International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Real-Time Multiple Human Pose estimation For Animations in Game Engines

Albert Cleetus

Student, Dept. of Dual Degree Computer Applications, Sree Narayana Guru Institute of Science and Technology N.Paravur, Kerala, India

---

Abstract - *We propose a deep convolutional neural network for 3D human pose and camera estimation from monocular images and live camera that learns from 2D joint annotations. And this Paper proposes to build a multi-source deep model in order to extract non-linear representation from these different aspects of information sources. The collected images cover a wider variety of human activities than previous datasets including various recreational, occupational and house holding activities, and capture people from a wider range of viewpoints. Using Video cameras we will capture the live situation from the world space and using this technology it will give estimated data to the screen and we will collect these data at every frames per second, Using Data direct to Game development. This technology will shows how to use the real Human specific pose data to game development at the time of Game 3D-Modelling. For example, a very popular Deep Learning app "HomeCourt" uses Pose Estimation to analyses Basketball player movements. To implement this motion capture for animation in game development and videos, we would require specialized hardwares.Xsens is a company it will able capture the pose motions in the cameras.*

Key Words: **HomeCourt is an App, Xsens Technologies B.V. is a supplier of 3D motion capture products and inertial sensors based upon miniature MEMS inertial sensor technology.**

1. INTRODUCTION

The problem of human pose estimation, defined as the problem of localization of human joints, has enjoyed substantial attention in the computer vision community. One can see some of the challenges of this problem – strong articulations, small and barely visible joints, occlusions and the need to capture the context. 3Dhuman pose estimation has also started to attract a lot of interest from researchers, especially in the activity recognition and tracking communities. Many different kinds of approaches have been tried, with the most success being achieved largely with generative modelling approaches, such as 3D versions of pictorial structure, pose conditioned joint angle models, and approaches based on Gaussian processes. Although progress has been made, reliably estimating 3Dhuman pose from a single image remains an unsolved problem. While significant breakthroughs, the utility of these approaches are limited by the single-view nature of the source data, Approaches exist which perform 3D pose estimation on a single actor in a single image and video, but the range of 3D poses is heavily limited by the training dataset and limbs which are occluded must be inferred rather than directly measured to Powerful research in multi-person, multi-camera 3D pose estimation, This dataset presents an opportunity to face challenges not present in single image pose estimation.

Thus, the main objective of this thesis is developing a framework that would produce an animated character from the real video of human motion. The workflow of the project is expected to be so that the user would provide the system with a video of person performing some action. The developed framework will run one of three available 3D PE methods to define the position of the human in the 3D space. The exact method is defined by the user during the submission of the input video.

2. RELATED WORKS

Human pose estimation is one of the most studied problems in computer vision [5, 30, 4, 26, 12, 32, 31, 22, 13, 28, 29, and 15]. 2D pose estimation approaches range from part-based models, such as pictorial structure [10] and deformable parts models to deep neural networks [30]. In the last decade, 3D pose estimation has become as object of much interest, attracting a wide variety of techniques and approaches.

In existing pose estimation approaches, the pair-wise part deformation relationships are arranged in tree models [52, 54, 2, 40, 57], multi-tree model [55], or loopy models [56, 53, 10]. Tree models allow for efficient and exact inference but are insufficient in modeling the complex relationships among body parts.

Exploiting the success of 2D human pose estimation techniques, there has been a lot of work done in the area of 3Dpose estimation, as inferred from a single image. For example, uses a modified stacked hourglass to progressively produce 3D pose detections with more and more depth.

The closest work to ours uses convolution NNs together with Neighborhood Component Analysis to regress toward a point in an embedding representing pose. However, this work does not employ a cascade of networks. Cascades of DNN regressors have been used for localization, however of facial points.

3. EXISTING SYSTEM

Now we have a full pipeline for single person, multiple camera 3D pose estimation. After presenting our technique, we empirically show the effectiveness of our technique on the Campus and Shelf datasets, where we in some cases significantly out-perform the previous state of-art. Only a Technology for Single Person to estimate pose. Current pose estimation methods utilizing standard CNN architectures heavily rely on statistical post processing or predefined anchor poses for joint localization. Pose estimation incorporates contextual segmentation and joint localization to estimate the human pose in a single stage, with high accuracy, without relying on statistical post processing methods. While maintaining multi-scale fields-of-view comparable to spatial pyramid configurations. Now some methods typically estimating the pose of a single person in an image which only had one person to begin with. This technology first will find the definite single part of single person and from there find another to another parts, and then connect with other joints to form pose.

We can Use the application of pose estimation in human movements and action easily through this technology. From virtual sports coaches and AI-powered personal trainers to tracking movements on factory floors to ensure worker safety, pose estimation has the potential to create a new wave of automated tools designed to measure the precision of human movement. But these methods are now not useful in the real life Scenarios because many of situations they had Multiple persons situations.

4. PROPOSED SYSTEM

Multi-Person pose estimation is more difficult than the single person case as the location and the number of people in an image are unknown. Typically, we can tackle the above issue using one of two approaches:

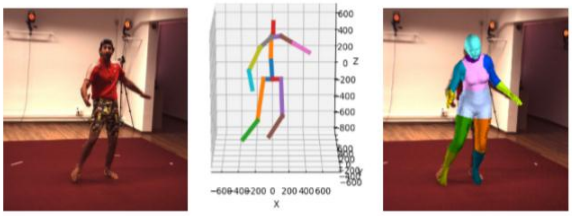
- The simple approach is to incorporate a person detector first, followed by estimating the parts and then calculating the pose for each person. This method is known as the top-down approach.
- Another approach is to detect all parts in the image (i.e. parts of every person), followed by associating/grouping parts belonging to distinct persons. This method is known as the bottom-up approach.

We will focus on multi-person human pose estimation using deep learning techniques. In the next section, we will review some of the popular top-down and bottom-up approaches for the same. And then Data used for animation in game development.

5. METHODOLOGY

Most of the 3D PE methods at some moment rely on the results of 2D PE methods so that first of all they estimate x and y coordinates of the key points [16, 32, 33, 34] or use pre-calculated data which is the output of the above-described 2D methods. This stage is very important because the final result is dependent on the accuracy of the 2D joint extraction method. As more precise are the joint coordinates as more trustworthy will be the 3D PE result. After obtaining the information about the 2D joints coordinates the methods "think" how to get 3D locations of the key points. However, there are also methods that go directly to 3D PE [36] without any intermediate calculations of 2D poses.

For the great number of 3D methods recovering the 3D pose, shown in Figure 1, is the main goal [16, 33, 34, 36, 37, 38, and 39]. It means that they aim to calculate the coordinates of each joint of the person in a 3D space. Sometimes the authors also try to visualize the obtained pose data by fitting it into the kinematic skeletons [16, 36].



(a) Input image (b) Predicted pose (c) Predicted shape

Fig -1: Results of 3D methods

A good starting point for this category is a method called Lifting from the Deep (LFTD) offered by Tome et. el. It proposes a new CNN architecture which is a combination of 2D joint location extraction and 3D pose predictions that are calculated simultaneously. It is done in this way for the better efficiency of the method. The offered framework is trained using data with 2D and 3D annotations. Furthermore, these two datasets can be independent of each other.

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper.

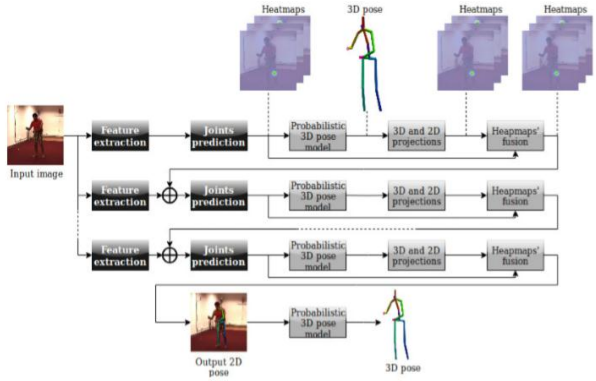


Fig -2: LFTD Architecture

5.1 Models and Representation

Given an image of a person, our goal is to estimate their 3D pose, as well as the parameters of the camera used to generate the image. We denote a pose by y = (y1,..., yM), wherein = (ym1,ym2,ym3)represents the 3D location of the MTh joint, with M = 14 (three joints for each limb, the neck, and the head). We use the perspective camera model and we assume that the camera is at the world origin, and is aligned with the world coordinate frame. Consequently, we only need to represent four intrinsic parameters: the focal lengths in each axis αu and αv, and the principal point coordinates u0 and v0. We use c to denote both the parameter vector (αu, αv, u0, v0) and the function,

$$\mathbf{c}(\mathbf{y}_m) = \frac{1}{y_{m3}} \begin{pmatrix} \alpha_u y_{m1} + u_0 y_{m3} \\ \alpha_v y_{m2} + v_0 y_{m3} \end{pmatrix}$$

Which projects 3D joint ym onto its 2D image location. Further, we let c(y) = (c(y1),...,c(yM)) represent the projection of a whole pose y onto the image, resulting in a 2M dimensional vector of 2D joint positions. We will also make use of the body parts (bones) of a pose, which we denote by wj, j = 1,..., 10, representing the eight upper and lower arms and legs, the neck, and the torso. Additionally, we define to be the length of bone j. Finally, we define a function l: R3M →RJ which computes the squared lengths of all body parts associated with y, i.e., l(y) = (l2 1,...,l2 10).

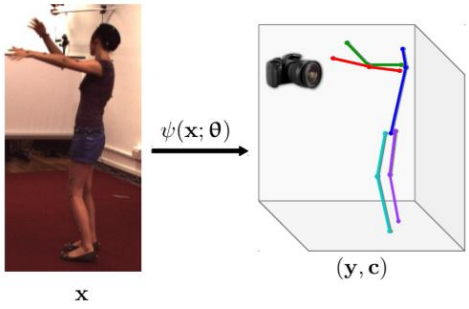


Fig -3: Representations

5.2 Regressions for 3D pose estimation

We wish to estimate the regression function ψ(x;θ) ∈R 3M+4 which maps an input image x to a 3D pose y and camera c, where θ denotes the model parameters. We train the model θ on labeled images D = {(x,u)}, where x is an image and u ∈ R2M is the ground-truth 2D pose, consisting of M joints, i.e., u = (u1,...,uM). Then, given a test image x∗, the predicted 3Dpose and camera is given by (y∗,c∗) = ψ(x∗; ˆ θ), where ˆ θ are the learned model parameters.

We use the deep learning framework, whereby ψ is a deep convolutional neural network (CNN) consisting of several layers, each representing different linear and nonlinear functions which are composed to give ψ. The first layer takes an input RGB image of a fixed size, and the last layer outputs the target values for the regression, in our case the 3D pose y and the camera c. Note that we train our network on images a notated with 2Djoints, but our output variables are in 3Dspace. This has two implications. First, we must project the 3D pose onto the image to compute the learning objective function, an operation which we encode in a network layer. More importantly, we must address the scale and orientation ambiguity inherent in estimating 3D information from images, i.e., there are an infinite number of 3D joints y which project onto the same 2D joint locations.

6. IMPLEMENTATION

The implementation can be divided into two main blocks: human PE and data mapping; and three auxiliary parts: human detection, cropping, and data normalization. The whole structure is presented below in Figure 4. Altogether, every block of the structure is equally important to the whole framework because if something will go wrong at any stage it easily can have a significant effect on the system outcome. The original input data to the framework is a video while the PE block accepts only images as an input. Thus, the video has to be divided into a sequence of frames. Typically, video frames have different width than height while most of the methods accept only images with equal dimensions that is why the images have to be cropped or reshaped as a square. Moreover, it is a common practice to crop the images before they will be processed in order to improve the result of the methods and its calculation speed.

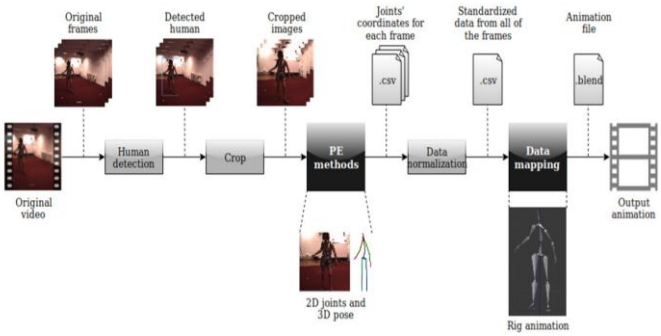


Fig -4: Pipeline implementation

Either a single pose or multiple poses can be estimated from an image. Each methodology has its own algorithm and set of parameters.

Fig -5: Single and Multiple pose estimation

In addition, to make the operation of associating the key point locations from the .csv files with empties easier and faster, it is better to merge all of the separate .csv files into one big file, which will contain individually the x, y and z coordinates of every joint for each frame in the video sequence. Another advantage of this approach is that in Blender each joint have a separate location channel as shown in Figure 6, thus such structure of the input PE data makes it easy to assign values for each parameter individually. After that normalized data is used for data mapping.

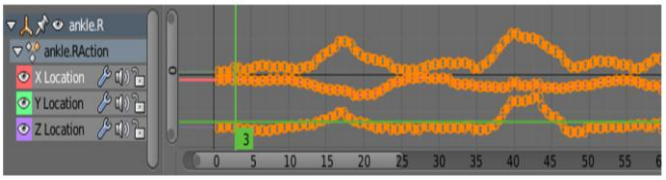


Fig -6: Separate x, y and z location channels of the right ankle.

To perform a proper mapping, every joint of the rig that is responsible for controlling it, has to be matched with the corresponding key point data from the .csv file but the amount the elements in those two sets of joints can be different. There by there is a need to filter out the extra joints. It can be done in two ways: before or after data import to Blender. Doing it before transferring the data will slightly shorten the calculation time because fewer data will go through the normalization process. In addition, Python script in Blender will be able to process the input from any PE method without a need to readjust the data assignment process for every method as it would be in the second case. That is why before being normalized and later mapped, a set of joints is filtered and ordered in a certain way. During the mapping stage, the data from a single .csv is matched to the joints of the character. This process results in the animation which can be imported to Unity together with the character for further development and used for animation and game purposes.

6.1 Data in 3D to the rig formation

In light of the above, to perform a proper mapping that will allow the created rig to follow the captured mocap data transformations, the combination of two rigs is used. This procedure helps to avoid aforesaid scaling problems which appear due to the fact that each of the tested methods provide the data at different scale. For instance, the data provided by one method assume that the model has to be 800 units high while the other method says it should be 1 unit. A great solution to that issue is to use double rigs for each animation.

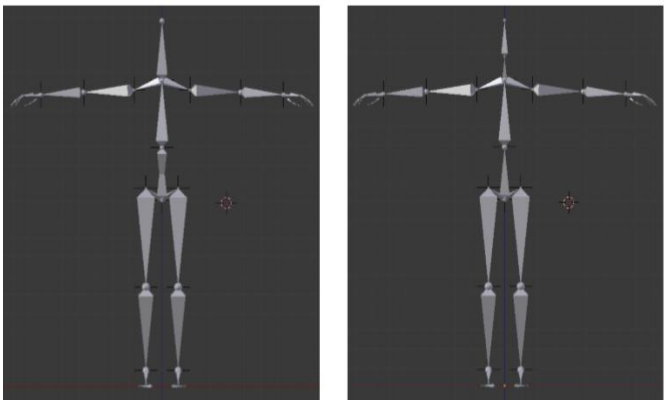


Fig -7: Control Rig and Main Rig

6.2 Blender to Unity transition

Traditionally, if to talk about the character animation in Blender using mocap data it is all about importing .bvh files to the project. Bio vision Hierarchy Data (BVH) or .bvh format is a commonly used datatype which contain the description of the rig hierarchy and its movements with time by holding the values of the key frames.

6.3 importing the animation to unity

Thus there are two main ways to import animation and objects from Blender to Unity: using .blend or generic .fbx files. First one is a native file format for Blender. It can contain the 3D models, texturing, light, sound, other objects and data. Basically, it is a whole project file which can be quite massive. While the .fbx (shortening from the name Film box) is a type of files for saving 2D and 3D content (meshes, drawings, animation, etc.). Its main aim is to provide the compatibility between multiple 3D design applications, so that the content of the file can be opened, used, and modified by those tools.

While importing the objects with project blend files Unity does not open them itself, but it makes an automatic call back to Blender, so it would generate the appropriate .fbx file for the elements of the Blender scene. Because of that, it might seem that these two ways are absolutely similar. However, it is not like that.

7. RESULT AND ANAYLSIS

The methods have been tested on 10,000 images, and its 3D pose prediction results were compared to the GT data. In addition, Human3.6M dataset provides the tool for validation of the methods with the GT files. The real data contain x, y, and z GT coordinates of 17 joints for each provided image while the output of tested methods contains from 17 to 25 joints. Moreover, all of the frameworks provide different

types of output which sometimes require additional processing because the predicted data has to be given in the .csv format. Also, the predictions of the methods often have big variations in scale. Thus it has to be scaled in accordance with the GT data for the proper validation process. Taking all of that into account, the validation tool offered by Human3.6M has been modified considering the validation needs of the project.

The Pose estimation methods output is ordered, aligned at the pelvis joint and labeled with respect to the GT data format. After, the preprocessed data goes through the validation where the data is filtered so that only matching pairs of joints are scaled and then compared in accordance with the chosen metric. As a result, the feedback about the accuracy of the method is received. Process The corresponding quantitative result of the 3D PE methods are given in Table 1. It contains the averaged values of original MPJPE error over 10,000 samples, error after scaling the predictions data, the scaling factor itself and time that the methods needed to perform the PE for all of the images.

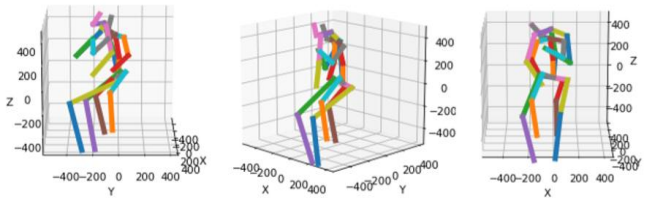
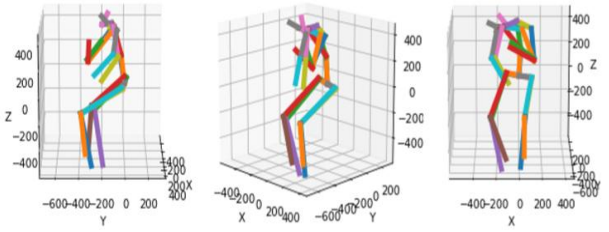
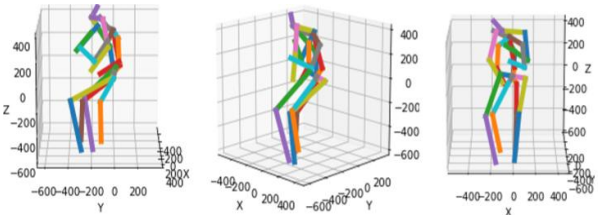
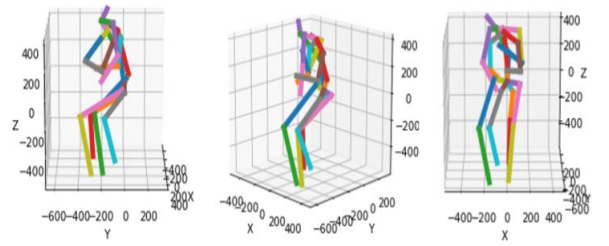
Table -1: validation results of the tested 3D pose estimation methods.

Method	MPJPE(mm)		Scaling Factor	Approximate calculation time (hours)
	Error	Scaled error		
LFTD	352,486	296,267	0,588	~~20
VNect	84,270	81,270	1,0289	1
SMPLify	438,485	306,303	628.315	~~800
NBF	424,741	310,723	622,062	9

7.1 Animation evaluation.

To evaluate the results of the implementation steps, it was decided to use two questionnaires with an almost similar question but different content. For the evaluation, 10 videos of different motions were recorded. The length of the videos varied from 6 to 8 seconds. From every input video, there were three output files, one per each 3D PE method that was involved in the framework. So after the videos were passed through the whole system, it returned altogether 30 .blend files.

Table -2: Results of PE in Different Methods

Results of 3D pose Estimation Methods		
0^0 View	45^0	90^0
		
VNEcT		
		
SMPLify		
		
Neural Full Body Fitting		
		

Then, these files were imported to Unity and applied to the standard third-person character. The animations from Blender files and Unity were recorded and provided in the questionnaire for the qualitative evaluation. Results of the Unity animation were evaluated in the first questionnaire. After, the participants were asked to fill in the second questionnaire which contained the videos made using original Blender animations.

8. CONCLUSIONS

This thesis project represents an attempt to create an affordable tool for everyone, that from a single RGB video can produce the mocap data and transform it into 3D character animation that would repeat the motion of the person from the input video. All of the steps the input data is going through before the animation file is generated, such as object detection, frame cropping, 3D pose reconstruction, data normalization, and finally, mapping and 3D character animation in Blender are performed automatically. It means that the user only needs to provide the system with an input video and to choose which 3D PE method to use. Hereby, developed framework is simple to use and affordable, because it utilizes only free open-source software such as YOLO object detector, a set of different NNs for pose estimation, and Blender.

9. FUTURE SCOPE

Also, a few problems were discovered during the evaluation, such as data shaking and partial motion mismatch with the original video. The discussion about these issues pointed to the directions on how this project can be improved and developed in the future. And in the future we can apply this technology for all game development engines, also for the realistic Games.

REFERENCES

[1] Dent S. (2014), what you need to know about 3d motion capture.

[2] Mikhalchuk D. (2017), Motion capture: What is it? URL: https: //teslasuit.io/blog/motion-capture/motion-capturewhat-it-is, accessed 20 April 2019.

[3] Mousavi Hondori H. (2014) A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. Journal of Medical Engineering 2014.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. Imagenet: a large-scale hierarchical image database. In CVPR, 2009.

[5] Nielsen M.A. (2015) Neural Networks and Deep Learning. Determination Press.

[6] Nibali A., He Z., Morgan S. & Prendergast L. (2018) 3d human pose estimation with 2d marginal heatmaps. CoRR abs/1806.01484. URL: http://arxiv. org/abs/1806.01484

[7] Ning G., Zhang Z. & He Z. (2017) Knowledge-guided deep fractal neural networks for human pose estimation. IEEE Transactions on Multimedia.

[8] CaoZ., SimonT.,WeiS.E.&SheikhY. (2017)Realtimemulti-person2dposeestimation using part affinity fields. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[9] XiaoB., WuH.&WeiY.(2018)Simple base lines for human pose estimation and tracking. In: The European Conference on Computer Vision (ECCV).

[10] Omran M., Lassner C., Pons-Moll G., Gehler P.V. & Schiele B. (2018) Neural body fitting: Unifying deep learning and model-based human pose and shape estimation. Verona, Italy

[11] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for computing face similarities. In ICCV, 2013.

[12] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In CVPR, 2014

[13] Y. Tian, C. L. Zitnick, and S. G. Narasimhan. Exploring the spatial hierarchy of mixture models for human pose estimation. In ECCV, 2012.

[14] D.TranandD.Forsyth. Improvedhumanparsingwithafull relational model. In ECCV, 2010.

[15] F.WangandY.Li.Beyond physical connections: Tree models in human pose estimation. In CVPR, 2013.