Virtual Character Animation based on Data-driven Motion Capture using Deep Learning Technique

Gopika Rajendran
Computer Science and Engineering
College of Engineering Kidangoor
Kottayam, Kerala, India
gopikaraj4@gmail.com

Ojus Thomas Lee
Computer Science and Engineering
College of Engineering Kidangoor
Kottayam, Kerala, India
ojusthomaslee@gmail.com

2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS) | 978-1-7281-8524-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICIIS51140.2020.9342693

***Abstract*—Perceptions in motion capture (mocap) technology are increasing every day as the variety of applications using it is doubling. By leveraging the resources offered by mocap technology, human activity characteristics are captured and can be used as the source for animation. The devices involved in the technology are therefore very costly and hence not practical for personal use. In this scenario, we implement a framework capable of producing mocap data from standard RGB video and use it to animate a character in 3D space, based on the action of person in the original video with the help of deep learning techniques. The Human Mesh Recovery (HMR) scheme is used to extract mocap data from the input video to determine where joints of the person in the input video are located in 3D space, using 2D pose estimation. The locations of 3D joints are used as mocap data and transferred to Blender with a simple 3D character using which the character can be animated. A subjective evaluation of our framework based on the metric called observation factor was performed and yielded an accuracy value of 73.5%.**

***Index Terms*—Computer Vision, Deep Learning, Human Action Recognition, Human Pose Estimation, 3D Reconstruction, Animation**

I. Introduction

Motion capture (mocap) is a process that uses special equipment mounted on actors and software to record and capture the motion of humans or other objects. It is very popular and expensive nowadays that a skilled actors motion is captured and applied to a 2D or more frequently to a 3D character model so that the character can mimic the actors actions in the video and is thus used mainly by professionals. Someone who wants to use mocap to create their own game, animation, or some other product will therefore face the problems of mocap systems availability. Our work aims to develop a cost-effective solution to the problem in support of 3D Human Pose Estimation (PE) which eliminates the need for expensive devices and seeks to develop a platform using free open source tools to generate mocap data. We make use of deep learning methods to realize this platform.

Human PE [1] belongs to the domain of Computer Vision (CV) and has the task of detecting human location, structure, and orientation from images and videos. Human PE really means locating various joints (keypoints), which belong to a specific object or a person in the image or video. The resulting joints can then be linked to a graph showing the skeleton of the person.

The main objective of our work is to develop a framework that would produce an animated character based on a real video of human motion, where the actor does not wear any tracking devices. The developed framework will execute a 2D pose estimation along with a Human Mesh Recovery (HMR) model to define human positions in 3D space. The output from the preceding stage is then used as mocap data that is manually mapped to the 3D character. As a result, the character must be animated based on the video input given to the system. Character and animation produced can be further imported into software like Blender, Autodesk Maya and can be used in movie making, games, etc.

The main highlights of our work are

- We has developed a framework to generate animated characters from input videos with the actor wearing no tracking devices.
- Use a deep learning approach to predict the joints of a human in 3D space without any 3D ground annotation dataset.
- The performance of the framework is evaluated using a metric called observation factor.

The paper is organized as follows: Section II presents the literature survey we had done on human action recognition dealing with the common approaches used in predicting action labels. Our proposed framework is detailed in Section III with submodules: *2D Pose Estimator*, *HMR Model* and *Motion Retargeting*. The results and discussion are made in Section IV which shows the performance of our framework. Finally, Section V summarizes the intention of this work and conclusions we arrived at.

II. LITERATURE REVIEW

Our research aims at exploring human behaviors and estimating human pose through a systematic simulation of human behavior and the interpretation of human behavioral characteristics. The most difficult task in wide-ranging applications such as computer vision and computer graphics is to interpret human motion from a picture. An investigation into these applications led to the Human Action Recognition (HAR) field [2] for understanding human behavior. HAR's ultimate aim is to build a cognitive system that can correctly perceive the video's human behavior and actions. Since the machine

Part No: CFP2058A-ART; ISBN No:978-1-7281-8524-8/20

has really no idea to produce image features from the pixel information which describes the behavior and how to infer from those representations. We divide the recognition of action into problems of representing action and classifying action based on the machine learning and deep learning approach.

First, *Kong et al.* [3], described the traditional process flow for recognition of actions as two main components which typically include action representation and action classification for machine learning. The objective of the action representation is to transform an action video into a feature vector which outputs sample and relevant information about human actions based on global features and local characteristics centered on spatial and temporal differences in the way they perform action representation. Local representation describes the local areas with specific details on the motion as features like Histogram of Oriented Gradients (HOG) were used by *Kobayashi et al.* [4]. Histogram of Optical Flow (HOF) was used by *Sahoo et al.* [5] to record the shape and description of the movement in a local area surrounding points of interest and trajectories. Global Representation extracts information from the source as a whole and represents global descriptors by forming a static image template called the Motion History Image (MHI) as proposed by *Silambarasi et al.* [6]. The action features are given to the action classifier learned from the training samples so that the class boundaries are calculated for specific action classes. Support Vector Machine (SVM) as proposed by *Hung et al.* [7] and Random Forest as presented by *Lu Xu et al.* [8] are common algorithms that recognize action classes among the features of the action.

Secondly, *Earnest et al.* [9], used CNN deep networks, which combine the action representation component and the action classification component into a simple successive trainable framework. CNN is mainly configured for drawing 2D and 3D spatial characteristics from images and videos with temporal information. CNN proposed by *Liu et al.* [10], is used to extract the feature map or heat map that shows the probability of joint occurrences in both 2D and 3D. *Moreno-Noguer* [11] have proposed a multi-stream network which fuses directly the outputs of different streams generated by their respective activation function. In this way, CNN has designed as architecture with multiple fully CNNs that infer the 3D pose from 2D features of the input and thus the system may not be suitable for the long-term collection of data as they do not permit interactions among the streams. As a solution, aggregating temporal information is to add a recurrent layer over the CNNs, such as LSTMs, to build hybrid networks as proposed in [12]. Such hybrid networks take advantage of both CNNs and LSTMs and have therefore shown impressive outcomes in spatial motion patterns, temporal arrangements, and long-range dependencies.

III. PROPOSED SYSTEM

In this section, we detail the architecture of the system and explain how it works.

The framework receives a video clip and a modeled character as input. We then incorporate a Biovision Hierarchy file (BVH) as a controller that allows a modeled character to demonstrate the motion of the actor as in the video. The process of learning is divided into two stages, first stage is *2D PE* which is a part of *learning process* while the second stage is *Human Mesh Recovery (HMR)* along with the animation generation process. The overall architectural design is shown in Fig. 1.

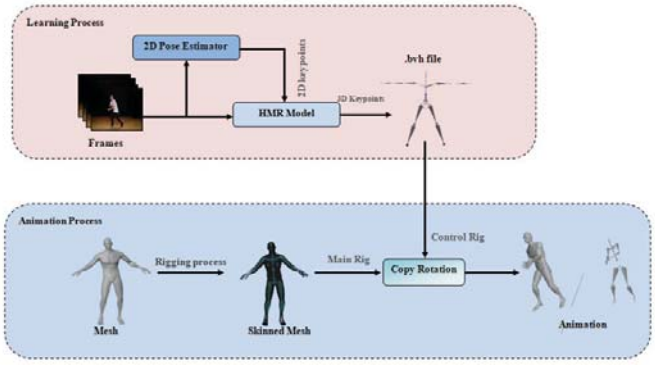

Fig. 1: Blocks of the proposed framework: A) *Learning Process* block performs the deep learning methods to infer the 3D joint/keypoint coordinates of human in a video using *2D Pose Estimator* and *HMR* model. B) *Animation Process* block maps the extracted joint/keypoint coordinates to the character in its virtual environment using *Motion retargeting*.

While the PE accepts only the images as input, the input video for the framework must therefore be divided into a sequence of frames. The input frames are first processed by the pose estimation stage, where learned *2D pose estimator* is applied to extract the 2D pose of the actor in each frame. The set of predicted 2D poses and input frame proceeds to the *HMR* model, which plays the role of 3D pose estimator to records the estimated x, y, z coordinate values for each keypoints as a *.csv* file for the respective frame. Furthermore, to make it easier and faster to associate the keypoint locations from the *.csv* files, it is better to merge all *.csv* files into one large file, which contains the x, y and z coordinates of each joint for each frame as they appear in video sequence individually. Then, the data imported by column from the *.csv* files are assigned to the respective objects named "empties" in the BVH file which acts as a motion capture data file. It can be done with the Blender environment using the Python scripting functionality and thus the mocap data can be imported directly there. The benefit of this method is that each joint in Blender has a separate location channel, so the input PE data structure makes it easy to assign values individually for each parameter. Each joint of the rig responsible for managing the action must be matched to corresponding keypoint data to do a proper mapping because the number of elements in those two sets of joints can be different. Therefore the extra joints are removed before applying motion retargeting. Retargeting motions is simply editing existing motions for the desired effect. During motion retargeting [13], the data from the BVH file is mapped

Part No: CFP2058A-ART; ISBN No:978-1-7281-8524-8/20

to the joints of the character.

Overall, except for the retargeting motion, the framework works automatically. The input video was given to the framework flow through each stage automatically and generate a Blender file (*.blend*) as the output, which needs to implement the retargeting section manually. The file includes all the structures and information required for its future use in Blender or other applications. Phases involved in the process are i) *2D Pose Estimator* ii) *HMR Model* iii) *Motion Retargeting*. We present a detailed discussion of the phases in the following sections.

A. 2D Pose Estimator

We discuss the steps involved in the process of the 2D estimator in this section.

The 2D pose estimator [14] provides a framework that uses a bottom-up approach to determine human poses directly from pixel-level image proof. For this purpose, a two-branch fully convolutional neural network is used to solve the problem using a multi-stage classifier where each stage improves the results of the previous stage. The feed-forward network sequentially predicts a set of 2D heat_maps H_{mp} of body part locations and a set of 2D vector fields P_{af} of part affinities defining the degree of association between parts. The last operation of the neural network returns a matrix consisting of 57 (19×3) vectors for 19 keypoints with x, y coordinates, and their visibility score.

$$\begin{aligned} Heat\_map(H_{mp}) &: \mathbb{R}^2 \longrightarrow \mathbb{R} \\ Part\_Affinity(P_{af}) &: \mathbb{R}^2 \longrightarrow \mathbb{R}^2 \end{aligned} \tag{1}$$

The network architecture is shown in Fig. 2 and consists of two branches with 6 stages: The top branch CNNs (ρ) predicts the probability of visibility on each body part and the bottom branch CNNs (ϕ) predicts fields of part affinity. The system takes input video frames F of size $width(W) \times height(H)$ and feeds these frames to CNNs. Thus the system generates the anatomical keypoints in 2D positions for each frame as the output of humans respectively. First stage takes the input image (frame) and predicts each keypoint's possible locations in the image with a confidence score (called the heat_map). A heat_map is a matrix which holds the confidence that a certain pixel contains a certain part of the network. There are 18 (+1) heat_maps linked to each of the parts. Preferably, if a person appears on the frame, each heat_map should have a single peak when the corresponding part is visible. The location value L_H in H_{mp} is defined as

$$H^*_{mp}(L_H) = e^{-\frac{\|L_H - X_j\|_2^2)}{\sigma^2}} \tag{2}$$

where σ controls the distribution over the peak and $X_j \epsilon \mathbb{R}^2$ be the ground-truth position of each body part j for the human in the frame. It then uses the fact that some joints are attached via limbs to introduce another CNN branch which predicts 2D vector fields called Part Affinity (P_{af}). Part Affinity Fields are matrices which provide information on joint pair position and orientation. They come in pairs: having a P_{af} in the 'x' direction and a P_{af} in the 'y' direction for each component. There are 38 (19×2) P_{af}s associated with each one of the pairs. A limb is created by connecting two parts and P_{af} encodes the path from one part to another; each limb is called a field of affinity between the parts of the body. If a point L_{af} lies in the limb then its value P_{Limb} in the P_{af} is a unit vector pointing from the starting point of the joint to the ending point of that limb; if it is outside the limb, the value is zero.

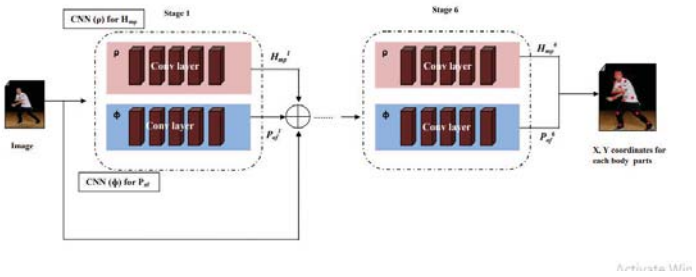

Fig. 2: Network Architecture of 2d pose estimation consists of two branch CNNs.

$$P^*_{Limb}(L_{af}) = \begin{cases} \vec{V} & \text{if limb on } L_{af} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$
$$\text{where } \vec{V} = \frac{X_{j_2} - X_{j_1}}{\| X_{j_2} - X_{j_1} \|_2}$$

Thus the network produces a set of confidence score H_{mp} ϵ $\mathbb{R}$ for each keypoint and a set of affinity vector field P_{af} ϵ $\mathbb{R}^2$ over the successive stages, $t \epsilon 1, , T$ as

$$\begin{aligned} H^t_{mp} &= \rho^t(F, H^{t-1}_{mp}, P^{t-1}_{af}), \forall t \geqslant 2 \\ P^t_{af} &= \phi^t(F, H^{t-1}_{mp}, P^{t-1}_{af}), \forall t \geqslant 2 \end{aligned} \tag{4}$$

where ρ and ϕ represents the CNNs for inference at each stage. At each stage, the predictions from both branches are convolved with the image features F for the next stage. The last operation of the network is just a concatenation of two different vectors as heat_maps and part affinity field.

B. HMR Model

In this section, we present the *HMR* model [15] which represents a framework for regaining a 3D human body structure (mesh) from a single RGB image of a human.

We execute *HMR* model (as in Fig. 3) with estimated 2D joints from *2D pose estimator* over the frames. The model uses the generative human body model called Skinned Multi-person Linear (SMPL) model [16], parameterizes the mesh by 3D joint angles, and a linear shape space of low dimensions. The joint positions alone will not limit the maximum degree of freedom at each joint, i.e determining the body's full pose from just the 3D joint positions is non-trivial. Alternatively, we produce the relative 3D rotation matrices for each joint, which collects information about 3D head and limb orientation. Predicting rotations also make sure the limbs are symmetrical and true in length.

Part No: CFP2058A-ART; ISBN No:978-1-7281-8524-8/20

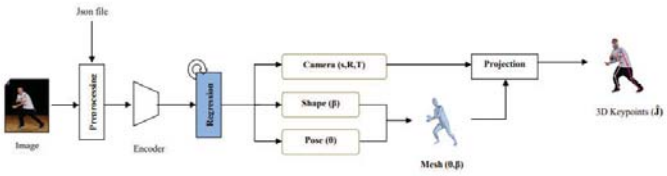

Fig. 3: Overview of HMR Model.

During training we assume that all frames are annotated with estimated 2D joints based on ground truth. The model requires a bounding box of humans in the image. The image is then scaled to 224 $\times$ 224, maintaining the aspect ratio so that the bounding box diagonal is approximately 150 px as a preprocessing step. For a frame centered on a human, the network must infer the 3D SMPL mesh parameters $M(\theta, \beta)$ ϵ $\mathbb{R}^3$ and the camera (s, R, T) so that after projection the 3D keypoints match the interpreted 2D keypoints. Convolutional features of the frame are sent to the iterative 3D regression module which aims to deduce body model and camera parameters so that its 3D joints extrapolate onto the inferred 2D joints. The purpose of the 3D regression module is to produce a set of parameters that describes the 3D reconstruction of the human body defined as $\psi = \theta, \beta, R, t, s$ for an image. The learning can be formulated as

$$L = \sum ||v_k(gt_k - g\hat{t}_k)|| \tag{5}$$

Here v_k the visibility score for k^{th} keypoints which take only either 1 (if visible) or 0 (otherwise) and the gt_k is the k^{th} ground truth 2D joint. Regress ψ in an iterative feedback loop where the current estimate is gradually updated. Especially, the 3D regression module takes the feature parameters and the current parameter ψ_t as an input and generates the residual $\Delta\psi_t$. By applying this residual to the current estimate, the parameter is modified as ψ_{t+1} = ψ_t + $\Delta\psi_t$.

Resultantly, $M(\theta, \beta)$ is obtained by shaping the template body vertices conditioned on β and θ, then articulating the bones according to the joint rotations via *forward kinematics* and finally reshaping the surface with linear blend skinning. The 3D keypoints $J(\theta, \beta)$ ϵ $\mathbb{R}^3$ are obtained by linear regression from the final mesh. Hence given ψ, the projection of $J(\theta, \beta)$ is

$$\hat{J} = s \prod RX(\theta, \beta) + t \tag{6}$$

where $\prod$ defines the projection function. Projection allows the network to create 3D human bodies that lie on the multiplicity of human bodies and serve as weak supervision for images without 3D annotations of ground truth.

C. Motion Retargeting

A detailed discussion of motion retargeting, which aims to copy the motion from the video to the target character is presented in this section.

When the joint data is mapped to the rig (Skeleton) joints and imported into Blender software, a combination of two rigs [13] is used to perform a proper mapping that makes the generated rig to the captured mocap data transformation specified in the BVH file. The first rig called "control rig" is used for the purpose of mapping, whereas the second rig is responsible for the deformation of the mesh and is called the "main rig". The main rig does not really imitate the empties but repeats the motion of the control rig. In order to connect the empties of the joints to the control rig and allow the main rig to follow the performance, Blender provides a set of constraints that can be applied to the skeleton and to objects like empties. Constraints allow the user to control the motion by giving them a tool to manipulate the properties of the object, such as rotation, translation, and scaling. It is therefore very important to assign all the constraints of the rig in the correct way and order. Otherwise, the skeleton may move incorrectly, resulting in an invalid animation. Precisely the main rig is skinned to the character's mesh and the whole process is representative of the control of the puppet doll, and the places where the empties are connected to the rig are the points where the thread is tied to the puppet.

The main skeleton has to be attached to the control rig in order to be animated. The Process of attachment can be done by using bone constraints called "Copy Rotation". It allows the corresponding bone to be chosen from the target object, which is the whole "Control rig" in this particular case, and thus the bone from the main skeleton would replicate the bone rotations from the puppet. So this kind of design helps the main skeleton to replicate the control rig's motion without corrupting its own skeleton with such things as stretching.

IV. RESULTS AND DISCUSSION

In this section, we discuss the results we obtained and analysis of the result. Section progress in two subsections i) Implementation details ii) Results.

Implementation Details: The learning process has been done in the Google Colab so that we can use it regardless of GPU utilization, CPU configurations, etc. It offers a cloud platform for research on machine learning and deep learning. Here we use TensorFlow 1.3.0 for the learning process. As said, video is first converted into sequential frames with a frame per speed (fps) of 20. 2D pose estimator is performed with 6 stages consisting of two branches one for each. The dataset that we had used for predicting 2D pose is the MS COCO [17] with 19 keypoints. The HMR model uses the ResNet-50 network [18] to encode the image, pre-trained on the task of classifying ImageNet. The 3D regression module consists of two fully connected layers with 2^{10} neurons each with an intermediate dropout layer. For all of our experiments, we use T = 3 iterations.

Results: Although we recover even more than 3D skeletons, it is difficult to evaluate the result because there are no 3D ground mesh annotations for the current dataset. Fig. 5 shows the qualitative results of our framework that imitate almost similar paths of motion as in the video frames. From all this,

Part No: CFP2058A-ART; ISBN No:978-1-7281-8524-8/20

(a) 99^{th} percentile

(b) 75^{th} percentile

(c) 25^{th} percentile

(d) 0^{th} percentile

Fig. 4: Subjective Evaluation of HMR model at 0^{th}, 25^{th}, 75^{th}, 99^{th} percentile of error.

we could see how our model mirrors the orientation of the head and limbs. We perform a subjective evaluation of 100 images that are tested at various percentiles of error against the HMR model. A percentile of error is a metric, that shows the value below a given percentage of observations falls within a group of observations. The percentile of error is defined by the wrongly estimated joint from the observed input image. In Fig.

Fig. 5: Results showing input image/video and corresponding character animation mapping done in our work

4 we demonstrate subjective evaluation at various percentiles of error trained on MS COCO dataset. High percentile implies high failure as the 99^{th} percentile (in Fig. 4a) indicates that almost all the estimated joints are misplaced with respect to visible joints in the frame. Whereas 0^{th} percentile (in Fig. 4d)indicates that the mesh obtained from the HMR model is overlaid with the human in the frame.

A metric called Observation factor (Ob_{fac}) is defined as the fraction of estimated outcome to actual outcome as in the equation below:

$$Ob_{fac} = \frac{\sum_{i=1}^{4} I_i N_i}{I_T N_T} \quad \text{such that N = 5, 10, 15, 19} \qquad (7)$$

where I_i indicates the number of images obtained in the $i = 1, 2, 3, 4$ percentile categories with the maximum number of correctly predicted keypoints as N_i. I_T and N_T are the total number of images, here 100 that is used to predict a total of 19 keypoints in this experiment. The values of N act as a grading scheme where 19 represent the 0^{th} percentile category since criteria are set as images I_1 in 0^{th} percentile will possess a range of 16-19 correctly predicted keypoints and so on. Fig. 6 shows the graphical representation of the 4 cases. We have obtained the ob_{fac} of .7352 on testing for over 100 images. Hence we claim that our framework has exhibited performance of 73.5%.

V. CONCLUSION AND FUTURE WORK

Our work is an attempt to create an inexpensive tool for generating the mocap data from a single RGB video and map it into a 3D character animation that would replicate the movement of the human. All the steps that the input data passes before the animation file is generated, such as frame extraction, 2D pose estimation, 3D pose reconstruction, and finally mapping of 3D character animation in Blender are

Part No: CFP2058A-ART; ISBN No:978-1-7281-8524-8/20

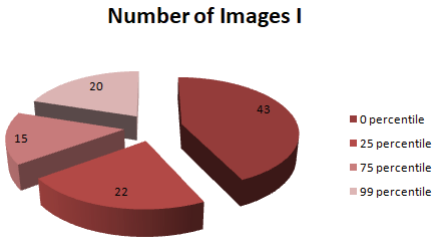

Fig. 6: Chart Representation of Subjective evaluation of 100 images

performed by hand. HMR model infers 3D mesh parameters directly from image features using a 2D pose estimator without 3D ground support. Predicted coordinate values are then used as motion capture data and imported for mapping. As a result, the character is animated on the basis of the video input provided to the framework. Hence, the developed framework is simple to use and affordable because it uses deep learning techniques for pose estimation and free open-source software such as Blender.

Although our system can mimic a video clip, there are a number of drawbacks to this. Since the performance of the motion imitation stage depends on the reconstructed gesture, the HMR may fail to replicate the action when the 2D pose estimators are unable to correctly predict an actor's pose. We performed a subjective evaluation on the framework using the metric called observation factor (ob_{fac}) which describes correctly estimated keypoints in the images from the expected one. Since the evaluation is based on a human perspective, we need to define a well-formed metric to evaluate our framework quantitatively in the future. Devising methods to improve the accuracy of prediction is scheduled as a future work.

References

[1] Alexander Toshev and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660.

[2] Gopika Rajendran et al. "Study on Machine Learning and Deep Learning Methods for Human Action Recognition". In: (2020).

[3] Yu Kong and Yun Fu. "Human action recognition and prediction: A survey". In: *arXiv preprint arXiv:1806.11230* (2018).

[4] Takuya Kobayashi, Akinori Hidaka, and Takio Kurita. "Selection of histograms of oriented gradients features for pedestrian detection". In: *International conference on neural information processing*. Springer. 2007, pp. 598–607.

[5] Suraj Prakash Sahoo, R Silambarasi, and Samit Ari. "Fusion of histogram based features for Human Action Recognition". In: *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE. 2019, pp. 1012–1016.

[6] R Silambarasi, Suraj Prakash Sahoo, and Samit Ari. "3d spatial-temporal view based motion tracing in human action recognition". In: *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE. 2017, pp. 1833–1837.

[7] Chin-Pan Huang et al. "Human action recognition using histogram of oriented gradient of motion history image". In: *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*. IEEE. 2011, pp. 353–356.

[8] Lu Xu et al. "Human activity recognition based on random forests". In: *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE. 2017, pp. 548–553.

[9] Earnest Paul Ijjina and C Krishna Mohan. "Human action recognition based on recognition of linear patterns in action bank features using convolutional neural networks". In: *2014 13th International Conference on Machine Learning and Applications*. IEEE. 2014, pp. 178–182.

[10] Yu Liu et al. "End-to-End Algorithm for Recovering Human 3D Model from Monocular Images". In: *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE. 2018, pp. 1082–1087.

[11] Francesc Moreno-Noguer. "3d human pose estimation from a single image via distance matrix regression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2823–2832.

[12] Charissa Ann Ronao and Sung-Bae Cho. "Human activity recognition with smartphone sensors using deep learning neural networks". In: *Expert systems with applications* 59 (2016), pp. 235–244.

[13] Anastasiia Borodulina. "APPLICATION OF 3D HUMAN POSE ESTIMATION FOR MOTION CAPTURE AND CHARACTER ANIMATION". In: (2019).

[14] Zhe Cao et al. "Realtime multi-person 2d pose estimation using part affinity fields". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7291–7299.

[15] Angjoo Kanazawa et al. "End-to-end recovery of human shape and pose". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7122–7131.

[16] Matthew Loper et al. "SMPL: A skinned multi-person linear model". In: *ACM transactions on graphics (TOG)* 34.6 (2015), pp. 1–16.

[17] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[18] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

Part No: CFP2058A-ART; ISBN No:978-1-7281-8524-8/20