Utilizing Neural Networks for the Correction of Motion Capture Data

Liam Lynch, Rochester Institute of Technology

***Abstract*—Single-camera motion capture devices offer a cheap method of obtaining mocap data at the cost of frequent motion artifacts in the rendered product. To accurately represent natural human motion in mocap data, these artifacts must be removed through the process of hand-cleaning. This process can be extremely tedious, requiring mocap artists to fix noisy data frame by frame. The goal of this project is to offer an alternative to hand-cleaning motion capture data: utilizing neural networks for the removal of common motion artifacts in joint rotational data. With two networks for gap filling and noise removal, noisy rotational data can be transformed into smooth, clean motion curves that match the intensity and structure of natural motion.**

I. Introduction

MOTION capture data consists of rotational information for a skeleton of designated joints on the human body. For every frame of captured motion, there are XYZ rotation values collected for each of these designated joints. Mocap data relies on the use of a base position from which these rotational values are established, such as a T-pose. As joint rotation deviates from this base pose, these XYZ values change in correlation with the motion of the subject. For this project, we will be focusing strictly on the correction of elbow joint rotation. Elbow joint rotational data is formatted as follows: for each frame of captured motion, there is a single joint angle value representing the angle formed between the elbow's current position, and its original T-pose position (Fig. 1A).

Motion capture data is contained within text-based AMC and ASF Acclaim files. The ASF file contains information about the skeleton, such as base pose, offset information, and degrees of freedom (DOF) for each joint. With the DOF values, the ASF asserts the planes within which the joint is allowed to rotate. The AMC file contains the rotational data for each joint across every frame of motion.

Single camera motion capture systems are used to capture movement from a single visual perspective. Therefore, errors in this data often result from a specific joint being blocked or occluded from the view of the camera. Occlusion can cause gaps within the motion capture data – series of frames in which rotational data is not accurately captured by the device and software. Joint position and rotational data are approximated from the RGB or depth-based images captured by the single camera. Subsequently, the skeletal rig is fitted to the motion of the subject present within this sequence of images. Skeleton fitting attempts to determine joint positions from areas of the depth image that correspond to features on the human body. Though the assignment of the skeleton is often successful, slight fluctuations in the approximated joint rotations can lead to shaking and instability in the rendered motion. These fluctuations are referred to as jitter.

Typically, corrupt mocap data is hand cleaned. This process requires the mocap artist to examine the rotation curves created in each individual XYZ plane over all frames of motion. As the artist recognizes errors in the motion capture data, they can restructure the rotational curves through the elimination of keyframes, as well as the application of smoothing functions. The accuracy of motion relies on the artist's interpretation of the data. Smoothing functions are applied to smooth the rotation in areas where key frames are no longer present. Using neural networks, we intend to eliminate the errors associated with jitter and occlusion in order to limit the effort required to clean mocap data. Ideally, our networks will transform corrupt rotational curves into clean data indicative of the intensity and smoothness of natural motion. Examples of clean and corrupt joint angle data can be witnessed in Fig. 1B.

II. Implementation

A. Neural Network Fundamentals

The intention of a neural network is to take a set of input data and approximate an output. Networks consist of layers. Each layer consists of nodes. Within the input layer each node will correlate to a specific input value that is supplied to the network. Similarly, the output layer nodes correspond to each approximated output value. Between the input and output layers are the hidden layers. These middle layers are responsible for the rigorous calculations that transform an input into an output. Each hidden layer node is associated with a weight and bias. Weights are values that are multiplied to a node input, while biases are added to the node input – similar to a typical linear equation, in which the weight is the slope and bias is the y-intercept. The full hidden layer node output is comprised of the summation of (weight*input+bias) values for all inputs connected to that node, with a transfer function applied to this summation. The transfer function is used to process an input and supply an adjusted output. One example that is frequently used within our networks is the tan-sigmoid transfer function, which transforms an input into an output value ranging from -1 to 1.

When a network is trained, we supply it with both a known input and known output. Networks are trained in cycles, or epochs. With each epoch, the known inputs are passed through the network until an approximated output is achieved. This approximated output is then compared to the known output data. If there is error between the approximated and known outputs, the weights and biases of the network are adjusted

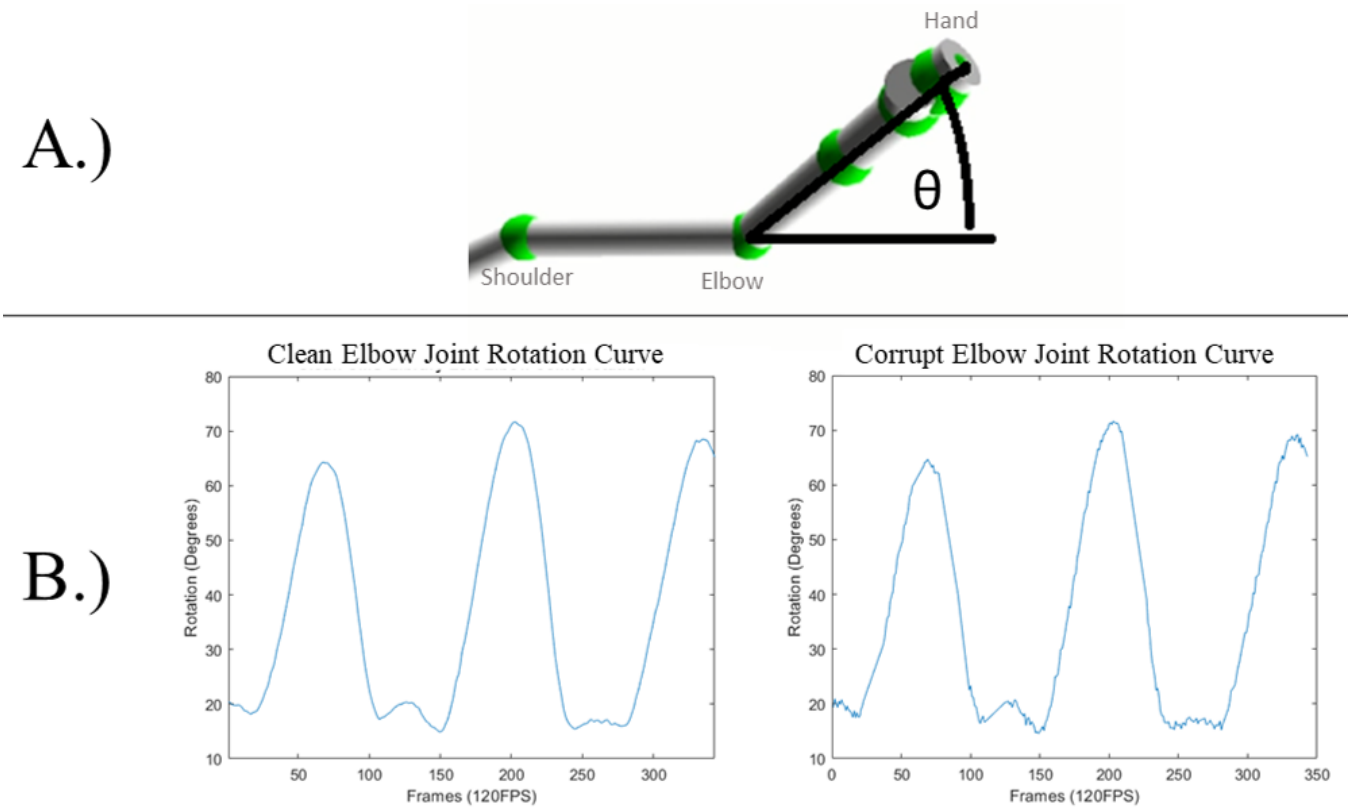

Fig. 1. The above figure demonstrates typical elbow joint movement. Figure 1A demonstrates the joint angle formed between the joint in motion and the base pose, while 1B depicts what clean and corrupt joint angle curves look like for a typical walk cycle.

to allow for a more accurate estimation of the output. This process is repeated for each epoch.

B. Creation of Simulated Data

For our case, neural networks offer an effective method for approximating joint data. We can supply sets of joint rotational values as the input for our network. In training, corrupt input joint angles can be compared against a clean ideal, with which it can approximate the weights and biases necessary to obtain clean joint angle values. Ideally, our training input data should consist of corrupt elbow joint data captured with the Kinect and passed through iPi Soft, while our training output should be hand-corrected joint data provided by a mocap artist. However, a similar result can be accomplished through the use of simulated data. Simulated gaps and jitter are applied to pre-cleaned rotational curves to mimic the structure and errors of typical markerless data. Our clean rotation data originates from AMC files taken from the CMU Motion Capture Library containing data captured using Vicon mocap gear, sampled from over 20 walk cycles at 120fps. These AMC files contain data accurate to human motion and will represent clean and ideal rotation values. When we pass altered or corrupt rotation values to the network series, we should expect the final output to reflect the structural attributes of this data: smooth elbow rotation accurate to lifelike walking motion. From this ideal, we work backwards in applying the various motion artifacts. For the simulation of jitter, noise values are randomly sampled from a Gaussian distribution generated using the average standard deviation associated with our corrupt Kinect data. Noise values are added or subtracted at each frame of rotation to create the jagged curve structure typically associated with jitter. Subsequently, gap lengths are sampled from an exponential distribution with a specified maximum. In this distribution, there is a higher likelihood that shorter gap lengths will occur over higher gap lengths. Furthermore, each frame of rotation has a 10-percent chance of starting a gap. Next, all gaps undergo linear interpolation from the starting point to ending point. Because the goal of our network is to interpret Kinect motion data at 30fps, the 120fps jitter-and-gap-applied data is downsampled to 30fps. This downsampling is also performed on the clean ideal joint curves. Fig. 2 details this process of intentionally corrupting joint curves for the creation of our simulated data.

Furthermore, all input training data was further augmented to allow for our networks to recognize greater variation in the input. In this process, 18 augmentations were constructed per walk cycle. Examples of these augmentations include reduced or increased jitter (0.5x to 2.0x), as well as variation in maximum gap length (0 to 100 frames). Variation in gap placement was also considered, particularly for instances in which gaps are more likely to occur at the top, middle, and bottom of arm movement.

C. Architecture and Training

The gap filling network is designed to recognize any frames of missing joint data and approximate new values for each

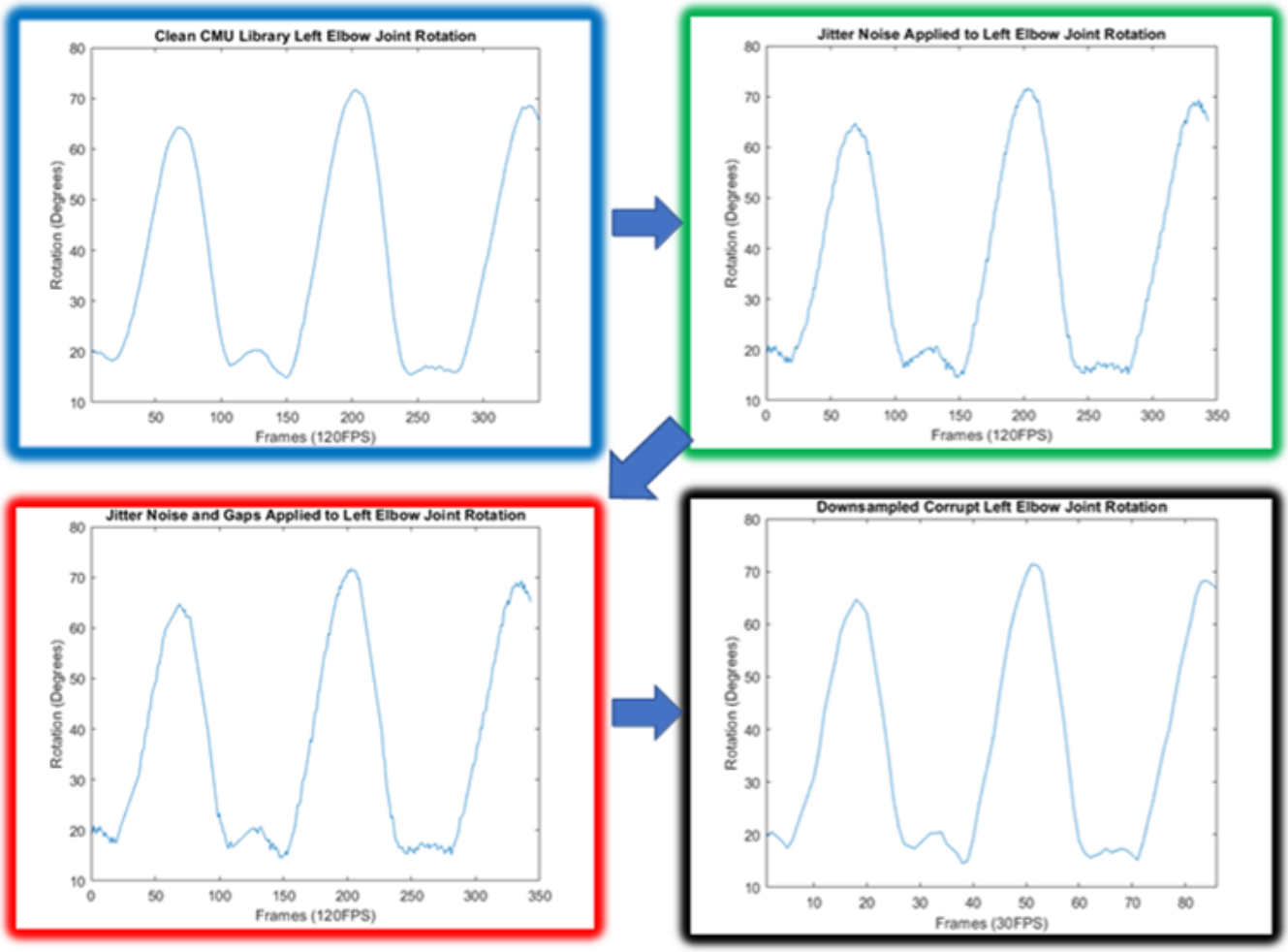

Fig. 2. The figure demonstrates each step for the application of jitter and gaps for the creation of simulated data.

of these frames. Based on a sampling approach described by P. Chaudhuri et al. in "A Deep Recurrent Framework for Cleaning Motion Capture Data," the simulated data input is restructured into 9 sample points: a current frame, 4 preceding frames, and 4 subsequent frames. Additionally, these 9 total sample points are each spaced by four frames. In adding this spacing to our input, we allow the network to better recognize the structure of the joint angle curve in the area local to the current frame, rather than the structure of the noise between the adjacent frames. To improve the gap filling network's ability to approximate values in the absence of data, we provide an additional input related to the joint data of the opposite arm. Therefore, the complete input for the network is comprised of two sets of 9 sample points, with one set containing 9 joint values for the left elbow, and the other set containing 9 values for the right elbow. Furthermore, the output of this network will be a single approximated joint value associated with the current frame. Two hidden layers for the gap filling network are each comprised of 31 nodes, utilizing tan-sigmoid transfer functions. [1]

For training the gap filling network, input was created from the simulated gap-applied and jitter noise-applied data for 20 walk cycles. The elbow joint data for each walk cycle was restructured to contain the required left-right pairs of joint angle sample points, with pairs generated for each frame in a walk cycle – excluding clipping of 16 frames at the beginning and end of each cycle. Separately, the training output was supplied in the form of individual joint values correlating with the current frame of the left input set. The network is designed to utilize joint correlation to solve for a single joint value in one of the two elbow motions. Therefore, we will be supplying the network with sample points from both the right and left elbows, but we expect to receive an output specific to only the left elbow. The following examples relate specifically to this input-output case. Additionally, because this network is designed to recognize and account for gap filling only, the training output was collected from simulated data with only jitter noise applied (utilizing the same set of walk cycles). This network was trained over 250 epochs, utilizing the resilient backpropagation training function for pattern recognition.

The noise removal network is intended to interpret the output of the gap filling network and provide joint rotation values indicative of smooth and natural human motion. It is the output of this network that will be recognized as the final corrected motion. The output of the preceding network is restructured into sets of 9 sample points: a current frame,

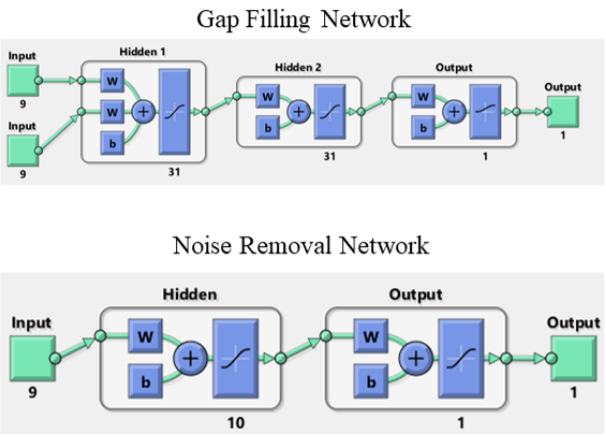

Fig. 3. The above figure demonstrates the network architecture for both the gap filling network and noise removal network.

4 preceding frames, and 4 subsequent frames. Unlike the previous network, there is only one set of sample points related to the specific elbow joint that requires correction – in our case, the left elbow. Additionally, there is no spacing between any of the sample points. This change in spacing aims to shift the focus of the second network to the noise profile of our data. The network may now recognize the change in joint rotation between adjacent frames of motion. The output of the noise removal network is a single frame of clean data associated with the current frame of the input. Only a single hidden layer of size 10 is utilized for this network.

In training the noise removal network, the gap-filled output of the previous network was restructured to sets of 9 adjacent sample points for each frame of the 20 original walk cycles – once again, excluding clipping of now 20 frames at the beginning and end of each cycle. The training output for this network was comprised of the original pre-cleaned data collected from the CMU library, downsampled to 30fps. Finally, this network was trained using Scaled Conjugate Gradient Backpropagation for plot fitting.

III. Results

A. Simulated Data

In supplying the networks with simulated data containing sizable gaps and a high degree of jitter, one may recognize that the network is successfully able to render cleaned joint data with the characteristics of natural motion. Fig. 4A demonstrates an instance in which extremely corrupt data was passed to the network. Though the network-approximated joint angle values do not perfectly match the intensity of the original pre-corrupted joint motion, the approximated curve still demonstrates a smooth and rounded peak. Additionally, in the areas of high slope – particularly the points at which the elbow is in mid swing – the desired slope intensity is matched by the network, resulting in elbow rotation speed and intensity that is accurate to natural motion. Similarly, as demonstrated in Fig. 4B, the network output demonstrates a significant improvement in structure over the corrupt data. While the corrupt data appears flat at its peak (indicating a

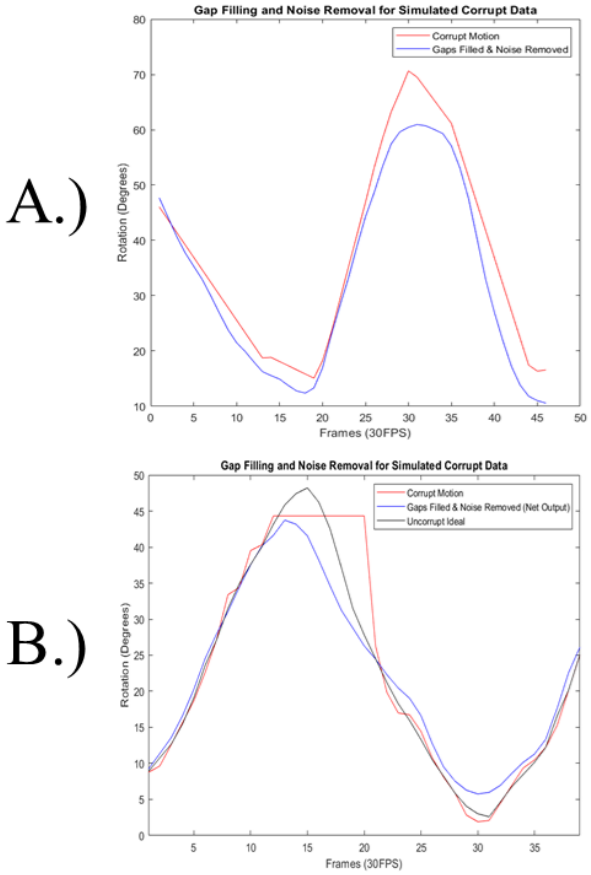

Fig. 4. The above figure demonstrates the corrupt motion, network output, and ideal clean motion for simulated test data. In this instance, the network output is intended to be approximated clean joint data for the left elbow. Each plot represents an independent test case.

point at which the elbow had become occluded), the network output renders a relatively smooth curve. Even in areas where the network output fails to match the uncorrupt ideal, the rendered motion is still void of jitter. The final network output is mostly smooth, albeit with slight bumps in the region of backswing (frames 20-30).

Unfortunately, the network approximated data often does not effectively render the subtle characteristics of the ideal joint curves. For example, Fig. 4A demonstrates how the networks fail to render the slight bounce of the elbow (indicated by the smaller peak between frames 5 and 20). Instead, because the region of this subtle bounce is encompassed by a gap in the corrupt data, the gap network interprets this bounce as the end point of a major elbow motion, creating a gradual slope in place of the bounce. This causes the first 20 frames of elbow motion to appear slower than intended. Additionally, the rendered output for simulated data often struggles to match the maximum and minimum of the ideal motion for major elbow rotation. For example, Fig. 4A depicts a 10-degree disparity between the maximum of the network-rendered motion and the ideal uncorrupt motion. Similarly, the minima are also dragged down to slightly lower rotation values, though this magnitude of this difference is much lower than that at the maxima.

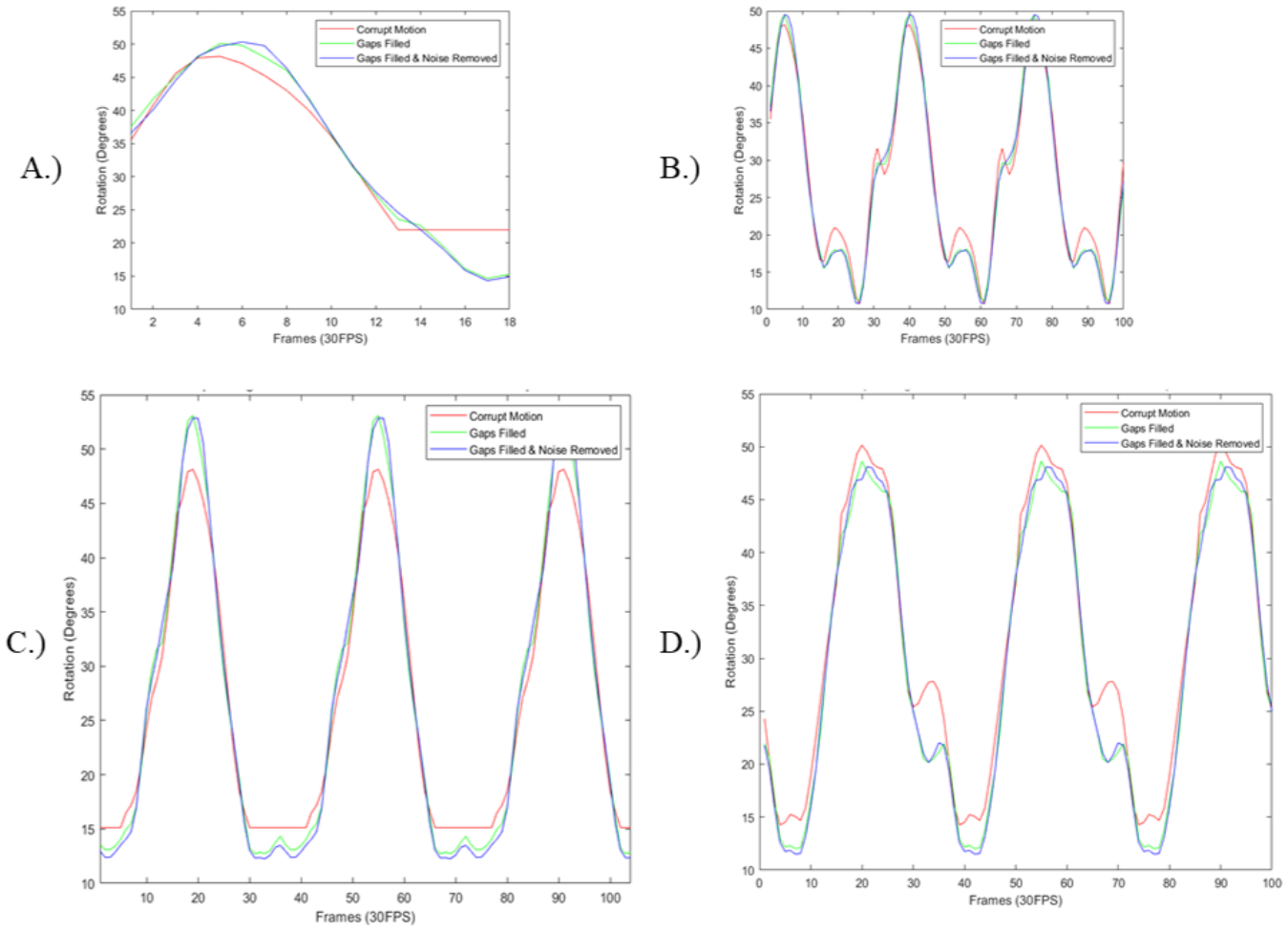

Fig. 5. The above figure demonstrates the corrupt motion, gap network output, and full gap filling and noise removal output for Kinect-captured elbow joint data. In this instance, the network output is intended to be approximated clean joint data for the left elbow. Each plot represents an independent test cases.

B. Kinect-captured Data

For testing, the Microsoft Xbox 360 Kinect was used in conjunction with iPi Soft for the capture of markerless test data. The Kinect captures two images: a typical 8-bit RGB image and infrared depth-based image. Both images are passed to iPi Soft, which approximates the location of human body parts from the subject's position in the depth image. Subsequently, joints from a specified skeletal rig are transformed to match the locations of these body parts in a 3D space. Joint labels are interpreted from the skeleton file and assigned to the proper body part. Finally, iPi Soft attempts to estimate joint rotation based on changes in the depth image between each frame of captured video (30fps). [2]

Unlike the simulated data, the network output for the Kinect-captured input surprisingly retains many of the positive aspects of the corrupt input. For example, while the simulated input may have led to issues with slope intensity, the Kinect-captured input does not include undesired deviation in slopes. Figure 5 depicts the network-corrected motion data for multiple Kinect-captured input curves. As demonstrated in Fig. 5B, C, and D, the slope of the network output – particularly the areas in which the elbow motion is in mid-swing – is very similar to that of the Kinect-captured data. In these instances, it is reasonable to believe that the network did not interpret any issues with these slope intensities (no gaps or recognizable jitter), and therefore did not make any unnecessary corrections. Separately, when jitter could be found on a slope, as with Fig. 5B (frames 27-33), the network successfully removed the jagged rotational values, and replaced this area of jitter with a smooth curve. In areas where gaps were present, Fig. 5A (frames 13-18) and Fig. 5C (frames 30-40), the network successfully approximated curve structure and magnitude in areas of absent data. Additionally, unlike with the simulated input, the network output for Kinect-captured data appears to overcompensate in areas of maxima and minima. Whereas this could be an issue for the simulated data, the structure of the Kinect-captured data is assisted by this overcompensation. A frequent issue associated with Kinect capture is the inability for software like iPi Soft to accurately match and render the intensity of elbow swing at the extents of motion. This often leads to stiff and dull arm movements. Therefore, there is no real concern for the network to slightly overcompensate by around 5-10 degrees. If anything, this feature improves the

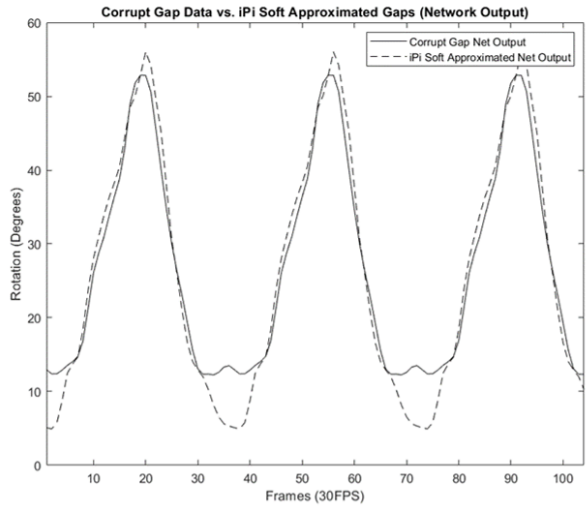

Fig. 6. The above figure demonstrates the full network output for two sets of inputs: corrupt data in which the gaps are left unaltered, and corrupt data that has had gaps filled by iPi Soft.

realism associated with this movement.

Occasionally, software such as iPi Soft may offer optional gap filling for occluded joints. Fig. 6 demonstrates two versions of a rendered joint movement: the first with gaps left untouched, and the second with iPi Soft gap filling applied. As depicted by the minima of Fig. 6, iPi Soft fails to accurately approximate joint movement, filling in negative values in place of the gap. These negative values cause the joint to appear as though it is bending backwards. However, if this incorrectly approximated joint curve is subsequently passed through our series of networks, the joint curve is restructured to appear much more accurate to human motion – the joint values now remain positive, with a smooth curve replacing the pointy iPi Soft output rotation values.

C. General Network Limitations

Finally, there are some limitations associated with our networks. These limitations are primarily associated with the specificity of the network inputs. For example, our network input is restricted to joint data of 30fps. Additionally, our training and test data is restricted to a single plane of rotation. If a user were to attempt passing in full XYZ rotational data, the network would be unable to properly interpret the greater variety of joint angle values. Furthermore, all training data obtained from the CMU Mocap Library utilizes the same skeletal rig. Therefore, if a network input is collected from a significantly different rig, or with skeletal files specifying different DOF or offset values, the network output will vary as a result.

Specifically for the noise removal network, there are some instances in which the network will overcompensate for jitter. For example, if the series of networks is fed a full set of joint data for the right elbow but no data for the left elbow (with all joint values equal to 0), the gap filling network will be forced to approximate an entire left elbow movement. The gap filling network is typically able to accomplish this task, as depicted in Fig. 7A. The output of the gap network, for this instance, is indicative of ideally smooth motion characteristics. However, if this gap filling output is passed to the subsequent noise removal network, one may recognize that the noise removal output has disturbed some of these smooth qualities, particularly in the peaks. These deviations do not ruin the motion by any means, but they are noticeably more jagged. Separately, for the case in which the network is supplied with a full set of left elbow joint data, but no right elbow data, one will recognize a significant and unwanted change in the curve structure. As demonstrated in Fig. 7B, the left elbow joint input already has an a smooth and clean shape. Therefore, the networks should ideally output a curve structure similar to the input. However, due to the network's dependency on joint correlation, the absence of right elbow joint data causes the output to become warped, with irregular motion between the two major peaks.

IV. Conclusion

The series of networks constructed for this project effectively interpret and correct Kinect-captured data to mimic the characteristics of smooth and natural motion. For the correction of other joints or limbs, the structure of these networks may act as a base framework. Through further implementation of joint correlation, as well as the extension of single-dimensional rotation to multi-dimensional rotation, these networks have the potential to allow for full skeletal correction.

References

[1] Mall, Utkarsh Lal, G. Chaudhuri, Siddhartha Chaudhuri, Parag. (2017). A Deep Recurrent Framework for Cleaning Motion Capture Data.

[2] "An introduction to the Kinect sensor," An Introduction to the Kinect Sensor — Microsoft Press Store, 15-Jul-2012. [Online]. Available: https://www.microsoftpressstore.com/articles/article.aspx?p=2201646. [Accessed: 16-Dec-2021].

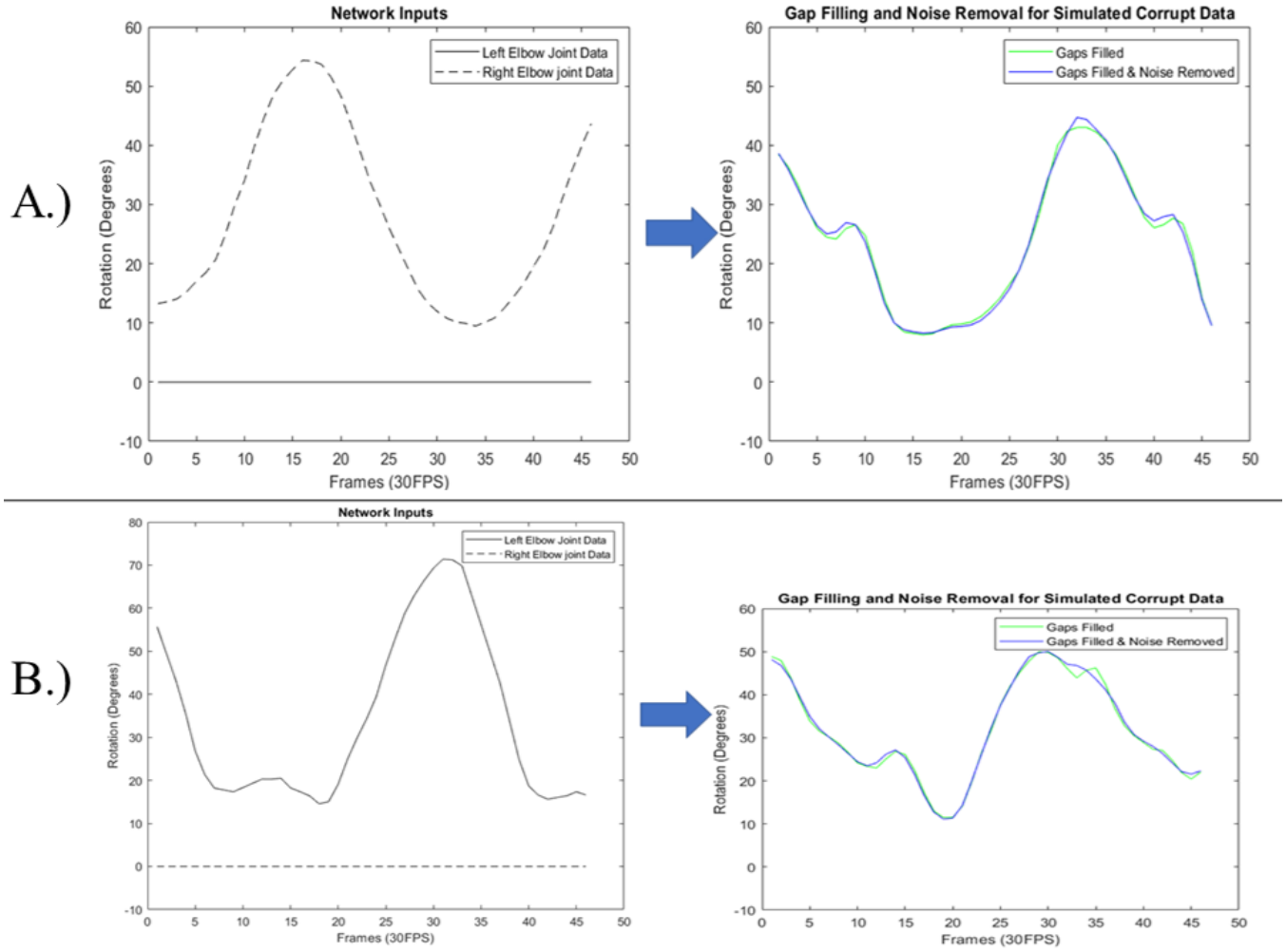

Fig. 7. The figure above demonstrates the network output for simulated input containing the case of complete data for the right elbow and no data for the left elbow (A), or full data for the left elbow and no data for the right elbow (B). In this instance, the network output is intended to be approximated clean joint data for the left elbow.