MoCap-Solver: A Neural Solver for Optical Motion Capture Data

KANG CHEN, NetEase Games AI LAB, China
YUPAN WANG, NetEase Games AI LAB, China
SONG-HAI ZHANG, Tsinghua University, China
SEN-ZHE XU, Tsinghua University, China
WEIDONG ZHANG, NetEase Games AI LAB, China
SHI-MIN HU, Tsinghua University, China

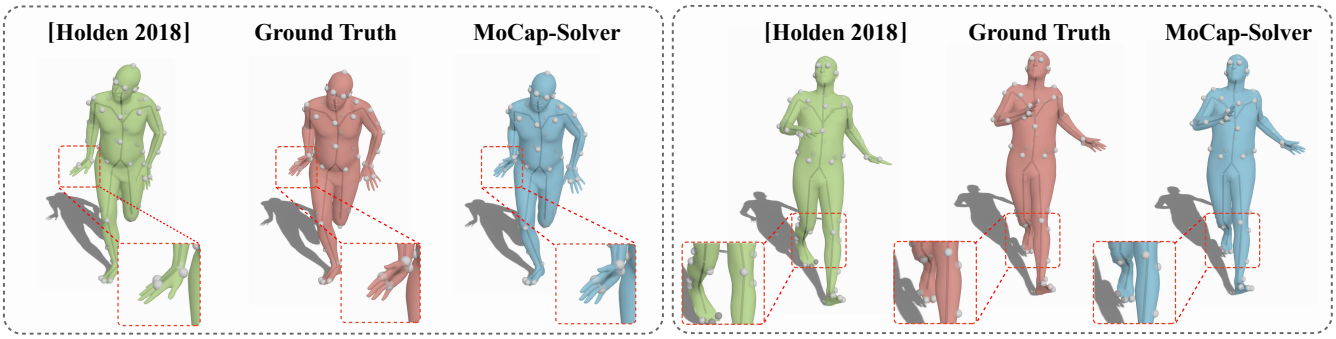

Fig. 1. Data solving results from a real MoCap capture sequence. Relative to the ground truth (red), MoCap-Solver (blue) can achieve more accurate results than the state-of-the-art [Holden 2018] (green).

In a conventional optical motion capture (MoCap) workflow, two processes are needed to turn captured raw marker sequences into correct skeletal animation sequences. Firstly, various tracking errors present in the markers must be fixed (*cleaning* or *refining*). Secondly, an agent skeletal mesh must be prepared for the actor/actress, and used to determine skeleton information from the markers (*re-targeting* or *solving*). The whole process, normally referred to as *solving* MoCap data, is extremely time-consuming, labor-intensive, and usually the most costly part of animation production. Hence, there is a great demand for automated tools in industry. In this work, we present MoCap-Solver, a production-ready neural solver for optical MoCap data. It can directly produce skeleton sequences and clean marker sequences from raw MoCap markers, without any tedious manual operations. To achieve this goal, our key idea is to make use of neural encoders concerning three key intrinsic components: the template skeleton, marker configuration and motion, and to learn to predict these latent vectors from imperfect marker sequences containing noise and errors. By decoding these components from latent vectors, sequences of clean markers and skeletons can be directly recovered. Moreover, we also provide a novel normalization strategy based on learning a pose-dependent marker reliability function, which greatly improves system robustness. Experimental results demonstrate that our algorithm consistently outperforms the state-of-the-art on both synthetic and real-world datasets.

CCS Concepts: • **Computing methodologies** → **Motion capture**; **Motion processing**.

Additional Key Words and Phrases: Optical Motion Capture, Mocap Marker Cleaning, MoCap Solving

ACM Reference Format:
Kang Chen, Yupan Wang, Song-Hai Zhang, Sen-Zhe Xu, Weidong Zhang, and Shi-Min Hu. 2021. MoCap-Solver: A Neural Solver for Optical Motion Capture Data. *ACM Trans. Graph.* 40, 4, Article 84 (August 2021), 11 pages. https://doi.org/10.1145/3450626.3459681

1 INTRODUCTION

Motion capture (MoCap) is the process of recording human motions and is widely used to produce realistic physical movements of virtual characters. MoCap systems typically belong to one of two categories, according to the type of sensors, inertial or optical. Inertial MoCap systems use attached inertial measurement units to detect movements of each marker relative to a base position, while optical MoCap systems use calibrated multi-view infrared sensors to track and calculate the absolute positions of markers in 3D space. Although inertial systems are more affordable, they lack accuracy and absolute positioning, limiting their use in applications where a high level of precision is required. Thus, optical MoCap systems are more common in the game and film industries.

Authors' addresses: Kang Chen, NetEase Games AI LAB, Hangzhou, China, chenkangnobel@gmail.com; Yupan Wang, NetEase Games AI LAB, Hangzhou, China, wangyupan@corp.netease.com; Song-Hai Zhang, Tsinghua University, Beijing, China, shz@tsinghua.edu.cn; Sen-Zhe Xu, Tsinghua University, Beijing, China, xsz15@mails.tsinghua.edu.cn; Weidong Zhang, NetEase Games AI LAB, Hangzhou, China, zhangweidong02@corp.netease.com; Shi-Min Hu, Tsinghua University, Beijing, China, shimin@tsinghua.edu.cn.

0730-0301/2021/8-ART84 $15.00
https://doi.org/10.1145/3450626.3459681

However, markers captured by optical senors inevitably contain positional noise and tracking errors (e.g. missing or misidentified markers). In a conventional workflow, artists have to manually fix these issue, which typically requires a huge amount of time and effort [Holden 2018; Perepichka et al. 2019]. Furthermore, the precision of marker configuration information also plays an important role in accurately retrieving skeletons from markers, so a template agent skeletal mesh has to be carefully set up and fitted to each actor or actress, placing another burden on MoCap artists. Thus, automated tools that can relieve the manual workload are in high demand.

Although the aforementioned tedious manual tasks seem to be repetitive and mechanical, carrying out these calculations on MoCap data is actually an ill-posed task. In an optical MoCap system, each captured marker corresponds to a physical marker pinned on the MoCap suit. Hence, true positions of markers in the 3D space depends on three types of intrinsic information: the actor's body shape, marker distribution on the suit, and the actor's pose. Successfully retrieving a skeleton and clean marker information relies on correctly recovering these three intrinsic components from raw markers, which is obviously an impossible task without prior knowledge of human shape, pose and kinetics.

To incorporate this required prior knowledge, existing methods either use hard-coded empirical rules, or adopt statistical analysis to mine it from a database. Holden's work [2018] is a state-of-the-art solution to this problem. It trains a simple neural network (i.e. a fully-connected backbone with residual connections) that predicts skeletal joint information from raw markers pose-by-pose. However, two main drawbacks of Holden's method severely limit its use in practical production environments. Firstly, the precision of the network is highly sensitive to noise and errors present at several key reference markers around the torso, which often leads to unstable results for real-world data. Secondly, the per-pose framework tends to break temporal continuity and shape consistency of captured actions. Although visual smoothness can be achieved with a Savitzky-Golay post filtering process, fidelity of motions is also damaged, a serious disadvantage in production environments where a high degree of accuracy is required (see Fig. 1).

To address these issues, we propose a new neural solver for this problem (Fig. 2). Starting from a large MoCap database, we first train neural encoders for three key intrinsic components: template skeleton, marker configuration and motion. Then, instead of directly inferring clean markers or skeletons, we train a deep neural network to predict these encoded latent vectors from raw markers, and recover clean markers or skeletons by applying linear blend skinning (LBS) to the decoded components. We demonstrate that this approach achieves higher precision and better temporal continuity than Holden's end-to-end per-pose solution. Furthermore, to eliminate both issues caused by root transformations and at the same time avoid over-reliance on the stability of key reference markers, we propose a novel pose normalization strategy incorporating a pose-dependent marker reliability function learned from real-world MoCap data. Experimental results show that this strategy greatly improves system robustness.

The contributions of this paper can be briefly summarized as:

- A deep learning based framework to accurately reconstruct sequences of clean markers and skeletons from raw markers by explicitly exploring the intrinsic relationships between marker positions, actor body-shapes, marker distributions and motions.
- A novel normalization strategy incorporating a pose-dependent marker reliability function learned from real-world MoCap data, which successfully avoids over-reliance on specific markers and greatly improves the algorithm's robustness.
- A production-ready MoCap data solution which consistently outperforms the state-of-the-art.

2 RELATED WORK

In this section, we review prior work in related areas.

2.1 MoCap Data Cleaning and Solving

To accurately recover skeletal motions from imperfect marker data, numerous methods, respecting prior beliefs of various kinds, have been proposed. According to the kind of priors, existing methods can be considered as evidence-driven or data-driven.

Evidence-driven methods typically build upon empirical rules derived from human kinematics, e.g. shape consistency, pose legality, or spatiotemporal continuity. The simplest observations are that throughout the reconstructed motion sequence, bone joints should preserve relative positions to corresponding markers, bone lengths should remain constant and bone joint angles should stay within a reasonable range. Based on these observations, Herda et al. [2000] use a fixed template skeleton to track and reconstruct the positions of clean markers, Zordan et al. [2003] map raw markers to a chosen template skeleton with fixed limb-length and perform physical simulation to avoid abnormal poses, Kirk et al. [2005] present a nonlinear optimization framework to automatically estimate from markers underlying skeletons obeying all distance constraints, and Hornung et al. [2005] improve the robustness of estimated skeletons by recognizing and removing unreliable markers based on local rigidity. Recently, a new evidence-driven method taking the self-similarity of human motions into account was introduced by Aristidou et al. [2018]. All of these methods are designed to work with specific types of noise or errors present in the input marker positions, and tend to produce broken or over-smoothed results when faced with more complex situations.

Another assumption often adopted by evidence-driven methods is the temporal smoothness of markers, based on which, two families of formulations have been proposed: linear dynamical systems [Aristidou and Lasenby 2013; Dorfmüller-Ulhaas 2007; Li et al. 2010], and low-rank matrix completion [Feng et al. 2015, 2014; Lai et al. 2011; Liu et al. 2014; Park and Hodgins 2006]. Methods based on linear dynamical systems (also known as Kalman filters) attempt to predict motions from noisy marker observations and assume that motions can be approximated by a linear Gaussian model. Alternatively, low-rank matrix completion approaches focus on the low-rank property of MoCap data, and cast the problem as motion reconstruction subject to low-rank matrix completion. Burke and Lasenby [2016] introduce a framework which combines these two types of method. These formulations all build upon solid theoretical foundations, and

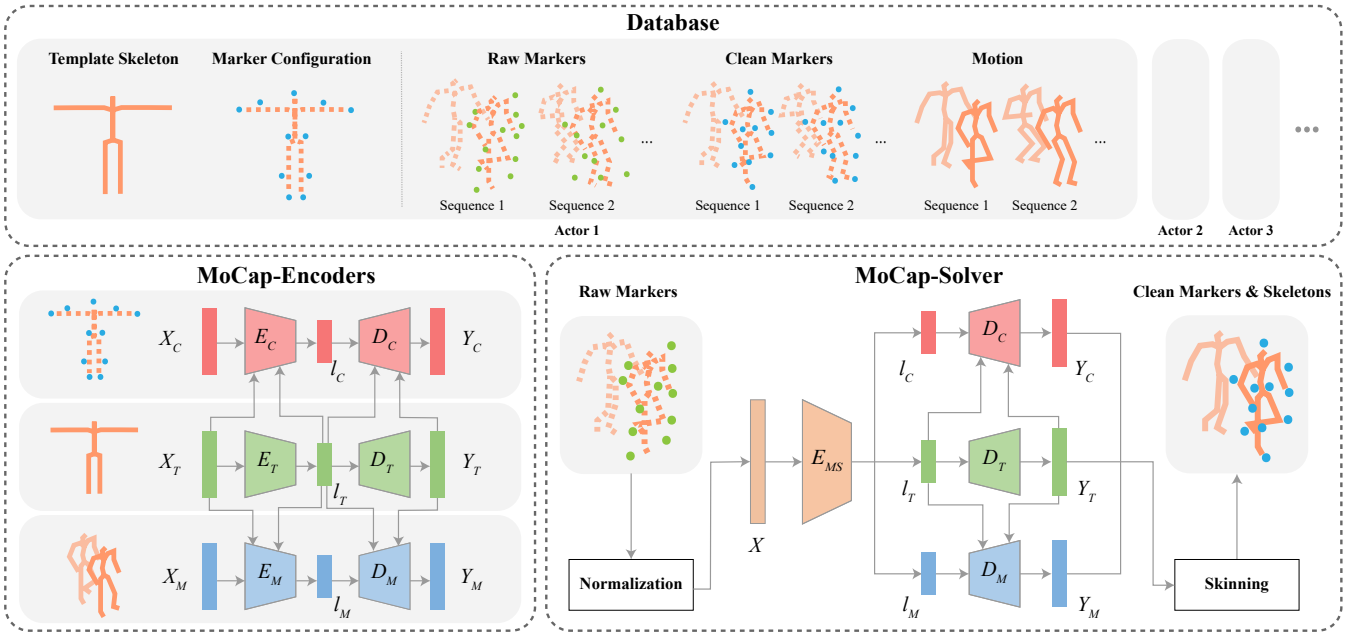

Fig. 2. Overview of our framework. Above: database collected to train our neural networks. Below: the two components of our framework, MoCap-Encoders and MoCap-Solver. The former comprises three autoencoders for the three intrinsic components of MoCap data. The latter maps from raw markers to the latent codes of the MoCap-Encoders to recover clean markers for use by a linear skinning function.

are guaranteed to produce optimal results if the assumptions about data and noise distributions hold, but conversely, may generate unrealistic results in complex circumstances where the assumptions do not hold. As pointed out by Li et al. [2010], certain basic distance constraints required by physics may not be preserved in the results.

Unlike evidence-driven methods which directly build upon explicit prior observations, data-driven approaches acquire prior knowledge from data. Specifically, a low-dimensional embedding is learned from a set of existing MoCap data, where each embedding vector corresponds to a valid pose or a motion segment. By encoding input markers into and decoding from the embedding space, errors in the markers are automatically corrected. Linear space embedding techniques, such as local PCA [Baumann et al. 2011; Chai and Hodgins 2005; Liu and McMillan 2006; Tautges et al. 2011] or sparse encoding [Wang et al. 2016; Xiao et al. 2015], are the basis for most solutions of this kind. A similar bilinear basis model factorizing temporal and spatial components is introduced by Akhter et al. [Akhter et al. 2012]. With the rise of deep learning techniques, deep neural networks have also been utilized to learn manifolds or latent spaces of human motions [Holden et al. 2015; Taylor et al. 2006] spanned by a non-linear basis. In a recent application by Pavllo et al. [2019], an auto-encoder is trained to recover hand motions from markers subject to two-handed mutual occlusion. Essentially, our algorithm shares the same spirit with existing data-driven methods. Specifically, we introduce a novel neural auto-encoder based framework which learns to decompose a sequence of markers into latent vectors for three intrinsic components: the template skeleton, marker configuration, and motion. As these are decisive factors contributing to the 3D positions of MoCap markers, clean markers can be directly reconstructed. We believe this is the first deep learning based framework fully addressing the relationships between captured full-body markers and underlying human shape, marker configuration and motion. We will demonstrate that our factorization significantly facilitates cleaning and solving of (determination of skeleton information from) MoCap data. Compared to existing linear basis models, the use of deep neural networks greatly improves the orthogonality and representation ability of the vectors in the learned latent space.

Holden's state-of-the-art work [2018] also uses a data-driven approach for automatically solving optical MoCap data. It tackles the problem by training a simple end-to-end neural network (i.e. a fully-connected backbone with residual connections) to predict skeletal joint information from raw markers pose-by-pose. Obviously, such a per-pose processing framework tends to lack temporal continuity and shape consistency of motions. Even the length of each bone in the predicted skeleton sequences varies slightly. To maintain visual smoothness, Holden applies a Savitzky-Golay filter to the predicted poses, and as a consequence, the fidelity of the captured data is damaged, as this post-processing operation destroys high-frequency action details. This drawback severly limits its usage in production environments where a high degree of accuracy is required. On the contrary, our framework provides excellent motion precision, spatio-temporal continuity and shape consistency by exploiting intrinsic priors encoded in the corresponding latent vectors. Moreover, Holden adopts a rigid-body registration algorithm to align poses in a local reference frame, leading to a lack of robustness, as its precision is highly sensitive to corruption in several chosen

markers, posing another drawback for practical usage. To prevent over-reliance on a small number of key markers and thus improve system robustness, we learn a pose-dependent marker reliability function from real-world MoCap data, and automatically pick the most reliable frame in the sequence for alignment. Experimental results demonstrate that our algorithm consistently outperforms Holden's method on both synthetic and real-world data.

2.2 Deep 3D Skeletal Motion Representation

A sound representation of 3D skeletal motions is of vital importance to our system. Despite various traditional solutions, deep learning based methods play a dominant role in this area. As skeletal motions are temporal sequences recording transformations of a predefined set of joints, sequential models can be naturally adapted to cope with motion data, for instance, conditional restricted Boltzmann machines [Taylor et al. 2006], temporal convolutional networks [Holden et al. 2016, 2015; Kaufmann et al. 2020], recurrent neural networks [Fragkiadaki et al. 2015; Harvey et al. 2020; Henter et al. 2020; Lee et al. 2018; Li et al. 2019, 2020; Mall et al. 2017], and spatio-temporal graph convolutional networks (GCN) [Aberman et al. 2020; Yan et al. 2018]. Since an articulated human skeleton can be organized as a graph, GCN-based methods prove more suited to skeletal motions as they not only capture temporal movements but also account for the underlying skeleton structure. Apart from network architectures, another notable difference between these methods concerns their descriptions of joint transformations [Pavllo et al. 2020]. Due to the constant bone-length constraints, a pose can either be described by joint rotations or joint positions. Forward and inverse kinematics can be used for the conversion. However, as inverse kinematics problems may have multiple solutions, in applications where joint rotations are required for driving a skinned mesh, rotational representation is to be preferred.

For these reasons, our backbone structure for learning the motion embedding is based upon the skeleton-aware graph convolutional network proposed by Aberman et al. [2020], a state-of-the-art network designed for motion re-targeting. However, unlike the problem of re-targeting, static template skeletons are unknown information which needs to be determined in our application. Hence, we feed the decoded static skeleton, instead of the ground-truth, to the motion decoder, in a major difference from [Aberman et al. 2020].

2.3 Parametric Human Models

Although addressing a different problem, our decomposition framework shares the same spirit as research into parametric human models, such as SMPL [Loper et al. 2015] and its subsequent extensions [Pavlakos et al. 2019; Romero et al. 2017]. Specifically, SMPL characterizes shape and pose variations of natural human bodies using a skinned vertex-based model, where the shape space is represented as vertex positions of each mesh in a rest pose while the pose space is described by angle-axis rotations of body joints. To some extent, SMPL can also be used to acquire shape and pose information from MoCap markers, as each marker physically corresponds to a point location on the mesh surface. In this way, Mahmood et al. [2019] create a dataset of human meshes from MoCap data. However, such frameworks are quite sensitive to noise and errors in the markers, and thus inapplicable in our scenario where the primary focus is the precision of reconstructed motions rather than mesh shapes. From a technical perspective, our framework can be viewed as a skinned marker-based parametric human model. As markers correspond to a subset of vertices in a skinned mesh, the same MoCap actor may be provided with different marker configurations, which differentiates our framework from existing vertex-based parametric human models.

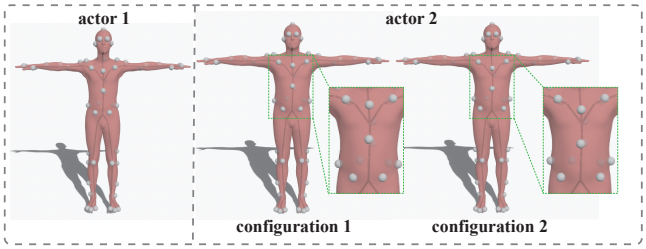

Fig. 3. Marker positions. Actors 1 and 2 have different body shapes, and their marker positions are different. Even for a fixed actor (2), different MoCap suits can have different marker locations.

3 METHOD

In this section, we provide detailed descriptions of our methods. We first give the formulation of the MoCap problem and introduce concepts that are related to this problem (Section 3.1). The key idea of our solution is to explore the compact embeddings of three key intrinsic components using MoCap-Encoders (Section 3.2) and to predict the latent codes of these key intrinsic components using MoCap-Solver (Section 3.3). Finally, the data normalization procedures used before feeding data to the MoCap-Solver are described (Section 3.4).

3.1 Problem

The problem we focus on is retrieving the skeleton and clean markers from raw markers that contain positional noise and missing or mislabeled markers. Given a sequence of raw marker data $X \in \mathbb{R}^{t\times N\times 3}$ in a temporal window of t frames, the question is how to get clean markers $Y \in \mathbb{R}^{t\times N\times 3}$ and skeleton transformations from these raw markers. This is an ill-posed problem. The true positions of the markers in 3D space depend on three types of intrinsic information: the actor's body shape, marker distributions on the suit, and the actor's pose. As shown in Fig. 3, the body shapes of actors differ, so marker positions differ from actor to actor. Even for the same actor, since markers on the capture suit are manually pinned in a setup stage, the positions may differ for different capture sessions.

Thus, we next consider three intrinsic features corresponding to the actor's body shape, marker distributions and the actor's pose respectively. The *marker configuration* $Y_C \in \mathbb{R}^{N\times J\times 3}$ corresponds to the marker distribution on the suit used for optical MoCap. The marker configuration Y_C is defined using the local offset from each marker to each joint. The *template skeleton* $Y_T \in \mathbb{R}^{J\times 3}$ corresponds to the body shape of the actor; it consists of a set of offsets of each joint relative to its parent joint, with the actor in a T-pose. The *motion* $Y_M \in \mathbb{R}^{t\times(J\times 4+3)}$ corresponds to the actor's pose; it

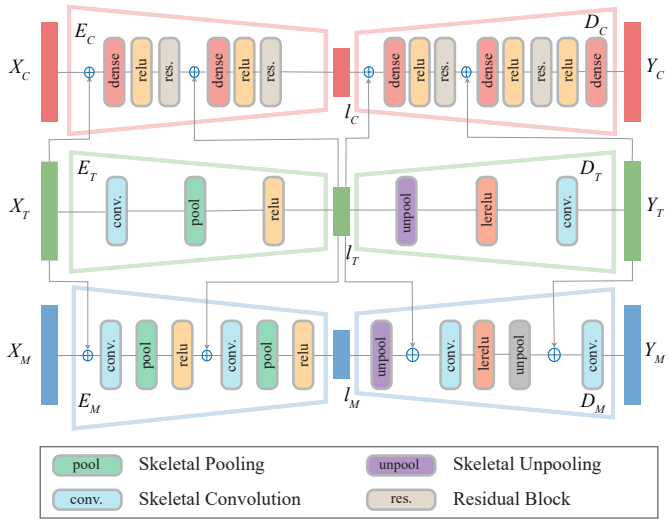

Fig. 4. MoCap-Encoder architecture. It consists of neural encoders for the three key intrinsic components: template skeleton, marker configuration and motion.

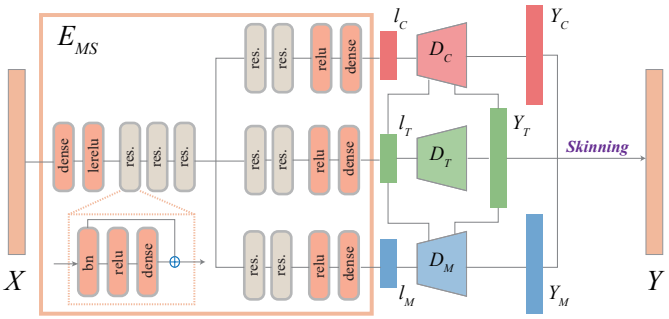

Fig. 5. MoCap-Solver architecture.

consists of a temporal sequence of local rotations of each joint relative to its parent's coordinate frame in the kinematic chain, and the global translation of the root joint. Rotations are represented by quaternions so have 4 elements. The rotation of the root joint represents the global rotation of the motion. The global translation is represented by the global coordinate system.

Given marker configuration Y_C, skeleton template Y_T and motion Y_M, we can reconstruct the marker positions Y by use of a linear blending function (LBS). Finally, we may now state the problem as: given the raw markers X for a sequence, how can we decompose X into the three intrinsic components: marker configuration Y_C, skeleton template Y_T and motion Y_M. We propose a neural network method to solve this problem.

In order to train our neutral networks, we have collected a high-quality extensive MoCap dataset. It contains multiple characters, each having its own template skeleton, marker configuration and multiple sequences that contain raw markers, clean markers and corresponding motion data.

3.2 MoCap-Encoders

Our MoCap-Encoders are neural encoders for the three key intrinsic components: template skeleton, marker configuration and motion. Their purpose is to exploit compact embeddings of each of these components. The structure of the MoCap-Encoders unit is shown in Fig. 4. It contains three sub-encoders: the marker configuration autoencoder (above), the template skeleton autoencoder (middle), and the motion autoencoder (below).

The basic observations are that motion and marker configuration are defined with respect to a template skeleton, i.e. relative joint rotations and relative marker offsets respectively, while the template skeleton is independent of motion and marker configuration. Thus, the skeleton branch is fed to the marker configuration and motion branches. The *template skeleton autoencoder* $[E_T, D_T]$ and the *motion autoencoder* $[E_M, D_M]$ follows the static encoder and dynamic encoder in [Aberman et al. 2020] , a DNN-based framework for motion re-targeting between skeletons. We found the skeleton-aware differentiable operators (i.e., skeletal convolution, pooling and unpooling) proposed by Aberman et al. [2020] effective in characterizing skeletal data. However, in re-targeting, the template skeletons are known, but in our application they need to be determined, so we add the template skeleton decoder D_T as the symmetric structure to the encoder E_T. The structure of *marker configuration autoencoder* is inspired by [Holden 2018], which shows that a stack of residual blocks work well with marker data.

In the training phase, we define the losses as follows to train the three autoencoders.

$$\mathcal{L}_T = \mathcal{D}(Y_T, X_T), \tag{3-1}$$

$$\mathcal{L}_M = \beta_1 \mathcal{D}(Y_M, X_M) + \beta_2 \mathcal{D}(FK(Y_M, Y_T), FK(X_M, X_T)), \tag{3-2}$$

$$\mathcal{L}_C = \beta_3 \mathcal{D}(Y_C, X_C) + \beta_4 \mathcal{D}(LBS(Y_C, Y_T), LBS(X_C, X_T)), \tag{3-3}$$

where $\mathcal{D}$ denotes weighted-L_1 loss; the weight distribution will be discussed in Section 4.1. $FK(\cdot,\cdot)$ is the forward kinematic function that computes the global positions of joints given motion and the template skeleton, $LBS(\cdot,\cdot)$ is the linear blend skinning function that computes the global positions of markers given the marker configuration and the template skeleton. Notice that $\mathcal{L}_C$ is only evaluated on the template skeleton (i.e., under a T-pose), because the marker configuration is physically fixed within each capture session, independent to motion.

3.3 MoCap-Solver

Having trained the MoCap-Encoders for the three intrinsic components: marker configuration, template skeleton and motions, our purpose is to map the input raw markers to the corresponding latent codes of the three intrinsic components from the input raw markers, and then decode the latent codes to compute clean markers using a linear blend skinning function. The benefit of our method is that we can predict the spatio-temporal information contained in the latent code of motion.

Thus, the MoCap-Solver is designed to predict the latent codes of marker configuration, template skeleton and motions. The network structure of the MoCap-Solver is shown in Fig. 5. We follow [Holden

2018], using residual blocks to predict the latent codes of the three intrinsic components.

In our neural network, four sub-unit objectives should be optimized: marker positions, marker configurations, template skeletons and motions. Overall, the final objective function is:

$$\begin{aligned}\mathcal{L} =&\alpha_1\mathcal{D}(Y,X) + \alpha_2\mathcal{D}(Y_C,X_C)\\ &+ \alpha_3\mathcal{D}(Y_T,X_T) + \alpha_4\mathcal{D}(Y_M,X_M) + \gamma\|\Psi\|_2,\end{aligned} \tag{3-4}$$

where $\mathcal{D}$ is as above, α_1–α_4 weight the four losses, $\|\Psi\|_2$ represents the l_2 regularization loss and γ is a weight controlling the penalty degree.

3.4 Normalization

In the real world, raw markers contain global translations and rotations that vary greatly, increasing the difficulty of training. Moreover, raw markers often include outliers which hinder convergence of the training phase of the MoCap-Solver. Thus, before feeding raw marker data into the MoCap-Solver, we normalize the raw markers to get rid of outliers and transform them to a local space, accelerating convergence and enhancing the robustness of our algorithm. Our normalization procedure has two phases: outlier detecting and replacement, and rigid registration. This section introduces our novel normalization strategy which incorporates a pose-dependent marker reliability function learned from real-world MoCap data. This successfully avoids over-reliance on certain specific markers and greatly improves the algorithm's robustness.

Inspired by the outlier removal method in [Holden 2018], we also use distance matrix tools to identify poorly positioned markers. A distance matrix is a square matrix recording the pairwise Euclidean distances between a set of markers. From the input raw markers, we extract the distance matrix of each frame, and find the frame whose distance matrix is closest to the mean distance matrix across all frames as the reference frame. By comparing each distance matrix with that of the reference frame, we can detect outliers by setting a threshold to $300mm$. Instead of setting detected outliers to the origin, as in [Holden 2018], we use an outlier replacement method to fill the gap left by the removed outlier. It works by optimizing the distance matrix to be consistent to the distance matrix of the reference frame for the whole sequence.

The raw marker data is in global coordinate system. To reduce the training difficulty, we need to remove the global transformation by re-representing the data in a local space. Similar to [Holden 2018], we take 8 markers surrounding the spine as reference markers, calculate the global rigid transformation to the T-pose using these markers and place all markers into this local space using the calculated rigid transformation matrix. However, in real-world raw marker data, these reference markers are, though much more reliable than other markers, still very likely to contain noise and error. Such errors greatly affect the precision of the network, which is also mentioned in [Holden 2018]. Actually, this is a major drawback of [Holden 2018] in production environments, as we found 4.43% of the frames in a large real MoCap dataset containing obvious problematic reference markers (i.e. if any of the reference markers has an distance error over $300mm$).

Unlike Holden's frame-by-frame solution, our MoCap-Solver takes a sequence of frames as input, making it possible to retrieve rigid transformation base on more frames. The key problem lies in finding frames in a sequence that contain least noise in the reference markers. We observe that the distribution of noise in the reference markers is highly related to the actor's pose. For example, the front-abdomen markers are often occluded when actors bend over. Thus, we make use of a pose-dependent marker reliability function F_{MR} to measure the reliability of the reference markers. We define marker reliability P^i of the i–th reference marker as a function of the distance d_i between the raw reference marker and clean reference marker, as follows, larger P^i indicating greater reliability:

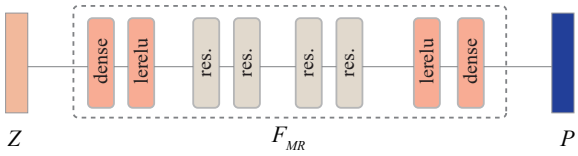

Fig. 6. The network structure of pose-dependent marker reliability function F_{MR}.

$$P^i = \begin{cases} 1 & d_i \le 50mm \\ 1.2 - 0.004d_i & 50mm < d_i < 300mm \\ 0 & d_i \ge 300mm \end{cases} \tag{3-5}$$

The raw markers are the input $Z \in \mathbb{R}^{N\times 3}$ to F_{MR}, and the output consists of the marker reliability of 8 reference markers $P \in [0,1]^8$. We formulate this problem as a regression problem. The network structure of F_{MR} is shown in Fig. 6 and follows the network structure of [Holden 2018]. To train F_{MR}, we collect a real MoCap dataset which contains both raw markers and corresponding clean markers for sequences. Then the marker reliability of the reference markers are computed by Eq. (3-5). The loss function of F_{MR} in the training phase is the cross-entropy loss $\mathcal{D}_{ce}$ of marker reliability:

$$\mathcal{L}_P = \mathcal{D}_{ce}(P,\hat{P}) \tag{3-6}$$

where $\hat{P}$ denotes the ground-truth marker reliability.

After training F_{MR}, given raw markers for a sequence, we compute the marker reliability function F_{MR} for all frames. Frames are identified as reliable if the predicted reliability of all 8 reference markers exceeds 0.8. For each reliable frame, we calculate its rigid transformation matrix to the T-pose. Since in our system the predicting is performed on a temporal window of size t, we need to specify a local reference frame for each input t frames, where if there are more than one reliable frames, we pick the frame with minimum rotation transformation, and if no reliable frames, the local reference frame in the closest previous temporal window is used.

4 EVALUATION

In testing our MoCap-Solver, we consider three things. Firstly, we discuss the training settings of the MoCap-Solver in Section 4.1. Then we focus on analyzing the performance of various components of the MoCap-Solver in Section 4.2, with an ablation study in Section 4.3. Finally we compare MoCap-Solver with [Holden 2018] and

[Mahmood et al. 2019], state-of-the-art MoCap data solving and recovery methods respectively.

4.1 Training Settings

In this section, we introduce the detailed training settings used in our experiments, including database collection and parameter settings.

4.1.1 Database Collection. We need an extensive, high-quality MoCap dataset with different characters, which consists of raw markers and corresponding clean markers and skeleton information. We collected two types of datasets, and trained a version of the MoCap-Solver for each dataset.

The first is the *real MoCap dataset* captured in the real world for character motion production for a game studio. The ground-truth markers were cleaned by hand, the skeleton information was produced using a commercial solver. We collected 10k sequences of MoCap data with a total of around 5m frames containing 81 characters and 125 sets of marker configurations. Two sets of markers configured for the same character are considered as different marker configurations if the distance between any pair of corresponding markers exceeds $10mm$.

The second is the *synthetic MoCap dataset* produced by driving SMPL model using pose parameters from the CMU MoCap dataset [CMU 2000] and shape parameters from the CAESAR dataset [Robinette et al. 2002]. We separately generated three sets of marker configurations for each character by randomly making small perturbations to a standard setup. Sampling the skeleton transformations of the SMPL model and the positions of surface locations specified by marker configurations provided the skeleton information and clean markers. To simulate the raw markers, we adopted the corruption function proposed by Holden [Holden 2018], which involves three parameters controlling the probability of occlusion σ_o, the probability of shifting σ_s and the intensity of shifting σ_i (i.e. β in [Holden 2018]) respectively. In our simulation, we set these parameters to $\sigma_o = 0.1$, $\sigma_s = 0.1$ and $\sigma_i = 0.3m$. We collected 5k sequences of synthetic MoCap data with a total of 8m frames containing 1700 characters and 5100 sets of marker configurations.

4.1.2 Parameter Settings. When training the MoCap-Encoders and MoCap-Solver, we set $\beta_1 : \beta_2 = 1 : 100$ in Eq. (3-2), $\beta_3 : \beta_4 = 1 : 2$ in Eq. (3-3) and $\alpha_1 : \alpha_2 : \alpha_3 : \alpha_4 = 1 : 3 : 4 : 8$ in Eq. (3-4). To weight the smoothed l_1 function $\mathcal{D}$ in Sections 3.2 and 3.3, we divided the human body into seven parts: head, shoulder, arm, wrist, torso, thigh and feet. Markers and joints belonged to each part were separately given a weighting ratio as $8 : 4 : 6 : 10 : 4 : 6 : 10$. The learning rate for MoCap-Encoders was set to 10^{-4} and each branch was trained with 2k epochs. The MoCap-Solver was trained with 2400 epochs with a decaying learning rate starting from 0.01 (decay factor 0.1 for every 600 epochs). The batch size was set to 512 for the MoCap-Encoders and 128 for the MoCap-Solver. The temporal window size t was set to 64 and a 32 frame overlap was adopted to ensure smoothness between adjacent windows; the predicted temporal windows were combined into one sequence by averaging the overlapping areas.

Table 1. Reconstruction errors of the MoCap-Encoders under 10-fold cross-validation.

Models	Synthetic data		Real data	
	Position	Orientation	Position	Orientation
$[E_C, D_C]$	$0.5mm$	-	$0.7mm$	-
$[E_T, D_T]$	$0.3mm$	-	$0.4mm$	-
$[E_M, D_M]$	$1.8mm$	$1.0°$	$1.9mm$	$1.3°$

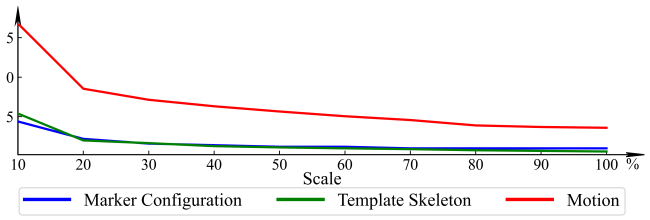

Fig. 7. Reconstruction error versus scale of training data of synthesis dataset for the MoCap-Encoders. The curves show marker position error (unit: mm) reconstructed by marker configuration (blue), skeleton position error (unit: mm) reconstructed by template skeleton (green) and the skeleton position error (unit: mm) reconstructed by motion (red).

4.2 Performance Analysis.

We analyzed the performance of each component of the MoCap-Solver. Firstly, we evaluated the the MoCap-Encoders in terms of reconstruction error and generalization. Then a comparison was made of precision achieved using different temporal window sizes. The precision is measured by the prediction errors in joints and markers, specifically, joint position error (JPE), joint orientation error (JOE) and marker position error (MPE).

4.2.1 MoCap-Encoders. The MoCap-Encoders include three autoencoders: the marker configuration encoder $[E_C, D_C]$, the template skeleton encoder $[E_T, D_T]$ and the motion encoder $[E_M, D_M]$. We first evaluated the reconstruction errors on both synthetic dataset and real MoCap dataset. Results produced under 10-fold cross-validation are given in Table 1. The positional and rotational errors were computed as mean Euclidean distances across all skeleton joints and markers, represented in millimeters and degrees respectively. We can see that the reconstruction errors are small enough to be useful.

To evaluate the generalizability performance of the MoCap-Encoders, we randomly split the *synthetic MoCap dataset* into a training set (90%) and a testing set (10%). Then, we randomly sampled the training data, choosing 10%, ..., 100% of it to train the MoCap-Encoders and calculated the reconstruction error on the testing set. The results are shown in Fig. 7. As the scale of the training dataset increases, the reconstruction error decreases correspondingly, finally converging to stable results. The marker configuration encoder and template skeleton encoder achieve stable results using 50% of the training data; the motion encoder needs 80% to achieve a stable result.

4.2.2 Temporal Window Size. The temporal window size can potentially influence the performance of MoCap-Solver. Particularly, a motion encoder with a larger window can encode more temporal

Table 2. Precision versus window size for the synthetic MoCap test dataset.

window size	JPE	JOE	MPE
16	9.97*mm*	3.73°	10.51*mm*
32	9.53*mm*	3.43°	10.12*mm*
64	9.48*mm*	3.19°	10.03*mm*
128	14.96*mm*	4.78°	15.51*mm*

information and improve prediction precision. However, a large window size will also increase the complexity of the motion encoder and thus require more data to train this encoder, tending to decrease the precision of the MoCap-Solver. We trained four versions of the MoCap-Encoder with window sizes 16, 32, 64 and 128, using the synthetic training dataset (90%), and compared the precision achieved on the test dataset (90%). The result (see Table 2) shows little variation with window size at first, but the size of 128 decreases the precision because of the above reasons. Therefore, a window size of 64 is chosen in our system.

4.3 Ablation Study

We next evaluated the effectiveness of the MoCap-Encoders and the normalization strategy on the real MoCap dataset.

4.3.1 MoCap-Encoders. The advantage of the MoCap-Encoders is to learn compact embeddings of template skeleton, marker configuration and motion, helping the MoCap-Solver to converge and improving its prediction accuracy. To evaluate the advantage of the MoCap-Encoders, we trained an end-to-end network bypassing MoCap-Encoders on the real MoCap dataset and compared prediction errors for the test data. The results are shown in Table 3. Compared to an MoCap-Solver without MoCap-Encoders, using MoCap-Encoders reduces 2*mm* positional errors of markers and joints, and 1.5° rotational errors of joints.

4.3.2 Normalization Strategy. Our normalization procedure involves two phases: outlier detecting and replacement, and rigid registration. To quantitatively evaluate its effectiveness, we trained/tested our MoCap-Solver on the real MoCap dataset with different strategy selections in each phase. Specifically, three strategies in outlier detecting and replacement, i.e., no outlier handling (O_0), setting detected outlier to origin (O_1) and our outlier replacement method (O_2), and three strategies in rigid registration. i.e., no rigid registration (R_0), aligning to the first frame (R_1), and our method which employs a pose-dependent marker reliability function F_{MR} to select the best frame for registration (R_2). The precision of MoCap-Solver with different normalization strategies are listed in Table 4.

The statistics show that normalization is of vital importance to this problem; significant increases in prediction errors can be found in MoCap-Solver without outlier handling (O_0) or rigid registration (R_0). Besides, our outlier replacement method (O_2) contributes greatly to the final precision compared with the zero-filling strategy (O_1) in [Holden 2018]. Actually, [Holden 2018] also benefits from such strategy, i.e., around 3.5*mm* reduces in marker and joint positional errors, and a 1.15° reduce in joint rotational errors can be achieved on the real MoCap dataset. To make the comparisons fair, all experiments in this paper were conducted with the same

Table 3. Precision of various methods for real MoCap dataset.

Models	JPE	JOE	MPE
[Holden 2018]	18.68*mm*	7.94°	19.88*mm*
[Mahmood et al. 2019]	60.00*mm*	21.90°	58.10*mm*
MoCap-Solver	9.23*mm*	3.78°	10.03*mm*
MoCap-Solver (w/o MoCap-Encoders)	11.44*mm*	5.22°	12.02*mm*

Table 4. Precision of MoCap-Solver with different normalization strategies for real MoCap dataset.

Normalization	JPE	JOE	MPE
O_0 and R_2	23.36*mm*	10.22°	25.30*mm*
O_1 and R_2	16.28*mm*	7.23°	17.14*mm*
O_2 and R_2	9.23*mm*	3.78°	10.03*mm*
O_2 and R_1	12.77*mm*	6.34°	13.60*mm*
O_2 and R_0	19.35*mm*	8.68°	20.97*mm*

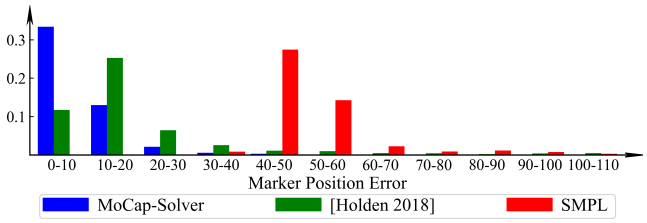

Fig. 8. Comparison of marker position error distributions for MoCap-Solver (blue), [Holden 2018] (green) and the SMPL method [Mahmood et al. 2019] (red) for test data of the real MoCap dataset. Horizontal axis stands for marker position error (unit:*mm*), and the vertical axis stands for the proportion of predicted frames.

outlier detecting and replacement strategy (O_2) for [Holden 2018] and MoCap-Solver. Moreover, our pose-dependent marker reliability function also proves to be effective, as using F_{MR} (O_2 and R_2) reduces over 3.5*mm* of marker and joint positional errors and 2.56° of joint rotational errors relative to MoCap-Solver without F_{MR} (O_2 and R_1). In our implementation, F_{MR} was trained on the real MoCap data, and its mean prediction error of all reference markers, under 10-fold cross-validation, is 0.0047.

4.4 Comparison

We next compared MoCap-Solver with Holden's state-of-the-art MoCap data solving method [Holden 2018] and the SMPL fitting approach [Mahmood et al. 2019], in 5 ways: precision, robustness to reference marker noise, shape variation, generalizability and speed.

4.4.1 Precision. We used the test data from the real MoCap dataset to evaluate the prediction errors of [Holden 2018], [Mahmood et al. 2019] and MoCap-Solver. See Table 3. A statistical analysis of marker position error distributions of the prediction results for this data is presented as a histogram in Fig. 8. As SMPL is a pre-trained model which may not be fully consistent with actors' shape distribution in our real MoCap dataset, the prediction error of [Mahmood et al.

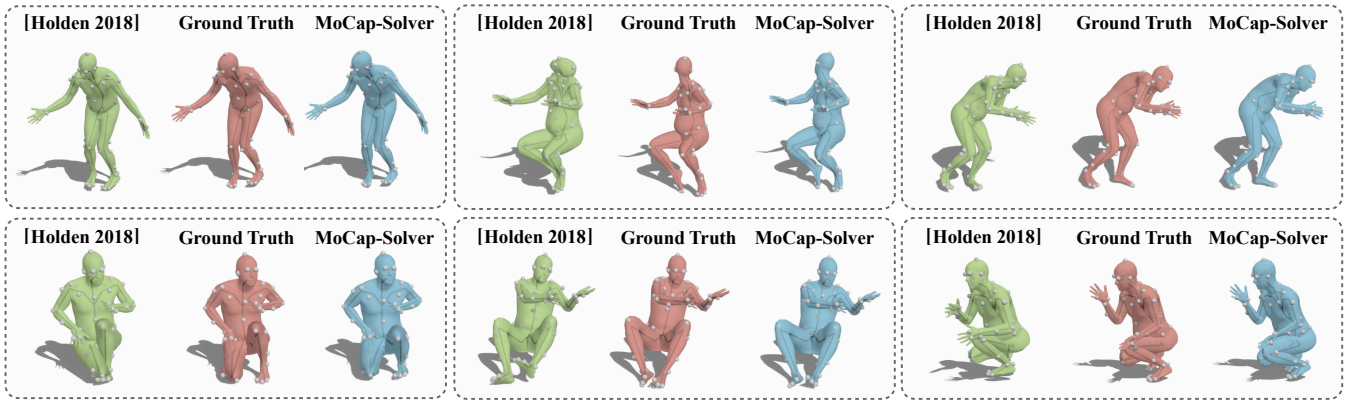

Fig. 9. Comparison of prediction results for the real MoCap dataset. Green: Holden [Holden 2018]. Blue: MoCap-Solver. Red: ground-truth. Orientations in our results are more accurate than Holden's for head, feet, and arms when walking, running, sitting and squatting.

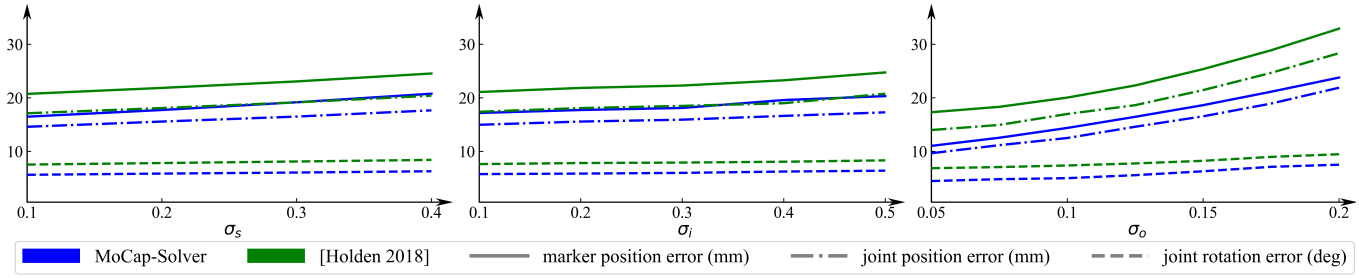

Fig. 10. Comparison of prediction errors of synthetic MoCap test dataset for different noise intensities on non-reference markers. Green: prediction errors for [Holden 2018]. Blue: prediction errors for MoCap-Solver. Left: prediction errors for shifting probability σ_s from 0.1 to 0.4 with $\sigma_o = 0.1$, $\sigma_i = 0.2m$. Center: prediction errors for shifting intensity σ_i from $0.1m$ to $0.5m$ with $\sigma_o = 0.1$, $\sigma_s = 0.2$. Right: prediction errors for occlusion probability σ_o from 0.05 to 0.2 with $\sigma_s = 0.2$, $\sigma_i = 0.2m$.

2019] is much higher than the other two. With the help of MoCap-Encoders and pose-dependent marker reliability function, MoCap-Solver clearly achieves higher precision, better motion fidelity, and far less abnormal prediction results, compared with [Holden 2018]. In practice, unacceptable visual artifacts can be found in predicted frames whose mean marker error is over $30mm$. Fig. 9 illustrates some of these cases. Joint errors of various body parts can be found in the poses generated by [Holden 2018], while MoCap-Solver performs much better. Generally, there are around 10% of the frames predicted by [Holden 2018] with mean marker error over $30mm$, while the number for MoCap-Solver is 1%.

To make an in-depth quantitative comparison, we tested the performances of MoCap-Solver and [Holden 2018] over different intensities of simulated noise on the synthetic test dataset. In this experiment, reference marker were not corrupted. By varying each control parameter (with the other two fixed) in the corruption function on non-reference markers, we can plot the prediction error curves over the intensities of different types of noise and errors. The results showing in Fig. 10 demonstrate that MoCap-Solver consistently outperforms [Holden 2018].

4.4.2 Robustness to Corruption on Reference Markers. As mentioned in [Holden 2018], the prediction of the network is significantly affected by the reliability of several reference markers used for rigid registration. As we found 4.43% of the frames in the real MoCap dataset contain obvious problematic reference markers, the over-reliance on reference markers is apparently a major drawback in production environments. With a learned pose-dependent marker reliability function, our framework is much more robust to corruptions happened in reference markers. To demonstrate this, we conducted an experiment using the synthetic dataset. Specifically, we set noise simulation parameters $\sigma_i = 0.2m$ and $\sigma_o = 0.1$ for all markers, fixed shifting probability $\sigma_s = 0.1$ for non-reference markers, and then varied σ_s from 0 to 0.25 for reference markers. The accuracy of MoCap-Solver and [Holden 2018] is shown in Fig. 11(a); the prediction errors of [Holden 2018] increases drastically when the shifting probability of reference markers is over 0.1, while MoCap-Solver performs much more stable.

4.4.3 Shape Variation. Apart from raw markers, [Holden 2018] also requires the corresponding marker configuration as input. Benefiting from the pre-trained MoCap-Encoders, MoCap-Solver can directly retrieve marker configuration from raw marker sequences which is obviously more convenient to use. Moreover, considering

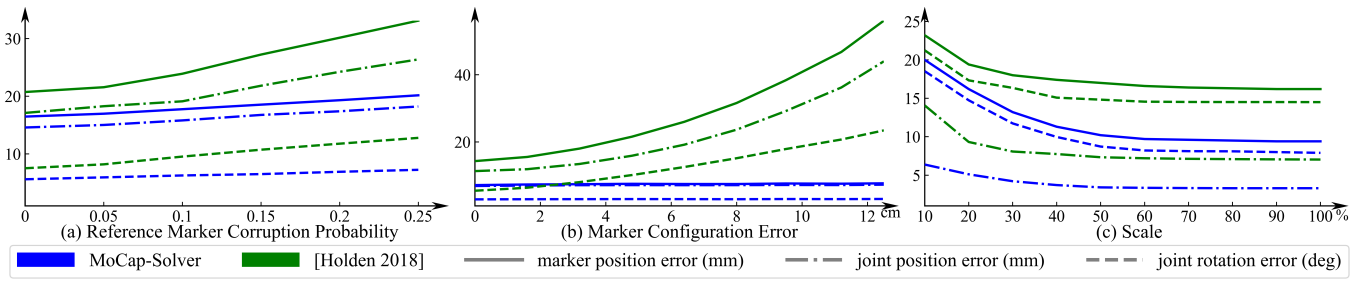

Fig. 11. (a) illustrates the prediction errors of synthetic MoCap test dataset for different noise intensities by varying the probability of shifting reference markers σ_s from 0 to 0.25 while fixing shifting probability $\sigma_s = 0.1$ for non-reference markers and keeping $\sigma_o = 0.1$ and $\sigma_i = 0.2m$ for all markers; (b) illustrates the prediction errors of synthetic data versus the marker configuration errors, the horizontal axis stands for marker configuration error (unit: cm), and the vertical axis stands for prediction errors; (c) illustrates the predicting errors versus scale of training data of synthetic dataset.

Table 5. Precision on real MoCap data after training on synthetic data.

Models	JPE	JOE	MPE
[Holden 2018]	29.15mm	15.43°	36.35mm
MoCap-Solver	12.77mm	6.47°	13.66mm

Table 6. Speed.

Models	Training Time	Prediction Speed
[Mahmood et al. 2019]	–	10 fps
[Holden 2018]	26h	43 fps
MoCap-Encoder	10h	–
MoCap-Solver	40h	34 fps

marker configurations as one of the intrinsic components grants MoCap-Solver with more resistance to noises presented in manually calibrated marker configurations. To demonstrate this, we randomly added small perturbations to marker configurations when generating test motions, and recorded the prediction error versus marker configuration error for both MoCap-Solver and [Holden 2018]. Here, the marker configuration error was calculated as the summed distance of all markers, in centimeters. The results are shown in Fig. 11(b), from which we can see that [Holden 2018] is actually quite sensitive to such error. In our real MoCap dataset, the mean marker configuration variation of the same character is around 2cm to 3cm, enough to affect the precision of [Holden 2018].

4.4.4 Generalizability. As data-driven methods, both MoCap-Solver and [Holden 2018] require a large set of training data with sound coverage of common human poses and shapes. To evaluate the generalizability of MoCap-Solver and [Holden 2018], we made two experiments. First, we recorded the precision of MoCap-Solver and [Holden 2018] trained over different scale of train data (i.e. 10%, ..., 100%) in the synthetic dataset. The results are shown in Fig. 11(c). Both methods can achieve stable performance using 50% of the training data, while MoCap-Solver consistently outperforms [Holden 2018] in terms of precision. Second, we tested the performance of trained models on data from different distributions with the training set. To this end, we trained MoCap-Solver and [Holden 2018] with synthetic MoCap data, and tested with real MoCap data. The results showed in Table 5 imply MoCap-Solver also has much greater generalizability than [Holden 2018].

4.4.5 Speed. To compare the speed of MoCap-Solver and other approaches, we recorded the training time (hours) and the average prediction speed (frames per second) on synthetic MoCap dataset in Table 6. We used a server with a Intel Xeon Gold 6240 CPU and a Nvidia RTX 2080Ti GPU. Since SMPL is a pre-trained parametric model, [Mahmood et al. 2019] does not require training but tends to be slow in prediction. We set the batch size to 512 for both [Holden 2018] and MoCap-Solver, and processed testing data sequence by sequence. Training MoCap-Solver (plus MoCap-Encoders) takes twice as long as the time for [Holden 2018] because it has more complex architecture. Once trained, both [Holden 2018] and MoCap-Solver can predict at over 30 fps, satisfying the speed requirements for production use.

5 LIMITATION AND FUTURE WORK

Like other data-driven methods, MoCap-Solver cannot produce good results for cases totally "unseen" in the training set. Therefore, building a large MoCap dataset covering more diversed human shapes and poses is definitely the next step to facilitate the performance of MoCap-Solver.

Besides, even though the pose-dependent marker reliability function can greatly improve our system robustness to noise in reference markers, MoCap-Solver may still fail to correctly handle frames where all reference markers are occluded. Although with a very low probability, such extreme cases might occur in real MoCap data. In our experiments, the SMPL fitting algorithm [Mahmood et al. 2019] performed much better in handling such extreme cases. In future work, we would like to explore the possibility of incorporating optimization-based approaches like [Mahmood et al. 2019] into our framework, to further promote the system's robustness.

Furthermore, currently our framework can only handle homogeneous skeletons. If the skeleton topology changes, both the MoCap-Encoders and the MoCap-Solver have to be re-trained. In the future, we would like to consider heterogeneous template skeletons in the MoCap-Encoders as well, which will significantly increase the applicability of our system in more production environments.

6 CONCLUSION

MoCap data solving is an indispensable step for motion data processing. In this paper, we have presented a production-ready MoCap data solution framework, MoCap-Solver, which can accurately recover clean markers and skeletons from raw markers by exploring the latent spaces of marker distributions, template skeletons and motions. Moreover, we show how to learn a pose-dependent marker reliability function from real MoCap data to avoid over-reliance on reference markers; this greatly improves the robustness of our framework. Experiments on both synthetic and real MoCap data show that MoCap-Solver consistently outperforms existing methods and achieves results of a quality and at a speed suitable for production use.

ACKNOWLEDGMENTS

We thank the reviewers for their constructive comments and Quantic Dream for providing the real MoCap data. This work was supported by the National Key Technology RD Program (Project Number 2017YFB1002604), the National Natural Science Foundation of China (Project Numbers 61521002, 61772298), and the Research Grant of Beijing Higher Institution Engineering Research Center.

REFERENCES

Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-aware networks for deep motion retargeting. *ACM Trans. Graph.* 39, 4 (2020), 62.

Ijaz Akhter, Tomas Simon, Sohaib Khan, Iain A. Matthews, and Yaser Sheikh. 2012. Bilinear spatiotemporal basis models. *ACM Trans. Graph.* 31, 2 (2012), 17:1–17:12.

Andreas Aristidou, Daniel Cohen-Or, Jessica K. Hodgins, and Ariel Shamir. 2018. Self-similarity analysis for motion capture cleaning. *CGF* 37, 2 (2018), 297–309.

Andreas Aristidou and Joan Lasenby. 2013. Real-time marker prediction and CoR estimation in optical motion capture. *Vis. Comput.* 29, 1 (2013), 7–26.

Jan Baumann, Björn Krüger, Arno Zinke, and Andreas Weber. 2011. Data-driven completion of motion capture data. In *Proc. of VRIPHYS*. 111–118.

Michael Burke and Joan Lasenby. 2016. Estimating missing marker positions using low dimensional Kalman smoothing. *J. Biomechanics* 49, 9 (2016), 1854–1858.

Jinxiang Chai and Jessica K. Hodgins. 2005. Performance animation from low-dimensional control signals. *ACM Trans. Graph.* 24, 3 (2005), 686–696.

CMU. 2000. CMU graphics lab motion capture database. *http://mocap.cs.cmu.edu* (2000).

Klaus Dorfmüller-Ulhaas. 2007. Robust optical user motion tracking using a kalman filter. (2007).

Yinfu Feng, Mingming Ji, Jun Xiao, Xiaosong Yang, Jian J. Zhang, Yueting Zhuang, and Xuelong Li. 2015. Mining spatial-temporal patterns and structural sparsity for human motion data denoising. *IEEE Trans. Cyber.* 45, 12 (2015), 2693–2706.

Yinfu Feng, Jun Xiao, Yueting Zhuang, Xiaosong Yang, Jian J. Zhang, and Rong Song. 2014. Exploiting temporal stability and low-rank structure for motion capture data refinement. *Inf. Sci.* 277 (2014), 777–793.

Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent network models for human dynamics. In *Proc. of ICCV*. 4346–4354.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher J. Pal. 2020. Robust motion in-betweening. *ACM Trans. Graph.* 39, 4 (2020), 60.

Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. 2020. MoGlow: probabilistic and controllable motion synthesis using normalising flows. *ACM Trans. Graph.* 39, 6 (2020), 236:1–236:14.

Lorna Herda, Pascal Fua, Ralf Plänkers, Ronan Boulic, and Daniel Thalmann. 2000. Skeleton-based motion capture for robust reconstruction of human motion. In *Proc. of CA*. IEEE, 77.

Daniel Holden. 2018. Robust solving of optical motion capture data by denoising. *ACM Trans. Graph.* 37, 4 (2018), 165:1–165:12.

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

Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia Technical Briefs*. 18:1–18:4.

Alexander Hornung, Sandip Sar-Dessai, and Leif Kobbelt. 2005. Self-calibrating optical motion tracking for articulated bodies. In *Proc. of VR*. 75–82.

Manuel Kaufmann, Emre Aksan, Jie Song, Fabrizio Pece, Remo Ziegler, and Otmar Hilliges. 2020. Convolutional autoencoders for human motion infilling. *CoRR* (2020).

Adam G. Kirk, James F. O'Brien, and David A. Forsyth. 2005. Skeletal parameter estimation from optical motion capture data. In *Proc. of CVPR*. 782–788.

Ranch Y. Q. Lai, Pong C. Yuen, and Kelvin K. W. Lee. 2011. Motion capture data completion and denoising by singular value thresholding. In *EG Short Papers*. 45–48.

Kyungho Lee, Seyoung Lee, and Jehee Lee. 2018. Interactive character animation by learning multi-objective control. *ACM Trans. Graph.* 37, 6 (2018), 180:1–180:10.

Lei Li, James McCann, Nancy S. Pollard, and Christos Faloutsos. 2010. BoLeRO: A principled technique for including bone length constraints in motion capture occlusion filling. In *Proc. of SCA*. 179–188.

Shujie Li, Yang Zhou, Haisheng Zhu, Wenjun Xie, Yang Zhao, and Xiaoping Liu. 2019. Bidirectional recurrent autoencoder for 3D skeleton motion data refinement. *Comput. Graph.* 81 (2019), 92–103.

Shu-Jie Li, Hai-Sheng Zhu, Liping Zheng, and Lin Li. 2020. A perceptual-based noise-agnostic 3D skeleton motion data refinement network. *IEEE Access* 8 (2020), 52927–52940.

Guodong Liu and Leonard McMillan. 2006. Estimation of missing markers in human motion capture. *Vis. Comput.* 22, 9-11 (2006), 721–728.

Xin Liu, Yiu-ming Cheung, Shu-Juan Peng, Zhen Cui, Bineng Zhong, and Ji-Xiang Du. 2014. Automatic motion capture data denoising via filtered subspace clustering and low rank matrix approximation. *Signal Process.* 105 (2014), 350–362.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM Trans. Graph.* 34, 6 (2015), 248.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of motion capture as surface shapes. In *Proc. of ICCV*. 5441–5450.

Utkarsh Mall, G. Roshan Lal, Siddhartha Chaudhuri, and Parag Chaudhuri. 2017. A deep recurrent framework for cleaning motion capture data. *CoRR* (2017).

Sang Il Park and Jessica K. Hodgins. 2006. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.* 25, 3 (2006), 881–889.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. 2019. Expressive body capture: 3d hands, face, and body from a single image. In *Proc. of CVPR*. 10975–10985.

Dario Pavllo, Mathias Delahaye, Thibault Porssut, Bruno Herbelin, and Ronan Boulic. 2019. Real-time neural network prediction for handling two-hands mutual occlusions. *Comput. Graph. X* 2 (2019).

Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2020. Modeling human motion with Quaternion-based neural networks. *Int. J. Comput. Vis.* 128, 4 (2020), 855–872.

Maksym Perepichka, Daniel Holden, Sudhir Mudur, and Tiberiu Popa. 2019. Robust marker trajectory repair for MOCAP using kinematic reference. In *Proc. of MIG*. 1–10.

Kathleen M Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, and Scott Fleming. 2002. *Civilian American and European surface anthropometry resource (CAESAR)*. Technical Report.

Javier Romero, Dimitrios Tzionas, and Michael J Black. 2017. Embodied hands: Modeling and capturing hands and bodies together. *ACM Trans. Graph.* 36, 6 (2017), 245.

Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. 2011. Motion reconstruction using sparse accelerometer data. *ACM Trans. Graph.* 30, 3 (2011), 18:1–18:12.

Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. 2006. Modeling human motion using binary latent variables. In *Proc. of NIPS*. 1345–1352.

Zhao Wang, Shuang Liu, Rongqiang Qian, Tao Jiang, Xiaosong Yang, and Jian J Zhang. 2016. Human motion data refinement unitizing structural sparsity and spatial-temporal information. In *Proc. of ICSP*. 975–982.

Jun Xiao, Yinfu Feng, Mingming Ji, Xiaosong Yang, Jian J. Zhang, and Yueting Zhuang. 2015. Sparse motion bases selection for human motion denoising. *Signal Process.* 110 (2015), 108–122.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proc. of AAAI*. 7444–7452.

Victor B. Zordan and Nicholas C. Van Der Horst. 2003. Mapping optical motion capture data to skeletal motion using a physical model. In *Proc. of SCA*. 245–250.