Pattern Analysis and Applications
https://doi.org/10.1007/s10044-019-00801-7

THEORETICAL ADVANCES

Single-shot 3D hand pose estimation using radial basis function networks trained on synthetic data

Vassilis C. Nicodemou[1,2] · **Iason Oikonomidis**[2] · **Antonis Argyros**[1,2]

Received: 16 July 2018 / Accepted: 14 February 2019

Abstract
In this work, we present a novel framework to perform single-shot hand pose estimation using depth data as input. The method follows a coarse to fine strategy and employs several radial basis function networks (RBFNs) that are trained on a dataset containing only synthetically generated depth maps. Thus, compared to most contemporary deep learning approaches, it does not require the laborious annotation of large, real-world datasets. At run time, an initialization RBFN is used to provide a rough estimation of the hand's 3D pose. Subsequently, several specialized RBFNs are employed to improve that initial estimation in an iterative refinement scheme. To train the RBFNs, we select a set of hand poses from a real-world sequence that are as diverse as possible. We use this representative set, along with a dense sampling of all possible rotations, as a seed to generate a large synthetic training set. The method is parallelizable, taking advantage of the inherent data parallelism of RBFNs. Furthermore, the method requires few real-world data and virtually no manual annotation. We perform a quantitative evaluation of our method on a testing sequence of our own. We also present quantitative and qualitative results on a public dataset that is commonly used to evaluate hand pose estimation and tracking methods. We show that in all cases, our approach achieves promising results. Moreover, it can achieve comparable or even faster computational performance than current deep learning approaches but on a single CPU core, i.e., without requiring GPU processing.

Keywords 3D hand pose estimation · Radial basis functions · Neural networks · Synthetic dataset · Hand pose regression · Iterative refinement · Depth map

1 Introduction

The task of estimating the full pose of a human hand observed using visual input comprises a very interesting problem in the field of computer vision [9]. Theoretically, the problem is interesting because it is an instance of the more general problem of estimating the pose of arbitrary articulated objects. Effective methods to solve the problem can be used as building blocks enabling virtual and augmented reality scenarios. Applications of such approaches include robotic teleoperation, game control, and medical rehabilitation. The problem in its full generality remains unsolved because of a number of interacting and complicating factors, such as the uniform hand appearance, the dexterity and speed of the hand as well as the potential interaction with the environment.

In this work, we present a method for single-shot 3D pose estimation of an isolated (i.e., not interacting with the environment) hand observed with a depth sensor. We deal only with the problem of pose estimation, assuming a bounding box provided by a hand detector [15, 20]. We propose the use of regressor radial basis function networks (RBFNs) for hand pose recovery. Recent approaches commonly rely on deep artificial neural networks [11, 22, 23]. Nevertheless, both the training and testing computational performance of RBFNs is favorable compared to most such approaches. Specifically, most successful deep learning approaches for 3D hand pose estimation (a) require training on large,

✉ Vassilis C. Nicodemou
nikodim@ics.forth.gr

Iason Oikonomidis
oikonom@ics.forth.gr

Antonis Argyros
argyros@ics.forth.gr

[1] Computer Science Department, University of Crete, Heraklion, Greece

[2] Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH), 100 N. Plastira, 700 13 Heraklion, Crete, Greece

Published online: 27 February 2019

human-annotated training sets and (b) achieve real-time performance using GPU acceleration. Our approach is trained on a completely synthetic dataset. Moreover, although it does not achieve as accurate results as the top performing deep learning approaches, it can achieve comparable or even faster performance using a *single CPU core*.

The proposed approach is split into an offline training phase and an online estimation phase. We start by capturing a long sequence of widely varying hand poses using an RGBD sensor. The sequence is processed offline similarly to [36]. In contrast to that work however, we discard the real-world depth maps and only keep the hand poses. These poses seed the synthetic generation of a large hand pose database. Firstly, the most distinctive hand poses are selected, essentially discarding very similar ones. A large number of rotations are uniformly sampled, and for each combination of rotation and hand pose, a depth map is generated. This constitutes a database of synthetic depth maps with known hand pose and global hand rotation. Using this dataset, we train a large number of RBF networks, each specialized in recovering the hand articulation given the global hand rotation or predicting global hand rotation given an articulation. Furthermore, an RBFN is trained in a subset of all the hand pose combinations so that it can provide a rough estimate of an observed depth map. During run time, this rough estimation is used to initialize an iterative search that employs the specialized RBFNs to refine its initial estimation. After we conclude with the training of our networks, we proceed to the estimation phase. Given a new observation of a hand in the form of a cropped depth map, we derive an initial estimation of the hand pose using the initialization RBFN. Then, with the use of an iterative search scheme employing the specialized RBFNs we improve the initial estimation. After a predetermined number of epochs, we stop the refinement, yielding the final estimation of the hand pose.

2 Related work

The problem of visual hand pose estimation is a long-standing one in the field of computer vision. Works as early as 1994 have addressed it [29]. The computational load of the method proposed by Rehg and Kanade [29] was too large for commodity processors back then, and therefore, specialized hardware was required. The steady, exponential increase in available computational power as well as the success of depth sensors have helped renew the interest in the problem [24, 27]. More recently, the success of deep learning in computer vision has also given new interest in this problem [11, 22].

An overview on the subject can be found in the review work by Erol et al. [9]. In that work, the authors categorize the methods on hand tracking according to the level of detail of the estimated pose. This ranges from simple 2D localization of some hand parts on the observed image (termed partial) to full estimation in 3D of all the rigid parts that comprise the hand (called full degrees of freedom—DoF). Another categorization discriminates between methods that can perform single-shot pose estimation (called single frame) and those that perform tracking (termed model-based tracking methods). In an evolution of these terms, single frame corresponds to discriminative methods and model-based tracking to generative ones, as used, for example, in [22].

In general, discriminative methods require a time-consuming training phase, learning a direct mapping from observations to the hand pose space. At run time, the computational requirements of these methods are usually low. On the downside, accuracy is determined during the training phase. In principle, it is not possible to improve this accuracy at test time without resorting to a slower generative methodology. On the other hand, given a candidate pose, generative methods are in a position to synthesize features that are directly comparable to the observations. By quantifying this comparison, the task is reduced to an optimization problem. The parameters of this optimization determine the desired hand pose. Because of the online feature generation, this category is usually more computationally intensive than discriminative approaches. On the other hand, the accuracy of a generative method can usually be improved by using more computational power.

2.1 Generative methods

Generative methods employ a kinematics and appearance model of the hand during the estimation process to synthesize image features. These features are compared to respective ones extracted from the observed image. Candidate feature types may be edges, depth information, skin color, or even the full-hand appearance. A quantification of the feature comparison for varying poses of the hand model serves as an objective function to an optimization procedure. Thus, the original problem is effectively reduced to an optimization one with search space the parameterization of the hand model. Due to the high dimensionality of the configuration space though, the computational performance of these methods is limited. A big disadvantage of such methods is the requirement of initialization. Most such methods begin the search for the optimum solution from previous estimations, effectively making the assumption of temporal continuity between consecutive frames in an image sequence. On the positive side, these methods can be easily adapted to different situations such as varying lighting conditions or object manipulation. The research areas of model-based methods include the construction of efficient and realistic 3D hand models, the dimensionality reduction in the configuration

space, and the development of fast and reliable tracking algorithms to estimate the hand posture.

The input to generative methods can be depth, multiview input, or even monocular RGB data. De la Gorce et al. [7] propose the use of temporal texture continuity and shading information, having monocular video as input. Athitsos and Sclaroff [2] achieve a 3D estimation from cluttered images. Other works try to retrieve depth information from RGB images through stereo matching [26, 40], where the acquired depth information may be treated like in depth images' scenarios. The first generative approaches appeared during the 1990s [13, 28]. In these works, the authors employed a simplified model of the hand in order to achieve tracking, estimating multiple degrees of freedom (DoF). Some methods [4, 17] use variants of particle filtering to track predetermined hand models, or shape invariant hand models [18] from depth information.

In order to optimize the objective functions formulated by these methods, particle swarm optimization (PSO) [1, 8] has been proposed [24, 25] as a general-purpose black-box optimizer that can effectively tackle the resulting problems. Oikonomidis et al. [24] used PSO to minimize the discrepancy between the 3D structure of hypothesized instances of a hand model and actual hand observations. Qian et al. [27] proposed a generative method combining the PSO and iterative closest point (ICP) algorithms to speed up the search of the hand pose space. More works use ICP [10, 33], employing the articulated variant of this technique (articulated ICP).

2.2 Discriminative methods

Discriminative methods estimate hand configurations directly from the input images using a precomputed mapping from the image feature space to the hand configuration space. Discriminative methods attempt to solve a difficult problem since the mapping from images to hand poses is highly nonlinear due to the variation of hand appearances under different views. In order to perform the mapping, a dataset is required that associates the input to the method, usually an image, with the target values that correspond to that input, that is, the respective hand configuration. This dataset is used to train a method and create the desired mapping. During estimation, the input can be mapped to the closest corresponding target that the method has learned, or interpolate to a new target value creating a regressed estimation of the input. For increased accuracy of these methods, a large training dataset is useful to cover the large hand space. Consequently, the training phase of such methods is usually time-consuming. On the positive side, discriminative methods are in general fast at run time, since the training phase is performed offline. They require only a single camera and have no need of initialization in order to perform an estimation. A particular property of discriminative methods is that they can be easily specialized to specific hand configurations. The research areas of discriminative methods include the selection of appropriate training algorithms, the use of suitable learning techniques, the creation and annotation of large datasets, and the development of training models that can generalize to unseen data.

For discriminative methods, many learning procedures and algorithms have been proposed and used, from *k*-nearest neighbor searches to deep convolutional neural networks. Wang and Popovic [39] use a glove to track a 3D hand and employ nearest neighbor approach to achieve tracking at interactive rates. Different variants of random decision forests are also used in several works [16, 34, 37], for regressing to a 3D hand estimation. A learning method based on relevance vector machines (RVM) [35] has also been proposed to estimate the hand pose from multiple cameras [6].

Ge et al. [11] present a discriminative method based on convolutional neural networks (CNNs). They estimate joint locations using three different projections of an observed point cloud. The projection viewpoints are determined using PCA on the segmented point cloud. The resulting heat maps are fused in a single-pose estimation by approximating them as Gaussians and using a prior of hand configuration constraints. The discriminative method by Oberweger et al. [23] employs three different CNNs as three parts of a hand pose estimation methodology. The input to the first net is a depth map and the output an estimated hand pose. The second net is trained to synthesize a depth map given a hand pose. The third map can compare the observed and the synthetic depth maps, proposing an update for the estimation of the hand pose. The method forms a loop out of these three nets, with the output of the comparison net provided to the synthesizer net and back to improve the initial estimation. Qian et al. [27] propose a generative method combining the particle swarm optimization and iterative closest point algorithms to speed up the search of the hand pose space.

2.3 Relation to the proposed approach

Our work is closely related to works by Romero et al. [30] and Tompson et al. [36]. Romero et al. [30] present a system to recover the hand pose from monocular RGB input using histogram of oriented gradients features. Similarly to our approach, they propose the search over a large synthetic database for the entry (or entries) with the closest features to the observed ones. On the other hand, the use of RGB data limits the discriminative ability of their feature space, making it imperative to use temporal coherency as a strong prior in the search. In contrast to this, our approach relies on depth data and therefore it can perform single-shot estimation. Tompson et al. [36] propose the use of an early generative RGBD-based approach [24] to annotate a large set of input hand poses instead of manual annotation. This dataset

is used to train a deep convolutional neural network that learns to estimate specific landmarks of the human hand. An inverse kinematics procedure produces the final hand pose based on this estimation of landmarks. Similarly, in our work we automate the task of annotating input data. In contrast to [36], we use the output of this annotation to generate a much larger synthetic dataset which we then proceed to learn. Learning from synthetic data is problematic in deep neural networks since they rely in the statistics of the input images across all scales. On the other hand, the regression method we adopt, RBF networks, is able to abstract away the small details of the input data, thus generalizing well from synthetic training data to real-world input.

Our work is also closely related to works on dataset generation and augmentation. A deep convolutional architecture is presented by Oberweger et al. [22] that can generate predictions of joint locations in the form of 2D heat maps. The authors also propose the use of a bottleneck in the deep network architecture, to enforce a strong prior on natural hand poses. The predicted joint positions are refined by another network to improve estimation accuracy. In a method proposed by Bellon et al. [3], a glove with motion sensors is used to capture hand poses and augment an existing dataset with these poses.

3 Method

Our goal is to recover the global position, orientation, and full articulation of a human hand observed by a depth sensor. Toward this end, we parameterize the pose space of the human hand as a 27-dimensional vector (see below for details). The task is therefore reduced to one of parameter estimation, where the parameters of interest are the full-hand pose vector $\mathbf{p}$. We propose to train radial basis function networks (RBFN) [5] to regress directly from an input depth map to this pose vector. This type of artificial neural network uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis function activations of the input using the learned neuron parameters. In our case, an RBFN accepts as input a depth map containing a hand and outputs the hand pose vector. We follow a training process that consists of two steps. The first step regards choosing the hyper-parameters of the network, namely σ, the standard deviation of the RBFs and the RBF centers $\mathbf{C}$. This evaluation is performed on a small dataset. The second step is the main one, which is responsible for learning the network weights $\boldsymbol{\theta}$ on the full training dataset.

In our work, we use a 27-parameter hand model for representation, similar to [24]. A hand pose $\mathbf{p}$ is a vector defined as $\mathbf{p} = (x, y, z, q_x, q_y, q_z, q_w, \phi_1, \phi_2, \ldots, \phi_{20})$. The first three values x, y, z define the global 3D position of the hand model, as an anchor point on it, specifically a predetermined point on the palm. The next four values q_x, q_y, q_z, q_w define a quaternion that determines the global rotation of the hand model in the 3D space around the anchor point of the model. Each of the remaining 20 parameters $\phi_1, \ldots, \phi_{20}$ describes an angle of a joint of the hand, defining the full articulation of the hand.

The method is divided into two phases, the training phase and the estimation phase. The training phase consists of preparing the training set and training an initialization RBFN and specialized RBFNs. The estimation phase uses the parameters learned during the training phase in order to estimate the hand pose given a single depth frame of a test set. Figure 1 illustrates the pipeline during the training phase and also demonstrates an example of the estimation phase.

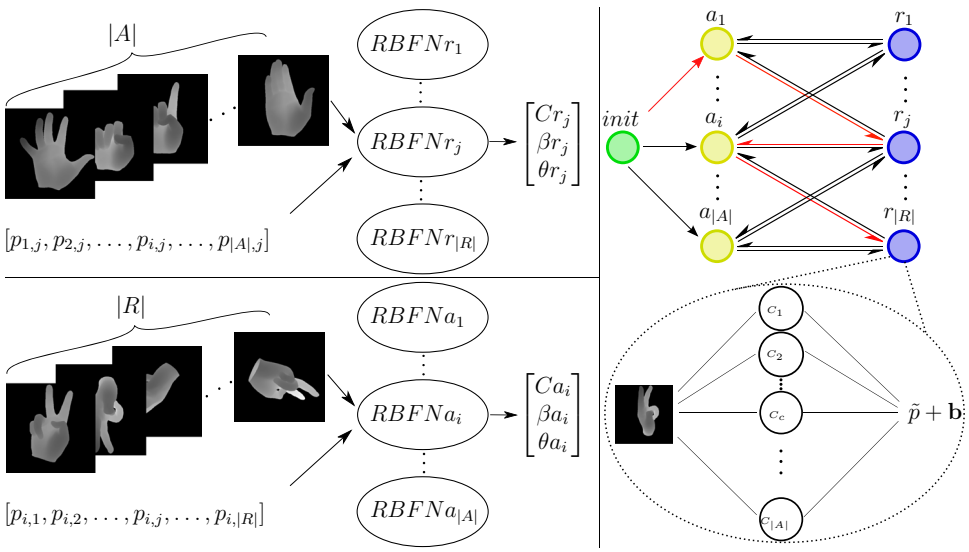

Fig. 1 Top left: Sets containing $|A|$ training samples are used to train $|R|$ different rotation-specialized RBFNs. This is achieved by fixing the rotation r_j each time and learning the pairs $(\text{Ren}(\mathbf{p}_{i,j}), \mathbf{p}_{i,j})$ for $i = 1, \ldots, |A|$. Bottom left: Symmetrically to top left, exchanging the roles of rotations and articulations. Top right: Sample execution of the IRA for 2 epochs, control flow shown using red arrows. See text for details. Bottom right: Illustration of an RBFN with $|A|$ number of centers, and the weighted sum of its hidden neurons composing the estimation $\tilde{\mathbf{p}}$ to which we add the bounding box center $\mathbf{b}$

3.1 Training preparation

In order to prepare our training dataset, we synthesize a large database of hand poses $\mathbf{p}$ along with their respective depth maps Ren($\mathbf{p}$). To do so, we use two sets $\mathcal{A}$ and $\mathcal{R}$ which contain articulations and rotations, respectively. The set $\mathcal{A}$ is obtained by tracking a real-world sequence that captures diverse hand poses. Section 4.1 provides details on this sequence and the Ren functionality. To obtain $\mathcal{R}$, we sample densely the quaternion space of rotations. The available rotation and hand articulation samples are assumed to be densely, or even redundantly covering the respective spaces of rotations and hand articulations. Therefore, out of the full rotation and articulation sets, we create a subset $A \subseteq \mathcal{A}$ and $R \subseteq \mathcal{R}$ using the KKZ initialization technique [14].

The KKZ algorithm [14] is an initialization technique for algorithms that employ the generalized Lloyd iteration. The KKZ initialization exploits the fact that the most distant input vectors are more likely to belong to different classes. KKZ orders the input so that the first two elements are the most distant ones, the third element is the most far apart from the previous two, and so forth. Specifically, in each iteration, the next selected vector is the one that maximizes the minimum distance from the already selected vectors.

By applying the KKZ algorithm, we sort the articulations in $\mathcal{A}$ and rotations in $\mathcal{R}$ and select the first $|A|$ articulations and the first $|R|$ rotations to use for training, where $|A| \leq |\mathcal{A}|$ and $|R| \leq |\mathcal{R}|$ are preselected sizes. Thus, we achieve a dense, evenly distributed training set that contains representatives of all the articulations in the captured sequence and of all rotations in the 3D rotation group $SO(3)$.

A significant design choice that must be made before the use of the KKZ algorithm is the metric that is required in order to quantify the distance of articulation or quaternion pairs. For quaternions, we use the dot product as a similarity metric. This can be converted to a distance metric by subtracting it from the unit:

$$DQ(\mathbf{q_1}, \mathbf{q_2}) = 1 - (\mathbf{q_1} \cdot \mathbf{q_2}), \tag{1}$$

where $\mathbf{q_1}$ and $\mathbf{q_2}$ are the two compared quaternions. In order to quantify the articulation distance, we employ the distance function that is also used for quantitative evaluation of the method (see Sect. 4.3). The metric quantifies the distance between two hand poses, taking into account their 3D global position, articulation and rotation. For two given hand poses $\mathbf{p_1}$ and $\mathbf{p_2}$, the function that measures the distance between them is defined as

$$D(\mathbf{p_1}, \mathbf{p_2}) = \frac{1}{n} \sum_{l=1}^{n} ||L_{1_l} - L_{2_l}||_2 \tag{2}$$

where L_1 and L_2 are the landmarks of hand poses $\mathbf{p_1}$ and $\mathbf{p_2}$, respectively, and n is the number of landmarks on our model. To quantify the distance between articulation pairs, we fix the rotation of the two compared hand poses. It is easy to see that, since this distance is defined as the average of point distances, its unit of measurement is the same as these distances, in our case millimeters (mm).

Given that the desired output of the method is a full 27-dimensional hand pose containing the global 3D position p_x, p_y, p_z of the hand, the naive approach would be to use the RBFNs to directly regress from the depth map to the global position. Learning this relationship is not desirable though, because it would require a prohibitively large amount of training samples. Furthermore, the relationship to be learned is rather simple, since it roughly amounts to translating and scaling a reference depth map according to the camera parameters.

Based on this intuition, we could use the 3D position of the center of the bounding box $\mathbf{b}$ that encloses the hand model as a rough estimation of the global position. But computing the center $\mathbf{b}$ of the bounding box at the observed hand depth is not sufficient because the anchor point of the employed hand model will not necessarily coincide with $\mathbf{b}$. For this reason, we also compute and learn the 3D offset $\mathbf{o} = \mathbf{p} - \mathbf{b}$ of the model's position from the center $\mathbf{b}$. We store $\mathbf{o}$ instead of p_x, p_y and p_z for the purposes of training and train accordingly our RBFNs to regress for these parameters. Later, during evaluation, we use the regressed offset and add it to the bounding box position of the estimated hand depth map to recover the global hand position.

RBFNs compare the data with centers to make an estimation, given their radial distance. In our case, both the data and the centers of the network are depth maps; thus, the employed RBFNs must quantify the comparison between depth maps. Given that the role of these RBFNs is to quantify the discrepancy between hand poses, or global hand orientation, it is necessary to disentangle this comparison from the global hand position. To do so, depth normalization must be applied to the compared depth maps. Specifically, each training depth map is normalized by subtracting the median of the nonzero values and adding a fixed depth value.

All the pairs of depth images Ren($\mathbf{p}$) and the respective hand poses $\mathbf{p}$ which are $|A| * |R|$ in total form our training set. A subset of this training dataset is used to train the initialization RBFN, whereas the whole training set is used to train the rest of the networks. Each specialized RBFN uses a different subset of the training set that corresponds to its specialization.

3.2 Training the initialization RBFN

An RBFN must be trained so that it can be used as an initialization step of the iterative refinement algorithm (Sect. 3.4). We select the k first articulations and rotations from our sets A and R that will be used to train the initialization RBFN,

due to their diversity from the KKZ ordering. We create the k^2 poses $\mathbf{p}$ by combining the articulations and orientations of this training set. Having the pairs of poses with their corresponding depth maps Ren(p) as input, we train the network and store its parameters $C_{\text{init}}, \beta_{\text{init}}, \theta_{\text{init}}$.

The selected articulations and rotations are limited to the number k which is smaller than the $|A|$ and $|R|$. Apart from practical limitations such as memory usage, it is undesirable to train a single RBFN on all the available training data because the resulting network would be very slow during evaluation. Instead, we choose the solution of sparsely sampling the full pose set to train the initialization RBFN. The specialized RBFNs take over the task of refining its initial estimation.

3.3 Training specialized RBFNs

Having a single RBFN trained on every possible articulation and orientation is not efficient; thus, the specialized networks take over the task of refining an initial estimation by learning all rotations for a fixed articulation and vice versa. Each articulation $a \in A$ and orientation $r \in R$ is used to train, respectively, $|A|$ and $|R|$ specialized RBFNs, so that each RBFN is specialized in a specific articulation or in a specific orientation.

3.3.1 Articulation-specialized RBFNs

Each RBFNa_i is trained separately, for $i = 1, \ldots, |A|$. The training input for RBFNa_i is a matrix containing hand poses that have a fixed articulation $a_i \in A$ for all rotations R and their corresponding depth maps. Specifically, for the RBFNa_i the training input consists of input and target pairs

$$\begin{bmatrix} \text{Ren}(\mathbf{p}_{i,1}) \\ \vdots \\ \text{Ren}(\mathbf{p}_{i,j}) \\ \vdots \\ \text{Ren}(\mathbf{p}_{i,|R|}) \end{bmatrix}, \begin{bmatrix} \mathbf{p}_{i,1} \\ \vdots \\ \mathbf{p}_{i,j} \\ \vdots \\ \mathbf{p}_{i,|R|}) \end{bmatrix} \tag{3}$$

where $\mathbf{p}_{i,j}$ is a hand pose with articulation a_i and rotation r_j, $j = 1, \ldots, |R|$ and Ren($\mathbf{p}_{i,j}$) its respective depth map. The output parameters of the training that are stored for use during the estimation phase are the parameters of each of the networks $Ca_i, \beta a_i, \theta a_i$ for each articulation-specialized RBFN.

3.3.2 Rotation-specialized RBFNs

Similarly to the previous case, we fix the rotation r_j and train the networks RBFNr_j for every articulation in A. The training of every network RBFNr_j is composed of the pairs of training input and their corresponding targets

$$\begin{bmatrix} \text{Ren}(\mathbf{p}_{1,j}) \\ \vdots \\ \text{Ren}(\mathbf{p}_{i,j}) \\ \vdots \\ \text{Ren}(\mathbf{p}_{|A|,j}) \end{bmatrix}, \begin{bmatrix} \mathbf{p}_{1,j} \\ \vdots \\ \mathbf{p}_{i,j} \\ \vdots \\ \mathbf{p}_{|A|,j}) \end{bmatrix} \tag{4}$$

The learned parameters of the networks in this case are $Cr_j, \beta r_j, \theta r_j$.

3.4 Iterative refinement algorithm (IRA)

The focus of this work is the estimation of the hand pose having as input a crop of the hand within a depth map. We do not focus on the detection of the hand in the image, assuming that it is provided to the system. Therefore, we assume the position of the bounding box that contains the hand to be available during the estimation phase. To estimate the global position of a hand pose $\mathbf{p}$, we use the position of the bounding box that contains the depth image of the given hand pose. The estimation of a hand pose is derived from an RBFN that is chosen according to an iterative refinement scheme. This scheme uses an initial approximation of the hand pose. Then, it iteratively finds an articulation from the set A, and using the corresponding RBFN that was trained on that articulation, it returns a new estimation. This estimation is used to find a rotation from the set R. In the last step of the iteration, the corresponding rotation-specialized RBFN is used, computing a new estimation, that can now be provided to the start of the iteration.

More specifically, to evaluate a single test depth image t we start by finding a rough approximation $\tilde{\mathbf{p}} = \text{RBFN}_{\text{init}}(t; C_{\text{init}}, \beta_{\text{init}}, \theta_{\text{init}})$ which is computed using our initialization RBFN. Subsequently, for a number of iterations which we call epochs, we find the closest articulation a_i, from the set A that was used for training, to the articulation $\tilde{a}$ of the estimated pose $\tilde{\mathbf{p}}$. We input the test image t to the RBFNa_i that was trained upon this closest articulation to obtain a new estimation $\tilde{\mathbf{p}} = \text{RBFN}a_i(t; Ca_i, \beta a_i, \theta a_i)$. Then, we find the orientation r_j from the set R that is closest to the orientation $\tilde{r}$ of the newest estimation $\tilde{\mathbf{p}}$. An update of the estimation $\tilde{\mathbf{p}}$ is performed by the output of the rotation-specialized RBFN $\tilde{\mathbf{p}} = \text{RBFN}r_j(t; Cr_j, \beta r_j, \theta r_j)$. We repeat these steps as mentioned, starting from the beginning of the iteration. Iteratively we look up the closest articulation and orientation and update the estimation accordingly, for a predetermined number of epochs. The last RBFNr_j provides the final estimation $\tilde{\mathbf{p}}$.

For estimating the position of a hand pose, we need the vector $(\tilde{p}_x, \tilde{p}_y, \tilde{p}_z)$ that contains the estimated regressed offsets from the center of a bounding box. Given the vector $\mathbf{b}$ that holds the center of the test's image bounding box, we

add the two vectors in order to obtain the final global position of the hand pose.

Algorithm 1 outlines the iterative refinement algorithm. The function *FindClosest Articulation* returns the closest articulation to its first argument, from the set of articulations A by measuring the distances of the articulations to the input articulation. The distance is computed as defined in Eq. 2 by fixing the rotation of the two comparing hand poses. Similarly, the function *FindClosestRotation* returns the closest rotation to its first argument from the set R by measuring rotation distances to the provided rotation. The rotations are actually quaternions; thus, the distance between two rotations is equivalent to the distance between two quaternions. The distance between two quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$ is defined in Eq. 1.

Algorithm 1 Iterative Refinement Algorithm (IRA)

Input: The test depth image t and the center 3D vector of bounding box $\mathbf{b}$ containing the hand.
Output: The hand pose estimation $\tilde{p}$.

$\tilde{p} \leftarrow RBFN_{init}(t; C_{init}, \beta_{init}, \theta_{init})$;
for $e = 1 \ldots epochs$ **do**
  $a_i \leftarrow FindClosestArticulation(\tilde{a}, A)$;
  // FindClosestArticulation returns the closest
  // articulation $a_i \in A$ to the estimated one $\tilde{a}$
  $\tilde{p} \leftarrow RBFNa_i(t; Ca_i, \beta a_i, \theta a_i)$;
  $r_j \leftarrow FindClosestRotation(\tilde{r}, R)$;
  // FindClosestRotation returns the closest rotation
  // $r_j \in R$ to the estimated rotation $\tilde{r}$
  $\tilde{p} \leftarrow RBFNr_j(t; Cr_j, \beta r_j, \theta r_j)$;
end for
$\tilde{p}_{xyz} = \tilde{p}_{xyz} + \mathbf{b}$;
// We add to the estimated offset $\tilde{p}_{xyz}$ the center of the
// bounding box $\mathbf{b}$ to translate it to its global position
return $(\tilde{p})$;

Since at each iteration we search the space of our training data to find the closest articulations and rotations, the run time of the estimation is proportional to the size of the training dataset. Thus, for the selection of the number of epochs we must consider that with increased training set we increase the run time of the estimation phase.

For a visualization of the proposed pipeline, we refer again to Fig. 1. The example illustrates the run of IRA for 2 epochs and indicates with red arrows the closest articulations and rotations found in sets A and R, respectively. Specifically, at the beginning $\text{RBFN}_{\text{init}}$ estimates an approximation $\tilde{\mathbf{p}}$ that is used to initialize the iterative search. Using this estimation, in the first epoch the method finds the articulation a_1 that is the closest articulation in the set A to the approximated articulation $\tilde{a}$. $\text{RBFN}a_1$ estimates a pose $\tilde{\mathbf{p}}$. Symmetrically to the previous step, the method proceeds to find the rotation r_j that is the closest rotation among the ones in the set R to the estimated rotation $\tilde{r}$. Then, $\text{RBFN}r_j$ estimates a pose $\tilde{\mathbf{p}}$. In the second epoch, we repeat the previous actions, finding this time the articulation a_i and rotation $r_{|R|}$ as the closest ones, respectively. Therefore, $\text{RBFN}r_{|R|}$ is used to estimate the final hand pose $\tilde{\mathbf{p}}$.

4 Experimental evaluation

4.1 Developed datasets

Training datasets For the purpose of generating a number of synthetic training datasets, we recorded a sequence S_1 of a human hand performing a large variety of finger articulations. S_1 is about 2 minutes long and contains 3180 frames.

We employ the method presented in [24] to track S_1 with a large budget, yielding accurate results. The manual initialization required by this method is the only manual annotation required in the proposed pipeline. From the implementation of [24], we also use the module that synthesizes candidate depth maps to generate our synthetic dataset. We refer to the functionality of this module as Ren in Sect. 3.

After tracking S_1, we use the obtained results to create the set $\mathcal{A}$ that contains 3180 articulations. By sampling densely the quaternion space, we also create the set $\mathcal{R}$ using 1024 distinct rotations. We then employ the KKZ algorithm to sort $\mathcal{A}$ and $\mathcal{R}$ so as the most diverse poses and rotations, respectively, come first in this ordering. This allows for the generation of datasets of various lengths that best cover the articulation and rotation spaces for a given dataset size. Throughout the design and training of the proposed method, we design in this way four synthetic datasets:

- We combine the full $\mathcal{A}$ and $\mathcal{R}$ sets to create $3180 \times 1024 = 3,256,320$ hand poses. Using the function Ren, we also acquire the rendered depth maps of these poses, ending up with the complete synthetic training set T_{C}.
- We select the first 1024 articulations and 1024 rotations to use as the full training dataset T_{F}. Thus, T_{F} consists of $1024 \times 1024 = 1,048,576$ hand poses and their corresponding rendered depth maps.
- For training the initial RBFN ($RBFN_{\text{init}}$), we set the number of chosen articulations/rotations to $k = 100$ as described in Sect. 3.2 and end up with the dataset T_{I} that contains a total of $100 \times 100 = 10,000$ hand poses.
- Two even smaller subsets with the first 40 rotations and 40 hand articulations are used to develop the dataset T_{M} of $40 \times 40 = 1600$ hand poses. This dataset was used to train initialization and specialized RBFNs to fine tune the meta-parameters of the proposed method, as outlined in Sect. 4.2.

As already outlined, the complete dataset T_C is KKZ ordered, and all the other ones are derived exploiting this order. This implies that the four datasets above are each a subset of its immediately larger one, i.e., the following relationship holds: $T_M \subset T_I \subset T_F \subset T_C$.

Testing dataset We recorded a second sequence, S_2 showing similar hand articulations as S_1. S_2 contains 2710 frames. We tracked S_2 with the method of [24] and a large computational budget. This sequence referred later in the text as dataset E was used to evaluate the performance of the proposed method on real data with similar articulation ranges to the training set T_F (see Fig. 5).

4.2 RBFN training details

We use the implementation of radial basis function networks by McCormick [19]. RBFNs have two free parameters, the number of radial basis functions (RBFs) also known as *centers* and the shape of these functions, determined by a parameter σ. In our method, we add one more hyper-parameter, *epochs*, which is the number of iterations required to complete the refinement of the initial pose by the iterative refinement scheme IRA. In order to estimate the best values for these three meta-parameters, we first trained our model on the T_M dataset as outlined in Sect. 4.1. We also created a validation dataset that contained frames from the T_C dataset that were not included in the T_F dataset.

We found that the best validation set performance was achieved using the training samples as the centers of our RBFNs as shown in Fig. 2. Intuitively, this means that the best performance is achieved when memorizing the training dataset, interpolating between its samples only for unseen poses.

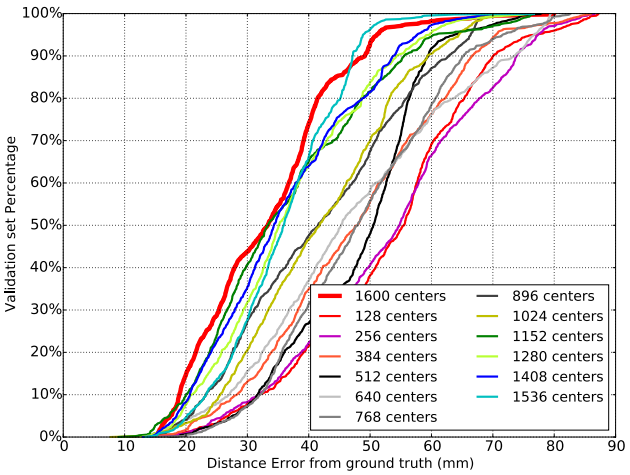

Fig. 2 Validating RBFNs for different number of centers plus a model trained on all training samples

We also varied the parameter σ over a large range of values, spanning several orders of magnitude. We found that the best validation set accuracy was achieved for $\sigma = 10,095$. Figure 3 presents the results of this experiment.

Finally, the number of iterations for IRA was set to 6, achieving the best trade-off between estimation time and accuracy. Even though the best performance is achieved for 8 epochs as presented in Fig. 4, we prefer 6 epochs as the estimation time increases for every additional epoch with only a slight increase in performance. This is clearly shown in Fig. 4 since the increase in run time is linear with the number of epochs (about 150ms per epoch in our implementation), while the performance improvement after 6 epochs is disproportionately smaller.

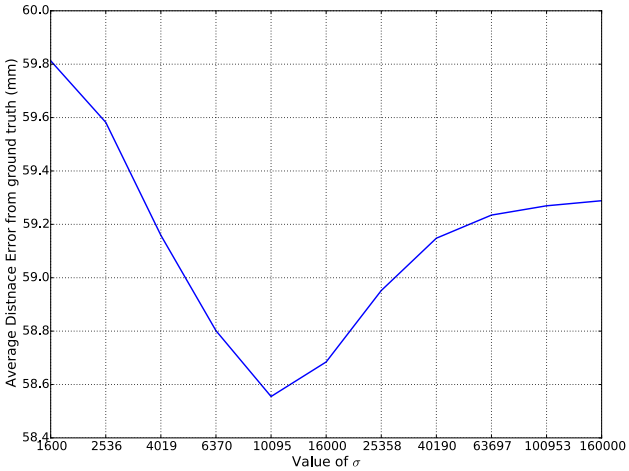

Fig. 3 Validating RBFNs for various σ values

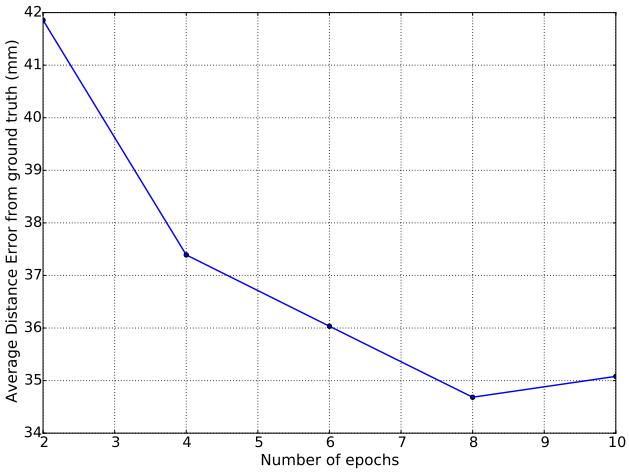

Fig. 4 Validating RBFNs for various numbers of epochs

4.3 Quantitative evaluation

In order to evaluate quantitatively the performance of our method, we reuse the error metric over hand poses in Eq. 2. We adopt the common approach of averaging the distance of multiple landmark points between two hand poses. In the results below, the distance is always in mm.

In a first experiment, we applied our method to the dataset *E*. Figure 5 illustrates the results of this experiment. The graph plots the percentage of poses that have an average error under a certain threshold. We compare our proposed method to that of using only the initialization RBFN. Evidently, IRA improves significantly the initial estimation.

We also evaluate our method on the publicly available MSRA hands dataset presented in [31]. MSRA consists of 17 different hand gestures performed by nine different subjects. These gestures are chosen from the American Sign Language, spanning as much as possible the finger articulation space. Each gesture is recorded for 500 frames, for a total of 76, 500 frames. The provided depth maps are annotated with 3D joint locations.

Table 1 presents average distance errors for four different methods that were tested on the MSRA dataset along with our proposed method. The four methods were trained on eight subjects of the MSRA dataset and tested on the held-out subject.

Table 1 Comparison of our whole method with other methods from the literature on the MSRA dataset

Method	Median distance error
Wan et al. [38]	25 ± 2 mm
Sun et al. [31]	28 ± 2 mm
Wan et al. [37]	32 ± 2 mm
Ge et al. [12]	20 ± 2 mm
Ours	48.93 mm

On the contrary, in our method, we tested the same RBFNs on all nine subjects, since our method is trained on the synthetic dataset T_F. This means that our approach has never seen any frame of the MSRA dataset, or any other real-world dataset for that matter. This is clearly disadvantageous to our method: As shown by the work by Supancic et al. [32], Table 5, cross-dataset generalization is far from being solved. It should be also stressed that there is no other way to compare these methods on the MSRA dataset, since our method can only be trained on a dataset that has a specific structure, as outlined in Sect. 4.1. Under these circumstances, we observe that our method has the largest error compared to the other methods. Even though the error we obtain is higher than the other approaches, it is satisfactory as seen by the qualitative results in Sect. 4.4, illustrating that our method is capable of generalization.

To fully understand the behavior of our method compared to a state-of-the-art deep learning method, we conduct another experiment. We used the implementation of *DeepPrior++* as presented in [21]. In order to achieve a fair comparison, we trained that network on, T_I, i.e., the synthetic dataset that was used to train $\text{RBFN}_{\text{init}}$. It should be noted that, due to memory and computational time limitations, it would not be possible to train *DeepPrior++* on T_F. Furthermore, training our method on the MSRA dataset (or any other real-world dataset for that matter) is not possible, given the specific dataset structure that is required by our method. For performance comparison, we evaluated the hand poses from the MSRA dataset for all the subjects. Figure 6 shows the percentage of hand poses having average error less than a threshold, for the cases: (a) our full method, (b) for $\text{RBFN}_{\text{init}}$ only, and (c) for *DeepPrior++*. Compared with the first experiment, we can observe that our method performs worse in MSRA than in *E*. This is attributed to the differences on the hand shapes and sizes as well as the varying depth ranges in the two datasets. Still, the performance is very appealing given the fact that the method achieves a generalization from synthetic to real data. The results also demonstrate that

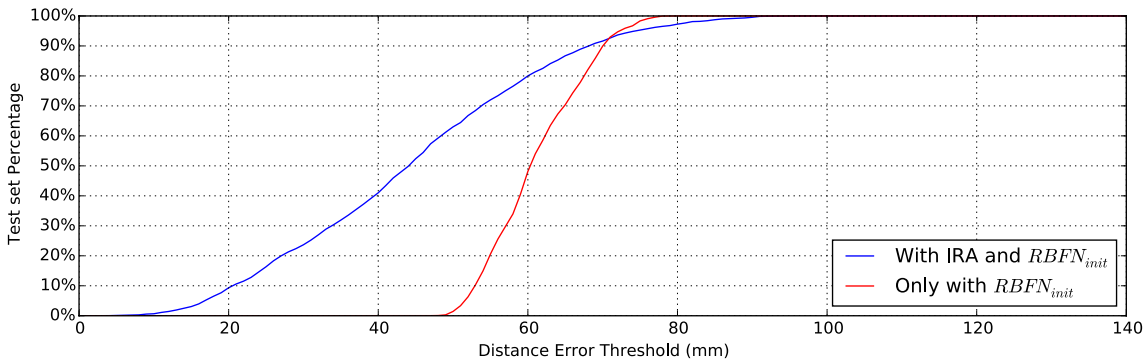

Fig. 5 Results on the test sequence *E*. Percentage of hand poses within an error threshold, plotted as a function of this threshold. The two different lines show the performance of the whole method compared to the initialization RBFN

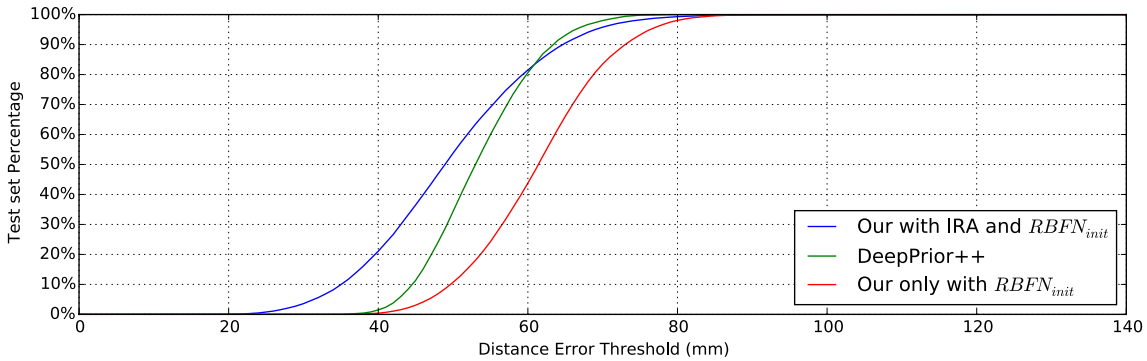

Fig. 6 Results on MSRA hands dataset. Percentage of hand poses within a threshold plotted as a function of this threshold. The three different curves show the performance of the whole method compared to the initialization RBFN and *DeepPrior++*. The latter was trained on the same training set as the initialization RBFN (i.e., on T_I)

although *DeepPrior++* can better generalize from synthetic to real data compared to the initial RBFN, the performance of $\text{RBFN}_{\text{init}}$ is still quite close. Even more importantly, the complete, proposed method ($\text{RBFN}_{\text{init}}$ + IRA) outperforms *DeepPrior++*.

4.4 Qualitative evaluation

Figure 7 shows representative frames of estimated hand poses on the captured test sequence with their respective manually annotated ground truths. In Fig. 8, we present images of estimated hand poses on the MSRA dataset. On left is shown the test depth image with the annotated ground truth and on right the estimated pose.

We observe that the estimations are quite accurate, and the method manages to generalize well on *E* and on MSRA, despite the increased average error. We may notice that for subjects with different hand shape variations (such as Subject 5), the estimations are slightly worse from other subjects, given the fact that our method is trained using a single hand shape.

In Fig. 9, we present some fail cases of hand pose estimations. We observe how our method tries to fit regressed hand poses to the input depth map by predicting wrong finger articulations that project to similar depth maps to

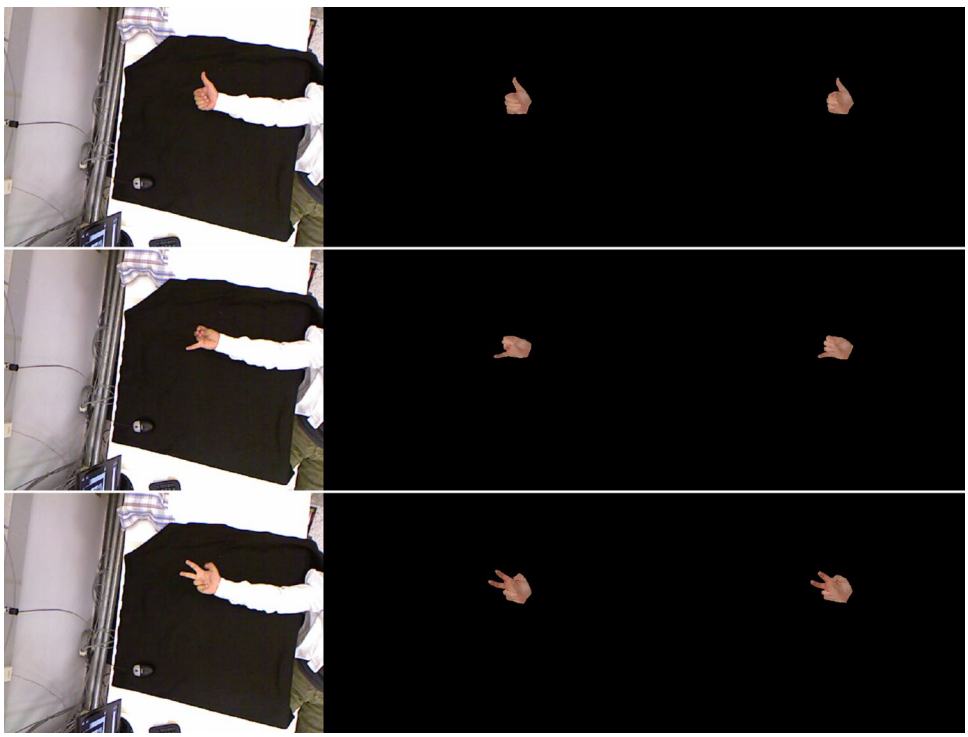

Fig. 7 Qualitative results on captured test sequence S_2. Left is the RGB test image, middle is the manually annotated ground truth, and right is the final estimation

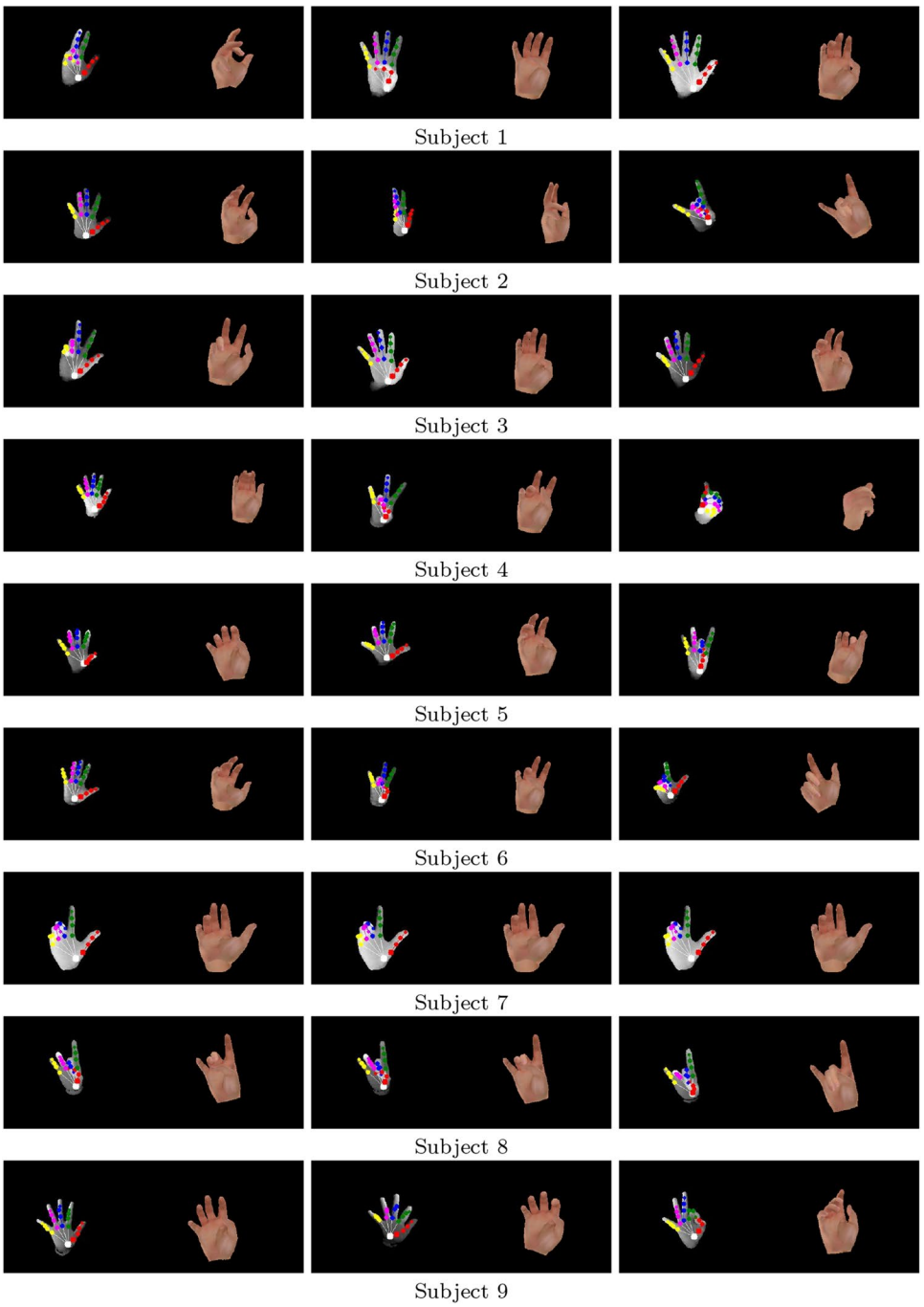

Fig. 8 Qualitative results on MSRA dataset: The left part of each image is the test depth image with the annotated ground truth. The right part of each image is the final estimation

the input. Other cases show how the method may fail on articulation estimations more often than on rotation estimations, as the rotation is close to the desired one but the articulation is the one that seemingly increases the error. Furthermore, other cases show that for difficult poses with few distinctive features (as in the case of a closed palm) the method fails to estimate accurately the rotation of the hand.

4.5 Computational performance

All the experiments presented above were conducted using a computer equipped with an Intel Core i7-4790 CPU at 3.60GHz × 8 and 16 GB of RAM. The whole training on T_{F} takes about 3 days, while the estimation takes, on average, about 1 second per test image. It should be stressed, however, that the evaluation of an RBF takes on average less than 1.6 milliseconds using a single core of the CPU. In our

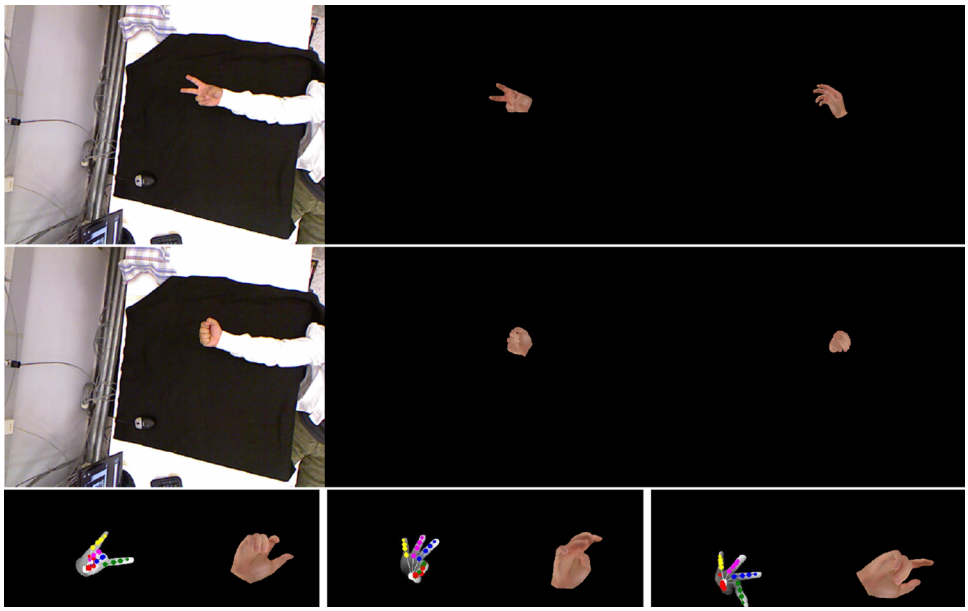

Fig. 9 Fail cases for both datasets (two first rows: *E*, bottom row: MSRA)

implementation, the real-world performance of the method is limited by input/output (I/O) operations. Since we have precomputed thousands of RBF networks at training time, it is impossible to have them all available in main memory. Therefore, whenever the iterative scheme needs to use a new RBF network, it is loaded from disk, slowing down the process. A smart caching scheme can significantly help, especially in the case of tracking a hand that moves rather smoothly. In such a case, the temporal continuity assumption will directly translate to fewer cache misses, effectively eliminating I/O operations time. Alternatively, one can just resort to adding enough RAM to hold all the RBFNs, 64 GB for the case of the RBFNs resulting from T_{F}.

5 Summary and discussion

We presented a method for single-shot hand pose estimation using a depth map as input. We created a large synthetic dataset and trained multiple RBFNs on it. We used an initial RBFN to compute a rough estimation of hand pose and articulation given a depth image. Then, with the use of specialized RBFNs, we managed to iteratively refine this approximation arriving to a final estimation. The use of RBFNs allows our method to make use of a large synthetic dataset. Despite the fact that the method is trained only on this synthetic dataset, we experimentally verified that it generalizes well to real-world data. Moreover, we validated our method in order to tune the hyper-parameters of the proposed model and we experimentally identified optimal values for all the hyper-parameters of the proposed method. We tested our method on two different testing datasets. The first one is a sequence we captured, and the second is a publicly available dataset that is commonly used to assess 3D hand pose estimation methods. We compared quantitatively our method with a recent state-of-the-art deep learning method from the relevant literature. To further illustrate the performance of our method, we also presented qualitative results. The obtained results demonstrate that RBFNs can generalize quite well when learning synthetic data. This is in contrast to most state-of-the-art methods that cannot even achieve cross-dataset generalization [32]. A RBFN does not search for patterns in local partitions of the data. Instead, it associates the input and output data according to the learned distances from its centers. A standalone RBFN might not perform satisfactorily. With the refinement of this initial suggestion, multiple specialized RBFNs can improve the final estimation, approaching the performance of state-of-the-art methods.

Future work includes the investigation of the effects of parallax distortion and the generalization across human hand sizes and shapes. Also, we may exploit the ability of the proposed method to incorporate rotation or pose constraints by applying it to the egocentric observation scenario as well as on the scenario of detecting a small vocabulary of predefined gestures.

Acknowledgements This work was partially supported by the EU project Co4Robots (H2020-ICT-2016-1-731869). Also co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE - INNOVATE (project code:T1EDK-01299 - HealthSign). The contribution of Paschalis Panteleris member of the CVRL/FORTH is gratefully acknowledged.

References

1. Angeline PJ (1998) Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. Lecture Notes in Computer Science: Evolutionary Programming VII, 1447:601–610, ISSN: 16113349. https://doi.org/10.1007/BFb0040811
2. Athitsos V, Sclaroff S (2003) Estimating 3d hand pose from a cluttered image. In: CVPR. IEEE Computer Society. Los Alamitos, vol 2, p 432, http://doi.ieeecomputersociety.org/10.1109/CVPR.2003.1211500
3. Bellon R, Choi Y, Ekker N, Lepetit V, Mike OL, Sonntag D, Tősér Z, Yoo K, Lőrincz A (2016) Model based augmentation and testing of an annotated hand pose dataset. In: Joint German/Austrian conference on artificial intelligence (Künstliche Intelligenz), Springer, pp 17–29
4. Bray M, Koller-Meier E, Van Gool L (2004) Smart particle filtering for 3d hand tracking. IEEE int'l conference on automatic face and gesture recognition, pp 675–680, https://doi.org/10.1109/AFGR.2004.1301612. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1301612
5. Broomhead DS, Lowe D (1988) Radial basis functions, multivariable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern(United Kingdom)
6. de Campos TE, Murray DW (2006) Regression-based Hand Pose Estimation from Multiple Cameras. In: 2006 IEEE computer society conference on computer vision and pattern recognition—Volume 1 (CVPR'06). IEEE, vol 1, pp 782–789. ISBN: 0-7695-2597-0. https://doi.org/10.1109/CVPR.2006.252. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1640833
7. de La Gorce M, Fleet DJ, Paragios N (2011) Model-based 3d hand pose estimation from monocular video. IEEE Trans PAMI, pp 1–15, ISSN 1939-3539. https://doi.org/10.1109/TPAMI.2011.33. URL http://www.ncbi.nlm.nih.gov/pubmed/21339527http://www.computer.org/portal/web/csdl/doi/10.1109/TPAMI.2011.33
8. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, 1995. MHS'95. IEEE, pp 39–43
9. Erol A, Bebis G, Nicolescu M, Boyle RD, Twombly X (2007) Vision-based hand pose estimation: a review. CVIU, 108(1-2):52–73. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314206002281
10. Fleishman S, Kliger M, Lerner A, Kutliroff G (2015) ICPIK: inverse kinematics based articulated-ICP. In: Computer vision and pattern recognition workshops, vol 2015, pp 28–35. ISBN: 9781467367592. https://doi.org/10.1109/CVPRW.2015.7301345
11. Ge L, Liang H, Yuan J, Thalmann D (2016) 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
12. Ge L, Liang H, Yuan J, Thalmann D (2017) 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In: Proceedings IEEE CVPR, pp 1991–2000. https://doi.org/10.1109/CVPR.2017.602
13. Heap T, Hogg D (1996) Towards 3D hand tracking using a deformable model. In: IEEE, vol 9, pp 140–145. ISBN: 0818677139
14. Katsavounidis I, Jay Kuo C-C, Zhang Z (1994) A new initialization technique for generalized Lloyd iteration. IEEE Signal Process Lett 1:144–146. https://doi.org/10.1109/97.329844
15. Le HNT, Quach KG, Zhu C, Duong CN, Luu K, Savvides M (2017) Robust hand detection and classification in vehicles and in the wild. In: IEEE conference on computer vision and pattern recognition workshops (CVPRW), 2017. IEEE, pp 1203–1210
16. Li P, Ling H, Li X, Liao C (2015) 3D hand pose estimation using randomized decision forest with segmentation index points. In: Proceedings of the IEEE international conference on computer vision, vol 2015 Inter, pp 819–827. ISBN: 9781467383912. https://doi.org/10.1109/ICCV.2015.100
17. Makris A, Kyriazis N, Argyros AA (2015) Hierarchical particle filtering for 3D hand tracking. In: IEEE conference on computer vision and pattern recognition workshops (CVPRW), 2015. pp 8–17. ISBN: 9781467367592. https://doi.org/10.1109/CVPRW.2015.7301343. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7301343
18. Makris A, Argyros A (2015) Model-based 3D hand tracking with on-line shape adaptation. In: British machine vision conference, pp 77.1–77.12. ISBN: 1-901725-53-7. https://doi.org/10.5244/C.29.77. URL http://www.bmva.org/bmvc/2015/papers/paper077/index.html
19. McCormick C (2013) Radial basis function network (rbfn) tutorial. http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial
20. Mittal A, Zisserman A, Torr PHS (2011) Hand detection using multiple proposals. In: BMVC, pp 1–11
21. Oberweger M, Lepetit V (2017) DeepPrior++: improving fast and accurate 3D hand pose estimation. In: Proceedings of the IEEE international conference on computer vision workshop, vol 840. https://doi.org/10.1109/ICCVW.2017.75. arXiv:1708.08325
22. Oberweger M, Wohlhart P, Lepetit V (2015a) Hands deep in deep learning for hand pose estimation. In: Computer vision winter workshop, pp 1–10. arXiv:1502.06807
23. Oberweger M, Wohlhart P, Lepetit V (2015b) Training a feedback loop for hand pose estimation. In: Proceedings of the IEEE international conference on computer vision, pp 3316–3324
24. Oikonomidis I, Kyriazis N, Argyros AA (2011a) Efficient model-based 3D tracking of hand articulations using kinect. In: BMVC, Dundee
25. Oikonomidis I, Kyriazis N, Argyros AA (2011b) Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In: ICCV. IEEE, pp 2088–2095
26. Panteleris P, Argyros A (2017) Back to RGB: 3D tracking of hands and hand-object interactions based on short-baseline stereo. In: Proceedings of the IEEE international conference on computer vision workshop, pp 575–584. https://doi.org/10.1109/ICCVW.2017.74
27. Qian C, Sun X, Wei Y, Tang X, Sun J (2014) Realtime and robust hand tracking from depth. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1106–1113
28. Rehg J, Kanade T (1994a) Visual tracking of high dof articulated structures: an application to human hand tracking. ECCV, vol 35–46. URL http://www.springerlink.com/index/MK50G5121V7N6236.pdf
29. Rehg JM, Kanade T (1994b) Visual tracking of high dof articulated structures: an application to human hand tracking. In: ECCV. Springer
30. Romero J, Kjellstrom H, Kragic D (2009) Monocular real-time 3d articulated hand pose estimation. In: IEEE-RAS int'l conference on humanoid robots. https://doi.org/10.1109/ICHR.2009.5379596. URLhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5379596
31. Sun X, Wei Y, Liang S, Tang X, Sun J (2015) Cascaded hand pose regression. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 824–832
32. Supancic JS, Rogez G, Yang Y, Shotton J, Ramanan D (2015) Depth-based hand pose estimation: data, methods, and challenges. In: Proceedings of the IEEE international conference on computer vision, pp 1868–1876

33. Tagliasacchi A, Schröder M, Tkach A, Bouaziz S, Botsch M, Pauly M (2015) Robust articulated-ICP for real-time hand tracking. In: Computer graphics forum
34. Tang D, Chang HJ, Tejani A, Kim T-K (2014) 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
35. Tipping ME (2001) Sparse Bayesian learning and the relevance vector machine. J Mach Learn Res 1(Jun):211–244
36. Tompson J, Stein M, Lecun Y, Perlin K (2014) Real-time continuous pose recovery of human hands using convolutional networks. ACM Trans Graph 33:169
37. Wan C, Yao A, Van Gool L (2016) Direction matters: hand pose estimation from local surface normals. In: European conference on computer vision, pp 554–569, Springer. arXiv:1604.02657
38. Wan C, Probst T, Van Gool L, Yao A (2017) Crossing nets: combining GANs and VAEs with a shared latent space for hand pose estimation. In: CVPR. https://doi.org/10.1109/CVPR.2017.132. arXiv:1702.03431
39. Wang RY, Popović J (2009) Real-time hand-tracking with a color glove. ACM Trans Graph 28(3):1. ISSN: 07300301. https://doi.org/10.1145/1531326.1531369. URL http://portal.acm.org/citation.cfm?doid=1531326.1531369
40. Zhang J, Jiao J, Chen M, Qu L, Xu X, Yang Q (2016) 3D hand pose tracking and estimation using stereo matching. arXiv:1610.07214