Inferring Mocap Data from Wrist Worn IMU Sensors

Ryan Baker and Zhiheng Wang

***Abstract*— The ability to have 3 dimensional position information of individual joints of upper limbs has great possibility to improve studies in countless fields such as healthcare, sports bio-mechanics, and countless others. However, obtaining this information currently requires a sophisticated motion capture (mocap) system. These can be prohibitively expensive, cumbersome to use, due to a trained professional being required, and cannot be used outside of a laboratory setting.**

We propose a solution to utilize Inertial Measurement Units(IMUs) to collect accelerometer and gyroscope data, to be used in a transformer based model to predict upper limb joint positions.

Our pipeline consists of leveraging existing motion capture datasets to convert into synthetic IMU data using VirtualIMU, to then be used to train our transformer model. This transformer model receives a window of IMU data, as well as a starting pose to produce an estimate of the pose at the end of the window. We apply limits based on the limitations of the human body to constrain our model's output to ensure that the predicted pose is feasible given constraints of human anatomy.

I. Introduction

Motion capture is considered the gold standard of human movement measurement. It has high accuracy and measures positions and orientations of a large number of joints on the body. It however can be cumbersome, expensive, and requires observation in a laboratory setting.

Inertial Measurement Units (IMUs) are relatively affordable, less intrusive, and can be collected outside of a laboratory setting. A major drawback of IMU measurements is the fact that it can only measure acceleration at one point on the human body, often the wrist.

Combining the ability of out of laboratory measurements with the ability to estimate positions of upper limb joints improves the accessibility of data. This data has the ability to provide insight about rehabilitation, bio-mechanics in sports, and other fields that require human motion data.

Our approach aims to define a new framework to generate synthetic mocap data to predict upper limb posture using two 6-axis IMU sensors located on each wrist.

II. Data processing

A. ACCAD Dataset

The Ohio State Advanced Computing Center for the Arts and Design (ACCAD) dataset is a subset of the data used to generate the AMASS dataset. From this dataset, 3 separate subjects' mocap data was used. Each subject had between 69 and 149 different mocap recordings.

These mocap recordings are stored in a BVH file format. Biovision Hierarchy file (BVH files) are a text based structure to store human body information, and key-frames to recreate motion of joints. The start of a BVH files defines a tree of joints, in our case, the root was the hip joint. From the root, children are defined with an offset from their parents. In the motion section, key-frames define changes in orientation for every joint, relative to their parent joint. These BVH files do not store positional key-frame information for any joint other than the root. Each joints position is known, and can easily be calculated using the offset and orientation provided.

B. IMU Pipeline

1) Extracting global coordinates: Utilizing extractWristCoordsBVH found in the VirtualIMU repository, the forward kinematics of the BVH file are computed. This function starts at the root of the tree and calculates its position relative to its parent. This is done using bvhtoolbox's get_affines function. To get the joint's global coordinates, the joints transformation relative to its parent is multiplied by its parents. Since this approach starts at the roots and traverses down the tree, and the root's global position is known, the resulting product is the global position and orientation of each joint. We chose to only extract six joints from the BVH file, both arm's shoulder, elbow, and wrist.

2) Locking neck position: We believe that in the use case of our system should not be dependent on any motion related to lower body motion. To ensure that the IMU data does not include acceleration due to lower body motion, we decided to "lock" the neck at the origin. This is achieved by subtracting the neck's global position and orientation from each joint.

We believe that doing this will achieve the following:

- Only acceleration from arm movements will be included in the virtual IMU data
- The ground truth data will always be 0 centered, making positions far more correlated with IMU data.

3) Wrist position to IMU: To obtain the IMU data from the wrist positions and orientations, the VirtualIMU Monte Carlo augmentation was used. This approach applies a Savitzky-Golay filter and an adaptive mean filter before taking the second derivative of the position and the first

derivative of the orientation. The IMU data is then aligned with IMU axes before having another Savitzky-Golay filter applied. It is finally interpolated before being saved.

C. Dataset Creation

Using the synthetic IMU data and the mocap, we aim to make a dataset that contains all information necessary to predict a set of joint positions from using our model, as well as a set of joint positions to compare it to. We decided on a windowing approach for the data. Each window is considered a gap in time where no mocap data is available, only IMU data. The task of the model is to infer using the IMU data how the mocap data will change. To achieve this, we create a 0.25 second window of IMU data, using the start and end times of the window, we find the mocap data that most nearly matches these time stamps. We use the first mocap data as an input to our model, and the last one as the expect output for training. We believe this approach has these benefits:

- It is possible to combine multiple recording into a single batch for training, since all windows are completely independent from each other.
- Since the mocap data is 30 Hz and the IMU data is 100 Hz, windowing the data in this way allows for time synchronization without interpolation or down sampling.
- We are able to specify the window size, and use it as a hyper-parameter.
  - Smaller window sizes would likely allow faster convergence, but too small would likely not allow the model to learn more than just to return the starting pose.
  - Larger window sizes would likely penalize incorrect more, possibly making a more robust estimator, too large could lead to too much variance to allow learning.

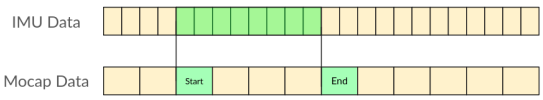

Fig. 1. IMU Windowing and Mocap Ground Truth

III. Method

A. Loss Function

The model's output is compared to the ground truth mocap joint positions at the end of the window. The Mean Squared Error(MSE) is used to get a loss value for the model's gradient update.

To constrain the output's motion by the limitations of human ability, a penalty is applied for accelerations beyond human ability. According to a published paper on the topic, humans are incapable of accelerating their hands above 6 m/s^2. Using this idea of penalizing excessive speeds, a ReLU term is added to the loss if their change in distance exceeds 0.2 meters in a single window, the added term is nonzero, and 0 otherwise. This 0.2 meters was chosen since windows in the dataset had a maximum motion of this distance. In order to limit the acceleration, the velocities would also need to be stored in the input and output data. If velocities were stored an accurate change in velocity, or acceleration could be measured.

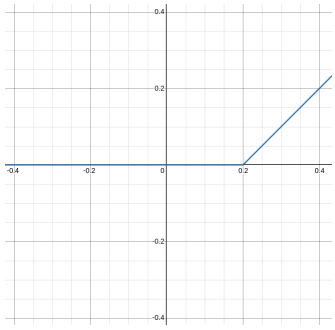

Fig. 2. ReLU Δd penalty

B. Constraining Model Output to Bio-mechanically Feasible Positions

Predicting points without restriction allows the majority of points to be impossible within human limits. An example of a human limitation is the length of human limbs. The distance between connected joints should remain constant between estimates.

To accomplish this, we add the output from the last network layer to the input positions. This will result in a vector pointing in the direction from the joint's parent that the joint will be. This vector is normalized to become a unit vector with the same direction. It is then multiplied by a pre-computed joint distance to give it the correct magnitude. Finally, the position is added to the parent's joint position to convert the joint's position relative to the parent to its position in the global space.

By enforcing the lengths of limbs, possible outputs have a lower expected loss value. I believe that this has the benefit of reducing the number of epochs needed to reach convergence, and that it is more likely to converge at a better prediction.

C. CNN implementation

Our main plan was to use a transformer model. With all of the issues we faced with the transformer model, we decided to create a CNN based model to be able to have something to submit.

For this implementation, we eventually settled on a model that utilizes 4 convolutional layers. Each of these convolutional layers are 128 channels, and kernel sizes of 5,3,3,3 respectively. Each of these convolutional layers are followed by a non-linearity. After the last convolutional layer, the

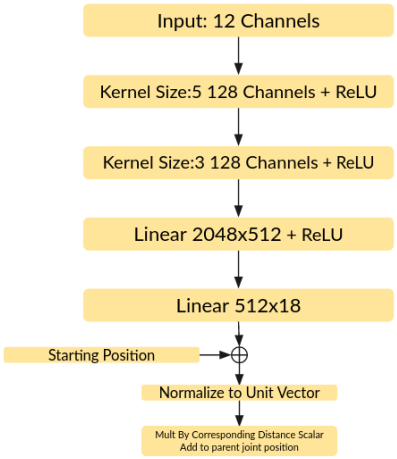

Fig. 3. CNN Model

matrix is flattened. Finally there are 2 linear layers of sizes $1920 \implies 512$ and $512 \implies 18$ respectively. Each of these linear layer have a dropout applied. The limb lengths are enforced on the output of this model by applying the process in section III-B.

D. Encoder

The two 6 axis IMUs give us 12 channels of data as an input to our model. It is preferable to have a higher dimensional input for our transformer model. To achieve a higher dimensional input, we use a pre-trained encoder to project the 12 channel input into a 64 channel representation of the data.

This encoding is achieved using an AutoEncoder network. This network is broken into two separate parts, an encoder and a decoder which is a mirror image of the encoder. For this, we utilized a very simple model, 3 convolutional layers of 64 channels with a kernel size of 1.

This model is easily able to achieve a low loss projection in a few number of training steps. Because of this, and issues face with gradients becoming NaN upstream from the transformer, we chose to pretrain this portion, and apply it to downstream task without back propagation.

E. Transformer Implementation

Our main idea was to utilize a transformer model. We believe that a transformer model can best represent the data presented, we believe that the self attention aspect of the model will be capable of transferring IMU information directly into the start position tokens before adding the output to the start position. The ability to utilize the information about the starting joint positions during layers is something that is not as possible using a CNN model architecture.

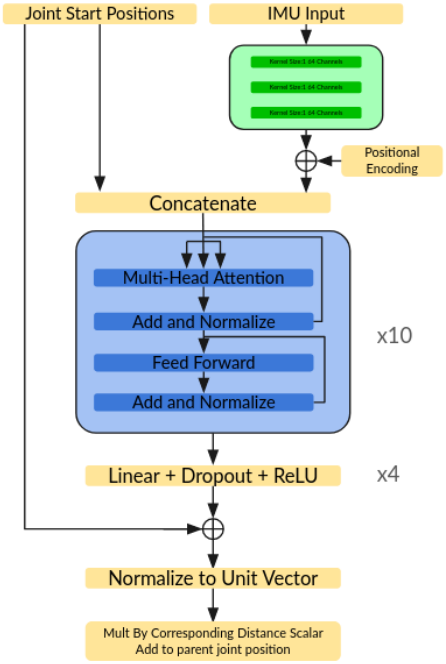

Fig. 4. Transformer Model

Each transformer layer starts with a multi-headed attention layer, which learns which tokens in the IMU/Start position data to attend to. For example, it should likely learn that the a certain direction of acceleration combined with a given orientation is most relevant for a certain axis of a joints position. In other words the attention heads learn which portions of the IMU data correspond to which joint position.

Our approach starts by encoding the IMU window into a higher dimensional space, as explained in section III-D. We start by concatenating the starting position to the higher dimensional encoding of the IMU data. This in theory should allow the contextual information of the IMU data be embedded into the starting position. The positional encoding information is then added to it, before going into the transformer layers. We chose to do 10 layers, each with 8 attention heads per layer. After the transformer layers, a total of 4 linear layers are applied to the output, each of which are a size of 64, with a dropout and non-linearity between each. Finally the limb lengths are enforced by applying the process in section III-B.

IV. Evaluation

In order to evaluate our approach, we sampled 1000 0.25 second windows from the test dataset. We first plotted the ground truth positions and made it into an animation.

Next, we also evaluated our model on a ”single shot” prediction. In this case, we provided our model with each window of IMU data, as well as the known ground truth data at the start of the window. We then plotted the output of the

model for each window. This plot resembles the motion of the ground truth. Providing the ground truth of the start of each window is an easier evaluation for the model, since it does not allow error in its predictions to compound over time.

Our last evaluation method involved compounding error over time. In this case, we gave the model the first ground truth pose, and all future inputs to the model were the previous window's prediction output. This allows the error in models to compound on each other, and we found that this highlighted some of the limitations in our current approach.

As we will mention in section V, our assumption about locking the shoulders with the spine allowed shoulders more mobility than expected. For the plots provided, we assumed that it would be possible to simply assign the shoulder position as the input shoulder position. This worked well for the "single shot" approach since the new ground truth inputs allow the shoulders to update, however the continuous test shows that the shoulders will never update, where they should. Since arm lengths are enforced and the shoulders aren't allowed to move, the rotation of the torso leads to a failure of the predictive ability of our model.

Video demonstrations for each evaluation are available here.

V. Improvement

While demonstrating the transformer results, it seemed that locking the spine coordinate for IMU and the shoulder for output was not an ideal solution. The assumption made was that the shoulder does not have a large amount of mobility relative to the spine. This assumption overlooked the fact that the human leaning allows the shoulders to move relatively large distances in the global space, while not moving far relative to the spine. A few solutions possible to fix this include:

- Lock an arbitrary midpoint between the shoulders rather than the spine.
- Lock the spine orientation along with position, making rotations of spine joint in global space not affect all upper body joints

Another faced problem was the issue of the transformer's gradient values suddenly shooting to NaN. We suspect that it is occurring somewhere within the transformer layers. When placing a conditional breakpoint for when a gradient is first calculated as NaN, we see no issues with weights being large, or any data being large. We also attempted to normalize prior to attention and feed forward layers, but it had no effect.

An improvement that could theoretically improve the results could be to take limitations of human joint angles into account. This could be as simple as clipping the output joint angles to withing a possible range. Another possibly more informative approach could be to interpolate the motion required to make the movement predicted and if the motion is not likely possible, penalize it, or disregard it completely.

There is also a possibility to make a probabilistic to this solution, instead of having a single output point for each joint, a probability distribution could be given, then using known priors about human kinematics, the most likely pose could be chosen.