FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Anastasiia Borodulina

APPLICATION OF 3D HUMAN POSE ESTIMATION FOR MOTION CAPTURE AND CHARACTER ANIMATION

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2019

Borodulina A. (2019) Application of 3D human pose estimation for motion capture and character animation. University of Oulu, Degree Programme in Computer Science and Engineering. Master's thesis, 57 p.

ABSTRACT

Interest in motion capture (mocap) technology is growing every day, and the number of possible applications is multiplying. But such systems are very expensive and are not affordable for personal use. Based on that, this thesis presents the framework that can produce mocap data from regular RGB video and then use it to animate a 3D character according to the movement of the person in the original video. To extract the mocap data from the input video, one of the three 3D pose estimation (PE) methods that are available within the scope of the project is used to determine where the joints of the person in each video frame are located in the 3D space. The 3D positions of the joints are used as mocap data and are imported to Blender which contains a simple 3D character. The data is assigned to the corresponding joints of the character to animate it. To test how the created animation will be working in a different environment, it was imported to the Unity game engine and applied to the native 3D character. The evaluation of the produced animations from Blender and Unity showed that even though the quality of the animation might be not perfect, the test subjects found this approach to animation promising. In addition, during the evaluation, a few issues were discovered and considered for future framework development.

Keywords: Mocap, Computer vision, Deep neural networks, Blender, Unity, YOLO, Video processing

TABLE OF CONTENTS

FOREWORD

The work on this master's thesis was performed in the Center for Machine Vision and Signal Analysis (CMVS) at the University of Oulu.

I want to express the gratitude to my supervisor Prof. Janne Heikkilä for all of the guidance, patience, and support that he provided during my studies and work on this thesis project.

Also, I would like to thank my family and friends for always supporting and helping me on my way to this Master's degree.

Oulu, Finland June 21, 2019

Anastasiia Borodulina

ABBREVIATIONS

2D	Two dimensional space
3D	Three dimensional space
AI	Artificial intelligence
API	Application programming interface
AR	Augmented reality
BB	Bounding box
BVH	Biovision hierarchy data
CG	Computer graphics
CGI	Computer-generated imagery
CNN	Convolutional neural network
CV	Computer vision
DLCM	Deeply learned compositional model
DNN	Deep neural network
DPNet	3D pose network
FFN	Feedforward neural network
GT	Ground truth
IMU	Inertial measurement unit
LED	Light-emitting diode
LFTD	Lifting from the deep
LSTM	Long short-term memory
Mocap	Motion capture
MPJPE	Mean per joint position error
NBF	Neural body fitting
NN	Neural network
PAF	Part affinity fields
PE	Pose estimation
PRCNN	Pairwise ranking convolutional neural network
ReLU	Rectified linear units
ResNet	Residual neural network
RGB	Red, green, blue
RNN	Recurrent neural network
SMPL	Skinned multi-person linear model
USD	United States Dollar
VFX	Visual effects
VR	Virtual reality
YOLO	You only look once

1. INTRODUCTION

Motion capture (Mocap) [1, 2, 3] is a procedure of tracking and capturing the motion of humans or other objects using special equipment and software. The recorded data can be applied to the computer model for further animation or research purposes. During the last decades this technology became widely used in such different areas as military, robotics [4], medical applications [5, 6], game design, animation, and creation of visual effects (VFX), etc.

Unfortunately, the mocap technology is expensive that is why it is used mostly by professionals. Therefore, any person who would like to try mocap for developing their own game, animation or some other product will face the problems of availability of mocap systems. This project is looking for a solution in the 3D human pose estimation (PE) methods and aims to develop a framework that utilizes free open-source tools to produce the mocap data.

Human PE is related to the field of computer vision (CV) and its task is to detect human position, shape, and orientation in different images and videos. It means to localize different keypoints, also called joints, that belong to a specific object or a person. After that obtained joints can be connected in a graph which represents the skeleton of the person at the image.

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 a 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 PE method is defined by the user during the submission of the input video. Next, the output from the previous step is used as a mocap data which is automatically mapped to the 3D character. As a result, the character should be animated in accordance with the input video that was given to the system. Produced character and its animation can be further imported to other software and used in motion pictures, games, etc.

The structure of the thesis consists of the following parts. Chapter 2 takes a closer look at the mocap technology, its history, and different types of systems that allow capturing the motion of the object. Chapter 3 gives an explanation on the topic of human PE and related concepts. In addition, this part provides a description of different types of 2D and 3D methods. Chapter 4 describes the most important concepts of 3D character animation. Chapter 5 is provided with the overview and detailed description of the framework that combines all of the above-described concepts. Chapter 6 consists of two parts. The first one provides the experimental evaluation of the 3D pose estimation methods while the second part explains the subjective evaluation of the animation process and its results. Chapter 7 contain the discussion of the obtained outcome and possible improvements to the developed framework. Finally, Chapter 8 makes conclusions about the thesis project.

2. MOTION CAPTURE

From the beginning of the 21^{st} century mocap popularity in the entertainment area started to grow exponentially. Nowadays, it is very common that the motion of a professional actor is recorded and applied to 2D or more often to 3D character model, so its movement during animation would entirely repeat the acting. Figure 1 illustrates the mocap recording process for Human3.6M dataset [7] creation. This method allows having a smooth animation that is extremely hard to get using traditional methods. It makes the characters look realistic and alive. A great example of the mocap use is famous *Gollum* from the epic adaptation of the *The Lord of the Rings* series played by Andy Serkis. This character made a revolution in computer-generated imagery (CGI).

Figure 1: Mocap data recording process for Human 3.6 dataset [7].

2.1. History

To most of the people, the term "animation" seems to be quite young. In fact, the story of animation has begun at the beginning of 19^{th} century with the creation of the phenakistiscope [8]. This device had a set of images that were moving in a circle. By reaching a certain rotational speed it created an illusion of motion. This effect is based on the persistence of the human vision which means that very fast motion of the object is not perceived by human eyes and is interpreted as a solid image. In general, it is the basis of the cinematography because anything which is shown on the screen is just a sequence of constantly changing images.

In early 20^{th} century the traditional animation was born when one of the first cartoons was drawn frame by frame *Fantasmagorie* by Émile Cohl in 1908. The same

technique was used to create the most famous cartoon character - *Mickey Mouse* back in 1923 by Walt Disney.

In 1914 Max Fleischer made the life of the animators easier by creating the rotoscoping technique. In the heart of this method is the video sequence of actors' performance which is projected to the thin paper and the animator just need to outline the picture in each shot frame by frame. Then all of the new pictures were photographed and transformed into one cartoon video. This helped to simplify and speed up the process. Shortly, rotoscoping became widely used in cartoon production among the animators including Walt Disney. His first work made with this technique was *Snow White and the Seven Dwarfs* (1937). Since then the method became very popular. Nevertheless, with the advent of the CGI era in the 1990s, the classical rotoscoping became rare although it is still used in some projects.

CGI has adopted the idea of rotoscoping for its own needs - usually, it is used to add different objects in the scene. For example, when it is needed to add a crowd or more decoration to the scene or any other effects after shooting on a green screen. As a result, a lot of motion pictures and even computer games would not exist without rotoscoping.

Back in 1995, the Pixar Animation Studios opened a great world of CGI to the viewer in their *Toy Story*. The main difference with everything that had been made before was that the characters were no longer flat 2D images, instead, they became 3D models of humans-alike creatures, animals and so on. It was a revolution in the animation, VFX and game industry. Now the studios are competing who will be first to create more realistic characters, more immersing scenes, and fascinating words.

Thus mocap took over the baton from the rotoscoping and occupied one of the most crucial positions in the entertainment industry.

2.2. Performance capture

Mocap name speaks for itself that the main target is to capture the subject's movement itself and that is how it was till the turn of the millennia. In the early 2000's the technology advanced enough to catch even small motion features like facial expressions, lips' and fingers' movements. Since then, it became more than just recording of a general body motion but a real performance capture. The actors have to really become their characters and live it. Benedict Cumberbatch, as an example, made a fascinating performance capture work as a Smaug for *The Hobbit* series.

Performance capture made a big step for the CGI. Over the last almost 20 years it has made a significant progress from iconic *Gollum* from *The Lord of the Rings* series (2001-2003) and *King Kong* (2004) to *Avatar* (2009) and *Rise of the Planet of the Apes* series (2011-2017).

2.3. Motion capture systems

Usually, the application of mocap technology in digital character animation is complicated and involves the use of professional equipment, like cameras, sensors, and other. Altogether, it is a massive and complicated system to set up. There are two main types

of mocap systems that are based on different principles [1, 2, 3, 6, 9, 10]: optical and non-optical.

2.3.1. Optical systems

The most popular mocap systems are optical. It is called so because the actor has to wear a special costume with a set of markers on it. The markers are located in the places of the most interest for good animation. Usually, it is at the positions of the joints. This allows the system to see them with no problems. To calculate an exact position of the markers multiple high-resolution well-calibrated cameras are used. The number of cameras varies from a minimum of 2 to tens. Each of them provides a 2D location of each reflector. Then the data from all of the cameras is combined by a specific software using triangulation to calculate the location of the markers in a 3D space and thus reproduce the performance.

Optical systems have a very high frame rate which allows recording quick movements like dances or martial arts. The frame rate depends on the resolution of the camera and can reach 200 frames per second and even more. At the same time, the costumes and reflectors are light and do not interfere with the natural movements of the actors [2]. The actors can move around big spaces but the technology is sensitive to light conditions and overlaps with other objects. So it can be successfully applied only in specially equipped premises. In addition, the setups that require such a big amount of equipment and sophisticated software are very expensive in use [2]. Its price easily goes over 250,000 USD [10].

Active markers

The optical systems use two types of markers: active and passive. Active markers represent a set of light-emitting diodes (LED) placed on the actor's costume that emits light. Each marker has own identifier which helps the system to distinguish joints from each other. It is very useful when there are multiple actors on the scene or some body parts disappear due to overlapping and then pop up again. But additional light sources on the scene can create an effect of ghost markers for the system [9].

Passive markers

Passive markers are also called reflective because the markers are covered with retro-reflective material that has to reflect the infrared light that comes from the LEDs located next to the lens of the tracking camera back to the camera. An example of mocap system with passive markers is shown in the Figure 1. When comparing to active markers, passive ones do not have any identifiers so when the markers are too close to each other the system may confuse them with each other [11]. This and other issues can result in noisy data and sometimes gaps in the data. To avoid these distortions it is good to use multiple cameras. For example, to solve this issue up to 300 cameras can be used while in a regular setup it might be from 6 to 24 cameras [10].

Optical systems are extremely popular among animators, VFX, and game development studios because it produces detailed and accurate data with a high frame rate which allows creating high-quality realistic character motion.

2.3.2. Non-optical systems

Aside from optical systems, there are other methods which are combined in a group of non-optical methods. Depending on the types of sensors there are magnetic, mechanical, inertial and other, less popular technologies [3]. There is no perfect system, as all of them have pros and cons, so the choice of the system depends on the task.

Mechanical

One of the cheapest mocap systems is mechanical. Its affordability allows this system to be quite popular. To capture the data the actor has to wear so-called, exoskeleton which repeats the subject's movements. It consists of a set of narrow metal or plastic strips that are fixed around the actor's joints and back. At each connection point, there is a goniometer which plays a role of the sensor that provides the data about the angles between the corresponding stripes [9].

The sensors might transmit the data using a wireless connection or be connected using cables. Wireless sensors allow the actor to move around big areas and not worrying where the camera is. It gives an advantage over optical and inertial systems. At the same time, systems that use cables can provide a lot of motion limitations and discomfort to the subject [2, 3]. Collected by goniometers data after processing by kinematic algorithms produces the body position.

This kind of setup creates on its own some complications as sensor displacement during the movement can result in big accuracy error. At the same time, it is hard to place the sensor correctly, in accordance to the joints. Also, the soft tissue volume can affect the measurements. In addition, some joints like shoulders or hips have a few rotational degrees of freedom, and thus angle measurement using just goniometers becomes a non-trivial task. All of these factors have a significant impact on the system accuracy, so it requires frequent re-calibration [2, 9]. Nevertheless, the pose data obtained in this way still do not look very realistic.

Magnetic

Magnetic systems by their appearance might remind the mechanical ones. However, there is a significant difference - the magnets play the role of sensors. In addition, there is a source of the low-frequency electromagnetic field. The magnets are placed all over the body and together with the source are connected to each other and to the processing block with a set of wires, which create a lot of limitation to the actor's motion. This unit is responsible for tracking the changes in the electromagnetic field caused by the sensors' movements. Further, this data is transmitted to the computer where a specific software recalculates the 3D location and rotation of each joint [2, 3, 11].

The use of magnetic fields entails multiple challenges. First of all the system is very sensitive to any influences from outside that can be caused by other electrical

devices and equipment. Moreover, the metal constructions in the walls, ceiling, other surrounding structures, electrical wires inside the premises can cause data distortion. Nevertheless, the system does not have any problems with body parts overlapping because the human tissue does not affect the magnetic field [9]. But the presence of a few actors on the stage at the same time will again have an impact on the accuracy due to the fact that the sensors of different people will interfere the measurements of each other [3, 11]. At the same time, magnetic field is inversely proportional to the distance from its source. Thus the capturing area is quite small [9].

Despite all the shortcomings, this system can be relatively cheap. If it is set up correctly it produces good quality data with high enough frame frequency which is around 100 frames per second to capture everyday scenarios [10].

Inertial

The core difference of inertial systems with the previous ones is in the sensors themselves. Each marker includes inertial measurement units (IMUs) like gyroscope, accelerometer, and magnetometer, and thus to capture the data there is no need in any cameras or intermediate recording devices as required in optical or magnetic systems, but a system is needed to provide with an absolute position. Considering all of that the system provides the data about the location and rotation of the joints, plus angular velocity and accelerations of the keypoints. Captured data is sent to the computer for further processing and storage.

Due to the specification of the IMUs that are used with time they accumulate the error so the data starts drifting. Usually, it appears within a few minutes [3]. This error can be minimized by a proper combination with complementary sensors [9].

Inertial systems are portable thus it is good for changing environments of different scales with changing amount of light. They have a good frame rate and can provide fairly accurate data. Often this kind of setups can also provide the hands' motion tracking. In addition, these systems are not very expensive. All of that makes them quite popular in the entertainment industry.

2.4. Motivation

Summarizing everything said above, it is clear that the use of mocap data for creating a good quality animation is not affordable for the beginners and enthusiasts because all of the systems are quite expensive for personal use.

The aim of this work is to make this process more affordable for non-professional users, so that everyone could make their own animation with a regular camera. In this work, 3D human pose estimation methods are utilized to provide the mocap data which is later adopted for the purpose of 3D character animation.

3. RELATED WORK ON POSE ESTIMATION

Recently, human PE became one of the most interesting and discussed topics in CV but also, it is one of the most complex problems due to the fact that the human body is a complicated mechanism itself, which has a set of limitations. In addition, things like clothing, different types of occlusions, and light conditions are making this issue even more challenging. This nontrivial task can be solved in different ways including various estimation techniques. These methods can be divided into two big categories: 2D methods - those, that produce the location of the keypoints in the 2D space; and 3D methods, which together with the location of the human body keypoints in 2D space also provide the data about their depth which allows to see the geometry of the object in a 3D space. These methods are base on deep learning architectures like Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN).

3.1. Neural networks

The neural networks is a programming paradigm which tries to imitate the structure of the human brain in order to solve such complicated tasks like image and speech recognition. The interesting thing here is that compared to other programming paradigms where the programmer himself defines how the program will solve the task, the neural network has to find it out on its own. In the simplest form, it can be represented as a black box (Figure 2) where some input data is given to the NN, and it is known what kind of output has to be produced but it does not matter how exactly this result will be calculated.

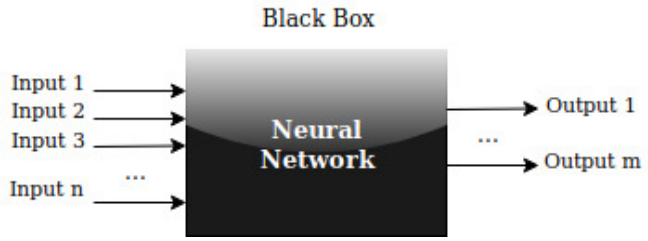


Figure 2: NN as a Black Box.

If to look inside this black box, the NN consist of layers and layers of "neurons". They are also called perceptrons. An example of a simple NN is given in Figure 3. The leftmost layer is called "input layer" and the rightmost one - "output layer". Everything in between got the name "hidden layers" because it is not important to know what are the values in the nodes there. Depending on the complexity of the task some networks might have multiple hidden layers.

Generally, the process goes from small concepts at the input to bigger ones at the output. For example, if the image of a handwritten number is given to the NN, each node in the input layer will contain the value of a specific pixel in the image. The number of nodes in this layer will be equal to the total number of image pixels. While there will be only 10 output nodes assigned to all possible numbers from 0 to 9. These

nodes will contain the values between 1 and 0. The neuron with the highest value points to the digit that is depicted in the image.

In order to give a correct answer, the system has to go through the training process where the NN is given the input and the correct answers that have to be at the output. In this way, the network has to calculate a set of suitable parameters for each connection and each node in the hidden layers which will allow it to match the input with given output, so that after the training the NN could make a correct decision for the similar input on its own. This process is called learning.

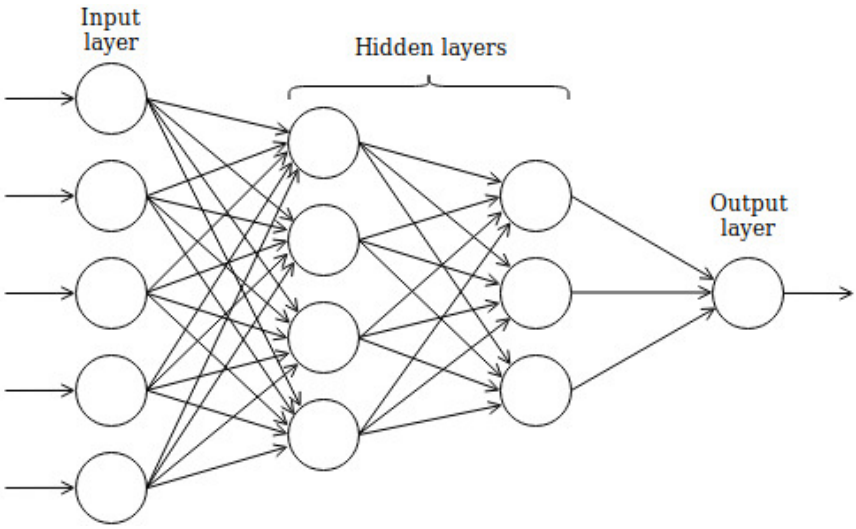


Figure 3: Example of a NN structure [12].

One of the most popular and used NNs in the field of CV for image recognition and classification is CNN whose typical structure is illustrated in Figure 4. In some layers instead of specific parameters CNN applies functions of convolution or pooling. Overall CNN is a combination of convolutional and pooling layers, activation functions, such as rectified linear units (ReLU) and fully connected layers [13].

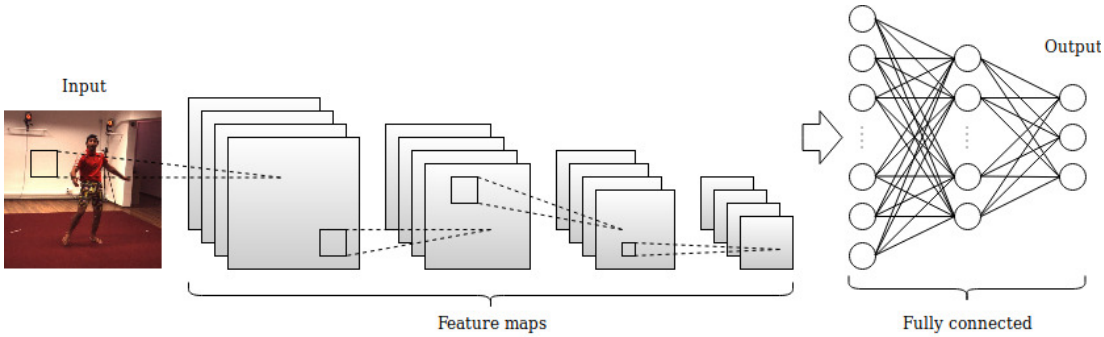


Figure 4: Typical CNN structure.

Sometimes, they are also called feedforward neural networks (FFN) because the information from one layer always goes to the next one and it is never passed to the previous one. That is why it is a loop-free network.

3.2. Heatmaps

The key components of successful PE in the latest research are the heatmaps [14, 15]. In general, a heatmap is a visual depiction of some data. Mostly, it is used for analytic purposes. Within the framework of PE, heatmaps represent the result of confidence maps predictions over the image plane for each joint (Figure 5). As more confident the system is about the location of the joint as more intense will be the color at that coordinate point. As a result, it shows with a heatmark where is the specific joint on the image.

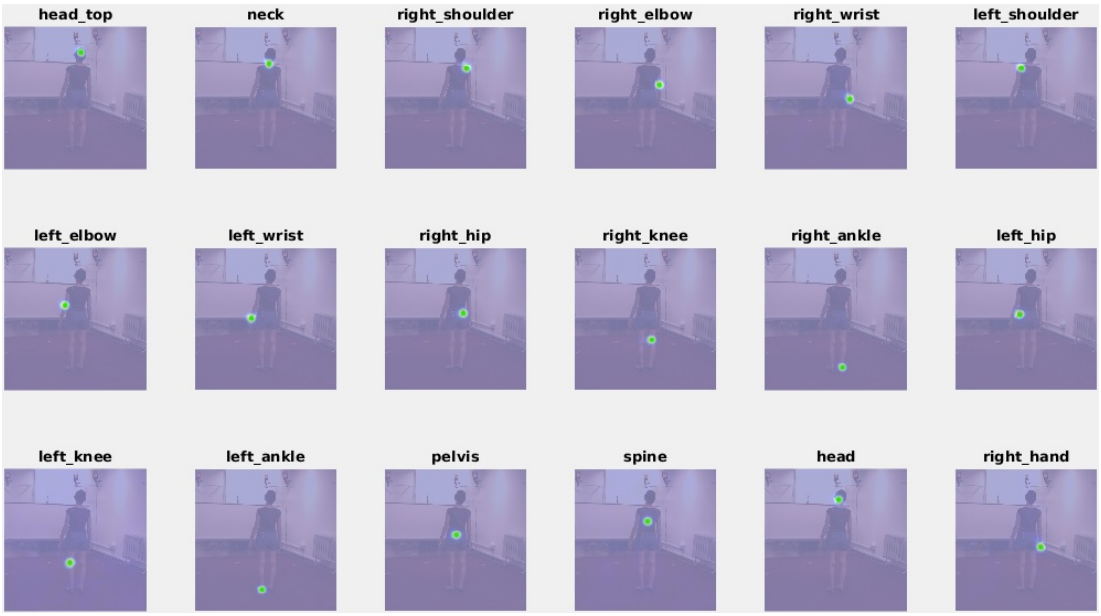


Figure 5: Heatmaps produced by VNect [16].

3.3. 2D Methods

The scientific community achieved a lot of progress in 2D PE and as a result, has provided a big variety of methods for solving this problem. Most of these methods share the same implementation principles. For example, utilizing the hourglass design [17], its modifications and different combinations of those. 2D methods can be divided into 4 categories described below.

3.3.1. Image based single person PE

The overwhelming majority of 2D methods refer to a single person estimation from a single RGB image [18, 19, 20, 21, 22, 23] and they are particularly targeting to improve the state-of-the-art methods and outperform them in efficiency and accuracy by fighting ambiguities, occlusion, and overlaps. These methods in one or the other way rely on the geometrical constraints of the human body. This data can contain any information which will help to have a better understanding of body geometry and its limits. For

example, human joints like knees and elbows cannot bend in all directions. It means that they have a specific range of available movements which is forming a structural constraint. Also, it can be the hierarchical structure and connections between different body parts [18] or a bone-based description which contain the information about the size, orientation, scale of the bones, etc.

Some methods [18, 20, 23] use those structural constraints as priors which are embedded in the process of training of the deep neural network (DNN). As a result, the trained network "understands" which poses are plausible and it leads to accuracy improvement. The others like Ke *et. al.* [19] encode the skeletal structure of the human body into the loss function, so it would try to balance the loss in each joint individually and the loss of entire structure as a whole. At the same time, Chou *et. al.* [22] makes it even more interesting by stacking two absolutely similar hourglass models [17] at the training stage. The first one also called the generator is used just for pose estimation, while the second one named a discriminator, thanks to the adversarial training with the geometrical priors, can distinguish if the pose provided by the generator is feasible. In the end, it returns the feedback to the generator about the result that could be improved.

3.3.2. Image based multi-person PE

At the same time, there are methods which can estimate multiple human figures from the single RGB image. In the images with more than one person, humans often overlap each others' figures. As a result, the pose estimation complexity grows because it is important to assign correctly all of the body joints of the same person to one skeleton. Thus accurate pose estimation is the aim of the majority of the multi-person methods [24, 25, 26, 27, 28].

Pishchulin *et. al.* in his work [27] presented one of the most accurate approaches for 2D joint extraction called DeepCut which was erelong improved to the DeeperCut together with Insafutdinov *et. al.* [28] using more powerful residual neural network (ResNet) [29] and improved optimization process. Nevertheless, the main process of solving the above-mentioned problem remained the same and is based on linear integer programming principles. First of all, from a single input image with multiple people DeepCut generates the hypotheses about the body parts. It means that the algorithm starts from a set of estimated joints, so-called body part candidates and body part classes, e.g., head, back, shoulder. Every element of body part candidates receive a value which allows to calculate how much does the candidate correspond to some specific class. In the next step, which is very similar to the first one, body candidates and classes are taken in pairs. Then these pairs get their own cost values which tell if the candidate from some specific class and the other candidate from the other class belong to the same person. In this way, body part hypotheses are formed. Later, all the body joints that were defined as those that belong to the same person are connected into one cluster based on the image. After that, inside each cluster, the joints are labeled according to their classes which were assigned earlier.

At the same time, there is another popular strategy presented by Cao *et. al.* [26] often called OpenPose. To be able to distinguish one person from another the method is using part affinity fields (PAF) which are vector fields that contain the location and orientation of the human body parts in the image. To predict the 2D joint locations the

algorithm uses FFN which calculates a set of confidence maps for each body element, and a set of affinity fields, one for every bone, which is defined by 2 endpoints. These predictions are calculated within multiple stages. Before that, a set of feature maps is formed from the input image. It is used to generate the first set of confidence maps and PAFs. Then in every new stage, the original feature maps are combined with confidence maps and affinity fields from a previous stage in order to get more accurate results. After the joints are detected the PAFs are used to define their belonging to the same limb and then to the same person. This process produces the final skeletons for all people in the input picture.

Meanwhile, to improve the results, some methods use the bounding boxes (BB) to define the areas where each person is. This helps to increase accuracy. But as it is expected, there are problems like incorrect localization of the BB which leads to the estimation problems. Fang *et. al.* offers a regional multi-person PE method [25]. In the beginning, the spatial transformer network is used together with the single person PE algorithm to provide a highly accurate BB and to generate pose proposals. To optimize this process, at the training stage one more single person PE branch is used. After the pose proposals are done they are led to parametric pose nonmaximum suppression block which cuts out all of the unnecessary data.

Some of the methods which are targeted for multi-person estimation can be also used in cases with one person. For example, the method offered by Nie *et. al.* [24] extract the parsing information from the input data with an eye to teach the PE model so it could adapt the result according to new parsing data. It helps to fix invalid locations for joints in single and multi-person cases which lead to the formation of more accurate joint candidates and hypotheses about the bodies in cases with multiple people.

3.3.3. Video based single person PE

Altogether there is a big variety of methods that are working only with images. However, there are some methods that can also process the video which is basically a sequence of images. It might sound simple but this process is more massive and needs more computational resources, so that it could calculate the output for each frame fast enough. In addition, the work with video has its own side effects like flicking or image blurring due to the motion and other problems.

Methods like LSTM Pose Machines [30] offered by Luo *et. al.* provides a solution for this kind of issues for videos with only one person in it. They transformed the CNN into the RNN by using a specific way of sharing the weights inside the network. RNN is able to do the calculations at each step taking to account the results from the previous ones. In this particular case, it also allows to reduce the number of connections in the transformed network. As a result, the processing time for videos improves a lot. In addition, Luo adds Long Short-Term Memory (LSTM) modules between frames processing which allows to "look back" for the previous calculations even further than the RNN can do without it. LSTM decrease the amount of parameters in the network and improve the result because the output of already computed frames is processed and stored to LSTM and used for next frame calculation adjustment. Overall, the offered schema sees the geometrical connections between the frames and it leads to fast and reliable results.

3.3.4. Video based multi-person PE

Some methods, like earlier described Openpose [26, 31], can also perform multiple estimations for video sequences. Due to the complexity of the problem in these circumstances Xiao *et. al.* [31] is looking for as simple as possible but still effective configuration. Their method uses the regular ResNet [29] supplemented with a few deconvolutional layers. This simple solution ended up to be quite effective for obtaining the heatmaps from the feature maps. For better reliability, they also use the BBs and track human movement through the frames using optical flow.

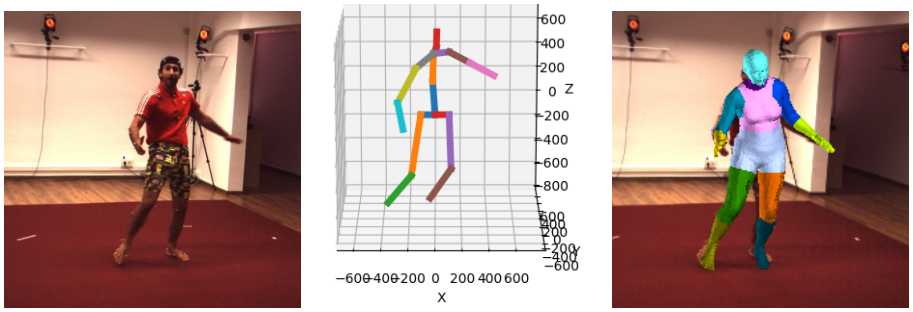
Nevertheless, the output of these methods is a 2D pose of the human skeleton which consists of 2D locations of each joint in the structure. That is why one of the biggest problems of 2D methods is ambiguity. A lot of those 2D poses can be interpreted in many different ways since different 3D poses can have similar projections to the 2D plane. Thus 3D methods are often used to solve the ambiguity issue.

3.4. 3D Methods

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 keypoints [16, 32, 33, 34] or use pre-calculated data [35] 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 keypoints. However, there are also methods that go directly to 3D PE [36] without any intermediate calculations of 2D poses.

3.4.1. 3D pose estimation

For the great number of 3D methods recovering the 3D pose, shown in Figure 6b, is the main goal [16, 33, 34, 36, 37, 38, 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

Figure 6: Results of 3D methods.

A good starting point for this category is a method called Lifting from the Deep (LFTD) offered by Tome *et. al.* [34]. 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.

LFTD claims that the geometric priors and constraints of the human body are necessary for a good approximation of the body shape. In the offered CNN the method is modified in the way that it has an embedded layer of 3D parameters. This allows to make sure that the predicted 3D model will be satisfying all the geometrical parameter and constraints.

The architecture of the framework presented in Figure 7 is complicated and consist of multiple stages which are connected successively. At the same time, all of them receive the input image. At each stage, first of all the input image goes through the CNN to extract features from the input image. These features are then used for joint location prediction. Thus the CNN outputs predicted heatmaps with 2D joint assumptions. At the next step, this information is used together with probabilistic 3D pose model to give a volume to the 2D coordinates and get the 3D pose. After that, the 3D joints of the probabilistic model are projected to the 2D surface aiming to obtain a new 2D joint location heatmaps. These belief maps projected from 3D joints are merged with the ones that were computed earlier in order to calculate the 2D loss. The merged set of heatmaps goes to the next stage as the reference for joint prediction recalculation and creation of a new set of belief maps from the original image and reference data. By repeating this procedure over and over through multiple stages the framework accuracy for 2D and 3D keypoint location raises progressively. This structure allowed LFTD method to reach state-of-the-art results.

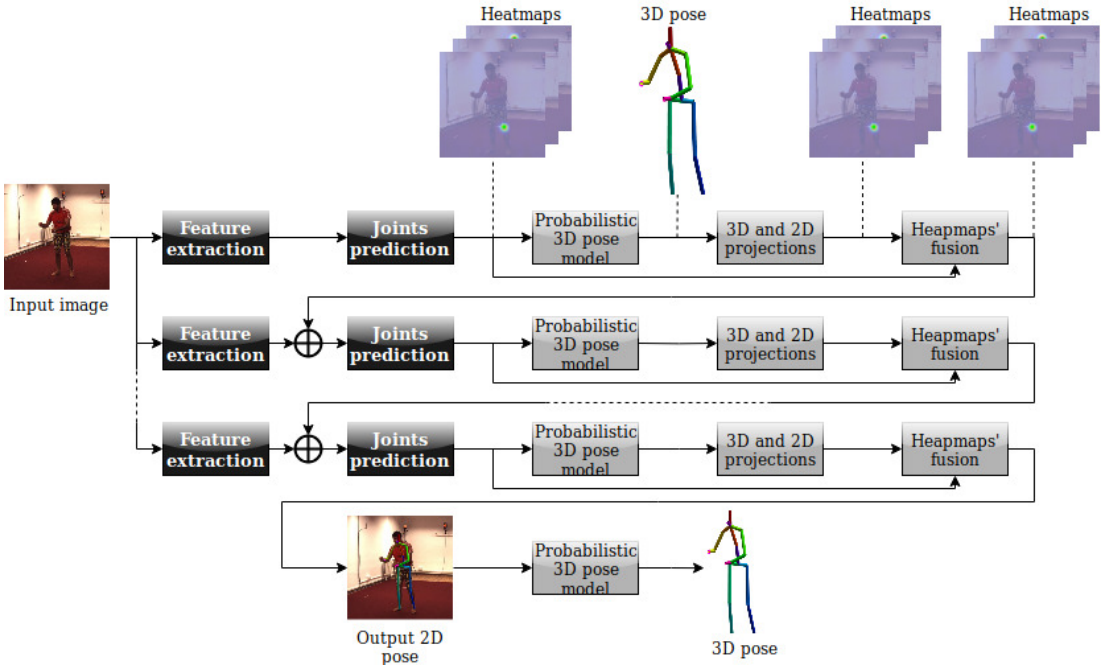


Figure 7: LFTD architecture.

The other interesting method whose aim is to recover the 3D pose is called DRPose3D and was proposed by Wang *et. al.* [33]. The architecture of this approach is easier than in the previous method. First of all, because this technique is based on the concept of depth, which is so easily perceived by humans, and can be learned by machines using NNs. The overview of the system is given in Figure 8.

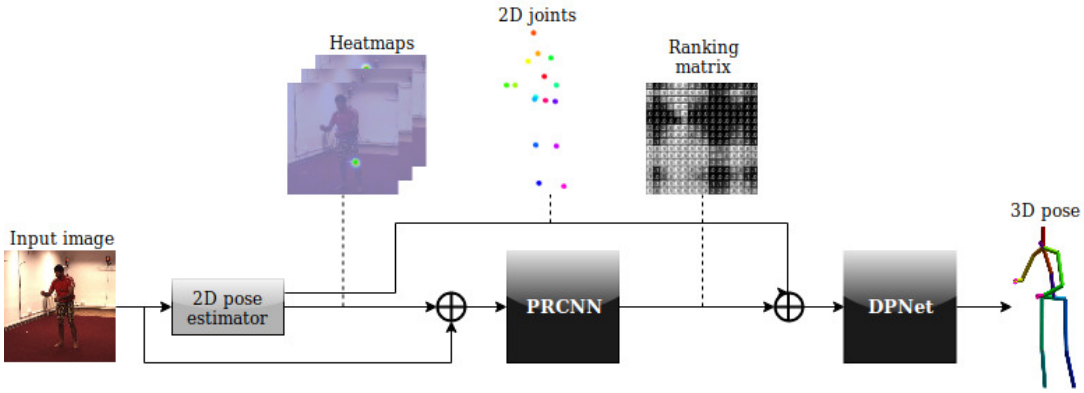


Figure 8: Pipeline of the DRPose3D method by Wang *et. al.* [33].

The input image is run through the 2D PE module which outputs 2 types of data: a set of heatmaps and the 2D keypoint locations. After that, the heatmaps paired with the input image are going to be processed by the newly offered Pairwise Ranking CNN (PRCNN). This network computes a so-called ranking matrix which reflects the location of the joints relatively to each other in terms of depth. Finally, this depth ranking is given to the 3D Pose Network (DPNet), also offered by authors. This network reconstructs the 3D pose from 2D joint locations given by 2D pose estimator, taking to account the depth data from the ranking matrix. Depth based approach allows DRPose3D to avoid pose ambiguities especially around the limbs and get more accurate angles between the body parts.

Another good example of 3D methods is an approach offered by Mehta *et. al.* [39] who decided to perform the 3D PE and at the same time create a new dataset called MPI-INF-3DHP for training PE systems.

The pipeline of the PE framework shown in Figure 9 consists of 3 steps. Within the first step, the input data is lead to the fully convolutional NN called 2DPoseNet which outputs the 2D heatmaps. These location maps easily provide the 2D joint locations to extract its maximum value from each of them. Then calculated data is used by the BB detector to locate the human on the image, root-center it and finally crop the picture. Next, all of this data goes to the second step where 3D PE is done by the other CNN, 3DPoseNet, that also uses a new supervision technique developed by authors for accurate regression. As the final step, the 3D and 2D data obtained from previous steps is aligned in order to get a global pose of the human with respect to the camera coordinate system.

The approach offered works well with the data from simple RGB cameras and achieves promising results. To make it possible, good data for training is a critical component. Existing datasets did not satisfy the requirements of the authors so they created their own MPI-INF-3DHP dataset. The data was captured with the state-of-the-art mocap system that contain a ground truth (GT) 3D annotations captured from

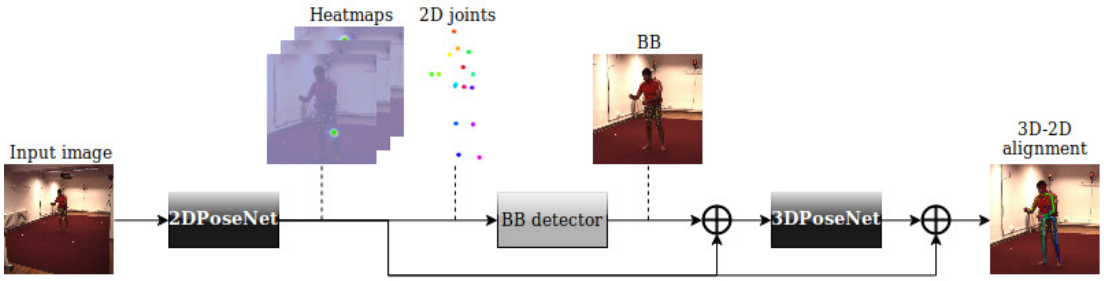


Figure 9: Architecture of method offered by Mehta *et. al.* [39].

different angles and in different conditions. The dataset created allowed the presented method to reach state-of-the-art results. MPI-INF-3DHP dataset has gained popularity among researchers [16, 36] and is a common tool to train and test PE systems under development.

One of the most interesting methods for this theses is VNect created by Mehta *et. al.* [16]. This approach goes further than most of the others and have 3 main key points: 2D and 3D joint extraction and kinematic skeleton fitting. The last stage allows using the method in the real-time character control applications, like games, virtual reality and so on. The goal of this approach is to create a tool for character animation which would not require the usage of the sensor mocap system and would work in real time using a simple RGB video sequence.

The whole pipeline given in Figure 10 can be divided into 2 parts. The first stage is responsible for the reconstruction of 2D and 3D keypoints. The CNN that performs this operation is fully convolutional mean what it can work without close-ups, so there is no need in using BBs. Nevertheless, for the result improvement, the image is centered on a human and cropped. The 2D coordinate regression is using the aforementioned heatmaps which are calculated for each joint independently. As for 3d reconstruction, the authors found an unusual solution. The heatmaps from 2D reconstruction are broadened to obtain the heatmaps for each of 3D coordinates. These are called the location-maps because the color intensity shows how close or far the joint is from the point of origin where x, y, and z coordinates are equal to zero. Pelvis is considered to be a constant point of origin. Then, the location-maps in conjunction with the 2D heatmaps are used for the prediction of the 3D coordinates by finding the points of a maximum correspondence using the loss function.

As for the last part, which is the kinematic skeleton fitting, the authors optimize, filter and smooth the regressed 2D and 3D data about the joint positions in order to get feasible character poses from frame to frame and as a result smooth and trustworthy animation.

The other interesting method suggested by Dabral *et. al.* [36] in comparison to all of the previous methods do not make any intermediate calculations of 2D poses and go directly to 3D pose estimation, but it does not mean that the offered method has nothing to do with 2D keypoints at all. For 3D pose reconstruction from a single image, the CNN called Structure-Aware PoseNet was presented. It was trained on data with 2D and 3D annotation. To make sure that the predicted 3D data will fit to the 2D landmarks and stay realistic, authors developed a loss function which is based on the anatomical structure of the human body. It consists of three components. The first and

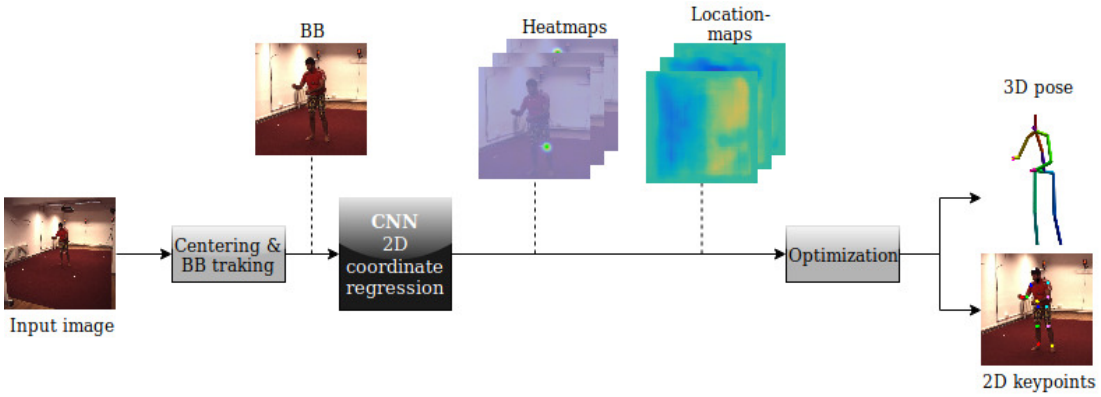


Figure 10: The architecture of the VNect [16] method.

the most important is the length of the bones which is the basis of the loss function. The other two - joint bending limits and symmetry of the left and right body sides - are supplementary. This structure allowed the method to outperform state-of-the-art methods.

Moreover, the authors did not stop here and offered to extend their approach to the video sequence by creating a temporal network which tracks structural hints from frame to frame to make the prediction more smooth and realistic. In this way, it takes the 3D prediction from the Structure-Aware PoseNet and compares to the predictions from the previous frames. Then, the network can adjust it a bit if there is a need. The output of this whole process can be applied to the kinematic skeleton to create an animation of any kind. Coupled together, the presented networks produce great estimation results which are good enough for character animation.

Some methods do not care about the 2D coordinate recovery and use 2D joint locations already as an input to their system. Their 3D pose restoration is based on the same idea of anthropometrical landmarks that were mention in Section 3.3. A good example of those methods are given by Ramakrishna [37] and Akhter [38].

Ramakrishna *et. al.* [37] have suggested to restore 3D pose from 2D landmarks. It is claimed that anthropometrical landmarks, for example in this particular case limb length, are very valuable prior which can help to avoid implausible poses. To make use of it, the overcomplete dictionary which contains the description of the limb length and deformability variations has been created. As a result, it allowed recovering the 3D pose of the human together with the position of the camera relative to the person.

Akhter *et. al.* [38] used the same idea of anthropometrical landmarks as Ramakrishna but instead of limb length they studied joint limits to form a prior for possible configurations in order to avoid unnatural deformation of the joints in the extracted skeleton. The work has started from collecting the data from the motion capture dataset and formation of the geometrical prior by learning the joint limits and how much they influence the pose. Nevertheless, even with the correct joint angles there still might be a few possible pose configurations, thus it was recognized to be important to restore the camera parameters. To solve this task, it was assumed from the learned angles that the torso part of the human body is having fewer variations than the whole body. This finding allowed to make the parameter estimation easy and as a consequence allowed the method to reach great results in 3D PE.

3.4.2. 3D pose and shape estimation

To make the 3D pose estimation more accomplished and for easier visual perception some methods together with pose aim to restore also the shape of the human body as illustrated in Figure 6c. Shape restoration complicates the whole process because depending on the volume of the body segments some parts can interpenetrate to each other which cannot happen in real life.

A great example in this category is the method called SMPLify [35] offered by Bogo *et. al.* It is based on the idea that a lot of information about human body shape can be withdrawn from a set of 2D joints. This information applied to the 3D model in combination with an anti-interpenetration technique allows to restore 3D body pose and shape avoiding ambiguities.

The whole method can be divided into two main stages: extracting a set of 2D joints from a single image using above-mentioned 2D PE method for joint extraction called DeepCut [27, 28] and restoring body shape and pose by mapping obtained 2D joints to a skinned multi-person linear 3D model of a human body (SMPL) [40]. This model represents a high-quality 3D-model of an average human body. In addition, SMPL takes to account how the body shape changes depending on the pose.

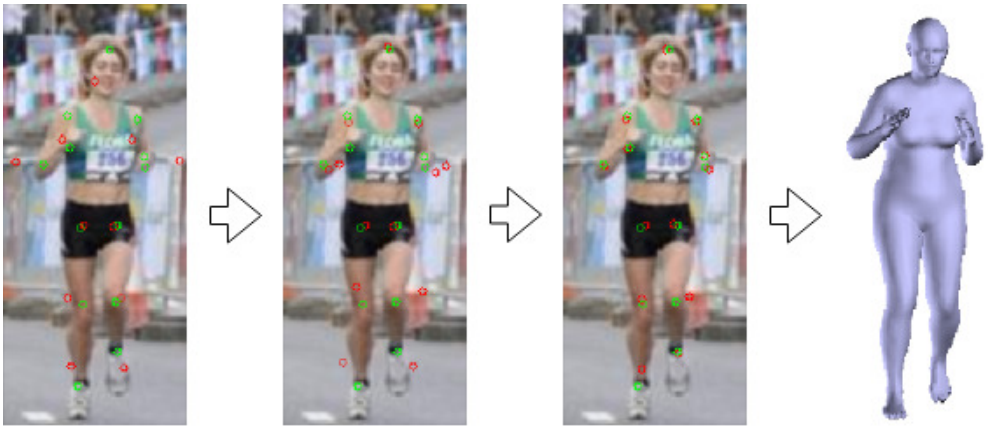


Figure 11: Visual representation of the objective function minimization performed by SMPLify.

When the 2D data is received from DeepCut, SMPLify goes directly to the mapping stage where the CNN minimizes the objective function. It means that SMPLify tries to make joints of the 3D model, projected to the 2D plane with a perspective camera model, to be as close as possible to the detected set of 2D joints. As an example, in Figure 11 green circles represent 2D joint predictions made by DeepCut and red ones are the projected joints of the SMPL model. In that kind of procedures, it is important to pay attention to the number of joints because it might be different for the output of the 2D joint estimation method and the model, such as SMPL. For instance, DeepCut produces a skeleton of 14 joints while the SMPL contains 23 joints. To solve this problem, 2D keypoints have to be associated with the most relevant joints of the 3D model. Moreover, Bogo *et. al.* understood that still a lot of poses are not feasible due to the body parts intersection and interpenetration. Therefore, SMPLify defines the shape of the body with a set of so-called capsules to get a proper result. The parameters of the capsule are linearly dependent on the shape coefficients of the SMPL

model. As a result, this method achieved outstanding performance in comparison with other methods. Consequently, this approach serves as the basis for many other studies.

Methods offered by Kanazawa *et. al.* [41] and Pavlakos *et. al.* [42] pursue the same target as SMPLify. Usually, for this goal, CNN is trained using the training data with 3D annotations. Unfortunately, this kind of data often has bad quality and there is not enough of it for good training results. Luckily, there is a big amount of the images offered for training that have 2D annotation. Similarly to Dabral *et. al.* [36] whose method was described earlier, Kanazawa and Pavlakos found a solution in it. They propose an end-to-end framework that can restore not only the 3D pose but also the shape using the SMPL model directly from a 2D image. This concept allows to avoid double training process (first stage for estimation of the joints and second for fitting the SMPL model projections into the output of the previous stage) and compress the whole procedure to a single training stage, that allows to reduce the complexity of the framework.

Both methods adapt the same idea of minimizing the objective function or in other words minimizing the error between the projected keypoints of the 3D model and 2D annotations with an eye on the 3D pose and shape reconstruction with the incorporation of SMPL model.

Nevertheless, these approaches follow a bit different pipelines. Pavlakos *et. al.* [42] whose architecture is shown in Figure 12 use the CNN named Human2D at the early stage of their framework to produce two types of data from a single input image - silhouette and a set of heatmaps for each joint. This data is used as an input for two other networks. The heatmaps are used to find 72 proper pose parameters (x, y, and z coordinates for each of 23 joints and 3 global rotation parameters) for the SMPL nodes. And the silhouettes are needed for calculating the shape of the person. The obtained results are used together to generate the correspondence to the input image mesh. Finally, the produced mesh is projected to the 2D plane and optimized according to the keypoints. At the same time, this structure has built-in 3D per-vertex loss as a part of the optimization process. It describes the correlation of errors from vertex to a vertex in the mesh and helps in evaluation and improvement of the training process.

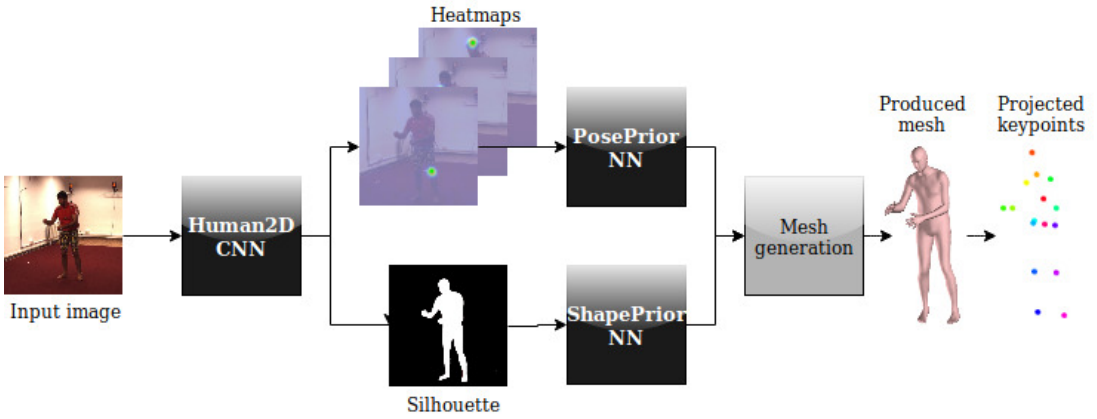


Figure 12: Architecture of method offered by Pavlakos *et. al.* [42].

Kanazawa *et. al.* [41] send the input through the convolutional encoder to the regression module as illustrated in Figure 13. In an iterative manner, it is looking for appropriate parameters for pose, shape, and camera that will minimize the reprojection

errors. At the same time, they are taking to account that even with this setup there might be ambiguous or even impossible postures. That is why they built in into the 3D PE pipeline the discriminator network whose task is to analyze provided SMPL parameters and determine whether they are corresponding to a plausible human pose or not. The discriminator is able to do that because during the training process it studies the available poses following the graph representation of the joints. Then it goes to each joint separately to learn its limitations. This process results in the geometry aware discriminator. This approach is giving very good results for 3D pose and shape estimation compared to other methods that work with 2D annotation.

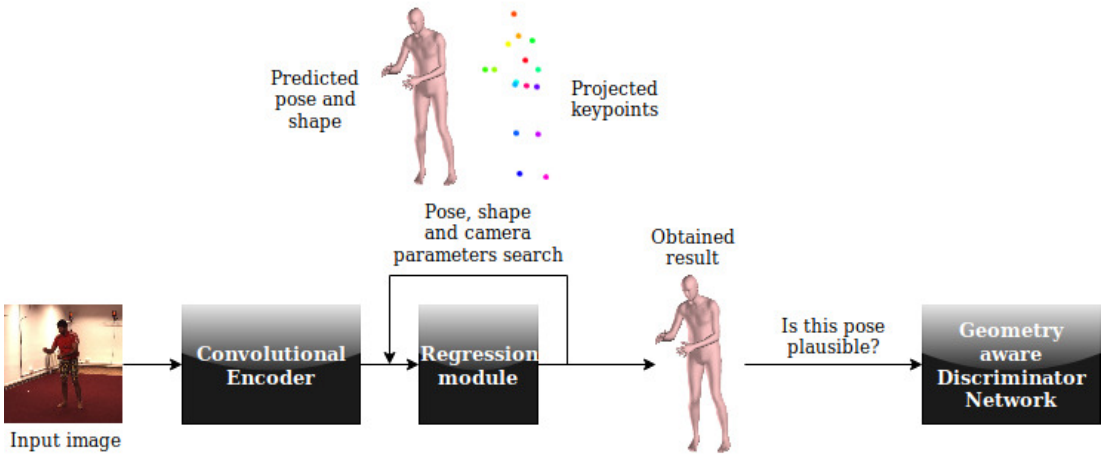


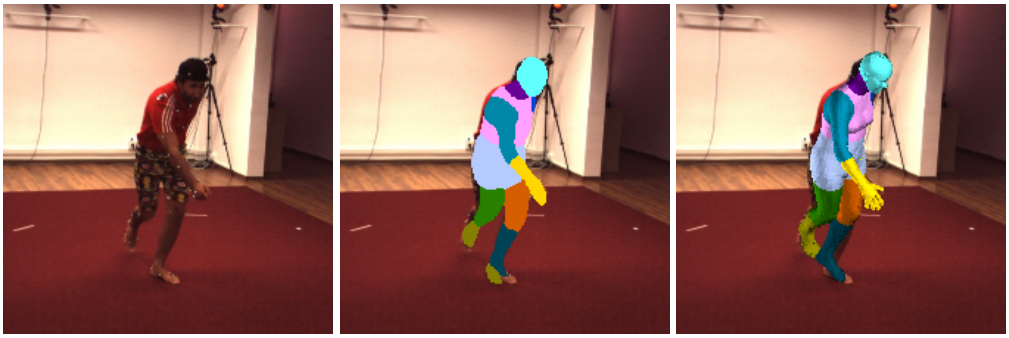
Figure 13: Structure of the end-to-end Framework offered by Kanazawa *et. al.* [41].

In contrast with previous methods, Neural Body Fitting (NBF) proposed by Omran *et. al.* [32] can be trained using data with 2D and 3D annotations which require weak or full supervision, respectively. Despite the aforementioned shortcomings of 3D annotation data, the authors want to demonstrate that there is no need in a big amount of 3D training data to get an acceptable outcome.

Its end-to-end pipeline is alike with the previous methods and also includes parameter search, SMPL mesh creation and further projection to the 2D surface. The difference is that for improvement of the performance of the framework the authors add a step for simplification of the image by means of semantic segmentation. The result of this step is illustrated in Figure 14b. The figure of the person is represented as a set of 12 colored areas where each color corresponds to a specific body part while all the irrelevant information is simply withdrawn. Omran *et. al.* assure that 12 segments are enough for a good restoration outcome. Nevertheless, the quality of the segmentation has a big impact on the final result.

Another difference is that for the optimization of the NBF, Omran *et. al.* offered and tested four different loss functions and its combinations to define how much the use of data with 2D and 3D annotations influence the result. They found out, that only 2D annotated training input does not give a good outcome because the system cannot fight the 3D ambiguities. A small amount of data with 3D annotations solves this issue and contributes to the achievement of worthy results.

The other approach worth of mentioning got the name BodyNet from its creators Varol *et. al.* [43] who decided to use four NNs each of which was responsible for



(a) Input image (b) Semantic segmentation (c) Predicted shape

Figure 14: Output of NBF method by Omran *et. al.* [32].

one of four tasks: 2D PE, 2D segmentation, 3D PE and 3D shape reconstruction. At first, all of them were trained separately so they could properly solve an assigned task. After that, the networks went through the general training together. In addition, this method uses three different loss functions each of which improves the reconstruction outcome. The BodyNet structure overview is presented in Figure 15. The input image is processed by two networks. One predicts the 2D keypoint locations and output it as heatmaps. At the same time, the other NN segment the body parts (there are 7 body parts, viz. hands, legs, head, torso, and back) from the image and deliver it also as a set of heatmaps, one per each body part. After that both sets of the heatmaps together with the input image are given to the 3D pose prediction NN which lift the 2D data to the 3D space relative to the camera perspective. In general, this procedure sounds similar to other methods, but also there is a big difference of this method from others because at this point it shifts from pixels to the concept of voxels. All of the further work is done in a 3D space defined by the voxel grid.

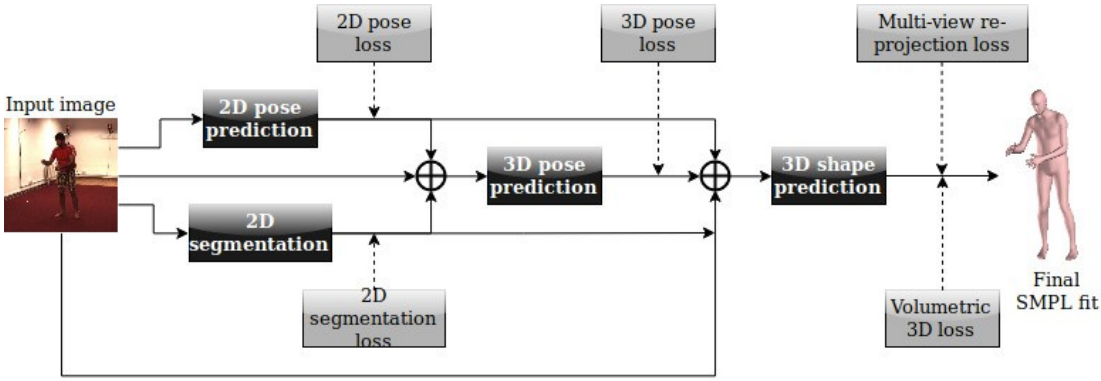


Figure 15: Overview of the end-to-end BodyNet framework [43].

In all of the previous steps, the loss functions are calculated. All three of them are the components of the multi-task loss function. They are collected together and along with 3D pose prediction and three other inputs go to the 3D shape restoration network. This module contains two more loss function: volumetric and reprojection losses for different planes. The volumetric loss allows to get a corrects alignment of the predicted shape with the segmentation data and get an accurate vision of body shape in space. The reprojection loss is used to improve the restoration of the limbs. Overall, BodyNet

structure is quite complected and massive but it gives outperforming results over the state-of-the-art approaches.

3.4.3. Other methods

Despite the fact that the majority of the 3D PE methods are aiming to get the 3D pose and sometimes the shape, there are methods which are also looking for a solution to some other problems that appear while the procedure of 3D reconstruction takes place, for example, too large training datasets or quantization errors.

In rare cases, the method aims to improve the learning process of the modern 3D PE by fighting over too massive training sets [44]. It is done, first of all by creating the latent body model without any references to joint coordinates. To do that the NN uses four images of the same person which are taken from different angles so it could learn the representation of the person taking into account its geometry and volume. After that, the obtained model has to be mapped to a 3D pose. It is done by the other shallow NN which needs only a small amount of data to be trained. It works because the latent model already contains the data about the geometry of the person's body thus there is no need to dub it while mapping process. In this way, the presented system learns to assume what is the 3D pose of the person on the input image if a single view image is given. Such construction allowed this method to outperform the other methods based on DNNs, trained on big sets of annotated data.

At the same time, some authors as Sun *et. al.* [45] define fighting the post-processing and quantization errors as their main goal. From the above-described methods, it is seen that often they owe their success to heatmaps but its use leads to aforesaid problems. The authors of the current method have found a solution to these issues in simple integral calculations. Thus by changing a simple operation of maximum extraction from heatmaps for joint regression to the integration of joint locations based on its probabilities, they create a simple and parameter-free way to overcome mentioned problems. In addition, this small but effective solution can be integrated into any architecture that uses heatmaps to improve its accuracy.

Finally, DensePose offered by Güler *et. al.* [46] represent an unusual method of estimation by mapping the RGB image of a human to a 3D model. To do that, first of all, a GT dataset that can be used to train the CNN so that it could map the data properly to the 3D surface model of a human body needed to be created. Thus the system starts with the identification of the body parts at the input image and its segmentation. Then, each segment is filled with points at the equal distance from each other and the system has to spread them in accordance with the shape of the surface. Thanks to this approach a proper point mapping to the 3D model can be performed successfully.

The authors of DensePose went further and proposed to use the method not only for creating the 3D body representations but also for generating synthetic images of a person while having only one image as an input [47]. Together with the input image offered, system gets the pose that has to be synthesized using the input data. To make it happen previously presented surface-based model is used to obtain the pose that corresponds to the input which is further remapped to the target pose. Moreover, this version of DensePose has a few additional modules, a combination of which gives the outperforming results in comparison to other image synthesis systems.

4. ANIMATION

The whole process of creating a realistic character animation based on real human motion from the point of view of the animation can be split into three parts: creating a character itself, generating the animation from the data produced by the PE methods and appliance of the motion to the 3D character.

4.1. Animation tools

Character creation and animation is a complicated process which involves a lot of skills and knowledge of specific software as Blender [48], ZBrush [49], Autodesk Maya [50], Autodesk 3D Max [51] and others, which are oriented to the professionals in the area. All of these tools have diverse sets of various instrument to work with all types of characters and objects. They are widely used by the industry professionals for creating different kinds of models and pictures, animated movies and VFX, video games and AR/VR applications, etc. Because of the high quality and multifunctionality of these products most of them [49, 50, 51] are expensive. Luckily, there are also free computer graphics (CG) toolsets that are available for everyone, for instance, Blender.

4.1.1. Blender

Blender is a powerful cross-platform, open-source software which can handle the whole creation pipeline - from character design and rendering to movie making and game development. Moreover, its functionality includes different kinds of simulators, various animation tools, instruments for use of mocap data, Python scripting and many others. For working with Python code Blender has a Python application programming interface (API) that provide the opportunity of working with native Blender libraries and tools. Moreover, this software has its own game engine, nonetheless, it will be removed from the upcoming release as it loses oneself in comparison with other, more powerful game engines like Unity [52], Unreal Engine, etc. Currently, Blender whose main screen is shown in Figure 16, is one of the most widely used 3D modeling tools.

Considering all of the above, Blender is a perfect tool for performing the implementation of the graphical part the thesis project: the creation of the simple character and its animation which afterwards can be transferred to the game engine.

4.1.2. Unity

Unity is a popular cross-platform game development environment. A great advantage over other game engines is that the users do not have to be good in programming to create their own game. Unity supports graphical development environment depicted in Figure 17 which make the whole process more open for the public. Nevertheless, it supports scripting with C# for event description and processing the users' actions. More advanced users can utilize scripting for physics description, and creation of visual effects up to implementing complicated artificial intelligence (AI) algorithms.

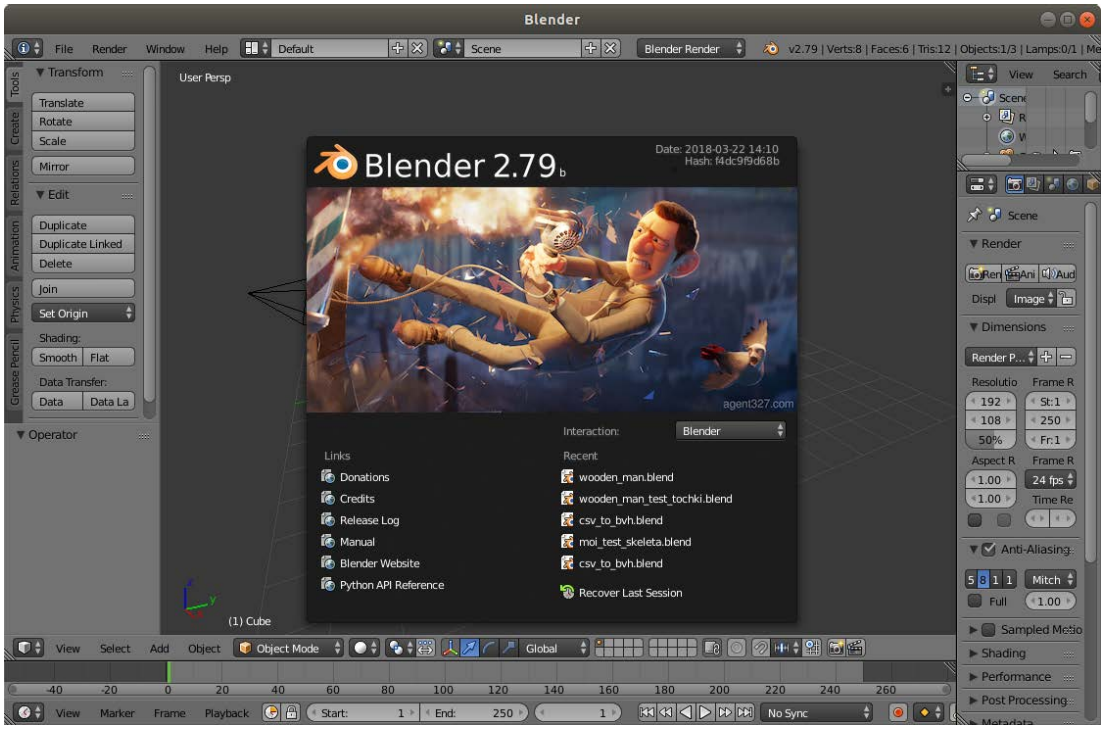


Figure 16: Blender splash screen.

Games created with Unity can run on different platforms whether it is a PC, smartphones, game consoles, etc. In addition, it is free of charge for personal use.

While Blender is getting concentrated on the character development and animation creation tools, Unity is targeted specifically to the game development itself, and thus it does not support any serious functionality to build the character, but offers to import it from external sources. Also it allows to import landscape meshes, textures, material and others.

4.2. 3D character design

One of the most popular techniques for 2D and 3D character creation and animation is called skeletal animation [53, 54]. It implies that every character consists of 2 main parts: the mesh and the skeleton.

4.2.1. Mesh

The mesh illustrated in Figure 18a defines how the character looks like. Sometimes it is also called the skin of the character. The only thing that limits its shape is the imagination of the character creator. At the same time, the mesh itself is just a shell, and a set of vertices and its animation for the game or a film might become a complicated process since the character artist has to animate all of the vertices, while some models can contain tens and even hundreds of thousands of vertices. A simple and effective solution for this issue is to use armatures.

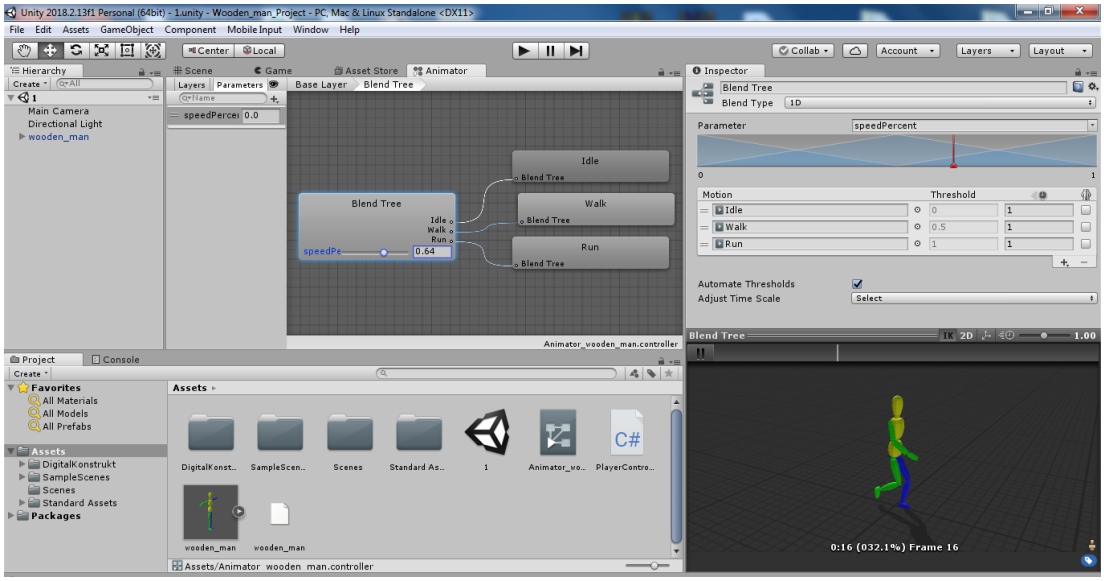


Figure 17: Unity graphical development environment.

4.2.2. Armature

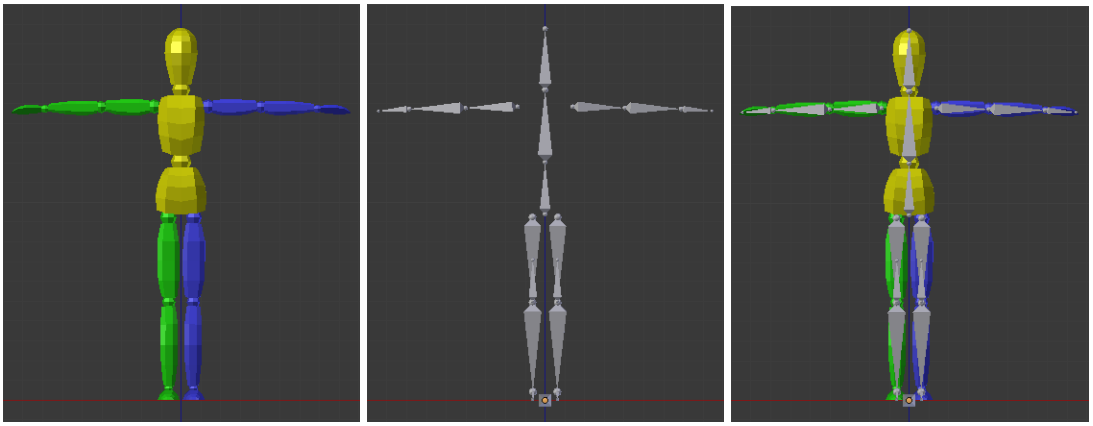
Armature shown in Figure 18b [55 Armatures] or a skeleton in most of the times is a hierarchical tree-like structure of elements which are called bones. In this structure, every next bone is connected to the previous parenting bone so the movement of the parenting bones will lead to the movement of the child ones. Usually, these skeletons of different characters look similar to a regular skeleton.

4.2.3. Rigging

The process of creating a character skeleton is called rigging [56, 55 Rigging]. Except for the bone structures, it might include constraints, object modifiers, and other components. The resulting armature is a rig that is used to control the mesh. To make it happen, the skin has to be linked to the skeleton of the character within the skinning process the result of which is illustrated in Figure 18c.

4.2.4. Skinning

During skinning [55 Skinning] the groups of vertices have to be associated with the corresponding bones in order to make it really move. Thereby, when a specific bone is moving, the assigned vertexes will follow it creating the motion. Usually, the vertices are assigned to the closest bone. Each of which has a vertex weight that shows how much influence this or another rig element has to each vertex. As bigger is the weight value as more the translation of the vertex will be dependent on the motion of the particular bone. At the joint locations, the vertices might be influenced by multiple bones.

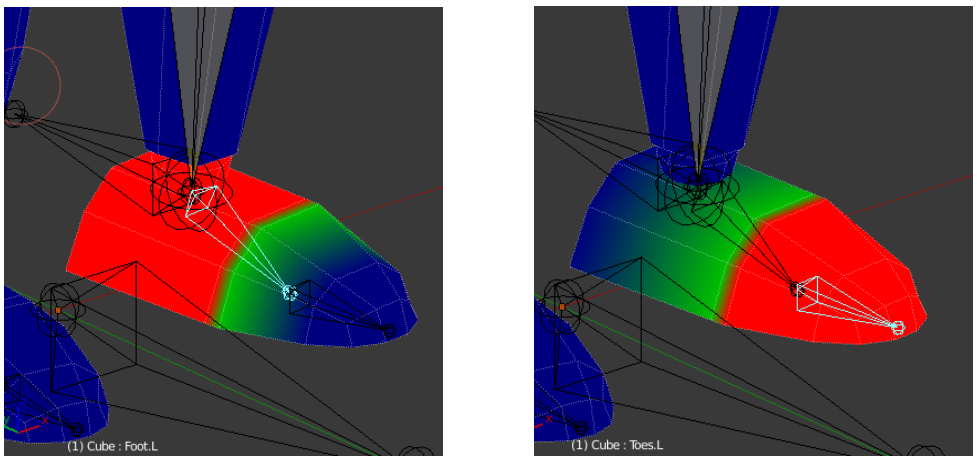


(a) Mesh (b) Armature (c) Skinned mesh

Figure 18: Elements of skeletal animation.

Automatic weight distribution sometimes can result in robotic and unnatural character movements. To fix it the skinning might requite manual assignment and distribution of the weights so that except the fully controlled groups of vertices the bones would also slightly affect the neighboring mesh areas around the joints. For instance, in Figure 19 red color means that the selected bone has a strong influence in all of the vertices of that area while for the green zone the bone's effect is not so strong, and blue vertices are not influenced by the chosen bone. So the joint area between the foot.L and toes.L bones is influenced by both of these bones. This procedure will help to create more authentic mesh deformations according to the skeleton.

In more advanced cases, for creating more realistic movement rigging and skinning there might also include the creation and connection of the muscles. As a result, the above-mentioned problem of multiple vertices, animation becomes as simple as the animation of the bones, which requires only location and rotation data. Thus, skeletal animation also decreases the amount of information that has to be saved since only bone motion data is saved and the location of the vertices in each frame can be calculated from there.



(a) Influence zone of the foot.L bone (b) Influence zone of the toes.L bone

Figure 19: Manually adjuster weight distribution between the foot bones.

4.3. Animation generation

4.3.1. Keyframing

Animation of the 3D characters is based on traditional animation principals and techniques [57]. One of the simplest and most applicable ways of animation is called keyframing [55 Keyframes] [58]. The idea behind this technique is that the animator creates only the main frames of the character motion which describe its movement within the timeline as illustrated in Figure 20. The keyframe description of the motion can contain coordinates, rotation angles or any other information that can help to describe the transformation of the object with time. The keyframes do not follow each other one after one so the in-between frames are generated and rendered by the software using interpolation methods. The appliance of this technique makes the character movement smooth and takes less time because there is no need to create a frame by frame animation.

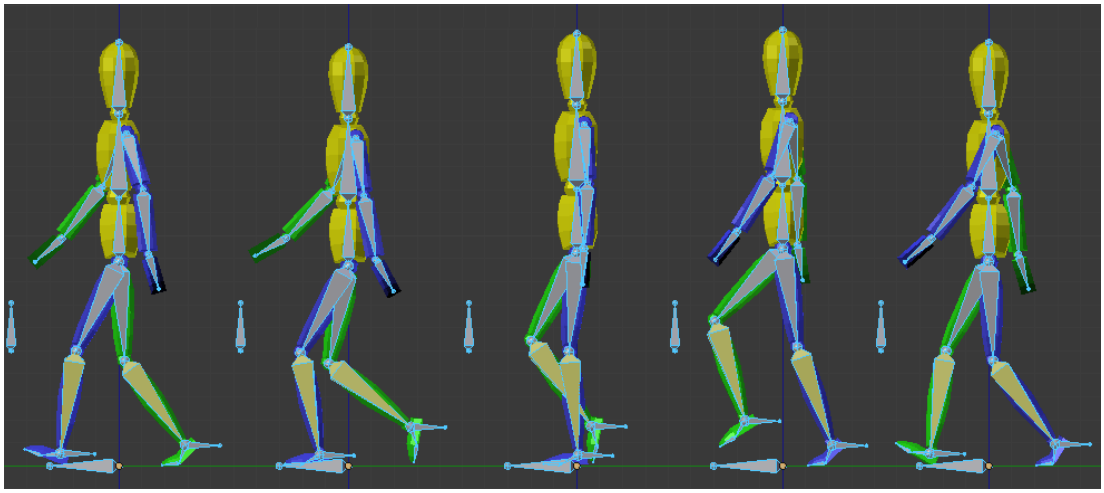


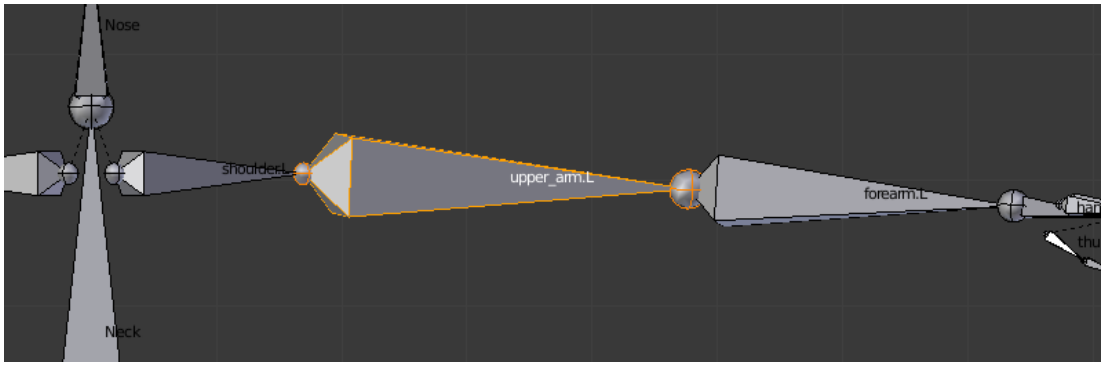
Figure 20: Keyframes for walk animation in Blender.

In the framework of this thesis mocap data for each frame of the video sequence is used as keyframes.

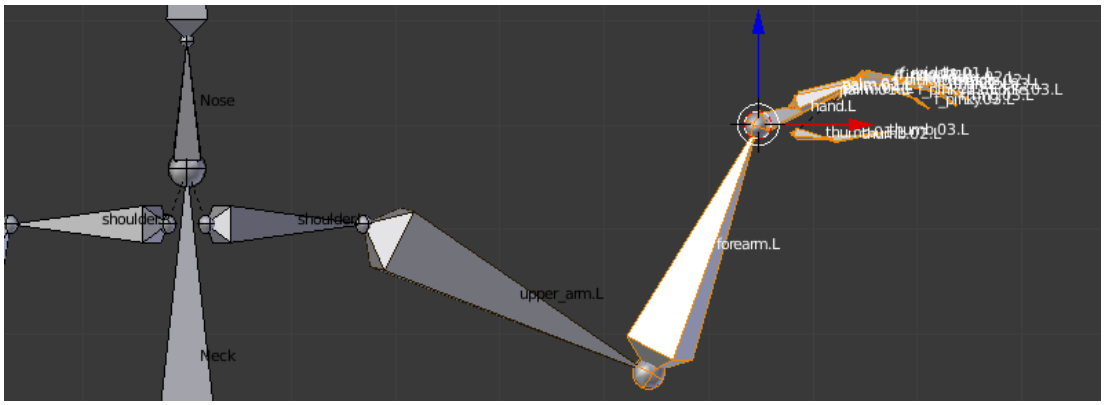
4.3.2. Using motion capture data for animation

In Blender, skeleton bones look like those presented in Figure 21. Each bone has a root or head and the tail. The tail of one bone might be at the same time the root of the next one. For example, in Figure 21a the upper arm bone's root is the shoulder joint and its tail is in the elbow. At the same time, the upper arm's tail is the root for the forearm bone which starts with the elbow and ends at the wrist. This structure creates a hierarchy of the rig so that the upper arm bone is a parent to the forearm, while the forearm is a parent to the hand and so on.

As a result, shown in Figure 21b, the motion of the children structures is dependent on the parents. It can be seen, the roots and tails of the rig bones often correspond to the real joints location. Nevertheless, it is hard to apply correctly the mocap data to



(a) Hierarchical structure of the left arm bones



(b) Parent-child motion dependency

Figure 21: Humanoid rig in Blender.

these joints because they are defined by the location of the bones which are basically the space between the neighboring joints but not the joints themselves. That can create a lot of confusion and incorrect data overlay. To solve this issue the empties can be used.

Empty or an empty object is a specific type of objects that are invisible, they do not have volume or surface or any other property of an object and thus cannot be rendered [55 Modeling]. The only features that describe them are their transformation settings. This makes empties an effective tool whenever it is needed to control and modify an object, texture, rig or any other elements of the project. Available empties are illustrated in Figure 22.

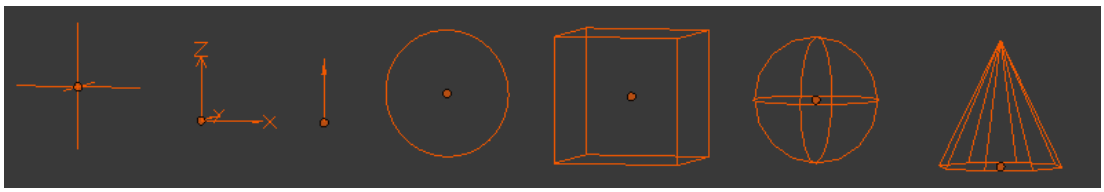


Figure 22: Empties in Blender.

Overall, empties are targeted for performing any kind of transformation and creating influences that have to be invisible in the final render, for example, a force field. One of the most popular ways to use empties is parenting. It means that an empty is used as a reference point for one object or a group of connected elements.

For this thesis project, empties can be used to parent the joints of the skeleton so they will be able to control this structure. The plain axis empties can be placed inside the corresponding joints as it is shown in Figure 23.

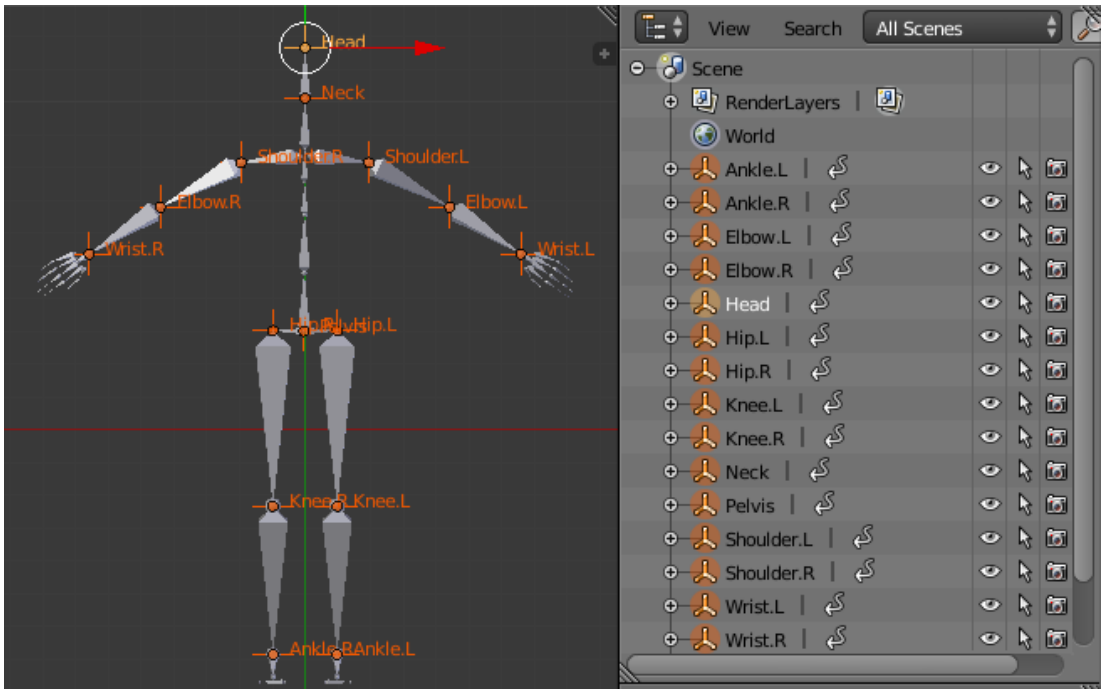
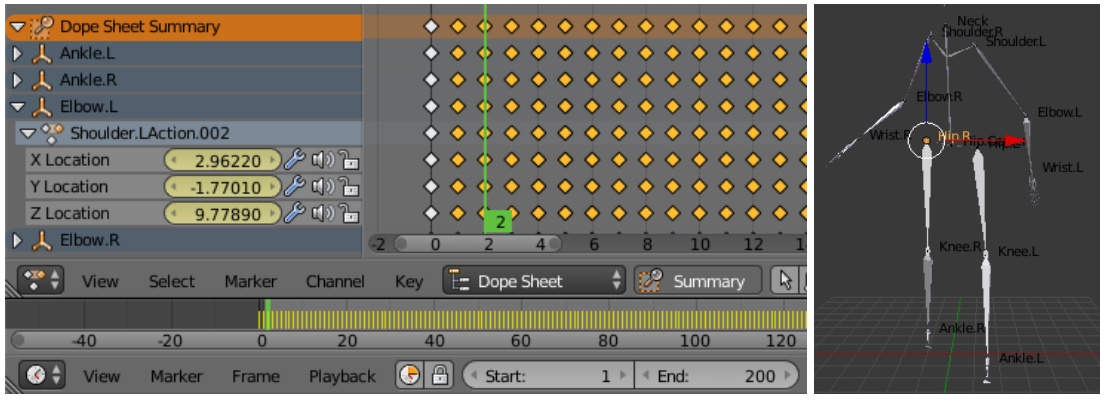


Figure 23: Empties bounded to the skeleton joints.

In this way, the data about the 3D location of each joint that is imported from the PE methods can be assigned to the corresponding empties. Each empty receives a list of x, y and z coordinates for every frame that is processed. The imported data can be seen in the Dope sheet Summary section as illustrated in Figure 24a.



(a) Dope sheet editor (b) Skeleton movement

Figure 24: Skeleton animation with empties.

The Dope Sheet [55 Editors] concept is also originating from the classical animation even though it had changed a bit with the advent of CG. The Dope Sheet contains the information about the motion of all of the objects in the scene: at which moment they appear, what is happening with them and for what period of time. It is using the

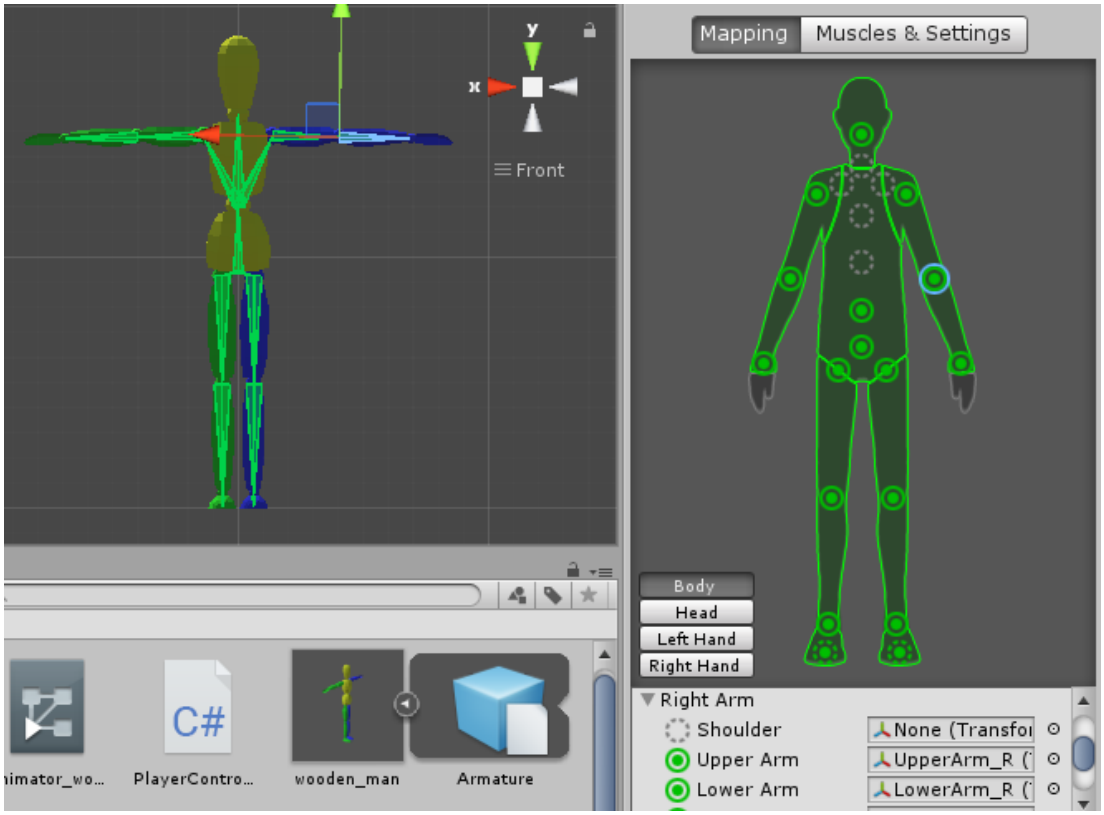


Figure 25: Manual avatar mapping in Unity.

keyframing technique for each separate element that is going to be animated within the timeline. Basically, every mark in Figure 24a is a keyframe for each empty connected to the joint and it contains the data for each action channel - X, Y and Z locations. Figure 24b demonstrates the corresponding animation for the frame number 2 that is marked in Figure 24a.

In this project, Figure 24 shows the ideal case when each joint empty has a keyframe value for every frame. At this rate, the imported movement will be as close as possible to the real human motion. If some of the frames are missing the data about the joint location in some specific time will be interpolated from the previous and the next available keyframes.

Finally, frame-to-frame animation can be saved as native Blender *.blend* file format or as an *.fbx* file and together with the character itself may be exported for the further use in Unity or other applications.

4.4. Character animation with Unity

As it was said before, Unity require importing the rigged models created by other means. So the character together with the all available set of animation for it can be imported right to the scene.

In Unity, human alike models are called humanoids. Unity has its own understanding of how the humanoid has to look like. Therefore, it has a set of requirements for that kind of characters. So one of the first things to do after the humanoid character

transferring is to match a hierarchy of the character with the Unity vision on how the humanoid has to be set up. It is also good to check these requirements before the import so that the structure of the imported character would fit the game engine environment. Otherwise, some problems can pop up after the character is transferred to Unity. All of this is needed to make sure that the animation of the model will be correctly overlaid with the character. It can be done automatically or manually using Mapping window illustrated in Figure 25.

Sometimes, it might be needed to disable the animation of the humanoid partially. Avatar masks are used for this purpose. They can define what parts of the character will not follow the animation and stay still.

To manipulate the animations, Unity has a complex Animation controller which provide a user with various functionality. It allows to create own animation schemes, import animations from other sources (Section 5.3.1), build a connection between different behavioral animations that allow making smooth translations during the gameplay from one action to another one, retarget the motion when the animation which was created for one specific character is applied to the other one, build different animation patterns for separate body parts using the avatar masks and so on.

5. IMPLEMENTATION

5.1. Framework implementation overview

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 26. 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.

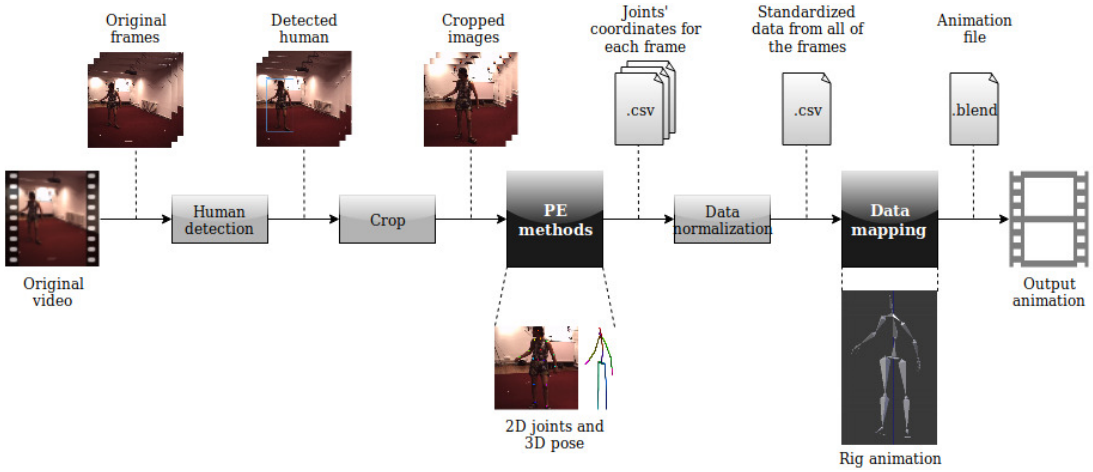


Figure 26: Implementation pipeline.

The cropping process has its own difficulties because in different frames the human can be located in different parts of the image. Hereby, first of all, it is necessary to detect where the person is in the image so that exactly the part with the human would be cut out. There are many ways how it can be done. The current project utilizes an algorithm called You Only Look Once (YOLO) [59].

YOLO is a NN that is used for detecting the objects in the image and providing its BBs each of which is given probability value. Those BBs that have the highest values are given as an output together with the assigned labels which say what kind of object is detected. For the thesis framework, only the BBs that are labeled as "person" are the objects of interest. Only for those the centers and the heights of the BBs are extracted. After that, the data provided by YOLO can be used to crop the image in accordance with the location of the human in the scene. Already human-centered and cropped image can be used as an input to the PE step.

In most of the cases, the output of the 3D PE step is a set of *.csv* files (comma-separated values). When the PE method provides the output in a different shape [32]

than it is needed, these results are processed to extract the necessary data which is then saved as *.csv* files. Each of these files contains the coordinates of the human body joints in the corresponding frame. This data further can be used for the animation but at this stage, it is still raw and the use of this data will make the process difficult due to the fact that the output from different methods might vary quite a lot in scale, centering and other parameters. Thereby, it had to be preprocessed before being used for animation purposes. Within this normalization process, all of the data is formatted following the specific requirements agreed for the project which will make the animation more natural.

In addition, to make the operation of associating the keypoint 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 27, 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.

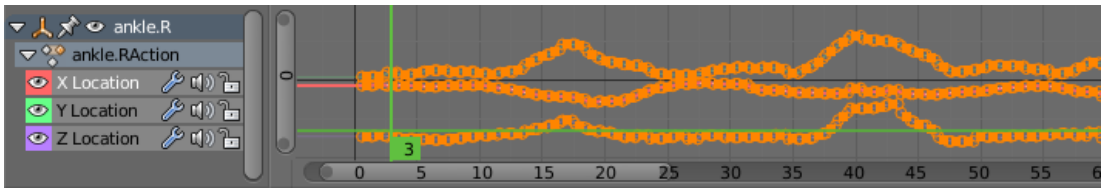


Figure 27: Separate x, y and z location channels of the right ankle.

To perform a proper mapping, every joint of the rig that is responsible for controlling it, has to be matched with the corresponding keypoint data from the *.csv* file but the amount the elements in those two sets of joints can be different. Thereby 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 (Section 5.2). 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 (Section 5.3).

Overall, the framework works automatically. It means that when the system is given an input video it will follow every stage of the process on its own and produce the standard Blender file *.blend* as an output. This file contains all required structures and information for its future use in Blender or other applications like Unity. Nevertheless, it is good to check the Blender file before importing it onwards and if it is necessary manually to adjust the animation. It might be needed because the PE methods are not perfect and produced data might rapidly jump between the frames while in the perfect conditions the animation should be smooth and stable.

5.2. Mapping and animation in Blender

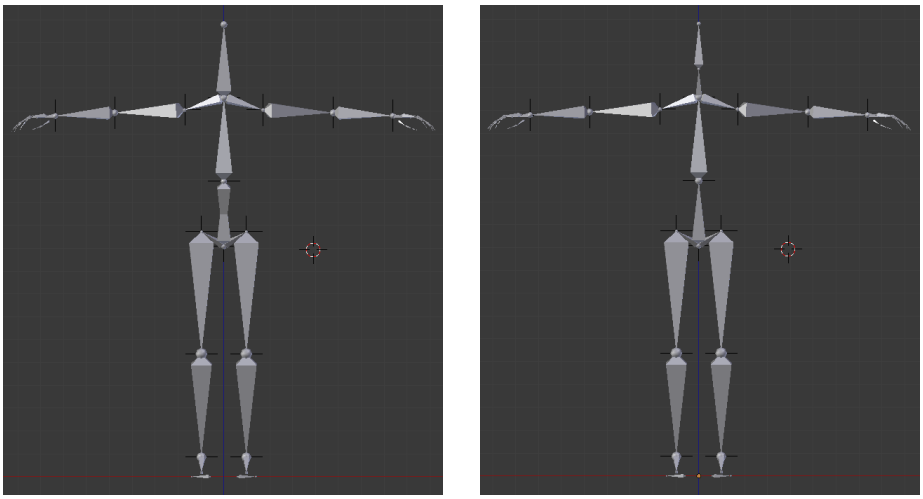
When the joint location data is imported to Blender and mapped to the rig joints it is kept as a set of keyframes. After the import is done they can be seen in the Dope Sheet Editor shown earlier in Figure 24a where every star depicts a keypoint.

On one hand, mapping can seem easy, on the other hand, when the imported mocap data is different from standard one provided by professional mocap tools (Section 2.3) it becomes complicated because the process requires to take into account what kind of data exactly is imported and then come up with a solution that transforms the nonstandard input into a proper animation file that can be successfully utilized in the future.

5.2.1. Overlay of 3D data to the rig

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 (Section 6.1) 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.

The first skeleton out of two is used exactly for the mapping purposes. In this project, it is called "Control_rig" and it is presented in Figure 28a. Because the data cannot be simply applied to the bones directly the plain axis empties are used (Section 4.3.2). The joints of the skeleton are connected to these empties using constraints (Section 5.2.2). Then, the data imported from the *.csv* files by columns is assigned to the corresponding empties. It can be done using Python scripting feature inside Blender's own environment, and thus the mocap data can be imported directly there.



(a) Control rig (b) Main rig

Figure 28: Double rig of the skinned model.

The second rig, illustrated in Figure 28b, is responsible for the mesh deformation and was called "Main_rig". In comparison to the controlling skeleton, the main rig does not really follow the empties but it repeats the motion of the first rig. That allows the main skeleton to keep the shape of the model and at the same time reflect the imported motion.

Depending on each particular case, the hierarchical structures of two linked rigs can be absolutely similar or have some differences as it is shown in Figure 28. However, it is common when the main rig is having more bones than the control one. At the same time, the controlling skeleton can also have some additional bones for proper distribution of skeleton control.

5.2.2. Constrains and modifiers

To perform the connection of the empties to the joints to the controlling rig and allow the main rig to follow the performance, Blender has a set of constraints that can be applied to the skeleton and some objects, like empties. Constraints provide to the user control over the motion by giving him tools for manipulating such properties of the object as rotation, location, and scale. In this particular case, a few constraints were applied to both rigs.

First of all, to connect the empties to the skeleton joints "Stretch to" and "Copy location" constraints are used. The first one permits the bone to stretch in the direction of an empty, assigned to the tail of this bone while the second one holds the location of its head at the position of the parent's bone tail. For example, the forearm bone will follow the motion of the wrist while it will remains connected to the elbow. At the same time if "Stretch to" has to be applied to all of the empties and corresponding to the bones, "Copy location" has to be applied only to the root element of the controlling skeleton hierarchy, and in this example it is a lower backbone which copies the location of the pelvis.

Now, when the rig is linked to the animated empties it can shrink or stretch following the empties' change of location, which is defined by the imported 3D coordinates. This acquired property of the bones is also very useful in the situations when the length of the same bone slightly changes from frame to frame due to the varying accuracy of the PE method. In addition, this can be used as a solution for the data of different scales.

One more crucial constrain for the controlling rig is called "Locked track", which allow to limit the available range of motion on some axis. It is used to keep the orientation of the back fixed regarding the shoulders and hips. If this is not done during the rotation of the character the back bones will not rotate enough and it will cause twist of the character's mesh as shown in Figure 29.

Therefore, it is very important to assign all of the rig constraints in a right way and order. Otherwise, the skeleton might move incorrectly and it will result in an invalid animation. An example is presented in Figures 29 and 30, where left images show how the rig and mesh should deform when all of the bone constrains are assigned properly while the right images show how a similar or the same pose correspondingly might look if some of the bones are missing correct constrains.

Correct control rig structure will lead to correct animation of the main rig because after the controlling skeleton is set up, it is used to manipulate the motion of the main

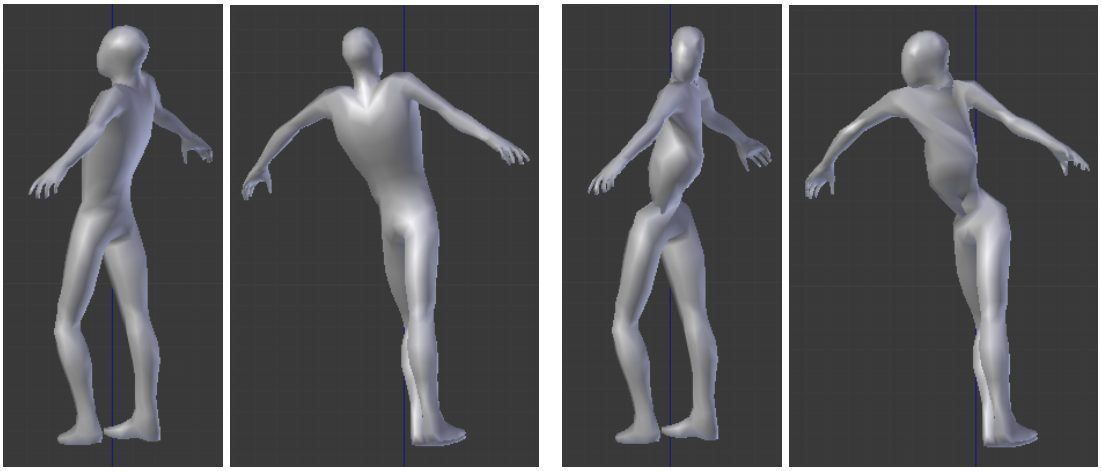
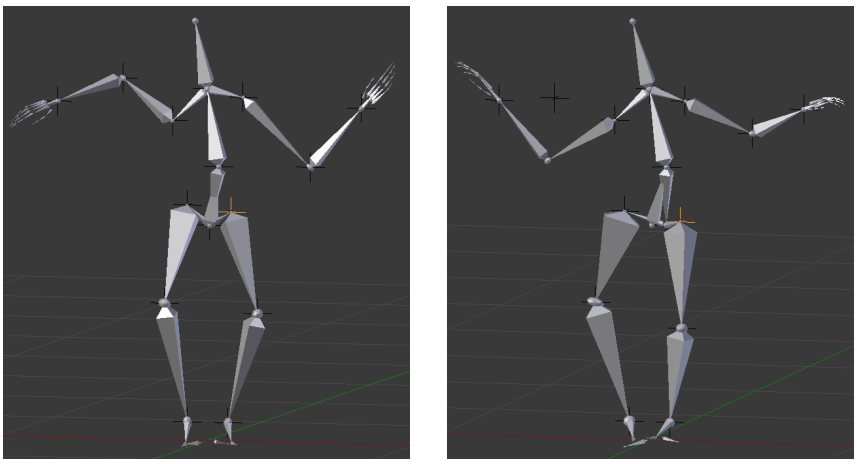


Figure 29: The effect of skeleton constraints appliance on the mesh. On the left images (front, side view correspondingly) constraints are applied correctly while on the right images (front, side view) the "Locked track" constraints are missing.

one. Exactly the primary rig is skinned to the mesh of the character. This whole process reminiscent of a puppet doll control and the places where the empties are connected to the rig are the points where the thread tied to the marionette.

In order to be animated, the main skeleton has to be connected to the control rig. This can be done through the same bone constraints but this time it is the one called "Copy Rotation". As shown in Figure 31 it allows to choose the corresponding bone from the target object, which in this particular case is the whole "Control_rig", and thus the bone from the primary skeleton will repeat the rotations of the bone from the puppet. This kind of set up lets the main skeleton repeat the motion of the controlling rig without corrupting own skeleton with such artifacts like stretching. Thus this method is also good for the characters which have different body proportions as the regular human being. For example, animation of the monkey whose legs are shorter but hands are longer will succeed as well as the animation of any other human character.



(a) Correctly applied constrains (b) Bad constrains

Figure 30: The result of constrains appliance.

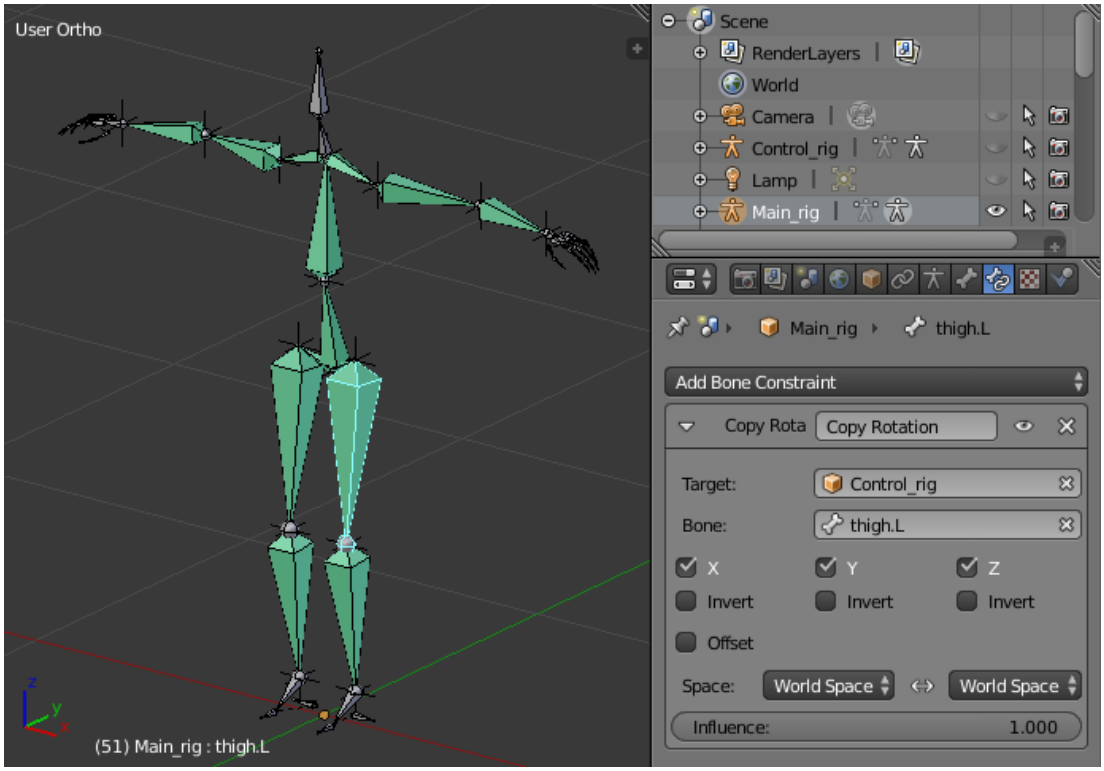


Figure 31: Bone constraints of the "Main_rig".

5.2.3. Animation baking

When the data is finally imported to Blender, the motion of each empty and its joint is a separate animation clip. It mean that empties are independent of each other and the motion of every element can be modified and removed without having any impact to other empties. For example, if to clean the motion data from one elbow and run the animation, all the other joints will keep moving according with the imported data as they did before, while the elbow will be simply following the motion of the hand because it is still a part of a skeleton structure and it simply cannot stay still if the shoulder and the wrist are moving.

This animation split happens because the empties are considered by the software to be separate objects from the rigs and the mesh. Even the outliner window in Blender, which shows the hierarchy of all elements in the scene, display the empties on the same level as the rigs themselves. In this way, because the data is first of all, imposed to the empties and only then the bones are connected to them, the animation clips are autonomous from other elements.

From one side, separate animations are good because then it is easy to adjust and modify the motion tracks of each separate joint. It might be very useful when the input data that is provided by the PE methods requires some adjustment. On the other hand, tools like Unity cannot even see these types of animation clips. However, to avoid these issues the animation can be baked.

Animation baking allows to connect independent animation tracks into one single piece of motion. The baked action can follow one after another which let the user to create more complicated and long animations by combining already existing short

motion clips. Animations also can run simultaneously, as it is done for the motion clips of the empties within the developed framework.

After the chosen joint animations are baked, they cannot be edited anymore. Nevertheless, if there is a need in manual adjustment of the animation it can be done by fixing separate joint motions. After it is done the actions will need to be baked again.

5.3. 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. Biovision Hierarchy Data (BVH) or *.bvh* format is a commonly used data type which contain the description of the rig hierarchy and its movements with time by holding the values of the keyframes.

Nevertheless, this approach for data animation is good for graphics software like Autodesk Maya [50] or Autodesk 3D Max [51] but when it is needed to import the mocap data into game engines like Unity it becomes quite complicated and requires the use of intermediary tools.

5.3.1. Ways of 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 Filmbox) 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 callback 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.

When the *.blend* file needs to be imported it can be done by simply dragging the file inside the Unity project to the Assets. During the import, Unity will automatically change some parameters of the model, so it would correspond to a new environment. It is done due to some differences between the two programs which are described below in Section 5.3.2. The elements which are depicted in the Assets folder as blue cubes can be dragged directly to the scene. In addition, imported in this way Blender projects keep the connection with its origin. It means that when the original *.blend* file which was imported to Unity is changed, it will also update inside the game engine.

In the situations, when the Blender scene is too massive, but it is required to import only some specific objects, it can be made using *.fbx*. However, even the whole scene can be transferred in this way. In this case, the user will have to set the importing parameters by himself and if some of them are wrong the whole project can be seriously corrupted. Thus, most of the further problems that will appear after the import to the game engine will be rooting at this point. Especially, it refers to the projects with ani-

mated models. Already generated *.fbx* files can be opened in Unity in the same way as the previous one. In comparison to the native file format, *.fbx* does not depend on the parenting project and will not be changed if the original Blender file will be updated.

In a nutshell, creation of the *.fbx* represents itself as a manual process, while native Blender files can be imported automatically. Taking into account the specification of the project, the use of *.blend* projects is more suitable and safe.

Nevertheless, it is impossible to say which one of these options is better because depending on the task it might be smart to change the preferences about the way of importing the Blender project.

5.3.2. Differences of environments

Unity and Blender have a slightly different coordinate systems. First one has the Y-axis as a vertical direction, while for the second one it is Z-axis. During the project import, Unity takes care of it and rotate the objects so they would look as it is needed. Still, during the manual import, it is important to keep this information in mind and set the appropriate import settings.

In addition, these two programs understand the scaling units in a different way. Blender as many other 3D packages measures the world and the objects in meters while Unity works with centimeters. It can seem like a simple unit mismatch while the scale remains the same. However, during the import, things might go wrong if the scale settings were applied incorrectly.

Based on these differences, it is good to prepare the Blender models before they are exported. For example, to make sure that the scaling factors are equal to 1 and the rotation of the objects is 0°, and set the point of origin to meaningful position regarding its object itself. This does not guarantee that the transition will be smooth but will reduce the probability of multiple errors and problems with the model after it is moved to Unity.

6. EXPERIMENTAL EVALUATION

6.1. Evaluation of 3D PE methods

From the list of 3D PE methods described in the Section 3.4.1 those which have their implementations available for the public and were the most suitable for the purposes of the current thesis were evaluated using the Human3.6M Dataset [7].

6.1.1. Dataset and evaluation metrics

Human3.6M is a massive dataset [7] of 3D human poses recorded from 11 subjects in different scenarios and scenes taken from 4 different view angles. Every configuration has 24 joint locations. The data is captured using the mocap technology in the controlled environment. The only difference with the regular procedure was that the actors were wearing their everyday clothes instead of the mocap costumes. This approach to the data collection contributed to obtaining the data close to reality. The dataset provides different types of data such as simple RGB images, time-of-flight data, silhouettes, 3D pose and high-quality 3D scans of the subjects.

This dataset is a great tool for training and validation of different NNs, and thus it is very popular among the researchers [16, 32, 33, 34, 35, 36, 42, 44, 45]. For such purposes, it provides the training data for 7 subjects in 15 different scenarios with the corresponding ground truth data. For the testing purposes, the data for 5 subjects is available.

As a validation metric, Human3.6M offers to use Mean Per Joint Position Error (MPJPE). The metric illustrates the difference between the ground truth data and the predicted values of joint coordinates as an average Euclidean distance (in millimeters) for all corresponding pairs of joints. For the better match, both datasets have to be aligned at a certain point. As smaller is the final value, as closer the calculated result to the GT.

6.1.2. Evaluation results

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 [34] to 25 [35] 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 PE 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 process

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.

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. At the same time, visual comparison of the GT and scaled predicted skeletons from a different angle of views are depicted in Table 2.

The exception in the way of evaluation is made only for one method - SMPLify. Because despite the fact that visually the reconstructed poses look precise and its quantitative results are similar to other methods the reconstruction of the pose takes in average 5 to 8 minutes per image which means that it will take over 800 hours which is more than a month to finish the estimation for 10,000 images. Thereby, the validation of the SMPLify method is made over 100 images.

Table 1: Validation results of the tested 3D PE methods

Method	MPJPE (mm)		Scaling factor	Approximate calculation time (hours)
	Error	Scaled error		
LFTD [34]	325,486	296,267	0,588	≈ 20
VNect [16]	84,270	81,270	1,029	1
SMPLify [35]	438,485	306,303	628.315	≈ 800
NBF [32]	424,741	310,723	622,062	9

Considering the purpose of the project to animate the character using the output of 3D PE methods as the mocap data it is important for PE approaches to be fast and accurate. From Table 1 it is seen that the accuracy of tested frameworks is not so good, except for VNect. However, if to take a look at the visual representation of the result given in Table 2 it is clear that mostly the predictions are quite accurate. Such a difference between the qualitative and quantitative results can be due to the fact that during the training of the methods different datasets were used. For example, the above-mentioned MPII [60].

Summarizing all of the above, VNect [16], NBF [32] and LFTD [34] methods are the most suitable for the current project and will be used for the 3D PE and further animation.

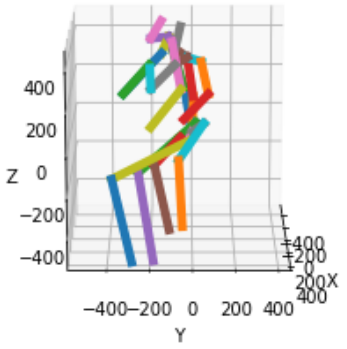
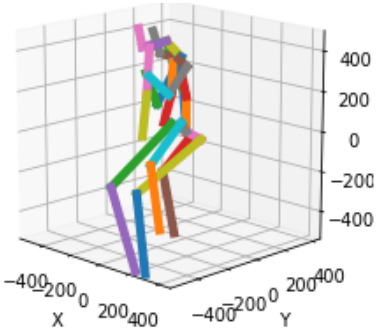
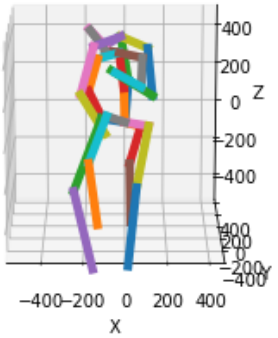
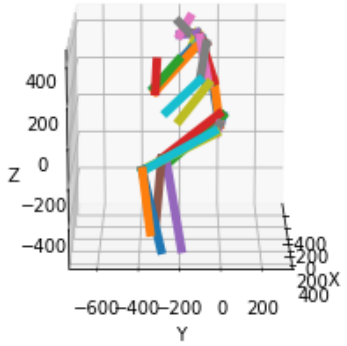
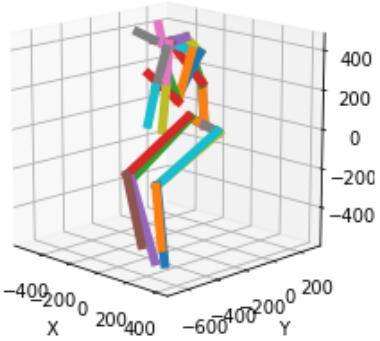
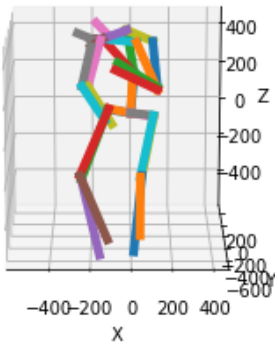
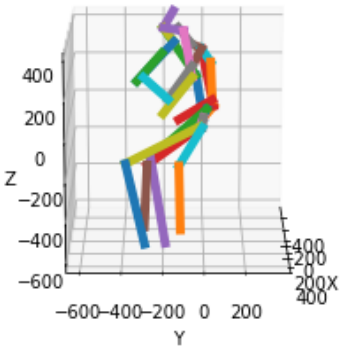
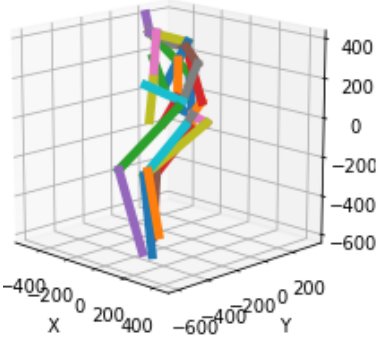
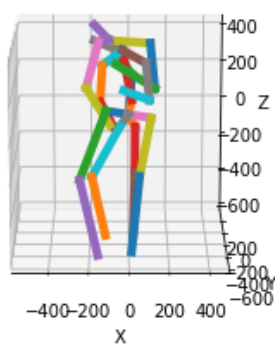
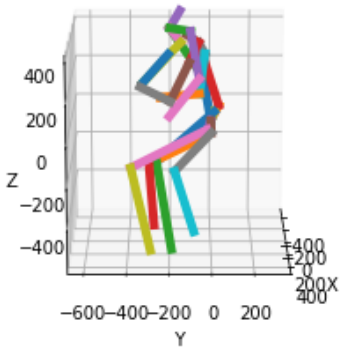
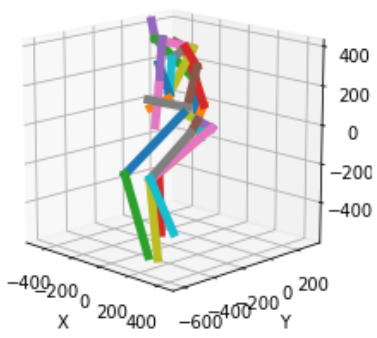
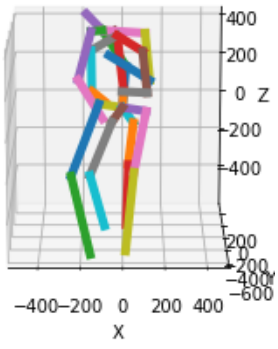
6.2. Animation evaluation

6.2.1. Data collection

To evaluate the results of the implementation steps from Section 5.1, 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: Visualization of the 3D PE methods' predictions and corresponding GT data

Visual results of 3D PE methods		
0° view	45° view	90° view
Lifting from the Deep		
		
VNect		
		
SMPLify		
		
Neural 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.

The questionnaires consisted of eleven blocks. In each out of ten first parts, the participants were asked to watch original video and corresponding animations produced by three methods used and then answer six questions related to the performance of the methods and resulted animations. Two first questions were multiple choice questions and its goal was to define the best performing method for the particular input video. They look similar but there is a difference because, for instance, in the entertainment field things that people like do not necessarily have to be realistic and vice versa.

1. Which animation result looks more realistic?

2. Which animation result do you like more?

The rest of the questions were given as a linear or Likert scale. Here the participants could express how much do they agree with the given statement by choosing a number between "1" and "5", where the answer "1" means that the participant disagrees with the statement and "5" reflect full agreement with the assertion.

3. Animation exactly match the original video.

4. Animation looks good and natural.

5. The quality of the animation is good.

6. Do you like this animation?

In addition, in the last part, there were 3 open questions which were related to general matters regarding the project and one more field if the participants wanted to leave a comment. The difference between the questionnaires was that one of them contained all of the questions except number 10 while the other was missing question number 9. The questions are given below.

7. What did you like about the animations?

8. What do you think can be improved?

9. Do you think this project has potential? Why?

10. Is there a difference with videos from the previous questionnaire? If yes, what is different?

11. Comments

Overall, the evaluation has been performed with the help of Google Forms service. The average time for completing one questionnaire was 10 minutes.

6.2.2. Experimental evaluation results

Eleven participants have watched all of the provided videos and answered the above-mentioned questions. After that the data have been analyzed.

The results of the first two questions were evaluated based on the number of votes they have received in each out of 10 main parts. The method that has got the most of the votes in each case was considered to be the best. The results of the Blender and Unity animations are given in the second column in Tables 3 and 4 correspondingly.

Summing up the answers of Blender performance evaluation, the 3D PE method (Section 3.4) called LFTD was the best in creating a realistic animation in 5 cases out of 10. Meanwhile, VNect produced the best animation 3 times and once its performance got the same amount of votes as NBF that was a leader only in one case.

In the other questionnaire which had to evaluate the animations recorded in Unity, NBF and LFTD performed almost similarly good. The first one produced the best animation in 4 cases and one time shared the votes with VNect. At the same time, LFTD kept its performance and was recognized as the best 4 times. Surprisingly, VNect whose validation results were presented in Section 6.1.2, could not tolerate well the import to Unity and performed well only once.

At the same time, from both questionnaires, it was seen that in 90% of the cases the participants liked the most the same methods that were considered to be more realistic.

In the second block of questions where the participants had to give the answer from 1 to 5 to express the level of their approval of the consent, the result was calculated as an average value. In the case of each video for every question, the answers were summed and divided by the number of participants. The numeric results from both questionnaires are presented in the Tables 3 and 4.

Table 3: Results of the Blender video evaluation

Number	Best method	Matching	Naturality	Quality	Liking
1	VNect	2.9	2.6	3	3.3
2	LFTD	3.5	3	3.2	3.4
3	LFTD	3.3	3.3	3.3	3.6
4	LFTD	3.1	3	3.1	3.4
5	NBF/VNect	3	2.9	3	3.4
6	LFTD	3.6	3.7	3.6	3.9
7	VNect	3.7	3.4	3.7	3.9
8	LFTD	3.1	2.9	3.2	3.2
9	NBF	2.6	2.6	2.8	2.9
10	VNect	3.6	3.1	3.5	3.7
Average values		3.3	3	3.2	3.6

From the Tables 3 and 4, it is clear that the animations from *.blend* files generated by the developed framework were better than the ones made in Unity.

The third block of questions showed that around a half of the participants have found the animations funny and entertaining and the same time the other half pointed that the animations well captured the general motion of the person in the video even though the small motions were often missing. Moreover, all of the participants saw the potential in

Table 4: Results of the Unity video evaluation

Number	Best method	Matching	Naturality	Quality	Liking
1	VNect	2.4	2.5	3	3
2	NBF/VNect	3.2	2.6	3	3.3
3	LFTD	3	3	3.1	3.5
4	NBF	3.5	3.1	3.4	3.5
5	NBF	2.6	2.7	3.1	3.2
6	LFTD	3.3	3.4	3.4	3.8
7	LFTD	2.9	2.8	3.2	3.2
8	LFTD	2.7	2.2	2.5	2.7
9	NBF	2.7	2.2	2.6	3
10	NBF	3.2	3.1	2.8	3.7
Average values		3.0	2.8	3.0	3.3

this project as a good starting point in the animation, and with some improvements as a higher level tool for game development, film making, and other purposes. It was also remarked that even though it might be successfully used for the entertainment industry but is not suitable for scientific use.

However, as for the improvements it was, first of all, recommended to improve the accuracy and stability of the motion, make it more smooth because the limbs of the character in the animation often were shaking and legs were lacking some bending of the knees.

During the comparison of the videos made in different environments (question 10), almost all of the participants agreed that the motion from the original Blender animations looks better. It is more smooth and realistic.

7. DISCUSSION

Since animation based on mocap technology is expensive and not accessible to everyone, the aim of this thesis project was to build a solution for the animation of 3D characters using simple mocap data generated from a regular RGB video.

The developed framework consists of two main parts: 3D PE and character animation. When the user is providing the system with the input video he/she can choose which one of three available PE methods is preferred to be used, otherwise, by default, the VNect will be run because it has the highest execution speed. Finally, the system will output a Blender file which contains the skeletal structure described in Section 5.2.1, simple character mesh and the animation that was created based on the input video. Thus, the system built is simple to use and does not require any technical knowledge about the work of mocap, deep learning techniques, etc. All of its elements are free and open-source software.

Additionally, the produced animation was imported to the Unity game engine to show one of the possible application of the framework output, and at the same time to test how much the produced animations are compatible with other CG software.

For performance evaluation, the animations created from Blender and Unity were quantitatively evaluated using two surveys. The results showed that the animations created by the framework do not always match the original video. This problem appears because of the way how the 3D PE methods work and how accurate they are. For examples, in some tested animations, it can be seen that the methods often get confused when the person is rotating or stands sideways to the camera. If to check the animation frame by frame, it is seen that in neighboring frames the person can be directed in different ways while the silhouette of the pose remains the same. It happens because the 3D PE methods still have problems fighting pose ambiguities. In addition, motion blur which is a common effect on the videos has a negative impact on the accuracy of the methods. Thus, the solution to this problem is to improve the PE methods themselves.

The other remark to the animation is that the motion of the character is not smooth enough. Visually it looks like the character is shaking all of the time. This problem is rooting in PE methods because they see the video as a set of images and even though the neighboring frames do not differ much the predicted poses might be slightly different. But this is already enough, so that the animation would start shaking. It happens because when the data is imported to Blender it creates the keyframes for each frame in the animation and the interpolation, which makes the motion smooth between the frames, is impossible since there is no more empty frames. It can be said that this shaky motion is hardcoded in the animation. The way to smooth it out is to have some empty spaces between the keyframes. Then it will be possible to apply the interpolation for the empty frames. For example, removing every second frame from the animation sequence will already make the difference. But the fact is that such an approach contradicts the concept of mocap, where it is assumed that the more keyframes, the more realistic and smooth the animation will look. Thus, taking into account the specification of the combined techniques and methods it is important to find a compromise that will help to get the best out of both to create smooth and realistic animation.

In addition, it was clear that after importing to Unity a lot of animations became a little bit less stable and its quality dropped. Partially it happened because of the problems described in Section 5.3. However, the solution to this problem was not envisaged

in this thesis project, since Unity was used as an example of the subsequent use of animation. Nevertheless, it can be considered as one more direction to improvement.

Even taking into account described problems, many of the evaluation participants liked the animation and found it entertaining and fun. Moreover, all of them noted that the project has a lot of potentials, especially if the above-mentioned problems will be eliminated. This gives motivation for further development of the framework. As it was said before, one way to improve the animation itself is to experiment with the density of the keyframes and the approach to the selection of the frames that will be ignored. Because this problem might be quite challenging due to the fact that during some simple selection, good poses might be skipped while corrupted ones can get in the animation, and their effect on the motion might be quite significant since the invalid pose will also corrupt the interpolated frames before and after it.

One more way of improvement may consist in solving the problem of shaky data. Improvement of the PE methods is beyond the scope of this project, but within this framework the filters can be applied to smooth the data and remove the outliers. Together with animation improvement offered before, it can be a next step in the project development.

8. CONCLUSION

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.

To experimentally test how the produced animations will behave in a different environment and, at the same time, illustrate how they can be used further, the animation clips were imported to the Unity game engine and used on the native 3D character. Unity animations obtained in comparison to the original motion clips produced by Blender have lost some precision of the character movement.

Qualitative evaluation of Blender and Unity animations showed that many people who have participated in evaluation found the produced 3D character animations interesting and fun, even though the accuracy of the estimated motion was not always high and the movements were a bit shaky. Moreover, survey participants noted that the developed framework has a lot of potential for further development.

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.

9. REFERENCES

[1] Dent S. (2014), What you need to know about 3d motion capture. URL: https://www.engadget.com/2014/07/14/motion-capture-explainer/?guccounter=1&guce_referrer_us=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_cs=YqJyeBwfIGaZrmgCJNEihw, accessed 21 April 2019.

[2] Lonkar S., Sagar lonkar - 3d / vfx / motion capture artist. URL: https://sagarlonkar.wordpress.com/, accessed 22 April 2019.

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

[4] Yamane K. & Hodgins J.K. (2009) Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems , pp. 2510–2517.

[5] Mousavi Hondori H. (2014) A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. Journal of Medical Engineering 2014.

[6] Galantucci L., Lavecchia F. & Percoco G. (2010) A simple photogrammetric system for automatic capture and measurement of facial soft tissues during movement. pp. 151–156.

[7] Ionescu C., Papava D., Olaru V. & Sminchisescu C. (2014) Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence 36, pp. 1325–1339.

[8] Prince S. (2010) Through the looking glass: Philosophical toys and digital visual effects. Projections 4, pp. 19–40. URL: https://www.berghahnjournals.com/view/journals/projections/4/2/proj040203.xml.

[9] Xsens.com | xsens | 3d motion tracking technology. URL: https://www.xsens.com/, accessed 20 April 2019.

[10] Nogueira P. (2012) Motion capture fundamentals a critical and comparative analysis on real-world applications.

[11] Meta motion - motion capture hardware software and services. URL: https://metamotion.com/, accessed 23 April 2019.

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

[13] Nigam V. (2018), Understanding neural networks. from neuron to rnn, cnn, and deep learning. URL: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90, last accessed 16 September 2017.

[14] Bulat A. & Tzimiropoulos G. (2016) Human pose estimation via convolutional part heatmap regression. CoRR abs/1609.01743. URL: http://arxiv.org/abs/1609.01743.

[15] 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.

[16] Mehta D., Sridhar S., Sotnychenko O., Rhodin H., Shafiei M., Seidel H.P., Xu W., Casas D. & Theobalt C. (2017) Vnect: Real-time 3d human pose estimation with a single rgb camera. vol. 36. URL: http://gvv.mpi-inf.mpg.de/projects/VNect/.

[17] Ronneberger O., Fischer P. & Brox T. (2015) U-net: Convolutional networks for biomedical image segmentation. CoRR abs/1505.04597.

[18] Tang W., Yu P. & Wu Y. (2018) Deeply learned compositional models for human pose estimation. In: The European Conference on Computer Vision (ECCV).

[19] Ke L., Chang M.C., Qi H. & Lyu S. (2018) Multi-scale structure-aware network for human pose estimation. In: The European Conference on Computer Vision (ECCV).

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

[21] Belagiannis V. & Zisserman A. (2017) Recurrent human pose estimation. In: International Conference on Automatic Face and Gesture Recognition, IEEE.

[22] Chou C., Chien J. & Chen H. (2017) Self adversarial training for human pose estimation. CoRR abs/1707.02439.

[23] Chen Y., Shen C., Wei X.S., Liu L. & Yang J. (2017) Adversarial posenet: A structure-aware convolutional network for human pose estimation. In: The IEEE International Conference on Computer Vision (ICCV).

[24] Nie X., Feng J., Zuo Y. & Yan S. (2018) Human pose estimation with parsing induced learner. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[25] Fang H.S., Xie S., Tai Y.W. & Lu C. (2017) Rmpe: Regional multi-person pose estimation. In: The IEEE International Conference on Computer Vision (ICCV).

[26] Cao Z., Simon T., Wei S.E. & Sheikh Y. (2017) Realtime multi-person 2d pose estimation using part affinity fields. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[27] Pishchulin L., Insafutdinov E., Tang S., Andres B., Andriluka M., Gehler P.V. & Schiele B. (2016) Deepcut: Joint subset partition and labeling for multi person pose estimation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[28] Insafutdinov E., Pishchulin L., Andres B., Andriluka M. & Schieke B. (2016) Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In: European Conference on Computer Vision (ECCV). URL: http://arxiv.org/abs/1605.03170.

[29] He K., Zhang X., Ren S. & Sun J. (2015) Deep residual learning for image recognition. CoRR abs/1512.03385. URL: http://arxiv.org/abs/1512.03385.

[30] Luo Y., Ren J., Wang Z., Sun W., Pan J., Liu J., Pang J. & Lin L. (2018) Lstm pose machines. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[31] Xiao B., Wu H. & Wei Y. (2018) Simple baselines for human pose estimation and tracking. In: The European Conference on Computer Vision (ECCV).

[32] 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.

[33] Wang M., Chen X., Liu W., Qian C., Lin L. & Ma L. (2018) Drpose3d: Depth ranking in 3d human pose estimation. CoRR abs/1805.08973. URL: http://arxiv.org/abs/1805.08973.

[34] Tome D., Russell C. & Agapito L. (2017) Lifting from the deep: Convolutional 3d pose estimation from a single image. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[35] Bogo F., Kanazawa A., Lassner C., Gehler P., Romero J. & Black M.J. (2016) Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In: Computer Vision – ECCV 2016, Lecture Notes in Computer Science, Springer International Publishing.

[36] Dabral R., Mundhada A., Kusupati U., Afaque S. & Jain A. (2017) Structure-aware and temporally coherent 3d human pose estimation. CoRR abs/1711.09250. URL: http://arxiv.org/abs/1711.09250.

[37] Ramakrishna V., Kanade T. & Sheikh Y. (2012) Reconstructing 3d Human Pose from 2d Image Landmarks. Computer Vision–ECCV 2012 , pp. 573–586.

[38] Akhter I. & Black M.J. (2015) Pose-conditioned joint angle limits for 3D human pose reconstruction. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2015), pp. 1446–1455.

[39] Mehta D., Rhodin H., Casas D., Fua P., Sotnychenko O., Xu W. & Theobalt C. (2017) Monocular 3d human pose estimation in the wild using improved cnn supervision. In: 3D Vision (3DV), 2017 Fifth International Conference on, IEEE. URL: http://gvv.mpi-inf.mpg.de/3dhp_dataset.

[40] Loper M., Mahmood N., Romero J., Pons-Moll G. & Black M.J. (2015) SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) 34, pp. 248:1–248:16.

[41] Kanazawa A., Black M.J., Jacobs D.W. & Malik J. (2018) End-to-end recovery of human shape and pose. In: Computer Vision and Pattern Recognition (CVPR).

[42] Pavlakos G., Zhu L., Zhou X. & Daniilidis K. (2018) Learning to estimate 3d human pose and shape from a single color image. CoRR abs/1805.04092. URL: http://arxiv.org/abs/1805.04092.

[43] Varol G., Ceylan D., Russell B., Yang J., Yumer E., Laptev I. & Schmid C. (2018) BodyNet: Volumetric inference of 3D human body shapes. In: ECCV.

[44] Rhodin H., Salzmann M. & Fua P. (2018) Unsupervised geometry-aware representation learning for 3d human pose estimation. In: ECCV.

[45] Sun X., Li C. & Lin S. (2018) An integral pose regression system for the eccv2018 posetrack challenge. arXiv preprint arXiv:1809.06079 .

[46] R{iza Alp Güler Natalia Neverova I.K. (2018) Densepose: Dense human pose estimation in the wild.

[47] Neverova N., Güler R.A. & Kokkinos I. (2018) Dense pose transfer. CoRR abs/1809.01995. URL: http://arxiv.org/abs/1809.01995.

[48] blender.org - home of the blender project, official website. URL: https://www.blender.org/, accessed 19 April 2019.

[49] Zbrush - the all-in-one-digital sculpting solution, official website. URL: http://pixologic.com/zbrush/, accessed 19 April 2019.

[50] Maya | computer animation modeling software | autodesk, official website. URL: http://autodesk.com/maya, accessed 19 April 2019.

[51] 3ds max | 3d modeling, animation rendering software | autodesk, official website. URL: http://autodesk.com/3dsmax, accessed 19 April 2019.

[52] Unity user manual (2019.1). URL: https://docs.unity3d.com/Manual/index.html, accessed 19 April 2019.

[53] Marc S. (2009), Video game creation design: Skeletal animation. URL: http://alumni.cs.ucr.edu/~sorianom/cs134_09win/lab5.htm, accessed 19 April 2019.

[54] Magnenat-Thalmann N., Laperri‘ere A. & Thalmann D. (1988) Joint-dependent local deformations for hand animation and object grasping. In: Proceedings of Graphics Interface ’88, GI ’88, Canadian Man-Computer Communications Society, Toronto, Ontario, Canada, pp. 26–33. URL: http://graphicsinterface.org/wp-content/uploads/gi1988-4.pdf.

[55] Team B.D., Blender reference manual. URL: https://docs.blender.org/manual/en/latest/index.html, accessed 19 April 2019.

[56] Pluralsight (2015), How to create your first character rig in blender: Part 1 - setting up the armature. URL: https://www.pluralsight.com/blog/tutorials/how-to-create-your-first-character-rig-in-blender-part-1, accessed 19 April 2019.

[57] Thomas F. & Johnston O. (1981) Disney Animation: The Illusion of Life. Abbeville Press.

[58] Mark E. (2007), University of virginia archive, 3d animation: Keyframing. URL: http://web.arch.virginia.edu/arch545/handouts/keyframing.html, accessed 19 April 2019.

[59] Redmon J., Divvala S., Girshick R. & Farhadi A. (2016) You only look once: Unified, real-time object detection. pp. 779–788.

[60] Andriluka M., Pishchulin L., Gehler P. & Schiele B. (2014) 2d human pose estimation: New benchmark and state of the art analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).