CEIG - Spanish Computer Graphics Conference (2017)
F. J. Melero and N. Pelechano (Editors)

Physically Based Skeleton Tracking

Axel López-Gandía and A. Susín

Universitat Politècnica de Catalunya UPC-BarcelonaTech

Abstract
Skeleton tracking has multiple applications such as games, virtual reality, motion capture and more. One of the main challenges of pose detection is to be able to obtain the best possible quality with a cheap and easy-to-use device. In this work we propose a physically based method to detect errors and tracking issues which appear when we use low cost tracking devices such as Kinect. Therefore, we can correct the animation in order to obtain a smoother movement. We have implemented the Newton-Euler Algorithm, which allow us to compute the internal forces involved in a skeleton. In a common movement, forces are usually smooth without sudden variations. When the tracking yields poor results or invalid poses the internal forces become very large with a lot of variation. This allow us to detect when the tracking system fails and the animation needs to be inferred through different methods.

CCS Concepts
•***Computing methodologies*** → *Skeleton Tracking;* •***Hardware*** → *Sensors and actuators;*

1. Introduction

Skeleton tracking is an active area of research since it allows for a large number of applications such as gesture interaction schemes, virtual reality frameworks, motion capture and more. Currently there are many proposed solutions to this problem, although there is a trade-off between accuracy, expense and realtime capability. Current accurate body reconstructions typically involves a set up of a number of cameras and optical marker suits. The advantages of this approach are very precise motion tracking and no heavy suits are involved. The drawbacks of this system are an expensive and fixed scenario. Also this approach does not allow for a real-time applications since it is usually necessary to be post-processed. On the other hand, there is a very cheap solution which consists of using a single depth sensor and reduce the skeleton tracking problem to a pixel classification problem. One known device capturing human motion through depth sensor is Kinect-2 [Kin13]. This allows for a real-time and portable way to capture a skeleton motion, however it results in a low accuracy tracking. This method also handles very poorly when parts of the body are occluded to the camera. We focus on single depth sensor devices because of its versatility and we believe that the same tracking system can be improved without a significant computation load. Therefore, the objective of this work is to filter the skeleton capture by a single depth sensor using physical information. Skeletons are physical systems and given some parameters such as weight and height of the user, we are capable of extract dynamic information from an animation such as internal torques and forces. Our hypothesis is when significant occlusions occur, the skeleton provided by the tracking system will show large unrealistic internal forces. Consequently, we can use this information to filter an animation captured from these kind of systems. We will use a Kinect device in order to test our hypothesis and find a way to smooth the forces and provide a better estimation of the motion under occlusions circumstances. Below we introduce a small state of the art relating this topic.

1.1. Skeleton Tracking

Grest et al. [GKK07] used Iterative Closest Points (ICP) approaches to solve this problem. Using a single depth camera and a color camera, a human 3D model is iteratively moved in order to best match the information from the depth camera. Also they take advantage of color camera information to better determine boundaries of the body and therefore the background can be dynamic. The drawback from ICP methods is that they may get local minimum as result and yield an incorrect pose. Also they may be time expensive if there is a large amount of points to iterate.

Siddiqui and Medioni [SM10] proposed an optimization method in a data driven Markov Chain Monte Carlo framework. They define a likelihood function, to measure how well the input data matches the predicted pose, and used prior frame information to discard unfeasible poses. They improve over previous ICP techniques obtaining interactive framerate.

Shotton et al. [SFC*11] reformulated the pose tracking into a pixel classification problem. The input data is a segmented depth image and its pixels need to be assigned to a body part. To solve the problem they use a random forest approach which are trained with a large number of synthetic images. This approach is used in the kinect [Kin13] game sensor and yields good results in an

DOI: 10.2312/ceig.20171203

interactive framerate (30 fps due to the sensor framerate) but may fail to classify some pixels when large occlusions occur. The main advantage of this techniques is that it performs well for arbitrary backgrounds so no special set up is necessary other than the device within a certain range from the tracked body.

Wei et al. [WZC12] applied a MAP framework to estimate body pose. They defined several functions to quantize how much input data matches the predicted pose taking into account silhouette, occluded pixels, and the depth image. Also use prior poses in order to obtain a better estimation of the skeleton by penalizing changes in the velocity of the joints of the skeleton. This approach is capable of producing good results in real-time, and solves some poses where previous attempts such as kinect is not able to provide a valid result. Despite the advantages of this method it is GPU intensive which may not be suitable for some applications.

Our intention is to improve an existing fast skeleton tracking such as kinect and filter its results to provide a better estimation of the pose when occlusions occur or in situations where the sensor is not able to provide acceptable results.

1.2. Articulated Body Dynamics

The main part of this work is focused in skeleton dynamics. An extensive work in the fields of robot dynamics is has been conducted by Featherstone [Fea08] with many publications.

There is a number of algorithms regarding skeleton dynamics, our focus is to find the most efficient to implement, these have been proposed by Featherstone [FO00] which are able to compute forces or acceleration in $O(n)$ cost with the number of joints of a skeleton.

Although much of these works focuses on robot development, it can be easily applied to perform computer simulation in a graphics environment such as games which was suggested by Kokkevis [Kok04].

2. Skeleton Dynamics

In this section we will show the framework for skeleton dynamics. We will show the algebra used when dealing with this kind of problems, show the relevant quantities and the possible operations. Finally we will show two algorithms for computing forces and accelerations of an skeleton.

2.1. Skeleton

We define a skeleton as a set of joints and bones. Joints have a corresponding bone and may have multiple children and it holds always true $parent(i) < i$. Each joint has a single degree of freedom defined by an axis ω. Bones have a certain length and mass and they are approximated by cylinder of which the inertia tensor I_{COM} is defined in equation 1 where r is the radius of the cylindric bone, L its length and m its mass.

$$I_{COM} = diagMatrix(\frac{mr^2}{2}, \frac{m}{12}(3r^2+L^2), \frac{m}{12}(3r^2+L^2)) \qquad (1)$$

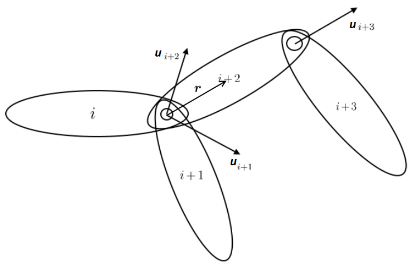

Figure 1: *Spatial Motion Axis. $\vec{r}$ is the vector connecting the center of mass of the bone with the axis of rotation and u is the axis of rotation.*

Although in this model joints have only a single degree of freedom (see fig. 1) multiples degrees of freedom can be achieved by concatenating more than one joint with a link of zero mass and length and having as their motion axis the desired directions.

2.2. Spatial Notation

Spatial Vector Algebra [Fea14] [Fea08] is convenient way of expressing rigid-body kinematics and dynamics. It conceptually reduces the amount of quantities and measures to be taken into account preventing mistakes or confusions.

There are two vector spaces being the first space for motion or kinematic vectors (velocities) and the second vector space where dynamic quantities (forces). These vector spaces are six-dimensional where lie both angular and linear quantities. For instance, the spatial velocity is written as $\hat{\boldsymbol{v}} = (\boldsymbol{\omega}, \boldsymbol{v})^T \in \mathbf{R}^6$ where $\boldsymbol{\omega}$ is the angular velocity of a body and $\boldsymbol{v}$ is its linear velocity. We remit to [Fea14] for the formulas and notation.

3. The Recursive Newton-Euler Algorithm

In order to obtain the underlying forces from a motion capture, we require an algorithm which, given the kinematic information (velocity and acceleration) and other known quantities (bone weights, lengths and external forces), computes the internal forces and torques in the articulated structure. This is the Newton-Euler Algorithm [FO00].

Algorithm 1 Newton-Euler Algorithm

```
for $i = 0$ to $N-1$ do
    $j \leftarrow parent(i)$
    $\hat{\boldsymbol{v}}_i = {}_iX_j\hat{\boldsymbol{v}}_j + \hat{\boldsymbol{s}}_i\dot{q}$
    $\hat{\boldsymbol{a}}_i = {}_iX_j\hat{\boldsymbol{a}}_j + \hat{\boldsymbol{v}}_i \times \hat{\boldsymbol{s}}_i + \hat{\boldsymbol{s}}_i\ddot{q}$
    $\hat{\boldsymbol{f}}_i^J = \hat{I}_i\hat{\boldsymbol{a}}_i + \hat{\boldsymbol{v}}_i \times^* \hat{I}_i\hat{\boldsymbol{v}}_i - \hat{\boldsymbol{f}}_i$
end for
for $i = N-1$ to $0$ do
    $\tau_i = \hat{\boldsymbol{s}}_i^T \hat{\boldsymbol{f}}_i^J$
    if $i \neq root$ then
        $j \leftarrow parent(i)$
        $\hat{\boldsymbol{f}}_j^J += {}_jX_i{}^* \hat{\boldsymbol{f}}_i^J$
    end if
end for
```

In Algorithm 1 subindex refers to the corresponding joint, f^J is the force transferred from a parent joint to its child joints, $\hat{\boldsymbol{f}}$ is the sum of external forces, τ is the internal torque in the joint, ${}_iX_j$ is transformation matrix from the frame linked to joint j to the frame linked to joint i and q, $\dot{q}$ and $\ddot{q}$ are angle, angular speed and angular acceleration of a joint respectively.

Algorithm 1 involves two loops, the first one traverse all the joints from the root to the child joints. It transfers velocity and acceleration in order to obtain this quantities in local coordinates for each joint. It computes also the forces acting in that joint.

The second loop traverses the joints from the child joints to the root and computes the torque of each joint as well as transfer the force to the parent joint.

4. The Articulated-Body Algorithm

In Section 3 we explained how to compute the underlying forces of the skeleton. In this section we will show the inverse process, given the internal and external forces compute the resulting acceleration in order to obtain a dynamic simulation of a skeleton. The algorithm is called The Articulated-Body Algorithm [FO00] [Kok04].

Algorithm 2 follows the same notation as Algorithm 1 and additionally $\hat{\boldsymbol{p}}$ is the bias forces and following the notation of Kokkevis E. [Kok04] $\hat{\boldsymbol{c}}$, $\hat{\boldsymbol{h}}$, d and u are temporal variables in order to prevent computing the same quantities multiple times. This algorithm is divided in three loops, two which traverse the chain from root to children and one from children to root.

The first loop is very similar to the first loop of algorithm 1 it only computes velocity and force acting on that joint in local coordinates. The second loop transfers the forces from children joints to its parents and also composes the inertia of the joints. The final loop computes the spatial accelerations as well as the angular acceleration of the joint.

5. Results

To be able to compute physical quantities of a skeleton we need to define a skeleton model. Real human bodies have a lot of bones and joints however we will focus in the most relevant ones. In Fig. 2 we show the base skeleton and its joints. We consider all joints to have two degrees of freedom except for the root joint which has three degrees of freedom.

Algorithm 2 Articulated-Body Algorithm

```
for $i = 0$ to $N-1$ do
    $j \leftarrow parent(i)$
    $\hat{\boldsymbol{v}}_i = {}_iX_j\hat{\boldsymbol{v}}_j + \hat{\boldsymbol{s}}_i\dot{q}$
    $\hat{\boldsymbol{p}} = \hat{\boldsymbol{v}}_i \times^* \hat{I}\hat{\boldsymbol{v}}_i + \hat{\boldsymbol{f}}_i$
    $\hat{\boldsymbol{c}}_i = \hat{\boldsymbol{v}}_i \times \hat{\boldsymbol{s}}_i\dot{q}$
end for
for $i = N-1$ to $0$ do
    for $k$ in $Childrens(i)$ do
        $\hat{I}_i += {}_kX_i{}^T(\hat{I}_k - \frac{\hat{I}_k\hat{\boldsymbol{s}}_k \otimes \hat{\boldsymbol{s}}_k\hat{I}_k}{d_k})_kX_i$
        $\hat{\boldsymbol{p}}_i += {}_iX_k(\hat{\boldsymbol{p}}_k + \hat{I}_k\hat{\boldsymbol{c}}_k + \hat{\boldsymbol{h}}_k\frac{u_k}{d_k})$
    end for
    $\hat{\boldsymbol{h}}_i = \hat{I}_i\hat{\boldsymbol{s}}_i$
    $d_i = \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{h}}_i$
    $u_i = \tau_i - \hat{\boldsymbol{h}}_i \cdot \hat{\boldsymbol{c}}_i - \hat{\boldsymbol{s}}_i \cdot \hat{\boldsymbol{p}}_i$
end for
for $i = 0$ to $N-1$ do
    $j \leftarrow parent(i)$
    $\ddot{q} = \frac{u_i - \hat{\boldsymbol{h}}_i \cdot {}_iX_j\hat{\boldsymbol{a}}_j}{d_i}$
    $\hat{\boldsymbol{a}}_i = {}_iX_j\hat{\boldsymbol{a}}_j + \hat{\boldsymbol{c}}_i + \hat{\boldsymbol{s}}_i\ddot{q}_i$
end for
```

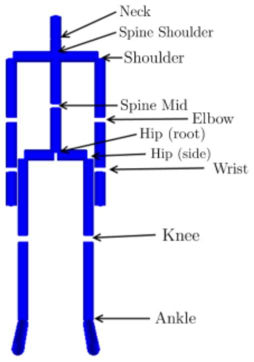

Figure 2: *Basic human skeleton with joint labels.*

This skeleton is a cut version of the one provided by the Kinect API. We ignore small bones such as thumb or fingers since these joints usually present heavy noise. We require additional information in order to compute forces and torques; mass, length and density needs to be fixed parameters through all the animation. From length and weight of the person, we use an anatomical standard model to represent the inertia of each of the body segments.

Kinect tracking needs to be able to track any kind of human body therefore, bone lengths are not fixed and may vary during the recording. Since our model needs to fix these quantities before starting any recording there is an initial inherent correction to the skeleton.

In order to correct the skeleton we need a way to detect errors in

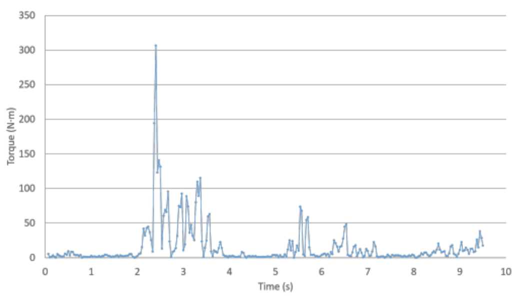

Figure 3: *Torque values for the left shoulder computed from Kinect's captured data.*

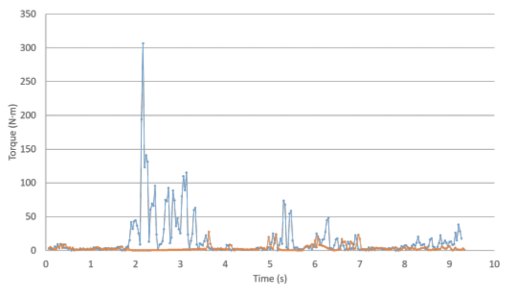

Figure 4: *Evolution of the torque magnitude of the Left Shoulder joint versus time without filtering (blue) and filtered (orange).*

the tracking system. The Kinect API outputs whether a joint is being properly detected or inferred however in our experimentation we found some situations where the joint was marked as tracked but in fact it was not. For this reason we will use the torque magnitude of the joints in order to detect whether a skeleton is properly tracked. In Fig. 3 we can see the torque values computed for the trajectory of one joint (the left shoulder in this case) and we can observe how these values are stable when the trajectory is properly tracked. When big oscillation on these values is observed, this is related to a bad behaviour in the tracking procedure.

$$\parallel \tau_{j,i} \parallel < \bar{\tau}_j + \alpha\sigma_j \tag{2}$$

A joint is consider to be a valid one, if the condition of Eq. 2 is fulfilled, where $\bar{\tau}_j$ is the average value of the torque magnitude of the animation to be processed, σ_j its standard deviation and α is a parameter to adjust the detection.

Once the wrong values of the torques are detected, the time interval is obtained and in order to correct the torque values, we use a cubic Hermite polynomial to interpolate those values. In the figure 4 we show the torque values interpolated in four different time intervals along the motion.

6. Conclusions

Although more work can be done in this direction, we have shown that restoration of the joint trajectories in a tracking system can be done using a dynamical approach.

Torques of the motion are a good feature to describe when a motion becomes wrong due to the the different limitations of a motion capture system. The great thing when using a dynamical simulator is that you can simulate the interval of time where you loose information.

Moreover, using recursive Newton-Euler and Articulated Body algorithms one can afford a real-time correction that we are not able to implement yet. This is one of our future work, together with the exploration of other significant dynamic features that can also contribute to recover good trajectories.

As another future work, we consider to also use the information from the RGB Kinect's camera and consider some other approaches like [MSS*17] that are able to do body tracking from color cameras alone.

7. Acknowlegments

Second author is partially supported by TIN2014-52211-C2-1-R.

References

[Fea14] Roy Featherstone 2014, Spatial Vectors and Rigid-Body Dynamics, (http://royfeatherstone.org/spatial)

[Fea08] Featherstone, Roy. Rigid Body Dynamics Algorithms. New York, Springer, 2008.

[FO00] Featherstone, R., and D. Orin. "Robot Dynamics: Equations and Algorithms." Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. (DOI: 10.1109/ROBOT.2000.844153)

[GKK07] Grest, D., Kruger, V., and Koch, R. 2007. Single view motion tracking by depth and silhouette information. In Proceedings of the 15th Scandinavian Conference on Image Analysis (SCIA), 719–729.

[Kin13] Kinect, 2013. (https://www.openkinect.org)

[Kok04] Kokkevis E. Practical physics for articulated characters. In Proceedings of Game Developers Conference 2004.

[MSS*17] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H-P Seidel, W. Xu, D. Casas, C. Theobalt. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. Procc SIGGRAPH 2017. (DOI: 10.1145/3072959.3073596)

[SM10] Siddiqui, M., and Medioni, G. Human pose estimation from a single view point, real-time range sensor. Proceed. CVPR-2010. (DOI: 10.1109/CVPRW.2010.5543618)

[SFC*11] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time human pose recognition in parts from a single depth image. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1297–1304.

[WZC12] Wei, X., Zhang, P., Chai, J. 2012. Accurate Realtime Full-body Motion Capture Using a Single Depth Camera. ACM Trans. Graph. 31, Article 188.

2

2