Computers & Graphics: X 2 (2019) 100011

Contents lists available at ScienceDirect

Computers & Graphics: X

journal homepage: www.elsevier.com/locate/cagx

Technical Section

Real-time neural network prediction for handling two-hands mutual occlusions

Dario Pavllo, Mathias Delahaye*, Thibault Porssut, Bruno Herbelin, Ronan Boulic

École Polytechnique Fédérale de Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 31 May 2019
Revised 8 October 2019
Accepted 8 October 2019
Available online 22 October 2019

CCS Concepts:
Human-centered computing
Virtual reality
Computing methodologies
Neural networks

Keywords:
Virtual reality
Neural networks
Machine learning
Motion capture
Inverse kinematics
Finger tracking

ABSTRACT

Hands deserve particular attention in virtual reality (VR) applications because they represent our primary means for interacting with the environment. Although marker-based motion capture works adequately for full body tracking, it is less reliable for small body parts such as hands and fingers which are often occluded when captured optically, thus leading VR professionals to rely on additional systems (e.g. inertial trackers). We present a machine learning pipeline to track hands and fingers using solely a motion capture system based on cameras and active markers. Our finger animation is performed by a predictive model based on neural networks trained on a movements dataset acquired from several subjects with a complementary capture system. We employ a two-stage pipeline that first resolves occlusions and then recovers all joint transformations. We show that our method compares favorably to inverse kinematics by inferring automatically the constraints from the data, provides a natural reconstruction of postures, and handles occlusions better than three proposed baselines.

1. Introduction

Virtual reality (VR) is becoming increasingly popular owing to a new generation of affordable head-mounted displays (HMD). However interactions with the environment can be challenging due to the overly simplified avatar representation leading to a sub-optimal experience. The avatar is often a static mesh, or an animated mesh that moves according to predefined algorithms and actions. One possible way to achieve a compelling immersive virtual reality experience is to capture the user's movements using motion capture and map them on a virtual avatar. Starting from the 3D positions of the markers placed on a suit, an inverse kinematics (IK) solver can recover the state of the avatar joints.

Optical motion capture (mocap) technologies can be classified as *active* (the user wears markers that emit light) or *passive* (the markers reflect light from another source). The advantage of an active system over a passive one is that each marker can be tagged with a unique ID, which simplifies many tasks owing to the fact that there is no confusion among markers. Our work focuses on the former category. Alternative technologies are based on inertial measurement units (IMUs) which, in the 9-axis variant, can track acceleration and angular rate. They can be very precise in a short-term context, but suffer from long-term drifts due to the lack of an absolute positional reference.

The experience of full-body motion control of an avatar into a VR experience is referred to as *embodiment*. Important limitations for a successful embodiment of the avatar are the discontinuities potentially caused by self-occlusions and the absence of movement of the hands, both leading to a reduced sense of body ownership [1]. In such scenarios, increasing the number of cameras is not always feasible, and does not solve the problem entirely.

As an alternative to inverse kinematics, we propose a real-time algorithm based on machine learning that addresses the issue of occlusions through a predictive model. In the first part of the paper (Section 1.1) we discuss the state of the art and analyze how our method relates to other approaches. In Section 2 we describe the features of the dataset used for training our model. In Section 3, we recall three baseline methods for correcting occlusions and we introduce a more complex model based on neural networks for handling both occlusions and inverse

* Corresponding author.
E-mail address: mathias.delahaye@epfl.ch (M. Delahaye).

https://doi.org/10.1016/j.cagx.2019.100011

kinematics. Section 4 describes our experimental methodology while Section 5 details the context of bimanual tracking. Finally, we present our results in Section 6 prior to the concluding discussion.

1.1. Related work

Occlusion robustness. The most common approach for correcting occlusions is to use interpolation algorithms. In this regard, some data-based interpolation techniques have been specifically designed for human body tracking and skeleton animation [2]. However, interpolation algorithms require knowledge of both past and future data, and can therefore be only applied in post-processing. More recently, denoising neural networks have been proposed for offline cleaning of motion capture data [3], producing results that are comparable to hand-cleaning.

Aristidou et al. proposed an approach based on Kalman filters for estimating the positions of occluded markers in real time [4]. Their method does not require any prior knowledge of the skeleton, but assumes that the distance between neighboring markers is approximately constant. The algorithm builds a skeleton model by estimating the centers of rotation between any two sets of points. When an occlusion occurs, the marker position is predicted using a Kalman filter, which takes velocity into account, as well as the positions of neighboring markers. Piazza et al. developed a real-time extrapolation algorithm which assumes that motion can be either linear, circular, or a combination of both [5]. As before, it does not rely on a predefined skeleton model. The prediction is performed through a moving average of the velocity of the marker, so as to minimize the effect of noise. An interesting optimization employed in this approach consists in the so-called constraint matrix (CM), which stores the minimum/maximum pairwise distances between all markers. At inference, the estimates are adjusted according to the constraints described in the CM. Both [4] and [5] focus their work on limbs, and do not address the particular case of fingers. Finally, a large portion of research in this field exploits the assumption that an underlying skeleton model is available, thereby allowing the algorithm to put some constraints on the solution. Recently Alexanderson et al. addressed the problem of labeling markers in a passive system for the fingers and the face [6]. Instead of tracking markers in the temporal domain, it estimates the most likely assignments using Gaussian Mixture Models (GMMs). This allows a fast recovery from occlusions and avoids the so-called *ghost markers*, i.e. detection of markers that do not exist. This approach, however, does not address the problem of predicting the marker positions during occlusions.

Current real-time machine-learning-based approaches for handling occlusions are restricted to the sub-problem of posture and gesture recognition [7]. In our case, we do not perform such classification tasks; our aim is rather to achieve a complete reconstruction of the hand posture.

Tracking and reconstruction. A common framework for capturing movement is to perform body or hand reconstruction through images and depth cameras. These approaches leverage computer vision and machine learning algorithms [8] and aim at providing an affordable consumer-ready alternative to complex motion capture systems. Solutions focusing on hand and finger movements have made significant progress for tracking isolated hands in free space. These techniques are designed for the context of desktop-range interactions using specialized devices – a noteworthy example is the Leap Motion controller. When mounted on a head-mounted display (HMD), these types of finger tracking devices can offer an interesting compromise for immersive VR [9]. Nevertheless, their field of view is still limited when compared to the range of motion of the hands, and they present weaknesses when the hand palm is not facing the head of the user, thereby resulting in self-occlusions. Previous work has tried to address this problem, for instance Tkach et al. fit the hand posture using a combination of sphere meshes [10] while Mueller et al. use a cascade of convolutional neural networks (CNNs) to first localize the hand center and then regress 3D joint locations [11]. They also employ a synthesized dataset that simulates cluttered environments via a merged reality approach, allowing the model to generalize better. These approaches, however, are still very limited in terms of range of motion as they are optimized for a user facing the camera.

As for tracking with motion capture, Han et al. frame the problem as a keypoint estimation task, which is tackled with CNNs [12]. While their approach allows using a passive system, the authors highlight some shortcomings with multiple occlusions. Other frameworks based on motion capture typically employ some sort of sensor fusion from multiple data sources. Andrews et al. propose a tracking system which uses IMUs and a physics model to recover from sensor dropout [13]. Our approach is related to [13] in the sense that we combine IMUs with motion capture to record a robust dataset, but differs in the fact that we use only motion capture at inference.

Machine learning for inverse kinematics. As an improvement over existing techniques, [14,15] proposed a deep learning framework in which a forward kinematics layer is added to a neural network to constrain the output to feasible postures. Specifically, Zhou et al. [14] focuses on hand pose estimation and [15] on full-body pose estimation. As with the MS Kinect, these techniques rely on regular cameras and computer vision algorithms. As such, they are unable to exploit the potential that a full motion capture system has to offer, in terms of both precision and range of motion.

A recent survey classifies inverse kinematics techniques into several categories, which can be summarized as “traditional” (analytic, Jacobian-based), and “data-driven” – often based on machine learning, and recently, on deep neural networks [16]. Most related work focuses on inverse kinematics in the most general setting, which consists in defining the desired positions of the end effectors and have the model find a configuration that achieves the desired result. This particular problem has already been tackled with machine learning, in an industrial control setting, i.e. robot arm [17,18], and in humanoid fingers [19].

1.2. Contribution

We propose a compromise between skeleton-less methods (i.e. zero knowledge) and those with skeletons. Despite being closely related to *data-driven* inverse kinematics, it is more correctly referred to as “reconstruction”. We map the captured markers to the transformations of a virtual hand, but the end effectors do not need to be aligned with the corresponding markers – in fact, markers and joints belong to two different sets, whose correlation is exploited by the model. Instead of defining constraints manually (as in IK systems), all the necessary information is automatically inferred from the data, so as to obtain the most precise and naturally-looking prediction. Our work focuses on motion capture with active markers and proposes a machine-learning-based alternative to analytic IK algorithms, as well as a method for correcting occlusions.

In this paper, we show how we acquired a dataset from a number of subjects, and devised an efficient two-stage pipeline that first corrects occlusions in the motion capture stream, and then reconstructs all the transformations of the hand joints. Both stages are based on neural networks, which are trained on the aforementioned dataset. We evaluate our model at different levels: reconstruction error of occlusions, end-to-end reconstruction error of joint positions, and computational cost in terms of CPU and memory usage – crucial factors for real-time applications. The preliminary version of this approach [20] has been extended as

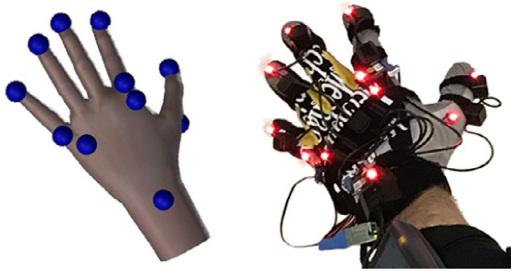


Fig. 1. The virtual hand (left) and the mocap glove (right).

(a) Hand template in neutral pose, with marker IDs

(b) The glove with all the trackers (IMU, LED).

Fig. 2. Comparison of the pose reconstruction before and after simple calibration.

follows. First, we add a calibration process to increase the fidelity of the reconstruction of the model. Second, we handle bimanual occlusions in real time. The system is also re-evaluated with a broader range of use cases.

We release our demo and data publicly.[1]

2. Data acquisition

We start by briefly introducing the characteristics of the motion capture pipeline, and the final goal that our model is expected to meet. The user wears a pair of gloves equipped with LED markers. The positions of these markers are collected by a motion capture system and passed to the pipeline, which outputs the transformations necessary for animating the virtual hands in a VR environment. The mapping is depicted in Fig. 1. More formally, the pipeline comprises the following inputs and outputs:

Inputs: The absolute positions (i.e. 3D points) of the markers. Given that we employ an active motion capture system, each marker is tagged with its own unique ID. Some positions can be missing from the data stream if the corresponding markers are not visible from a minimum number of cameras, i.e. they are *occluded*.

Outputs: The angles of each joint in the hand, as well as the absolute position and orientation of the latter.

As mentioned, this mapping is learned from a dataset that we have collected for this specific task. The next section shows how we acquired the data, and how we built our ground truth for the purposes of training and evaluating the model.

2.1. Devices

We acquired the data using two devices:

- PhaseSpace ImpulseX2 motion capture system, based on active LED markers. Each marker is tagged separately with a unique ID via a frequency modulation mechanism. This system can track the entire body by means of a suit equipped with markers.
- Noitom Perception Neuron, a low-cost hand-tracking device based on inertial measurement units (IMUs).

The two were combined on the custom glove shown in Fig. 2 and used simultaneously for computing the ground truth through a sensor fusion algorithm, which we describe in Section 2.2.

The original PhaseSpace glove, prior to customization, comprised 8 markers (denoted as "Original markers" in Fig. 2). The positions of some markers are not optimal, especially due to the lack of a marker at the wrist level, a crucial location for estimating the orientation of the hand. For this reason, we discarded two markers on the original glove and decided to add three additional markers (referred to as "Alignment Marker" in Fig. 2). The secondary system (Perception Neuron) served the role of collecting the mapping between marker positions and joint positions/angles. It consists of a flexible glove with several 9-axis IMUs placed on top of fingers. Due to its nature, this system is immune to occlusions but presents the issue of drifting over time. This is an intrinsic problem of inertial tracking; it cannot be corrected without an absolute reference. Additionally, although the IMUs can in theory detect all degrees of freedom, their particular finger reconstruction algorithm can sense only one axis: the finger flexion-extension. As a consequence, finger spread/crossing cannot be detected.

We solved these issues by combining the readings from the Perception Neuron with the ones from the PhaseSpace, in a process known as *sensor fusion*. The details are explained in Section 2.2. Accordingly, we also moved the Perception Neuron's sensors to our custom PhaseSpace glove (Fig. 2). The IMUs were used only during the dataset recording phase, and they were removed afterwards.

2.2. Sensor fusion

The PhaseSpace motion capture system provides us with absolute tracking, whereas the Perception Neuron offers relative tracking. As the data streams between these sources are very different, it is crucial to devise a sensor fusion algorithm that yields plausible results. From a high-level perspective, the algorithm is divided into a series of steps.

Setup. Each marker is assigned to one of the joints of a hand template – the rigged 3D model in Fig. 2 – with the possibility of specifying an offset relative to the joint (i.e. in object space). The offsets are static and must be known in advance because they depend on where the markers have been physically placed on the glove. Three extra markers visible on Fig. 2 are tagged as *alignment markers* as they are used for estimating the location of the hand in space. Our choice was to form a triangle on the back of the hand, namely the markers corresponding to the wrist, the base of the index, and the base of the pinky.

Estimation of hand position/orientation. We compute an optimal *rigid motion* transformation – which comprises only a rotation and a translation – from the hand template in local space to the hand in world space. More formally, we denote the positions of the joints in the hand template (Fig. 2) as $\mathbf{U}$, and the positions of the markers as $\mathbf{V}$. The algorithm takes the two lists of points $\mathbf{U}$ and $\mathbf{V}$ (which have the same number of points and the same dimension) as input, and returns a transformation $\mathcal{T}$ such that:

$$\mathcal{T}(\mathbf{u}) = \mathbf{u}\mathbf{R} + \mathbf{t} \tag{1}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is an offset. We assume that all vectors are in row-major order. This transformation minimizes the *mean squared error* (MSE) between the source positions and the target positions, defined as:

$$\mathrm{MSE}(\mathbf{U}, \mathbf{V}) = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{V}_i - \mathcal{T}(\mathbf{U}_i) \right\|^2 \tag{2}$$

[1] https://github.com/dariopavllo/robust-finger-tracking.

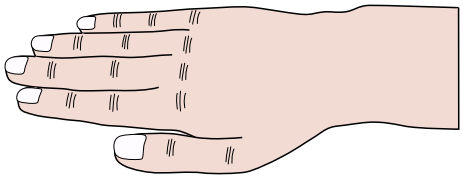
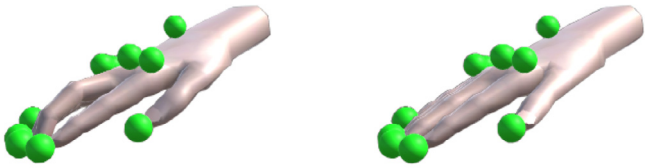


Fig. 3. Illustration of the hand calibration pose.



(a) Before (b) After

Fig. 4. Illustration of the simple calibration.

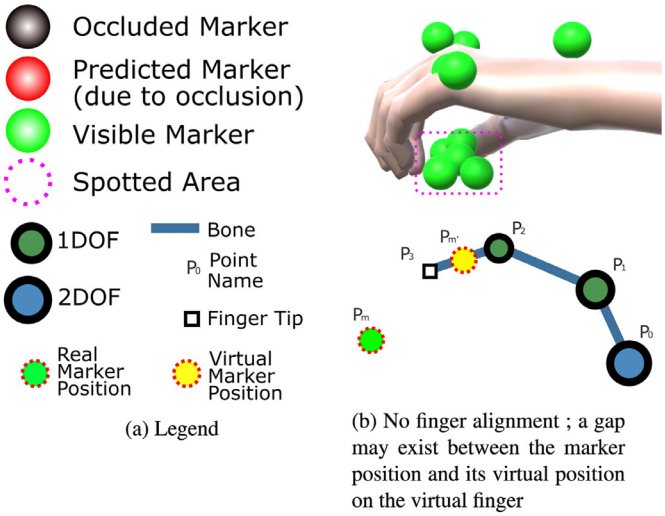


Fig. 5. Finger alignment post-processing.

Fortunately, there exists a closed-form solution for this problem, which is also very efficient. It is based on the singular value decomposition (SVD), and can be computed using Kabsch's algorithm [21]. In our case, the transformation is calculated using only the alignment markers (i.e. the 3 markers on the back of the hand).

Joints calibration. An additional calibration stage is necessary to make the proposed approach more robust to hand variety (Fig. 4). The user has to adopt an occlusion-free flat pose corresponding to the default pose of the animated avatar with the identity transformation for all the joints (Fig. 3). While performing this pose, we record all the predicted joint transformations and store their inverse as the (constant) calibration offset transformation. By construction, combining each joint prediction with its calibration offset produces the desired identity transformation for that pose (Fig. 4). This calibration offset is then applied systematically to the prediction at run-time to partially handle the variety of user hands. It is completed with the post-processing stage described below.

Post-processing. In Fig. 5(b), one can notice a gap between the fingertips and their associated markers. This is caused by inaccuracies of the Perception Neuron. We mitigate this issue by applying an artificial rotation to every finger, so that, after the transformation, every finger points in the direction of the corresponding marker. Specifically, we denote with $\mathbf{p}_0$ the position of the base of the finger (metacarpophalangeal joint), with $\mathbf{p}_3$ the position of the fingertip (returned by the Perception Neuron), and with $\mathbf{p}_m$ the position of the marker (returned by the PhaseSpace). We are not interested in modifying intermediate joints as they have no associated marker. However, of course, the position $\mathbf{p}_3$ depends on the orientation of $\mathbf{p}_2$ and $\mathbf{p}_1$ (the intermediate joints, see Fig. 5). Ideally, we would want $\mathbf{p}_3 = \mathbf{p}_m$, and this is what a traditional inverse kinematics (IK) solver achieves. However, this constraint is too strong since it would force unnatural postures in certain cases. On the other hand, our aim is just to apply a small correction to the data already obtained from the Perception Neuron, and therefore a simple rotation is sufficient. The approach adopted in [20] limited our ability to reproduce a contact between the thumb and other finger tips. Fig. 5(d) illustrates the principle of the proposed approach to reduced such a gap. We first compute the position of the last finger mid-segment $\mathbf{p}_{m'} = (\mathbf{p}_3 + \mathbf{p}_2)/2$ to better reflect the marker location. Then we rotate the finger by the shortest-arc quaternion rotation from vector $\mathbf{p}_{m'} - \mathbf{p}_0$ to vector $\mathbf{p}_m - \mathbf{p}_0$. At this point, all the limitations of the Perception Neuron have been overcome: all possible gestures/postures can be detected, including the most problematic ones (e.g. finger spread and finger crossing).

3. Machine learning model

3.1. Pipeline

We adopt a two-stage model: the first step (marker predictor) predicts the positions of the occluded markers, and the second step infers the angles of all joints from the output of the first step, assuming that there are no occlusions. We train the two models separately, and not in an end-to-end fashion, as our approach for enforcing temporal consistency (described in Section 3.2) is not differentiable. Having two stages presents some advantages from a flexibility standpoint. If occlusion correction is not required by a particular task, the joint predictor could be used out of the box as if it were an IK solver. Moreover, a potential developer could use different algorithms for each system: the occlusion manager could be based on neural networks, linear models, or anything else, and it would not affect the behavior of the second model. Similarly, the marker prediction model could be used solely for the purpose of handling occlusions, and a traditional IK solver could be added on top of it – note however that this requires a 1–1 correspondence between markers and joints, which is not a requirement of our system. We show a block diagram of our full pipeline in Fig. 6.

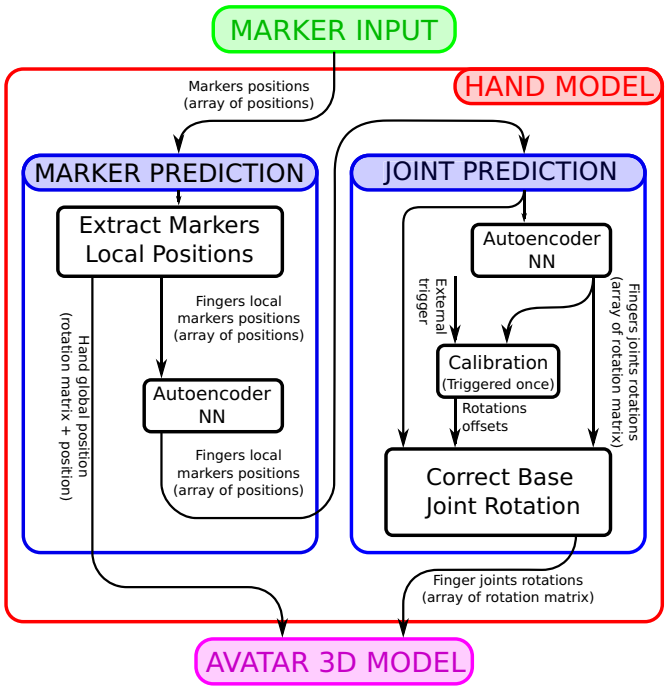


Fig. 6. Conceptual full prediction pipeline.

3.2. Marker predictor

Before presenting our model based on neural networks, we recall the three simple baselines that are evaluated against our method.

It is worth mentioning an important property that this step must implement: *temporal consistency*. The model should enforce a "smoothness" condition between subsequent frames, so as to avoid *discontinuities* (sudden jumps in the joint transformations). We can identify two types of discontinuities:

Discontinuity on occlusion: when a marker is visible at time t and becomes occluded at time $t+1$.

Re-entry discontinuity: when a marker that was previously occluded at time t becomes available again at time $t+1$.

While discontinuities on occlusions can be corrected explicitly by enforcing temporal consistency in the model, re-entry discontinuities cannot be solved without having future knowledge of the data. In a real-time system like ours, this means that they must be smoothed manually as a post-processing step.

3.2.1. Baselines

We now introduce the three aforementioned baselines, in order of increasing complexity:

Last known position. The simplest baseline consists in keeping the last known position of an occluded marker. With regard to discontinuities on occlusions, this method is temporally consistent.

Moving average. Inspired by Piazza et al. [5], we take velocity into account. We keep a moving average of the velocities of each marker over the last k frames (we use $k = 20$, i.e. one third of a second) to minimize the effect of noise. When a marker is occluded, this baseline simply moves the marker along the trajectory defined by the average velocity.

Affine combination model. Finally, we propose an improvement over the previous baselines. Another simple (yet effective) method consists in expressing an occluded marker as an affine combination of the other available markers, i.e. a linear combination with weights that sum up to 1. The computation is performed using the data from the previous frame, where the occlusion was not present. In order to enforce the affine property, it is sufficient to add a homogeneous coordinate to each point, and fix it to 1. More formally, we denote with $\mathbf{X_i}$ $(i = 1..N)$ the set of all known positions ($\mathbf{X}$ is a $N \times 4$ matrix), $\mathbf{Y_j}$ $(j = 1..M)$ the set of occluded points that must be predicted (the result $\mathbf{Y}$ would be a $M \times 4$ matrix), and $\mathbf{W}$ the weight matrix of size $M \times N$. It must follow that:

$$\mathbf{Y} = \mathbf{W}\mathbf{X} \tag{3}$$

$$\sum_{i=1}^{N} \mathbf{W}_{j,i} = 1 \quad \forall \mathbf{j} \tag{4}$$

where $\mathbf{X_i} = [X_{ix}, X_{iy}, X_{iz}, 1]$ and $\mathbf{Y_j} = [Y_{jx}, Y_{jy}, Y_{jz}, 1]$. We are again assuming a row-major vector notation. This problem can be solved using exactly 4 non-coplanar markers. If more markers are available, the problem is underdetermined, as there are infinite solutions to the linear system. Therefore, we apply L2 regularization, which means that among all possible solutions, we choose the one that minimizes the squared norm of the weight vector. In other words, the predicted position should depend on all the other markers, each of which has a small weight; this leads to a better robustness to noise. We minimize the loss function:

$$\mathcal{L}(\mathbf{W}) = \sum_{j=1}^{M} (\|\mathbf{Y}_j - \mathbf{W}_j\mathbf{X}\|^2 + \lambda\|\mathbf{W}_j\|^2) \tag{5}$$

$$= \|\mathbf{Y} - \mathbf{W}\mathbf{X}\|_F^2 + \lambda\|\mathbf{W}\|_F^2 \tag{6}$$

where $\|\mathbf{M}\|_F$ denotes the Frobenius norm of $\mathbf{M}$, and λ is a small positive regularization constant ($\lambda = 10^{-8}$ is suitable in our case; in general, one should choose the smallest value that does not cause numerical precision issues). Fortunately, the function is convex and there exists a closed-form solution for its minimum. We derive the gradient with respect to $\mathbf{W}$ and equal it to zero:

$$\nabla\mathcal{L}(\mathbf{W}) = -2\,(\mathbf{Y} - \mathbf{W}\mathbf{X})\,\mathbf{X}^T + 2\lambda\mathbf{W} = \mathbf{0} \tag{7}$$

Solving for $\mathbf{W}$ we obtain:

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\,\mathbf{I_N})^{-1} \tag{8}$$

where $\mathbf{I_N}$ is the $N \times N$ identity matrix. This approach is closely related to *ridge regression* [22].

As before, discontinuities on occlusions are avoided by design. Furthermore, this baseline is intrinsically invariant to translations and rotations. Since $\mathbf{Y_j}$ is expressed as an affine combination of all $\mathbf{X_i}$, any rigid transformation applied to $\mathbf{X}$ would be applied to $\mathbf{Y}$ as well, i.e. $f(\mathcal{T}(\mathbf{X})) = \mathcal{T}(f(\mathbf{X}))$ (where $\mathcal{T}$ is a rigid transformation). From a practical standpoint, if the hand is kept in a static posture and moved around the capture space, the occluded markers is reconstructed perfectly. We also found this baseline to perform relatively well on gestures that do not involve complex movements.

3.2.2. Marker regressor

In theory, neural networks (NNs) can approximate any function (provided that a sufficient number of neurons is available) [23] but, in practice, the result is strongly dependent on how the data is pre-processed.

Similarly to the affine combination model, we want our prediction to be spatially invariant, in the sense that any translation/rotation transformation applied on the input points should not affect the output of the neural network. Therefore, we enforce, for this step, a pre/post-processing scheme that allows the network to learn a proper mapping thanks to the reduced search space. These are named:

Marker position extraction in hand referential (registration): The rotation and translation of the hand in space are removed.

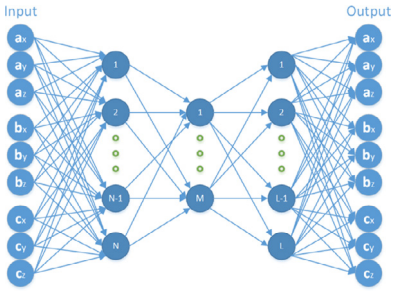


Fig. 7. An autoencoder with three hidden layers. In a scenario where **a** and **c** are available, and **b** is occluded, we disconnect the inputs corresponding to **b**, and we get the prediction of **b** in the output. Here, only 3 markers are depicted; in practice, we would have 9 markers.

Table 1
Full list of layers in the marker regressor.

Type	Shape
Input	9×3
Flatten	27 neurons
Fully connected + ReLU	200 neurons
Fully connected + ReLU	150 neurons
Fully connected + ReLU	200 neurons
Fully connected + Linear	27 neurons
Reshape (output)	9×3

This can be achieved by aligning the hand to the standard template, which is centered on the world origin and is oriented toward a predefined axis. The alignment can again be performed by finding the lowest-error rigid motion transformation. This process can be regarded as the inverse operation of the hand position estimation presented earlier: instead of moving the hand template towards the markers, here the markers are moved towards the hand template. The only difference is that here we use all available markers, and not only the 3 alignment markers (since they may be occluded). The hand template is kept in neutral pose (see Fig. 2), and therefore this step is dependent on the hand posture, but this does not represent a problem as the goal of this step is to perform spatial normalization.

Reconstruction of the markers positions (de-registration): The inverse transformation is applied to the predicted points, that is, the markers are put back to their original positions in world space.

With regard to how occlusions are handled, it is important to note that neural networks cannot operate on missing data. Hence, a special architecture and/or training procedure is required. A thorough approach consists in building an ensemble of different models [24], one for each possible set of available markers, and train them independently from each other. It is clear that this method presents a severe limitation: the number of models to train increases exponentially as more markers are added, not to mention the tremendous computational (and memory) cost both at training and inference.

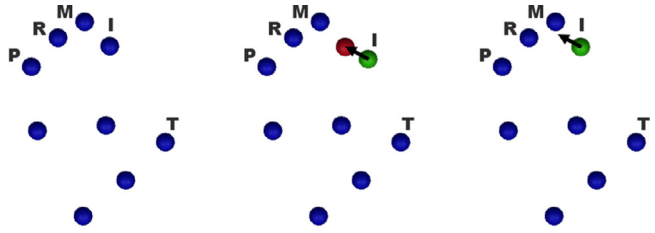
Instead, we employ a single feed-forward neural network configured as an *autoencoder*, i.e. a topology that maps the identity function $\mathbf{x} \longrightarrow \mathbf{x}$, as depicted in Fig. 7. The network comprises $3N$ input neurons and $3N$ output neurons, where N is the total number of markers (9 in our case). Each group of 3 neurons encodes the XYZ positions of a particular marker, after the pre-processing step described above.

The structure of the neural network is shown in Table 1. All layers except the last one use ReLU (Rectified Linear Unit) activation functions [25], defined as $y = \max(0, x)$, as they have been shown to yield the best results in a wide range of tasks [26,27]. The output layer uses a linear activation function, thereby allowing an unbounded output range. All hyperparameters were chosen to minimize the reconstruction error on the validation set, also taking into account performance and latency constraints. We also experimented with varying numbers of layers and discovered that more layers lead to overfitting on this specific task (regardless of regularization).

Our mechanism for handling occlusions is closely related to *Dropout* [28], a training technique traditionally used to avoid overfitting the training set. Dropout works as follows: during training, at each iteration, a random fraction of neurons are disconnected (which is equivalent to setting their output values to 0). At inference, all neurons are used. We apply a procedure similar to Dropout on the input layer. The model is trained using a data augmentation procedure: the dataset is generated in real time by setting a random number of points (groups of 3 neurons) to 0 from frames containing exclusively all visible markers, according to the distribution observed in Fig. 13 (with a number of occlusions between 1 and 4). The exact distribution is not crucial, but it helps with improving the error in realistic cases. It is worth noting that the pre/post-processing scheme still applies to this approach. The inputs must be disconnected *after* the positions are registered (i.e. are transformed into object space). The prediction algorithm is trivial: all available (non-occluded) points are registered and passed as inputs to the neural network, whereas the inputs corresponding to missing values are set to 0; the relevant outputs (i.e. the ones corresponding to the occluded markers) are extracted and de-registered.

Autoencoders learn a compressed representation of the data [29], instead of just copying the input to the output. In our particular case, the bottleneck layer learns a *positional embedding*, i.e. a vector that encodes a particular posture. Our representation is overcomplete, meaning that the number of neurons in the bottleneck is greater than the number of input neurons. However, our training procedure acts as a regularizer, effectively forcing the model to learn a sparse representation that is suitable for reconstructing missing values. ReLU activations also contribute to sparsity [26].

Discontinuities. Unlike the affine model discussed earlier, the feed-forward neural network approach tends to suffer from discontinuities because it does not enforce temporal consistency explicitly. Since a feed-forward model does not contain any state information, it simply finds a solution that minimizes the error in the average case, without being able to take into account any previous context. From the user's point of view, this results in a bad experience. Other neural network architectures, such as recurrent neural networks (RNNs), can exploit past information. However, even with them, handling missing values is a non-trivial task that could still result in discontinuities. Our preliminary experiments showed that this is indeed the case. Hence, we stick with feed-forward networks due to their lower computational cost and ease of training, and we adopt special measures to correct discontinuities. When a marker becomes occluded, we compute an offset term and we apply it to all subsequent outputs, until the occlusion is resolved. More specifically, given an occlusion at time t, we perform a prediction with the data from the previous frame $t-1$ (where the real position was known). Afterwards, we calculate an offset that cancels out the discontinuity; this offset is retained as state information and is modified only if another occlusion happens, or if the occlusion is resolved. The offset is applied to the output of the marker predictor network before the points are de-registered. To calculate the offset, we simply compute the difference between the predicted position and the actual position in local space. We also explicitly correct re-entry discontinuities using the same technique; the only difference is that the offset is decayed to zero over time (using a linear decay



(a) Frame $t-1$ (b) Frame t (c) Frame $\geq t$

Fig. 8. Handling of discontinuities. **(a)** At $t-1$ no marker is occluded. **(b)** At t the index marker is occluded. The NN predicts its hypothetical position at $t-1$ (green), which results in a small discontinuity from the true position (red). **(c)** From t onwards, the discontinuity is explicitly canceled by moving the marker by the offset vector (black arrow). For re-entry discontinuities, the process is reapplied in the opposite direction, but the offset vector is progressively shrunk to remove the bias.

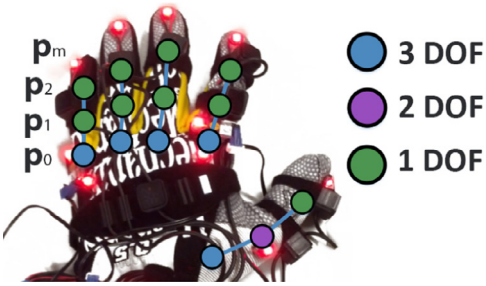


Fig. 9. Degrees of freedom of the hand joints (26 in total).

Table 2
Full list of layers in the joint regressor.

Type	Shape/Notes
Input	9×3
Flatten	27 neurons
Fully connected + ReLU	200 neurons
Dropout	$p = .1$
Fully connected + ReLU	200 neurons
Dropout	$p = .1$
Fully connected + ReLU	200 neurons
Fully connected + Linear (output)	26 neurons

function) in order to remove the bias. We observed that a decay speed of 25 cm/s offers a good compromise between reactivity and smoothness. Fig. 8 depicts this process.

3.2.3. Joint regressor

The joint regressor predicts the angles of the fingers, given the marker positions as input. It solves a task similar to that of an IK solver, but instead of using a calibrated skeleton, it adapts to the user hand, according to a dataset of realistic motions. Furthermore, it does not need to handle missing values, as they are assumed to be predicted by the previous stage of the pipeline. We adopt a dense neural network for this task, which takes the marker positions as inputs ($9 \times 3 = 27$ neurons), and predicts the Euler angles of all relevant joints, for a total of 26 Euler angles. The use of Euler angles instead of other representations, such as exponential maps [30] or quaternions, is motivated by the observation that our fingers have limited degrees of freedom. We need to predict only certain angles, thereby obtaining a smaller neural network. Fig. 9 shows the degrees of freedom that are modeled, while Table 2 shows the structure of the neural network. As before, all layers except the last one use ReLU activation functions [25]. Moreover, we used Dropout [28] in the intermediate layers (with a probability of 0.1, meaning that 10% of neurons are randomly dropped at each training iteration) to avoid overfitting. This proved effective in improving the validation error.

During both training and inference, the inputs are registered (all rotations/translations are removed). Additionally, we found the same post-processing technique employed in the sensor fusion (i.e. artificial joint rotation, Section 2.2) to be effective.

3.3. Training

For both models, we optimize the mean squared error (MSE) loss using the Adam optimizer [31] with an initial learning rate $\eta = 0.001$. The learning rate is automatically adjusted once the error reaches a plateau; more specifically, it is halved if the error has not improved over the last 5 epochs. The model is trained only on simulated occlusions, as they are the only ones for which a reliable ground truth can be obtained, and with a batch size of 32 samples.

4. Experimental protocol

4.1. Left hand dataset

The dataset was recorded from four subjects (three males and one female, age range 22–30), all right-handed, and with different hand sizes. Every subject underwent eight recording sessions of approximately 60–80 s each, and the Perception Neuron was calibrated before each session (with a quick follow-up check) (Fig. 10). This approach ensures that IMU drifts do not degrade the dataset accuracy. As for the movements, the subjects were left free to execute any movement, but were also instructed to perform at least some key gestures. In order to evaluate the model, the dataset was partitioned into a *training set* (2 subjects), a *validation set* (1 subject), and a *test set* (1 subject). The validation set was used for tuning the hyperparameters and testing different architectures, whereas the test set was used only for the final evaluation.

4.2. Two hands dataset

A second dataset was recorded from five subjects (four men and one female, age range 24–42), all right-handed with different hand sizes. Each subject spent 100 s wearing the two gloves but this time without the perception neuron system as it was no longer required. The subjects had to achieve three tasks (both hands finger crossing, palms in contact, fingers in contact) and

Fig. 10. Left: a person wearing the recording equipment (Phase-Space glove and Perception Neuron). **Right:** a person testing the application in a VR environment with an Oculus HMD.

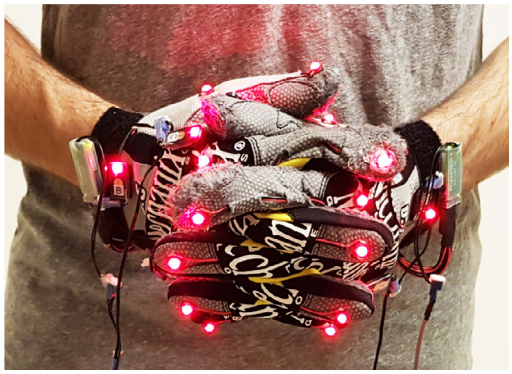


Fig. 11. A subject wearing the two gloves during the recording phase of the dataset.

were free to move the rest of the time. This dataset was used in order to compute the rate of occlusions per hand in comparison with a system with only one hand (Fig. 11).

5. Mirroring

The reconstruction of the right hand pose exploits the model trained for the left hand. For this we simply transpose the behavior of the left hand pipeline to the right hand, using the natural plan of symmetry of our skeleton to flip the markers coordinates according to this plan (Fig. 12). The new pipeline handling both hands is a composition based on the pipeline describe in Section 3.

Then the set of mirrored coordinates is given as an input to the pipeline with the neural network for the prediction of the markers positions. The neural network trained on the left hand dataset now sees an input matching a left hand and predict the occlusions for this virtual left hand.

These data are stored in the hand model object in order to be accessed for the hand pose estimation and for the next step. The next step consists in filling these predicted markers to the neural network trained on the left hand to predict the joints rotations of each finger, and as above, it sees a left hand.

As for the marker position prediction, the output is also stored in the hand model object, and are forwarded to the 3D model for its animation.

This design allows us to train only once the neural network with the dataset of one hand, and using it as many times required for the number of hand required in the simulation. Also we were able to use four hand in our simulation environment.

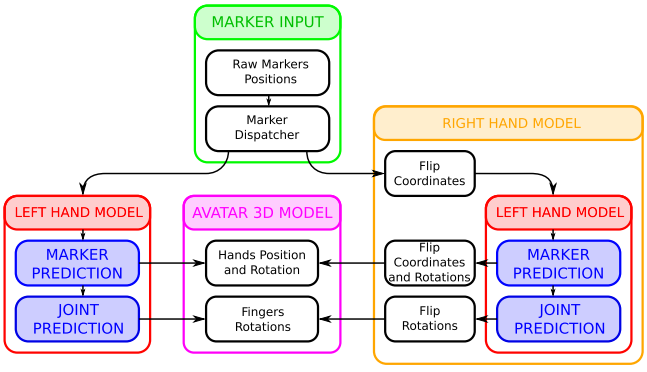


Fig. 12. Main steps of the pipeline used to transform raw markers position from VRPN to the position of the avatar's body. Refer to Fig. 6 for details of the left hand model.

6. Results

6.1. Datasets

6.1.1. Left hand dataset

The training set consists of ≈30 min of data recorded at 60 FPS. In theory, the PhaseSpace system can record at up to 480 FPS, but we limited the sample rate to 60 FPS to avoid collecting too many redundant samples. Fig. 13 reveals some insights: most occlusions involve a small number of markers, that is, the probability that multiple markers are occluded at once is low. Moreover, the duration of an occlusion follows a heavy-tailed distribution (90% of occlusions last less than 0.36 s).

6.1.2. Two hands dataset

This dataset is used to compute the number of occlusion occurring with two hands instead of one hand. We used the same frame rate for this comparison. As we can see in Fig. 14, the probability to have more than one or two occluded markers is higher than in the single hand case. The occlusion duration is likely to be longer

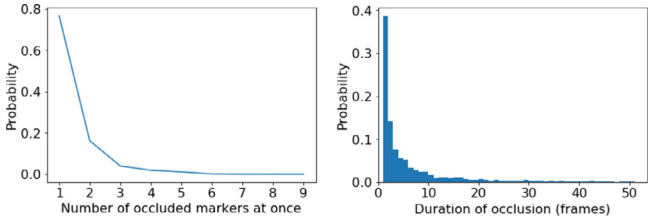
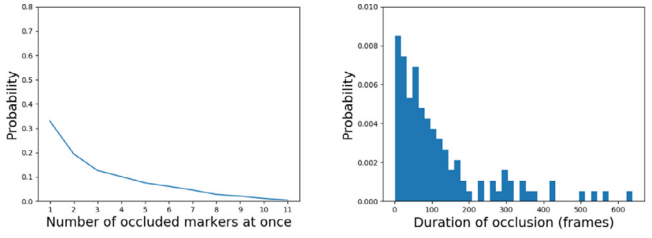
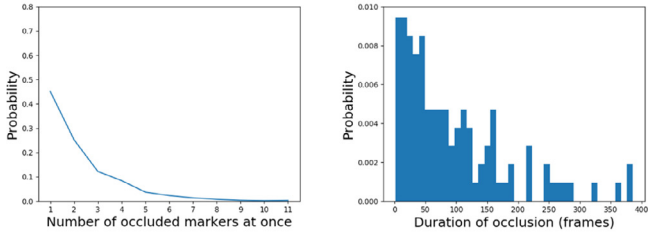


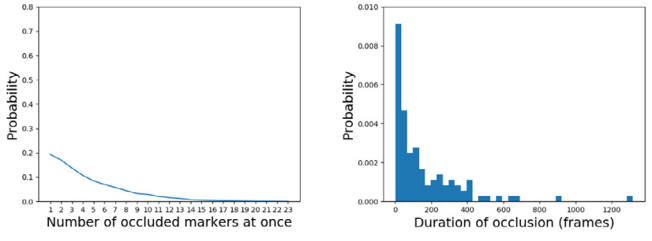
Fig. 13. Probability of N occlusions at once (Left)/Occlusion duration histogram (Right).



(a) Left Hand



(b) Right Hand



(c) Both Hands

Fig. 14. Probability of N occlusions at once (Left)/Occlusion duration histogram (Right). Plots represent the same as in Fig. 13.

Table 3
Evaluation of the error on the marker neural network (error units=centimeters, lower=better). Legend: **LK** last known position, **MA** moving average, **AC** affine combinations, **NN** neural network.

Method	# Occlusions	Occlusion duration (seconds) 0.1	0.2	0.5	1.0	2.0
LK	Any	1.54	2.58	4.43	5.15	8.06
MA	Any	2.23	4.28	9.99	20.19	44.28
AC	1	0.97	1.54	2.42	2.92	3.67
	2	1.06	1.69	2.61	3.47	4.12
	3	1.22	1.96	2.95	3.65	4.52
	4	1.66	2.68	4.06	5.34	5.45
NN	1	0.56	0.84	1.19	1.46	2.09
	2	0.60	0.91	1.33	1.57	2.08
	3	0.68	1.03	1.48	1.81	2.32
	4	0.79	1.20	1.78	2.14	2.72

Table 4
Evaluation of the error on the final joint positions (error units=centimeters, lower=better). Legend: **IK** inverse kinematics, **FT** fine-tuned, **NN** neural networks.

Method	# Occlusions	Occlusion duration (seconds) 0.1	0.2	0.5	1.0	2.0
IK	0	1.87 (no occlusions)				
IK FT	0	1.08 (no occlusions)				
NN	0	0.07 (no occlusions)				
	1	0.11	0.14	0.17	0.19	0.29
	2	0.17	0.23	0.29	0.35	0.41
	3	0.26	0.36	0.48	0.57	0.79
	4	0.38	0.55	0.79	0.89	1.32

too. This can be explained by the fact that interacting hands may temporarily hide each other.

6.2. Reconstruction error

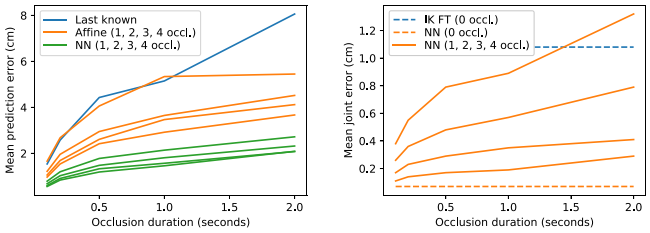
6.2.1. Marker predictor error

Table 3 and Fig. 15(a) reveal the error of the first stage of the pipeline – the marker prediction model. We compare the three baselines with our neural network approach, and we report statistics over varying occlusion durations and number of markers occluded simultaneously. For each trial, we report the average Euclidean distance between the predicted position and the ground truth *on the last frame* before the occlusion is resolved, and only for the occluded markers. For instance, in the scenario "2 markers after 100 ms", we occlude two random markers at once and measure their error after 100 ms. The errors are evaluated across the entire test set and repeated 5 times with different random seeds to smooth out their variance.

Our evaluation methodology addresses both short-term occlusions (100 ms, 200 ms, 500 ms) and long-term occlusions (1 s and 2 s). Each method is tested on a number of occlusions between 1 and 4, except for the first two baselines (*last known position* and *moving average*), which are independent of this parameter. We observe that the moving average baseline exhibits the worst performance, which is caused by the markers drifting away on long-term occlusions. The affine combination model is better than the simplest baseline (last known position) except when many markers are occluded at once. Finally, our neural network approach consistently outperforms all the other methods.

6.2.2. End-to-end error

In Table 4 and Fig. 15(b), we report the error relative to the joint positions by running the entire pipeline. As in the previous section, we measure the average Euclidean distance between the predicted joint positions and the ground truth. The averages are computed only over the finger joints, i.e. $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ (as described in Section 2.2). All errors are relative to the test set. We do not report angle errors because they would not be easily interpretable – errors in the first joints would accumulate along the kinematic chain.



(a) (b)

Fig. 15. **(a)** Comparison between the baselines and our method for the marker predictor error. The moving average baseline is not included because of its excessive error. **(b)** End-to-end error: our approach at varying conditions versus an IK baseline.

We compare our work to an IK library, "Final IK" by RootMotion. We fine-tuned the IK configuration to the best of our ability: we use a Cyclic Coordinate Descent (CCD) solver, with an angle constraint (3 degrees of freedom, max. 45° for flexion–extension/abduction–adduction, and 20° for twist) on the root finger joints, and a hinge constraint (1 degree of freedom – flexion-extension – from −90° to 10°) on middle joints. We show that our approach achieves a significantly lower error (0.07 cm) than inverse kinematics (1.87 cm unconstrained, 1.08 cm fine-tuned) when there are no occlusions. This suggests that a data-driven approach is better at modeling the angle distributions/constraints than a handcrafted setup, thus producing a more naturally-looking reconstruction. For the occlusion scenario, we report only the statistics associated with our method, as IK solvers cannot handle occlusions (some IK approaches such as [32] enable a reduced set of markers, but not a dynamically-changing one).

6.3. Performance

6.3.1. One hand

Our reference implementation is written in C# and runs on Unity Engine. Running the entire pipeline on an Intel Core i5-4460 CPU requires less than 1.2 ms (≈ 833 frames per second). Additionally, the two neural networks have minimal memory footprint (300 kB each).

6.3.2. Both hands

With an Intel Core i9-9900K, the full pipeline takes 2.2 ms on average, based on a simulation of 5 minutes (≈ 455 frames per second), against 1.2 ms for a single hand with this pipeline on this machine.

7. Interacting with both hands

7.1. New possibilities

Having both hands in VR gives the user a more natural way to interact with elements of the virtual world. It allows us to perform simultaneous actions, like changing gears while driving, but also to achieve more complex tasks like handling large objects. The provided video illustrates the actions of opening a drawer to grab an object, fingers crossing and shaking hands.

7.2. Limitations

The autoencoder for the markers positions might not be trained to handle every position. Indeed, some complex positions might

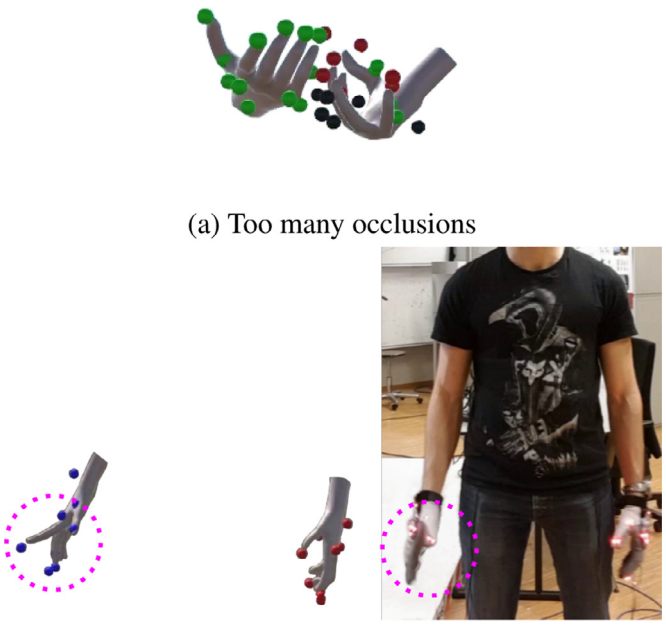


Fig. 16. Examples of failure cases.

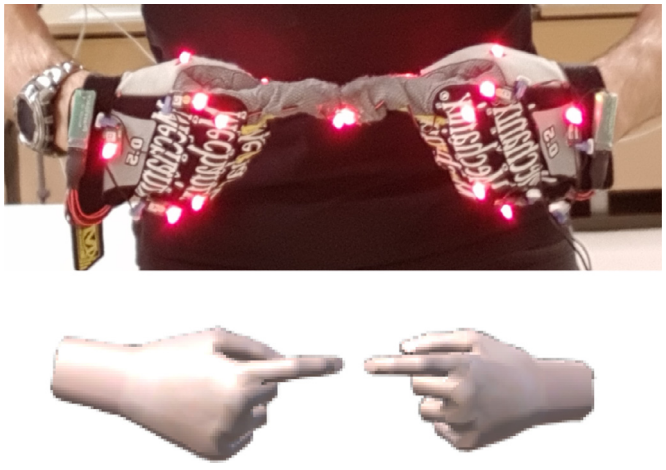


Fig. 17. An illustration of the gap between the two fingers.

induce each time at least one occlusion thus excluding these frames from the training dataset.

As both hands are tracked, this may increase the number of mutual occlusions. In the limit case of too many simultaneous occlusions, the pipeline might also fail and give results like in Fig. 16(a). In that regard, we expect that training a new neural network handling both hands simultaneously could help to predict their correct relative position in contexts where one hand is hidden by the other one.

It also has to be noted that, the thumb having a more complex structure than the other fingers, the post-processing we developed by simply applying a rotation on the base joint should be improved for that finger. We also noticed that, depending on the environment (number of cameras, reflectives surfaces, etc.), the PhaseSpace might give wrong positions instead of empty ones. This might occurs with fast motion (Fig. 16(b)).

Another limitation pertain to the quality of self-contacts as a small gap may remain visible in some cases despite the post-processing stage Fig. 17.

Finally, we may produce a collision in the reconstruction of the model: a finger of the left hand can penetrate the finger of the right hand due to the fact that the system doesn't check for collision.

8. Discussion and future work

To sum up, we present a hand-tracking pipeline to animate virtual hands using raw 3D marker positions. In particular, we address the issue of occlusions by proposing an approach based on neural networks, and compare it to three baselines. Finally, we propose a method for mapping markers to joint angles, which does not require the laborious process of setting up an IK solver.

Our system provides a natural reconstruction of the hands in most real-case scenarios, which we demonstrate by comparing the reconstruction error with that of a traditional solver based on inverse kinematics. Our data-driven approach does not require defining a set of rules or constraints, as these are learned automatically from data. Occlusions are corrected with good accuracy in most cases, and with minimal latency. From an interaction perspective, our finger animation is suitable for object grasping and manipulation, but we observe that we behavior of the thumb – which falls short on pinching gestures – could be improved by using a 3D hand model that resembles the glove more closely. The reader is referred to the supplementary material for a video of our results. We do not employ physical simulation or collision detection, nor do we enforce kinematic constraints when objects are touched, i.e. the good fit between hands and objects comes from the reconstruction alone.

With regard to related work on this subject, previous methods have mainly addressed passive motion capture and limb reconstruction. Out of the few occlusion-handling solutions targeted at active-marker technologies, we investigated [4], which has already been employed in some studies [33]. However, this method requires at least three markers for each segment, which follows from the assumption that the distance between neighboring markers is approximately constant. Given that our hand model comprises at most two markers per finger (tip, and optionally, base), we suggest that a data-driven approach is more suited to this task because it adapts better to the specific domain that should be addressed (hand and fingers reconstruction with only 1 or 2 markers per finger, in our case). This also explains why, in our setting, analytic inverse kinematics perform significantly worse: in the absence of intermediate markers, the algorithm has no knowledge of the priors that constitute a good-looking posture.

Finally, we discuss some limitations of our method and some possible future developments. One drawback that limits the practical applications of our model is the requirement for an active-technology motion capture system for training. On the other hand, an active system presents the undoubtful advantage of being able to track multiple hands (and, possibly, multiple people) with greater precision than its passive counterpart. For instance, Han et al. [12] involve a delicate clustering step to address left/right hand confusion with passive markers. Nonetheless, if a robust tagging layer were integrated into the pipeline, such as the one proposed by Alexanderson et al. [6], our method would seamlessly adapt to passive systems. Another disadvantage of our approach lies in the feed-forward neural network architecture, which does not model an internal state. This does not only concern discontinuities – which are corrected manually in our case – but also how predictions are computed. Our model performs *deterministic* predictions, in the sense that equivalent postures (same input with different outputs) are averaged so as to minimize the reconstruction error. Although we observed that this does not happen frequently, it might be undesirable in some cases. Recurrent neural networks can produce an output that is conditioned on the

previous frames, and they can potentially handle discontinuities without manual corrections, although it is not trivial to enforce temporal consistency on occlusions while keeping the target function differentiable. In the future, we would like to experiment with convolutional, *long short-term memory* (LSTM), and *gated recurrent unit* (GRU) architectures. We would also like to add seamless support for passive motion capture systems, perhaps as a pluggable encoder.

With the current architecture, we have a single-hand neural network that is exploited independently for each hand. It means that each hand position is reconstructed only with its own markers positions regardless the relative position of the second hand, leading to possible collisions. In the future, we would like to extend the architecture with a second neural network trained with a dataset of two interacting hands. Another important point to address in future work is to improve the enforcement of consistent self-contact as this feature has been shown to be critical for supporting the avatar body ownership [1].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the Swiss National Science Foundation, “Embodied Interaction” No. 200020_159968.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cagx.2019.100011.

References

[1] Bovet S, Debarba HG, Herbelin B, Molla E, Boulic R. The critical role of self-contact for embodiment in virtual reality. IEEE Trans Vis Comput Graph 2018;24(4):1428–36. doi:10.1109/TVCG.2018.2794658.
[2] Wiley DJ, Hahn JK. Interpolation synthesis of articulated figure motion. IEEE Comput Graph Appl 1997;17(6):39–45. doi:10.1109/38.626968.
[3] Holden D. Robust solving of optical motion capture data by denoising. ACM Trans Graph 2018;37(4) 165:1–165:12. doi:10.1145/3197517.3201302.
[4] Aristidou A, Cameron J, Lasenby J. Real-time estimation of missing markers in human motion capture. In: Proceedings of the 2nd international conference on bioinformatics and biomedical engineering. IEEE; 2008. p. 1343–6. ISBN 978-1-4244-1747-6. doi:10.1109/ICBBE.2008.665.
[5] Piazza T, Lundström J, Kunz A, Fjeld M, et al. Predicting missing markers in real-Time optical motion capture. LNCS 2009;5903:125–36.
[6] Alexanderson S, O'Sullivan C, Beskow J. Real-time labeling of non-rigid motion capture marker sets. Comput Graph 2017;69:59–67. doi:10.1016/j.cag.2017.10.001.
[7] Mousas C, Anagnostopoulos C-N. Real-time performance-driven finger motion synthesis. Comput Graph 2017;65:1–11. doi:10.1016/j.cag.2017.03.001.
[8] Moeslund TB, Hilton A, Krüger V. A survey of advances in vision-based human motion capture and analysis. Comput Vision Image Underst 2006;104(2):90–126. doi:10.1016/j.cviu.2006.08.002.
[9] Rafferty K, Nickleson K, Devine S, Herdman C. Improving the ergonomics of hand tracking inputs to VR HMDs. In: Proceedings of the international conferences in central europe on human computer interaction; 2017. Pilsen, Czech Republic
[10] Tkach A, Pauly M, Tagliasacchi A. Sphere-meshes for real-time hand modeling and tracking. ACM Trans Graph 2016;35(6) 222:1–222:11. doi:10.1145/2980179.2980226.
[11] Mueller F, Mehta D, Sotnychenko O, Sridhar S, Casas D, Theobalt C. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. In: Proceedings of the international conference on computer vision (ICCV), Vol. 10; 2017.
[12] Han S, Liu B, Wang R, Ye Y, Twigg CD, Kin K. Online optical marker-based hand tracking with deep labels. In: Proceedings of the 2018 SIGGRAPH; 2018.
[13] Andrews S, Huerta I, Komura T, Sigal L, Mitchell K. Real-time physics-based motion capture with sparse sensors. In: Proceedings of the 13th European conference on visual media production (CVMP). ACM; 2016. p. 5.
[14] Zhou X, Wan Q, Zhang W, Xue X, Wei Y. Model-based deep hand pose estimation. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence, IJCAI'16. AAAI Press; 2016. p. 2421–7. ISBN 978-1-57735-770-4.
[15] Zhou X, Sun X, Zhang W, Liang S, Wei Y. Deep kinematic pose regression. In: Proceedings of the ECCV workshop on geometry meets deep learning. Springer; 2016. p. 186–201.
[16] Aristidou A, Lasenby J, Chrysanthou Y, Shamir A. Inverse kinematics techniques in computer graphics: a survey. Comput Graph Forum 2017;37(6):35–58. doi:10.1111/cgf.13310.
[17] Almusawi ARJ, Dülger LC, Kapucu S. A new artificial neural network approach in solving inverse kinematics of robotic arm (denso VP6242). Comput Intell Neurosci 2016;2016:5720163. doi:10.1155/2016/5720163.
[18] Waegeman T, Schrauwen B. Towards learning inverse kinematics with a neural network based tracking controller. In: Lu B-L, Zhang L, Kwok J, editors. Proceedings of the 2011 neural information processing. Berlin, Heidelberg: Springer; 2011. p. 441–8. ISBN 978-3-642-24965-5.
[19] Kim B-H. An adaptive neural network learning-based solution for the inverse kinematics of humanoid fingers. Int J Adv Rob Syst 2014;11(1):3. doi:10.5772/57472.
[20] Pavllo D, Porssut T, Herbelin B, Boulic R. Real-time finger tracking using active motion capture: a neural network approach robust to occlusions. In: Proceedings of the 11th annual international conference on motion, interaction, and games, MIG '18. New York, NY, USA: ACM; 2018. 6:1–6:10
[21] Kabsch W. A solution for the best rotation to relate two sets of vectors. Acta Crystallogr Sect A 1976;32(5):922–3. doi:10.1107/S0567739476001873.
[22] Hoerl AE, Kennard RW. Ridge regression: biased estimation for nonorthogonal problems. Technometrics 1970;12(1):55–67. doi:10.1080/00401706.1970.10488634.
[23] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Netw 1989;2(5):359–66 arXiv:1011.1669v3. doi:10.1016/0893-6080(89)90020-8.
[24] Jiang K, Chen H, Yuan S. Classification for incomplete data using classifier ensembles. In: Proceedings of the 2005 international conference on neural networks and brain, (ICNN&B 05), 1; 2005. p. 559–63. doi:10.1109/ICNNB.2005.1614675.
[25] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on international conference on machine learning, ICML'10. USA: Omnipress; 2010. p. 807–14. ISBN 978-1-60558-907-7.
[26] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics; 2011. p. 315–23.
[27] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th international conference on neural information processing systems – Volume 1, NIPS'12. USA: Curran Associates Inc.; 2012. p. 1097–105.
[28] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R, et al. Dropout: a Simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58 arXiv:1102.4807. doi:10.1214/12-AOS1000.
[29] Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybern 1988;59(4–5):291–4. doi:10.1007/BF00332918.
[30] Grassia FS. Practical parameterization of rotations using the exponential map. J Graph Tools 1998;3(3):29–48.
[31] Kingma DP, Ba J. Adam: a method for stochastic optimization. In: Proceedings of the 3rd international conference on learning representations (ICLR); 2014. arXiv:1412.6980.
[32] Schröder M, Maycock J, Botsch M. Reduced marker layouts for optical motion capture of hands. In: Proceedings of the 8th ACM SIGGRAPH conference on motion in games, MIG '15. New York, NY, USA: ACM; 2015. p. 7–16. doi:10.1145/2822013.2822026.
[33] Molla E, Galvan Debarba H, Boulic R. Egocentric mapping of body surface constraints. IEEE Trans Vis Comput Graph 2017:1. doi:10.1109/TVCG.2017.2708083.