Master of Science in Engineering: Game and Software Engineering
June 2020

Tree structured neural network hierarchy for synthesizing throwing motion

Mattias Fredriksson

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Engineering: Game and Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Mattias Fredriksson
E-mail: mattias-fredriksson91@hotmail.se

University advisor:
Dr. Prashant Goswami
Department of DIDA Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Realism in animation sequences require movements to be adapted to changing environments within the virtual world. To enhance visual experiences from animated characters, research is being focused on recreating realistic character movement adapted to surrounding environment within the character's world. Existing methods as applied to the problem of controlling character animations are often poorly suited to the problem as they focus on modifying and adapting static sequences, favoring responsiveness and reaching the motion objective rather than realism in characters movements.

Algorithms for synthesizing motion sequences can then bridge the gap between motion quality and responsiveness, and recent methods have shown to open the possibility to recreate specific motions and movement patterns. Effectiveness of proposed methods to synthesize motion can however be questioned, particularly due to the sparsity and quality of evaluations between methods. An issue which is further complicated by variations in learning tasks and motion data used to train models.

Rather than directly propose a new synthesis method, focus is put on refuting existing methods by applying them to the task of synthesizing objective-oriented motion involving the action of throwing a ball. To achieve this goal, two experiments are designed. The first experiment evaluates if a phase-functioned neural network (PFNN) model based on absolute joint configurations can generate objective oriented motion.

To achieve this objective, a separate approach utilizing a hierarchy of phase-function networks is designed and implemented. By comparing application of the two methods on the learning task, the proposed hierarchy model showed significant improvement regarding the ability to fit generated motion to intended end effector trajectories.

To be able to refute the idea of using dense feed-forward neural networks, a second experiment is performed comparing PFNN and feed-forward based network hierarchies. Outcome from the experiment show significant differences in favor for the hierarchy model utilizing phase-function networks.

To facilitate experimentation, objective oriented motion data for training network models are obtained by researching and implementing methods for processing optical motion capture data over repeated practices of over-arm ball throws. Contribution is then threefold: creation of a dataset containing motion sequences of ball throwing actions, evaluation of PFNN on the task of learning sequences of objective oriented motion, and definition of a hierarchy based neural network model applicable to the motion synthesis task.

Keywords: Motion Synthesis, Neural Networks, Global Multibody Reconstruction.

Sammanfattning

Realism i animerade karaktärer kräver rörelser som är anpassade för föränderliga miljöer som finns inom virtuella världar. För att uppnå en djupare upplevelse har forskning därmed börjat fokusera på möjligheten att återskapa realistiska rörelser anpassade till karaktärers omgivning, något som tidigare metoder inte har möjligheten att uppnå då de fokuserats på att effektivt anpassa rörelserna till kommandon och mål som utsatts av spelaren på bekostnad av rörelsers realism. Således har flera metoder för att generera rörelse sekvenser lagts fram vilka har möjligheten att återskapa specifika rörelser och rörelsemönster. Perspektivet från vilka metoder lagts fram är dock problematisk eftersom jämförelser mellan metoder saknas eller är otillfredsställande. Problemet kompliceras således av att förutsättningarna för metoder varierar då rörelse mönster som metoder anpassats för skiljs åt, samt att data från vilka rörelsernas mönster baseras på varierar.

Fokus på motbevisning av möjlig applikation av olika metoder kan därmed ses likvärdig med problemet att framlägga ytterligare metoder. För att både specificera en möjlig lösning till problemet med att skapa rörelser för bollkastning, samt att visa oapplicerbarhet för en existerande metod utförs två experiment. Första experimentet påvisar att ursprungliga metod för fasbaserade neurala nätverk (PFNN), där leder parametriseras inom en fast referensram, inte kan prestera likvärdigt som en metod där en hierarki av neuronnät uppdelar problemet och möjliggör sekventiell lösning av delproblemen som uppstår. Jämförelse av metoderna påvisar därmed skillnader, där slutsats kan dras att den hierarkiska strukturen bättre kan följa de rörelsebanor som utsetts för specifika leder inom karaktärens representation. För att påvisa att användningen av andra artificiella nätverk inom hierarkin inte presterar likvärdigt jämförs applikationen av fasbaserade nätverk med 'feed-forward' nätverk. Slutsatsen utifrån det andra experimentet påvisar därmed en signifikant prestationsförbättring för fasbaserade nätverk.

Möjliggörandet för experimenten kräver därmed framtagning av skelettanimationer vilka avspeglar kaströrelser, vilket uppnås genom att återskapa skelettrepresentationer från punktmoln som anskaffas genom registrering av markörer under rörelsers utförande. Bidraget som uppnåtts är därmed trefaldigt: framställning av dataset innehållande kaströrelser, evaluering av PFNN och dess möjliga applikationer för att återskapa mål orienterade rörelser, samt hur en hierarkisk modell kan struktureras och appliceras på problemet med rörelsers återskapande.

Nyckelord: Rörelsesyntes, Neurala nätverk, Rekonstruktion av flerkropps system.

Acknowledgments

I would like to extend my thanks to Prashant for enduring as I was endlessly sidetracked trying to complete the thesis, and to my parents for supported my efforts at completing it.

Contents

Chapter 1

Introduction

1.1 Background

Synthesizing human motion has become an important area of research in the development of animation tools and interactive applications (e.g. games or virtual reality - VR). While synthesizing entirely realistic animations are not yet possible, a number of approaches has been proposed for synthesizing motion that adheres to some of the necessary criteria (e.g. [16, 32, 67, 63, 48, 54, 59]).

Complexity in human motion is a reason to why achieving realism in synthesized motion is not yet possible. Motion as expressed in character animations do not only need to be based on accurate body motion, but animations also need to be consistent in relation to facial motion and the surrounding environment. Additionally, expressed motion must contain consistent stylistic features that are important for the personality of the animated character [21].

Creating stylistic consistent motion are essential when the focus in the visual medium are emotion and personality exhibited in a character's motion (as in animated cartoons and movies). For interactive applications on the other hand, animation systems cannot rely on fixed animation sequences adapted to a known environment. Dynamic character animations must be generated and adapted to an environment changing in real-time. Character's animation sequences also need to be both immersive and responsive regarding the user experience. The challenge is then quite different from the problem when providing animations for a known and unchanging environment. The ability to synthesize motion sequences is therefore a key in improving animation systems operating in real-time contexts.

Accessibility of computational power, development cost and flexibility are some of the factors that has made motion synthesis viable for adaption in state-of-the-art animation systems. Older solutions like parameterized blend-trees which utilize manually coded animation sequence transitions and blend states [26] have been replaced. Newer techniques such as motion matching [16] can now be used to automatize some of the manual labor involved in blend tree construction. Automation then help by improving the 'rapid iteration' [26] during development of a character's animation controller and simplifies the development process.

The term motion synthesis within the context of animation systems is a bit diffuse. Methods like motion matching primarily act as animation controllers rather motion synthesizers, utilizing interpolation techniques to generate new animation sequences during transition between two or more animation clips. Motion matching instead rely on playback of recorded animation sequences and post processing to reach the

intended goal rather than generate accurate sequences adapted to the environment.

On the other end, approaches based on time-series models [32, 67, 63] provide the ability to generate continuous motion sequences. Utilizing the current joint configurations to generate future motion states from feature descriptions of the intended motion. With different motion synthesis techniques available in research, determining the method best suited for implementation can be complicated. Complications in determining suitable methods is a problem that is exacerbated by the lack of adequate comparison between different methods.

The problem posed for this project is therefor to analyze existing motion synthesis methods and investigate if the method can generalize beyond training data. To facilitate an analysis of existing techniques, motion data defined by a specific objective-oriented action is used. By extracting skeletal representations from optical motion data, experiments can be constructed to evaluate the ability of existing techniques to generate time-series models. Action chosen for this purpose is then the activity of throwing a ball. Ball throwing provide motion that is both extensively researched and require accurate joint movements to achieve the end goal (of hitting a target) [10].

Once implemented, synthesis algorithms are tested in their effectiveness and ability to synthesize motion completing an action and reaching its goal. Furthermore, due to the number of possible applications (such as for robotics [2], games [16, 57] and biomechanics [31]), there exist a large set of approaches solving slightly different problems utilizing different input data.

Due to the number of available methods, a recently proposed algorithm PFNN (Phase Functioned Neural Network) [32] is implemented and tested in regard to both a hierarchical network structure and a single network approach similar to the original design. The ability of the neural network algorithm to generate time-series models over objective oriented motion sequences is also evaluated in comparison to an equivalent feed-forward neural network. Contribution will then be in evaluating the specified methods by measuring their performance within the context of a designed experiment. Evaluating the ability of trained models to generalize over the task of synthesizing throwing motion, an area where the algorithm has not been tested before.

1.2 Aim and research questions

Goal for the project is to investigate existing neural network (NN) algorithms ability to train time-series models for the purpose of synthesizing goal-oriented action sequences for biped characters. For this purpose, a few selected methods are evaluated in their ability to synthesize motion reaching the intended goal associated with each generated action sequence. Due to the number of possible applications, execution time for trained models should be limited within real-time constraints defined by a simple interactive application. To reach the goal, the project consists of three steps:

1. Processing of optical motion capture data to reconstruct skeletal representations within each sequence.

2. Construction of a dataset containing action sequences (animations) of biped skeletal models throwing a ball from reconstructed sequences.

3. Evaluation of motion models trained on the generated dataset. Motion models are tested on the task of synthesizing throwing motion and evaluated by their ability to generate motion where the simulated ball impacts the intended target.

In order to analyze the effectiveness and immersion of synthesized motion an experimental approach is used. Experiments are performed to answer following research questions (RQ):

Research questions

RQ1: *"Comparing the original PFNN approach presented in [32] with a PFNN based hierarchy model, which algorithm perform better at synthesizing objective oriented motion sequences of biped characters throwing a ball?"*

RQ2: *"In the case of motion state predictor algorithms based on dense feed-forward or phase-functioned neural networks, which algorithm performs better at training motion models effective at synthesizing motion for biped characters throwing a ball?"*

1.3 Outline

The project takes a data set containing recordings of optical motion trackers for multiple subjects, reconstructs the skeletal representation, and ends by encoding a minimal set of features in different neural net models.

To facilitate a structure in the report and cover all relevant subjects, sections follow the order in which tasks are resolved. Each chapter begins with the problem of skeletal reconstruction, followed by trajectory reconstruction, and ends with mentioning methods for encoding and decoding neural nets for motion synthesis purposes.

During introduction and related work, focus is put on covering the primary subject of NN based motion synthesis. Primary subject in this context refer to the topics associated with the research questions introduced in previous section. The theory chapter provide a balance to earlier sections regarding focusing on covering the problem of skeletal reconstruction.

Methodology is divided in four chapters. The first chapter briefly presents research questions and experimentation protocol. Following three methodology chapters describe the approach to solving each problem before training models can be evaluated and analyzed in the subsequent chapter. Remaining chapters complete the thesis by discussing the outcome and providing a conclusion to the introduced research questions.

1.4 Mathematical notation

Mathematical notation used in the thesis.

$\boldsymbol{v}$: Column vector.
$\boldsymbol{v}^T$: Row vector.
$\boldsymbol{M}$: Matrix.
$\odot$: Element wise multiplication for vectors and matrices.
$\Phi(\boldsymbol{X}, \boldsymbol{\beta})$: Evaluation of neural network in respect to input $\boldsymbol{X}$ and parameters $\boldsymbol{\beta}$.
$\Theta(\boldsymbol{p}; \boldsymbol{\beta})$: Evaluate weight and bias for all layers in a phase function network.
$\boldsymbol{W}^{[l]}$: Neural network weights for the l:th layer.
$\boldsymbol{a}^{[l]}$: Activation output the l:th layer.
$\boldsymbol{b}^{[l]}$: Bias vectors for layer l.

1.5 Dictionary

Abbreviations or frequently occurring concepts.

- **CoR**: Center of rotation, pivot point, for a rigid body segment.
- **Multi-link chain model**: Modeling of the musculoskeletal structure as a sequence of rigid bodies connected by joints.
- **SBO**: Single-body-optimization, the process of estimating a single rigid body at a time.
- **MKO**: Multibody kinematics optimization, the process of estimating parameters for a chain of rigid body links.
- **HJC**: Hip-Joint-Centre, internal anatomical landmark for the CoR of the joint between femoral head and hip.
- **STA**: Soft tissue artifacts, non-rigid deformations that occur in relation between the skeletal representation and the skin when modeling musculoskeletal structures.
- **DoF**: Degrees of freedom, number independent parameters used to define a configuration state. Defines the number of generalized coordinates of a system.
- **MSE**: Mean squared error.
- **NN**: Neural network.
- **PFNN**: Phase functioned neural network.

Chapter 2

Related work

Application of neural networks within the context of learning and encoding motion has enabled a large variation of network structures to be proposed. A common method applied to the motion synthesis problem is to learn time series models for predicting following motion state given a parameterization of the current state. However, other approaches exist and a few of those are covered here.

With the stated purpose of evaluating suitability of existing methods as applied for the motion synthesis task, this section intends to give a background to existing methods. Determining the method suitable for a given task is not straight forward, however. The problem is also often further complicated by the lack of, or minimal comparison with previous methods.

Autoencoders

Convolutional autoencoders are used by Holden et al. [34] to learn encoded motion manifolds within overlapping motion windows (overlapping sub-sequences) in the training data. By posing denoising of corrupt input data as a minimization problem, they show that a trained NN model can reconstruct noisy or missing motion data by feeding it through the encoding and decoding process of a convolutional neural network.

Improving the initial work by convolution autoencoders. A feed-forward network is combined with the previous work in [33]. Utilizing a feed-forward network to generate activation signals in the space of hidden units as encoded within a trained convolutional network. The feed-forward network can learn to generate activations from higher level features such as trajectory parameterizations. Synthesis of motion sequences is then possible by using the synthesized activation as input to the decoding function of the convolutional network.

Using the synthesized activations, spatial minimization constraints can be defined on poses decoded from the synthesized hidden state. Optimization of an initial set of generated activation parameters is then possible by defining a loss function with regard to the set of decoded skeletal poses.

Synthesis of activation signals are also possible through optimization as shown in the same work. Simultaneous minimization of differences between **Gram matrices** is shown to be useful for transferring stylistic features content between motion sequences. New activation signals can be generated by minimizing the sum of Gram matrix differences between a randomly generated signal and encodings from motion sequences. Through minimization, features from the encoded sequences are then

transferred to the generated signal. Indicating that stylistic transfer is possible using a similar approach for transferring artistic style between images.

Transferring features between two motions is perhaps interesting but irrelevant for this work. Constraints on the decoded spatial domain and the ability to synthesize motion using high-level features such as motion trajectories is on the other hand relevant. The approach also indicates the ability of neural network encodings to generate motion from high level features. Highlighting the importance of cost functions in the spatial domain for constraining generated output to the motion associated with those features.

Phase function networks

Phase functioned neural networks (PFNN)[32] has displayed natural results in biped locomotion, and has been applied to the task of motion state prediction. Phase function network expands the concept of feed-forward networks, utilizing a function (or 'phase function') to interpolate between a set of different weight configurations over a domain defined by a phase. Since a phase can be hard to determine in many situations, further development of the idea used a mixture of expert approach to create quadruped locomotion controllers.

Mode-Adaptive Neural Networks (MANN) [67] use a separate gating network to calculate the coefficient for each weight configuration using a subset of the motion state features. Replacing the phase function and phase definition by an expert network. A separate 'motion prediction network' equivalent to a PFNN takes the calculated weights and feature vector and generates the next motion state. Results from MANN indicate improved foot skating artifacts compared to other NN solutions with the primary benefit of being able to learn a phase in the motion in an unsupervised approach.

Generative adversarial networks

Generative adversarial networks (GAN) also provide approaches that shouldn't be ignored. Usable as an unsupervised learning method it can be used to reduce the necessity for manual labeling or other tasks required by other methods. GAN introduces a separate 'critic network' to ensure quality in generated motion. In common with other applications of GAN, critic networks can also be applied to refine the loss function during supervised learning.

Introduction of multiple networks is also possible as shown in [6]. In addition to training a generative network using an adversarial critic, a third network is introduced as a discriminator network (commonly the critic is called the discriminator). By training the generative network to predict a set of future poses from a sequence of previous sequence to form a continuous set. The discriminator is trained in parallel for the purpose of evaluating quality of generated motion. By training on distinguishing between real and generated motion, the discriminator learns to predict the probability that the full sequence representing valid human motion. Acknowledging that methods focused on motion validation is sparse, the discriminator provide an interesting approach albeit at a computational cost.

Further uses of GAN are also proposed by Wang et al. [63]. In the proposed

method GANs are used to further refine a motion sequence generated with a **recurrent neural network** (RNN) trained separately. Applied to the learning task of generating gait motion, the proposed method proved to generate motion with a realism close to that of original motion capture data used for training. It also improved accuracy in following the intended locomotion trajectory compared to PFNN, but at a greater processing cost. However, as the method relied on multiple motion specific cost functions, the authors acknowledge that further work is necessary to adapt the approach to other types of motion.

Both mentioned approaches show that GAN is useful when applied to motion synthesis tasks, particularly as they gradually optimize the training procedure during training. The problem with utilizing GAN is then only the increase in training time for generating new prediction models. While utilization of proposed methods in a real time context is complicated due to the structure of the generative networks. Application is dependent on the specific algorithm designs rather than limitations of the method itself. As shown using the refinement approach in [63] foot sliding issues in generated motion sequences can be resolved using GAN.

Reinforcement learning

Other approaches for motion synthesis are also possible, for example **reinforcement learning** has shown to be a relevant approach [55]. In comparison to a pose prediction method based on training data, reinforcement learning is oriented toward learning policies for controlling physical simulations. In some cases, concepts of policy learning and motion state prediction do not have to be distinct. Motion sequences can be used as references to improve the learning process and avoid policies generating unrealistic motion as done in the **deep mimic** approach[54]. While one deep mimic policy is learned specifically for a ball throwing task, it can be complicated to apply the solution to the task of pose prediction with real time constraints.

Reinforcement learning can however approach the problem differently when real time constraints are a concern. One such case is to use reinforcement learning to resolve the problem of finding optimal traversals of complex data structures derived from pose data. Early adoption of reinforcement learning is in relation with **motion fields** [44].

Motion fields are built on the concept of motion states assembled in graph structures. Motion states are discrete parameterizations of a motion sequence whereby movement from a frame to the next is described by the pose for the frame and its delta (finite difference) to following frame. By finding suitable transitions matching the current state of the character, a motion field can be traversed by integrating over joints delta movements associated with interpolation of one or multiple motion states. To find suitable traversals in the motion field, policies to find a suitable set of motion primitives to integrate are trained using reinforcement learning.

Motion fields can be classified as a motion state predictor and is able to generalize over existing data through interpolation of similar states. The limiting factor for application is the increased memory footprint, requiring a larger memory footprint for the motion field in comparison with the original motion data. This is a stark contrast to other methods based on encoding motion data to achieve similar objectives. Encoding time-series models for motion state prediction provide output at reduced,

not increased, memory footprints.

Motion graphs

Optimized traversals are not a new concept in regard to motion synthesis. **Motion graphs** as proposed in [40] is an earlier adoption to generating connected graphs. The major difference to motion fields is the use of motion sequences as nodes while edges in the graph represent near-optimal transitions between two frames in-between connected sequences. Transitions between animation sequences are then possible by traversing edges encountered within a node, utilizing interpolation over short windows to switch between nodes. In similarity motion fields, longer animation sequences are formed through graph search algorithms. By finding paths through the graph with the purpose of reaching an intended goal, a sequence of reference clips and transitions are formed.

A generative approach extending the same concept is **motion graph++**[48]. Rather than define nodes using animation sequences, Gaussian mixture models are estimated over *motion primitives* after reducing dimensionality of the parameters using PCA. Motion primitives in this case represent a sequence containing a common movement within the motion such as a step with the left foot. By labeling and grouping animation sequences, primitives are formed over a group by aligning motion within the sequences over the time domain using dynamic time warping.

Gaussian mixture models estimated over matching sequences then act as prior distributions from which a random motion sequence can be generated. Using associations between motion primitives, a graph structure is formed by connecting the Gaussian mixture model priors. Utilizing Gaussian processes to model transitions between nodes, planning algorithms can be used to optimize the generated motion regarding intended goal.

Motion graphs provide a solution to a larger problem relative to the task of learning a single motion. In a sense, the goal of a phase functioned neural network as applied to the motion synthesis task is to identify similar sequences within the motion and separate them in primitives. Definition of a phase in the time domain then aligns the joint motion within each primitive. Small motion variations for a frame can then be encoded and used as a prior when synthesizing motion using the features to describe the intended motion. The graph structure is then not formally stated, but rather inferred as a forward directed acyclic graph over motion primitives identified within the motion.

Motion matching

To further delve into the concept of using a database of animation sequences to synthesize motion, a common technique is referred to as **motion matching** [16]. In similarity with the original motion graph approach, motion matching could be defined as an animation controller rather than a method for synthesizing motion. Motion matching operate by finding an animation clip that matches the current objective of the character. In similarity with motion graphs, it takes the approach of finding a sequence of animations by utilizing a cost function to determine the cheapest transition from the current state. Finding the animation sequence best matched to

the character's predicted traversal path. Continuous optimization of best matching animation clip and by fitting matched clips to the traversal path, preferable results can be achieved.

Primary benefit of motion matching is a flexible approach that retain expressiveness within the motion as it relies on using the original motion data. Combined with computation cost and ease of use, the method provides a solid ground for driving animation systems in a real time context. The downside then, as shared with motion graphs, is scaling to large sets of motion sequences and inability to generalize over motion data. However, in comparison with motion graphs, it does not suffer from lack of edges connecting sequences. Avoiding problems with inaccuracies by allowing foot sliding and unrealistic motion sequences to occur when no better solution is found.

Problem formulation

Discussed methods provide opportunity to generate motion sequences be it for a single goal-oriented motion or a sequence of motions reaching a specific goal. There are however a few concerns that remain as evident in existing work. One of the overreaching concerns is the ability of methods to reproduce reference data (or ground truth) without loss in expression, a question that could be asked of all generative models. While methods like motion matching utilize reference data to define the motion states and are unaffected by the reproduction concern. Motion matching face problems concerning generalization and memory footprints.

Another issue with presented methods is in regard to the sparse comparison between different methods. Lacking a standardized method for comparing generated synthesis models, analyzing existing methods is in many cases impossible. The lack of comparisons complicates further research by limiting the options for determining suitable methods for a given task.

The purpose for this thesis is then to apply discussed concepts for the purpose of encoding specific objective-oriented motion sequences, a more versatile subject in comparison to the common task of learning locomotion synthesizers. Application of existing methods to new learning tasks provide new opportunities for evaluating the method while testing the method's ability to generalize over new tasks. The contribution is then in evaluation of existing methods which can be used to compare results from different methods. While a large case study would be required, the implementation complexity is out of scope of this thesis. As such, the thesis is limited to evaluating different methods as based on the same approach.

Chapter 3

Theory

Following sections are intended to provide a background and explain the concepts on which the thesis is built upon. Introducing algorithms and ideas for which a reader may or may not be familiar with and try to explain the concepts in such a way that they can be understood in the context of the thesis. With multiple problems faced for the thesis, a wide number of subjects are presented. With subjects ordered in a way to allow each subject to be understood, not all subjects are relevant for understanding all aspects of the thesis.

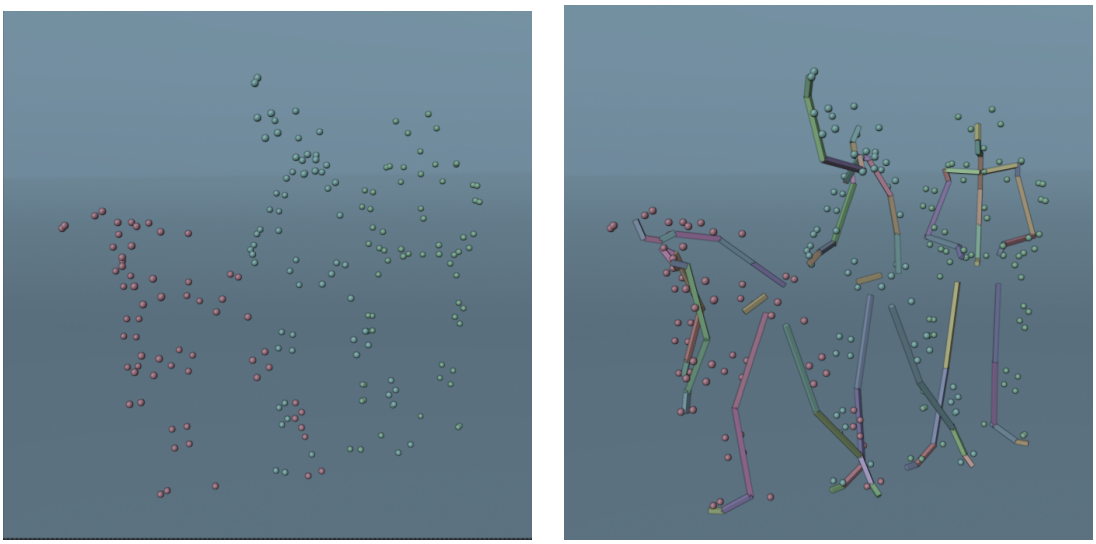
3.1 Optical motion tracking

Optical motion tracker data obtained using non-invasive surface markers and recorded through stereophotogrammetry is the gold standard for human movement analysis [1]. Optical motion capture systems track optical markers within the capture space of the system using multiple synchronized cameras distributed around the capture space. The notion of the *capture space* then refers to the limited volume in which an optical marker can be identified and triangulated using the images from 2 or more cameras in the system.

Utilizing specific marker configurations placed on different body segments, human motion can be represented within the capture space. By sampling and recording position coordinates of the optical markers, positions and orientations for the body segments can be reconstructed. Motion is then formed from a time-series, or sequence, of recorded data frames each on the form $N \times 3$, where N is the number of optical markers sampled within a frame. A representation from a subset of data frames within a sequence is displayed in *fig. 3.1*.

3.2 Segmental kinematics

Segmental kinematics refers to the problem of reconstructing spatial information of a body from kinematic information [12]. Specifically, it involves spatial reconstruction of rigid body segments from optical motion capture data. Given a time series of recorded coordinate vectors within some space, representing markers placed on surfaces of segments, skeletal pose animations can be reconstructed. The problem is then, generally, posed as finding the frame of reference for each rigid segment at each sampled time instance in the motion sequence.



(a) Optical motion trackers (b) Reconstructed skeleton

Figure 3.1: Optical markers as captured by a motion capture system. Each image contains three separate frames from the same sequence. In the left image, only the optical markers are included. The purpose for musculoskeletal reconstruction is then to find the skeletal pose for each frame as can be seen in the right image.

To mathematically define a segment's frame of reference, affine transformations represented by an orientation and translation relative to the global reference frame is used[1]. A segments frame of reference can namely be expressed as the equation

$$\boldsymbol{x}_g = \boldsymbol{R}\boldsymbol{x}_l + \boldsymbol{v} \tag{3.1}$$

where $\boldsymbol{R}$ is the 3×3 rotation matrix representing the segment orientation. The column vector $\boldsymbol{v}$ then define the segment's translation. The linear transformation takes a coordinate vector $\boldsymbol{x}_l$, defined in some local orthonormal basis, and defines it as the point $\boldsymbol{x}_g$ in the global reference frame.

Reconstruction of a segment is then achieved by finding a transformation from the local basis in which the morphological description is defined, to the segment's global reference frame. Extending the definition to an ensemble of multiple musculoskeletal segments, a hierarchy is formed from relationships between connected segments. Connections between segments in the hierarchy are then defined by a joint. A musculoskeletal model consisting of multiple segments and associated joints is referred to as a multi-link chain model [45][23].

So far, considerations are made regarding how a model of the musculoskeletal system can be formulated. While the exact solution can be found for *Eq. 3.1* when the musculoskeletal segment is rigid. Assumption of rigidity can only be made when considering the underlying skeletal structure, and assuming rigidity is a simplification when considering the entire musculoskeletal segment.

[1]The global reference frame in this case is the identity of the capture space, defined by of the motion capture system

Since rigidity in the underlying bone structure is not reflected in rigid skin motion due to movement in muscles and other soft-tissue, soft-tissue-artifacts (or STA) are introduced as the surface (skin) deforms during movement [43]. Consequently, methods for reconstructing the skeletal structure rely on approximation, minimizing the error between the segment description and measured optical data. Even so, the formulation of rigid segments is convenient as it reduces complexity when measuring and modeling the underlying structure from optical markers.

3.2.1 Local reference frames

In the process of fitting parameterized segment models to recorded marker coordinates, transformations between multiple different reference frames are necessary. The equation for an affine transform within a nested reference frame can be expressed as

$$\begin{aligned} \boldsymbol{x}_g &= \boldsymbol{R}_p(\boldsymbol{R}_l\boldsymbol{x}_l + \boldsymbol{v}_l) + \boldsymbol{v}_p \\ &= \boldsymbol{R}_p\boldsymbol{R}_l\boldsymbol{x}_l + \boldsymbol{R}_p\boldsymbol{v}_l + \boldsymbol{v}_p \\ &= \boldsymbol{R}_c\boldsymbol{x}_l + \boldsymbol{v}_c \end{aligned} \tag{3.2}$$

where

$$\begin{aligned} \boldsymbol{R}_c &= \boldsymbol{R}_p\boldsymbol{R}_l \\ \boldsymbol{v}_c &= \boldsymbol{R}_p\boldsymbol{v}_l + \boldsymbol{v}_p \end{aligned} \tag{3.3}$$

The equation takes a vector $\boldsymbol{x}_l$ within the local reference frame defined by the rotation $\boldsymbol{R}_l$ and translation $\boldsymbol{v}_l$, and moves it through the parent frame to its position in the global frame of reference $\boldsymbol{x}_g$. The combined affine transform can then be calculated as the rotation $\boldsymbol{R}_c$ and translation $\boldsymbol{v}_c$, simplifying the problem on the same form as in *Eq. 3.1*.

Calculation of a local frame of reference can easily done if the composite transform is known in the global space, or if the local reference frame is calculated within the parent frame itself. The equation to determine the frame of reference local to another is done by identifying that the transpose of the orthonormal rotation matrix is equal to its inverse $\boldsymbol{R}^T = \boldsymbol{R}^{-1}$, and previous equations can be solved for $\boldsymbol{R}_l$ and $\boldsymbol{v}_l$ as

$$\begin{aligned} \boldsymbol{R}_l &= \boldsymbol{R}_p^T\boldsymbol{R}_c \\ \boldsymbol{v}_l &= \boldsymbol{R}_p^T(\boldsymbol{v}_c - \boldsymbol{v}_p) \end{aligned} \tag{3.4}$$

Homogeneous transformation

For a simpler mathematical representation of the concept of affine transformations, it can be useful to express the rigid body transform and reference frame equation using homogeneous coordinates. Expanding the problem into the fourth dimension, affine transformation can be expressed as a 4×4 matrix defined in *Eq. 3.5*. The orientation defined by the rotation matrix $\boldsymbol{R}$ is here used to define the expanded matrix components of the affine transformation $\boldsymbol{T}$.

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{v} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{e}_{0,x} & \boldsymbol{e}_{1,x} & \boldsymbol{e}_{2,x} & \boldsymbol{v}_x \\ \boldsymbol{e}_{0,y} & \boldsymbol{e}_{1,y} & \boldsymbol{e}_{2,y} & \boldsymbol{v}_y \\ \boldsymbol{e}_{0,z} & \boldsymbol{e}_{1,z} & \boldsymbol{e}_{2,z} & \boldsymbol{v}_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$
$$\begin{pmatrix} \boldsymbol{x}_g \\ 1 \end{pmatrix} = \boldsymbol{T} \begin{pmatrix} \boldsymbol{x}_l \\ 1 \end{pmatrix}$$

Ignoring the unchanging homogeneous component of 1 appended in the fourth dimension of the transformed coordinate vector $\boldsymbol{x}_g$, the equation is identical to the equation as defined in [*section.3.2*]. Concepts of local reference frames are then easier to formulate. Namely, the composite transform *Eq. 3.3*, can be expressed simply as the product of the affine transformation matrices $\boldsymbol{T}_c = \boldsymbol{T}_p \boldsymbol{T}_l$, avoiding a second variable to represent translation.

3.2.2 Reference frame terminology

Since data is constantly transformed between different reference frames, terminology for describing different frames of reference are in accordance with [11], *technical* and *anatomical frames*. Technical frames refers to reference frames that can be considered as arbitrary, and varies depending on the methods or parameters from which they are defined. Two types of technical frames are of interest[12]:

- *Morphological technical frame*, reference frame in which a segment's morphological description is defined, arbitrary since any of the segment's reference frames could be used.
- *Marker cluster technical frames* are defined from the marker clusters associated with the musculoskeletal segments. Since methods and marker placement vary, they have no fixed representation.

Anatomical reference frames are differentiated from technical frames due to being standardized and defined from anatomical landmarks (palpable bone protrusions). The primary purpose for their definition is repeatability and reproducibility. Anatomical landmarks ensures, to a degree, that a specific reference frame can be reconstructed even if other parameters change[11].

Similarly, rotational degrees of freedom (DoF) are defined using the same principles as anatomical reference frames. Rotational DoF is however defined in the form of rotational axis rather than orthonormal basis (e.g. axis in which a segment's flexion and extension is measured). Rotation axis are therefore often defined in relation with the segment's anatomical frame [65][66].

It should be noted that the purpose of defining rotational DoF using separate rotational axis is primarily for the purpose of standardizing measurements and allowing data to be compared across different studies over human motion. However, it also has relevance in minimizing effects of gimbal lock caused by representing rotations using Euler angles. While gimbal lock does not arise when using matrix representations, utilization of Euler angle representations is relevant during reconstruction as shown in later sections.

3.2.3 Anatomical calibration and parameter estimation

Acquisition of anatomical data from a multitude of sources makes calibration a necessity to avoid introducing significant errors during measurements. Regarding reconstruction of existing data sets, anatomical calibration refers to the process of fitting musculoskeletal models to the subject's anatomy.

For spatial reconstruction to be possible: anatomical reference frames, marker cluster frames, and morphological frames need to be defined and calibrated. Calibration ensures that uniform morphological descriptions (including musculoskeletal segment models) can be constructed.

Optimally, medical equipment should be used to measure parameters necessary for model calibration, but this is rarely possible in practice [11]. Instead, parameters are acquired primarily using three methods:

- Body measurements, measuring distances such as the length of body segments.
- Anatomical markers, utilizing optical markers for identifying and recording the position of anatomical landmarks.
- Motion trials, movement patterns used to estimate joint centers [23].

Estimation of internal landmarks

Internal anatomical landmarks cannot be accurately determined using non-invasive surface measurements. Considering the problem of reconstructing the musculoskeletal transform at every frame, internal landmarks such as center of rotation (CoR) need to be determined for all musculoskeletal segment joints. Techniques for estimating locations of internal landmarks then become a necessity. Avoiding the need for expensive and invasive measurements by estimating parameters using data measured using non-invasive methods.

Generally, estimation techniques for internal landmarks can be divided in two groups: *predictive models* and *functional approaches*. *Predictive models* utilize empirical relationships between measurable anatomical landmarks to estimate internal landmarks [11]. *Functional approaches* on the other hand infer CoR from motion data using the assumption that there is a fixed pivot in the movement between two connected segments. Estimating CoR by minimizing an objective function using either iterative optimization or a least squares approach [19, 24, 14].

When determining the appropriate method for a specific task, predictive models are standardized by the International Society of Biomechanics (ISB). Application of functional methods then depend on available predictive models and ISB recommendations. Suitability of different functional methods also depend on the musculoskeletal segment for which CoR is estimated, the type of joint used in the model (e.g. universal, hinge, or spherical joint), and if joint translations are modeled. Generally, minor translations occur in functional joints. Translational degrees of freedom are however rarely considered as available equipment lack the necessary precision for the small variations to be resolved [15].

To take a concrete example of estimating an internal landmark such as Hip-Joint-Centre (HJC) which represents the center of rotation for the joint connecting the

pelvis with the femur. Functional methods have shown to reduce errors in regard to predictive models [9]. Determining the appropriate functional method is on the other hand complicated. Existing comparisons in accuracy between different methods and their ability to estimate HJC has been partially in-conclusive with varying results between experiments.

Even so, functional methods, and in particular geometrical-sphere-fit (GSF)[19], has shown to be effective at estimating HJC [61, 35]. In the large scale review GSF methods are also recommended as they "should be used in people with sufficient active hip range of motion" [35]. It is also noted that functional methods are affected by marker placement. Marker placement impact can however be reduced if markers are placed on surfaces with minimal skin deformation and by ensuring marker placements are distanced from the CoR. Surfaces with major skin deformation then refer to surfaces associated with e.g. muscle bellies which should be avoided.

3.2.4 Musculoskeletal segment models

A musculoskeletal segment model is defined by a technical or anatomical reference frame centered at the pivot point (CoR) around which the segment rotates. The reference frame is determined within the capture space, or within a secondary frame of reference that can be defined as the model space. Within this morphological reference frame, local frames of reference can be calculated and parameterized. Associating different reference frames with a single morphological reference frame allow different representations of the model to be reconstructed as show in *fig. 3.2*.

Model of the underlying musculoskeletal structure is important, and the crucial part is the association between the musculoskeletal segment and the marker cluster from which the segments reference frame are estimated during the reconstruction process. In general, each musculoskeletal segment model includes a marker model containing optical trackers associated with the segments. Markers within a marker model can then be modeled within the morphological reference frame of the segment as seen in *fig. 3.3*.

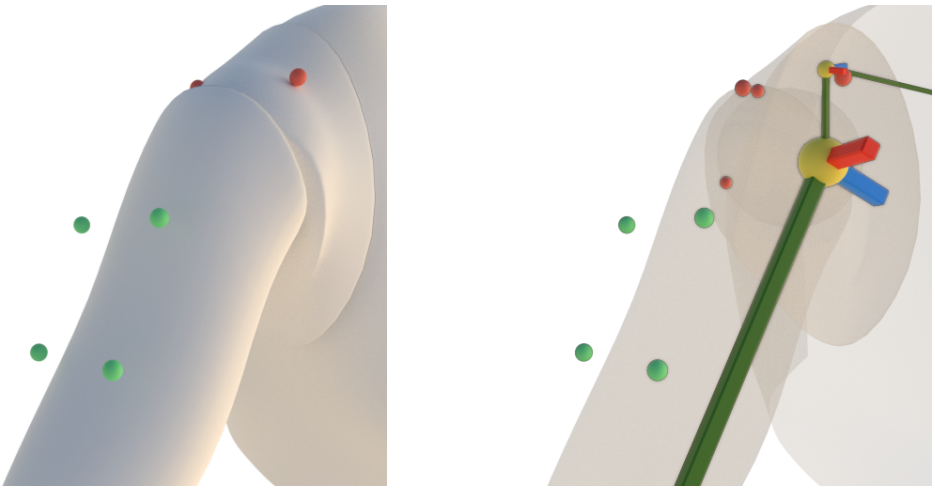
3.2.5 Calibrated anatomical system technique

Before further concepts are introduced it can be useful to provide examples of how previous concepts can be applied in practice. In particular, many terms are formulated in regard to an experimental procedure referred to as "CAST" [11][12]. The procedure explains the necessary steps involved in the reconstruction of reference frames from marker positions recorded using stereophotogrammetry. The procedure can be summarized with the following:

1. Placement of optical markers on recorded subjects using guidelines determined in previously published research.

2. Define necessary technical reference frames using subsets of the recorded marker clusters.

3. Using the technical reference frames in combination with recording static trials, or other means, to identify positions of anatomical landmarks within these



(a) Opaque surface (b) Translucent surface

Figure 3.2: Musculoskeletal model of the right upper arm and scapula depicted in the capture space of the stereophotogrammetry equipment. On the left, skin approximation is rendered as a white surface, and on the right, a translucent representation is used to highlight the model beneath and behind the skin. Pivot points (CoR) for the segments are represented by yellow spheres with colored rectangles used to depict the axis for the reference frames. To highlight the musculoskeletal marker models, light green spheres are positioned on markers associated with the upper arm, and light red spheres illustrate the positions of scapula markers.

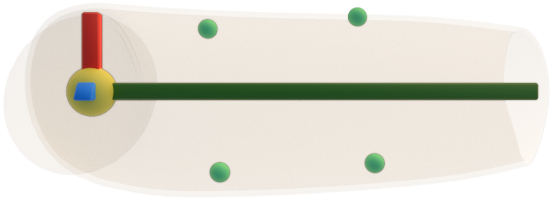


Figure 3.3: Musculoskeletal segment model associated with the right upper arm rendered relative the morphological reference frame of the segment. The model is oriented with the camera facing the positive Z axis. The segment model includes the axis definition (colored boxes), CoR or pivot point (yellow sphere), marker cluster model (light green spheres), and skin approximation (translucent surface).

reference frames. And as inferred from the reference frame equation, identify a mapping of the parameterization to the capture space itself.

4. Once relationships between technical frames and anatomical landmarks have been established, technical reference frames can be estimated at each time instance within recorded time-series data for which these relationships apply.

5. Lastly, anatomical reference frames can be calculated relative to the estimated technical frames, and the pose of the musculoskeletal segment can be reconstructed.

While the fourth of and fifth step of CAST could be rephrased when regarding recent reconstruction methods, the procedure for creating a parameterized model within an anatomical pose (or reference pose) is still relevant. Models defined for reconstruction purposes need to be both fully parameterized in relation to all modeled musculoskeletal segments. Calibration also needs to be performed when regarding the capture space in which the optical data is recorded. The last point refers to the fact that the rigid model constructed for a subject do not change, parameters such as marker placements can however change between recording sessions.

3.3 Inverse kinematics

Reconstructing the orientations and locations of musculoskeletal segments from a given set of optical trackers can be summarized as an inverse kinematics (IK)[2] problem [20]. For a limb (or kinematic chain) the problem then lies in solving the kinematic equation of the rigid bodies from the view of given end effector(s) reaching a desired position and orientation. In the case of reconstructing optical motion tracker data, the end effectors consist of optical markers. The solution is then the joint configuration where recorded markers are matched by the configuration of the modeled segments with some minimal residual r. The configuration state $\boldsymbol{s}$ for the chain model can therefore be expressed as a function of its parameters using the forward kinematics equation

$$\boldsymbol{s} = g(\boldsymbol{q}) \tag{3.6}$$

with the generalized coordinates $\boldsymbol{q}$ defining the chain model parameters. Given some partial state $\boldsymbol{e} \in \boldsymbol{s}$, an inverse kinematics problem can be expressed as the equation

$$\boldsymbol{q} = g^{-1}(\boldsymbol{e}) \tag{3.7}$$

Solution for the IK problem is then subject to the constraints defined in the chain model and the given state $\boldsymbol{e}$ [5]. In the general case, IK problems are underconstrained and multiple solutions exist satisfying the constraints. For the same reason, multiple algorithms exist for finding a suitable solution and a comprehensive list can be found in [5]. If the modeled system (or sub-system) involved is small, the simplest solution is often to apply basic trigonometry and the cosine rule.

[2]Forward kinematics, as the inverse of inverse kinematics, is the system configuration as defined by a given set of joint parameters.

3.3.1 Trigonometry and cosine rule

Trigonometric formulas can be used for solving simple IK problems. Examples of simple IK problems can be to adjust arms and legs to the environment using animated end effectors (see e.g. [49]). Depending on the approach used for musculoskeletal reconstruction, trigonometry could be applied as a complement with other methods when generating initial estimates.

Applications of the cosine rule primarily involve adjustments to flexion and extension of the knee and elbow joints due to their function as paired hinge joints. By adjusting flexion and extension followed by twisting motion along the longitudinal axis[3] of the forearm and lower leg, reach of the limb can be corrected. The cosine rule can then be applied to solving a minimal problem in regard to the limb orientation since the knee and elbow joints exhibiting minimal rotation around the remaining axis [3].

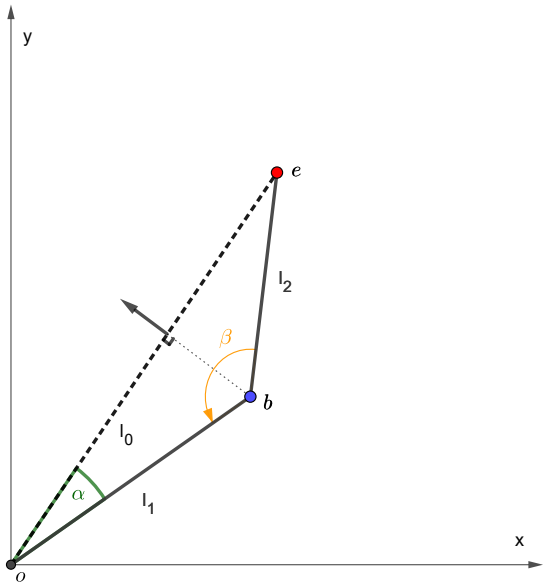


Figure 3.4: Application of cosine rule for solving a 2-dimensional IK problem. A representation for the problem could be an arm in the frontal plane flexed at an angle β.

A concrete 2-dimensional example of the application is provided in *fig. 3.4*, to reach the designated end effector target $\boldsymbol{e}$, the angle α is calculated using the cosine rule. Once determined, the coordinate for $\boldsymbol{b}$ can be calculated using *Eq. 3.8*, by rotating the norm $\boldsymbol{u}$ 90° and utilizing the sides of the right-angled triangle associated with l_1. If there are no defined joint limits, a second solution exist for $\boldsymbol{b}$ on the opposite side of $\boldsymbol{u}$.

While flexible, trigonometric formulas cannot solve the full reconstruction problem. Instead, following sections will provide examples of other approaches better

[3]Longitudinal axis is the axis along the length of, or proximal-distal direction of the limb.

suited to minimizing the residual arising due to soft tissue artifacts (STA) and discrepancies between model determined and sampled marker frames.

$$\begin{aligned} \alpha &= \cos^{-1}\left(\frac{l_1^2 + \boldsymbol{e}_x^2 + \boldsymbol{e}_y^2 - l_2^2}{2l_1\|\boldsymbol{e}\|}\right) \\ \boldsymbol{u} &= \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \\ l_0 &= l_1\cos(\alpha) \\ \boldsymbol{b} &= l_0\boldsymbol{u} - \sqrt{l_0^2 + l_1^2}\begin{pmatrix} -\boldsymbol{u}_y \\ \boldsymbol{u}_x \end{pmatrix} \end{aligned} \tag{3.8}$$

3.4 Single body optimization

Single-body-optimization (SBO) approach the musculoskeletal reconstruction problem by solving the rigid body transform in *Eq. 3.1* for each segment individually. Reconstruction of the skeletal pose is then achieved by finding the best fit transform between the instantaneous position of recorded markers associated with the musculoskeletal segment, and modeled marker positions in the morphological reference frame of the segment.

While other techniques have proven to yield better estimates for musculoskeletal reconstruction, techniques for finding the best fitting transform for a single segment have multiple uses outside reconstructing the final pose. For example, most joint center estimation techniques require a coherent reference frame to be calculated for one of the segments.

3.4.1 Gram-Schmidt procedure

Gram-Schmidt orthonormalization is a simple solution that is often applied to determining technical reference frames from a set of at least three markers. Using one of the markers to define a reference point, two separate non-parallel markers form a 2-dimensional subspace (a plane) in relation to the reference point, leaving the plane normal to define the third axis. Application of Gram-Schmidt orthogonalization then allow two orthonormal axis to be defined on the plane, letting the third axis, $\boldsymbol{e}_2$, be calculated using the cross product as seen in *Eq. 3.9*.

$$\begin{aligned} \boldsymbol{e}_0 &= \frac{\boldsymbol{p}_1 - \boldsymbol{p}_0}{\|\boldsymbol{p}_1 - \boldsymbol{p}_0\|} \\ \boldsymbol{e}_1 &= \frac{\boldsymbol{p}_2 - \boldsymbol{p}_0 - \langle \boldsymbol{e}_0, \boldsymbol{p}_2 - \boldsymbol{p}_0\rangle \boldsymbol{e}_0}{\|\boldsymbol{p}_2 - \boldsymbol{p}_0 - \langle \boldsymbol{e}_0, \boldsymbol{p}_2 - \boldsymbol{p}_0\rangle \boldsymbol{e}_0\|} \\ \boldsymbol{e}_2 &= \boldsymbol{e}_0 \times \boldsymbol{e}_1 \end{aligned} \tag{3.9}$$

Here $\boldsymbol{p}_i$ defines the position for the i:th marker and the vectors $\boldsymbol{e}_i$ form an orthonormal basis which can be written on the form of a 3×3 matrix

$$\boldsymbol{R} = (\,\boldsymbol{e}_0 \;\; \boldsymbol{e}_1 \;\; \boldsymbol{e}_2\,) = \begin{pmatrix} \boldsymbol{e}_{0,x} & \boldsymbol{e}_{1,x} & \boldsymbol{e}_{2,x} \\ \boldsymbol{e}_{0,y} & \boldsymbol{e}_{1,y} & \boldsymbol{e}_{2,y} \\ \boldsymbol{e}_{0,z} & \boldsymbol{e}_{1,z} & \boldsymbol{e}_{2,z} \end{pmatrix} \tag{3.10}$$

Once the orientation $\boldsymbol{R}$ is determined, a reference frame is formed from the position of the reference marker as the center point. Less useful compared to method minimizing the error when estimating the reference frame, Gram-Schmidt orthogonalization is commonly used to define anatomical reference frames due to simplicity and requiring fewer parameters [65][66]. While the method is susceptible to measurement errors or deformations (e.g. STA), this is less relevant when only anatomical reference frames are considered due to higher precision in anatomical landmark measurements.

3.4.2 Single-value-decomposition

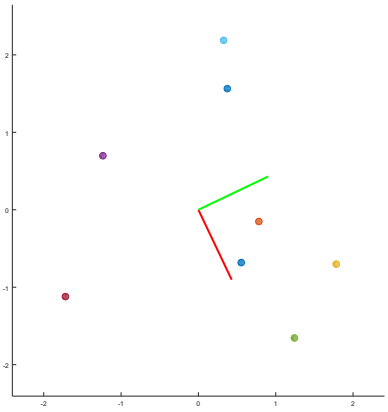
Another method for finding the affine transformation from a rigid marker cluster $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_n\}$, to sampled coordinates $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, .., \boldsymbol{y}_n\}$ is achieved using Singular-Value-Decomposition (SVD). In comparison to Gram-Schmidt orthonormalization, SVD can be used to find the best fit transform minimizing the sum of squared distances between sampled markers $\boldsymbol{y}_i$ and the N markers under transformation $\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{v}$ [13] as given by *Eq. 3.11*.

$$\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{v} - \boldsymbol{y}_i\|^2 \tag{3.11}$$

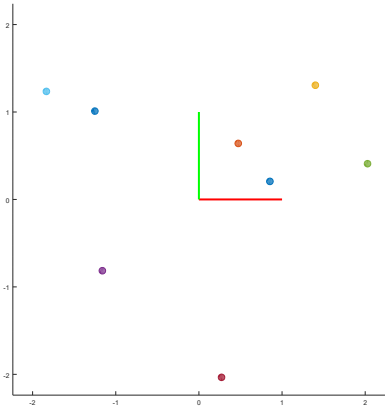
To give a context to how *Eq. 3.11* is minimized, SVD is a factorization of a matrix into 3 components, $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are sets of orthonormal vectors (with each set forming an orthonormal basis), and $\boldsymbol{S}$ being a diagonal matrix containing the singular values of the decomposition. Using a simple geometrical interpretation for the decomposition, a matrix of a marker cluster centered at the mean, $\boldsymbol{X} - \bar{\boldsymbol{X}} = \boldsymbol{U}_x\boldsymbol{S}_x\boldsymbol{V}_x^T$; gives that $\boldsymbol{U}_x$ represents the permuted orientation $\boldsymbol{O}_x$ of the largest varying subspaces associated with the singular values in $\boldsymbol{X} - \bar{\boldsymbol{X}}$. The diagonal matrix $\boldsymbol{S}_x$ then define a scaling, on form 3×3, along each of the orthogonal subspaces in decaying order. Finally, $\boldsymbol{V}_x^T$ holds the orthonormalized representation, on form $3 \times N$, of the N markers in the cluster as defined by the combined transformation $\boldsymbol{U}_x\boldsymbol{S}_x$. Geometrical representation for the decomposition of a point cluster in $\mathbb{R}^2$ is highlighted in *fig. 3.5*. Finding the affine transformation between two marker clusters $\boldsymbol{X}$ and $\boldsymbol{Y}$ then involve finding the rotation around the mean of the clusters and can be derived from *Eq. 3.12*

$$\boldsymbol{C} = (\boldsymbol{Y} - \bar{\boldsymbol{Y}})(\boldsymbol{X} - \bar{\boldsymbol{X}})^T = (\boldsymbol{U}_y\boldsymbol{S}_y\boldsymbol{V}_y^T)(\boldsymbol{U}_x\boldsymbol{S}_x\boldsymbol{V}_x^T)^T = \boldsymbol{U}_y\boldsymbol{S}_y\boldsymbol{V}_y^T\boldsymbol{V}_x\boldsymbol{S}_x\boldsymbol{U}_x^T \tag{3.12}$$

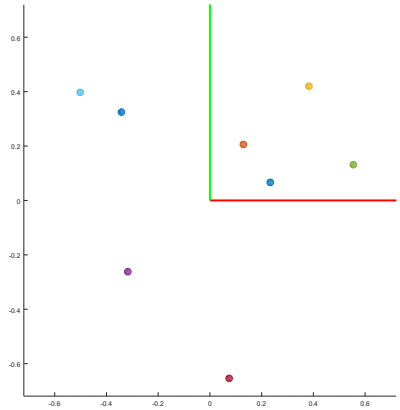
Here, $\boldsymbol{C}$ relate to the cross-covariance matrix between the two coordinate clusters where $cov(X, Y) = \frac{1}{n}\boldsymbol{C}$. While there are no exact solution for finding the exact rotation when marker clusters are deformed, under the rigid body assumption for which no deformation occur, only the orientation of the cluster $\boldsymbol{O}$ is varied under rotation. Utilizing the rigidity assumption, following relationships are inferred namely:



(a) $\boldsymbol{USV}^T$



(b) $\boldsymbol{SV}^T$



(c) $\boldsymbol{V}^T$

Figure 3.5: Geometrical interpretation of singular value decomposition (SVD) for a cluster of points in $\mathbb{R}^2$. In the top image the original cluster representation is presented along with the axis associated with the largest varying subspace (red axis). In the bottom-left, the cluster is oriented and aligned in accordance with the basis $\boldsymbol{U}$ defined by the singular values, but scaling of the singular values are retained. In the bottom-right image, the orthonormal representation of the cluster is shown where the sum of squared values associated with the x or y axis is equal to one.

The important notion is the realization that the representations in the bottom images remains the same when the cluster is rotated, as rotation only change the orientation of the largest varying axis while the other cluster representations remain the same. It should be noted however that the decomposition only defines the orthogonal subspaces with greatest variation, it does not define a direction within the subspace. That means that reflections over the green or red axis to the left would only be reflected in the following representations to the right, both are defined by the algorithm for finding the decomposition rather than by the decomposition itself.

$\boldsymbol{V}_y\boldsymbol{V}_x = \boldsymbol{J}_y\boldsymbol{J}_x$ and $\boldsymbol{S}_y = \boldsymbol{S}_x$. Here $\boldsymbol{J}_y\boldsymbol{J}_x$ represent diagonal reflection matrices, reflection occur due to the matrices for the decomposition is not unique even if the

decomposition has an unique solution. In particular, a decomposition could be written as 5 components represented by $\boldsymbol{U}_5\boldsymbol{JSJV}_5^T$. Since inversion over an axis in both of the orthonormal basis is equivalent, simplification as $\boldsymbol{JSJ} = \boldsymbol{JJS} = \boldsymbol{S}$ is possible due to the diagonality of the matrices involved. Therefor the formulation is equivalent to the earlier description but with the caveat that each matrix component has an unique solution except for $\boldsymbol{J}$. Finally, *Eq. 3.12* can now be simplified using the expanded definition in accordance with *Eq. 3.13*

$$\boldsymbol{C} = \boldsymbol{U}_y\boldsymbol{S}_y\boldsymbol{V}_y^T\boldsymbol{V}_x\boldsymbol{S}_x\boldsymbol{U}_x^T = \boldsymbol{U}_y\boldsymbol{J}_y\boldsymbol{S}^2\boldsymbol{J}_x\boldsymbol{U}_x^T = \boldsymbol{O}_y\boldsymbol{S}^2\boldsymbol{O}_x^T \tag{3.13}$$

Using the geometrical interpretation of the decomposition as indicated in *Eq. 3.13*, remaining variables represent two rotations where the decomposition reflection has been canceled out, and the scaling $\boldsymbol{S}^2$ along the dominant subspaces of the cluster. Simplification of the rotations is possible due to the previously mentioned relationships of the decomposition in three components, where $\boldsymbol{UJ}$ is equivalent to $\boldsymbol{U}_5\boldsymbol{JJ} = \boldsymbol{U}_5 = \boldsymbol{O}$. Expression derived in *Eq. 3.13* then prove that the decomposition $\boldsymbol{C} = \boldsymbol{O}_y\boldsymbol{S}^2\boldsymbol{O}_x^T$ is equal to the full expression, avoiding multiple uses of SVD as the decomposition has an unique solution satisfied by the right hand side of the equation.

Finally, in accordance with the geometrical interpretation, only the orthonormal basis are relevant for the cluster orientations, and rotation $\boldsymbol{R}$ and translation $\boldsymbol{v}$ for the affine transformation aligning marker cluster $\boldsymbol{X}$ with $\boldsymbol{Y}$ (*Eq. 3.11*) can be calculated in *Eq. 3.14*. Further proof for the relationships discussed, and why the rotation found from the decomposition of $\boldsymbol{C}$ minimizes *Eq. 3.11* in the general case, is available in [13].

$$\begin{aligned} \boldsymbol{R} &= \boldsymbol{U}_y\boldsymbol{J}_y\boldsymbol{J}_x\boldsymbol{U}_x^T = \boldsymbol{O}_y\boldsymbol{O}_x^T \\ \boldsymbol{v} &= \bar{\boldsymbol{y}} - \boldsymbol{R}\bar{\boldsymbol{x}} \end{aligned} \tag{3.14}$$

In certain cases it's important to note that the equation for the rotation in *Eq. 3.14* can be ill-conditioned. When one of the singular values in *Eq. 3.13* approaches zero, the rotation $\boldsymbol{R}$ can represent an arbitrary rotation and/or reflection in regard to the subspace associated with the singular value. This occur if all markers in the cluster are represented within a lower dimensional subspace of the full three dimensional space $\mathbb{R}^3$. With the singular value arbitrarily close to zero, the subspace can be regarded as a null-space for the cluster with an undetermined orientation.

Considering only the 2 dimensional problem for which the problem has a simple solution, with markers defined within a plane in $\mathbb{R}^3$. Singular values associated with the subspace $\mathbb{R}^2$ are then dominant in the decomposition, and reflections occur along the decompositions third axis for which the singular value is near zero. To identify if reflection occurs, the determinant of the final rotation from *Eq. 3.14* can be used, and reflection can be accounted for by ensuring that the rotation matrix is right-handed in accordance with *Eq. 3.15*.

$$\boldsymbol{R} = \boldsymbol{O}_y \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(\boldsymbol{O}_y\boldsymbol{O}_x^T) \end{pmatrix} \boldsymbol{O}_x^T \tag{3.15}$$

3.5 Multibody kinematics optimization

Multibody kinematics optimization (MKO) represent a set of algorithms designed for approximating joint poses from sequences of optical tracking data, and have proven to be an effective way to reduce error introduced by STA [7]. MKO algorithms operate on the idea of optimizing the fit of a kinematic chain model by letting adjacent segments influence each other. Incorporating relations between rigid body segments to constrain the reconstructed pose in each frame. By allowing neighboring segments to influence and constrain each other, MKO has shown to improve results in comparison with approaches reliant on single-body-optimization [45].

MKO algorithms require a set of parameters that need to be determined before the pose of the musculoskeletal segments can be estimated from optical data. First and foremost, the chain model with accompanied constraints need to be determined from human anatomy. Multiple musculoskeletal models are available in literature for different sets of body segments. Surveys for models on upper limbs (shoulder, upper arm and forearm) [18], lower limbs [42], and applications for whole body models [23] are available.

The purpose for different models varies as they balance between anatomical biofidelity, simplicity and complications associated with determining model parameters. However, for purposes where detail and biofidelity in reconstructed joint movement is less important, spherical joints provide a good alternative to more complex models [42].

In particular, constraints implied with the use of a specific joint in the model primarily affect orientations of neighboring (proximal) joints and has less influence on (distal) joints further away in the hierarchy [17]. Furthermore, joint types also have minimal impact in the estimated flexion and extension of a joint, but do influence reconstructed segment orientations [25]. It should be noted however, that these results are primarily associated with research in lower limbs and may not translate to other segments. Simpler models do on the other hand provide comparable results to complex models and is highly relevant when coherency between the chain model and the musculoskeletal structure has less relevance.

3.5.1 Global optimization method

Global optimization method, also referred to as GOM or just GO, is one of the most used methods in biomechanics[23]. In contrast to single-body-optimization based on SVD (discussed in [*section.3.4.2*]), the problem is no longer limited to finding the affine transform of a single segment. GOM is a MKO approach and generalizes the problem formulation by expressing the pose of all n musculoskeletal segments in a multi-link chain model as a function of the generalized coordinate vector $\boldsymbol{q} \in \mathbb{R}^k$. Expressing the chain model as a system with k degrees of freedom by defining an objective function $f(\boldsymbol{q})$ for the system. A best fitting pose, as considered by the objective function, can be determined in a least square sense by solving the following non-linear optimization problem [28][45]

$$\min_{\boldsymbol{q}\in\mathbb{R}^k} \quad \|f(\boldsymbol{q})\|^2 \tag{3.16}$$

Definition of loss in previous equation is then equivalent to the problem formulated for SVD. In fact, when considering a system consisting of a single rigid body segment, the problem is linear, and the problem formulations are nearly identical. Since the system consist of multiple segments, the problem is no longer linear. The residual error associated with the function of a set of generalized coordinates $f(\boldsymbol{q})$ can however still be formulated as a vector containing differences between model determined and sampled marker coordinates. Residual or loss can therefore be expressed as in the following equation [45]

$$\begin{aligned} f(\boldsymbol{q}) &= (\,\boldsymbol{s}_1 \;\; \boldsymbol{s}_2 \;\; \dots \;\; \boldsymbol{s}_n\,) \\ \boldsymbol{s}_i &= (\,\boldsymbol{d}_0^T \;\; \boldsymbol{d}_1^T \;\; \dots \;\; \boldsymbol{d}_m^T\,) \\ \boldsymbol{d}_m &= \boldsymbol{R}_i\boldsymbol{m}_m + \boldsymbol{v}_i - \boldsymbol{y}_m \end{aligned} \tag{3.17}$$

Here, $\boldsymbol{y}_m$ is the coordinates of the m:th marker associated with the i:th segment $\boldsymbol{s}_i$ as sampled within the data frame. The vector $\boldsymbol{m}_m$ on the other hand refer to the coordinate of the m:th marker modeled within the segments morphological reference frame (for a visual representation see *fig. 3.3* in [*section.3.2.4*]).

In consideration to *Eq. 3.17*, the function for the system $f(\boldsymbol{q})$ is still poorly defined, segment rotations $\boldsymbol{R}_i$ and translations $\boldsymbol{v}_i$ are not known. Rather, transformations are determined within the objective function from the generalized coordinates $\boldsymbol{q}$. Since the generalized coordinates can be found by minimizing the error using an adequate optimization function available in a numerical programming package (e.g. [62]), the following section is designated for the description of evaluating the objective function $f(\boldsymbol{q})$.

3.5.2 Generalized coordinates

To formulate the musculoskeletal segment hierarchy as a system of generalized coordinates defined in the previous section. Rigid body transforms need to be expressed as a function of its independent parameters. As such, a segment's joint can be defined within a local reference frame of the parent using a reference pose defined by the chain model, expressing a joint's reference frame $\boldsymbol{T}_j$ as

$$\boldsymbol{T}_j = \boldsymbol{T}_p \boldsymbol{T}_m \boldsymbol{T}_r \tag{3.18}$$

Here, $\boldsymbol{T}_p$ defines the parent transform expressed in the same form in relativity to its own parent, while $\boldsymbol{T}_m$ refers to the model determined orientation of the joint in relativity to the parent frame. $\boldsymbol{T}_r$ is then the transform explicitly defined by a subset of the generalized coordinates, defining the segment orientation fitted by the optimization function in *Eq. 3.16*.

Within the nested reference frame $\boldsymbol{T}_m$, the segment's rigid body transform $\boldsymbol{T}_r$ is simplified to a rotation expressed using only Euler angles $(\alpha \;\; \beta \;\; \gamma)$. In cases where the joint's center of rotation is displaced during movement, a translation vector $\boldsymbol{v} = (x \;\; y \;\; z)$ is still necessary to express displacement occurring relative to the frame $\boldsymbol{T}_p \boldsymbol{T}_m$. Finally, generalized coordinates of a musculoskeletal system without joint displacement is a vector $\boldsymbol{q}$ on the form

$$\boldsymbol{q} = (\xi_0 \;\; \xi_1 \;\; .. \;\; \xi_n) = (x_0 \;\; y_0 \;\; z_0 \;\; \alpha_0 \;\; \beta_0 \;\; \gamma_0 \;\; \alpha_1 \;\; \beta_1 \;\; \gamma_1 \;\; \ldots \;\; \alpha_n \;\; \beta_n \;\; \gamma_n) \tag{3.19}$$

Here $\boldsymbol{q}$ expresses the i:th joints rotation as the Euler angles $(\alpha_i \;\; \beta_i \;\; \gamma_i)$, defining the rotation in relativity to the joint's modeled orientation in the parent reference frame. Positioning of the system itself within the capture space is then defined as a displacement vector $(x_0 \;\; y_0 \;\; z_0)$.

Minimization of the loss function in *Eq. 3.17* can now be applied to determine a set of transformations $\boldsymbol{T}_j$ by allowing the varying vector $\boldsymbol{q}$ to be optimized in relation with the loss differential, given some initial estimate for $\boldsymbol{q}$.

3.6 Neural networks

Topics and research regarding neural networks are extensive and the following sections can therefor only cover core concepts as applied for structuring the motion models. Relevant topics such as back propagation and stochastic gradient descent which is used for fitting network parameters can be found in other resources, e.g. [56]. For the first section, a brief introduction of the concept of neural networks covers feed-forward neural networks, followed by the definition of phase-functioned neural networks.

3.6.1 Feed-forward neural networks

Feed-forward neural network is the oldest network type and is structured so that data only flows in a forward direction within the network. With the core to any given neural network represented by its atomic unit, the network node (or just 'unit'). Network structures are formed through ordered sets of nodes grouped in layers, forming a graph structure by connecting node pairs with unidirectional edges (or links). Links in the network function as carriers for discrete activation signals between a predecessor to its successor, and the strength of each arriving activation signal is weighted regarding some weight w associated with each transmission link.

In a common multi-layered feed-forward network, each node in a layer connects with every node in the previous layer. With fully connected layers, link weights form dense weight matrices and can be denoted $\boldsymbol{W}^{[l]} = [\boldsymbol{w}_0^{[l]}, \boldsymbol{w}_1^{[l]}, .., \boldsymbol{w}_n^{[l]}]^T$. Weight vectors $\boldsymbol{w}_i^{[l]}$ in a matrix contain signal weights between nodes in layer $l-1$ to node i in layer l, and consist of $n^{[l-1]}$ weights which equals the number of nodes in layer $l-1$. Once input is received in a node, the node generates its own activation signal in regard to the sum of weighted inputs, bias parameter $b^{[l]}$, and activation function $g^{[l]}$. Activation signals $\boldsymbol{a}^{[l]}$ contain the activation of each node in an activated layer layer l and can be calculated using the bias vector as[4]

$$\boldsymbol{a}^{[l]} = g^{[l]}(\boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}) \tag{3.20}$$

With each layer activation, the generated signal $\boldsymbol{a}^{[l]}$ is feed to the following layer, and as the following layer is activated, a recursive pattern is formed. With a network of l layers, the first layer represents the *input* layer and receives the feature determined input signal $\boldsymbol{x}$. Following network layers are thereafter referred to as *hidden layers*, containing nodes referred to as *hidden units*. Final layer for which the recursion ends is then referred to as the *ouput* layer. The output layer contain an equal number of nodes as the output feature vector y, and is calculated without activation as shown in *Eq. 3.21*.

$$\boldsymbol{a}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b} \tag{3.21}$$

3.6.2 Phase function neural network

Phase function neural networks (PFNN) [32] utilize a phase to vary parameters in the network through a phase function Θ. While otherwise similar to a dense feed-forward neural network, phase networks consist of variable sets rather than weight and bias pairs. A phase network layer l is then defined by a set containing k weight matrix and bias vector pairs. Variable set $\boldsymbol{\alpha}^{[l]}$ for a layer can be denoted as

$$\boldsymbol{\alpha}^{[l]} = \{\boldsymbol{W}_0^{[l]}, \boldsymbol{W}_1^{[l]}, .., \boldsymbol{W}_k^{[l]}, \boldsymbol{b}_0^{[l]}, \boldsymbol{b}_1^{[l]}, .., \boldsymbol{b}_k^{[l]}\} \tag{3.22}$$

[4]Depending on the definition for a node in a given layer, different equations exist for it's activation.

Here the i:th row in matrix $\boldsymbol{W}_j^{[l]}$ is associated with the i:th node in the layer, namely $\boldsymbol{w}_j^{[l]} \in \alpha_i^{[l]}$. For any given value of the phase $p = [0, 2\pi]$, the phase network form a feed-forward network by interpolating the layer's node weights and biases in regard to the phase function

$$\Theta(p; \boldsymbol{\beta}) = \{\boldsymbol{W}^{[0]}, \boldsymbol{W}^{[1]}, .., \boldsymbol{W}^{[n^l]}, \boldsymbol{b}^{[0]}, \boldsymbol{b}^{[1]}, .., \boldsymbol{b}^{[n^l]}\} \tag{3.23}$$

where the set of dense feed-forward weights and bias pairs are formed by evaluating the weights $\boldsymbol{W}^{[l]}$ and biases $\boldsymbol{b}^{[l]}$ for each layer as

$$\begin{aligned} \boldsymbol{W}^{[l]} &= \sum_{j=0}^{k} \theta_j \boldsymbol{W}_j^{[l]} \\ \boldsymbol{b}^{[l]} &= \sum_{j=0}^{k} \theta_j \boldsymbol{b}_j^{[l]} \end{aligned} \tag{3.24}$$

The first formulation in *Eq. 3.23* provide a convenience similar to the one used in the original paper over the entire network configuration $\boldsymbol{\beta} = \{\boldsymbol{\alpha}^{[0]}, \boldsymbol{\alpha}^{[1]}, .., \boldsymbol{\alpha}^{[n^l]}\}$. As shown in *Eq. 3.24*, linear weights can be calculated for each weight matrix $\boldsymbol{W}_i^{[l]}$ from the phase function $\Theta(p) = (\theta_0, \theta_1, .., \theta_n)$. With weight and bias for a layer determined in relation to the phase, the equation for layer activation is identical to a regular feed-forward network *Eq. 3.20*.

Differentiation between a phase functioned network regarding a feed-forward network can be summarized by the phase p. For any given p, the uniquely formed feed-forward network is only exposed to a subset of the full training set $\boldsymbol{X}_p \subset \boldsymbol{X}$ limiting the expected outputs to $\boldsymbol{Y}_p \subset \boldsymbol{Y}$. In regard to predicting motion states, the phase ensure that predicted joint configurations $\boldsymbol{y}$ are associated with a future state $\boldsymbol{y} \in \boldsymbol{Y}_p$ rather then $\boldsymbol{Y}$.

If a time mapping is used to define the phase, all inferred predictions associated with t, where $t \gg 0$, can no longer generalize regarding the subset $\boldsymbol{Y}_0$ associated with the timing $t = 0$. Disassociations between different subsets within the network occur since the network representation associated with the phase timing t is not exposed to the subset during training. Finding the right phase for a give motion sequences is therefore crucial for good performance in predictions, and generated motion.

Non-cyclic phase function

Originally, the phase function suggested for PFNN is a cubic Catmul-Rom spline with a cyclic phase. Since not all motions exhibit features of repeating sequences, a non-cyclic phase function using a Catmul-Rom spline and four weight layers ($n = 4$) is defined here. Interpolation weights for each weight layer can be calculated as

$$\begin{aligned}
\theta_{k_0} &= -\frac{1}{2}p_m + p_m^2 - \frac{1}{2}p_m^3 \\
\theta_{k_1} &= 1.0 - \frac{5}{2}p_m^2 + \frac{1}{2}p_m^3 \\
\theta_{k_2} &= \frac{1}{2}p_m + \frac{1}{2}p_m^2 - \frac{3}{2}p_m^3 \\
\theta_{k_3} &= -\frac{1}{2}p_m^2 + \frac{1}{2}p_m^3
\end{aligned} \tag{3.25}$$

where p_m is the normalized repeated phase $p_m = modulo((p(n-1))/(2\pi), 1.0)$ and θ_{k_i} refers to the interpolation weight for layer k_i. The index k_i is determined by the equation

$$\begin{aligned}
k_1 &= \left\lfloor \frac{p(n-1)}{2\pi} \right\rfloor \\
k_0 &= max(k_1 - 1, 0) \\
k_2 &= min(k_1 + 1, n-1) \\
k_3 &= min(k_1 + 2, n-1)
\end{aligned} \tag{3.26}$$

By not adding a fourth control point to connect the fourth and first layers in the spline the interpolation is no longer cyclic as in the original approach. Interpolation weights for the first and last weight layer then no longer share an overlapping interval (see *fig. 3.6*)

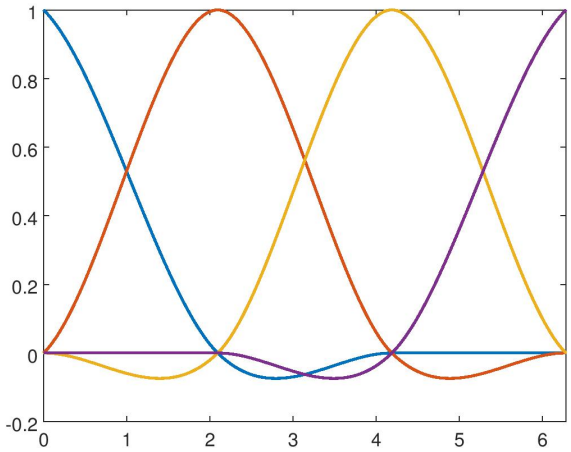


Figure 3.6: Phase interpolation weights for each layer θ_0 (blue), θ_1 (red), θ_2 (yellow), θ_3 (purple) as a function of the phase p using only three control points for the Catmul-Rom spline.

3.6.3 Dropout

Regularizing training of neural networks is important to prevent trained models from overfitting to training data. Dropout define a powerful regularization method which has shown to outperform other approaches on a wide variety of tasks [58].

Dropout regularization is an extension of the training process in which node activations are randomly dropped through a Bernoulli distribution with some probability p. The reason for dropping node activations is to prevent co-adaptation between neurons, allowing models to generalize over training data[30]. In principle, the method can be compared to training of multiple models for which the output is averaged during activation once the models are trained.

Since neurons are made redundant during training but not during inference, activations need to be adjusted by the inverse of the probability factor. Rescaling of the regularized activation received by each node is achieved either by scaling activations during training (see the biased layer activation in *Eq. 3.27*) or once the model is trained.

In the general case, dropout improves performance of feed-forward artificial neural networks but require a minimal training set to be effective. It also increases the number of iterations necessary for a model to converge and benefit from an increased network size equal to the inverse of the dropout probability used. Pairing with other regularization methods such as L2 and max-norm is also advisable to limit the activation weights from growing large. [58]

$$\begin{aligned} \boldsymbol{r} &\sim Bernoulli(p) \\ \boldsymbol{a}^{[l]} &= g(\boldsymbol{W}^{[l]}(\boldsymbol{r} \odot \boldsymbol{a}^{[l-1]}) + \boldsymbol{b}^{[l]}) \end{aligned} \tag{3.27}$$

Chapter 4

Method

Methodology for the project is focused on providing necessary motion data to facilitate experimentation, with the goal to evaluate if objective oriented motion can be generated using phase functioned neural networks. Once a final set of suitable feature sets able to model the data are determined, quasi-experiments are performed to provide quantitative measurements from which conclusions can be made. Models trained for synthesizing motion using specified features and learning algorithms are tested for significance in their ability to solve the task of synthesizing objective-oriented motion. Research questions for which experiments are set to answer are then as previously mentioned:

RQ1: *"Comparing the original PFNN approach presented in [32] with a PFNN based hierarchy model, which algorithm perform better at synthesizing objective oriented motion sequences of biped characters throwing a ball?"*

RQ2: *"In the case of motion state predictor algorithms based on dense feed-forward or phase-functioned neural networks, which algorithm performs better at training motion models effective at synthesizing motion for biped characters throwing a ball?"*

Quasi-experiments permit conclusions regarding effectiveness of trained models. Before models can be trained however, providing data for the training process is necessary. Initially, a specific set of accessible tools were decided to be used for data processing. Due to project constraints, the initial approach was deemed ill-suited to the task. Following sections try to explain reasoning behind why certain decisions was made, and why other methods were found suitable for the project. Once reasoning for the chosen approach is covered, following chapters will go deeper and explain details of applied methods. In essence, each method is a step in providing the necessary data for answering research questions.

Furthermore, since a large part of the project involve obtaining the necessary data through reconstruction of a skeletal representation from optical data, analyzing the results from the reconstruction is necessary. To facilitate this process, a set of auxiliary questions are provided but not concluded, including:

1. Are reconstructed motion sequences an accurate reconstruction of recorded motion?

2. Does the implemented reconstruction method provide skeletal motion data suitable for training motion models?

4.1 Thesis overview

To give an adequate overview of the thesis, a flow graph over the processes involved in construction of the motion dataset is available in *fig. 4.1*. The overall objective for refining the data is then for the purpose of answering the research questions which are represented in the top-right of the image. Each of the research questions mentioned in the previous section and introduction are then associated with a separate experiment.

Before execution and evaluation of the experiments are possible, the objective can be divided in a several sequence of separate processes or tasks. The notion of process in this case is used due to the dual meaning of both implementation and application of different functions to process and refine the optical data to intermediate forms.

Important intermediated forms of the refined data are shown in the flow graph and include the 'motion dataset' which contain sequences of skeletal pose data converted from the optical motion data recordings. The other important data form is then the motion models, which are trained on the refined motion sequences using different neural network algorithms described in [*chapter.7*]. Once motion models are trained, experiments can be performed by using the motion dataset to evaluate the ability of the motion models to synthesize new motion sequences.

Processes included in *fig. 4.1* can then be divided in two groups, with each group responsible for a separate processing sequence. The first sequence involves reconstructing the optical motion capture data and creating skeletal pose animations. The second sequence involves defining and extracting features to form training sets, which thereafter are used to train motion models. Since the groups involve multiple steps, an overview of both the reconstruction process and feature extraction is provided in the following two sections.

Overview of the musculoskeletal reconstruction process

Reconstruction of musculoskeletal segments from optical data as covered in the theory section involves resolving a long chain of problems. With regard to the processing steps as included in *fig. 4.1*, the entire process can be summarized by the following steps:

1. **Cleaning of optical motion data**: Before recorded optical motion capture data can be used for any purpose, it first needs to be pre-processed. Preprocessing such as labeling of motion data is described in [*section.5.1*].

2. **Multi-link modeling**: Before skeletal pose animations can be reconstructed from optical data, a kinematic chain model must be defined and parameterized for each subject. The process of defining the model with regard to the optical data is described in [*section.5.2*].

3. **Reconstruction of musculoskeletal data**: Robust conversion of optical motion capture sequences to skeletal pose animations involve minimizing marker residuals for each frame with regard to the chain model. Description of the optimization process and reconstruction is available in [*section.5.3*] and [*section.5.4*].

4. **Estimation of ball parameters**: Since throwing motion involve more data than obtained from the musculoskeletal reconstruction, a representation of the

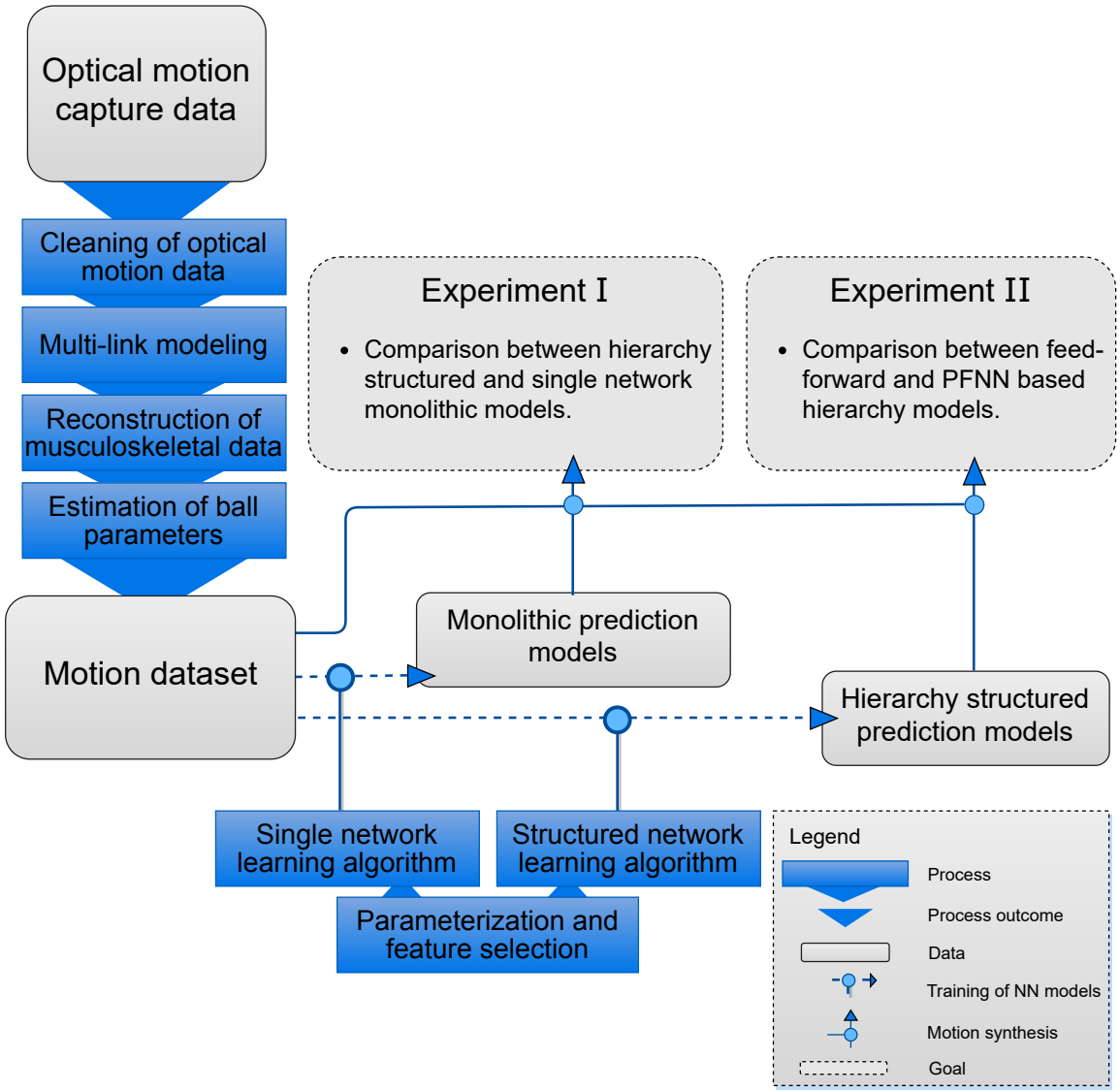


Figure 4.1: Overview over data flow and processes involved in the thesis, leading toward the goal of answering the **RQ**.

ball must also be determined from the optical data. Reconstruction of ball parameters are therefore described in [*chapter.5*]

Overview of feature extraction and synthesis processes

Training of motion models is not as complex as the reconstruction process and can be summarized in the following steps:

1. **Feature extraction**: Extracting motion features from skeletal pose animations provides training examples used for training motion models. Since multiple algorithms are used to train models, different feature sets need to be defined. Feature extraction is described in the first three sections of [*chapter.7*].

2. **Model training**: For synthesis to be possible, motion state predictor models must first be learnt from the extracted training examples. Algorithms used for training the different models are therefore described in the later sections of [*chapter.7*].

3. **Motion synthesis**: Synthesizing motion sequences using trained models require issues such as initiation of the motion state prediction process to be resolved. Description of the synthesis process is available in [*section.7.7*].

Once all processes and task are resolved, results from the experiments are presented and evaluated in [*chapter.8*] and conclusions are provided in [*chapter.10*] .

4.2 Data acquisition

Data gathering is a key part to answering the primary research questions as NN models require large quantities of data to be effective. Approach for acquiring the necessary data involved researching publicly available datasets and evaluating their relevance to the problem. With a wide variety of datasets referenced in other literature (e.g. [47, 50]), the use case for a specific data set is limited due to the narrow constraints set for the project.

Finding enough skeletal animation data for encoding ball throwing motion is a challenge and would involve creating a new data set. The problem could either approached using re-targeting techniques to fit data from different data sets to a single skeletal representation, or by processing a large set of optical motion tracking data.

From the initial investigation of public data sets, availability of skeletal animations involving throwing motion was limited. As such, the decision was made to use a data set containing optical motion tracking data created in the process of researching overarm throwing motion [51]. While the data would require non-trivial processing to be used in further stages of the project, it had qualities that made it suitable including

- Over 3000 motion sequences involving overarm throwing motion.
- Motions over multiple subjects, with 450 motion sequences recorded for each of the 7 subjects.
- Well defined and repeatable motion capture process.
- Included a minimal set of information regarding the object being thrown.

Compared to other public datasets, which lacked sufficient throw motion recordings. The selected dataset provided enough data for NN training to be feasible while also including multiple subjects, an important part for validity and repeatability of results. However, with the data consisting of only optical motion data, reconstructing a skeletal representation would pose a non-trivial problem.

4.3 Data processing

Initially the method for processing optical data would involve the use of an external tool. With a limited set of options, MotionBuilder [1] was determined suitable as it's an industry standard tool which can be applied to the problem. Unfortunately, tools involve a learning process and after extensive testing (in the context of the available time frame), the method proved to be ill suited for the project. Rather than generating a minimal number of 3 degree of freedom joint parameterization, joints in the skeleton was fit to the optical tracking data using 9. The tool thus adds 6 redundant degrees of freedom by translating and scaling joints in the reconstruction process. Considerations of another approach was therefore unavoidable.

The two other options available was either to find a more suitable tool to process the data, or to implement algorithms able to solve the problem. Both solutions posed major problems, and each involve different risks. To get a better understanding of the issue and provide repeatability, the decision was made to implement suitable algorithms applicable to the reconstruction task.

For selecting methods suitable for musculoskeletal reconstruction, methods available in literature relate to different fields and applications. Biomechanical research can however be viewed as the field in which new methodology is proposed and validated. Approaching the problem from a biomechanical point of view is therefore an obvious choice.

Biomechanics is however primarily interested in accuracy and repeatability in the reconstruction process. Applications of reconstruction methods can be defined by other factors such as performance (for real-time execution) or immersive purposes (e.g. skeletal animation for visual consumption such as Virtual Reality - VR).

Focus for selected methods is therefore similarly defined as in [31]. The goal is to minimize the amount of manual processing necessary while preserve immersive qualities. Even so, methods and qualities are not disjoint, and in contrast to a NN driven approach, the problem is approached from techniques defined in biomechanics. Reconstruction of musculoskeletal kinematics are then based in theory of segmental kinematics.

4.4 Feature selection

Feature selection for the training is based on an experimental approach and utilizes existing research to determine the features and methods used (for example see [32, 67, 2, 63]). While questions posed for the project are not unique, the action involved in the motion sequences is the primary differentiator to existing research.

Due to the number of parameters and training time involved for each model, machine learning methods for comparing and evaluating different feature sets on validation data such as cross validation [22] proved problematic. As such, an iterative approach by training models over a single subject was adopted for determining the feature set evaluated in the final experiment.

[1] https://www.autodesk.com/products/motionbuilder/overview

4.5 Experiment design and qualitative metrics

Design for the quasi-experiments was based on testing different training algorithms over motion data for which processing was completed. Tests consisted of a cross-validation approach, evaluating efficiency of models trained on a single subject's data by comparing its ability to recreate motion descriptions extracted from remaining subjects. Since reach of a subject impact the ability to conform to feature descriptions, test data was scaled in relation to the height ratio between modeled and sampled subjects.

Model effectiveness was tested using one-sided paired statistical tests under the hypothesis that a specific training method would perform better over the test data. Significance level[2] for tests where determined at an α of 0.025. Taking into considering limitations in number of subjects, variance would be reduced as measurements are averaged over ≈ 450 motions, with each motion sequence containing hundredths of frames. Probability for type II errors of $\beta = 0.1$ then allow detecting a difference of 1.62 cm from the equality assumption if the standard deviation σ is estimated to 1 cm.

Metrics for comparing effectiveness of a model's ability to adhere to motion objectives include comparing average absement, and difference between intended and estimated impact points. Absement refer to the mean distance between intended and generated end effector positions. Both qualities test the end effector's deviation from its intended trajectory, but impact differences evaluate the end effector for the throw arm during the moment the ball is released. While impact differences are self-explanatory, means over deviations from intended and generated end effector locations can be calculated as

$$deviation = \frac{1}{n_{seq} n_f} \sum_{s=1}^{n_{seq}} \sum_{i=1}^{n_f} \|\boldsymbol{p}_i - \boldsymbol{e}_i\| \tag{4.1}$$

where $\boldsymbol{p}_i$ is the end effector position generated at frame i for sequence s, and $\boldsymbol{e}_i$ is the expected position for the frame as described by motion features.

4.5.1 Evaluation of hierarchal and single PFNN models

To determine if an approach in line with the original PFNN model is fit to the purpose of objective oriented motion synthesis, the approach is compared to a structured hierarchy model trained from the same input parameters. Statistical tests can then identify if the original PFNN approach is suitable for learning objective oriented motion.

If the alternative hypothesis can be made that a poorly optimized approach using the same prior information can perform better regarding synthesis of objective-oriented motion, the null hypothesis that original form of PFNN performs equal can be used to reject the suitability of the method. If the null hypothesis can be rejected the original approach as suggested in [32] would be less than ideal for the given task.

[2]Significance level of 0.05 was adjusted to consider the case where two metrics require two separate hypothesis tests, and Šidák's correction for α of 0.05 give $\alpha = 1 - (1 - 0.95)^{1/2} \approx 0.025$

The experiment is therefor based on the simple approach of disproving a specific method for the purpose of contradicting its applicability to the specific task.

4.5.2 Evaluation of PFNN and feed-forward models

To provide an argument regarding feed-forward networks inability to perform equal in comparison to phase function networks, a paired statistical test comparing the two different learning algorithms was performed. The second experiment utilized the final models as designed during the feature selection process (see [*chapter.7*]). As such, the experiment provides performance metrics regarding a hierarchical model, while comparing feed-forward networks to phase-functioned networks.

4.6 Validity threats

Threats to validity of the results are important to consider. While issues and threats are mainly discussed in the discussion, a summary regarding threats to validity is presented in *table. 4.1*.

4.7 Ethical considerations

Ethical considerations regarding methodology is complicated only in consideration to the data for which skeletal representations are reconstructed and used in experimentation. Since the data represent human subjects, ethical considerations are appropriate. Consent from participants are a requirement when human subjects are concerned. Maintaining anonymity for participants are also necessary [64]. Due to the author not being involved in collection and execution of the initial study in [52], and as only publicly available data was obtained from [51], subject anonymity was preserved in regard to results presented in the thesis.

Subject's consent is on the other hand complicated for the same reason. In this case, the author decided to rely on the declaration available in [51], both with regard to the ethical considerations taken for the study as well as declaration of subject's consent. While further steps could be appropriate, with the original authors publishing the dataset in a cited peer-reviewed data repository[3] for scientific datasets using permissive licenses for re-use, taking further steps to verify ethical declarations seemed unnecessary.

[3]The data repository journal is available at `https://www.nature.com/sdata/`

Table 4.1: Validity threats and actions taken to mitigate the threat.

Validity threat	**Mitigation & description**
Validity of conclusions regarding statistical power and statistical tests.	Since multiple statistical tests are performed regarding the same data, determining the correct alpha value is important and is discussed in [*section.4.5*]. Assertions of assumptions regarding statistical tests are presented in [*section.8.3*].
Reliability of algorithm implementations regarding experiment outcomes.	Correct implementation of algorithms used in the process of refining the original dataset is a major concern since most algorithms are implemented for the purpose of reaching the thesis objective. To mitigate risks, tests and analysis of implemented algorithms are performed. However, most of the implemented functions only affect the musculoskeletal reconstruction. Since reconstructed animation sequences can be validated manually, inconsistencies can be limited in the ability to impact results directly. While small discrepancies are possible, risks involved in implementations associated with the reconstruction are restricted. The major validity concerns are instead associated with implementations of algorithms used in the training of motion models. To mitigate the issue, implementations either rely on external code packages or code have been tested to produce consistent outcomes. While implementations are not problem free, specific concerns are discussed in [*chapter.9*].
Bias introduced by using too few metrics.	Number of metrics measured in the experiments can threaten the conclusion since the use of a single metric can bias the result toward a certain outcome. Correct experiment design is therefore important to mitigate the issue. To avoid relying on a single metric, two different but related metrics are measured and described in the design [*section.4.5*].
Bias regarding generalizability is introduced by only comparing a single type of motion.	Generalizability for the outcome is a concern that is complicated to mitigate. Results only compare motion models trained on synthesizing throwing motion. Performing experiments on different actions would be important to generalize results. However, due to the complications involved with obtaining relevant data, the only mitigation is in the selection of an action which involve precise movements. If a synthesis algorithm can synthesize accurate throwing motion, it might be possible to generalize the process to other motions which require less precision.
Loss in generalizability due to negative effects.	Trained motion state predictor models may be efficient regarding a single metric but can impact others negatively. While using two different metrics, the metrics measure similar traits of the synthesized motion. Negative impacts are therefore possible and some of the issues are discussed in [*chapter.9*].

Chapter 5

Musculoskeletal reconstruction

Gathering correct data for evaluating the synthesis process is paramount for validity in the results and provides groundwork for realism in synthesized motion. Methods discussed in theory need to be applied in practice. Following sections give an overview of methods applied for the purpose of musculoskeletal reconstruction, and implementation is explained in four sections: pre-processing, model parameterization, multi-body kinematic optimization and filtering (representing the post-processing of the reconstructed musculoskeletal motion).

5.1 Pre-processing

Before optical tracking data recorded using stereophotogrammetry can be used for skeletal reconstruction, recorded positions in each data frame need to be correctly labeled. While labeling had partially been performed on the data, instances with missing labels, or mislabeled markers caused by marker swaps or manual error was still present. Furthermore, GOM require all markers to be available in every frame, and the problem was resolved through interpolation over sequences with marker occlusions. For pre-processing, MotionBuilder proved to be a tool suitable for the task. Downside with using the tool however was that the process involved a significant amount of error prone manual labor, to limit the time involved, pre-processing was performed for issues found in data recordings in 4 of the 7 subjects.

5.2 Model parameterization

Marker models associated with musculoskeletal segments are the base from which the musculoskeletal reconstruction is made possible. Accurately calibrated and parameterized models are detrimental for enabling the optimization function to generate an accurate reconstruction from optical marker data. Parameterization of the model involve reducing the available data to a single kinematic chain model adapted to the marker variations present between capture sessions. Overview of steps involved in model parameterization is summarized in *fig. 5.1*, and following subsections will give a short overview over each step.

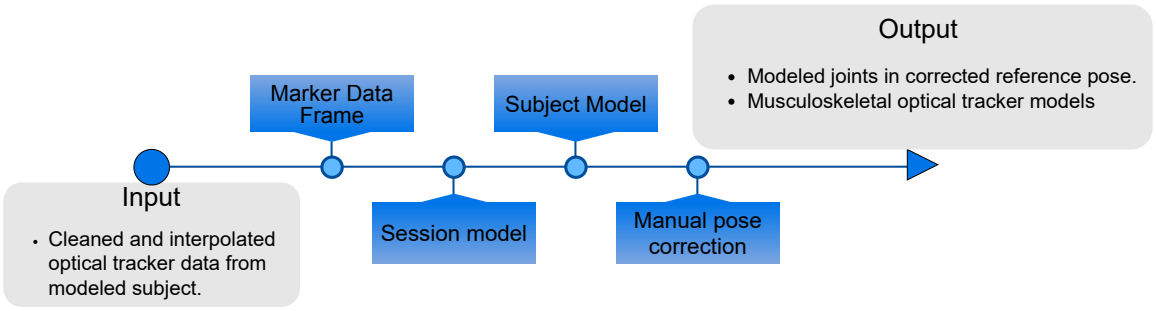


Figure 5.1: Overwiew over processing steps for parameterization of the musculoskeletal model.

5.2.1 Marker data frame

Creating a homogeneous skeleton model for a subject requires recorded sequences to be reduced to a single data frame which is representative of the markers in the space defined by the capture system. The simplest approach would be to manually select a frame from a recorded sequence and use it to represent the data (similar to [23]). However, each frame contains noise from the capture system used to record it. To minimize influence of noisy frames, the approach for determining the reference frame was instead to average marker position over multiple frames within the static trials associated with each subject and session.

Since minor movements still occur during the static trial, optical markers are aligned using a simplified segment hierarchy defined from marker clusters in the sequence. The process can be summarized in the following steps:

1. Calculate technical reference frames for each marker cluster and frame.
2. Ordered iteration of each segment from the root, utilizing SBO to minimize orientation differences between marker clusters.
3. Once marker clusters are aligned, mean for each marker position reduces the reference sequence to a single data frame.
4. Final pose for the marker model is calculated by averaging segment rotations in relativity to their parent using [46].

5.2.2 Session model

Construction of a single multi-link chain model containing a hierarchy of musculoskeletal segment models [*section.3.2.4*], primarily involve applying methodology as explained in [*section.3.2.5*]. Within this thesis, morphological reference frame refers to the reference frame estimated for a segment's joint. Parameterization of a session model largely involve determining the number of joints modeled, the type of each modeled joint, and methods for estimating the morphological reference frame of the associated musculoskeletal segment. Methods for determining morphological reference frame are discussed in [*section.3.2.3*], and models were constructed using the averaged data frame discussed in the previous section [*section.5.2.1*].

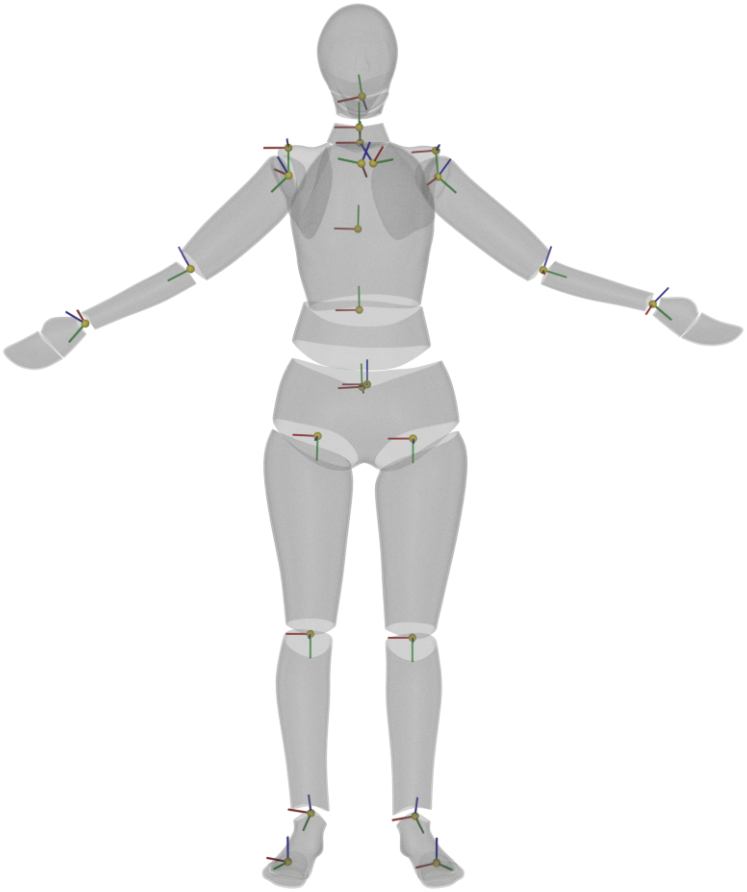


Figure 5.2: Musculoskeletal whole body model in an anatomical pose. Skin approximations are rendered as translucent surfaces, spherical joints associated with the segments are represented by yellow spheres, and colored rectangles depict reference frame axis.

Implemented multi-link chain model include 25 spherical joints, expressed as a system of 78 DoF. Following musculoskeletal segments are included for the upper body: thorax; left and right upper arms each including clavicle, scapula, humerus, lower arm[1], and a spherical joint to represent the joint complex associated with the wrist. Finally two joints are then used for the cervical spine (neck), and one to represent head orientation.

For the lower body, the model included segments for the hip; as well as the left and right legs consisting of the femur, lower leg [2], ankle joint complex, and a single joint representing toe orientation in regard to metatarsal markers. To connect the upper and lower body models, the spine was modeled using two joints with one for each of the lumbar and thoracic spine sections. A full overview of the full body model is depicted in *fig. 5.2*.

Appropriate *predictive* or *functional* method for estimating joint parameters was

[1]Lower arm contain the combined representation of the ulna and radius.
[2]Lower leg refer to the segment defined by the combined tibia and fibula.

primarily determined by recommendations by ISB [65, 66]. Predictive methods were used to the extent necessary parameters are available, which was the general case for the data set. Exception to the rule was that the lateral and medial epicondyle of the femur was used instead of lateral and medial condyle as suggested by ISB[65], in similarity with [45].

Joints for toes, neck and head did not have a recommended model, the associated joint centers were therefore approximated and left as relatively free moving joints. Finally, for joints where functional methods was recommended, the functional CoR estimation method presented in [14] was implemented. Implemented functional method was used to estimate the gleno-humeral (GH) joint center connecting humerus with scapula, and hip-joint-center (HJC) connecting pelvis (hip) with femoral head. CoR estimation was performed by taking all available data frames recorded for a session and using SVD to move the data into the parent frame in which estimation was possible.

5.2.3 Subject model

In a similar approach to the reduction to define a single data frame; a subject model is constructed by calculating an average of each of its session models. By iterating over the model hierarchy, technical reference frames are aligned using discrepancies found between the morphological descriptions between sessions. Differences between musculoskeletal segments in each session are then minimized in a least-squares sense.

Once adjustments are applied to a segment's reference frame, estimated CoR of child segments are transformed into the adjusted technical frames of their parent, recalculating CoR as the average determined within the previously adjusted reference frame. Using the new CoR estimates, technical reference frames for each child segment and session are re-estimated using the adjusted CoR and transformed back into the model space of each session.

By repeating the process, a single representation of the musculoskeletal segment hierarchy can be determined, retaining a mapping from the reduced representation to each subject and session model. Finally, using the mapping, marker cluster models unique for each session and segment are calculated within the reference frame of each segment.

5.3 Model pose optimization

Optimizing of reconstructed pose in each frame requires the multi-link chain model, optical tracker data, and an initially estimated pose to be packaged with the loss function and feed to the optimizer. The optimization process therefor involve a few steps to make this possible as summarized in figure *fig. 5.3*

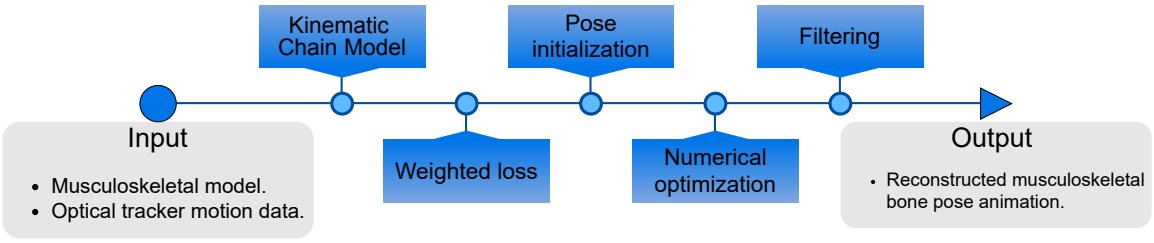


Figure 5.3: Overwiew over processing steps involved in reconstruction of the musculoskeletal hierarchy using MKO.

5.3.1 Kinematic chain model

Function evaluating the error vector, or loss, over the generalized coordinates $f(\boldsymbol{q})$ needs to be implemented and feed to the optimizer. Since the primary problem is to evaluate the affine transformations $\boldsymbol{T}_j$ for each joint (see [*section.3.5.2*]), a tree structure was used to represent the hierarchy. Weighted loss function (see [*section.5.3.2*]) is then simply evaluated by converting the hierarchy to an ordered vector representation, which define the generalized coordinate vector $\boldsymbol{q}$. Utilizing relations in the tree hierarchy, marker loss can be evaluated for any constrained vector $\boldsymbol{q}$ and optimized in relation to the loss function.

Construction of the multi-link chain model is done through multiple steps as described in previous sections. Once constructed, each hierarchy node in the hierarchy contain the relevant data to model the represented segment within its morphological reference frame defined in relation to its parent segment: optical tracker model, joint DoF, SBO estimator, boundary (hard) MKO constraints, and hierarchy relations.

It should be noted that due to sparsity in the optical marker set around the waist and spine. Segments in the multi-link model associated with the lumbar and thoracic spines are implemented as a simple pair of musculoskeletal segment nodes constrained in relation to the upper and lower body models. In effect, the MKO model can be considered as consisting of an upper body chain model and a lower body model constrained by a segment pair representing the lumbar and cervical spine.

Finally, boundary constraints and axis order for the Euler rotations are similarly defined as in [23]. Implementation of constraints representing limitations of human movement in different upper body segments used information available in [36] for reference. Minimal considerations are taken for implementation of lower body constraints. Rotation order for Euler axis was either implemented in accordance with ISB recommendations [65, 66], or from a representation simplifying the definition of the boundary constraints.

5.3.2 Weighted loss

Rather than minimize the squared error of the expanded row vector containing differences between model determined and sampled markers, the loss function is defined as the squared error of the weighted norms of differences *Eq. 5.1*.

$$\min_{\boldsymbol{q}\in\mathbb{R}^k} \quad \sum_{i=1}^{m} w_i \|\boldsymbol{R_q m}_m + \boldsymbol{v_q} - \boldsymbol{y}_m\|^2 \tag{5.1}$$

GOM utilizes a weighted error in the original paper with the purpose for weighting loss contribution to adjust influence of marker clusters as deformed by STA. Using a cluster dependent error would however be misleading as STA deformation include both a rigid cluster transformation and a non-rigid deformation of the cluster [8]. Only cluster dependent deformation is considered in the original approach, as SVD is used to define the weights (see either the original paper for GOM [45] or [13]). Instead, weighting of marker specific error contributions is performed to reduce influence from marker samples causing large deformations. Weighting marker contributions is important as marker contribution can in certain situations be considerate since errors are amplified by the error squared.

To improve robustness of the reconstruction, the weight w_i is defined as the ratio between the standard deviation for the edge deformation attributed to a marker, and the standard deviation found for all edge deformations for the shape *Eq. 5.2*.

Variance for the edge deformation associated with a marker i in a data frame is then calculated in regard to the ratio between the edge lengths for the modeled cluster $\boldsymbol{m}_j - \boldsymbol{m}_i$, and edges formed from sampled coordinates in the data frame $\boldsymbol{x}_j - \boldsymbol{x}_i$.

Utilizing the rigid body assumption, distance between markers are preserved, and the true mean of the edge ratio is 1. Variance of the edge deformation associated with a marker are then calculated as in *Eq. 5.3*, and variance for all edges within a single marker model over f data frames can then be calculated as in *Eq. 5.4*.

$$w_i = \begin{cases} \frac{\sigma_i}{\sigma_s} & \text{if} \frac{\sigma_i}{\sigma_s} \leq 1 \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

$$\sigma_i^2 = \frac{1}{n-1}\left(\sum_{j=1}^{i-1}\left(\frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{\|\boldsymbol{m}_j - \boldsymbol{m}_i\|} - 1\right)^2 + \sum_{j=i+1}^{n-1}\left(\frac{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|}{\|\boldsymbol{m}_j - \boldsymbol{m}_i\|} - 1\right)^2\right) \tag{5.3}$$

$$\sigma_s^2 = \sum_{k=1}^{f}\sum_{i=1}^{n}\frac{\sigma_{i,k}^2}{f} \tag{5.4}$$

5.3.3 Pose initialization

Initially the approach for initiating the optimization process as discussed in the original paper was tested by using single-body-optimization to initiate the optimization parameters in each frame [45]. The primary goal for such an approach would be that each frame could be estimated independently, making it possible to estimate and analyze frames without processing the entire sequence to ensure the output is correct in regard to the final output. While using a SBO estimate for each frame acted in a similar way to a simulated annealing process by introducing noise and avoiding the optimization to get stuck in local minima relative to the cost function. The

output from the SBO process generated to much noise, generating non-continuous movements in the output by introducing different local minima to the optimization process between two sequential frames.

Furthermore, due to using a separate function to initialize the optimization, the initiation approach could increase the distance from the optimized output relative to the output generated from a previous frame. Thus, a SBO approach required more iterations before the optimization process converged on a local minimum, increasing the processing time. Due to the increased noise in the output as well as the increased computation time, output from the previous frame were used for initiating each frame instead (in similarity with [23]).

5.3.4 Numerical optimization

Once problems of feeding data to, and implementation of the objective function (*Eq. 5.1*) has been resolved. Determination of an appropriate optimization method used to numerically solve the object function remain. Since evaluation time and requirement for the solver to handle boundary constraints, Sequential Least SQuares Programming (SLSQP) as implemented in [62] was used. Specifically, the solver was the available option provided by SciPy that satisfied the requirements with an acceptable convergence time per optical data frame.

5.4 Filtering

In similarity with the original data analysis [52], a fourth order low-pass Butterworth filter with a cutoff frequency of 13.4 Hz was used to remove noise present in a low dimensional representation of the joint motion. Butterworth filters are commonly used for filtering motion data and lower cutoff frequencies can be used if smoother motion is preferred over retaining high frequency motion details [23, 7]. Since movements involved in a throwing motion contain high frequency detail, using a high cutoff frequency is advisable.

After application of the Butterworth filter, a second filter process was applied to toe and ankle joints. Using a window of 100 frames, frame sequences where average position in the first half of the window differentiated by a maximum of 1.5 millimeters from the average position in the remaining filter window frames was marked. The average position within each such marked sequence was then set as the new joint position for the entire sequence, smoothing the transition in and out of each sequence by interpolation over a set number of frames at the endpoints using a Bezier curve. Further readjustments were then made by applying IK constraints to affected joints, preserving bone lengths in the reconstructed motion.

Use-case for the first filter is well documented. Purpose of the second filtering process is to remove low frequency noise produced during frame optimization in the reconstruction process. Low frequency noise create a floating effect that can be referred to as foot sliding [31, 26] or skating [44], and the outcome is unrealistic motion in the foot visible when placed firmly on the ground.

Chapter 6

Ball parameter estimation

Synthesizing throwing motion require more than reconstructed musculoskeletal poses, positioning of the ball within the hand and forces exerted from the hand onto the ball need to be estimated. In the optimal case, ball position should be determined directly from optical markers in the data frames, providing constraints for optimization of the multi-link chain model. Since only a single marker tracked the ball, the limited data made such an approach unfeasible and ball positions had to be estimated using parameters extracted from the musculoskeletal reconstruction.

6.1 Estimation of relative ball displacement

Position of the ball in the hand during the throw is determined for all throws rather than for each individual throw. Reasoning behind estimating a uniform position relative to the hand is that it simplifies the motion synthesis by reducing the number of varying variables during synthesis. There is also the problem regarding the reference frame for the wrist as used in the process, itself a product of parameter estimation. Estimations for the ball center within the local reference of the hand would therefore be unreliable and inaccurate regardless of chosen approach.

Finally, since only one marker was captured for the ball, determining the ball's center point is not possible within an optical data frame as three markers would be required when the radius is known. With further unknowns and sources of error, e.g. no finger movement, the estimation is inaccurate at best, and using a single estimate of the ball position relative to the hand can be justified.

To find a reasonable ball position within the hand, all n ball markers $\boldsymbol{m}$ from every frame before the release of the ball was moved into the reference frame of the hand. Ball center $\boldsymbol{c}$ was then minimized with regard to the transformed markers and the radius $r = 0.37$ using the equation

$$\min_{\boldsymbol{c}\in\mathbb{R}^3} \quad \sum_{i=1}^{n}(\|\boldsymbol{m} - \boldsymbol{c}\| - r)^2 \tag{6.1}$$

6.2 Ball release

Estimating trajectory for the ball require finding the release timing t_0 at which position and velocity for the ball can be interpolated using a B-spline fitted to the trajectory of the ball's center point. In relation with the previous section, the center point is approximated within the motion sequence using the ball's displacement (*Eq. 6.1*) relative to the reconstructed reference frame for the hand. Once position and velocity estimates for the ball has been found, simple laws of motion are used to determine the trajectory of the ball

$$\boldsymbol{p}_t = \boldsymbol{p}_0 + t\boldsymbol{v}_0 + \frac{t^2}{2}\begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \tag{6.2}$$

using the estimated velocity $\boldsymbol{v}_0$, position $\boldsymbol{p}_0$, and gravity constant g. The method is similar to the approach in [54], and is based on the assumption that there are no further influence from the hand on ball after its release.

The only remaining problem is to determine timing for the ball's release within the time sequence. An initial estimation for the release timing was available within the dataset, defining the release as the moment where distance between the markers placed on metacarpal head and the ball marker increased by 10% relative markers in the hand [51]. Since the timing produced trajectories which did not match the trajectory of the ball marker in the air, the release instant t_0 was re-estimated using a minimization process explained in following section.

6.3 Trajectory optimization

Rather than use the time instant for the release available in [51], the hypothesis was made that a better estimation for a believable trajectory would be to find the release time t_0 by formulating the timing of the release as an optimization problem. Similar to the method used for reconstructing the musculoskeletal segments, an objective function is formulated to find an optimal release timing.

The optimal release instant t_0 is the point in time where the trajectory as determined by *Eq. 6.2* minimizes the distance between the model estimated ball marker and sampled marker coordinates for the ball. Since the marker is placed at some offset $\boldsymbol{o}$ from the ball's center point at a distance equal to the ball's radius, the marker moves with the ball's rotation and the equation for the ball marker at a time t is expanded to *Eq. 6.3*. Rotation for the ball is then expressed as the rotation around some axis defined by the rotation $\boldsymbol{R}_{x,\gamma_0}\boldsymbol{R}_{y,\alpha_1}$ and is rotating around the axis at some angular velocity ω.

$$\boldsymbol{p}(t) = \boldsymbol{p}_{t_0} + (t - t_0)\boldsymbol{v}_{t_0} + \frac{(t - t_0)^2}{2}\begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + \boldsymbol{R}_{x,\gamma}\boldsymbol{R}_{y,\alpha}\boldsymbol{R}_{z,\omega(t-t_0)}\boldsymbol{o} \tag{6.3}$$

Considering some timing t_s as the initial estimate for the release (taken from the initial estimate in the data), finding release time $t_0 = t_s - t_d$ is solved using the

minimization problem *Eq. 6.4*. The equation is optimized using the generalized coordinates expressed as the vector $\boldsymbol{q} = (\, t_d \;\; \gamma \;\; \alpha \;\; \omega \,)$ and f_s is the sample rate (Hz) for the optical motion data.

$$\begin{aligned} &\min_{\boldsymbol{q}\in\mathbb{R}^4} \quad \sum_{i=1}^{n} \| (\frac{i-1}{f_s} + t_s + t_d)\boldsymbol{p} - \boldsymbol{y}_i \|^2 \\ &s.t. \quad 0 < t_d \leq t_{max},\ 0 \leq \gamma \leq \pi,\ -\pi \leq \alpha \leq \pi,\ 0 \leq \omega < lim_\omega \end{aligned} \tag{6.4}$$

Considering the ball marker rotating around its center and traveling along the trajectory as determined from the release time $t_s - t_d$, the optimization finds an optimal time t_0 within a time window t_{max} from t_s. The optimization thus finds an approximate solution by minimizing the sum of squared distances between the ball marker, and marker coordinates sampled for the n frames where the ball marker is visible to the capture system (not occluded). If no visible frames could be determined from frames close to the initial estimate, the sequence was discarded for further processing.

Chapter 7

Models for synthesizing throw oriented motion

Synthesizing motion through motion state prediction require the multi-link chain to be defined regarding the pose in the current and following frame, and for movements within the near future and past. Using an absolute joint parameterization for motion states provide a simple approach of doing so that is comparable with the original PFNN approach.

Even if joint movement could be generated through an absolute parameterization, issues with using the original PFNN approach to synthesize objective oriented motion sequences were apparent. Invalid joint configurations and unrealistic motion was clearly visible, and a different approach for modeling the problem would be necessary. Visible issues consisted of small to large joint dislocations, floating noise in end effectors (e.g. movement in feet joints when the foot should be firmly positioned on the ground), and inability to conform to the motion trajectory as described by features.

Resolving aforementioned problems would involve reducing the scope of the task posed to the NN. The apparent solution for doing so would involve dividing the problem in smaller parts that could be solved independently. Common approaches in the utilization of IK (inverse kinematics) provide an interesting approach. Limbs can be associated with an IK solver fitting the limb to associated end effector, providing a natural partition of the full multi-link chain model.

Approaching the problem from the angle of mapping a hierarchy of neural networks to subsets of the joint hierarchy, complexity of inference tasks posed for each network could be reduced as each network can be used to solve a shorter joint chain. Simple motion trajectories can then also provide a motion description for the NN prediction network while providing guidance for IK corrections to inaccurate joint orientations.

Rather than use a conventional IK approach with static motion sequences corrected by IK solvers to fit the world. A neural network model is proposed able to generate motion consistent with the training data, driven by relevant features, and with each frame being corrected by IK constraints to ensure generated poses match defined end effector trajectories. Before going into detail how the NN hierarchy is structured, descriptions over model features and PFNN model for which the hierarchy can be compared are defined.

7.1 Motion features

Encoding motion state predictors for inferring future motion state in frame $n+1$ from the known state at frame n, require a finite number of features from which examples in the training set can be parameterized. Examples then provide a consistent description of labeled pairs, with the given state $\boldsymbol{x}$ and inferred state $\boldsymbol{y}$, from which the time-series model can be trained.

To formalize model descriptions in later sections, a given motion state $\boldsymbol{x}$ is a row vector over known feature parameters $\boldsymbol{x} = (\boldsymbol{x}_{\boldsymbol{q}_n}^T, \boldsymbol{x}_{\boldsymbol{m}}^T)$ where the current joint state $\boldsymbol{q}_n$ is described by features $\boldsymbol{x}_{\boldsymbol{q}_n}$. Features $\boldsymbol{x}_{\boldsymbol{m}}$ then contain a description of the motion over time. The second vector $\boldsymbol{y}$ contain 'labeled' output features, which in this case is a description of the joint configuration $\boldsymbol{q}_{n+1}$ for following frame. Since multiple models with different structures are compared in the project regarding the same task, feature sets vary between models. Sub-sections will then cover features used to define a feature set, followed by descriptions of feature sets for evaluated models.

7.1.1 Feature parameterization

Numerical description of features is necessary for regression to be possible. Feature parameterization and symbols used in association with feature descriptions are declared in following subsections.

Constants

In regard to continuous motion, some parameters are constant. One case is if the ball is held, and being thrown by, the left or right arm. Descriptions of the motion therefor include two binary constants b_l and b_r of which one are set to 1 if the associated arm is throwing the ball with the other set to 0.

To define the motion goal, a row vector $\boldsymbol{i} \in \mathbb{R}^3$ contains the impact coordinate for the original motion.

Motion trajectory features

To describe the motion over a time window rather than to infer it from the current state with no temporal context, trajectories are used in similarity with [32, 67]. Description for which joint movement is inferred, motion for each limb are described by the trajectory of its end effector. End effectors used in this context for arms and legs are then locations for wrist and toe joints, respectively. The symbols used for a row vector containing trajectory features are defined by the set $\{\boldsymbol{t}_{ltoe}, \boldsymbol{t}_{rtoe}, \boldsymbol{t}_{lwrist}, \boldsymbol{t}_{rwrist}\}$ where *rwrist* is the subscript associated with the right arm (wrist).

Feature description of a trajectory include positions $\boldsymbol{p} \in \mathbb{R}^3$ for the end effector and its delta position from previous frame. Positions and their finite difference are sampled for several frames within a given window from the current frame n. Trajectory features can be summarized as the row vector

$$\begin{aligned} \boldsymbol{t} &= (\boldsymbol{t}_p, \boldsymbol{t}_\delta) \\ \boldsymbol{t}_p &= (\boldsymbol{p}_{n-fs} \ , .., \boldsymbol{p}_n, .., \boldsymbol{p}_{n+fs^+}) \\ \boldsymbol{t}_\delta &= (\delta\boldsymbol{p}_{n-fs} \ , .., \delta\boldsymbol{p}_n, .., \delta\boldsymbol{p}_{n+fs^+} \\ \delta\boldsymbol{p}_k &= \boldsymbol{p}_{k+1} - \boldsymbol{p}_k \end{aligned} \tag{7.1}$$

where f is the number of frames between each sample and s^- and s^+ is the number of samples backward and forward respectively. For following models these are set to $s^- = 2$, $s^+ = 6$ and $f = 20$, with each vector containing a total of 54 parameters. In some cases, an extra position sample is included for frame $n + 1$ and is denoted as $\boldsymbol{o}_{ltoe}$ using the same subscripts.

Joint feature representation

Joint features are defined by their absolute positions $\boldsymbol{p}_j \in \mathbb{R}^{3j}$, which represent j joint positions. Absolute orientations $\boldsymbol{a}_i \in \mathbb{R}^{6i}$ are parameterized using the concatenation of the x and y axis as sampled from the orthonormal basis. The axis parameterization includes i separate axis pairs and use the same approach as in [67]. When referring to an absolute context, joints are calculated within the reference frame in which output is estimated. For relative joint parameterizations, axis are denoted $\boldsymbol{r}_i \in \mathbb{R}^{6i}$.

Together, or individually, each parameter set form a reference frame for a given joint and is used for both input and output features. In cases where a joint position is already determined for following frame $n + 1$, positions and axis are labeled $\boldsymbol{p}_j^1$ and $\boldsymbol{a}_j^1$ respectively. In addition, delta locations (velocities) for a joint are denoted $\delta\boldsymbol{v}_i \in \mathbb{R}^{3j}$ and backward differentiation is used to calculate only the difference as parameters are standardized before being passed to the network.

To include a distance between two given joints, a description regarding the distance d is used. For models where joint parameters are calculated in relation with a moving frame of reference, delta rotation for the reference frame is included for the origin $\delta\boldsymbol{a}_o$. Forward differentiation is then used to calculate the delta axis.

7.2 Single network model

In accordance with the original paper [32], a model based on an absolute joint representation was implemented and used as reference for the final model. Feature parameterization of the network model can be summarized as the vector $\boldsymbol{x_m} = (b_l, b_r, \boldsymbol{i}, \boldsymbol{t}_{lwrist}, \boldsymbol{t}_{rwrist}, \boldsymbol{t}_{ltoe}, \boldsymbol{t}_{rtoe})$. The feature vector contains the throw arm, impact point as well as trajectories for all end effectors. Joint state input $\boldsymbol{x_q}$ is then $\boldsymbol{x_q} = (\boldsymbol{p}_j, \boldsymbol{a}_j, \boldsymbol{v}_j)$ for the full hierarchy of $j = 25$ joints. In the output, the same joints as used in the input are inferred but with the exclusion of velocity vectors, yielding the form $\boldsymbol{y} = (\boldsymbol{p}_j, \boldsymbol{a}_j)$.

Implementation differs slightly from the original approach for locomotion using PFNN. Rather than use the moving reference frame for the hip orientation projected on the xy-plane, a static definition of the capture space is used. Features are instead

defined in the static reference frame defined by the hips first position on the xy-plane. Reason for doing so is that minimal hip movements are involved in the motion sequences, and the motion remain within a limited volume in the capture space.

Another change is that in the original approach future velocities and direction of the body was inferred, adjusted, and feed back into the input. Feeding inferred information back into the network is not advisable when the intention is to ensure joint movements are responsive to the intended motion objective. Calculating features relative to the objective improves responsivity of the system.

7.3 NN hierarchy model

PFNN models in similarity with the original approach are for multiple reasons ill-suited to the task of motion synthesis when objectives for the motion is clearly defined. For one, joint features are defined in an absolute context, which essentially means that movements in different joints are independent. The upside is instead that errors are not propagated through the hierarchy.

Independent joints can be beneficial but what happens if a specific task is defined for the network? Enforcing any constraint is largely meaningless as joints can move independently of each other, fitting the last joint in a chain to an objective is redundant when all other joints are unaffected. In the case the option is chosen, joint dislocations need to be constrained as well, so why not model the problem as a connected multi-link chain instead?

The use of a single multi-link chain is possible, but a similar problem occurs as for the original model. Even if an optimization constraint is defined for the objective, learning such a complex problem with available data is not reasonable and joints will continue to float. Furthermore, limb motion does not follow the same motion phase, while joint movement is dependent between joints, dependency between limbs are not given. ‘

Motion with multiple phases are discussed in regard to quadruped locomotion for which MANN[67] was proposed. Quadruped movement[1] can have different phases for different limbs depending on gait. Multiple phases mean that the learning task for quadruped movement is poorly suited for a simple PFNN to solve since it relies on a single phase. In essence, MANN could be applied, but MANN does not address any of the problems regarding constraining training and learning objectives.

Separating the task into subtasks where separate multi-link chains are used for each subtask simplifies the network training for each separate chain. Using multiple smaller chains also provide opportunity for different phases to be defined for each chain. The proposed hierarchy then consist of multiple chain models as shown in *fig. 7.1*. Each node in the tree represent a multi-link chain (or a single joint for inferring root of the hierarchy itself) with a separate NN model trained for each chain.

Reference frame in which features are defined are then the transformation as determined from the estimated chain's parent joint (i.e. the joint the chain is attached to, using the already inferred joint orientation for following frame). Joints in each

[1] Quadruped movement refers to movement for quadrupeds, four legged animals such as dogs or cats. Two legged animals are referred to as bipeds.

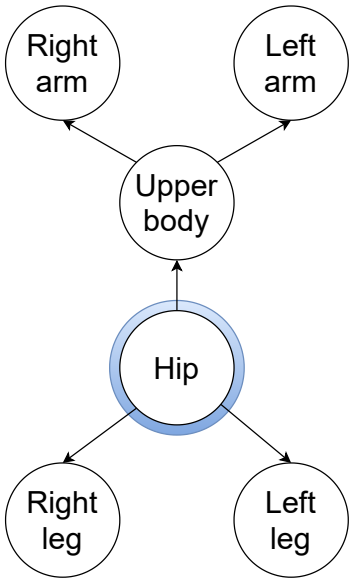


Figure 7.1: Hierarchy model as defined by the separate multi-link chain models estimated using independently trained neural network models. Each node (circle) represents a separate joint chain and neural network from which configurations of the chain is inferred. During synthesis, each network utilizes information inferred from ancestors in the hierarchy. For quality in inferred motion, movements generated for the hip joint (hierarchy root, marked in blue) is important. In particular, the generated movements in the root are important because it defines the initial reference frame which limits the possible solutions for remaining chains.

network model are then described only by their rotation in relative form with an absolute rotation for the chain's root joint. Following subsections specifies the feature sets associated with each link-chain as separately estimated within the hierarchy shown in *fig. 7.1*.

Hip features

Ambiguity for possible configurations of a predicted pose given any four end effector trajectories are problematic. Even so, determining the orientation for the root of the whole chain model can help reduce some of the ambiguity while providing modularity for the solution (i.e. providing a known position and orientation for the hip, leaving remaining joints unknown). In similarity with the absolute model, the hip was parameterized within the fixed orientation of the capture space.

Feature parameterization of the motion can be summarized as the vector $\boldsymbol{x_m} = (b_l, b_r, \boldsymbol{i}, \boldsymbol{t}_{lwrist}, \boldsymbol{t}_{rwrist}, \boldsymbol{t}_{ltoe}, \boldsymbol{t}_{rtoe})$. Feature configuration for the joints are $\boldsymbol{x_q} = (\boldsymbol{p}_j, \boldsymbol{a}_j)$ with $j = 14$. Output features are $\boldsymbol{y} = (\boldsymbol{p}_{hip}, \boldsymbol{a}_{hip})$ since only the hip is inferred.

Body features

Features used for modeling the body (hip, spine, thorax, scapula, neck and head) include motion features $\boldsymbol{x_m} = (b_l, b_r, \boldsymbol{i}, \boldsymbol{t}_{lwrist}, \boldsymbol{t}_{rwrist}, \boldsymbol{o}_{ltoe}, \boldsymbol{o}_{ltoe})$. Joint feature configuration are then $\boldsymbol{x_q} = (\delta\boldsymbol{a}_o, \boldsymbol{r}_j, \boldsymbol{p}_i, \boldsymbol{a}_i)$ for $j = 10$ and $i = 8$.

Joints not included in the output are defined in absolute form in relativity to the hip root. Joints included in the output prediction are relative. The set of joints included in the model is similar to the hip model but infers orientations for body joints in the output $\boldsymbol{y} = (\boldsymbol{r}_j)$.

Arm features

Models for paired limbs are identical reflections of each other, which means that the same feature set is used for both arm models. For the two different experiments, two near identical models are used for arm chains with either inferred or determined wrist orientations.

Motion features for models of the arms when wrist orientation is known are summarized as $\boldsymbol{x_m} = (b_l, b_r, \boldsymbol{t}_{wrist}, \boldsymbol{o}_{wrist})$ with joint features $\boldsymbol{x_q} = (\delta\boldsymbol{a}_o, \boldsymbol{r}^1_{wrist}, d, \boldsymbol{r}_{upper}, \boldsymbol{r}_{lower}, \boldsymbol{p}^1_i, \boldsymbol{a}^1_k)$ for $i = 7$ and $k = 11$. Here the distance d refers to the distance between wrist and the root joint. Joints not associated with upper and lower arm segments are also determined for following frame. Model predicted output consist of the upper and lower arm $\boldsymbol{y} = (\boldsymbol{r}_{upper}, \boldsymbol{r}_{lower})$.

Leg features

Final pair in the hierarchy use the following feature set used to describe the leg limbs. Motion features can be summarized as $\boldsymbol{x_m} = (\boldsymbol{t}_{toe}, \boldsymbol{o}_{toe})$ with joint features $\boldsymbol{x_q} = (\delta\boldsymbol{a}_o, d, \boldsymbol{r}_j, \boldsymbol{p}^1_i, \boldsymbol{a}^1_k)$ for $j = 4$, $i = 3$ and $k = 6$.

Leg movement are poorly defined relative the phase of the throwing motion. Rather than rely on the calculated motion phase in comparison with previous models, a separate phase function is defined for the joints and is described in [*section.7.4.2*].

7.4 Phase

The key problem when modeling motion based on PFNN is to define the phase. With a hierarchical structure of phase networks, different phases (and functions) can be defined for each network. For the hierarchy model, two separate phase parameterizations are used. A motion phase based on accumulated distance of the end effector for the throwing arm are used for the upper body, arms, hip chains as well as for the single model. For the leg model, rather than to use the motion phase, distance between the upper hip and toe are used to map extension and flexion of the entire leg to a configuration within the phase network.

7.4.1 Motion phase

Throwing motion (in regard to professional pitching) can be divided in six separate phases corresponding to: wind-up stride, late cocking, acceleration, deceleration and follow through [10]. With throwing motion performed by subjects with no professional training as consistent with recorded data in [52, 51], motion sequences did not contain all phases common for professional pitchers.

To automatize the labeling process, motion sequences are divided in four phases: preparation, acceleration, deceleration and follow through. With the preparation and

follow through phases including minimal movement of the throwing arm, a simple phase was defined for the motion by normalizing of the accumulated distance (velocity integral) of each sequence. Separation of the two sequences are defined by the release event t_r. Motion phase p_t at time t for a sequence of length t_e was calculated as

$$p_t = \begin{cases} c\frac{\int_0^t v\,dt}{\int_0^{t_r} v\,dt}, & \text{if } t < t_r \\ c + \frac{\int_{t_r}^t v\,dt}{\int_{t_r}^{t_e} v\,dt} \end{cases} \tag{7.2}$$

where $c = 0.66$ shifted the range to align the release with a control point in the phase function.

7.4.2 Distance phase

Leg movement did not correspond well with movement in the throwing arm. To make leg models in the hierarchy respond better to its movement as defined by feature input, distance between the position of the root of the leg chain and its end effector is used instead. To limit the phase parameter to the range $[0, 1]$, minimum and maximum distance $[d_{min}, d_{max}]$ as measured within the training data are used to normalize the phase parameter. Phase p_d associated with the distance d are then calculated as

$$p_d = \frac{d - d_{min}}{d_{max} - d_{min}} \tag{7.3}$$

7.5 Loss

Training of network models is approached as a supervised learning problem and uses stochastic gradient descent to minimize the difference between inferred joint configurations and ground truth in a least-squares sense. The original objective (cost) function for PFNN is then the sum of mean squared error (MSE) and L^1 regularization loss [32]

$$Cost(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{p}, \boldsymbol{\beta}) = \frac{1}{N}\|(\boldsymbol{Y} - \boldsymbol{\Phi}(\boldsymbol{X}, \Theta(\boldsymbol{p}; \boldsymbol{\beta}))\|^2 + \lambda|\boldsymbol{\beta}| \tag{7.4}$$

The first term in the equation is the MSE loss which defines the supervised training by differentiating predicted network output $\boldsymbol{\Phi}(\boldsymbol{X}, \Theta(\boldsymbol{P}; \boldsymbol{\beta}))$ with the expected output $\boldsymbol{Y}$. The second regularization term facilitate generalization over the data by increasing cost for overfitting units within the network, as L^1 loss penalizes large network weights.

Using a relative joint configuration in the hierarchy model do provide an opportunity for improving the objective function, however. Addition of a regularization term to penalize generated joint configurations for not reaching an objective can be used to improve the model training. Objective-oriented regularization reduces importance

of overfitting specific joint parameters to ground truth in the training set and allows trained network models to generalize beyond the training data.

A cost in regard to a determined trajectory is in similarity with the approach in [63] added to the objective function. Added cost is calculated from the mean of the end effector difference from predicted and expected output. The full cost function for the hierarchy model can be summarized as

$$Cost(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{p},\boldsymbol{\beta}) = \frac{1}{N_o}\|(\boldsymbol{Y}-\boldsymbol{\Phi}(\boldsymbol{X},\Theta(\boldsymbol{p};\boldsymbol{\beta}))\|^2+\lambda_1|\boldsymbol{\beta}|+\lambda_2^2\frac{1}{N_e}\sum_i^{N_e}(E_i(\boldsymbol{Y})-E_i(\boldsymbol{\Phi}(\boldsymbol{X},\Theta(\boldsymbol{p};\boldsymbol{\beta}))))^2 \tag{7.5}$$

where the function $E_i(\boldsymbol{O})$ calculates the position for the i:th end effector given the feature output $\boldsymbol{O}$. λ_2 represent the hyper parameter for adjusting importance of the added regularization term. N_e is the number of end effectors for which the loss is calculated. During training, one end effector is used for the limb models and two for the body models, the value for λ_2 is set to 50 and 33 respectively.

7.6 Model training

Training are performed using the Adam-trainer [39] as implemented for the Theano library based on the framework provided for the PFNN algorithm. Dropout was used for the single network model, as well as body and hip models for the hierarchy, using a retention probability of 0.7. Training sequence was divided in macro-epochs with 10 epochs each, with weight updates calculated over batches of 32 examples each. Training of the different models consisted of 40 macro-epochs for hierarchy models and 60 macro-epochs for models based on the single network approach.

7.7 Synthesis

Process for generating motion is straight forward, motion features necessary to generate a motion sequences consist of wrist and toe joint trajectories and a timing for when the ball should be released. Since the necessary feature data for evaluating the models can be extracted from subject data not used in training (i.e. data from the other subjects), the dataset provide opportunity to test generated models and allow experimenting with variations of different models.

Before motion can be generated a few conditions need to be met however, such as finding a starting pose to initiate the state prediction sequence. How these conditions are met is discussed in following sections.

7.7.1 Pose initialization

Initializing the skeletal pose from which the motion synthesis process can be initiated is achieved by finding a known pose from the original data for which end effectors closely match with their intended positions. Once a matching pose is found, iterations of the state prediction algorithm is performed until a given tolerance for the end effectors is reached or a given number of iterations are performed. Tolerance for the norm of the end effector difference was set to 1 *cm* and a maximum of 75 iterations are performed before motion frames are generated.

7.7.2 Phase input

Allowing networks to define the phase (as in the original paper [32]) proved problematic for the synthesis process, as it could diverge from valid joint parameterizations when end poses are reached before the motion ended. The effect of such behavior could be a continuation of the pose predictions beyond a valid end pose, which ended in unpredictable outcomes.

To avoid deviating from poses within the training data and create a feedback loop where the network start to extrapolate; utilization of valid motion features, emphasizing on a determined motion phase, ensured preservation of event timings in synthesized motion. One should note however that the problem persists. Providing motion descriptions for motions not defined by the training data could generate similar behaviors (e.g. feeding motion features for another type of action as used for training).

7.7.3 IK corrected model

To improve adherence for end-effectors to intended trajectories as determined for the synthesis process, a simple IK solver was implemented in similarity with the approach described in [*section.3.3.1*]. The solver utilized extension and flexion for knee and elbow joints to adjust the length of the limb, followed by rotation of the limb to fit an interpolated target position for the end effector.

Predicted movements deviating from the trajectory is given. To not over-correct joint orientations, the target for the IK correction was set to a linearly interpolated point between the preferred future trajectory and the position inferred from the motion state prediction in the frame. The IK solver is thus a post-hoc adjustment to the joint configuration and small imperfections in inferred joint orientations can be resolved.

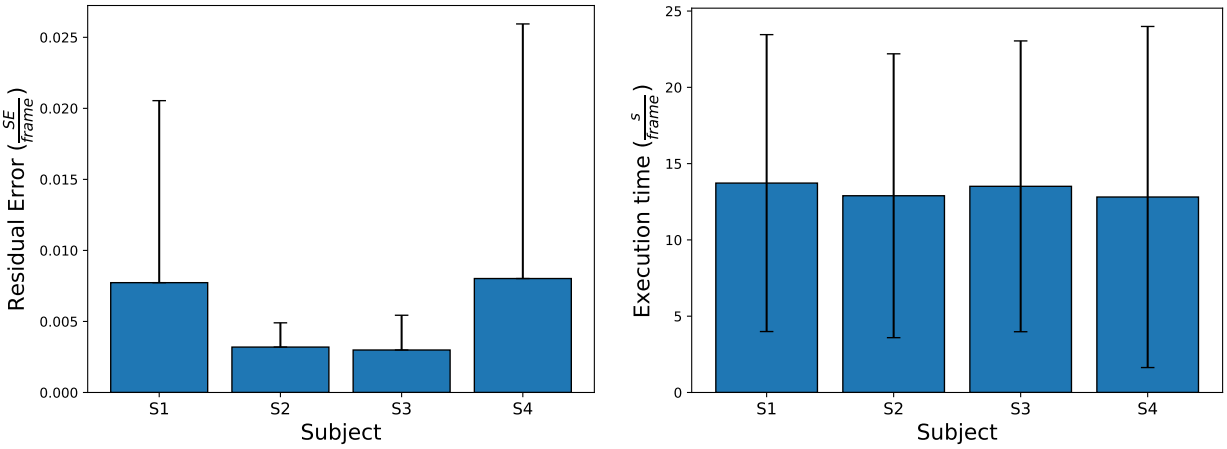
To further facilitate convergence between a preferred and originally intended trajectory unrelated to the IK correction. Sampled motion trajectory features are offset from the current joint position and blended toward the originally intended motion trajectory using a tanh function.

Chapter 8

Results and analysis

8.1 Musculoskeletal reconstruction

Measured processing time and marker residuals for reconstructing skeletal representations from optical motion capture data using MKO. Measurements in *fig. 8.1a* are per frame averages for each individual subject included in the project. Due to time constraints, there are minor differences between the optimization models used for processing subjects one and four compared to subjects two and three. To avoid unnecessarily reprocessing all motion sequences, a manually selected subset of motion sequences for subjects two and three was also re-processed using the same model.



(a) Marker residuals

(b) Processing time

Figure 8.1: Optimization results for all subjects and frames measured in the musculoskeletal reconstruction process. The first bar graph (left) show per frame subject averages of squared error (SE) residuals defined in [*section.5.3.2*]. The second graph (right) show processing time in average seconds per frame. Error bars indicate one standard deviation of variation within measured data.

8.1.1 Analysis

Residuals minimizing joint configurations are not straight forward to analyze as they are sums of squared marker residuals modulated by per frame weights. However, as a single model contain a total of 66 markers, the average marker residual can be estimated to a few cm. Markers can however have a large variation between different time frames in the motion sequence. One factor influencing residuals during movement is STA. Assumption of rigidity ensure variations in marker residuals occur since marker clusters are distorted and deformed in relation to the rigid marker model.

Observable differences in the graph show that the first and fourth subject's error residuals have a larger variation in comparison to the second and third subjects. Increased variation indicate that several sequences are poorly fit to the model in relation to the motion capture data. A primary reason for the difference is due to changes made to the musculoskeletal model between processing of different subjects.

Changes to the model was necessary as uncertainty of how to handle the spine ended with recurring issue in the skeletal representation of the lumbar spine for the first and fourth subjects. Initially the lumbar and lower thoracic spine segments only represented length constraints between the lower body (hip and legs) and upper body (thorax, arms and head) due to the lack of marker representations between the hip and thorax. The change limited joint movement to avoid free movement of the two joints. Underlying issues within the model do remain however and further considerations for resolving the spine can be made.

Finally, the overall error is relatively small for all subjects even if model variations caused small differences in reconstruction results. Reconstruction results should be adequate for the purpose of answering the posed questions in the ability to learn encodings of the data. If validity of the outcome is affected by the model variation, one could assume validity concerns to be minor since the underlying methods are the same. Purpose of testing the NN models is also to evaluate the ability to consistently learn and reproduce the data over variations in subject data, and in effect, varying subject models.

8.2 Trajectory estimates

Outcome from the trajectory estimation process are summarized in *fig. 8.2*. The heatmap contain the impact distribution for estimated trajectories as they intersect the xy-plane. The image highlights the distribution of the impacts in relation to the resolution of the histogram, denoting the total number of throws with estimated impact point within each area partition.

Measured data is defined within the reoriented capture space, aligned so that the throwing target (and therefore direction of the throw) is in the direction of the positive y-axis. Origin for each throw has also been repositioned, placing it at the subject's hip as recorded for the first frame of each sequence. It's also worth noting that the throwing target used for the trials is a 3×3 meter square placed roughly four meters away from the subject [51].

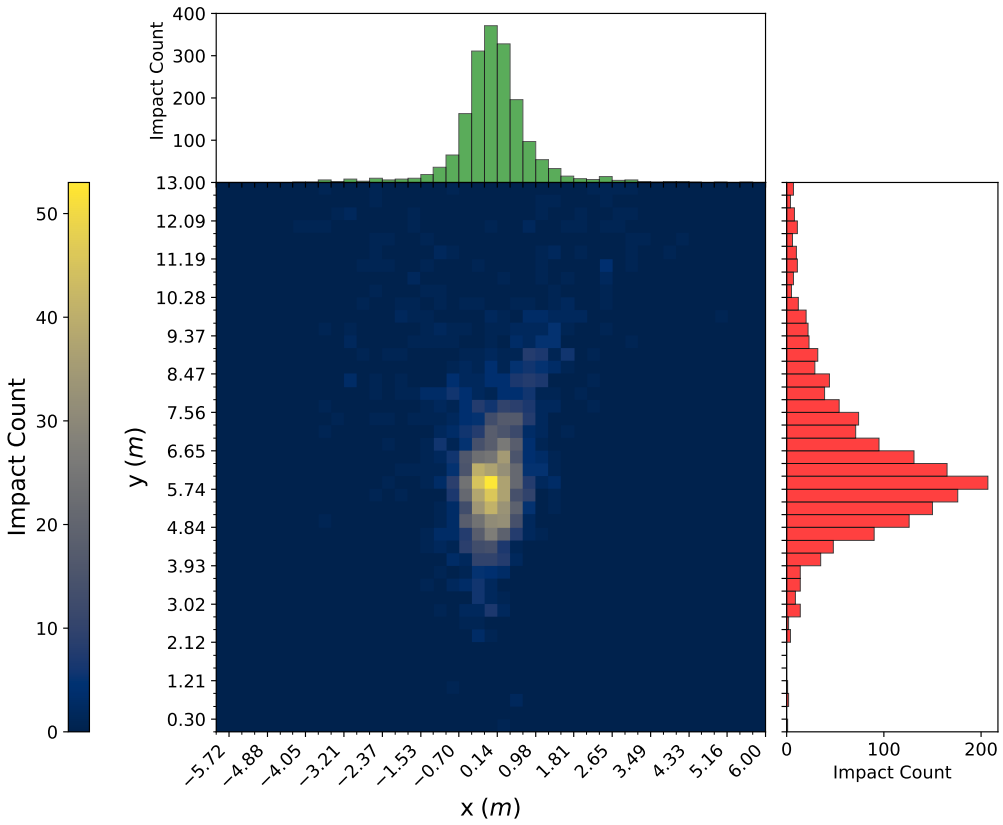


Figure 8.2: Overview over estimated impact points relative to subject's positioning in the motion capture space. The heatmap indicate spread of estimated data by counting the number of impact estimates occurring within each area of the graph. Bar graphs are included to highlight the impact distribution on the x and y axis independently. To reduce sparsity within the graph, only samples within 3 standard deviations from the sample mean are included.

8.2.1 Subject estimates

Example of subject specific impact point estimates are indicated using a scatter plot in *fig. 8.3*. Trajectory estimates are represented by their impact points as determined by intersecting the trajectory with the xy-plane. The capture space orientation is identical to the adjustments mentioned in previous section.

8.2.2 Ball speed distribution

Data in previous section indicated spatial relationships between impacts of estimated ball trajectories. Since ball speeds are recorded using a speed gun during motion trials, distributions of measured and estimated ball speeds are presented in *fig. 8.4*. Analysis of differences between measured and estimated ball speeds using a Wilcoxon test[1] showed a significant reduction in average velocity for estimated data at a significance level of $\alpha = 0.01$. Variations and average reduction in ball speed for estimated data are indicated by the mean of differences for paired samples. Estimations then

[1]Normality for the distribution of paired differences was rejected for the combined data set.

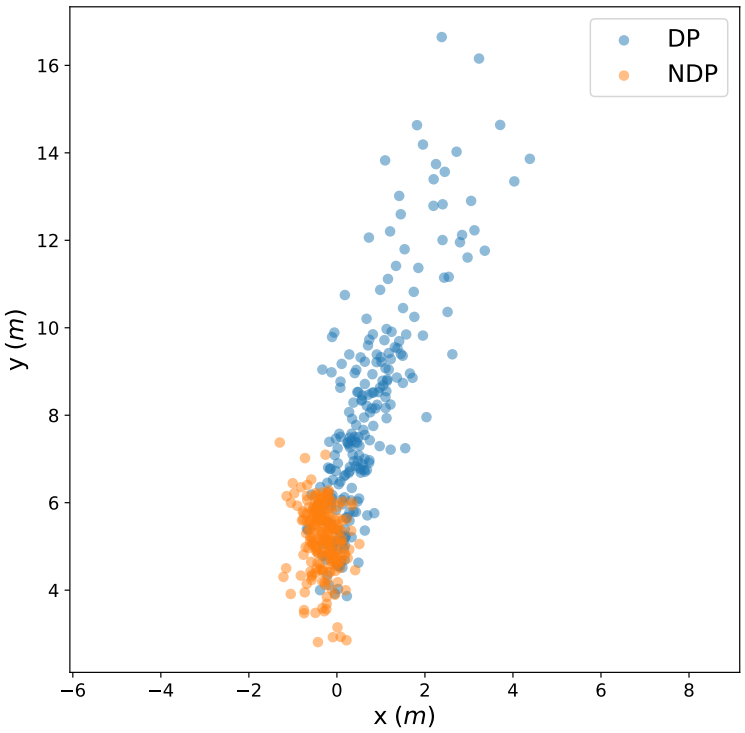


Figure 8.3: Scatter plot containing impact points as estimated for first subject. DP marks impacts from balls thrown by the dependent arm (blue), while NDP mark throws performed by the non-dependent arm (orange). A larger spread is visible for throws performed by the dependent arm. Possible outliers are also visible with a few abnormally short throws that could be evaluated further to improve the algorithm.

had an average reduction of $-1.152m/s$, and a standard deviation for paired differences of 1.685.

8.2.3 Analysis

Observations from the impact distribution and statistical test indicates that optimized estimates of release timings generate trajectories that do not accurately represent measured data. Rather, ball velocities are under-estimated relative to their paired trial measurements.

Measured differences could originate from the assumption that the initial data over-estimated release timings, release timings after the initial estimate are not considering by the optimization algorithm. Another issue with the ball reconstruction could also originate from the decision to approximate the ball offset in the hand as a fixed offset for all motions, which introduces errors in the reconstruction. To re-iterate reasoning behind the approach, due to lacking parameters for an accurate estimation, true reconstruction of the ball's trajectory would not be possible.

Considering time limitations and the necessity to provide as enough data for the

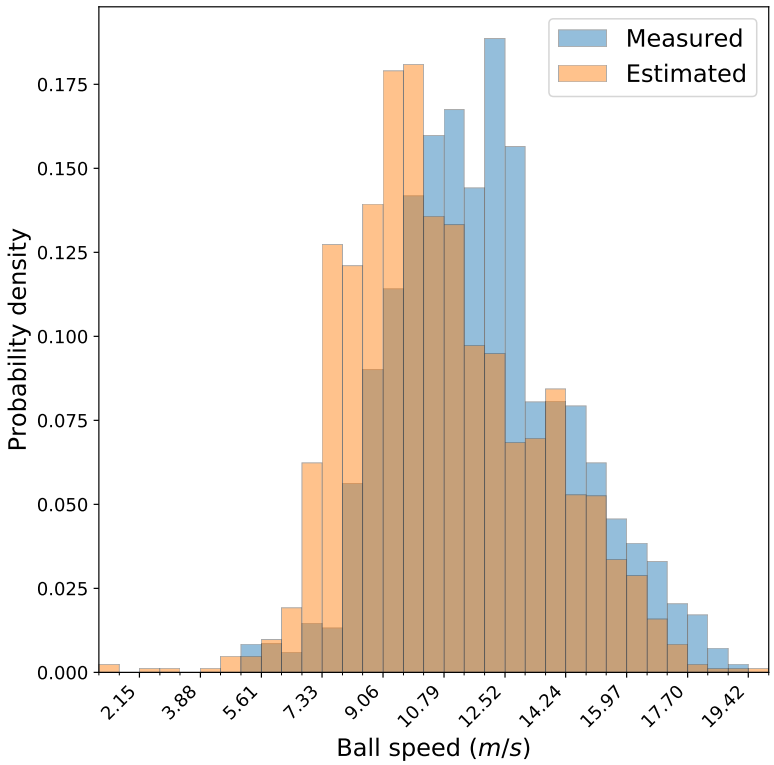


Figure 8.4: Probability histogram over ball speeds measured for paired throws in the dataset. The two histograms compare distribution between ball speeds *measured* during trials and ball speeds *estimated* from reconstructed trajectories. Estimated velocities were sampled at the time of release. Statistical testing showed estimated throws having, on average, significantly lower speed which is observable in the graph.

learning process, reconstruction accuracy is affected. Quality in joint motion when trajectory estimates are poor are still equal or similar to other reconstructed motions. In some sense, the use of an early or late release timing could be regarded as a regular throw with a poorly timed release. Again, the question for the ball reconstruction is primarily concerned with *if* estimations are accurate enough to provide valid data for training motion models. Improving the reconstruction accuracy can be viewed as future work.

To summarize, estimated trajectories for end effectors provide an indication of the conditioning of methods implemented regarding musculoskeletal reconstruction. While residual loss presented in previous section is ambiguous to interpret in part due to STA and filtering. Reconstructed wrists and ball motion provide an explicit indication of the quality of reconstructed motion. Overview and further analysis of a few selected results are therefor provided in *appendix A*.

8.3 Motion synthesis

Results from the synthesis process and experiments conducted to answer the research questions. Results from the separate experiments are presented, followed by a short analysis of the results. The section is concluded by an analysis regarding the combined results from both experiments.

8.3.1 Comparison of hierarchal and single PFNN models

Measurements from the experiment comparing models generated using a hierarchal structure of phase networks with the single phase-function network is presented in *table. 8.1* and *table. 8.2*. Evaluation of paired differences between the single network structure and the hierarchal structure is performed using Shapiro-Wilks to verify normality in measured differences. Null hypothesis of normal distribution could not be rejected for measured end effector differences but could be rejected for impact measurements.

One-sided statistical tests for an alpha of 0.025 was performed with the null hypothesis that the single-phase function model performed equal or better. Outcome showed significant improvement in the ability of all end effectors to adhere to the objective for the hierarchical model *table. 8.3*. Accuracy between different subject models and training algorithms varied, an average improvement for the hierarchy model could however be observed in *table. 8.4*.

Data samples outside 3 standard deviations from the mean was culled for the impact metric. Training data and motion sequences were sampled at a data frame rate of 200 hz. Measurements were obtained without further IK corrections, meaning predicted outcomes generated by the networks are directly feed back into the input for following frame. Extra iterations are neither performed to align the initial pose.

The monolithic approach of a single network used 4 layers with 512 nodes each. Models trained using a network hierarchy contained 6 networks of 3 layers each, with 256 nodes in each layer.

Table 8.1: Average end effector and impact measurements for the single network model. Values represent on average distance to intended outcome.

Subject	Left arm (m)	Right arm (m)	Left leg (m)	Right leg (m)	Impact (m)
1	0.1597	0.1433	0.0814	0.0915	2.034
2	0.1263	0.1001	0.0816	0.0798	2.054
3	0.2120	0.1950	0.1331	0.1444	3.975
4	0.1677	0.1420	0.1014	0.1312	1.456

Table 8.2: Average end effector and impact measurements for the hierarchy based network model. Values represent on average distance to intended outcome.

Subject	Left arm (m)	Right arm (m)	Left leg (m)	Right leg (m)	Impact (m)
1	0.0728	0.0639	0.0323	0.0328	2.007
2	0.0511	0.0496	0.0266	0.0275	2.542
3	0.0992	0.0962	0.0483	0.0427	2.181
4	0.0767	0.0574	0.0452	0.0400	1.819

Table 8.3: One sided paired t-tests comparing Single PFNN - Hierarchy models. End-effectors for limbs are measured by average deviation in displacement over all generated frames. Negative values are on average closer distance to intended end effector position for a given limb as achieved by hierarchically structured models. Null hypothesis that the single model performs equal or better are rejected for pvalue of < 0.025.

Subject	Left arm (m)	Right arm (m)	Left leg (m)	Right leg (m)
1	-0.0869	-0.0794	-0.0492	-0.0587
2	-0.0751	-0.0506	-0.0551	-0.0523
3	-0.1128	-0.0988	-0.0848	-0.1017
4	-0.0909	-0.0846	-0.0563	-0.0912
avg.	-0.0914	-0.0784	-0.0613	-0.0760
stdev.	0.0136	0.0175	0.0138	0.0209
pvalue	0.0007	0.0022	0.0023	0.0040

Analysis of the first experiment

The experiment compares different learning algorithms over multiple models. As such, small variations between algorithms add uncertainty to measured differences. Significance regarding rejection of the null hypothesis indicate that hierarchal models can generate motion which better adheres to the intended end effector trajectories. Minor variations between model parameterizations would not be able to explain the significant outcome. In particular, average distance in deviation for end effectors is roughly halved in favor for the hierarchy model.

Before the original PFNN can be rejected in its entirety however, it is important to refute it on all metrics. Even so, the minimal differences in throw accuracy between models is likely due to the experiment design. The single PFNN model in its original form inferred all joint orientations, to compare approaches on the same grounds, trajectory for the ball is left undefined relative to input features provided during synthesis.

To resolve the issue of parameterizing the relationship during testing, two different approaches were tested: using a trajectory for the ball instead of the wrist, or provide wrist motion as a known parameter during synthesis. The first approach generated situations where ambiguity in the wrist caused unrealistic motion to occur. During testing the problem was therefore left unsolved, and wrist orientation was provided

Table 8.4: One sided paired comparison between Single PFNN - Hierarchy models in regard to differences in impacts for the ball. Negative values represent an on average closer impact over all synthesized throws for the hierarchy model in regard to the intended target. Since normality for the data was rejected, no p-value was calculated.

Subject	**Impact (m)**
1	-0.0273
2	0.4873
3	-1.7758
4	0.3626
avg.	-0.2383
stdev.	0.9077
pvalue	-

as feature input to avoid inconsistencies in generated motion.

Since the approach for resolving the wrist parameterization complicated comparison between methods compared in the first experiment, neither approach was used. As such, the experiment does not provide enough parameters for the models to infer motion in the wrist, and in effect, synthesizing accurate ball trajectory is not possible. Arguments can therefore be made that to synthesize accurate objective oriented motion, no ambiguity can exist regarding the motion objective and provided input features. To achieve a given motion goal, constraints need to be well defined for accurate motion state estimators to be possible.

8.3.2 Comparison between PFNN and feed-forward networks

Results comparing structured hierarchies based of phase-functioned and feed-forward networks is presented in *table. 8.5* and *table. 8.6*. Evaluation using paired differences, under null-hypothesis that feed-forward networks performed equal or better, were conducted on all measurements as normality could not be rejected using Shapiro-Wilks tests.

Using one-sided statistical tests with an alpha of 0.025, significant differences in end effector deviation associated with leg models and the left arm could be observed. Results then favored the alternative hypothesis that PFNN models perform better as visible in *table. 8.7*. Accuracy in generated throw trajectories are also significantly improved for PFNN based models as seen in *table. 8.8*.

Outliers outside 3 standard deviations from the mean was culled for impact deviation measurements. Training data and motion sequences were sampled at a data frame rate of 200 hz. Data was obtained by IK correcting limbs and up to 75 iterations were performed to improve initial pose before synthesis process began. To provide a full representation of intended ball trajectories, wrist orientation was feed as an input feature to the network.

Feed-forward hierarchies are constructed using 6 networks with 8 layers and every layer contained 256 nodes. PFNN based hierarchy models used 6 networks with 3

layers and 256 nodes each.

Table 8.5: Average end effector and impact measurements for the feed-forward hierarchy model. Values represent on average distance to intended outcome.

Subject	Left arm (m)	Right arm (m)	Left leg (m)	Right leg (m)	Impact (m)
1	0.0662	0.0612	0.0415	0.0493	1.654
2	0.0706	0.0875	0.0409	0.036	2.226
3	0.1106	0.0817	0.0507	0.0431	1.985
4	0.0881	0.0760	0.0406	0.0594	1.252

Table 8.6: Average end effector and impact measurements for the PFNN based hierarchy model. Values represent on average distance to intended outcome.

Subject	Left arm (m)	Right arm (m)	Left leg (m)	Right leg (m)	Impact (m)
1	0.0525	0.0505	0.0178	0.0219	0.831
2	0.0391	0.0410	0.0175	0.0157	0.705
3	0.0714	0.0818	0.0380	0.0227	0.867
4	0.0583	0.0451	0.0291	0.0217	0.861

Analysis of second experiment

Measurements in *table. 8.7* show PFNN based hierarchies performing slightly better with less deviation in end effectors and marginal improvements in accuracy as observed in *table. 8.8*. An interesting outcome is that only one arm performed significantly better. While it is likely that the outcome was based on random chance, it does provide an indication that the phase parameterization used for the phase networks associated with the legs performed better overall. It is then possible that improving the phase parameterization of the motion phase is likely to improve affected motion state predictors.

Still, feed-forward networks used within a hierarchy structure can be observed to perform better then models based on a single network. While they might be interesting to reevaluate in certain cases, accuracy in motion sequences generated by PFNN hierarchy models are significantly better. This is a quite clear indication that a feed-forward approach would be a less optimal approach when throwing motion is concerned.

Table 8.7: One sided paired t-tests comparing hierarchy based PFNN and feed-forward models. End-effectors associated with each limb are measured using average displacement over all generated frames. Negative values are on average closer distance to intended end effector position for a given limb as achieved by PFNN models. Null hypothesis that feed-forward models performs equal or better are rejected for pvalues of < 0.025.

Subject	**Left arm (m)**	**Right arm (m)**	**Left leg (m)**	**Right leg (m)**
1	-0.0138	-0.0106	-0.0244	-0.0275
2	-0.0315	-0.0465	-0.0248	-0.0177
3	-0.0393	0.0001	-0.0119	-0.0211
4	-0.0299	-0.0309	-0.0109	-0.0357
avg.	-0.0286	-0.0220	-0.0178	-0.0264
stdev.	0.0093	0.0180	0.0058	0.0071
pvalue	0.0064	0.0625	0.0064	0.0038

8.3.3 Motion synthesis analysis

Both experiments are designed to compare different approaches as applied to the motion synthesis problem and evaluates trained subject models on valid features extracted from other subject's data. While every outcome did not prove significant, measurements do indicate a general trend favoring hierarchy structured PFNN models. Conclusion of the outcome regarding the research questions is stated in the following paragraphs.

Regarding the first research question **[RQ1]**, while a different variation of a monolithic PFNN based motion state predictor may perform better then proposed approach. Utilization of absolute joint parameterizations in combination with a single phase-function network proved to be an approach ill-suited to the learning task. Particularly, the monolithic PFFN approach was unable to follow intended limb movement when compared to the hierarchy approach as can be observed in *table. 8.3*.

To answer the second research question **[RQ2]**, when in-depth analysis of the motion is possible and accurate phase parameterizations can be determined, Phase-function networks perform significantly better then dense feed-forward networks. In the general case, PFNN can improve the results even with non-optimal parameterizations of the phase as can be seen in *table. 8.7*.

Table 8.8: One sided paired t-tests comparing hierarchy based PFNN and feed-forward models in regard to differences in ball impact deviation. Negative values represent on average closer impact for the PFNN model in regard to the intended target. Null hypothesis that feed-forward models performs equal or better are rejected for pvalues of < 0.025.

Subject	**Impact (m)**
1	-0.8235
2	-1.5209
3	-1.1184
4	-0.3909
avg.	-0.9634
stdev.	0.4130
pvalue	0.0136

Chapter 9

Discussion

Underlying idea for separating the motion synthesis problem in a hierarchy is to alleviate some of the problems with inferring the state of a single whole-body multi-link chain model. By dividing the problem in smaller subproblems, ambiguity for each inference is reduced as there are fewer number of possible configurations for remaining chain models. The hierarchy also provide flexibility, if a few joint parameters within the full chain model is known, nodes in the tree can be replaced or ignored. Only networks associated with unknown parameters are needed for filling gaps in the whole chain model.

Thus, the approach forms a multi-purpose solution, it can be used to fill in missing joint information, retarget animation data, and synthesize motion sequences from descriptive motion features. Final configuration only depends on the prior information available for a given sequence.

9.1 Outcome from musculoskeletal reconstruction

Analysis the musculoskeletal reconstruction in [*section.8.1*] provide clear evidence that there are imperfections in the reconstructed skeletal representations. Even if reconstructed skeletal animation is important for realism both within the data itself and synthesized sequences, imperfections does not affect the general outcome as qualities of realism are not tested. As the optical motion capture data contained valuable qualities in other areas (particularly the number of sequences and utilization of medical expertise for marker placements), reconstruction process and generated skeletal data can be viewed as a limitation of the thesis.

There are a few possible improvements to the reconstruction process when considering estimation of ball parameters. Constraining the ball marker within the optimization function similarly to how constraints are used in [23], representations for both the ball itself and wrist reconstruction can be improved.

Looking at the problem in a larger scope, replacing methods entirely is appropriate. Kalman filtering has been successfully applied and proved to be able to improve skeletal representations reconstructed from optical motion data [27, 23]. Using specialized solvers such as [20] is also an option.

Improving modeling of the multi-link chain can also be important for the reconstructed musculoskeletal representations. One example for improving the chain model could be to model markers within the morphological reference frame independently. Utilizing pre-calculated models to define unknown information (such as the

spine), all subject models can be generated based on variations of the same model, facilitating retargeting sequences between subjects.

9.2 Experimental outcome

Experiment results presented in [*section.8.3*] provide clear answers to the research questions. There is however still a possibility that improved variations of a single network model could perform better than a hierarchy model. Validity of measurements can also be a concern, particularly as models can be equalized regarding different criteria such as memory consumption and execution time. Changing parameterizations of the neural models might therefore improve validity in results.

Another important factor regarding the outcome is that the original data frame rate of 200 hz was retained through all processing steps. For models to be used in a real time context, a lower sample rate of 30 hz would be important for performance reasons. Generating models for a lower data frame rate is also important to consider when results from other research is factored in such as [63].

Discussion regarding the first and second experiments

Analyzing the outcome for the first experiment in [*section.8.3.1*] further, one key issue is if differences occurred primarily due to using an improved loss function. Inclusion of a regularization term to learn end effector movement may be the sole reason for why hierarchy models perform better.

Another important issue to consider is if performance improvements caused by the loss function improvements also had the opposite effect on realism and reconstruction accuracy. While the model's ability to adhere to given trajectory improved. Joints unaffected by the regularization term can be less expressive. For example, reduced fidelity in head movements could be observed when visually comparing results generated by single and hierarchy network models. Since neither outcome can be distinguished from current results, the problem can be considered future work.

To discuss the second experiment in [*section.8.3.1*], PFNN models improved visible smoothness in generated motion sequences when compared to feed-forward models. Continuity in the wrist may therefore be an underlying factor to why PFNN models improved accuracy in synthesized throw trajectories. Further tests on generated motion could be able to determine if that is the case. Due to dense feed-forward networks underperforming in the experiment, further tests were not considered.

Early attempts for application of PFNN did on the other hand indicate that pre-computation of phase network layers as mentioned in the original paper [32] created short but visible discontinuities in the output. Fewer overlapping weight layers may therefore be preferable for performance gains, linear interpolation of pre-computed layers might also solve the issue.

Final notes

In the end, applied techniques could be revised and improved both for the sake of validity or to provide better training data for the motion synthesis. However, decisions to apply certain techniques are dependent on the limited time available for the project. The problem of evaluating different synthesis methods is not a simple problem, the right data must be made available as well as implementations of evaluated methods.

9.3 Phase based time-series modeling

PFNN provide a solution for efficient training of time-series models due to forming a continuous set of feed-forward networks adapted to movements local to a small interval within the phase parameterization. Complexity in the method is embedded in defining the phase which demand in-depth knowledge of modeled motion. Labeling of motion sequences can therefore be an expected task to facilitate parameterization of the phase [32, 67, 4]. Manually labeling sequences to form motion primitives provide a formal method for grouping sequences representing similar movements [48].

Concepts of phase labeling and motion primitives are essentially equivalent. Motion primitives define sequences representing similar movements between two labels within the motion. Phase labels define the separation of sequences which share a common phase. Dynamic time warping as used in [48] can therefore be used to replace the motion phase approach as defined in [*section.7.4.1*].

With regard to the motion phase, assumptions are made that the end-effector movement are optimized and do not involve superfluous movement unrelated to the objective. If no unnecessary movement is included, it can be assumed that similar motions will move the end effector by the same relative distance when executing the same task. It can be stated however that the assumption does not apply to all sequences, movement is varied in the start and end of motion sequences.

Dynamic time warping as defined in [38] could therefor provide a robust approach and alleviate problems associated with the motion phase. However, introducing more phase labels by separating the sequences into specialized motion primitives is still important. Dynamic time warping is only able to adjust the phase to variations within similar sequences, not dissimilar sequences.

Automatizing phase labeling

Approaches for separating motion sequences into primitives benefit from automatic labeling processes, particularly as availability of labeled motion data is sparse in comparison to datasets available in other research areas [4].

Methodology for automatic labeling is a complicated subject. In part, segmentation of motion sequences involves viewing the synthesis problem in a larger scope. Automatic labeling relates to motion segmentation [68, 41, 4], mining of time-series data [37], and time-series classification [4, 53]. Navigating mentioned topics are not simple as evident in [37].

If automatic labeling is not possible for a given task, manual labeling should be an expected task to facilitate motion learning as evident from [32, 67]. While an optimal process would include time for adequate analysis and labeling of data, project constraints generally prevented in-depth analysis of the motion. For cases where analysis is not possible, MANN rather than PFNN might be appropriate.

Encoding IK solvers in neural networks

Constructing the motion state predictor based on a method for encoding accurate IK solvers using feed-forward networks [2] was attempted but failed at replicating the success. It can be assumed that this was due to limiting model training to only use the recorded subject data. To successfully encode a function for the limb rather than to learn a set of state changes, it's likely that training data need to saturate at least the full range of motion to get similar results to [2].

Augmenting reconstructed data by generating new joint configurations with simple or no constraints would defeat the purpose of using neural-network models. Particularly, augmenting the data with unrealistic motion is redundant since the general IK problem can be solved numerically. The purpose for training neural network models is to reproduce valid joint configurations based on a training set of realistic examples.

Non-cyclic phase networks could on the other hand be used as data-driven IK solvers by using a similar approach to the distance phase as defined for the leg chains in [*section.7.4.2*]. In comparison with a phase over accumulated distance, the approach could plausibly work on a minimal set of data. Application of phase-networks as data driven IK solvers by using the reach of a limb to be enforced by the phase could prove to be simple and efficient solutions to solving smaller IK problems.

As a final note, a case can be made for the multi-link chain model of the arm to be expanded. Range of motion for the arm is dependent on the configuration of the shoulder complex [36]. Associating the shoulder complex in the multi-link chain model with the arm rather than body chain may improve quality in generated arm movement.

9.4 Viability of hierarchal network structures

Hypothesis tests conducted in [*section.8.3*] allow the statement that an absolute joint parameterization within a fixed coordinate system is probably not suitable for synthesizing ball throwing motion using a single PFNN. The statement does not concern itself with the usefulness of hierarchal PFNN structures as applied to the learning task. Taking other possible approaches in consideration is therefore important when discussing the viability of training motion state predictors using phase-function networks.

Encoding motion primitives

A majority of published research focuses on generating sequences of walking or running motion [29, 33, 63, 6], and while not all methods as described are directly applicable to other learning tasks [63]. Motion synthesizers designed for locomotion does not translate to other types of motion primarily due to the specific techniques and/or parameters used. Concepts suitable for generating periodic walking motion generally does not map to aperiodic motion. Instead, the methods and network structures available in research can be adapted for the purpose of synthesizing sequences of objective-oriented motion.

The idea behind separating a motion synthesis problem in a hierarchy structure can therefore be more relevant then the effectiveness of trained PFNN models. Synthesizing realistic motion that adhere to a specific set of objectives with high accuracy is the key from which other problems such as locomotion controllers can be solved. Encoding lower dimensional manifolds as done in [33, 29] over motion primitives, in combination with learning policies or generator networks based on non-ambiguous motion features, may prove to be a better approach compared to training motion state predictors.

In the case other solutions are preferable, for real-time problems and cases such as augmented reality (AR) or virtual reality (VR), accurate motion state predictors may still be viable. Skeleton representations then need to be generated from the current state in real time with minimal prior knowledge regarding future trajectories. In such cases there is no opportunity to wait until more information is available.

Problems where motion is generated based on limited temporal contexts are then unavoidable, a trait shared by motion state prediction methods. If there exist ambiguity in the possible configuration of the state, state predictions eventually end up in situations where limbs are misaligned with the intended motion. Realignment of affected limbs can therefore become unavoidable. Hence, for real-time cases where solutions need to be efficient with minimal considerations to future states, motion state predictors may be able to provide simple but efficient solutions. In specific cases, motion state prediction and PFNN can therefore be viable even if other methods prove superior in general.

Instability in hierarchy models

Before application of proposed approach can be realized, the proposed hierarchy structure is not without unresolved problems. One of the primary issues originates from the separation and modeling of the problem as a connected set of multi-link chains. Since different joint chains are inferred independently using separate models, generated joint configurations for different chains may originate from different motion sequences. This provide opportunity for unrealistic joint configurations to take form if different prediction models infer poorly matched configurations. While the problem is alleviated by sharing phase parameterizations between models and limiting the number of possible outcomes relative to the current phase. Mismatching joint configurations inferred by different hierarchy models are still possible.

One example of mismatching predictions occurred during configuration tests where the network model associated with a link-chain for the arm is unable to dis-

tinguish the feature for not being the throwing arm. Generated motions contained extreme angular velocities in the joints under assumption that the arm is throwing the ball, a clear example of the network extrapolating when faced with an unknown situation.

In certain cases, effects causing extreme behaviors can also resonate within the whole model and cause undefined behavior. Retargeting subject data to create larger training sets might therefore be able to reduce the problem. Removing ambiguities in prediction models is another possible solution. Necessity of adequate amount of data is however common when applying NN to synthesis problems [59, 60, 4]. Instability in current models are then the reasons to why evaluating realism in human oriented trials is unnecessary. While it would be interesting to compare the current solution to assert the ability of phase-functioned networks to generate continuous motion in comparison with feed-forward approaches. Focus on improving the prediction models has more relevance.

Training dependent hierarchy models

An underlying problem within the hierarchy approach originate in the training of independent network models. To create a coherent state prediction model, dependency between models need to be introduced in the training process. Essentially, the hierarchy re-introduces a similar problem the PFNN approach resolves in comparison to using a sequence of feed-forward networks. Training of piecewise or disconnected network models introduce discontinuities during transitions between different prediction networks [32].

Combining the training of ancestors and their children within the tree structure can be an approach for introducing dependency between models and solve aforementioned problems. The combined training can then either be achieved by training children using outcomes generated by their ancestors, or by introducing a loss penalizing end-effector residual when training ancestors. Introducing dependencies by inferring the full poses using pre-trained networks for the remaining joint chains.

Integrating the IK solver in the training process may also be key to adapt models to the post-processing of each frame. Taking IK integration even further, controlling smaller multi-link chain models using IK techniques (e.g. [49]) in combination with neural network models can also be an interesting but separate approach.

Summary

To summarize, structuring the motion synthesis problem based on a hierarchy of separate multi-link chain models, can have viable use cases. For hierarchy models to be viable however, stability of proposed methods needs to be improved. Reworking the training procedure and introducing dependencies between network models is therefore necessary to improve stability within the hierarchy model.

Chapter 10

Conclusions and future work

To summarize topics discussed in previous sections, the chapter reiterate some of the interesting thoughts and provide a clear conclusion to the research questions.

10.1 Conclusions

Dividing the learning task in a hierarchy of multi-link chains and distributing each task to a separate neural network improves the ability to learn objectives such as fitting a chain of joint segments to follow a given trajectory. As evident from experiment outcomes, results presented in [*section.8.3*] can be used to answer the research questions:

RQ1: Comparison between the hierarchy structured and single PFNN models showed that the single PFNN approach using absolute joint parameterization proved to be inadequate for the purpose of synthesizing ball throwing motion. Instead, hierarchy models performed significantly better at adhering to the motion trajectories determined for inferred motion sequences.

RQ2: Comparison between hierarchy models based on phase-function and feed-forward networks indicate that PFNN models outperform feed-forward models regarding the task of training motion state predictor models for the purpose of synthesizing ball throwing motion.

Contribution can then be summarized three-fold. First, conclusion regarding the research questions show that a similar approach to the original method used to synthesize motion using PFNN does not generalize well to new objective-oriented synthesis tasks. Instead, approaching the motion synthesis problem from a separate point of view should be the focus in future research. A reasoning that can be further supported by results achieved by solutions such as [63] in regard to the locomotion problem.

Secondly, when considering application of different neural network structures, phase-function networks provide a clear benefit over feed-forward networks when considering the problem of inferring future states in time-series data.

Third, applied methodology provide an approach for skeletal representations to be generated from the dataset used in the study. Opportunity to further test existing and future methods is then made possible on motion containing the specific goal-oriented action.

Taking a wider view on the results, the experimental outcome also provide opportunity to reflect over recent work such as neural state machines [59] and the approach

of using localized phases [60]. Both methods rely on generalizations of PFNN by extending the use of features and network structures to synthesize a continuous stream of objective oriented motion. While largely different from the approach used here, utilizing other approaches might be better suited when the task is to synthesize a specific motion sequence. Particularly, generating specific motion sequences can be complicating when using solutions based on phase-function networks.

Furthermore, some of the relevant topics raised in the discussion can be summarized as:

- Musculoskeletal reconstruction is a complex and diverse subject. Minimization of marker residuals in relation to a system of links, referred to as a multi-link chain model, modeling the musculoskeletal structure allow accurate skeletal representations to be reconstructed in relation to the rigidity assumption.

- Motion state predictors, and motion manifold encoding in general, operate as lossy compression of motion data. Without adequate parameterization of the motion objective, ambiguities arise in how the features should be interpreted during synthesis.

- Motion state predictors may not be applicable to all motion synthesis tasks. However, for problems such as AR or VR operating in real time contexts where motion states must be continuously generated with no or minimal information of future states, motion state prediction may prove to be a relevant approach when compared to other methods.

10.2 Future work

Motion synthesis is a complex problem and developing new methods to approach the subject will be key in future research. Since research in the area is moving forward at a fast pace, takeaways from the project may therefore be in the reconstruction of throwing motion which can be used to evaluate future models. Even so, continuation of the research can take many forms, and following paragraphs cover some of the interesting topics.

Retargeting motion from different subjects to a single training set allows models to be trained with more data. Resampling data at 30 hz allows models to be used in a real time context which is key to be able to put results in a wider perspective. Testing realism in reconstructed and generated motion is also an important step for widening the understanding of applied methods.

In contrast, further improvements to a hierarchy structured motion state predictor is necessary. Particularly in consideration to stability in generated models. One interesting approach would be to introduce dependency in the training process both within hierarchy models and between the post-processing as applied to inferred output. Once implemented, comparison with other motion synthesis would be a requirement for further evaluation.

Constructing small IK solvers from phase-function networks which can be adapted to training data may also produce interesting outcomes. Creating a small simple

network able to fit a small link-chain to an end effector can be an efficient way for generating valid joint configurations learned from training data.

Regarding future approaches for synthesizing motion data, hierarchy models may not be appropriate depending on the learning task. Other methods based on motion primitives in combination with encoding of short motion sequences into motion manifolds using phase-function or recurrent-neural networks may be better at the task of synthesizing objective-oriented motion.

Other possible options are to investigate motion state parameterizations as used in neural network-based approaches to motion synthesis. Evaluating possible parameter variations or comparing key concepts can be more relevant then proposing new methods to synthesizing specific motions.

For musculoskeletal reconstruction, Kalman filtering can be used to improve the reconstruction process. Processing all subjects would also increase the accuracy in statistical tests and analysis performed regarding methods applied on the data.

References

[1] Markerless motion capture through visual hull, articulated icp and subject specific model generation. *International Journal of Computer Vision*, 87(1):156–169, 2009.

[2] Ahmed R. J. Almusawi, L. Canan Dülger, and Sadettin Kapucu. A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242). *Computational Intelligence and Neuroscience*, 2016, 2016.

[3] Carolyn Anglin and Urs Wyss. Review of arm motion analyses. *Proceedings of the Institution of Mechanical Engineers. Part H, Journal of engineering in medicine*, 214:541–55, 02 2000.

[4] Andreas Aristidou, Daniel Cohen-Or, Jessica K. Hodgins, Yiorgos Chrysanthou, and Ariel Shamir. Deep motifs and motion signatures. *ACM Trans. Graph.*, 37(6), December 2018.

[5] Andreas Aristidou, Joan Lasenby, Yiorgos Chrysanthou, and Ariel Shamir. Inverse kinematics techniques in computer graphics: A survey. *Computer Graphics Forum*, 37:35–58, 09 2018.

[6] E. Barsoum, J. Kender, and Z. Liu. Hp-gan: Probabilistic 3d human motion prediction via gan. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1499–149909, 2018.

[7] Mickaël Begon, Colombe Bélaise, Alexandre Naaim, Arne Lundberg, and Laurence Chèze. Multibody kinematics optimization with marker projection improves the accuracy of the humerus rotational kinematics. *Journal of Biomechanics*, 62:117 – 123, 2017. Human Movement Analysis: The Soft Tissue Artefact Issue.

[8] T. Bonci, V. Camomilla, R. Dumas, L. Chèze, and A. Cappozzo. Rigid and non-rigid geometrical transformations of a marker-cluster and their impact on bone-pose estimation. *Journal of Biomechanics*, 48(15):4166 – 4172, 2015.

[9] Vicky Bouffard, Mickael Begon, Annick Champagne, Payam Farhadnia, Pascal-André Vendittoli, Martin Lavigne, and François Prince. Hip joint center localisation: A biomechanical application to hip arthroplasty population. *World journal of orthopedics*, 3(8):131 – 136, 2012.

[10] Gary J. Calabrese. Pitching mechanics, revisited. *International journal of sports physical therapy*, 8(5):652–660, Oct 2013.

[11] A Cappozzo, F Catani, U Della Croce, and A Leardini. Position and orientation in space of bones during movement: anatomical frame definition and determination. *Clinical Biomechanics*, 10(4):171 – 178, 1995.

[12] Aurelio Cappozzo, Ugo Della Croce, Alberto Leardini, and Lorenzo Chiari. Human movement analysis using stereophotogrammetry: Part 1: theoretical background. *Gait & Posture*, 21(2):186 – 196, 2005.

[13] John H. Challis. A procedure for determining rigid body transformation parameters. *Journal of Biomechanics*, 28(6):733 – 737, 1995.

[14] Lillian Y. Chang and Nancy S. Pollard. Constrained least-squares optimization for robust estimation of center of rotation. *Journal of Biomechanics*, 40(6):1392 – 1400, 2007.

[15] Lorenzo Chiari, Ugo Della Croce, Alberto Leardini, and Aurelio Cappozzo. Human movement analysis using stereophotogrammetry: Part 2: Instrumental errors. *Gait & Posture*, 21(2):197 – 211, 2005.

[16] Simon Clavet. Motion matching and the road to next-gen animation. In *GDC*, 2016.

[17] Sonia Duprey, Laurence Cheze, and Raphaël Dumas. Influence of joint constraints on lower limb kinematics estimation from skin markers using global optimization. *Journal of Biomechanics*, 43(14):2858 – 2862, 2010.

[18] Sonia Duprey, Alexandre Naaim, Florent Moissenet, Mickaël Begon, and Laurence Chèze. Kinematic models of the upper limb joints for multibody kinematics optimisation: An overview. *Journal of Biomechanics*, 62:87 – 94, 2017. Human Movement Analysis: The Soft Tissue Artefact Issue.

[19] Rainald M. Ehrig, William R. Taylor, Georg N. Duda, and Markus O. Heller. A survey of formal methods for determining the centre of rotation of ball joints. *Journal of Biomechanics*, 39(15):2798 – 2809, 2006.

[20] Kenny Erleben and Sheldon Andrews. Solving inverse kinematics using exact hessian matrices. *Computers & Graphics*, 78:1 – 11, 2019.

[21] S. Ali Etemad, Ali Arya, Avi Parush, and Steve DiPaola. Perceptual validity in animation of human motion. *Computer Animation and Virtual Worlds*, 27(1):58–71.

[22] Peter Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York, NY, USA, 2012.

[23] Vincent Fohanno, Mickaël Begon, Patrick Lacouture, and Floren Colloud. Estimating joint kinematics of a whole body chain model with closed-loop constraints. *Multibody System Dynamics*, 31(4):433–449, Apr 2014.

[24] Sahan S.Hiniduma Udugama Gamage and Joan Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of Biomechanics*, 35(1):87 – 93, 2002.

[25] Xavier Gasparutto, Nicola Sancisi, Eric Jacquelin, Vincenzo Parenti-Castelli, and Raphaël Dumas. Validation of a multi-body optimization with knee kinematic models including ligament constraints. *Journal of Biomechanics*, 48(6):1141 – 1146, 2015.

[26] Jason Gregory. *Game Engine Architecture*. CRC Press, Boca Raton, Florida, USA, 3rd edition, 2018.

[27] F. De Groote, T. De Laet, I. Jonkers, and J. De Schutter. Kalman smoothing improves the estimation of joint kinematics and kinetics in marker-based human gait analysis. *Journal of Biomechanics*, 41(16):3390 – 3398, 2008.

[28] Ivar Gustafsson and Holmåker Kjell. *Numerisk Analys*. Liber AB, Stockholm, Sweden, 2016.

[29] Félix G. Harvey and Christopher Pal. Recurrent transition networks for character locomotion, 2018.

[30] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[31] Daniel Holden. Robust solving of optical motion capture data by denoising. *ACM Trans. Graph.*, 37(4), July 2018.

[32] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4):42:1–42:13, July 2017.

[33] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans. Graph.*, 35(4):138:1–138:11, July 2016.

[34] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, SA '15, New York, NY, USA, 2015. Association for Computing Machinery.

[35] Hans Kainz, Christopher P. Carty, Luca Modenese, Roslyn N. Boyd, and David G. Lloyd. Estimation of the hip joint centre in human motion analysis: A systematic review. *Clinical Biomechanics*, 30(4):319 – 329, 2015.

[36] Adalbert Kapandji. *The Physiology of the Joints, vol. 1: Upper Limb*. Handspring Publishing, Pencaitland, Scotland, 7 edition, 2019.

[37] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8:154–177, 08 2005.

[38] Eamonn Keogh and Michael Pazzani. Derivative dynamic time warping. *First SIAM International Conference on Data Mining*, 1, 01 2002.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[40] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, July 2002.

[41] Björn Krüger, Anna Vögele, Tobias Willig, Angela Yao, Reinhard Klein, and Andreas Weber. Efficient unsupervised temporal segmentation of motion data, 2015.

[42] Alberto Leardini, Claudio Belvedere, Fabrizio Nardini, Nicola Sancisi, Michele Conconi, and Vincenzo Parenti-Castelli. Kinematic models of lower limb joints for musculo-skeletal modelling and optimization in gait analysis. *Journal of Biomechanics*, 62:77 – 86, 2017. Human Movement Analysis: The Soft Tissue Artefact Issue.

[43] Alberto Leardini, Lorenzo Chiari, Ugo Della Croce, and Aurelio Cappozzo. Human movement analysis using stereophotogrammetry: Part 3. soft tissue artifact assessment and compensation. *Gait & Posture*, 21(2):212 – 225, 2005.

[44] Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. Motion fields for interactive character locomotion. *Commun. ACM*, 57(6):101–108, June 2014.

[45] T.-W. Lu and J.J. O'Connor. Bone position estimation from skin marker coordinates using global optimisation with joint constraints. *Journal of Biomechanics*, 32(2):129 – 134, 1999.

[46] F. Landis Markley, Yang Cheng, John L. Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007.

[47] James McCann and Nancy S. Pollard. Responsive characters from motion fragments. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.

[48] Jianyuan Min and Jinxiang Chai. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Trans. Graph.*, 31(6):153:1–153:12, November 2012.

[49] Eray Molla and Ronan Boulic. Singularity free parametrization of human limbs. In *Proceedings of Motion on Games*, MIG '13, page 187–196, New York, NY, USA, 2013. Association for Computing Machinery.

[50] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn, June 2007.

[51] Gizem Ozkaya, Hae Ryun Jung, In Sub Jeong, Min Ra Choi, Min Young Shin, Xue Lin, Woo Seong Heo, Mi Sun Kim, Eonho Kim, and Ki-Kwang Lee. Three-dimensional motion capture data during repetitive overarm throwing practice, Nov 2018.

[52] Gizem Ozkaya, Min Soo Kim, Hye Ree Kim, Seongjun Kim, and Ki K. Lee. Relationship between the ball velocity and upper extremity kinematics during an overarm throwing self-practice program. 2017.

[53] Fotini Patrona, Anargyros Chatzitofis, Dimitrios Zarpalas, and Petros Daras. Motion analysis: Action detection, recognition and evaluation based on motion capture data. *Pattern Recognition*, 76:612 – 622, 2018.

[54] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018.

[55] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.*, 35(4), July 2016.

[56] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[57] Wei Shen, Ke Deng, Xiang Bai, T. Leyvand, Baining Guo, and Z. Tu. Exemplar-based human action pose correction and tagging. pages 1784–1791, 06 2012.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[59] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6), November 2019.

[60] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM Trans. Graph.*, 39(4), July 2020.

[61] Swati Upadhyaya and Won-Sook Lee. Survey of formal methods of hip joint center calculation in human studies. *APCBEE Procedia*, 7:27 – 31, 2013. The 3rd International Conference on Biomedical Engineering and Technology - ICBET 2013.

[62] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[63] Z. Wang, J. Chai, and S. Xia. Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2019.

[64] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[65] Ge Wu, Sorin Siegler, Paul Allard, Chris Kirtley, Alberto Leardini, Dieter Rosenbaum, Mike Whittle, Darryl D D'Lima, Luca Cristofolini, Hartmut Witte, Oskar Schmid, and Ian Stokes. Isb recommendation on definitions of joint coordinate system of various joints for the reporting of human joint motion—part i: ankle, hip, and spine. *Journal of Biomechanics*, 35(4):543 – 548, 2002.

[66] Ge Wu, Frans C.T. van der Helm, H.E.J. (DirkJan) Veeger, Mohsen Makhsous, Peter Van Roy, Carolyn Anglin, Jochem Nagels, Andrew R. Karduna, Kevin McQuade, Xuguang Wang, Frederick W. Werner, and Bryan Buchholz. Isb recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion—part ii: shoulder, elbow, wrist and hand. *Journal of Biomechanics*, 38(5):981 – 992, 2005.

[67] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.*, 37(4):145:1–145:11, July 2018.

[68] F. Zhou, F. De la Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2013.

Appendix A

Examples of reconstructed ball trajectories

Overview of possible outcomes from the estimation process using a selection of 2-dimensional scatter plots. Trajectories in 3-dimensional space are projected on either the xy-plane viewing the trajectory from above (left in the paired image), or the yz-plane providing a side view of the trajectory (right image). The direction of the trajectories follows the positive y-axis, which flows from the bottom-to-up in images to the left, and left-to-right in images to the right. One possible grouping of the images used here is to divide the images over the perceived goodness of fit for the trajectory before release, and the quality of the trajectory after the ball is released.

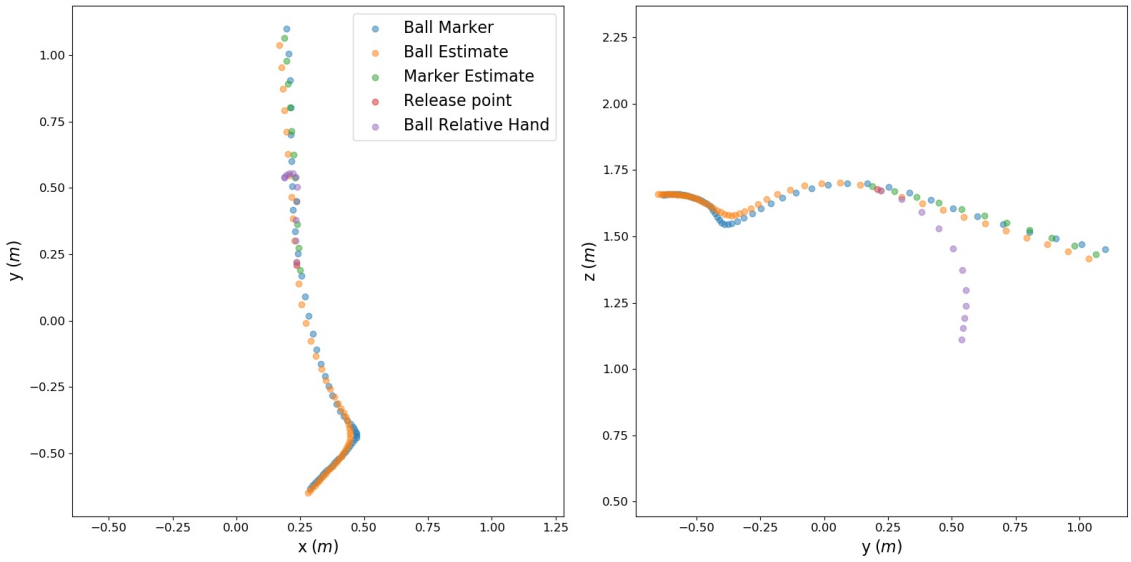


Figure A.1: **Good fit - Similar Trajectory**; Trajectory for the reconstructed center point of the ball (orange) is able to fit well to the ball's marker trajectory (blue) both before and after the ball is released (red marker). Still there is a small difference in the trajectory angle visible in both images as can be seen by both the estimated ball marker (green) and estimated center point for the ball.

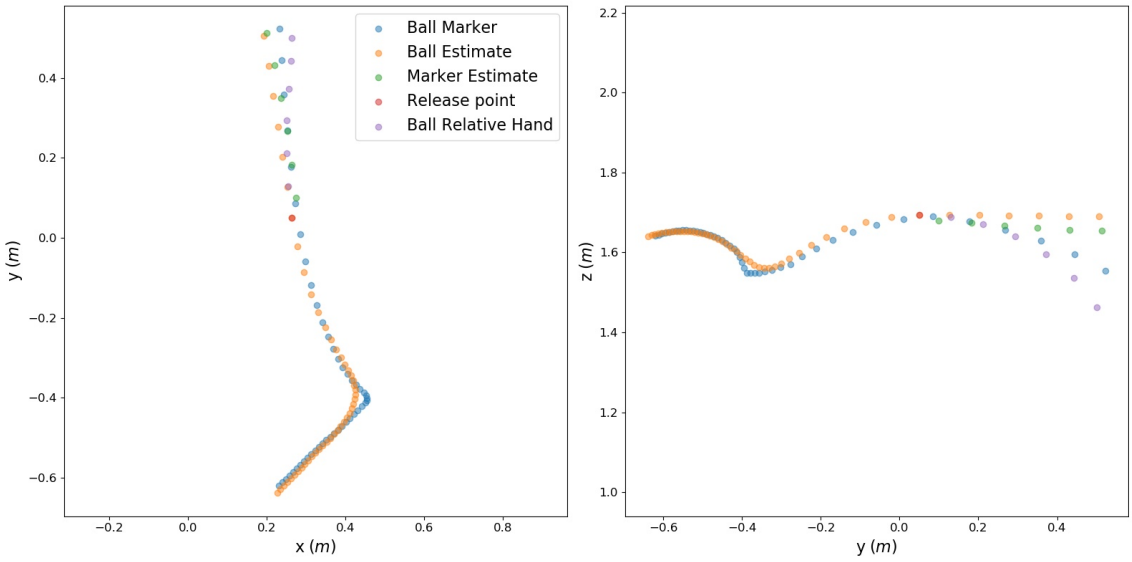


Figure A.2: **Good fit - Poor Trajectory**; In this sequence the wrist motion is again able to fit well to the recorded motion of the ball marker (blue). However this does not result in a trajectory estimate adapted to the ball marker after being released (red marker), the effect is clearly visible in the right image as the ball marker has a greater downward trajectory compared to the ball estimate (orange). Determining why such cases occur is important as it indicates possible improvements to the implemented approach; one reason could be that the true release occur after the initial estimate in the original data set, or that the speed and direction of the velocity vector for the ball in the hand do not match the original motion. In such cases, there is no good fit available for the optimization to find.

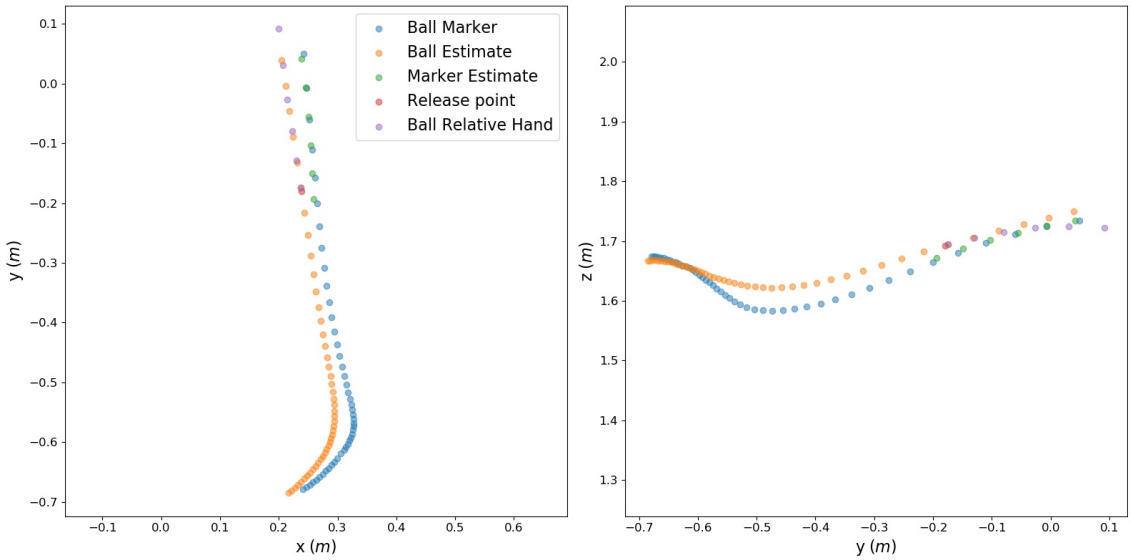


Figure A.3: **Poor fit - Similar Trajectory**; Trajectory for the reconstructed ball (orange) does not follow the ball marker (blue) as the reconstruction is unable to adjust well to the wrist rotation occurring in the initial stage of the throw. Due to the straight trajectory before release, this do not prevent the optimization from aligning of the wrist motion before the ball is released, and the provide a good outcome regarding the ball's trajectory in flight. However, the sequence is also a common case where the number of data points available for the ball after being released is limited.

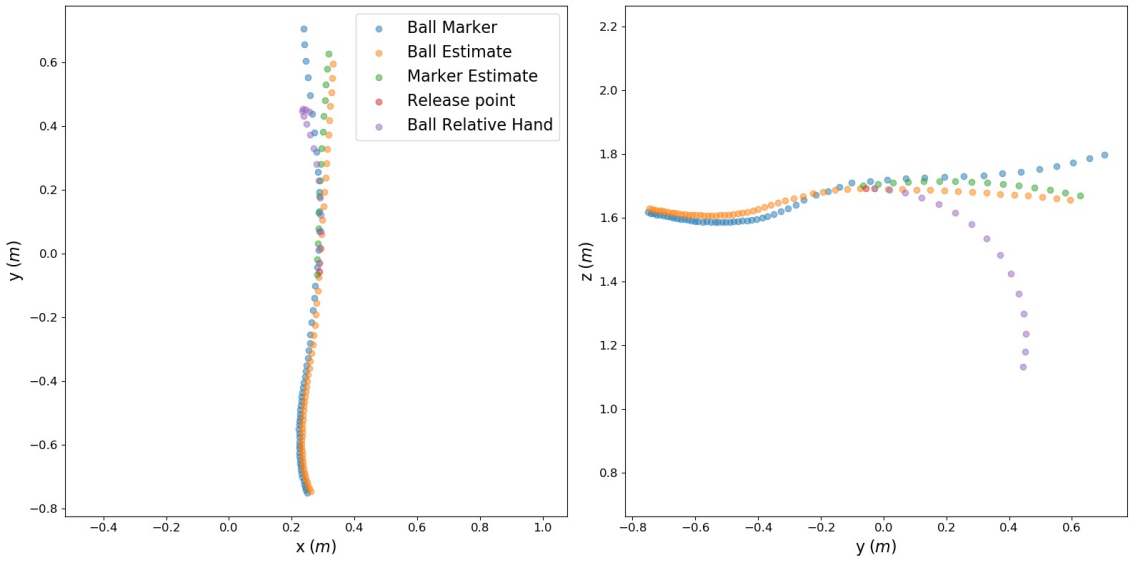


Figure A.4: **Poor fit - Poor Trajectory**; A subtle case where the reconstructed trajectory for the ball (orange) has a similar trajectory to the marker (blue). However, as the reconstructed ball follow a flatter trajectory, as seen in the right image, there is no upward momentum for the optimization to fit to in comparison with the sampled trajectory for the ball. In such cases, improving the MKO process, e.g. by implementing and evaluating other algorithms such as Kalman filtering, would be necessary to improve the overall result.

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden