Motion Capture Generation From Videos Through Neural Networks

Technical University of Crete

September 14, 2022

Overview

Introduction

Thesis Diagram

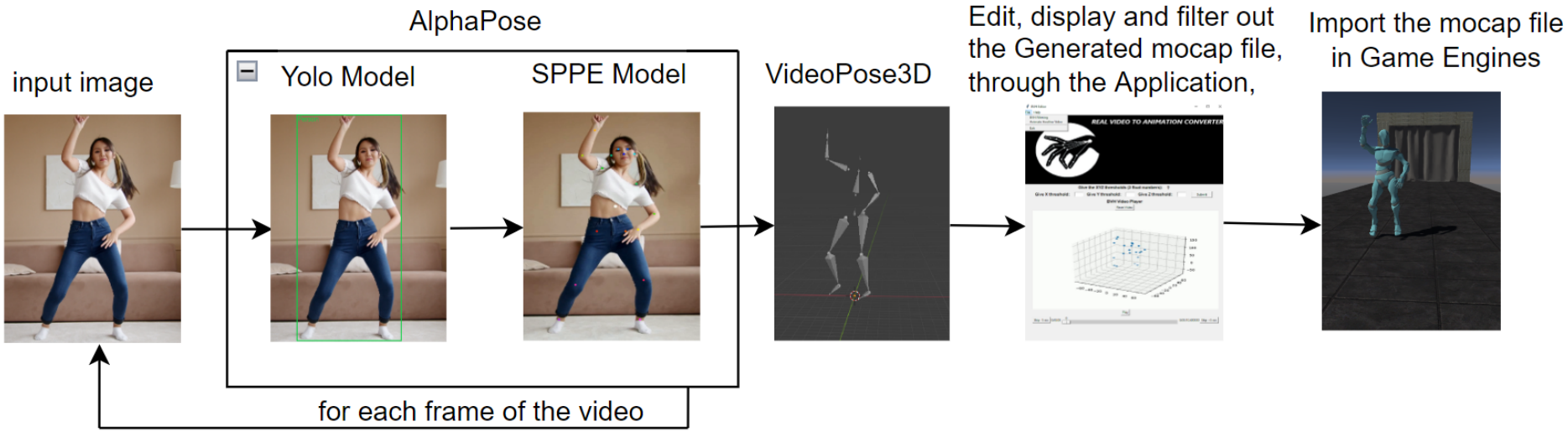


Figure: Thesis Diagram

What is Motion Capture

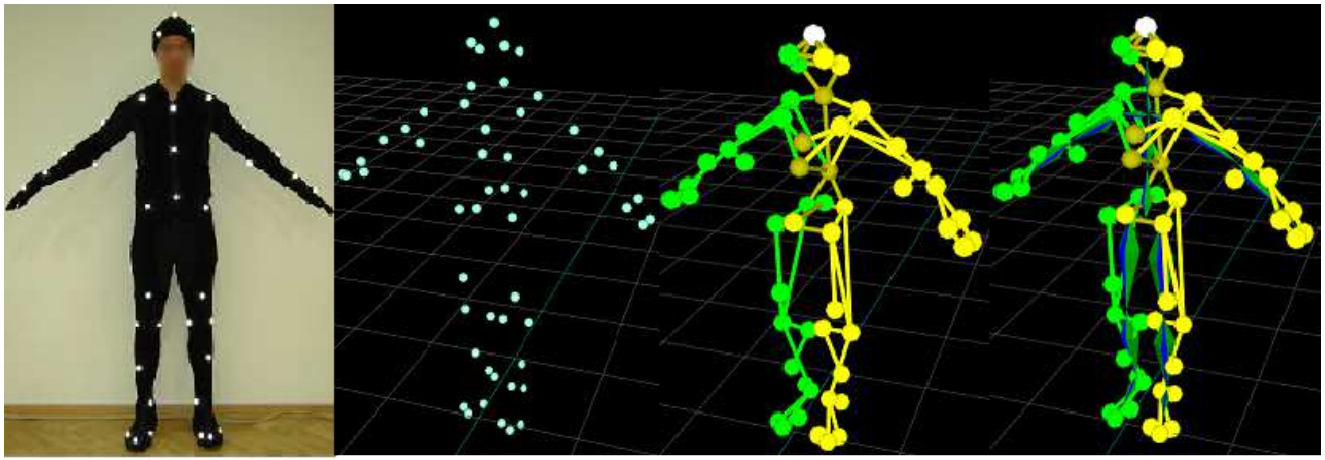


Figure: Motion Capture Pipeline

Motion Capture

Motion capture (abbreviated as mo-cap or mocap) is the process of recording the movement of people. It is used in military, entertainment, sports, and medical applications, as well as for computer vision and robot validation.

Problem Statement

- **Motion Capture Techniques:**
  At the moment the most popular and accurate method to do such a thing is to use a motion-capture suit. Subsequently, the results are to be checked by animators for possible corrections.
- **Problematic Situation:**
  Unfortunately, these suits cost over 3000 dollars, thus many studios and students that want to use motion capture, have limited options. The free motion capture that someone can find has either poor quality or small quantity.

Our approach

- **Solve this problem using Neural Networks**
  Nowadays, many researchers are trying to train Neural Networks in different innovative ways, to estimate human motion from a single video.
- **Adding Filters to the output of the Neural Networks**
  When someone uses a Neural Network to generate an output, it will also produce a type of noise in this generated data. We propose to use some filters that will reduce this noise while maintaining the info of these data.
- **Windows Application**
  This Application will be a friendly user environment for the animators that want to use the proposed algorithm.

Theory

Biovision Hierarchical (BVH)

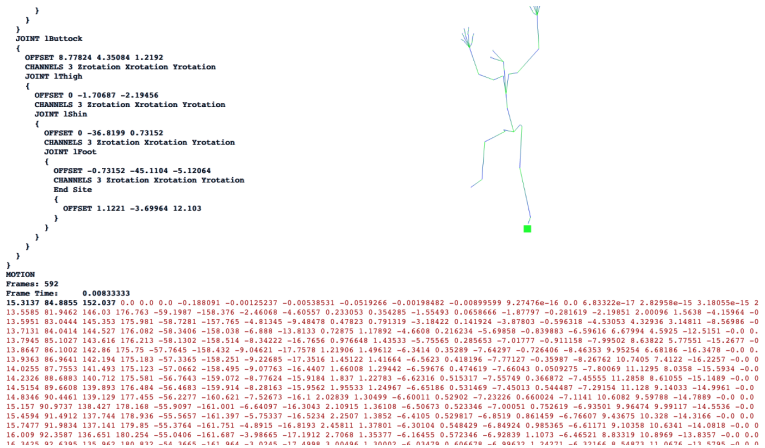


Figure: BVH files are the most popular way to save motion data in digital form.
They have 2 major components:

- **Skeleton data**
  They contain information about the skeleton form as well as the relation between each bone.
- **Motion data**
  A list of floating point values that contain the orientation and location of each bone for every frame of the motion.

Neural Networks I

- **Deep Neural Network**
  Each neuron in the fully connected layer transforms its input (x) using the weighted vector W and the bias b to compute the output:

$$Z = Wx + b$$

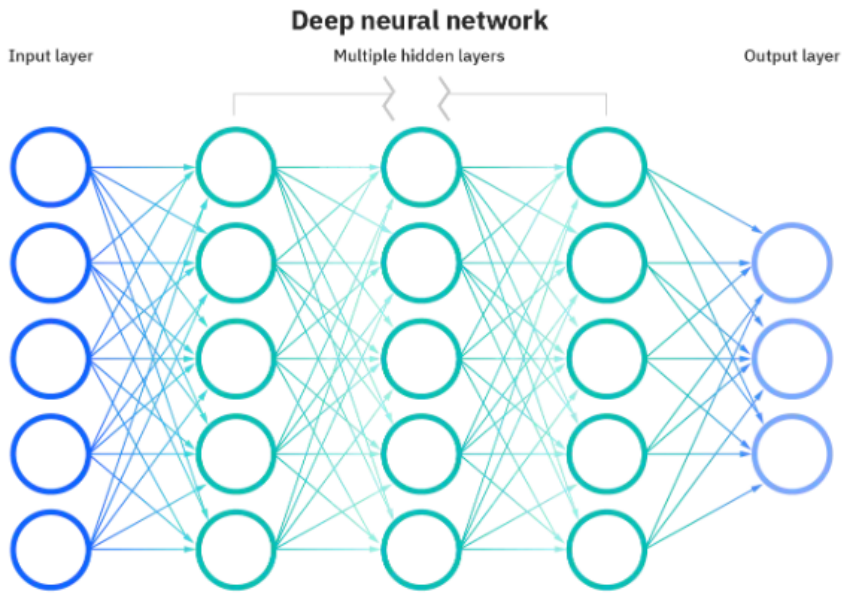


Figure: Deep Neural Network Visualization

Neural Networks II

- **Pretrained Deep Neural Networks**
  A pre-trained model is a saved network that was previously trained on a large dataset.

Our pretrained Models

We will use three pre-trained models. These Pre-trained models are the state-of-the-art methods for mo-cap estimation, thus we decided to employ them.

Neural Networks III

- **Classification**
  Classification is a predictive model that approximates a mapping function from input variables to identify discrete output variables, which can be labels or categories.
- **Regression**
  Regression models predict a continuous value based on the input variables. The main goal of regression problems is to estimate a mapping function based on the input and output variables.

Our Models

The first model labels the persons in the videos and the other two models estimate the floating-point values of human motion. Thus, we are using one Classification model and two regressions

Approach

Why choosing AlphaPose for 2D pose estimation

AlphaPose pre-trained Models

AlphaPose is a pose estimation algorithm, which chooses Yolo as a human detector and a single-person pose estimator (SPPE) that detects the human key points.

- **GPU Load Available**
  Both of the models run simultaneously, so we should be able to load both of them in GPU at the same time. AlphaPose models for their version, consume 1.6 GB, while our VRAM is 2 GB.
- **Sufficient results**
  We will talk more about the results in the evaluation section, but the AlphaPose can achieve a very good 2D single-person pose estimation.

YoloV3 model

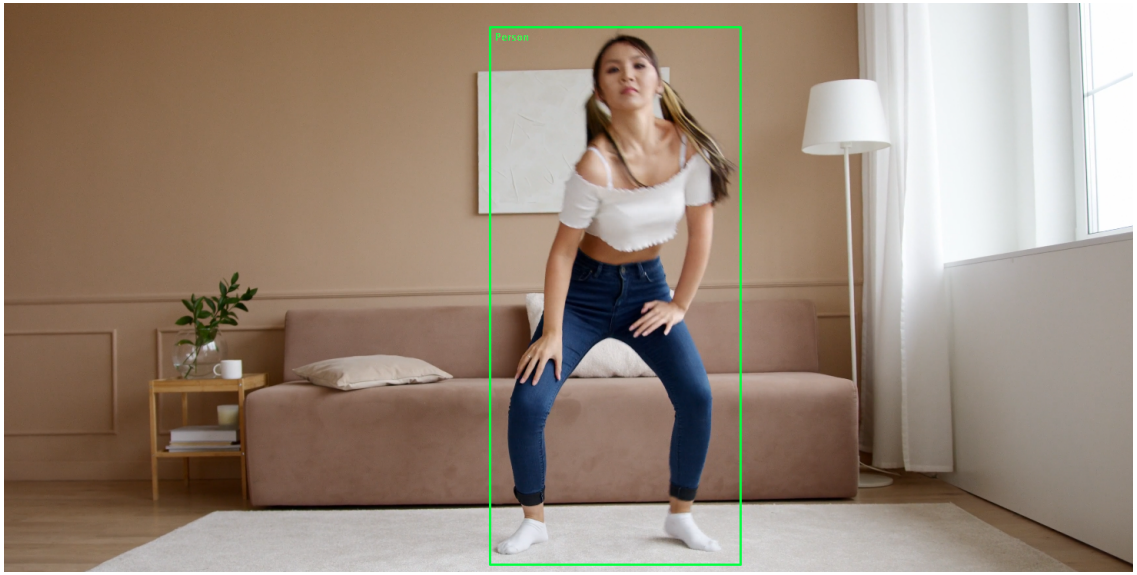


Figure: Yolo Model human localization

- This version of the Yolo model, can classify 80 different objects and segment them inside a box.
- The only classification that we need is the human classification, so we can simplify the model in order to get the results faster.
- The localization of the human in the image helps the SPPE model to work better and faster.

SPPE model

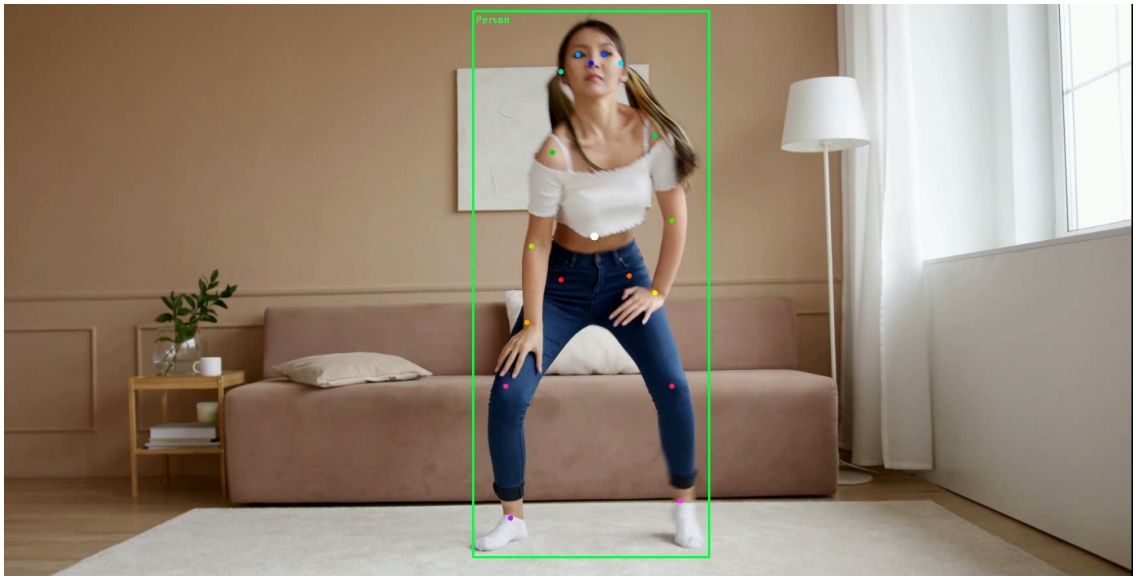


Figure: SPPE Model pose estimation

- The main purpose of the Yolo model's segmentation in the above image is to have the person dominate in that image's content.
- The person has 17 associated key points, and each keypoint is again identified with a location and confidence.

AlphaPose model

- The AlphaPose model that we discussed gives us an array (N,17,3) with N the number of the frames, 17 the estimated key points per frame, and three variables (x,y,z).
- The x and y are the estimation data and the z is the model confidence for this estimation.
- If the confidence for a frame is below a threshold we remove this frame.
- The estimation data contains a small estimation error in each frame. This estimation error between each frame produces a statistical noise that affects the quality of the motion.

3D pose estimation I

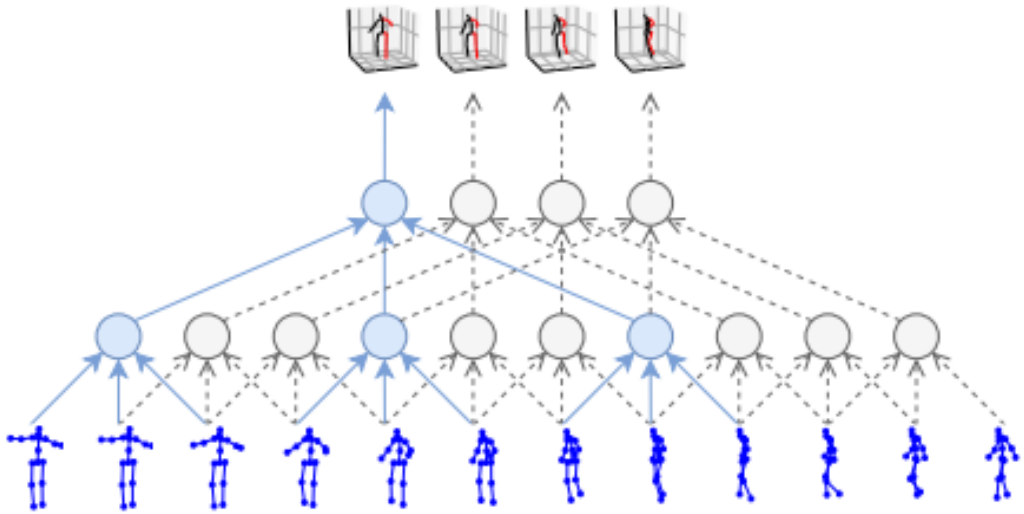


Figure: Model visualization

- The VideoPose3D model takes a sequence of n frames with 2D joint positions as the input and outputs the estimated 3D pose for each frame.
- This model was trained with an enormous Dataset, which contains 3.6 million video frames for 11 subjects, of which seven are annotated with 3D poses.

Position Estimation

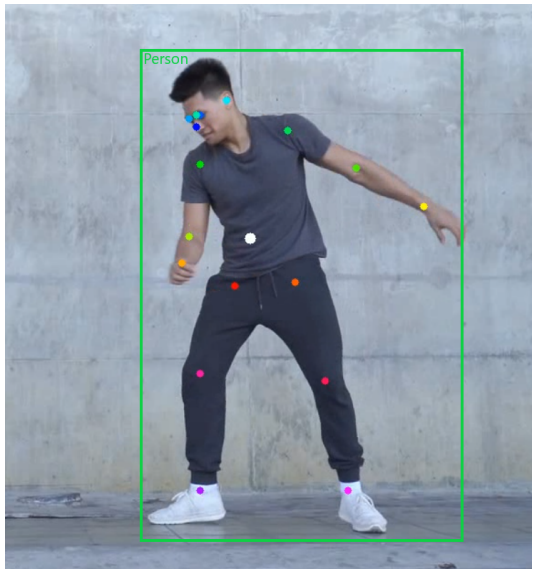


Figure: Using the 17 key points that the SPPE model estimated, we compute the average coordinate of these key points, which should be close to the location of the human hip for each frame, the white dot.

- Regarding the z dimension, we will use the Yolo model bounding box. We will find the area of that box, and we will normalize this number. The bigger the box means that the person is closer to the camera, and the smaller means that the person is moving away from it.

Estimation data Visualization

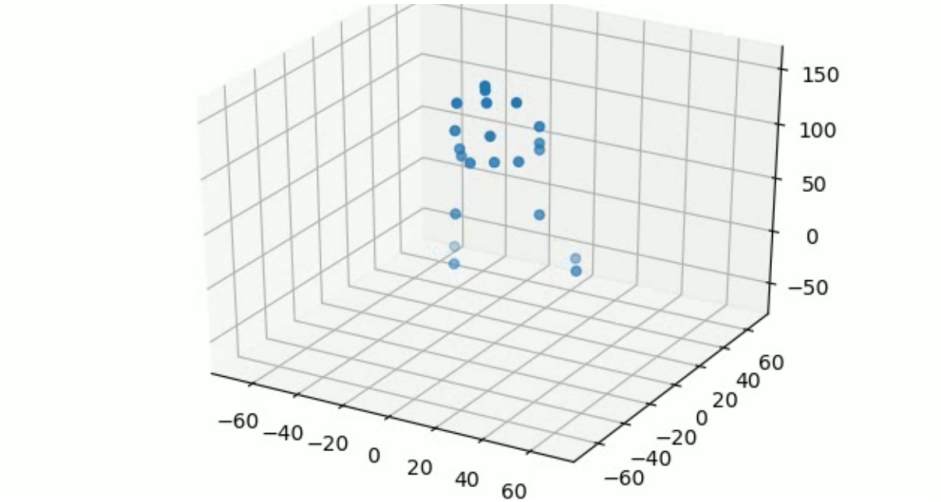


Figure: 3D human pose visualization in our Application

- In the figure above we visualize the 3D human pose from the VideoPose3D model.
- The estimation of the model contain only raw data.
- In order to import it to studios such as Blender, we have to create a BVH file.

BVH Construction

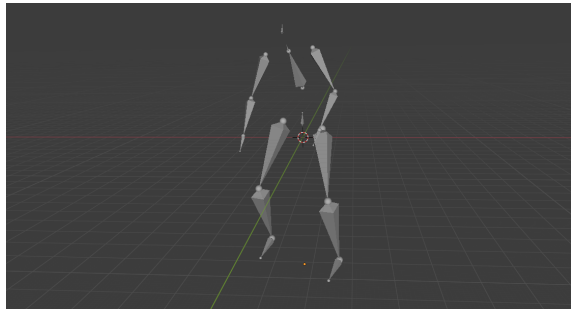


Figure: 3D human pose visualization in Blender

What we extracted from the video

In the previous steps, we found the 3D human pose estimation. However, in order to import these data into studios such as blender, we have to create a BVH file.

- **Skeleton:** We have a Skeleton template that we should create, the AlphaPose Skeleton.
- **Motion Data:** We already estimated the motion data, so after some normalization of the data, we are ready to create the BVH file.

Noise in the models estimation

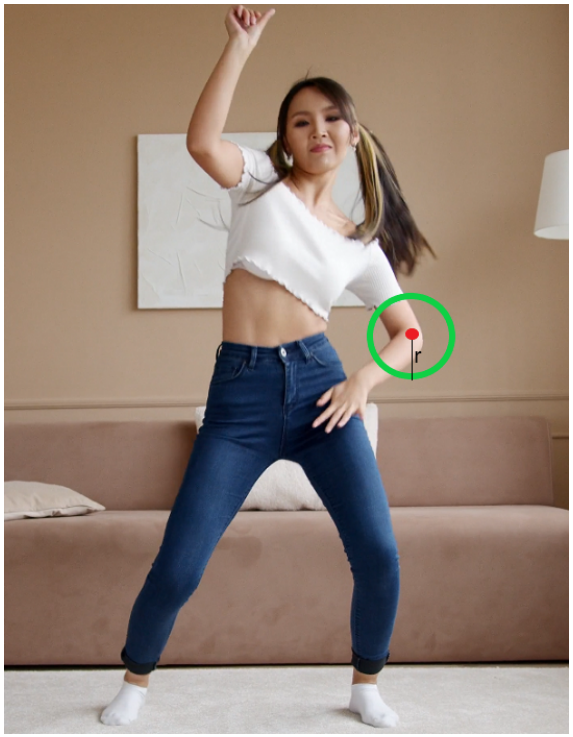


Figure: Noise in the estimation example

- **Green Circle:** The maximum error that the SPPE model can make is inside that circle.
- **Red Dot:** The true position of the human joint.

Filters I

- We decided to implement three different filters that will remove the statistical noise. The Gaussian, the Median, and the Butterworth filter.
- Each filter needs some parameters as an input, and by calibrating those, we can significantly reduce the noise.
- The filter does not affect the motion information when we find the ideal parameters.

Motion Data Shape

The motion data are a 2D array, so it is like we have an image with some noise. Therefore the shape of our data helps us to use some well-known denoising filters.

Filters II

- Each row of the motion data contains information about the orientation and location of each bone for that frame.
- Each column of the motion data contains information on a single dimension for a specific bone for all the video frames.
- The filter kernel size in median and gaussian is 1xN where N is a parameter chosen by the user.
- We apply the filter in every value of the array.
- Regarding Butterworth filter, we ask for the order, lower and higher cut off frequency and we compute the transfer function. We chose this filter since it removes noise from data without affecting the curve's minimum or maximum values.

Windows Application I

About Windows Application

We decided to convert the raw python code into a Windows Application that anyone can easily use.

- We used the python Library Tkinter to create the Application interface.
- There is another python Library, pyinstaller which converts the python code into a .exe file.

Pyinstaller limitations

Unfortunately, pyinstaller can not include all the necessary python libraries that we need. Therefore, we run the python code from a bat file, and with the AdvancedBatToExeConverter we created our .exe file which wanted.

Windows Application II

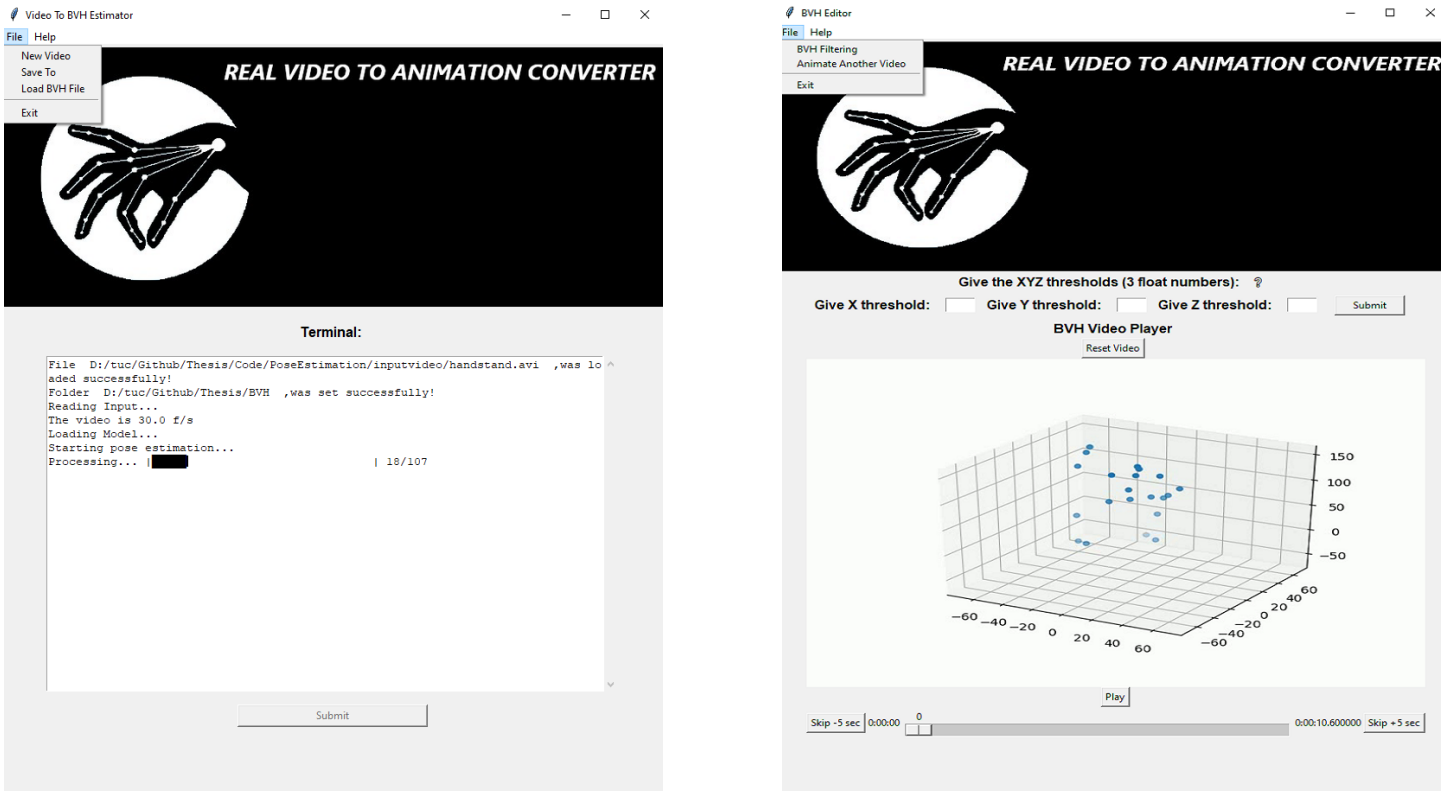


Figure: Application Main Panels

- We created a friendly user environment
- The user does not need to have python or Machine learning knowledge
- The user only has to install the Application.

Windows Application III

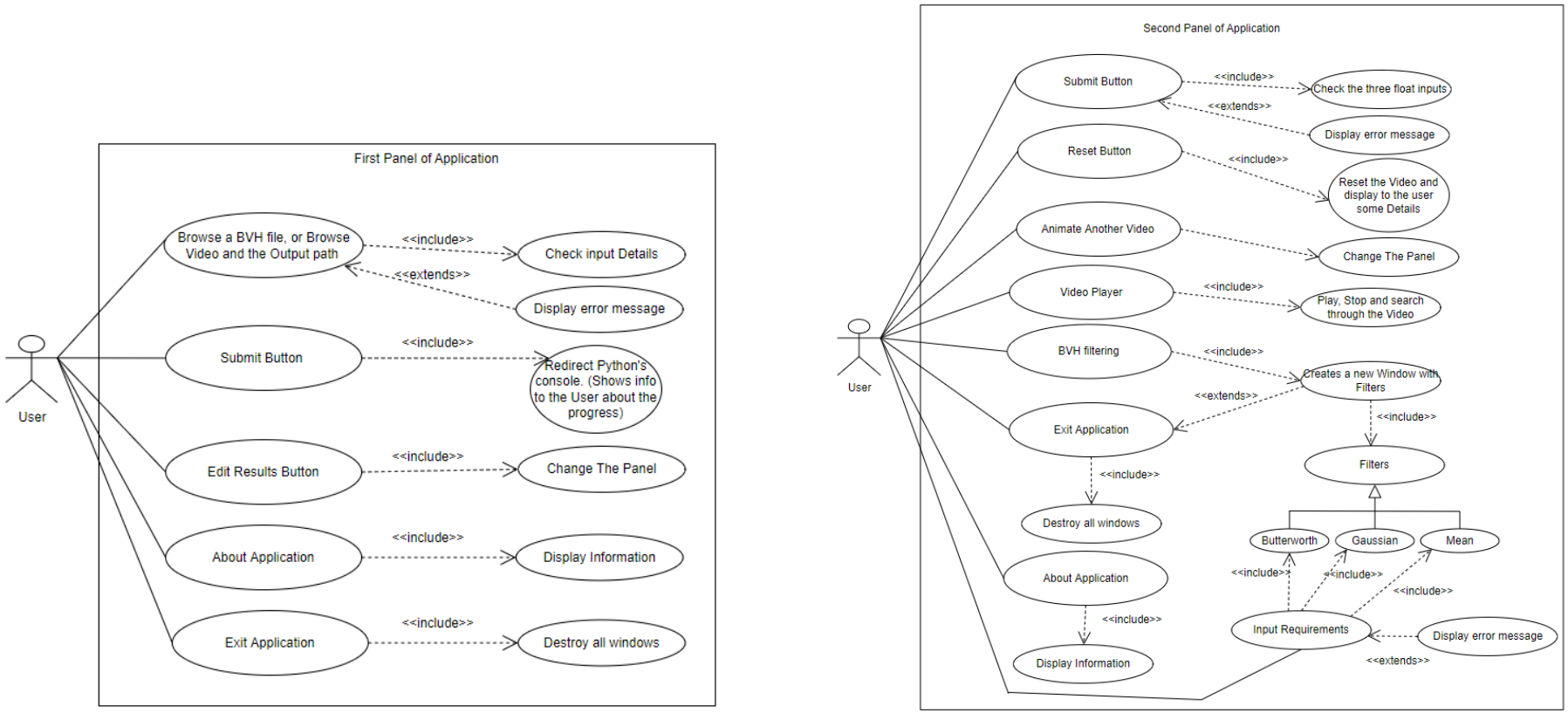


Figure: Application Main panel use case diagrams

Requirements

Requirements

- **Video Resolution** We suggest that the resolution of the video is at least 720p, in order that every human joint can be easily recognizable by the model.
- **Application Audience:** The results (depending on the movement complexity) may not be clear enough to be imported directly into a game engine studio so this application is addressed to animators.
- **Slow-Motion Video** Importing a slow-motion video, helps to eliminate all the sharp moves, however, it needs more processing time.
- **Camera Settings:** Firstly, the camera should be motionless, because it affects the position estimation. Secondly, the person's joints should be visible, otherwise, the model may struggle to specify left from right joints, (legs and arms). It is important that the body of the person should be visible for every frame of the video, and the distance between him and the camera should be greater than two meters and less than six meters.

Evaluation

Evaluation

- Evaluation of the three models that we use for the 3D human pose estimation. This evaluation had been done by the model creator with some well-known metric systems.
- Evaluation of the filters that we propose. We use our own metric system since there was any known convenient method to evaluate them.

Yolo Model Evaluation I

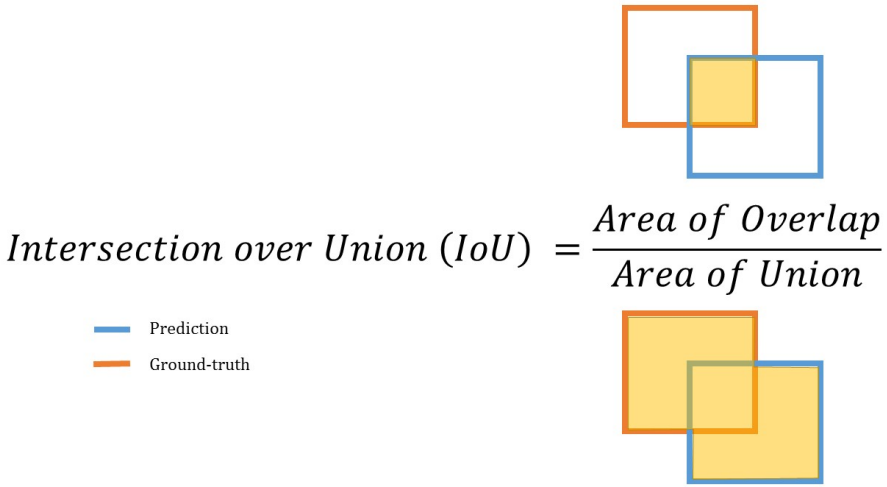
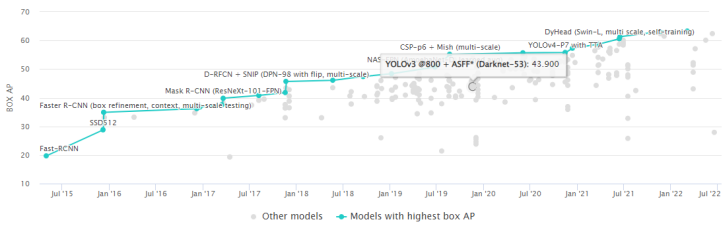


Figure: Intersection over Union

- In object detection, we want to classify an object and we also need to find the bounding box.
- To evaluate such a model, we compare the label that we classified, and about the box, we check the Intersection over the union (IoU) of the true box location and the estimated box location.

Yolo Model Evaluation II



Task	Dataset	Model	Metric Name	Metric Value	Global Rank
Object Detection	COCO test-dev	YOLOv3 @800 + ASFF* (Darknet-53)	box AP	43.9	# 119
			AP50	64.1	# 89
			AP75	49.2	# 78
			APS	27.0	# 79
			APM	46.6	# 96
			APL	53.4	# 116

Figure: Yolov3 Object Detection results on the MS COCO test-dev dataset of some typical baselines. AP, AP50 , AP75 scores (%). APS:AP of small objects, APM:AP of medium objects, APL:AP of large objects.

SPPE Model Evaluation I

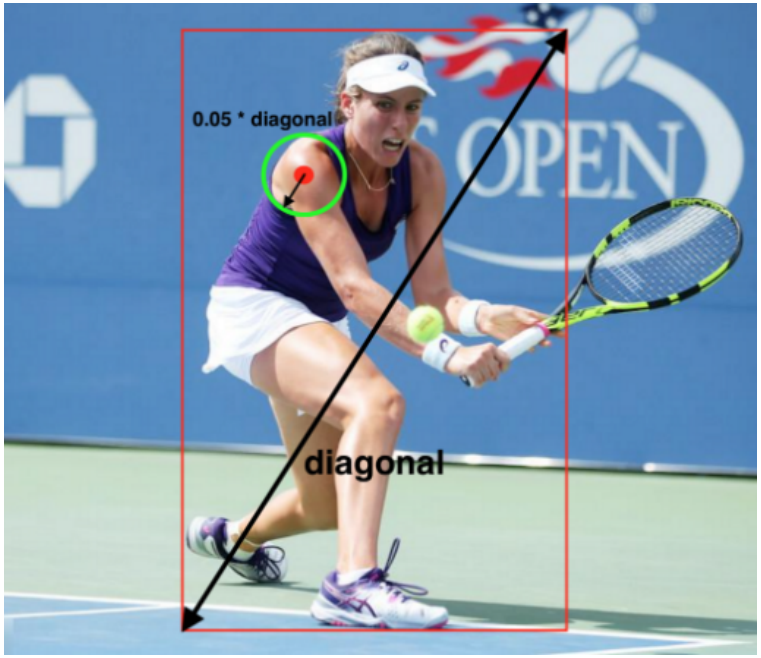


Figure: SPPE evaluation Example

In SPPE the detected joint is considered correct if the distance between the predicted and the true joint is within a certain r.

SPPE Model Evaluation II

Multi-Person Pose Estimation	MPII Multi-Person	Associative Embedding	AP	77.5%	# 5
2D Human Pose Estimation	OCHuman	Associative Embedding+	Test AP	32.8	# 5
			Validation AP	40.0	# 4
Pose Estimation	OCHuman	Associative Embedding	Test AP	29.5	# 8
			Validation AP	32.1	# 8
Pose Estimation	OCHuman	Associative Embedding+	Test AP	32.8	# 6
			Validation AP	40.0	# 5
Keypoint Detection	OCHuman	Associative Embedding	Test AP	29.5	# 7
			Validation AP	32.1	# 7
2D Human Pose Estimation	OCHuman	Associative Embedding	Test AP	29.5	# 8
			Validation AP	32.1	# 8
Keypoint Detection	OCHuman	Associative Embedding+	Test AP	32.8	# 5
			Validation AP	40.0	# 4

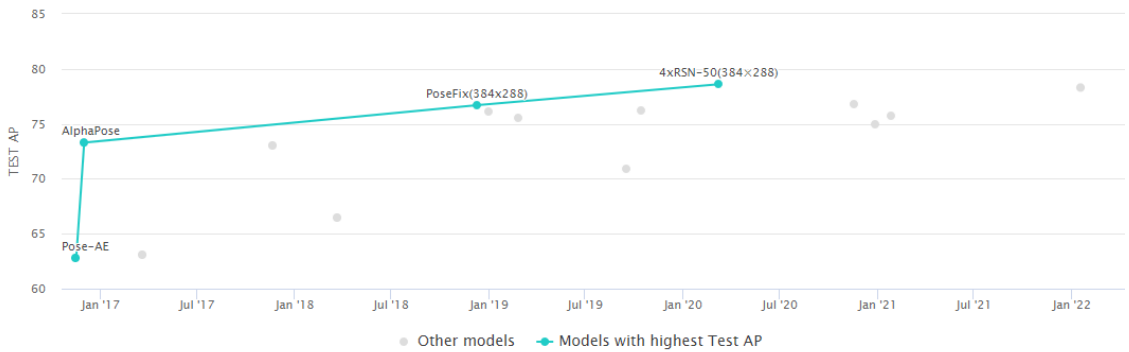


Figure: SPPE results on the MS COCO test-dev dataset of some typical baselines.

VideoPose3D I

To evaluate the VideoPose3D we will use the MPJPE (mean per joint position error) metric system. More specifically, this metric system computes the average Euclidean distance between the ground truth and predicted joints in millimeters. The formula of the MPJPE for a frame f and a Skeleton S is

$$E_{MPJPE}(f, S) = \frac{1}{N_S} \sum_{i=1}^{N_S} \| m^f_{f,S}(i) - m^f_{gt,S}(i) \|_2$$

where NS is the number of the joints in skeleton S. For a set of frames the error is the average over the MPJPEs of the frames.

VideoPose3D II

Task	Dataset	Model	Metric Name	Metric Value	Global Rank
Monocular 3D Human Pose Estimation	Human3.6M	VideoPose3D (T=243)	Average MPJPE (mm)	46.8	# 6
			Use Video Sequence	Yes	# 1
			Frames Needed	243	# 27
			Need Ground Truth 2D Pose	No	# 1
3D Human Pose Estimation	Human3.6M	VideoPose3D (T=243 GTt)	Average MPJPE (mm)	37.8	# 48
3D Human Pose Estimation	Human3.6M	VideoPose3D (T=27 GTt)	Average MPJPE (mm)	40.6	# 55
3D Human Pose Estimation	Human3.6M	VideoPose3D (T=243)	Average MPJPE (mm)	46.8	# 95
			Using 2D ground-truth joints	No	# 1
			Multi-View or Monocular	Monocular	# 1

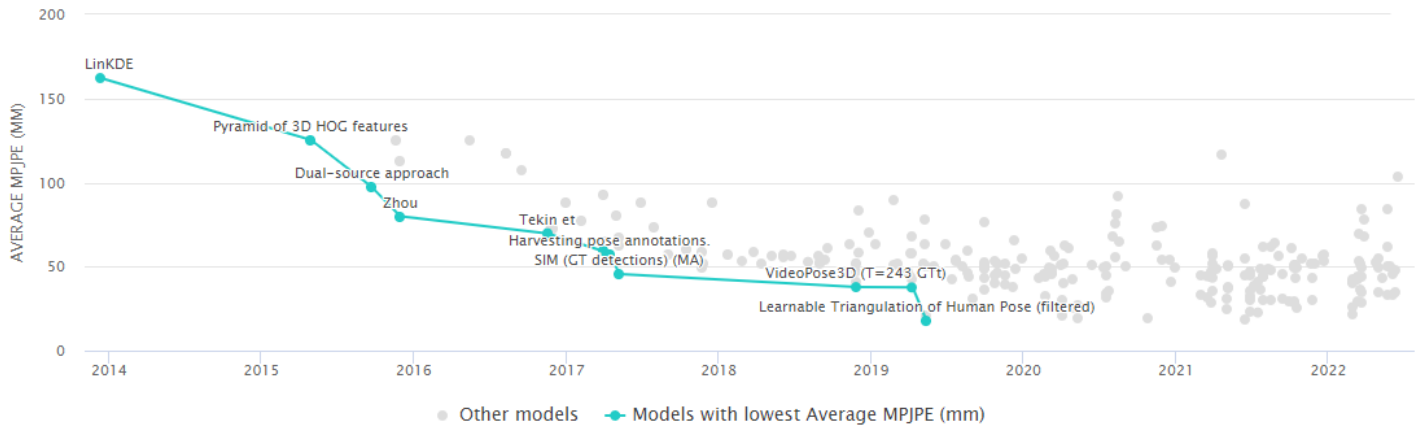


Figure: VideoPose3D results on Human3.6M dataset with the MPJPE metric system

Filters evaluation I

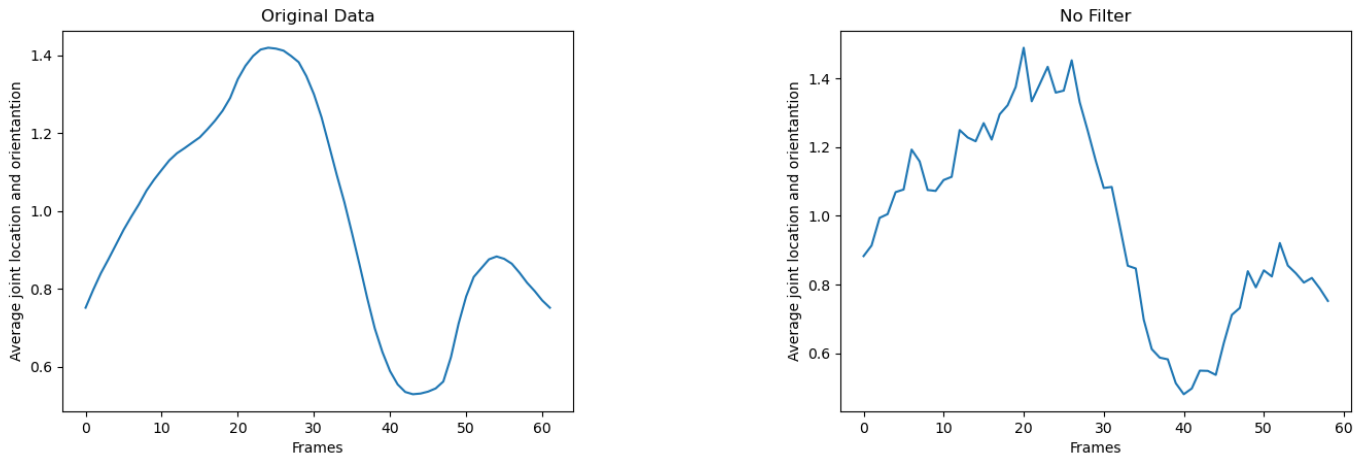


Figure: Professional Mixamo BVH file compared with this file when we add noise with our metric system, CRPF

Metric System

To evaluate the filters, we created a metric system. This metric system will find the average motion data value for each frame, and we will compute the change rate per frame (CRPF).

Filters evaluation II

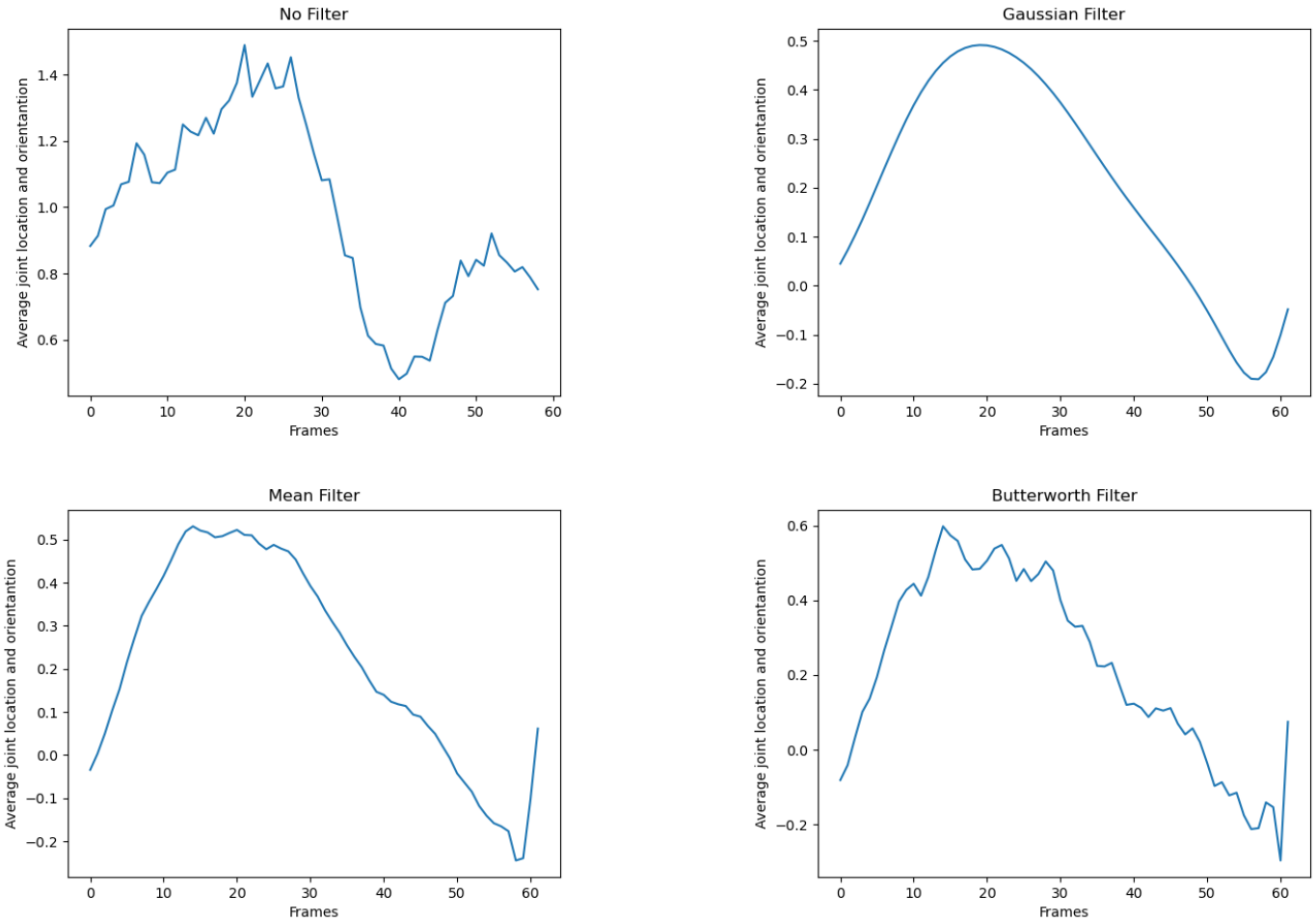


Figure: Filtering comparison based on the mixamo BVH file with the added noise with our metric system, CRPF

Filters evaluation III

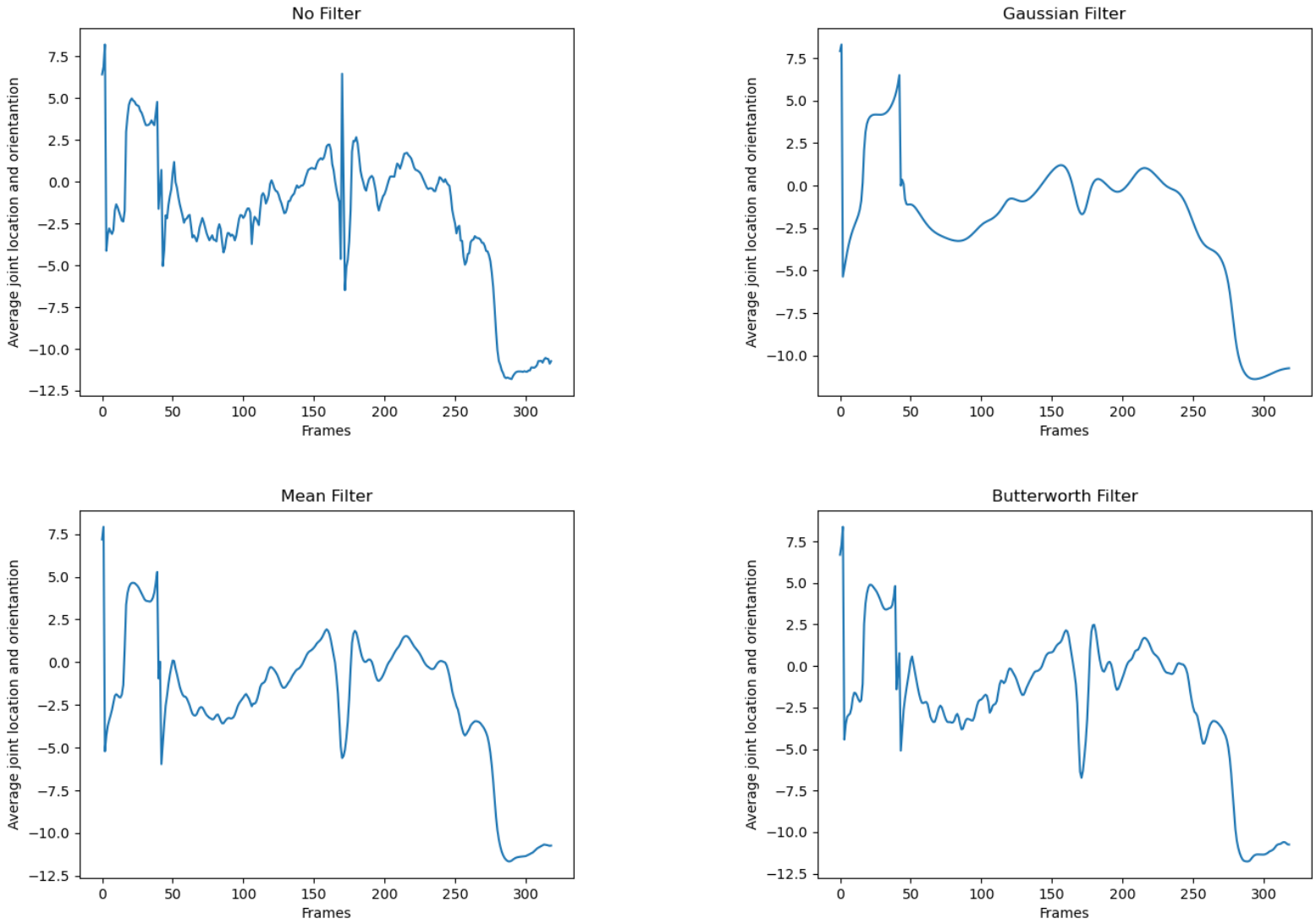


Figure: Filtering comparison on a BVH file generated by Neural networks based

Conclusion

Contribution of our proposal

- A Windows Application for animators that can edit and display a mocap file.
- A method that only needs as input a video and produces a mocap file from scratch.
- Filters that can remove the noise from mocap files.

Limitation of our proposal

- First of all, there are some harsh requirements in the input video that we are asking from the user, that reduces the number of motions that someone can produce.
- The Skeleton that we produce has only 17 bones, containing only the most essential human bones while the mocap-suit method contains many bones.
- The researchers that created these models will continue to improve them, however, in our program, the models would stay stable. More specifically, we will have to keep updating our models with the newest versions of them in order to have a state-of-the-art method.

Future Work

- **Import Facial and hand models:** As we mentioned before, our model contains only 17 bones, if someone could create a model that can estimate facial and hand points with great accuracy, it would be a massive enhancement for our Application.
- **Import motion clips into Robots:** The algorithm runs in our system runs at 4-5 frames per second in an above-average system. If we use a better system, it could run in Real-Time. This means that someone could use a Robot, that contains a camera, that would find a human and would try to copy his motion in Real-Time.
- **Improve the current Neural Network models:** In our approach, we used pre-trained models in order to estimate the 3D human-Pose from a video. These models can be improved with further training or with an enhancement to their architecture.

Acknowledgements

I want to express my gratitude to:

- Professor Mania Aikaterini (Supervisor)
- Professor Chalkiadakis Georgios
- Professor Michail G. Lagoudakis
- The Surreal team of our University