IET Computer Vision

ORIGINAL RESEARCH PAPER

End-to-end global to local convolutional neural network learning for hand pose recovery in depth data

Meysam Madadi[1,2] ⓘ | Sergio Escalera[1,3] | Xavier Baró[1,4] | Jordi Gonzàlez[1,2]

[1]HuPBA Lab, Computer Vision Center, Bellaterra (Barcelona), Spain

[2]Department of Computer Science, Universitat Autònoma de Barcelona (UAB), Bellaterra, Spain

[3]Department of Mathematics and Informatics, Universitat de Barcelona, Barcelona, Spain

[4]Faculty of Computer Science, Multimedia and Telecommunications, Universitat Oberta de Catalunya, Barcelona, Spain

Correspondence

Meysam Madadi, Computer Vision Center, Edifici O, Campus UAB, 08193 Bellaterra (Barcelona), Spain.
Email: mmadadi@cvc.uab.es

Funding information

MINECO/FEDER, Grant/Award Numbers: PID2019-105093GB-I00, PID2020-120611RB-I00, RTI2018-095232-B-C22, TIN2015-65464-R; CERCA Programme/Generalitat de Catalunya; ICREA

Abstract

Despite recent advances in 3-D pose estimation of human hands, thanks to the advent of convolutional neural networks (CNNs) and depth cameras, this task is still far from being solved in uncontrolled setups. This is mainly due to the highly non-linear dynamics of fingers and self-occlusions, which make hand model training a challenging task. In this study, a novel hierarchical tree-like structured CNN is exploited, in which branches are trained to become specialised in predefined subsets of hand joints called local poses. Further, local pose features, extracted from hierarchical CNN branches, are fused to learn higher order dependencies among joints in the final pose by end-to-end training. Lastly, the loss function used is also defined to incorporate appearance and physical constraints about doable hand motions and deformations. Finally, a non-rigid data augmentation approach is introduced to increase the amount of training depth data. Experimental results suggest that feeding a tree-shaped CNN, specialised in local poses, into a fusion network for modelling joints' correlations and dependencies, helps to increase the precision of final estimations, showing competitive results on NYU, MSRA, Hands17 and SyntheticHand datasets.

KEYWORDS

computer vision, data acquisition, human computer interaction, learning (artificial intelligence), pose estimation

1 | INTRODUCTION

Recently, hand pose recovery attracted special attention, thanks to the availability of low cost depth cameras, like Microsoft Kinect [1–16]. Unsurprisingly, 3-D hand pose estimation plays an important role in most human-computer interaction application scenarios, like social robotics and virtual immersive environments [17]. Despite impressive pose estimation improvements, thanks to the use of convolutional neural networks (CNNs) and depth cameras, 3-D hand pose recovery still faces some challenges before becoming fully operational in uncontrolled environments with fast hand/fingers motion, self-occlusions, noise, and low resolution [18].

Two main strategies have been proposed in the literature for addressing the aforementioned challenges: Model-based and data-driven approaches. Model-based generative approaches fit a predefined 3-D hand model to the depth image

[4, 6, 19–22]. However, as a many-to-one problem, accurate initialisation is critical; besides, the use of global objective functions might not convey accurate results in case of self-occlusions of fingers.

Alternatively, the so-called data-driven approaches consider the available training data to directly learn hand pose from appearance [2, 8–10, 12–16, 23, 24]. Data-driven approaches for hand pose estimation have benefitted from recent advances on CNNs. Convolutional neural networks, as in many other computer vision tasks, have been successfully applied in data-driven hand-pose recovery approaches either for heat-map regression of discrete outputs (corresponding to joint estimation probabilities) [3, 16, 25] or direct regression of continuous outputs (corresponding to joint locations) [5, 12–14, 26, 27]. On the one hand, heat-map regression models require additional optimisation time for computing the likelihood of a joint being located at a particular spatial region. Also, these methods

are subject to propagate errors when mapping images to the final joint space. On the other hand, a main issue with CNNs as direct regression models is how to deal with high non-linear output spaces, since models that are too complex jeopardise generalisation. Indeed for CNNs, learning suitable features (i.e., with good generalisation and discrimination properties) in highly non-linear spaces, while taking into account the structure and dependencies among parameters, is still a challenging task.

In this study, direct regression of the 3-D hand pose is implemented as a specific tree-shaped CNN architecture designed to avoid training a coarse global hand motion model but allowing instead finer local specialisations for different fingers and hand regions. So we break the hand pose estimation problem into hierarchical optimisation subtasks, each one focussed on a specific finger and hand region. Combined together in a tree-like structure, the final CNN shows fast convergence rates due to computations applied at a local level. In addition, we model correlated motion among fingers by fusing the features, learned in the hierarchy, through fully connected layers and training the whole network in an end-to-end fashion. The main advantage of this strategy is that the 3-D hand pose prediction problem is attained as a global learning task based on local estimations.

Moreover, it has been proved that $L2$ loss, in regression problems, is sensitive to outliers and ground-truth noise [28]. Therefore, in order to further improve the final estimation in high non-linear spaces of hand configurations, we incorporate appearance and physical penalties in the loss function, based on the physical constraints typically applied in 3-D reconstruction of human poses [29]. By including such penalties during the network learning stage, unrealistic pose configurations are avoided. We qualitatively compare state-of-the-art pose estimation approaches with respect to ours in Figure 1.

Lastly, as it is common in deep learning problems, variability and amount of data defines the success of a model and its generalisation to unseen data. In this study, we introduce a non-rigid augmentation approach to generate realistic data from training data. To the best of our knowledge, this is the first time such augmentation is being applied in depth images. We use ground-truth joints to compute hand kinematic parameters and deform hand joints. We then apply interpolation techniques to deform point cloud based on the joints. Results demonstrate that our proposed framework trained on augmented data is competitive against state-of-the-art approaches in NYU and MSRA datasets.

Our main contributions are as follows:

- We propose a tree-structured network for the 3-D hand pose estimation problem. In the proposed architecture, localised features are learned in each branch for each finger. Such independent features are fused at the end to learn a global interconnection of hand parts.
- We also propose a novel loss function, helping the network to be aware of appearance and hand dynamics. The appearance criterion is not differentiable. Therefore, we propose an estimation of its gradients in backpropagation.
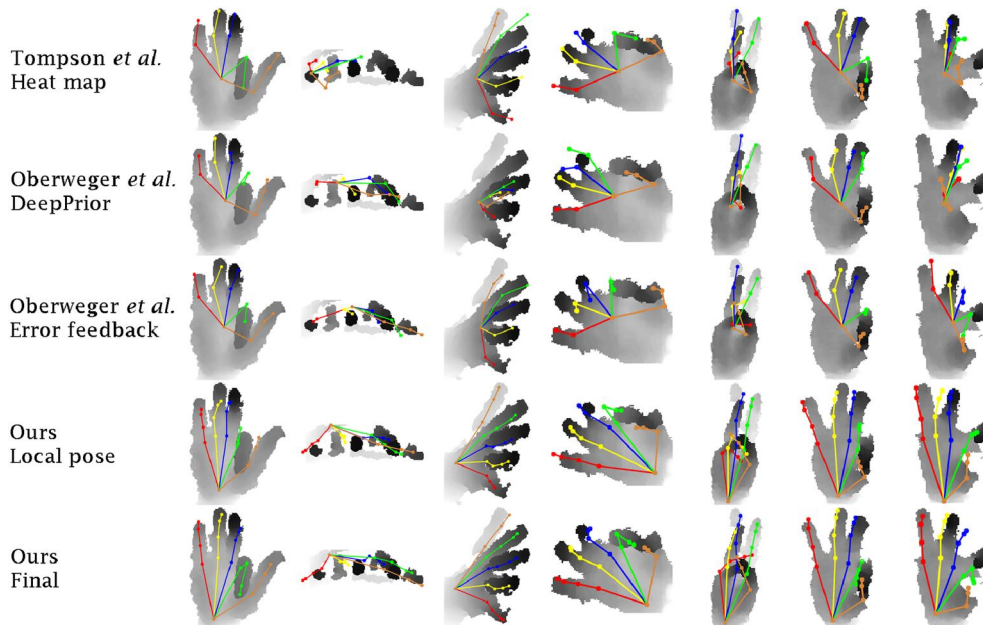
FIGURE 1 Qualitative comparison of our proposed approach versus state-of-the-art methods: [5, 25, 26]. The reported results on the state-of-the-art methods are publicly available in their web pages. The work of Tompson et al. [25] estimates a 2-D pose using a joints' heat map only, thus providing poor pose estimation results in the case of noisy input images (second column). Oberweger et al.'s [26] results (DeepPrior) show that principle component analysis is not able to properly model hand pose configurations. Oberweger et al. [5] improved previous results by applying an error feedback loop approach. However, error feedbacks do not provide accurate pose recovery for all the variability of hand poses. In essence, in our proposed local-based pose estimation framework, a separate network is trained for each finger. Subsequently, we fuse such learned local features to include higher order dependencies among joints, thus obtaining better pose estimation results than previous approaches

- We propose a novel non-rigid data augmentation approach to boost network performance. By employing this augmentation approach, the network can better generalise to the test set. Our experiments on the NYU dataset shows effectiveness of this idea performing well in practice.

2 | RELATED WORK

Hand pose estimation has been extensively studied in the literature [30]; we refer the reader to [19] for a complete classification of state-of-the-art works in the field. Here, we focus mostly on recent works using CNNs and depth cameras.

Most CNN-based architectures in data-driven hand pose estimation approaches are specifically designed to be discriminative and generalisable. Although the success of such approaches depends on the availability and variability of training data, CNN models cope reasonably well with this problem, and two main families of approaches can be distinguished in the literature, namely heat-map and direct regression methods.

Heat-map approaches estimate likelihoods of joints for each pixel/voxel/point as a pre-processing step. In [25], a CNN is fed with multi resolution input images and one heat map per joint is generated. Subsequently, an inverse kinematic model is applied on such heat maps to recover the hand pose. Nevertheless, this approach is prone to propagate errors when mapping to the original image, and estimated joints may not correlate with the hand physics constraints. The work of [3] extends this strategy by applying multi-view fusion of extracted heat maps, where 3-D joints are recovered from only three different viewpoints. In this approach, erroneous heat maps are expected to be improved in the fusion step using complementary viewpoints. The key idea in this work is to reduce the complexity of input data by aligning all data with respect to the hand point cloud eigenvectors. Recently, researchers paid attention to 3-D heat-maps and 3-D alternatives to depth image as input to the network (i.e., voxels [16] or point clouds [31]). In this regard, it is shown that a CNN can be effectively made to learn to correspond input 3-D points to output dense 3-D probabilities of joints.

As an alternative, a number of works propose direct regression for estimating the joint positions of the 3-D hand pose based on image features [5, 10, 12–14, 26, 27]. As mentioned in [32], contrary to heat-map based methods, hand pose regression can better handle the increase in complexity of modelling highly non-linear spaces. Although some approaches propose principle component analysis to reduce the pose space [3, 26], such linear methods typically fail when dealing with large pose and appearance variabilities produced by different viewpoints (as shown in Figure 1).

Recently, error feedback [5, 12, 33], cascading [10] and global-local [9, 10, 27, 34] approaches have proven to avoid local minima by iterative error reduction. Authors in [5] propose to train a generative network of depth images by iteratively improving an initial guess. In this sense, Neverova et al. [35] use hand segmentation as an intermediate representation to enrich pose estimation with iterative multi-task learning. Also, the method proposed in [10] divides the global hand pose problem into local estimations of palm pose and finger poses. Thus, finger locations can be updated at each iteration relative to the hand palm. Contrary to our method, the authors use a cascade of classifiers to combine such local estimations.

Authors in [27] apply a CNN to make use of the resulting feature maps as the descriptors for computing k-nearest shapes. Similarly to our approach, in their method the CNN separates the palm and fingers and computes the final descriptor by dimensionality reduction. Different from our approach, they factorise the feature vectors and nearest neighbours hyperparameters to estimate the hand pose. In a different way, we propose training the network by fusing local features to avoid non-accurate local solutions without the need of introducing cascading strategies or multi-view setups. Contrary to the methods trying to simplify the problem by dividing the output space into subspaces, Guo et al. [12] divided the input image to smaller overlapping regions and fused CNN feature maps as a region ensemble network.

In CNN-based methods, data augmentation is a common approach to boost the network to generalise better. Ge et al. [36] were the first to apply data augmentation in the problem of hand pose recovery and showed a meaningful improvement in the results. Even Oberweger and Lepetit [14] extended the DeepPrior model in [26] and showed the effectiveness of a simple model trained with data augmentation. However, the aforementioned approaches use simple and rigid data augmentation like scaling, rotation and translation, which may not represent the visual variability in terms of 3-D articulated joints. Here, we propose a non-rigid data augmentation by deforming hand parameters and interpolating point cloud.

Global-local solutions or applying physical constraints have been explored in domains other than hand pose estimation as well [37–39]. In the object detection domain, Felzenszwalb et al. [37] aggregated score maps generated by applying global and local object templates. Object templates can be seen as convolutional filters. In this sense, we perform similarly by learning specific filters with respect to the local parts in the tree-structure network. In the 2-D human body pose estimation domain, Chu et al. [38] applied higher order joint relationships by designing special message passing layers among joint feature maps with respect to the body skeleton. This is different from our approach in which we apply a simple multi-layer perceptron network on top of fused features. Finally, physical constraints on joints have been used before as well in the domain of weakly supervised 3-D body pose estimation [39]. In this approach, a sample without 3-D ground truth data is regularised by the body symmetry between left-right parts with respect to the limbs' length. However, this is not applicable in the hand domain since the hand does not have a symmetrical geometry.

3 | GLOBAL HAND POSE RECOVERY FROM LOCAL ESTIMATIONS

Given an input depth image $\mathcal{I}$, our goal is to estimate the 3-D locations of n hand joints as the set $J = \{j \in \mathbb{R}^3\}_1^n$. We define $n = 20$ for the wrist, finger joints and finger tips, following the hand model defined in [10]. We assume that a hand is initially visible in the depth image, that is not occluded by other objects in the scene, although it may present self-occlusions and has properly been detected beforehand (i.e., pixels belonging to the hand are already segmented [25]). We also assume that intrinsic camera parameters are available. We refer to the global pose as the whole set J, while a local pose is a subset of J (e.g., index finger joints).

The hand pose space is a highly non-linear manifold. In the literature, different architectures have been proposed to deal with the data complexity [5, 32, 40]. For example, in multi-task learning, different branching strategies are typically applied to solve subproblems [41, 42], and the different subproblems are solved jointly by sharing features. Similarly, considering hand pose recovery as a regression problem, we divide the global hand pose into simpler local poses and solve each local pose separately in a branch by means of a tree-shaped network. In such a design, each network branch is specialised in a local pose. We show this architecture in Figure 2.

We define the amount of locality by the number of joints contributing to a local pose. On the one hand, keeping such locality high (i.e., lower number of joints) causes fingers to be easily confused among each other or detected in a physically impossible location. On the other hand, a low locality value (i.e., higher number of joints) increases the complexity. Besides, local joints should share a similar motion pattern to maintain lower complexity. So we follow the hand kinematic tree in the particular implementation in this study and assign to each local pose one finger plus palm joints. This means eight joints per branch, thus leading to a 24 (=3 × 8) dimensional vector as the branch output.[1]

The proposed architecture has several advantages. Firstly, correlated fingers share features in earlier layers. By doing this, we allow the network to hierarchically learn more specific features for each finger with respect to its most correlated fingers. Secondly, the number of filters per finger can be adaptively determined. Thirdly, the estimation of the global pose is reduced to the estimations of simpler local poses, causing the network to train at fast convergence rates.

Training the network only based on local poses omits information about inter-fingers relations. Tompson et al. [43] included a graphical model within the training process to formulate joints' relationships. Li et al. [44] used a dot product to compute similarities for embedded spaces of a given pose and an estimated one in a structural learning strategy. Instead, we apply late fusion based on local features; thus, the network learns the joint dependencies through fully connected layers for estimating the final global pose. The whole network is trained end-to-end jointly for all global and local poses, given a constrained loss function (see Section 3.2).

3.1 | Hand pose estimation architecture

We preprocess depth images before feeding the network, that is, the hand area is cropped and normalised. To do so, a fixed size window in 3-D coordinates, perpendicular to the image plane and centred on the hand mean point, is projected on the image plane. Subsequently, the resulting window is cropped and resized to a 192 × 192 fixed size image using nearest neighbour interpolation. Then, the mean depth is subtracted from the hand pixel values. Similarly, the hand mean point is subtracted from ground-truth joints to have a translation invariant network.

As intermediate layers, the network is composed of six *branches*, where each branch is associated with specific fingers as follows: two branches for index and middle fingers, two branches for ring and pinky fingers, one branch for thumb, and one branch for palm. In the palm branch, we assume that the palm is a rigid object. Therefore, it can be estimated by a linear transformation of a reference palm. Let $\hat{J}_P \in \mathbb{R}^{3 \times 5}$ be the set of reference palm joints in a canonical viewpoint (e.g., frontal upright hand). We set $\hat{J}_P$ as the average palm joints of all training samples after transforming them to the canonical viewpoint. Then, palm joints $J_P \in \mathbb{R}^{3 \times 5}$ can be reconstructed by $b \times R \times \hat{J}_P + c$, where b is a scaling factor, R is a rotation matrix and c is a translation vector. As shown in the results, regressing b, R and c leads to more accurate results than directly regressing J_P. We compute the ground-truth values for b, R and c by Procrustes analysis. We also convert R to quaternions $Q \in \mathbb{R}^4$ to make sure we always estimate a valid rotation. Another reason why we use the palm branch is to provide more intermediate features for the global pose regression.

As shown in Figure 2, each convolutional block consists of a convolution layer with 3 × 3 filter kernels and a ReLU followed by a max-pooling layer, except for the last block. All pooling layers contain a 2 × 2 window. The last block contains a convolutional layer with 6 × 6 filter kernels, providing a feature vector. Fully connected layers are added to the end of each branch for both local and global pose learning. For the local pose at each branch, there are two hidden layers with 1024 neurons with a dropout layer in between. Similarly, for the global pose at each branch, the feature vector is followed by two hidden layers with 1024 neurons with a dropout layer in between. Then, the last hidden layers are concatenated, followed by a dropout and a hidden layer with 1024 neurons. Finally, the global and local output layers provide the estimation of joints with one neuron per joint and dimension.

3.2 | Constraints as loss function

In regression problems, the goal is to optimise parameters such that a loss function between the estimated values of the network and the ground-truth value gets minimised. Usually, in

---

[1] Note palm is overlapped in all local poses for the stability of the network estimations.
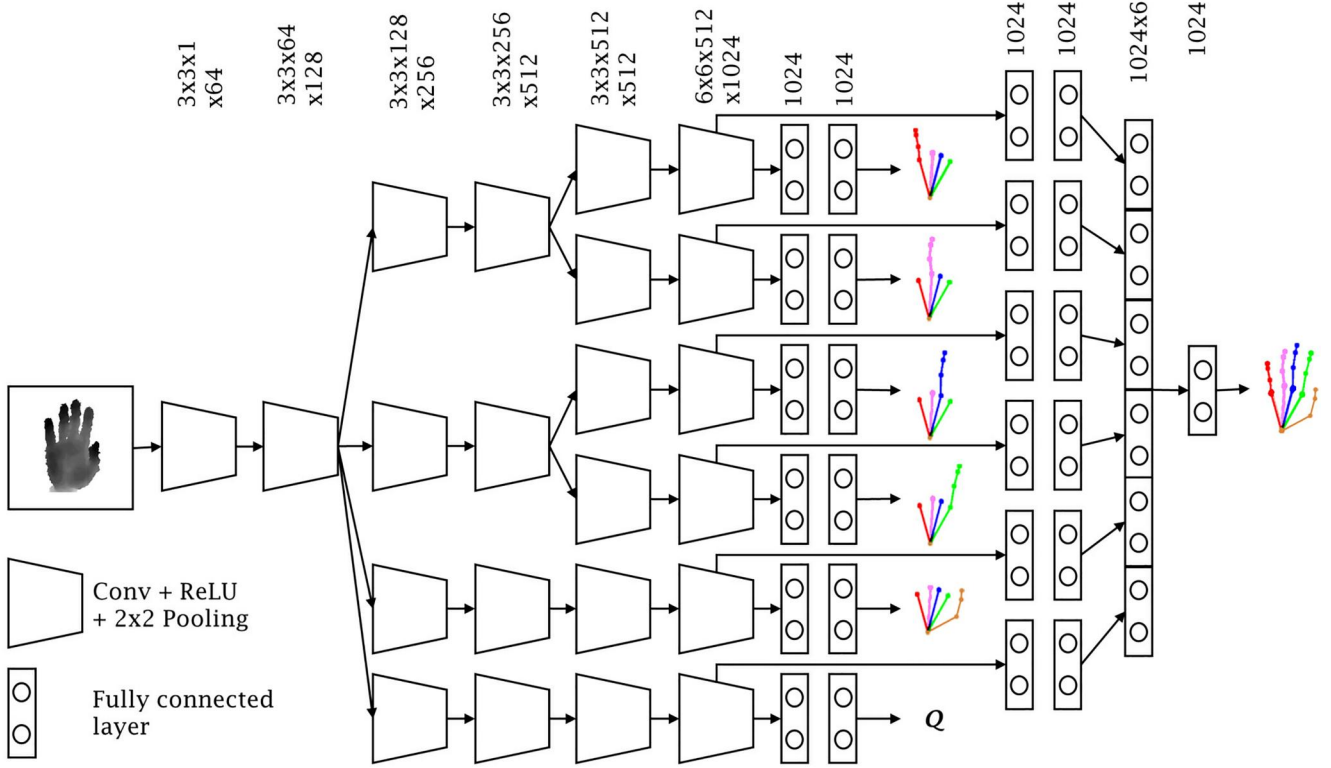
FIGURE 2 Proposed network architecture. The branching strategy connects convolutional neural network blocks into a tree-shaped structure while regressing the local pose at each branch. Each local pose is a 24 dimensional vector. We also include an extra branch in the network to regress viewpoint Q, which is a quaternion rotation matrix. We then fuse all the features of the last convolutional layers to estimate the output global pose. We use Q features in the fusion to extract palm joints more accurate

the training procedure, an $L2$ loss function plus a regularisation term is optimised. However, it is generally known that, in an unbalanced dataset with availability of outliers, $L2$ norm minimisation can result in poor generalisation and sensitivity to outliers where equal weights are given to the training data [28]. Weight regularisation is commonly used in deep learning as a way to avoid overfitting. However, it does not guarantee weight updating to bypass the local minima. Besides, a high weight decay causes low convergence rates. Belagiannis et al. [28] proposed Tukey's biweight loss function in the regression problems as an alternative to $L2$ loss robust against outliers. We formulate the loss function as $L2$ loss along with constraints applied to hand joints regarding the hand dynamics and appearance, leading to more accurate results and less sensitivity to ground-truth noise. We define the loss function for one frame in the form of

$$L = \lambda_1 L_{loc} + \lambda_2 L_{glo} + \lambda_3 L_Q + \lambda_4 L_{app} + \lambda_5 L_{dyn}, \quad (1)$$

where λ_i $i \in \{1 .. 5\}$ are factors to balance loss functions. L_{loc}, L_{glo}, L_Q, L_{app} and L_{dyn} denote the loss for the estimated local and global pose, palm quaternion, appearance, and hand dynamics, respectively. Next, each component is explained in detail.

Let $F^l \in \mathbb{R}^{3 \times m}$ be the concatenation of the m estimated local joints in each branch of the proposed network and $G^l \in \mathbb{R}^{3 \times m}$ be the ground-truth matrix. Note that m is not necessarily equal to $n = 20$. Also, let $F^g \in \mathbb{R}^{3 \times n}$ and

$G^g \in \mathbb{R}^{3 \times n}$ be the outputs of the fusion network for the estimated global joints and ground truth, respectively. Finally, let $\hat{Q} \in \mathbb{R}^4$ be the ground truth quaternion. Then, we define local, global and quaternion[2] losses as follows:

$$L_{loc} = \sum_{i=1}^{3m} \left(F_i^l - G_i^l \right)^2, \quad (2)$$

$$L_{glo} = \sum_{i=1}^{3n} \left(F_i^g - G_i^g \right)^2, \quad (3)$$

$$L_Q = \sum_{i=1}^{4} \left(Q_i - \hat{Q}_i \right)^2. \quad (4)$$

A common problem in CNN-based methods for pose estimation is that in some situations the estimated pose does not properly fit with the appearance. For instance, joints are predicted in physically incorrect locations where there is no evidence of the presence of hand points [3, 5, 26]. We show examples of such cases in Figure 9b,c. In this study, during training, we penalise those joint estimations that do not fit with the appearance or are physically not possible and include such penalties in the loss function.

---

[2]Note that we compute b and c by the aid of F^g.

We first assume that, rationally, joints must be located inside the hand area and have a depth value higher than the hand surface. Besides, joints must present physically possible angles in the kinematic tree. We denote the model output as $\hat{j}^{xyz}$ in the world coordinate system and show its projection to the image plane by $\hat{j}^{uvz}$. Then, for a given joint $\hat{j}^{xyz}$ the inequality $\mathcal{I}(\hat{j}^u, \hat{j}^v) - \hat{j}^z < 0$ must hold, where $\mathcal{I}(\hat{j}^u, \hat{j}^v)$ is the pixel value at location $(\hat{j}^u, \hat{j}^v)$. To avoid violating the first condition (i.e., when a joint is located outside the hand area after projection to the image plane), we set the background with a cone function as follows:

$$5\sqrt{(u - 0.5w)^2 + (v - 0.5h)^2} + \phi,$$

where w and h are the width and height of the image and ϕ is a fixed value set to 100. The reason for using a cone function instead of a fixed large value is to avoid zero derivatives on the background. We use hinge formulation to convert inequality to a loss through the following:

$$L_{app} = \sum_{i=1}^{m} \max(0, \mathcal{I}(\hat{j}_i^u, \hat{j}_i^v) - \hat{j}_i^z). \tag{5}$$

We subsequently incorporate the hand dynamics by means of the top-down strategy described in Algorithm 1. We assume that all joints belonging to each finger (except thumb) should be collinear or coplanar. The thumb has an extra non-coplanar form and we do not consider it in the hand dynamics loss. A ground-truth finger state $s_G \in \{1 .. 4\}$ is assigned to each finger computed by the conditions defined in Algorithm 1. Each finger has a ground-truth normal vector $\mathbf{e}_G$, which is the finger direction for case 1 and the finger plane normal vector for the other cases. Therefore, we define four different losses, one of them triggered for each finger (as shown in Algorithm 1). Let A, B, C and D be four joints belonging to a finger starting in A as the root joint and ending in D as the fingertip. Then the dynamics loss is defined as follows:

$$L_{dyn} = \sum_{i=1}^{4} \Delta_i(A, B, C, D, s_G, \mathbf{e}_G), \tag{6}$$

where i denotes a finger index. Now we consider each case in Algorithm 1 in the following:

We consider a collinear finger in case 1. A finger is collinear if

$$\|B - A\| + \|C - B\| + \|D - C\| < \|D - A\| + \kappa,$$

where κ is a threshold defining the amount of collinearity and set to $0.01\|D - A\|$. To compute the loss for a collinear ground-truth finger, the following condition has to hold: $\rho < \cos(\angle(\overrightarrow{AD}, \mathbf{e}_G)) \leq 1$, where ρ is a threshold set to 0.9962 for all the experiments, found experimentally. This condition has to be met for $\overrightarrow{AB}$ and $\overrightarrow{AC}$ as well. The cosine function can be extracted through the dot product. Therefore, using hinge formulation, the loss is defined as

$$\Delta_i(A, B, C, D, 1, \mathbf{e}_G) =$$

$$\max\left(0, \rho - \frac{\overrightarrow{AB} \cdot \mathbf{e}_G}{\|\overrightarrow{AB}\|}\right) +$$

$$\max\left(0, \rho - \frac{\overrightarrow{AC} \cdot \mathbf{e}_G}{\|\overrightarrow{AC}\|}\right) +$$

$$\max\left(0, \rho - \frac{\overrightarrow{AD} \cdot \mathbf{e}_G}{\|\overrightarrow{AD}\|}\right) +$$

$$\mu \max(0, \|\overrightarrow{AB}\| + \|\overrightarrow{BC}\| + \|\overrightarrow{CD}\| - 1.01\|\overrightarrow{AD}\|), \tag{7}$$

where μ is a factor to balance the different components of the loss function.

Algorithm 1 Top-down strategy for finger dynamics

---

input: normal vector $\mathbf{e}_G$ belonging to groundtruth defining either finger direction or finger plane normal
input: finger joints A, B, C and D
  1: /**** $a//b$ means vector a is parallel to b ****/
  2: **if** $\overrightarrow{AB}//\overrightarrow{AC}//\overrightarrow{AD}//\mathbf{e}_G$ **then**

  3:      $s_G \leftarrow 1$ ▷
  4: **else if** $\overrightarrow{AB} \times \overrightarrow{BC}//\overrightarrow{AC} \times \overrightarrow{CD}//\mathbf{e}_G$ **then**

  5:      $s_G \leftarrow 2$ ▷
  6: **else if** $\overrightarrow{AB} \times \overrightarrow{BC}//\overrightarrow{BC} \times \overrightarrow{CD}//\mathbf{e}_G$ **then**
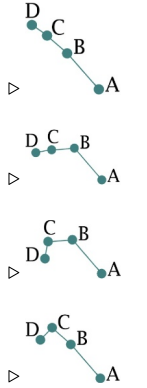
  7:      $s_G \leftarrow 3$ ▷
  8: **else if** $\overrightarrow{AB} \times \overrightarrow{BD}//\overrightarrow{BC} \times \overrightarrow{CD}//\mathbf{e}_G$ **then**

  9:      $s_G \leftarrow 4$ ▷
10: **end if**
11: **return** $\Delta(A, B, C, D, s_G, \mathbf{e}_G)$

We consider a coplanar finger for cases 2, 3 and 4. We define a finger as coplanar if the cross products of all subsets of the finger joints with three members are parallel. Note that a collinear finger is necessarily coplanar. However, we exclude collinear fingers from this definition due to cross-product ambiguity, as shown in Algorithm 1. For a ground-truth coplanar finger, such cross products must be parallel to the plane normal vector. Therefore, for the given joints A, B and C, the following condition must hold:

$$\rho < \cos\left(\angle(\overrightarrow{AB} \times \overrightarrow{BC}, \mathbf{e}_G)\right) \leq 1.$$

Given that the ground-truth finger is coplanar for case 2, we compute the loss function as follows:

$$\Delta_i(A, B, C, D, 2, \mathbf{e}_G) = \max\left(0, \rho - \frac{(\overrightarrow{AB} \times \overrightarrow{BC}) \cdot \mathbf{e}_G}{\|\overrightarrow{AB} \times \overrightarrow{BC}\|}\right)$$
$$+ \max\left(0, \rho - \frac{(\overrightarrow{AC} \times \overrightarrow{CD}) \cdot \mathbf{e}_G}{\|\overrightarrow{AC} \times \overrightarrow{CD}\|}\right). \quad (8)$$

The loss functions for the other coplanar finger cases are computed in the same way.

3.3 | Loss function derivatives

All components in Equation (1) are differentiable; thus we are able to use gradient-based optimisation methods. In this section, we explain derivatives of the constraint loss function in Equation (5). Derivatives of the rest of the loss functions are computed through matrix calculations. We first define the derivative of L_{app} with respect to $t \in \{j_i^x, j_i^y, j_i^z\}$ through the following:

$$\frac{\partial L_{app}}{\partial t} = \begin{cases} 0 & \text{if } \mathcal{I}(j_i^u, j_i^v) - j_i^z \leq 0 \\ \partial \mathcal{I}/\partial t - \partial j_i^z/\partial t & \text{otherwise.} \end{cases} \quad (9)$$

In the following, we just consider the positive condition of Equation (9). Besides, we omit index i (which denotes i-th joint) from the notations for the ease of reading. Depth image $\mathcal{I}$ is a discrete multi-variable function of j^u and j^v, where j^u is a multi-variable function of j^x and j^z, and j^v is a multi-variable function of j^y and j^z. Consequently, the total derivative of a depth image can be computed by the chain rule through the following:

$$\frac{d\mathcal{I}}{dt} = \frac{\partial \mathcal{I}}{\partial j^u}\frac{dj^u}{dt} + \frac{\partial \mathcal{I}}{\partial j^v}\frac{dj^v}{dt} \quad (10)$$

$$\frac{dj^u}{dt} = \frac{\partial j^u}{\partial j^x}\frac{dj^x}{dt} + \frac{\partial j^u}{\partial j^z}\frac{dj^z}{dt} \quad (11)$$

$$\frac{dj^v}{dt} = \frac{\partial j^v}{\partial j^y}\frac{dj^y}{dt} + \frac{\partial j^v}{\partial j^z}\frac{dj^z}{dt} \quad (12)$$

Next, we present components of j^u derivative in detail.[3] Depth image $\mathcal{I}$ is a function of the hand surface. However, the hand surface given by the depth camera may have noise and may not be differentiable at some points. To cope with this problem, we estimate depth image derivatives by applying hand surface normal vectors. Let $\mathbf{s}$ be the surface normal vector for a given joint. Then, derivative of $\mathcal{I}$ with respect to u axis is given by the tangent vectors through the following:

$$\frac{\partial \mathcal{I}}{\partial j^u} = \frac{\mathbf{s}^x}{\mathbf{s}^z}. \quad (13)$$

As mentioned, j^{uvz} is the projection of the estimated joint j^{xyz} from the world coordinate to the image plane. Note that joints have zero mean and j^{uvz} is extracted after the image has been cropped and resized. Let f_x, p_x, M^{xyz} and M^{uvz} be the camera focal length and image centre for x axis, world coordinate hand point cloud centre, and its projection to the image plane, respectively. Then, j^u is computed as follows:

$$j^u(j^x, j^z) = \left(\frac{f_x(j^x + M^x)}{j^z + M^z} + p_x - M^u\right)\text{scale}_x + \frac{w}{2},$$
$$\text{scale}_x = \frac{wM^z}{cf_x}, \quad (14)$$

where c is the cube size used around the hand point cloud to crop the hand image. Using this formulation, the derivative of j^u can be easily computed and replaced in Equation (11).

4 | EXPERIMENTS

In this section, we evaluate our approach on three real-world datasets NYU [25], MSRA [10] and Hands17 [45], and one synthetic dataset SyntheticHand [46]. The NYU dataset has around 73k annotated frames as training data (single subject) and 8k frames as test data (two subjects). Each frame has been captured from three different viewpoints and the ground truth is almost accurate. The MSRA dataset has 76k frames captured from nine subjects each in 17 pose categories. This dataset does not provide an explicit training/test set and a subject exclusive nine-fold cross validation is used to train and evaluate this dataset. The MSRA dataset has smaller image resolution and less pose diversity and accurate ground truth compared to the NYU dataset. The Hands17 dataset is a large scale dataset with 957k training and 296k test data. This is a challenging dataset with diverse viewpoints and number of subjects. The data is captured from the third and egocentric views. The training data contains five subjects while the test data has five unseen subjects plus training subjects. Finally, the Synthetic-Hand dataset has over 700k training data and 8k test data consisting of a single synthetic subject performing random poses from all viewpoints, thus being useful for the analysis of our methodology under occlusions. All three datasets have at least 20 hand joints in common. However, the NYU dataset has 16 extra joints.

We evaluate our approach using two metrics: average Euclidean distance in mm for all the joints with respect to the ground-truth joints and success rate error [47], which evaluates the proportion of test images with their maximum joint error within a threshold. Next, we provide the details of the method parameters and evaluate our approach both quantitatively and qualitatively in comparison to state-of-the-art alternatives.

[3]Derivatives belonging to j^v are computed in the same way as j^u.

4.1 | Training

We utilise MatConvNet library [48] on a server with graphics processing unit (GPU) device *GeForce GTX Titan X* with 12 GB memory and Cuda 8. We optimise the network using the stochastic gradient descent algorithm. We report hyperparameters used in the NYU dataset. We set the batch size, learning rate, weight decay and momentum to 50, 0.5e-6, 0.0005 and 0.9, respectively. We set the probability of dropout to 0.5. Our approach converges in almost six epochs while reducing the learning rate by a factor of 10 for two more epochs. Overall, the training takes 2 days on the original NYU dataset while testing takes 50 fps.

Loss function parameters tuning. We set a low value for parameter $\mu(= 0.0005)$ in Equation (7) since it behaves like a regularisation and it is not connected to ground truth. Regarding the ρ, it is the lower bound in the hinge loss, that is Equations (7) and (8). For instance, in Equation (7) $\rho = \cos(\angle_{\max}(\vec{AD}, \mathbf{e}_G))$, where $\angle_{\max}$ is the maximum threshold angle. Increasing this angle makes the lower bound smaller and, thus, Equation (7) becomes less effective due to the hinge loss. We set this maximum angle to 5°. L_{dyn} is mainly a summation of cosine functions while L_{app} is in millimetres. Therefore, we set λ_5 higher than λ_4 to balance the cosine space against millimetre. Finally, we set parameters λ_1, λ_2, λ_3, λ_4 and λ_5 experimentally to 4, 4, 1, 3 and 20, respectively. Note that we train the Q branch isolated from the rest of the network because it converges slower than the other branches. We show derivatives of appearance and dynamics loss functions for a number of joints in the first five epochs in Figure 3 as well as qualitative images of estimated joints.

Data augmentation is a common approach to boost CNN models with small deformations in the images. The mainly used data augmentation approaches are rotation, scaling, stretching and adding random noise to pixels. Such approaches are primarily rigid (in-plane rotation and scaling) or unrealistic (stretching). Here, we apply rigid data augmentation and additionally propose a realistic non-rigid data augmentation approach. Our non-rigid data augmentation consists of the palm size, fingers length and pose modification. The main idea is two folds: (1) deforming ground-truth joints and (2) interpolating point cloud based on new joints. We show some generated images in Figure 4a. In the following section, we explain the details of the proposed data augmentation.

A first possible shape deformation is the change of hand scale. However, simple scaling does not guarantee generalisation to all sorts of hand skeletons, especially when the number of subjects is limited in the training set. Instead, we deform the shape non-rigidly by independently changing the palm size or fingers' length. Some examples can be seen in the fourth row of Figure 4a. As the first step, we define the hand coordinate with the aid of palm joints such that, in a quite open hand, the thumb defines the x coordinate direction, other fingers define the y coordinate direction and the z coordinate is perpendicular to the palm plane. For the ease of computations, we align the hand coordinate to the world coordinate. This is possible
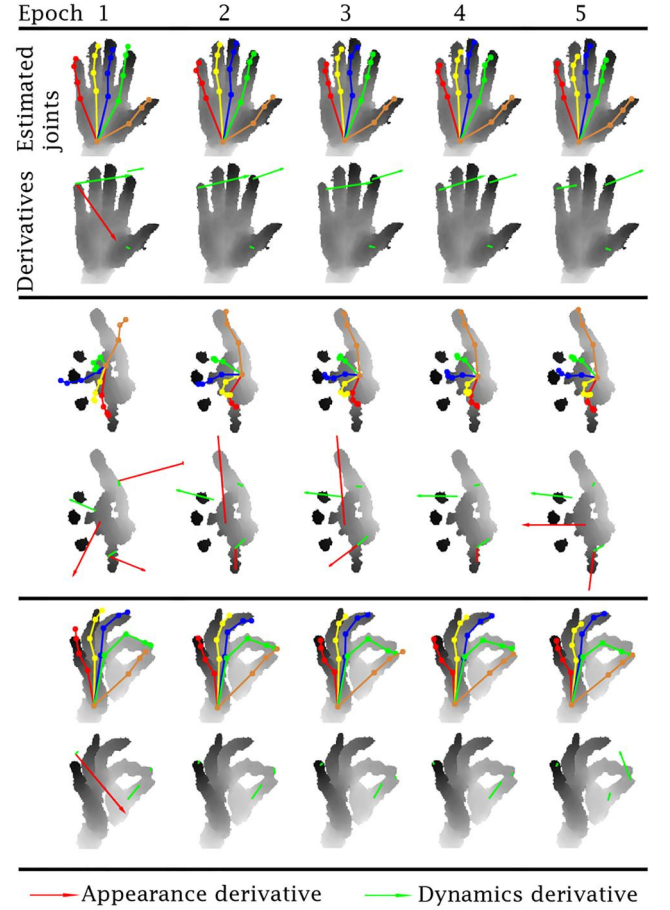

FIGURE 3 Constraints' derivatives during training process (original NYU dataset). Estimated joints along with derivatives of appearance and hand dynamics are illustrated for the first five epochs in the training process. We qualitatively show how the proposed network converges very fast in few epochs

by transforming the joints (and point cloud) into $\hat{J}_P$ by R^T. Then, the palm joints can be stretched in the direction of x or y. We stretch each direction by a random factor. We apply other sorts of skeleton deformations by computing the skeleton kinematic parameters in terms of 19 degrees of freedom (three DoF for thumb and four DoF for the other fingers following [46]). This representation is invariant to size; thus, on having the kinematic parameters fixed, we are able to randomly modify fingers' length and reconstruct new joints for each finger. It is also likely to slightly modify kinematic parameters and reconstruct joints in a new pose. However, we keep kinematic parameters close to the original values to avoid unrealistic point cloud deformations and possible big holes in the depth image.

After deforming the joints, we use thin plate spline (TPS) [49] as a standard interpolation technique to deform the point cloud. We use the original and deformed joints as anchors in the TPS interpolation. However, to avoid extrapolation problems and unrealistic warping, we add some auxiliary points to the set of joints. Auxiliary points are built by adding fixed offsets to certain joints. We show some possible auxiliary points in Figure 4b: we mainly add points around the
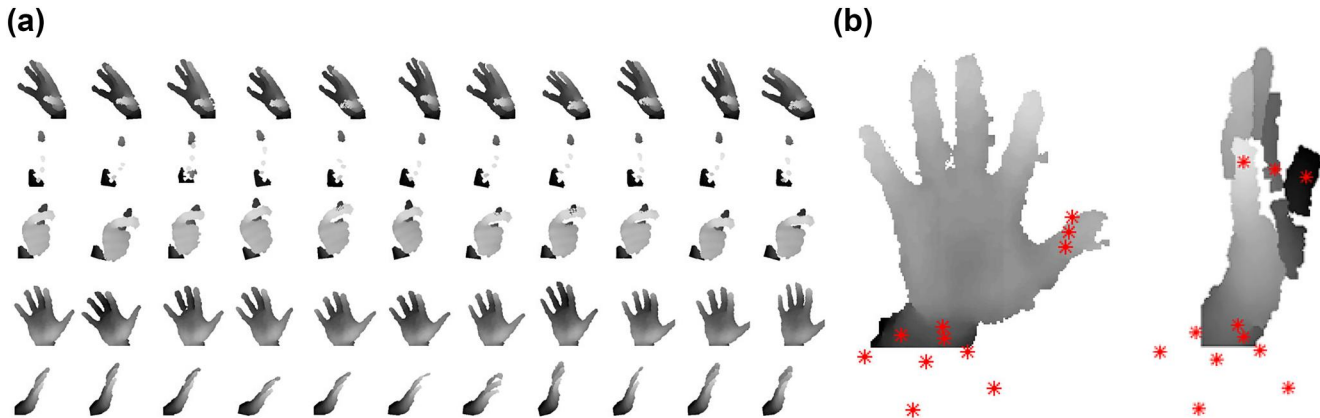
FIGURE 4 Data augmentation. (a) We generate new data by applying non-rigid hand shape deformation along with rigid transformations like in-plane rotation. Hand kinematic parameters are slightly deformed and new hand joints are used to interpolate the hand point cloud. We also change the palm and fingers' size. Therefore, given a pose, different hand shapes can be generated, which helps to generalise better to unseen subjects. The first column shows the original images and others are generated samples. (b) Auxiliary points '*' are added to the set of joints to avoid unrealistic warping in non-rigid hand augmentation

wrist and thumb. We observed unrealistic deformation around the thumb and by adding three fixed points we avoid extrapolation problems. For the wrist case, we do not want to deform the points of the lower arm. Fixed auxiliary points around the wrist add constraints to space, avoiding unrealistic warping. Finally, we project the new point cloud to the image plane, build depth image and apply morphological operations to fill small gaps.

4.2 | Ablation study

In this section, we study different components of the proposed architecture trained on the NYU and Hands17 datasets. We denote each experiment by a number explained in Table 1. All the models are trained without any data augmentation unless it is specifically mentioned.

4.2.1 | Ablation study on NYU dataset

We study the following models on this dataset: *1:local*, *2:1 + palm*, *3:2 + constraint*, *single channel + constraint*, *FC branching + constraint*, *4:3 + viewpoint+fusion*, *5:4 + rigid aug*, *6:4 + aug1*, *7:4 + aug2* and *8:7 + BN*. We also study the accuracy on the occluded joints, Q versus palm joints regression and the robustness against overfitting.

Locality. Locality refers to the number of joints in the network output. In the first case, we analyse the hierarchical network in Figure 5a trained with L_{loc} loss just on one finger in each branch (so called *1:local*). This network shows a high locality value. As one can expect, this network can easily overfit on the training data and exchange estimations for similar fingers. We show a significant improvement by decreasing the locality by including palm joints in each branch (so called *2:1 + palm*). Palm joints are located in a near planar space and thus do not add high non-linearity to the output of each branch,

while helping with better finger localisation. We compare these methods in Figure 6a (red vs. green lines).

Constraints. We train method *2:1 + palm* by including constraints in the loss function: $L_{loc} + L_{app} + L_{dyn}$ (so called *3:2 + constraint*). We still do not explicitly model any relationship among fingers in the output space but let the network learn each finger joints with respect to the hand surface and finger dynamics. In Figure 6a we show the effectiveness of this strategy (magnet line) against method *2:1 + palm*.

Branching strategy versus single-channel architecture. It is also possible to study the impact of the tree-structure network in comparison with single channel architectures. Therefore, we create a single channel network with 6 convolutional layers, as shown in Figure 5b. The output of this network (so called *single-channel* network) is 3-D locations of the full set of joints. In this architecture, the capacity of convolutional layers is kept similar to the whole branching network in Figure 5a by merging parallel layers. This network is trained with $L_{glo} + L_{app} + Ldyn$ loss function. As one can see in Figure 6a, *single-channel* network (dashed magneta line) performs worse than method *3:2 + constraint*, showing the effectiveness of the tree-structure network. Additionally, we trained the single-channel network with the capacity of one branch, which performed 1 mm worse than the full capacity one. This means that regardless of the capacity of the network, in a single-channel network, backpropagation of the gradients of the loss is not able to train network filters to map the input image to a highly non-linear space in an optimal and generalisable solution.

Moreover, we study the impact of branching if applied on FC layers. For this task, we create the network in Figure 5c (so called *FC-branching* network). The outputs of this network and the loss function are similar to method *3:2 + constraint*. The capacity of convolutional layers in this architecture is similar to one branch in Figure 5a. The results of the *FC-branching* network (dashed dark brown line) are even worse than the *single-channel* network in Figure 6a.

TABLE 1 The list of studied ablation experiments

Name	Network	Output	Loss
1:local	Figure 5a	One finger per branch	L_{loc}
2:1 + palm	Figure 5a	One finger plus palm per branch (F^l)	L_{loc}
3:2 + constraint	Figure 5a	F^l	$L_{loc} + L_{app} + L_{dyn}$
single channel + constraint	Figure 5b	All joints (F^g)	$L_{glo} + L_{app} + L_{dyn}$
FC branching + constraint	Figure 5c	F^l	$L_{loc} + L_{app} + L_{dyn}$
4:3+viewpoint + fusion	Figure 2	F^l, Q and F^g	L
4v:4 w/o viewpoint	Figure 2 without Q branch	$\simeq$ method *4* without Q	L without L_Q
4a:4 w/o app loss	Figure 2	$\simeq$ method *4*	L without L_{app}
4d:4 w/o dyn loss	Figure 2	$\simeq$ method *4*	L without L_{dyn}
5:4 + rigid aug	Figure 2 trained with 1500k rigid augmented data	$\simeq$ method *4*	L
6:4 + aug1	Figure 2 trained with 700k non-rigid augmented data	$\simeq$ method *4*	L
7:4 + aug2	Figure 2 trained with 1500k non-rigid augmented data	$\simeq$ method *4*	L
8:7 + BN	Figure 2 including batch normalisation trained same as *7:4+aug2*	$\simeq$ method *4*	L
palm joints regressor	Q branch in Figure 2	Palm joints (J_p)	$L2$ loss

Abbreviation: BN, batch normalisation.

Global versus local pose. We add a fusion network to method *3:2 + constraint* to model correlations among different local poses in an explicit way (so called *4:3 + viewpoint + fusion*). We include Q branch features in the fusion as well. We illustrate the results in Figure 6a (dashed blue line). Compared to method *3:2 + constraint*, method *4:3 + viewpoint + fusion* improves performance for error thresholds below 30 mm.

Data augmentation. As the first step, we remove redundant data by checking the ground-truth joints. In this sense, a redundant data is an image which has a high similarity to at least one image in the training set. Such similarity is defined by the maximum Euclidean distance Ψ among corresponding joints. Therefore, two images are similar if Ψ is below a threshold. We used threshold 10 mm for this task.

In the NYU dataset, around 60k images were left after removing redundant images from all 218k samples in the training set (including all cameras). We then generated two sets of non-rigid augmented images including around 780k and 1500k samples. We used random scaling factors in the range $[0.85, 1.05]$ for the palm and fingers. The kinematic parameters were changed by summation to random degrees in the range $[-7.5, 7.5]$. The only difference in the generated sets is the in-plane rotation degrees. The first and second sets have in-plane rotation in the range $[-30, 30]$ and $[-90, 90]$ degrees, respectively. To show the effectiveness of the proposed non-rigid data augmentation approach, we also generated an extra set of 1500k samples with rigid data augmentation, that is scaling and rotation. We used random scales in the range $[0.8, 1.05]$ and random rotation in the range $[-90, 90]$ degrees.

Therefore, our data distribution is similar to the second non-rigid set.

We train method *4:3 + viewpoint+fusion* on these three new sets, so called *method 5:4 + rigid aug*, *method 6:4 + aug1* and *method 7:4 + aug2*, respectively. We compare the results of these three models in Figure 6a (cyan, brown and dark green lines). One can see that the model trained on the set with more samples and wide in-plane rotation degrees (*method 7:4 + aug2*) generalises better to the test set. Also, a significant improvement is achieved compared to the original data (method *4:3 + viewpoint + fusion*). On comparing rigid and non-rigid data augmentation, one can see a clear advantage of the proposed data augmentation over standard rigid ones. Even our smaller non-rigid set can better generalise to the test set than the rigid set, showing its effectiveness for small error thresholds. Interestingly, we observed that the wrist joint has the maximum error in 20% of the cases in method *7:4 + aug2*. Also, fingertips have the highest error among the joints, which are significantly improved by data augmentation, as we can see in Figure 10a comparing different baselines qualitatively.

Training with batch normalisation (BN) is a common practice in deep learning. We update our model by adding a BN layer after all convolutional layers including fully connected layers. We train the model using the second non-rigid set, so called *method 8:7 + BN*, and learning rate $1e-4$. As a result, *method 8:7 + BN* performs the best, as one can see in Figure 6a (pink line) and 6b. Using the BN layer we improve *7:4 + aug2* fingertips by 2 mm while the overall improvement is 1 mm. However, it slightly worsens the wrist by 1.5 mm.
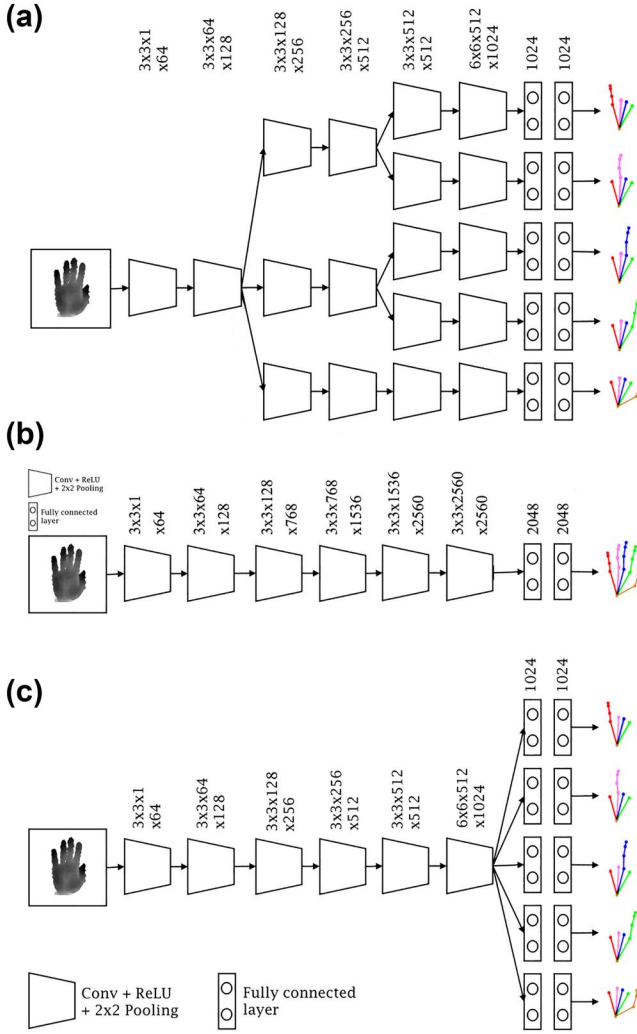
FIGURE 5 Baseline architectures. (a) Tree-structure network without Q branch and fusion layers. (b) A single channel network with the same convolutional capacity as (a). (c) A single channel network with the same capacity as one branch in (a) in which branching is applied on FC layers. We train all the networks with the same loss as in Equation (1), omitting L_Q from all, L_{glo} from (a) and (c), and L_{loc} from (b)

Per joint mean error. We also illustrate the per joint mean error in Figure 6b. It can be observed from the figure that, as expected, a very local solution (method 1) performed the worst among the baselines. Comparing method 2 and *3:2 + constraint* in average error shows the benefits of applying constraints as loss as well. By including viewpoint features in the fusion network, the palm joints' mean error was considerably reduced by method *4:3 + viewpoint + fusion*. Although method *4:3 + viewpoint + fusion* performed better for the pinky and ring fingertips, it did not achieve the best results for index and thumb fingertips.

Palm viewpoint versus palm joints regression. As we explained in Section 3.1, Q can be used to transform the reference palm joints $\hat{J}_P$ and reconstruct the palm joints. Here, we study how accurate the palm viewpoint regression can be to reconstruct the palm joints. Therefore, we build a network with the same architecture as Q branch to regress the palm joints

trained with $L2$ loss. We then evaluate our palm joints versus the palm viewpoint regression in terms of the success rate error in Figure 6c. As can be seen in the figure, the palm viewpoint regression significantly reduces palm joints' error.

Occlusions. To analyse the performance of the occluded joints, we first need to compute which joints are occluded. To do so, we base on [50] to first segment the hand based on the nearest distance of each ground-truth joint to the point cloud. Then, we count the number of pixels assigned to each joint. If the number of pixels for joint i is less than 20, we consider that joint as occluded. On average, 11% of the joints are occluded in the test set. Visible and occluded joints' error of our final methodology (i.e., *8:7 + BN*) are 9.2 and 16.1 mm, respectively.

Robustness against overfitting. We also analyse the effect of constraints in the training process in Figure 6d. As can be seen, by applying the proposed constraints, method *3:2 + constraint* is more robust against overfitting than method 1. The validation error in method 3 does not significantly change from epoch 7 to 15. On comparing both methods in epoch 20, it can be observed that method 1 has a lower error in training while its validation error is almost 1.5 times the validation error of method 3.

4.2.2 | Ablation study on Hands17 dataset

We study the following models on this dataset: *4:3 + viewpoint + fusion + BN, 4v:4-viewpoint + BN, 4a:4-app loss + BN, 4d:4-dyn loss+BN* and *8:7 + BN*. Note that we include BN in all the models on this dataset.

We compare the maximum error success rate in Figure 7a. As expected, *3:2 + const. + BN* method performs the worst. We also study the impact of Q branch, L_{app} and L_{dyn} on *4:3 + viewpoint + fusion + BN* by individually removing them from the training. The results show that L_{dyn} works better than L_{app} for error thresholds smaller than 25 mm (cyan vs. brown line). Perhaps one reason is because of the high rate of occluded joints due to egocentric captures in this dataset. Moreover, L_{dyn} has an impact similar to the Q branch (cyan vs. green dashed line). The combination of all these components in the training improves the results, as visible in *4:3 + viewpoint + fusion + BN* method (blue line). Finally, training the model with the proposed data augmentation improved the results by a large margin. To generate the augmented data, similar to the NYU dataset, we removed redundant samples. As a result, 343k samples remained in the training set. Then, we generated four additional augmented data per sample, resulting in a total of 1700k training data samples. We used random scaling and rotation in the range $[0.95 .. 1.05]$ and $[-90 .. 90]$, respectively. We expect more improvements by increasing the amount of augmented data.

We also compare the average error per joint in Figure 7b. The results are consistent among all the joints. Interestingly, the thumb has the smallest fingertip error among all the fingers in this dataset. The overall average error is 15.3 and 12.7 mm for *4:3 + viewpoint + fusion + BN* with and w/o data augmentation, respectively.
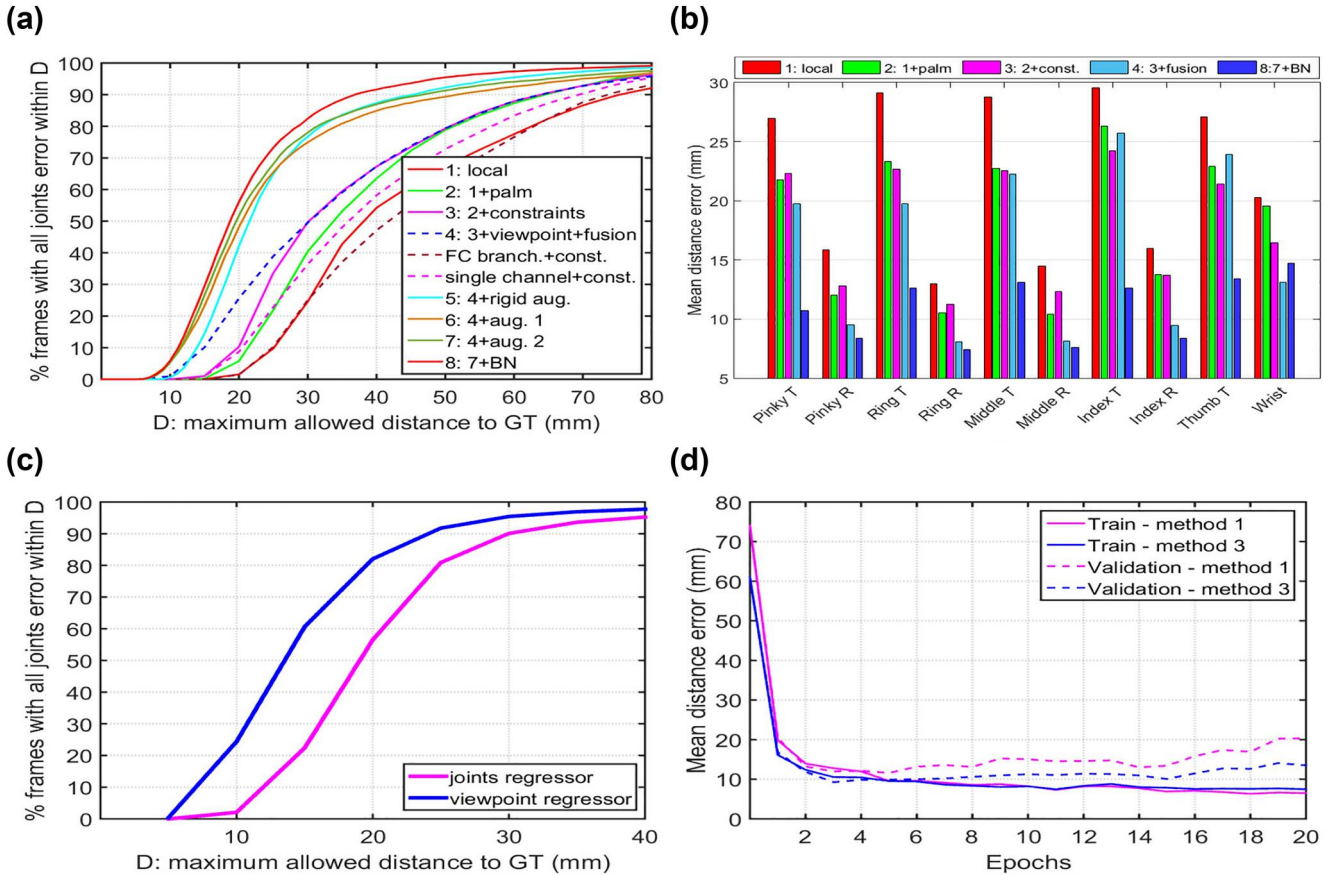
FIGURE 6 Ablation results comparing baselines on NYU dataset. (a) Maximum error success rate. (b) Per joint average error. (c) Comparing palm joints regression versus viewpoint regression. (d) Training process in terms of average error per epoch
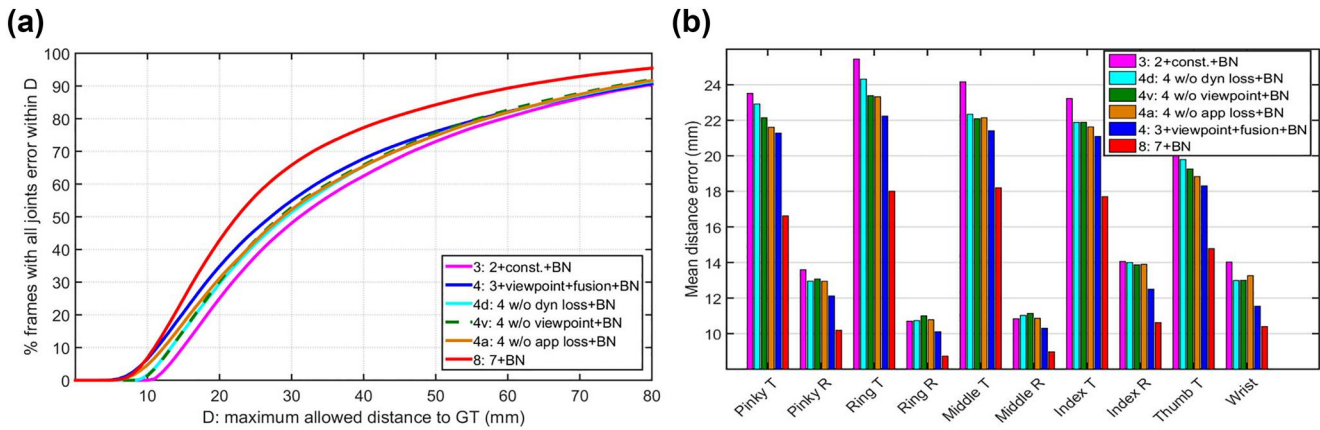

**FIGURE 7** Ablation results on Hands17 dataset. (a) Maximum error success rate. (b) Per joint average error

4.3 | Comparison with state-of-the-art approaches

We report the performance of our final model compared to state-of-the-art data-driven approaches like [5, 12–16, 25, 27, 31, 35, 36, 51] on the NYU dataset. On the MSRA dataset we compare to [3, 10, 13–16, 31, 36, 52]. Finally, we compare to [26, 46] on the SyntheticHand dataset.

NYU dataset. The works mentioned in the comparison use 14 joints (as proposed in [25]) to compare on the NYU dataset. For a fair comparison on this dataset we take 11 joints that are most similar to [25] out of our 20 joints. We show the maximum error success rate results in Figure 8a. As one can see, we outperform the state-of-the-art approaches' results. Although [15] performs slightly better than *8:7 + BN* for error thresholds lower than 19 mm, our approach performs the best
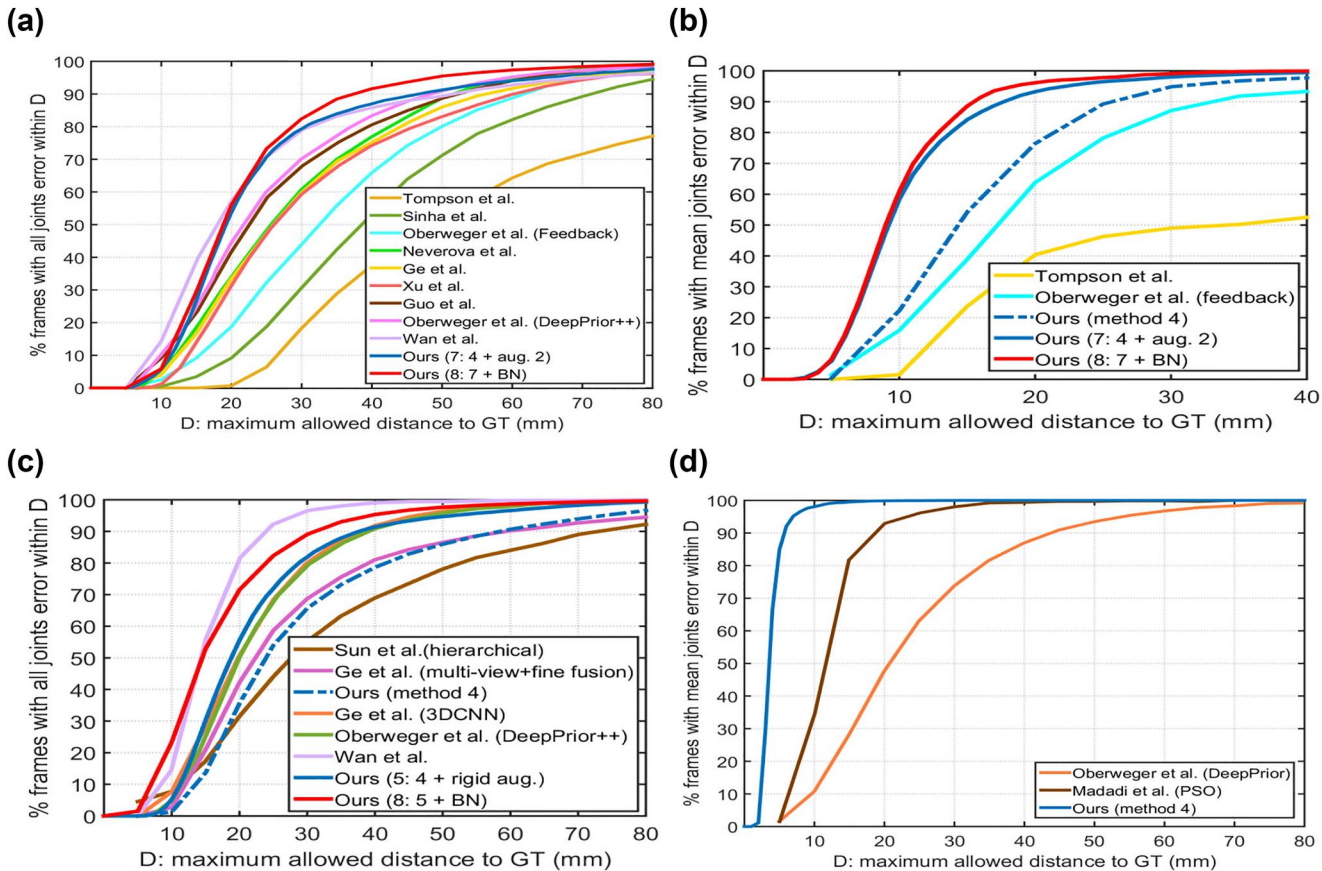
FIGURE 8 State-of-the-art comparison. (a) and (b) Maximum and mean error success rate on NYU dataset. (c) Maximum error success rate on MSRA dataset. (d) Mean error success rate on SyntheticHand dataset. We compare to [5, 25] on (a), [5, 12, 14, 15, 25, 27, 35, 36, 51] on (b), [3, 10, 14, 15, 36] on (c) and [26, 46] on (d)
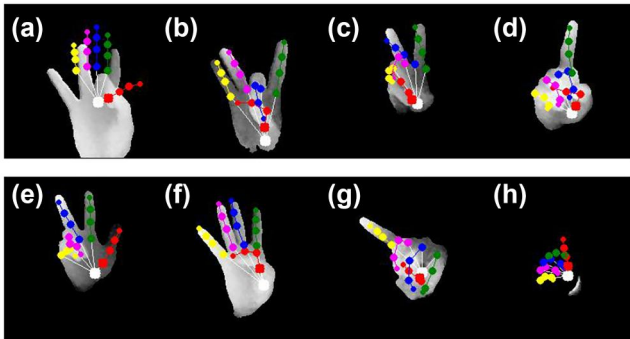

FIGURE 9 MSRA dataset examples with wrong annotations. We can observe that the following elements are wrong: (a) the whole ground truth, (b) thumb, (c) palm and thumb and thus the rest of fingers, (d) thumb, (e) pinky and ring fingers are confused, (f) palm and wrist, (g) thumb and index and (h) empty image

for higher error thresholds. We also illustrate the average error success rate in Figure 8b. This shows that our method performs well on average for a majority of frames, that is less than 10 mm error for 60% of the test set. We compare to state-of-the-art approaches regarding the overall mean error in Table 2. Although recent point-to-point heatmap-based approaches

perform better than our method on this dataset, our approach outperforms regression-based state-of-the-art approaches. All these results show a significant improvement using data augmentation and BN.

MSRA dataset. We applied introduced non-rigid hand augmentation same as the NYU dataset. However, we observed a divergence during training. A possible reason could be the accuracy of ground-truth annotations in the MSRA dataset, which caused wrong hand point cloud warping. We show some MSRA examples with wrong annotations or missing data in Figure 9. Therefore, we applied standard augmentation techniques such as random scaling (in range $[0.9, 1.05]$) and rotation (in range $[-90, 90]$ degrees). We show the maximum error success rate results in Figure 8c. As can be seen, with and without data augmentation, our methods *4:3 + viewpoint + fusion* (dashed blue line) and *5:4 + rigid aug* perform slightly worse than [3] and similar to [14], respectively. When we use BN, our method *8:5+BN* outperforms all the methods in the comparison except [15] for the error thresholds higher than 15 mm. We also illustrate the average error in Table 3, showing the best results (7.1 mm in average) among methods in comparison (including point-to-point heatmap-based approaches). We show some qualitative results in Figure 10b.
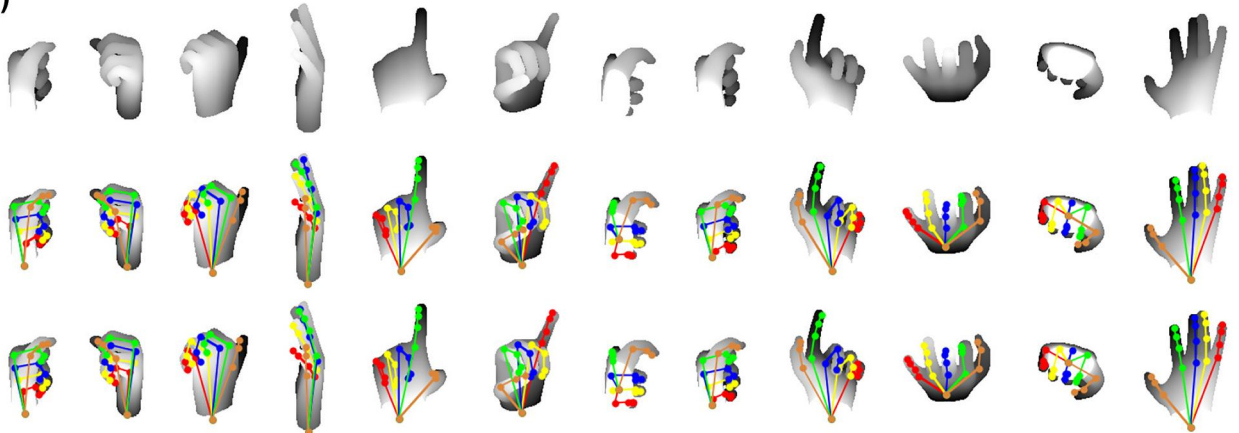
(a)
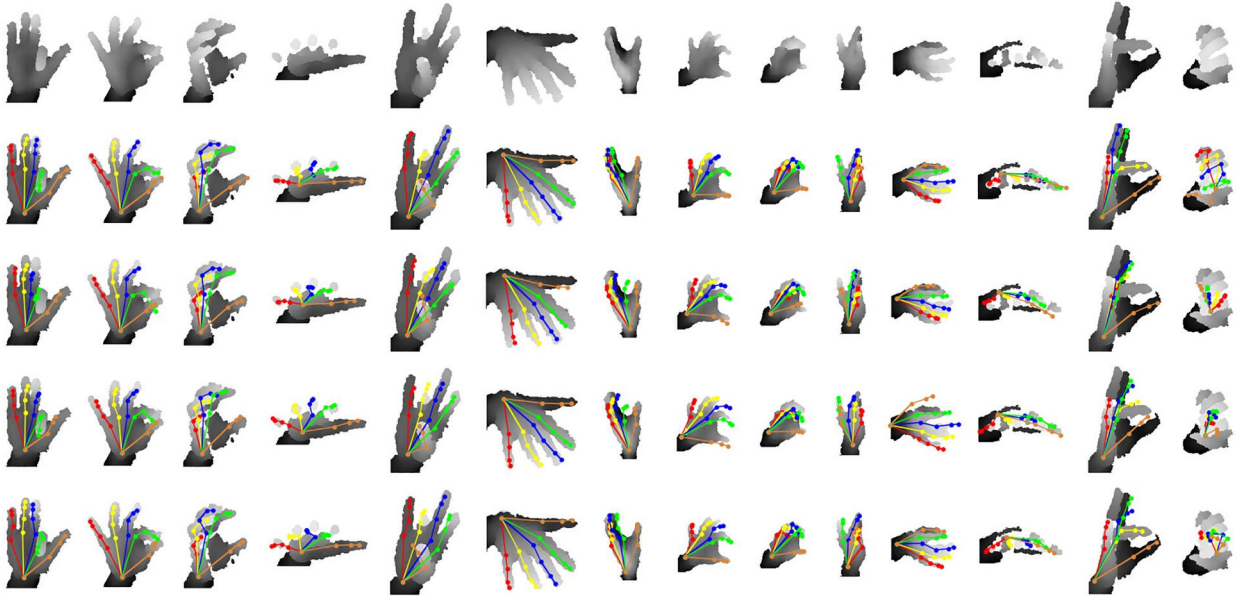
(b)
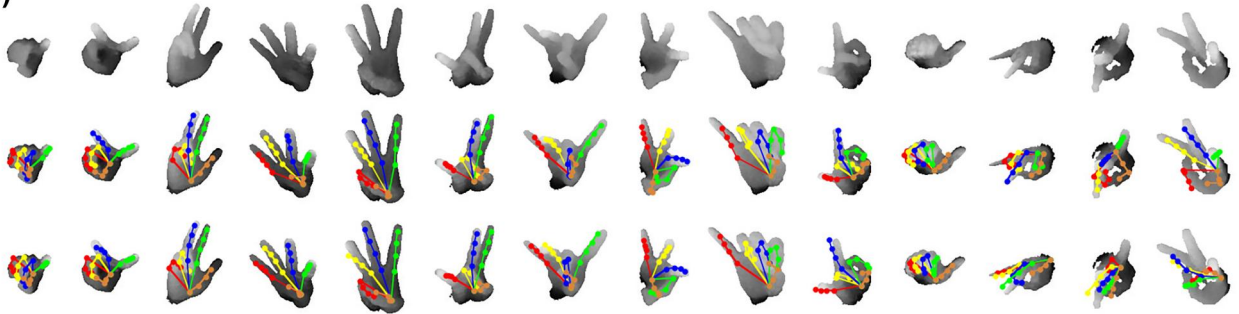
(c)

FIGURE 10 Qualitative results. (a) NYU dataset. Rows from top to bottom: depth image, ground truth, single channel network, methods *4:3 + viewpoint + fusion* and *7:4 + aug. 2*. Last three columns show maximum error higher than 50 mm. (b) MSRA dataset. Rows from top to bottom: depth image, ground truth and method *5:4 + rigid aug*. Last three columns show maximum error higher than 50 mm. Ground-truth annotations show inaccurate ground truth for this dataset, especially for thumb; 12th column has inaccurate ground truth for little finger. (c) SyntheticHand dataset. Rows from top to bottom: depth image, ground truth and method *4:3 + viewpoint + fusion*. See text for details of the methods

Hands17 dataset. We compare our approach to the state-of-the-art approaches on this dataset in Table 4. As it is common on this dataset, along with the overall average error, the results are reported on the seen subjects in the training set versus the unseen subjects. Our *8:7 + BN* approach generates good results on this dataset. However, it is not among the top

performing approaches. We believe that by training on the higher rates of augmented data (currently 4:1 vs. 36:1 in the NYU dataset), we can close the margin to the state-of-the-art

T A B L E 2 Average 3-D error on NYU dataset

Method	Average 3-D error (mm)
Oberweger et al. [26] (DeepPrior)	19.8
Oberweger et al. [5] (feedback)	16.2
Neverova et al. [35]	14.9
Guo et al. [12] (Ren)	13.4
Oberweger et al. [14] (DeepPrior++)	12.3
Ge et al. [13] (PointNet)	10.5
Wan et al. [15]	10.2
Moon et al. [16]	9.2[a]
Ge et al. [31] (point-to-point)	**9.1**
Ours (*4:3 + viewpoint + fusion*)	15.6
Ours (*5:4 + rigid Aug*)	12.5
Ours (*7:4 + Aug 2*)	11.0
Ours (*8:7 + BN*)	10.1

Note: Lowest error in bold.

Abbreviation: BN, batch normalisation.

[a]The result is shown without epoch ensembling for fair comparison.

T A B L E 3 Average 3-D error on MSRA dataset

Method	Average 3-D error (mm)
Sun et al. [10]	15.2
Wan et al. [52] (CrossingNet)	12.2
Oberweger and Lepetit [14] (DeepPrior++)	9.5
Ge et al. [13] (PointNet)	8.5
Wan et al. [15]	7.2
Ge et al. [31] (point-to-point)	7.7
Moon et al. [16]	7.6
Ours (*4:3 + viewpoint + fusion*)	12.9
Ours (*5:4 + rigid Aug*)	9.7
Ours (*8:5 + BN*)	**7.1**

Note: Lowest error in bold.

Abbreviation: BN, batch normalisation.

T A B L E 4 Average 3-D error on Hands17 dataset

Method	Average 3-D error (mm)	Seen	Unseen
Vanora [53]	11.9	9.5	13.9
Chen et al. [33]	11.7	9.1	13.8
Ge et al. [13] (PointNet)	11.3	8.9	13.3
Moon et al. [16]	**9.9**	**7.0**	**12.4**
Ours (*8:7 + BN*)	12.7	10.2	14.7

Note: Lowest error in bold.

Abbreviation: BN, batch normalisation.

approaches. Interestingly, the error rate between seen and unseen subjects is homogeneous among all the methods in comparison.

SyntheticHand dataset. We use the original training set without augmentation to train our model (*4:3 + viewpoint + fusion*) on this dataset. Our model converges in seven epochs. The mean error success rate is shown in Figure 8d. As can be seen, our method performs quite well on this dataset even for complex poses and viewpoints. Some qualitative results are shown in Figure 10c. The overall average error on this dataset is 3.94 mm.

4.4 | Time complexity

We compare test time complexity of our proposed tree-structure network with some state-of-the-art architectures [12, 14–16, 31] in Table 5. Our approach performs near 50 fps, which is applicable in real-time scenarios. We also compare with common benchmark architectures in the same setup (GPU and library) in Table 6. Although our full model has a slightly higher number of parameters (152M) and FLOPs (16.6B) than VGG-16, it performs 11% faster for a batch size of 32. It shows the parallelisation capability of the proposed tree-structure network.

T A B L E 5 Comparing test time complexity of the proposed architecture to the state-of-the-art architectures

Model	Im. size	Library	GPU	Time (ms)
Guo et al. [12]	96	Caffe	Titan X	0.31
Ge et al. [31]	6k[a]	PyTorch	GTX 1080	20.5
Oberweger and Lepetit [14]	128	Theano	GTX 980 Ti	33
Wan et al. [15]	128	Tensorflow	Titan X	36
Moon et al. [16]	88[b]	Torch7	Titan X	285
Ours (one channel)	192	Matconvnet	Titan X	4.5
Ours (full)	192	Matconvnet	Titan X	20.8

Note: The information in this table is directly taken from the corresponding articles.

Abbreviation: GPU, graphics processing unit.

[a]The input is a point cloud.

[b]The input is a 3-D voxelised tensor.

T A B L E 6 Comparing time complexity of proposed architecture to standard networks

Model	Im. size	# param.	# FLOPs	Time (ms)
AlexNet	227	61M	726M	11.1–15.2
VGG-16	224	138M	16B	16.2–330.3
ResNet-152	224	60M	11B	161.6–582.8
Ours (one channel)	192	25M	4.8B	4.5–79.4
Ours (full)	192	152M	16.6B	20.8–294.4

Note: We compare number of parameters, number of FLOPs and feed-forward time of 1 versus 32 batch size. All the models have been tested in the same setup.

5 | CONCLUSIONS

We proposed a novel hierarchical tree-like structured CNN for recovering hand poses in depth maps. In this structure, branches are trained to become specialised in predefined subsets of the hand joints. We fused a network based on learned local features to model higher order dependencies among joints. The network is trained end-to-end. By including a new loss function incorporating appearance and physical constraints about doable hand motions and deformations, we found that our network helps to increase the precision of the final hand pose estimations for quite challenging datasets. In particular, we found that a fusion network can help to better localise joints for easier hand configurations while it behaves similar to a local solution for more complex cases. We improved palm joints by applying a viewpoint regressor and by fusing its learned features into the global pose. Finally, we introduced a non-rigid hand augmentation technique to deform original hands in terms of the shape and pose, helping to generalise better to the test set. As a result, we significantly improved estimations on the original NYU dataset by 4.6 mm in average. As future work, we will consider the network architecture optimisation for local branches in which we learn which fingers are more correlated to share earlier layers. Moreover, we will apply more complex data augmentation techniques to cope with noise in the depth image. Realistic data can be combined with synthetic data as well. In this sense, we will work on filling gaps realistically when more complex pose deformations are applied in the augmentation. Another possibility is to deform the hand surface to generate new subject hands.

CONFLICT OF INTEREST
None.

PERMISSION TO REPRODUCE MATERIALS FROM OTHER SOURCES
None. The datasets we used have all public license.

DATA AVAILABILITY STATEMENT
The data that support the findings of this study are openly available in (1) NYU at https://jonathantompson.github.io/NYU_Hand_Pose_Dataset.htm; (2) MSRA at https://www.dropbox.com/s/c91xvevra867m6t/cvpr15_MSRAHandGestureDB.zip?dl=0; (3) HAND2017 at http://icvl.ee.ic.ac.uk/hands17/challenge/; (4) SYNTHETIC HAND at http://chalearnlap.cvc.uab.es/dataset/25/description/#

ORCID
Meysam Madadi https://orcid.org/0000-0002-7384-5712

REFERENCES

1. Choi, C., et al.: A collaborative filtering approach to real-time hand pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2336–2344. (2015)
2. Steven Supancic, J., et al.: Depth-based hand pose estimation: methods, data, and challenges. arXiv:150406378v1 (2015)
3. Ge, L., et al.: Robust 3d hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3593–3601. (2016)
4. Qian, C., et al.: Realtime and robust hand tracking from depth. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1106–1113. (2014)
5. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3316–3324. (2015)
6. Tan, D.J., et al.: Fits like a glove: rapid and reliable hand shape personalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5610–5619. (2016)
7. Sridhar, S., Oulasvirta, A., Theobalt, C.: Interactive markerless articulated hand motion tracking using RGB and depth data. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2456–2463. (2013)
8. Keskin, C., et al.: Real time hand pose estimation using depth sensors. In: IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1228–1234. (2011)
9. Tang, D., et al.: Latent regression forest: structured estimation of 3d articulated hand posture. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3786–3793. (2014)
10. Sun, X., et al.: Cascaded hand pose regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 824–832. (2015)
11. Perez-Sala, X., et al.: A survey on model based approaches for 2d and 3d visual human pose recovery. Sensors. 14(3), 4189–4210 (2014)
12. Guo, H., et al.: Region ensemble network: improving convolutional network for hand pose estimation. In: IEEE International Conference on Image Processing, pp. 4512–4516. (2017)
13. Ge, L., et al.: Hand pointnet: 3d hand pose estimation using point sets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8417–8426. (2018)
14. Oberweger, M., Lepetit, V.: Deepprior++: improving fast and accurate 3d hand pose estimation. In: Proceedings of the IEEE international conference on computer vision Workshops, pp. 585–594. (2017)
15. Wan, C., et al.: Dense 3d regression for hand pose estimation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5147–5156. (2018)
16. Moon, G., Chang, J.Y., Lee, K.M.: V2V-posenet: voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5079–5088. (2018)
17. Rautaray, S.S., Agrawal, A.: Interaction with virtual game through hand gesture recognition. In: 2011 International Conference on Multimedia, Signal Processing and Communication Technologies, Aligarh, 17–19 December 2011, pp. 244–247. (2011)
18. Usabiaga, J., et al.: Global hand pose estimation by multiple camera ellipse tracking. Mach Vis Appl. 21, 1–15 (2009)
19. Sharp, T., et al.: Accurate, robust, and flexible real-time hand tracking. In: Proceedings of the 33rd annual ACM Conference on Human Factors in Computing Systems, pp. 3633–3642. (2015)
20. Makris, A., Kyriazis, N., Argyros, A.: Hierarchical particle filtering for 3d hand tracking. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 8–17. (2015)
21. Oikonomidis, I., Kyriazis, N., Argyros, A.: Efficient model-based 3d tracking of hand articulations using kinect. In: Proceedings of the British Machine Vision Conference, pp. 101.1–101.11. (2011)
22. De La Gorce, M., Fleet, D.J., Paragios, N.: Model-based 3d hand pose estimation from monocular video. IEEE Trans. Pattern Anal. Mach. Intell. 33(9), 1793–1805 (2011)

23. Xu, C., Cheng, L.: Efficient hand pose estimation from a single depth image. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3456–3462. (2013)

24. Kirac, F., Kara, Y.E., Akarun, L.: Hierarchically constrained 3d hand pose estimation using regression forests from single frame depth data. Pattern Recogn. Lett. 50(0), 91–100 (2014)

25. Tompson, J., et al.: Real-time continuous pose recovery of human hands using convolutional networks. ACM Trans. Graph. 33(5), 1–10 (2014)

26. Oberweger, M., Wohlhart, P., Lepetit, V.: Hands deep in deep learning for hand pose estimation. In: Computer Vision Winter Workshop, pp. 1–10. (2015)

27. Sinha, A., Choi, C., Ramani, K.: Deephand: robust hand pose estimation by completing a matrix imputed with deep features. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4150–4158. (2016)

28. Belagiannis, V., et al.: Robust optimization for deep regression. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2830–2838. (2015)

29. Akhter, I., Black, M.J.: Pose-conditioned joint angle limits for 3d human pose reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1446–1455. (2015)

30. Erol, A., et al.: Vision-based hand pose estimation: a review. Comput. Vis. Image Understand. 108(1–2), 52–73 (2007)

31. Ge, L., Ren, Z., Yuan, J.: Point-to-point regression pointnet for 3d hand pose estimation. Proceedings of the European Conference on Computer Vision (ECCV), pp. 475–491. (2018)

32. Tompson, J., et al.: Efficient object localization using convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 648–656. (2015)

33. Chen, X., et al.: Pose guided structured region ensemble network for cascaded hand pose estimation. arXiv preprint arXiv:170803416 (2017)

34. Zhang, Z.: On the epipolar geometry between two images with lens distortion. In: Proceedings of the International Conference on Pattern Recognition, pp. 407–411. (1996)

35. Neverova, N., et al.: Hand pose estimation through semi-supervised and weakly-supervised learning. CVIU (2017)

36. Ge, L., et al.: 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1991–2000. (2017)

37. Felzenszwalb, P.F., et al.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. 32(9), 1627–1645 (2009)

38. Chu, X., et al.: Structured feature learning for pose estimation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4715–4723. (2016)

39. Zhou, X., et al.: Towards 3d human pose estimation in the wild: a weakly-supervised approach. Proceedings of the IEEE International Conference on Computer Vision, pp. 398–407. (2017)

40. Wei, S.E., et al.: Convolutional pose machines. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4724–4732. (2016)

41. Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1538–1546. (2015)

42. Fan, X., et al.: Combining local appearance and holistic view: dual-source deep neural networks for human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1347–1355. (2015)

43. Tompson, J.J., et al.: Joint training of a convolutional network and a graphical model for human pose estimation. NIPS (2014)

44. Li, S., Zhang, W., Chan, A.B.: Maximum-margin structured learning with deep networks for 3d human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2848–2856. (2015)

45. Yuan, S., et al.: The 2017 hands in the million challenge on 3d hand pose estimation. arXiv preprint arXiv:170702237 (2017)

46. Madadi, M., et al.: Occlusion aware hand pose recovery from sequences of depth images. In: IEEE International Conference on Automatic Face and Gesture Recognition, pp. 230–237. (2017)

47. Taylor, J., et al.: The vitruvian manifold: inferring dense correspondences for one-shot human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 103–110. (2012)

48. Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for matlab. In: Proceeding of the ACM Int Conf on Multimedia, pp. 689–692. (2015)

49. Bookstein, F.L.: Principal warps: thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell. 11(6), 567–585 (1989)

50. Madadi, M., et al.: Top-down model fitting for hand pose recovery in sequences of depth images. Image Vis. Comput. J. 79, 63–75 (2018)

51. Xu, C., et al.: Lie-x: depth image based articulated object pose estimation, tracking, and action recognition on lie groups. Int. J. Comput. Vis. 123(3), 454–478 (2017)

52. Wan, C., et al.: Crossing nets: dual generative models with a shared latent space for hand pose estimation. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, vol. 7. (2017)

53. Yuan, S., et al.: Depth-based 3d hand pose estimation: from current achievements to future goals. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2636–2645. (2018)