Project Report

Real Time Surface Modeling Using Body Pose Estimation

Group: Jiaqi Wang, Qiao Qiao, Yang Liu, Anja Kempf

1 Introduction

This project is aimed to develop a pipeline that uses Kinect sensor to first estimate the body pose as input, and then uses the body pose to control the movement of an virtual avatar. In addition to pose estimation, we also utilize the skeleton we get from the pose to roughly estimate the body size of the scanned person. In order to visualize the result of our pipeline in an entertaining way, we implement a mini dancing game in Unity3D.

2 Game Design

The game consists of two main parts, first the player stands in front of the Kinect camera, a pose will be estimated and we use the generated skeleton (from pose) to calculate the rough size of the player's body, and change the shape parameters on the virtual avatar accordingly.

The second part is the dancing game, we use 5 pre-defined poses as the targets, and they will appear in the game sequentially with certain time intervals in between. There is also an option provided for the user to record 5 new target poses in another menu. The player have to perform the corresponding pose and in the meantime the virtual avatar in the game will mirror the pose of the play. We then compare the performed pose and target pose by calculating the angles of each joint.

3 Implementation Details of the Pipeline

The general process of our pipeline can be described in Figure 1:
Kinect sensor will provide us the RGB-D output and the skeleton which is generated by the build-in pose estimation package. We extract this data, and apply the filters on the joint positions to make sure each one of them is temporal coherent, so the pose in each frame will give a more natural and smooth movement when we are tracking the skeleton in real time. The skeleton will be mapped to the virtual avatar, so that the avatar can mirror the movement of the player. We use the SMPL model as our virtual avatar, which provides shape parameters to change the body size as well as built-in skeleton associated with linear blend skinning for easy deformation and animation.

The above mentioned key elements in our pipeline that makes everything work smoothly will be discussed in the following sections.

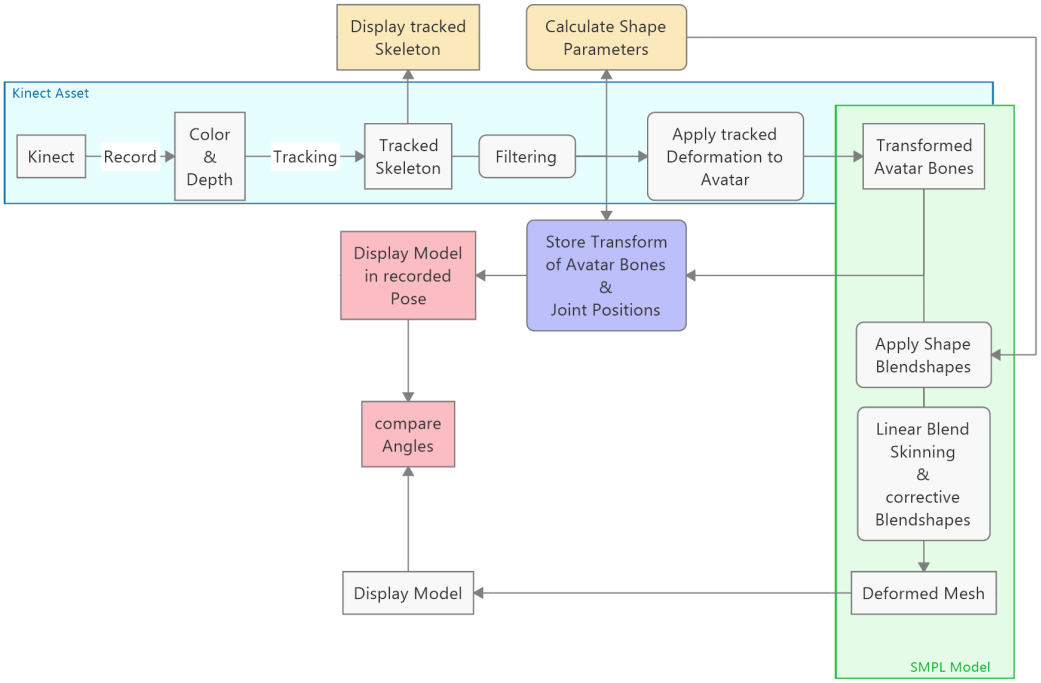


Figure 1: Method overview.

3.1 Packages

3.1.1 KinectSDK

Kinect SDK is a tool provided by Microsoft, together with the Kinect sensor, the KinectSDK extract RGB-D data and gives pose estimation out of the box, but we use filters to make the joints estimation more accurate, which will be dicussed in detail in section 3.3.

3.1.2 SMPL Model

SMPL, Skinned Multi-Person Linear Model, is a realistic 3D model of the human body that is based on skinning and blend shapes and is learned from thousands of 3D body scans [6]. The model can be downloaded from its official website. We use it in Unity to present our body movement.

3.2 Avatar Configuration and Control

3.2.1 KinectManager and AvatarController

KinectManager is the main and the most basic Kinect-related component. It controls the sensor and polls the data streams. [1] For example, joint position, joint rotation and

other useful information.

AvatarController gets joint positions and orientations from the KinectManager and then applies the transformations to the bones of the model's avatar. In Figure 2a you can see the corresponding joints.

3.2.2 SMPLBlendshapes

SMPLBlendshapes is responsible for adjusting the blendshapes according to the current pose. Additionaly it uses new set of shape parameters to adjust the shapeblendshapes.

3.2.3 Calculate Shape Parameters

Out of 10 parameters that are responsible for changing the shape of the SMPL model, We use only two of them, body height and body width, to achieve our goal of approximately reflecting the player's body shape onto the virtual avatar. In order to reshape the model in config-menu, keypoits for head and both feet of the person need to be tracked correctly. The body height is estimated by the torso length plus the average of the length of both legs. The joint positions can be obtained by the **KinectManager**. The torso length is calulated by the euclidean distance bewteen the head, shouldercenter, spine and hipcenter. In addition to that, we add 15 centimeters offset (nearly half of the head length) to the whole body height to compensate the fact that, the keypoint representing the head is tracked roughly in the middle of the player's head. The leg length is calulated by the euclidean distance bewteen the hip, knee, ankle and foot. The body width is approximated by the euclidean distance bewteen two elbows, see figure 2b. These values are then mapped to the $[-5, 5]$ range of the blendshape parameters.

3.2.4 Change Gender

For different genders, the bodyshape parameters for the SMPL model are different. We use two SMPL models in Unity to display, one for male and one for female. Users can change the gender of the model in the config-menu.

3.2.5 Record Pose

Target poses are required for the dancing game, in order to simplify the process of aquiring the target poses, we provide a recording function for it. This is done by storing the Kinect joint informations and the model's bone informations. To display each recorded target pose, the stored local bone rotations are applied to a new smpl model. Poses can only be recorded when all the joints are detected by Kinect, otherwise the scene will pop up a window to remind user to try a new pose.

3.3 Filtering

We explored different filtering methods that are commonly used in refinining the joint positions.

Exponential Smoothing is applied on time series data like the joint positions over time, we use is to assgin exponentianally decreasing weights for observations. In other words, joint positions that are captured by Kinect in early time points will have less weights, while newer data is assigned with more weighs. In this project we tried:
Single Exponential Smoothing which uses a weighted moving average:

$$S_t = \alpha p_{t-1} + (1-\alpha) S_{t-1}$$

where $S_t = \{\hat{x}_t, \hat{y}_t, \hat{z}_t\}$ denotes the smoothed value of observation (i.e. x,y,z position) at time t, $p_t = \{x_t, y_t, z_t\}$ stands for the original observation, and $\alpha \in [0, 1]$ is the smoothing constant.
Double Exponential Smoothing is another formulation of Exponential Smoothing, which is more reliable when applying it to data that shows a trend.

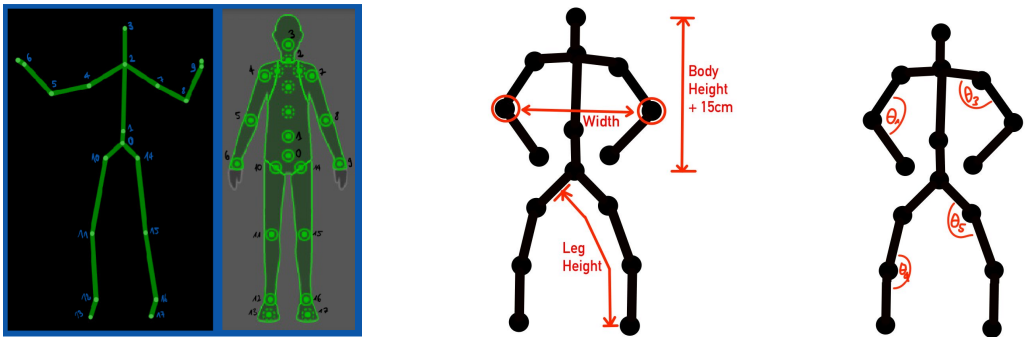
$$S_t = \alpha p_t + (1-\alpha)(S_{t-1} + bt - 1) \quad 0 \leq \alpha \leq 1$$

$$b_t = \gamma(S_t - S_{t-1}) + (1-\gamma) b_{t-1} \quad 0 \leq \gamma \leq 1$$

In our case, since joint positions in a time series shows the trend of the movement, this smoothing method makes more sense.

3.4 Compare Poses

We use the information of joint positions from kinect to compare the poses bewteen targetmodel and player. According to previous work [4], we calculate 8 important joint angles of the skeleton based on the joint postitions (fig. 2c) and compare the current player's pose with target pose. Only when all these 8 joint angles are under a certain predefined threshold (30 degrees), the player will then get a score and the scene will move to the next pose. The player have 100 seconds to achieve the goal for each target pose.



(a) Skeleton Mapping. (b) Shape Calculation (c) Angles

Figure 2

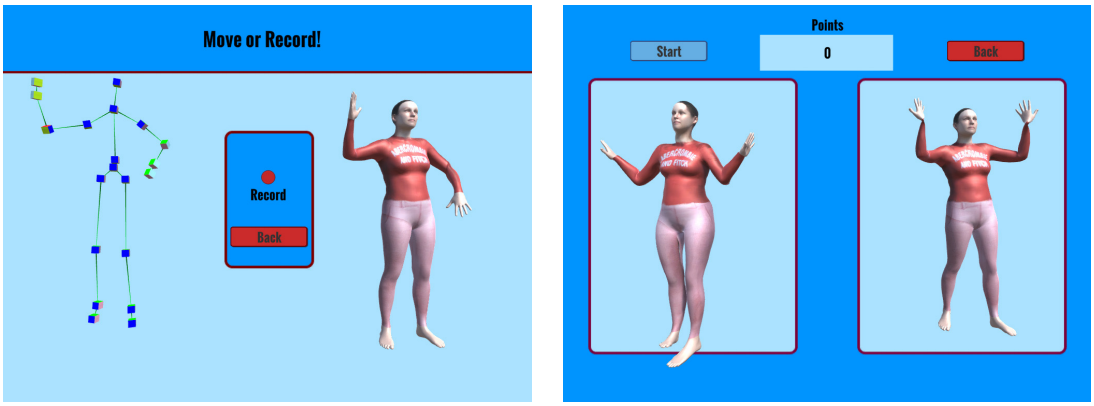
4 Results

4.1 Transfer Movement

In figure 3a we show that the avatar is quite a good match to the skeleton rig on the left. The movement transfer works well without any lacking, only when Kinect has some difficulties to detect some joints. After changing the shape paremeters, we see some artifacts on some joints of the model. These can be results of the linear blend skinning (e.g. Candy Wrap [5]).

4.2 Compare Angles

For most poses the matching is quite easy, but in some cases it's very hard to get under the defined threshold (30 degrees). Further analysis is necessary to find the right threshold for angle equality. And sometime the models seems to be very similar, but some of the angle differences are still greater than 30 degrees.



(a) DisplayScene (b) DanceScene

Figure 3

5 Further improvement in the future

There are still a lot of improvements can be doned in our project. First, instead of comparing poses, it would be more nature to compare a continuous series of poses in a period of time. How to compare poses at diffenrent time steps and to give it a corresponding score would be a problem to solve. Second, only calculating some joint angles is not good enough to compare poses accurately. For example, our elbow can turn when holding the same angle bewteen upper arm and lower arm. Third, by testing, it always happens that kinect cannot detect all the joints sometime so some joint information are missing. We need to wait or try carefully to let our joints being detected, which is sometime really annoying. So, an algorithm which can estimate missing joint position and rotation using

data at last time step or data of adjacent joints may help in this case. The last one is further optimizing the 3D model, eliminating strange distortions, applying better filters, smoothing the movement and so on.

6 Appendix

Hardware and Software specifications in this project:

- Kinect v1
- Unity3D, version: 2018.14.f1
- KinectSDK [3]
- SMPL Model [2]

References

[1] `https://ratemt.com/k2docs/KinectManager.html`. Accessed: 2019-12-22.

[2] `https://smpl.is.tue.mpg.de/`. Accessed: 2019-12-22.

[3] `https://www.assetstore.unity3d.com/en/?stay/content/115950#!/content/7747`. Accessed: 2019-12-22.

[4] Pradnya Krishnanath Borkar, Marilyn Mathew Pulinthitha, and Mrs. Ashwini Pansare. Match pose - a system for comparing poses. In *International Journal of Engineering Research & Technology (IJERT)*, 2019.

[5] Ladislav Kavan. Part i: Direct skinning methods and deformation primitives. In *SIGGRAPH Course 2014 Skinning: Real-time Shape Deformation*, 2014.

[6] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. 2015.