gong and Y.-H. Yang, "Disparity flow estimation using orthogonal reliability-based dynamic programming," in *IEEE International Conference on Pattern Recognition* (IEEE, 2006), pp. 70–73.
20. R. L. Carceroni and K. N. Kutulakos, "Multi-view scene capture by surfel sampling: from video streams to non-rigid 3D motion, shape and reflectance," *Int. J. Comput. Vis.* **49**, 175–214 (2002).
21. T. Can, A. O. Karali, and T. Aytac, "Detection and tracking of sea-surface targets in infrared and visual band videos using the bag-of-features technique with scale-invariant feature transform," *Appl. Opt.* **50**, 6302–6312 (2011).
22. C. Theobalt, J. Carranza, M. Magnor, and H.-P. Seidel, "Combining 3D flow fields with silhouette-based human motion capture for immersive video," *Graph. Models* **22**, 540–547 (2004).
23. J.-Y. Bouguet, "Pyramidal implementation of the Lucas-Kanade feature tracker," *Microprocessor Research Labs Tech. Rep.* (2000).
24. K. Levenberg, "A method for the solution of certain nonlinear problems in least squares," *Q. Appl. Math.* **2**, 164–168 (1944).
25. Z. Zhang, H. S. Seah, and J. Sun, "A hybrid particle swarm optimization with cooperative method for multi-object

- tracking,” presented at the IEEE Congress on Evolutionary Computing, Brisbane, Australia, 10–15 June 2012.
26. NVIDIA, NVIDIA CUDA programming guide, <http://www.nvidia.com/object/cudahomenew.html>.
 27. J. Marzat, Y. Dumortier, and A. Ducrot, “Real-time dense and accurate parallel optical flow using CUDA,” 7th International Conference WSCG (2009).
 28. L. Sigal and M. J. Black, “HumanEva: synchronized video and motion capture dataset for evaluation of articulated human motion,” Tech. Rep. (Brown University, 2006).
 29. T. Horprasert, D. Harwood, and L. S. Davis, “A statistical approach for real-time robust background subtraction and shadow detection,” in *IEEE International Conference on Computer Vision* (IEEE, 1999), pp. 1–19.