Top-down model fitting for hand pose recovery in sequences of depth images

Meysam Madadi[a,b,], Sergio Escalera[a,c], Alex Carruesco[d], Carlos Andujar[d], Xavier Baró[a,e], Jordi Gonzàlez[a,b]

[a] *Computer Vision Center, Edifici O, Campus UAB, 08193 Bellaterra (Barcelona), Catalonia Spain*
[b] *Dept. of Computer Science, Univ. Autònoma de Barcelona (UAB), 08193 Bellaterra, Catalonia Spain*
[c] *Depl. Mathematics and Informatics, Universitat de Barcelona, Catalonia, Spain*
[d] *ViRVIG-Moving Research Group, UPC-BarcelonaTech*
[e] *Universitat Oberta de Catalunya, Catalonia, Spain*

Abstract

State-of-the-art approaches on hand pose estimation from depth images have reported promising results under quite controlled considerations. In this paper we propose a two-step pipeline for recovering the hand pose from a sequence of depth images. The pipeline has been designed to deal with images taken from any viewpoint and exhibiting a high degree of finger occlusion. In a first step we initialize the hand pose using a part-based model, fitting a set of hand components in the depth images. In a second step we consider temporal data and estimate the parameters of a trained bilinear model consisting of shape and trajectory bases. We evaluate our approach on a new created synthetic hand dataset along with NYU and MSRA real datasets. Results demonstrate that the proposed method outperforms most recent pose recovering approaches, including those based on CNNs.

Keywords: hand pose recovery, shape description, depth image, hand segmentation, temporal modeling

1. Introduction

Hand pose recovery has attracted great interest in recent years due to the availability of affordable depth cameras. Depth sensors have allowed researchers to use non-invasive, accurate approaches to hand pose estimation, which are more robust to illumination and color changes than standard RGB cameras. These features have lead to significant advances in multiple applications including human-computer interaction, virtual reality, robot learning and gesture recognition, just to name a few [4, 5, 7, 8].

Although recent hand tracking approaches based on depth cameras achieve high performance for some applications, there are still several open challenges to tackle, such as finger self-occlusion, hand-body occlusions, low resolution/noisy depth images, and above all, the inherent complexity of modeling hand motion due to its highly articulated nature. Available datasets mainly provide front-face hand deformations, which are not suitable to compare state-of-the-art approaches against hard cases with large occlusions. To the best of our knowledge, little attention has been paid to incorporate temporal motion information in hand pose recovery problems. As an example, Oikonomidis *et al.* [15] only initialized the model using previous frame.

In this paper, a solution to the problem of hand pose recovery in depth image sequences is proposed. The solution combines both spatial and temporal information in a top-down strategy. We present a system for efficient hand pose recovery in non-controlled settings involving self-occlusions. Based on current trends towards minimizing pose parameters in the space of nearest candidates [34, 21], we exploit an effective shape descriptor to extract such nearest candidates. As in [26] we estimate each object part separately while reducing the search space. We first extract palm joints, which provide a basis for fingers, using nearest candidates. Following [18] we

[*] Corresponding author: mmadadi@cvc.uab.es (Meysam Madadi)

define an efficient objective function and then minimize parameters of each finger model to fit with its appearance. Our function is different from [18] since they extract fingertips while we accurately segment fingers. Thanks to this objective function we get a fast convergence to the finger model parameters while handling occluded parts.

Motivated by [37], our estimated joints are applied in a sequence of frames to minimize parameters of a trained bilinear model [1] consisting of shape and trajectory bases. This process further refines the estimation of occluded parts. Fig. 1 shows our method pipeline: nearest neighbors extraction, hand segmentation, single-frame pose recovery, and temporal pose recovery. Our approach has proven to be more robust under large viewpoint sets and complex hand poses than state-of-the-art approaches when data is balanced for different viewpoints and poses. To evaluate our method under such situations, we created a synthetic dataset with +600K hand pose samples for single-frame pose recovery and +1M frame sequences for temporal pose recovery, with high deformations and occlusions in both learning and test sets. Although, egocentric datasets have been recently introduced [20], hand-object interaction is not within the scope of this paper. Though, we evaluate on real datasets like NYU [31] and MSRA [26].

The rest of the paper is organized as follows. Section 2 reviews state-of-the-art works in the field. Section 3 presents the proposed system. Results are shown in section 4, and finally, section 5 concludes the paper.

2. Related Work

The field of hand pose estimation has become very active due to the use of depth sensors. An excellent survey on existing methods can be found in [8]. In this section we focus on those approaches most related to our contribution. Hand pose estimation methods can be roughly divided into model-based methods and data-driven methods [6, 17].

Model-based techniques consider an a priori 3D hand model whose pose is determined over time by some tracking procedure [21, 11, 18], like the Particle Swarm Optimization presented in [15]. Unfortunately, these approaches require some kind of accurate initialization, and due to the fast motion and non-rigid nature of hands, together with finger self-occlusions, it is still a challenge for single-hand trackers to correctly maintain the state of an animated 3D hand model over time. In recent works, while some works propose more advanced hand models [30], others try to sample hypotheses by physical constraints [19].

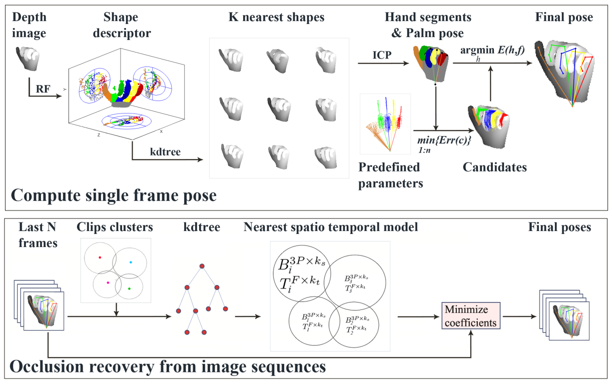

Depth image
Shape descriptor
K nearest shapes
Hand segments & Palm pose
Final pose
RF
kdtree
ICP
argmin E(h,f) h
min{Err(c)} 1:n
Predefined parameters
Candidates
Compute single frame pose
Last N frames
Clips clusters
kdtree
Nearest spatio temporal model
Final poses
$B_i^{3P\times k_s}$ $T_i^{F\times k_t}$
$B_3^{3P\times k_s}$ $T_3^{F\times k_t}$
$B_1^{3P\times k_s}$ $T_1^{F\times k_t}$
$B_2^{3P\times k_s}$ $T_2^{F\times k_t}$
Minimize coefficients
Occlusion recovery from image sequences

Figure 1: Diagram of the proposed method. In the first step, a single-frame hand pose is estimated. First palm joints and finger segments are recovered through nearest shapes. Then finger models are fitted using extracted candidates. In the second step, temporal data is incorporated to refine first step estimation.

On the other side, data-driven methods directly predict at each frame the pose of the hand by learning depth and image features [26]. Contrary to using hand trackers, which lead to model drift over time, single-frame detection methods are initialized at each frame, thus recovering more easily from estimation errors [21]. Multiple procedures based on Random Forests (RF) have emerged including Hough Forests [34], Random Decision Forests [10] and Latent Regression Forests [27], as detailed in [8]. Unfortunately, the number of occluded joints is commonly bigger in hands than in human bodies. As a result, techniques based on RF usually require huge training sets, and some kind of viewpoint estimation is needed in order to improve performance [28]. Some data-driven works analyze the hand in the space of nearest shapes in order to reduce the search space [21] or approximate unknown pose parameters through matrix factorization [3].

Following current trends in Computer Vision, although both the architecture and weight initialization of a neural network strongly determine its performance, CNN-based techniques continuously improve the state-of-the-art accuracy on different benchmarks. Tompson *et al.* [31] optimized an inverse kinematic approach based on joint heatmaps generated by CNN for 2D joints estimation. Oberweger *et al.* [13] optimized a shallow CNN based on embedded space of hand pose. While Ye *et al.* [35] integrates cascaded and hierarchical regression into a CNN framework, Ge *et al.* [12] define three viewpoints based on hand point cloud eigenvectors and fuse heatmaps generated by CNN for each view. Researchers even combined generative models with CNNs where generative model is used to compute a feedback error [14], as a forward kinematics layer [38], or share latent space [32].

Temporal information and trajectory analysis, besides the shape itself, provide discriminative information to analyze shape and recover occluded parts. Works from structure from motion, such as matrix imputation [23], statistical model analysis and non-rigid structure from motion [16, 29], showed the benefits of using temporal information for shape analysis. Zhou *et al.* [37] proposed a spatio-temporal model for the problem of human pose recovery. Although their approach obtains promising results, the complexity of the minimization problem makes it not applicable for all types of pose deformations.

In all the aforementioned approaches, a balanced tradeoff between real-time and accuracy performance is maintained. Techniques exhibiting high accuracy typically work at low frame rates, thus becoming unsuitable for interactive systems of spatially-immersive scenarios.

3. Methodology

The basic idea of the proposed method is to recover a hand pose through a combination of part-based model fitting and data-driven approaches in a single frame and, afterward, refine occluded joints in a sequence. As illustrated in Fig. 1, we first extract nearest shapes by introducing a shape descriptor (Sec. 3.1). We apply nearest shapes with two purposes: 3D palm joints recovery and hand segmentation (Sec. 3.2). Given the palm joints and segmented fingers, we extract a number of candidates for each finger using a set of predefined examples. We then send these candidates to the optimization process to minimize an objective function which fits a finger model to the segmented finger (Sec. 3.3). We minimize the parameters of each finger separately. Finally, occluded joints are refined by solving the coefficients of the trained bilinear model in a sequence of F images (clip). We cluster clips in order to reduce non-linearity (Sec. 3.4).

In order to evaluate our method on highly-variable poses and viewpoints, as well as temporal analysis, we created a rich synthetic dataset mimicking the features of commodity depth cameras (Sec. 4.1). We illustrate some properties of the hand model used to create this dataset in Fig. 2. We created a hand model with 25 semantic segments used as low-level pixel labels in the dataset. At a higher level of semantics, we segmented the hand by assigning each pixel a label from the set $L = \{l_1, ..., l_6\}$, where L represents fingers and the palm. Next, we detail the main components of the proposed approach.

3.1. Nearest shapes extraction

Several state-of-the-art works [21, 34, 10] use Random Forest (RF) to extract viewpoint or nearest neighbors from the deeper branches of the trees trained on a particular dataset. Such methodology can be seen as stochastic shape extraction and leads to some irrelevant nearest shape recovery. Besides that, this approach is not efficient for large scale datasets. On the other hand, common statistical shape descriptors try to find a correlation among the components composing the shape and grouping them into bins.

In this work we train a classifier to segment a hand into a set $S = \{s_i\}_{i=1}^{25}$ with 25 classes defined in the dataset and group probability responses of the classifier into log-polar bins. Therefore we first select a fixed random number of pixels from the hand and estimate each class response for each pixel applying the trained classifier. For aggregating the responses into bins, we reconstruct a point cloud of selected pixels and divide XYZ axes into three axis pairs XY, XZ and YZ. Thus, we map the point cloud to front, top and side views and apply measurements separately on each view.

We compute the log-polar binning based on shape context [2]. Let $q = \frac{1}{N}\sum_{i=1}^{N} P_i$ be the center of the point cloud where N is the number of points and let $P_i \in \mathbb{R}^3$ denote the i-th point in world coordinates. We set q as the center of the log-polar coordinate system. Then histograms of different views (front view for instance) are computed as:

$$H_{xy}(k, c) = \sum_{i=1}^{N} \{R_{ic} | (P_i^{xy} - q^{xy}) \in bin_{xy}(k)\}, \qquad (1)$$

where R_{ic} denotes probability responses of the i-th point and c-th class predicted by the classifier, and k is the bin number. Finally histograms at each view are concatenated and normalized. Applying such descriptor we discriminate both spatial and class dependencies of different shape points into bins, being fast to compute, invariant to slight rotations of the hand and robust against boundary noise due to the random selection of points. We show an illustration of our descriptor in Fig. 1. We set 8 angle and 5 radius bins as the log-polar binning parameters of the shape descriptor. Finally, for a fast extraction of the K nearest shapes, a kd-tree is trained based on the extracted features.

Segmentation classifier details. We apply the work of Shotton *et al.* [22] as our segmentation classifier. Next we explain offset features extraction. Let $O \in \mathbb{R}^{2\times n}$ contains n random offsets uniformly distributed in the range $[-1, 1]$. Offsets O can be adapted to any depth camera by taking camera focal length into account. Therefore we update O by multiplying it by a scaling factor 120mm and camera focal length. Afterward we compute the feature δ_{ij} at pixel P_i given the offset j as:

$$\delta_{ij} = I\left(P_i + \frac{O_j^1}{I(P_i)}\right) - I\left(P_i + \frac{O_j^2}{I(P_i)}\right) \tag{2}$$

$$\theta_{ij} = \begin{cases} \infty & \text{if } \delta_{ij} > d_{max}, \\ -\infty & \text{if } \delta_{ij} < -d_{max}, \\ \delta_{ij} & \text{otherwise}, \end{cases} \tag{3}$$

where $I(.)$ denotes depth at given pixel in the image, d_{max} is the maximum depth capturing device can take and O_j^1 and O_j^2 denotes the jth offset of the first and second offset array, respectively. Image I must be set by a high value for background pixels beforehand. We train 14 trees with depth 20 using 100 random features, 150K random samples with a subset of 500 randomly selected pixels per frame and 1000 offsets. To fit data in memory we train each tree with 23% of random data.

3.2. Palm and finger extraction

Given the nearest shapes and their corresponding joints, one could minimize coefficients of a weighted sum of basis models (like PCA) to extract hand pose. However we observed that this process does not perform well in practice. Instead, we divide the problem of pose estimation into two subproblems: palm pose estimation, as global hand pose, and fingers pose estimation. Each problem

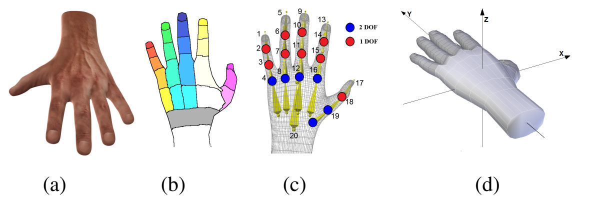


Figure 2: Finger models. a) Hand 3D model used for the dataset generation, b) unique color labels used to identify surface points on the hand, c) DOF for different joints. Joints are indexed by assigned numbers. This figure also shows how skeleton is fitted inside hand. d) Palm coordinate system. Finger parameters are computed based on this coordinate system.

is solved separately. In the model, palm pose is first detected. We assume palm is rigid and refer to palm pose as a composition of wrist and base joints of all fingers except the thumb in 3D space (i.e. joints 4, 8, 12, 16 and 20 in Fig. 2(c)). Sun *et al.* [26] regress palm pose by iterative refinement of an initial pose. Sharp *et al.* [21] estimate a global view point and iteratively fit a model by generating some hypothesis candidates. It has been shown that NN-based approaches perform well in practice [8]. In this work we rely on extracted nearest shapes to both estimate palm pose and segment the hand.

Nearest shapes can vary in shape and pose and need to be aligned to each other beforehand. We use palm joints of nearest shapes to align them through Procrustes analysis. This provides a uniform and smooth distribution of palm points in the point cloud of the nearest shapes. Given this point cloud with their corresponding labels l_i, we find an affine transformation A with scaling factor s to hand point cloud P by applying iterative closest point matching (ICP) [36]. For a faster convergence, we modify ICP process to find closest points from group of points with the same label. Pixel labels of test frame were estimated by RF beforehand. Then, we get the palm joints by transforming the nearest shape joints given A and s.

Although our trained RF could segment the hand, it is not reliable under some situations, especially for distinguishing fingers (See Fig. 4.2 for some samples). Correct hand segmentation is critical for the accuracy of our approach. Since we fit a finger model based on segmented pixels of that finger, an incorrectly segmented finger instantly causes a failure pose. Quadratic discriminant analysis provides a proper way to assign each point in the point cloud in query a label from aligned point cloud of nearest shapes efficiently.

3.3. Pose estimation

We fit a simple finger model for each finger separately to get fingers poses. Each finger model S is composed of three cylinders and half-spheres except for the thumb, which is composed of an ellipsoid, two cylinders and three half-spheres. Finger model parameters are computed based on the palm coordinate system (see Fig. 2).

Given hypothesis parameters h, camera calibration parameters, palm pose and finger properties like length and diameter of bones, we can render a 3D model of the finger S and project it onto the image plane. Let I_M, M_M and M_F be the depth image of the projected finger model, the projected finger model mask and the segmented finger extracted from Sec. 3.2, respectively. Then, we set the background of I_M to zero and define $M_{in} = M_F \wedge M_M$ and $M_{out} = \neg M_F \wedge M_M$ (see Fig. 3). The goal is to find hypothesis parameters h that best fit the model to the finger in query. Therefore we define the objective function $E(h, I)$ to compute the amount of discrepancy between I_M and I with respect to M_F through:

$$E_1 = 1 - \frac{\#M_{in}}{\#M_F + \epsilon}, \tag{4}$$

$$E_2 = \begin{cases} 10 & \text{if } M_{in} \subset \oslash, \\ E_1 \frac{mean(min(|I_M(M_{in}) - I(M_{in})|, \lambda))}{\lambda} & \text{if } M_{in} \not\subset \oslash, \end{cases} \tag{5}$$

$$E_3 = \frac{\#(I_M(M_{out}) < (I(M_{out}) + \tau))}{\#(M_{out}) + \epsilon}, \tag{6}$$

$$E(h, I) = w_1 E_1 + w_2 E_2 + w_3 E_3, \tag{7}$$

where λ and τ are some depth difference thresholds. Term E_1 computes overlapping area between M_M and M_F normalized by $\#M_F$. Term E_2 controls the mismatching of depth in the overlapping area M_{in}. Such a mismatching depth energy is directly related to $\#M_{in}$. We consider this situation in the first case of Fig. 3. A small area M_{in} can generate a lower depth mismatching energy which can cause a wrong matching. Therefore we scale E_2 by multiplying it to E_1 as a function of $\#M_{in}$ to reduce the effect of $\#M_{in}$ in the depth mismatching energy. We add term E_3 to avoid finger collision to non overlapping pixels M_{out}. We consider this situation in the second and third cases of Fig. 3. We add the term ϵ to avoid division by zero and

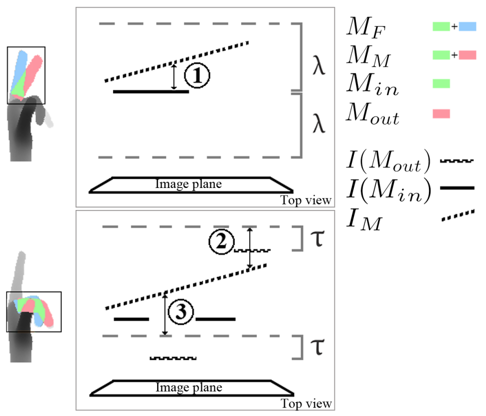


Figure 3: Objective function E. We jointly maximize overlapping area M_{in} (E_1) and minimize depth discrepancy between generated model and hand finger (E_2). We show the overlapping area (green) can have a relation to the depth difference. A small depth difference may not guarantee a good matching, and therefore, we penalize it by multiplying it to the normalized non-overlapping area (blue). Hence a small depth difference is only useful if blue area is small as well. In the second and third cases we should avoid collision between the model surface I_M and other finger surfaces available in M_{out} (E_3). τ controls the area between fingers.

set it to a low value. The number 10 in E_2 is a maximum energy, and w_1, w_2, and w_3 are some fixed weights.

Particle swarm optimization (PSO) is a commonly used approach to minimize such an objective function. However, it is not efficient for minimizing over all possible parameters and it is easily trapped to local minima [9]. In order to cope with such problems, we predefine a low number (300 in our case) of sample fingers which cover most finger poses and evaluate a simple function over all predefined samples to select the best candidates. We use simple facts to design this evaluation function. As the first rule, all finger joints should be located in the hand mask after projecting them onto the image plane. Secondly, the joints should have at least a depth equal to the hand surface depth plus a threshold. Let $J_f^{xyz} \in \mathbb{R}^{3 \times f_N}$ be the matrix of 3D locations of the joints belonging to the finger f and $J_f^{uv} \in \mathbb{R}^{2 \times f_N}$ be the matrix of 2D locations of the joints of finger f after projecting onto the image plane where f_N is the number of joints. Therefore all joints should meet the constraint $I(J_f^{uv}) + \omega \leq J_f^z$, where ω is a constant value. Since we set the background of I to a high value, this constraint satisfies the first rule as well. We consider a third rule for visible fingers such that the joints should not be far from the finger point cloud. We formulate these rules for finger f as:

$$C_{fd} = \{I(J^{uv}_{fi}) - J^{z}_{fi} + \omega\}, i \in 1, ..., f_N, \tag{8}$$

$$C_f = \begin{cases} C_{fd} & \text{if } M_F \subset \oslash, \\ \{C_{fd}, \gamma \| \overline{I^{xyz}(M_F)} - \overline{J^{xyz}_f} \|\} & \text{if } M_F \not\subset \oslash, \end{cases} \tag{9}$$

$$Err(C_f) = \sum_{\{c | c \in C_f \wedge c \geq 0\}} min(c, \varphi), \tag{10}$$

where $\overline{I^{xyz}(M_F)}$ is the center of finger point cloud and $\overline{J^{xyz}_f}$ is the center of the candidate joints. ω is a depth threshold that controls the distance of the joints to the hand surface. γ is a weight to balance different terms. Eq. 9 is treated as a constrained inequality and therefore negative values are desirable. As a consequence we sum over positive costs limited by constant threshold φ Eq. 10 to evaluate each sample finger. Finally a number of samples with the lowest error are selected as candidates and feed into PSO. We set the number of generations and population size to 5 and 30, respectively. For completely occluded fingers (i.e. $M_F \subset \oslash$) we apply Eq. 9 and make an average finger from outcomes. All the thresholds and weight terms are experimentally set to some fixed values as follows: $\tau = 15$, $\lambda = 25$, $w_1 = 0.25$, $w_2 = 0.65$ and $w_3 = 0.1$, $\omega = 8$, $\gamma = 4$ and $\varphi = 50$.

3.4. *Spatio-temporal pose recovery*

Time-varying spatial data is involved in a vast range of computer vision applications [33, 29] and proved to be useful in extracting missing data. Spatial correlation or trajectory analysis of independent points solely fails to model all information in spatio-temporal data. Akhter *et al.* [1] combined two linear shape and trajectory bases learned by discrete cosine transform and SVD to exploit spatio-temporal regularities. We follow this work to generate linear bases of hand data. To train bilinear bases, we have generated a dataset including smooth deformation of fingers in a reference view in a sequence. The advantage of keeping a reference view is that all the frames are previously aligned by their palm joints. Then we extract fixed-length clips by a sliding window over the sequences. A clip is represented by $Q \in \mathbb{R}^{F \times 5D}$ where F is the number of frames and D is the number of parameters for each finger. Clip Q can be factorized by TCB^T (as introduced in [1]) where $T \in \mathbb{R}^{F \times k_t}$ and $B \in \mathbb{R}^{5D \times k_s}$ are learned trajectory and shape structures and $C \in \mathbb{R}^{k_t \times k_s}$ is the coefficient matrix. Given the learned T and B, the goal is to minimize a function over coefficients C in order to extract clip Q at test time.

A common problem with linear basis models like PCA and SVD is that they are sensitive to the correlation coefficient or distribution of the data. A solution is to divide the space of clips (e.g. clustering) in order to provide more correlation among data. However, this solution is not exact. In, [37] authors search over all clusters to find best models. However, this is not suitable for a huge number of clusters, as in our case. In order to cope with previous issues, we propose a fast and approximate solution to find best models.

In the training step, we apply k-means to cluster data. We regenerate each cluster by extracting νN nearest clips to the cluster centroid where N is the number of clips in the cluster and $\nu > 1$. In fact, we extend each cluster with overlapping to its adjacent clusters. Afterwards, we train bilinear models T and B on each cluster (as described in [1]). This causes the models to be more robust at cluster boundaries.

At test time, given the last clip Q (initialized using Sec. 3.3) and parameters visibility $V \in \{0, 1\}^{F \times 5D}$ (extracted from RF), we are able to find nearest clips in a dataset by a trained kd-tree. However, visible and invisible joints have the same weight in the clips and possible errors in the initial estimation can cause a false nearest cluster. More specifically, the task is to find a cluster that best describes both the appearance and occluded parts, and then minimize a function on coefficients C. Therefore we define the objective function $STC(Q, V, T, B, \mu, \sigma)$ as:

$$STC = \sum_{f=1}^{F} \sum_{i=1}^{5D} V_{fi} |Q_{fi} - Q^r_{fi}| + \beta \sum_{f=1}^{F-1} \Psi^{f,f+1}, \tag{11}$$

where Q_{fi} extracts the i-th parameter in frame f, $Q^r = TCB^T$ denotes reconstructed parameters through coefficients C, Ψ is a smoothness function among correspondent parameters in frames f and $f+1$, and β is a regularization weight. We define the smoothness function as:

$$\Psi^{f,f+1} = \sum_{i=1}^{5D} \neg(V_{f,i} \wedge V_{f+1,i}) \left| \frac{Q^r_{f,i} - Q^r_{f+1,i} - \mu_{fi}}{\sigma_{fi}} \right|, \tag{12}$$

where μ_{fi} and σ_{fi} are precomputed mean and standard deviation distance for i-th parameter in the frame f for each cluster, respectively. The first term in Eq. 11 denotes the appearance cost and the second term penalizes large movements of the occluded joints.

We approximate the best cluster by first extracting a number of nearest clusters, traversing a trained kd-tree using clip Q. This kd-tree is trained based on clusters centroids. Subsequently, we generate a number of random poses around clip Q and evaluate function STC on them for each extracted nearest cluster. Finally, we take that cluster which generates minimum average error.

Efficient minimization of Eq. 11 is required. Levenberg-Marquardt algorithm is a standard minimization technique, although finding a good initial point to minimize Eq. 11 makes the problem intractable. In order to overcome this problem, we use PSO with a number of randomly selected particles around Q and apply $T^T R(B^T)^{-1}$ for all random clips to generate initial particles, where R is a random clip and T and B are trained bliniear structures of the best cluster. To have a fair distribution of fingers and removing undesired clips, we apply Eq. 10 on all fingers for all random clips and select a subset of best candidates by sorting clips regarding their maximum finger error. As a consequence, the solution is achieved in a few generations. We set the number of generations and population size to 5 and 100, respectively.

We use finger parameters in all frames as a trajectory descriptor which is invariant to finger length and hand shape. Finger parameters have an advantage versus the 3D joints locations since we have more control on them, like adding constraints or generating a more meaningful shape without adding extra regularization. Given that this process mainly improves occlusion recovery, we combine the recovered invisible joints to the visible joints estimated in the initial step as the final pose. In the experiments, we show that initial pose estimation has a low error which is reliable enough to be used in the occlusion refinement process. We apply full rank matrices to train the bilinear model, with $k_s = 7$, $k_t = 7$, clip length $F = 7$ frames and $\beta = 0.1$.

4. Experiments and results

In order to present the results, we first discuss the considered data and the experimental setup.

4.1. Dataset and setup

Data generation. Datasets were generated with Blender 2.74 using a detailed, realistic 3D model of a human adult male hand (Fig. 2(a)). The model was rigged using a hand skeleton (Fig. 2(c)) with four bones per finger, reproducing the distal, intermediate, and proximal phalanges, as well as the metacarpals. The thumb finger had no intermediate phalanx and was controlled with three bones. Additional bones were used to control palm and wrist rotation. Unfeasible hand poses were avoided by defining per-bone rotation constraints. All finger phalanges had only 1-DoF rotation (for finger flexion/extension) but metacarpals had 2-DoF rotation to allow for finger adduction/abduction. This resulted in 4-DoF per finger (except for the thumb), which proved to be enough to reproduce all reasonable poses in the context of gesture-based interaction (see some sample poses in Fig. 4).

Points on the hand's surface were assigned a unique color label identifying the underlying skeleton joint, as shown in Fig. 2(b). The palm center was assumed to be roughly at the metacarpals' centroid.

The animated hand model was rendered using a virtual camera reproducing the image resolution and the intrinsic parameters of the target depth sensor (Kinect-2). The virtual camera was always aiming at the hand, from a view direction which was chosen randomly from a uniform discretization of the Gauss sphere (we used 320 directions associated with the normal vectors of a subdivided icosahedron).

Training datasets. We generated two different training sets. For the first dataset, we generated three pieces of data: a color image (pixel labels), a depth image, and a text file containing the location of the skeleton joints. Each training example was generated by randomly choosing a view-direction and a hand pose (Fig. 4). We generated over 600K samples for this dataset and used it for RF training and nearest neighbor extraction.

For the second dataset, we just produced the text files containing the joints locations. Camera viewpoint was fixed in this dataset in order to benefit from a reference

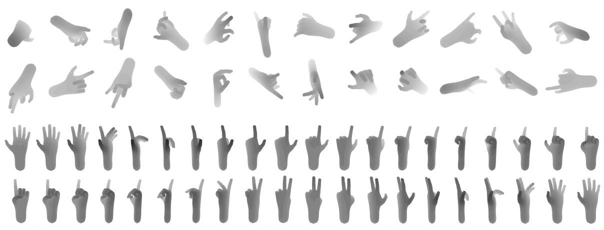


Figure 4: Upper two rows are some sample poses, lower two rows are a small sample set of the depth images generated for the test set. The image shows ten interpolation frames between four predefined hands poses.

viewpoint and palm joints were aligned. We provided temporal data in this dataset including a smooth interpolation between pairs of key poses. Key poses were chosen either randomly or from a small set of predefined poses. We included different deformation speeds in this dataset. The unique motion range of the thumb (which includes opposition-reposition, besides flexion-extension and adduction-abduction) forced us to prevent finger self-intersections by inserting additional frames. This guaranteed feasible and natural hand movements. We generated over 1200K frames for this dataset and used it to extract clips and train bilinear model.

Test dataset. For generating this dataset we followed the same rule as our second dataset except we produced the color labels, depth images, and text descriptions, and camera rotations were smooth along pose interpolation frames (see Fig. 4). We generated over 8K frames for this dataset.

Real dataset. To the best of our knowledge there is no real dataset consisting of both hand segments and pose at the same time. Some datasets just provide fingertips [25]. However, we selected MSRA [26] and NYU [31] datasets to evaluate our method on real benchmarks. MSRA contains 17 hand posture categories captured from 9 subjects including 76,500 frames. Hands are rotated in a 90 degree range near to the camera, depth image has low amount of noise and the definition of joints locations is similar to ours. A few invalid frames are available in this dataset and in a few cases joints locations do not follow hand appearance. NYU dataset has a broader range of hand pose and viewpoint than MSRA. However, it contains just one subject in the training set and two subjects in the test set. Depth images in this dataset are noisy and fingers are missed in some cases. This dataset contains around

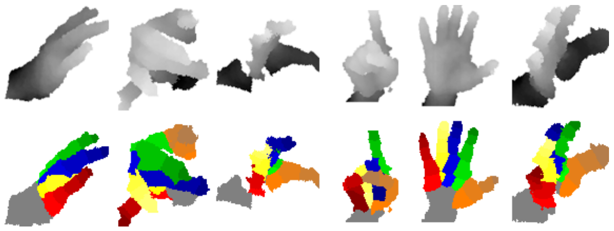


Figure 5: Some examples of NYU dataset segmented by using joint distances to points.

73K training and 8K testing frames which are captured by three depth cameras from multiple viewpoints to provide accurate groundtruth pose. We then provide a groundtruth segmentation on these datasets using available joint locations and the distance to the point cloud. Specifically, we compute the distance from each joint to all points. This creates a matrix with size equal to the number of points and number of classes. Then, the index of minimum value is taken as groundtruth label for each point. To follow the same definition of segments as our synthetic hand dataset, we add auxiliary joints to the palm. We show some examples of this procedure in Fig. 5.

4.2. Results

Evaluation metric. We used the 3D Euclidean distance from joints to groundtruth for evaluating the different approaches. We also measured the success rate as in [26] to compute percentage of each error threshold. We compute success rate based on worst joint error per frame and average joints error per frame.

Evaluation on the synthetic dataset. For comparison, we used as the baseline a transformed average shape from the nearest neighbors according to our shape descriptor and ICP. We compared PSO vs. greedy for single-frame pose recovery as well. In the greedy approach after applying our population selection proposal (Eq. 10), the best candidate was selected by evaluating Eq. 7. We include our occlusion refinement approach and we show how it can be combined with greedy to slightly improve occluded joints. Fig. 6(a) shows the per-joint average error (mm) for different approaches. As it can be expected PSO performs slightly better than the greedy approach. However, the difference is not significant and the greedy approach runs faster than PSO. Joints belonging to the palm exhibit accurate palm pose recovery even in quite difficult poses which is quite critical for recovering the pose of individual fingers. Notice that the baseline is the most accurate approach for the thumb joints. A possible explanation is that the thumb has higher movement range than other fingers and it is thus hard to recover with model-based approaches. Bilinear optimization solely does not improve the overall error and resulted in lower accuracy than single-frame techniques, but when combined with the greedy solution we could improve occluded parts poses by 3.7mm (i.e. visible joints from greedy and occluded ones from bilinear optimization). Although this is not a big improvement, the results show the benefits of incorporating temporal data. However, increasing the number of frames within each clip adds complexity to the bilinear coefficient optimization and precludes real-time performance.

For the current version of the system, the hand can not be occluded by any other object. Since we use ICP and QDA, model drifts might occur when the number of visible pixels from the hand is dramatically reduced (due to pose, viewpoint, camera noise, or missing data). Not availability of nearest shapes does also influence the pose recovery process for both hand segmentation and palm pose recovery tasks.

We also compared our proposal with the DeepPrior [13] Convolutional Neural Network approach. Fig. 6(b) illustrates the success rate error among proposed methods and DeepPrior. DeepPrior shows the lowest accuracy. This could be because of the high pose variability and presence of occlusions [13]. We trained DeepPrior with 300K samples, 200 epochs and learning rate 0.001. We also show some qualitative results in Fig. 7(a) and 7(c).

By incorporating QDA for segmenting hand into the set L, we could improve RF segmentation performance. Since each segment l_i has a number of sub-segments from the set S, for a given pixel P belonging to segment l_i, we discard those probabilities (given by RF) not belonging to l_i, and consider the index of the maximum probability as the final estimated label for that pixel. Fig. 4.2 illustrates some qualitative results of RF segmentation performance and its improvement in a number of frames. We compared greedy vs. baseline and greedy vs. bilinear optimization for some examples in Fig. 13 and 9(b). The purpose of these graphs is to compare how different methods behave in a sequence of frames.

Evaluation on MSRA dataset. To report results and compare to the state-of-the-art on this dataset we applied

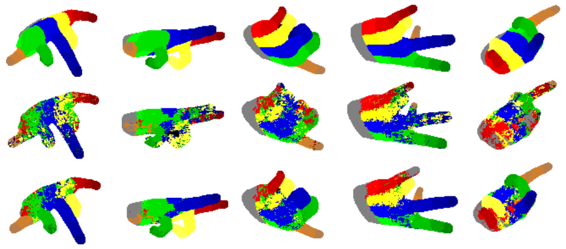
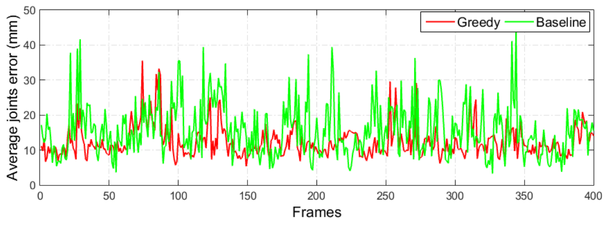
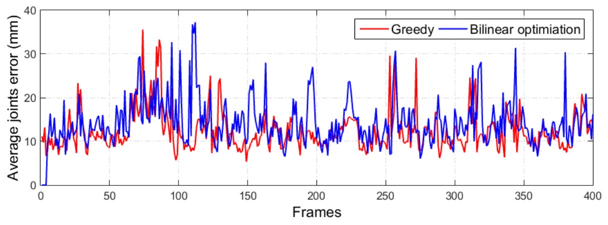


Figure 8: Qualitative RF performance. Rows from top to bottom: ground truth, RF, and improved RF results. We improve RF segmentation performance by around 20%.



(a)



(b)

Figure 9: a) Greedy vs. baseline. b) Greedy vs. bilinear optimization.

a 9-fold cross validation, where each fold corresponds to one subject. Fig. 6(c) illustrates success rate of our baseline approach comparing to [26]. Table 1 shows per-joint average error in comparison to state-of-the-art approaches. Notice that our baseline method clearly outperforms most of the state-of-the-art approaches on this dataset. This datast has a uniform distribution of pose and viewpoint and these results show the robustness and accuracy of our methodology against highly variable poses. Fig. 7(b) shows some qualitative results on this dataset.

Evaluation on NYU dataset. To evaluate our procedure on this dataset, we first extracted non-redundant training samples to balance data and generated new samples by randomly rescaling hands to make RF robust against hand size. We used 76K samples in total. We

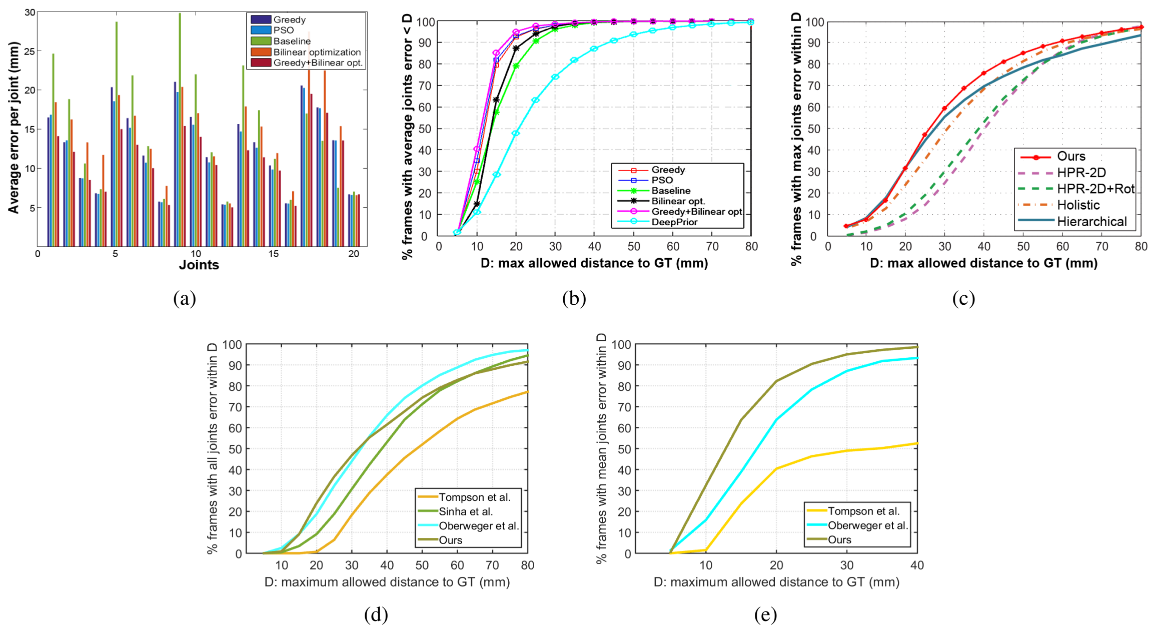


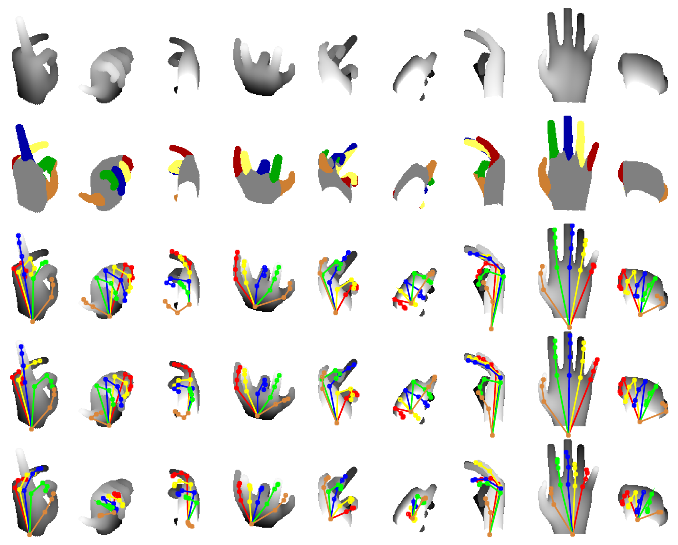
Figure 6: a) Error per joint. Joint arrangement is shown in Fig. 2(c). The mean errors are 12.86, 12.40, 15.16, 14.72 and 11.09, respectively. b) Success rate over different error thresholds on our dataset comparing to DeepPrior [13]. c) Our baseline worst case success rate on the MSRA dataset. Note that we took the state-of-the-art results instantly from [26]. See [26] for details on the methods. d) and e) Our greedy success rate on NYU dataset for worst and average case, respectively.

Table 1: Quantitative results on MSRA dataset. Values are per-joint error in millimeters. Letters *R* and *T* go for finger root and finger tail, respectively. We extracted values from the results reported in the papers. Results for [15] obtained from [3].

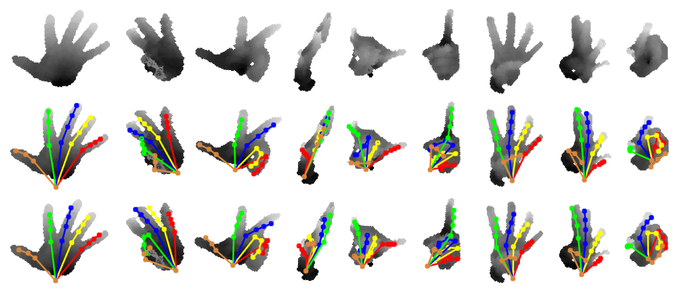
	IndexR	IndexT	MiddleR	MiddleT	RingR	RingT	LittleR	LittleT	ThumbT	Mean
Oikonomidis *et al.* [15]	31.0	56.0	32.9	56.0	32.9	49.3	35.1	53.7	22.2	38.2
Choi *et al.* [3]	22.6	43.5	24.0	44.9	23.1	43.1	21.8	39.5	31.1	29.8
Ge *et al.* [12]	11.5	16.0	9.0	15.6	9.9	15.1	13.2	16.0	16.7	13.0
Ours (KNN+ICP)	9.5	17.3	7.7	17.1	8.3	15.5	10.6	17.7	14.8	12.8

compared our greedy approach with [31], [14] and [24] on this dataset. Quantitative results are shown in Fig. 6. Although, our approach performs worse than [14] for error thresholds larger than 35mm for worst case joints error in Fig. 6(d), it outperforms [14] for average joints error in Fig. 6(e) with a large margin. As we show qualitatively in Fig. 10, one can see that RF estimation is quite noisy in the last column. This shows a lack of data in the training set to cover uncommon cases. This leads to a wrong nearest neighbor extraction which directly effects accuracy, main reason of larger number of worst case errors for higher error thresholds. Average error on this dataset is 14.19mm.

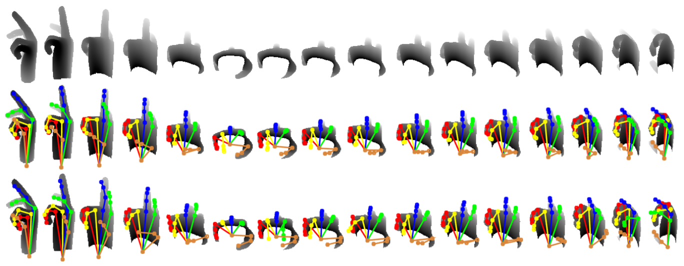
Component analysis on the synthetic dataset. The first step of our approach is nearest shape extraction. RF segmentation responses are used to compute the proposed descriptor, and therefore, we consider the relationship between the accuracy of RF and 1-NN average joints error in Fig. 11(a). As it can be seen from the figure, such dependency is minimal since for example with 10% of RF performance, the proposed method could find the 1-NN with an average error of 23 mm compared to the 1-NN error of 90% performance obtained by RF. On the other hand, KNNs are good enough to generate an accurate hand segmentation as can be observed in Fig. 11(b). Therefore, we can argue that the final hand segmentation performance



(a)



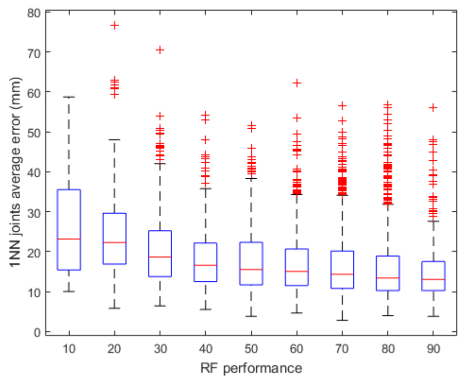
(b)



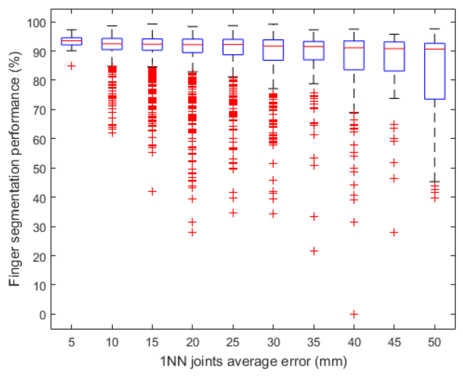
(c)

Figure 7: Qualitative results. a) Comparing different approaches. Rows from top to bottom: depth image, segmentation, baseline, greedy, and DeepPrior [13]. The columns show frames 1, 12, 27, 42, 75, 83, 244, 301 and 352 from left to right. b) Results on MSRA dataset [26]. Rows from top to bottom: depth image, groundtruth, and our baseline estimation. c) Greedy+bilinear optimization in a sequence of frames. Rows from top to bottom: depth image, ground truth, and final results. Results are generated with error lower than 30mm per visible joint for initial step.

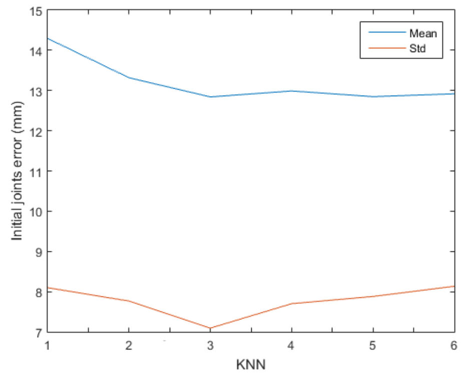
has a low dependency on RF segmentation accuracy. This statement is valid as long as we have a uniform distribu-



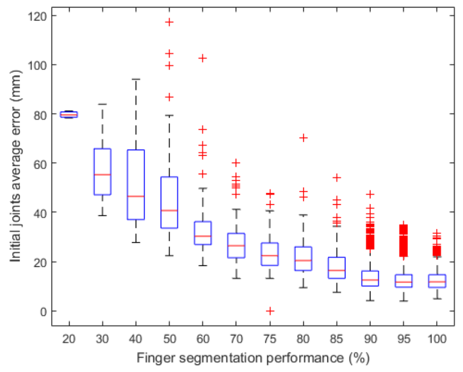
(a)



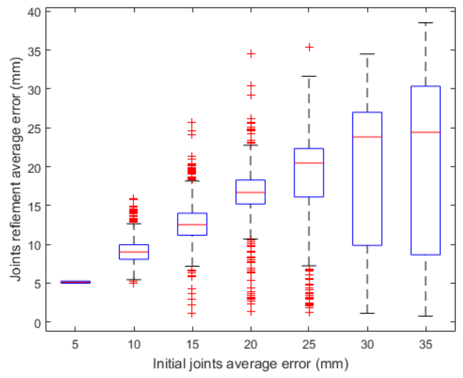
(b)



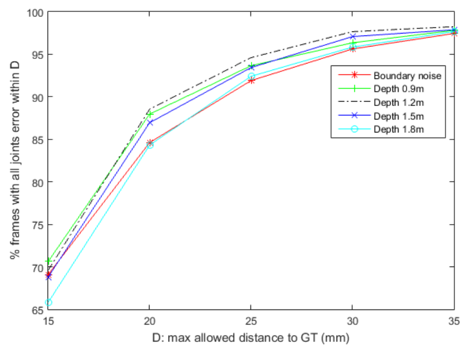
(c)



(d)



(e)



(f)

Figure 11: Quantitative results comparing different components of the methodology. Bar graphs are generated by excluding 5% outliers.

tion of pose and viewpoint in the data.

We illustrate in Fig. 12 how RF performance improved in a number of frames based on our finger segmentation strategy. We added ICP MSE of extracted nearest shapes to Fig. 12 which shows a meaningful relationship between the accuracy of RF and the alignment error among ex-

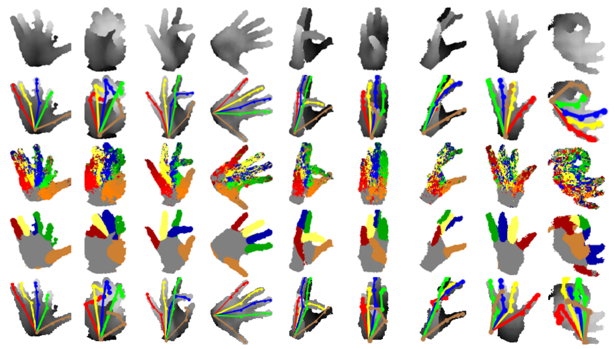


Figure 10: Qualitative images for NYU dataset. Rows from top to bottom: depth image, groundtruth pose, RF segmentation, refined hand segmentation and estimated pose. We show a failure case in the last column where a wrong nearest neighbor effects accuracy.

tracted nearest neighbors. We normalized MSE by a maximum threshold of 200 for the sake of visualization.

The relationship between the number of the nearest neighbors and the final joints error is illustrated in Fig. 11(c). Setting K to 3 leads to more stability regarding the standard deviation and the correlation of the nearest neighbors.

Hand segmentation accuracy is critical for final pose recovery. This relationship is shown in Fig. 11(d). However, one can observe how complex poses (ICP MSE) affects the mean pose error by comparing Fig. 13 to Fig. 12. As an example, we refer to frames 70 to 100. This is because nearest neighbor extraction and hand segmentation mainly depends on the difficulty of the pose or the availability of it within the training data. Therefore data availability in the training step is a key issue for the success of the method.

As we have shown, we could refine occluded joints recovery in the sequence based on initial pose recovery. However, the accuracy of temporal pose recovery completely depends on the accuracy of the initial pose estimation. We considered this case in Fig. 11(e).

We performed additional experiments by modifying test data. For the first experiment, we added artificial boundary noise to depth images. For this task we extracted nested boundaries and added increasing Gaussian noise from the inner boundary to the outer boundary. The maximum noise was 30 mm for the outer boundary. We also considered the effect of hand distance to camera by increasing the distance of the hand point cloud to the cam-

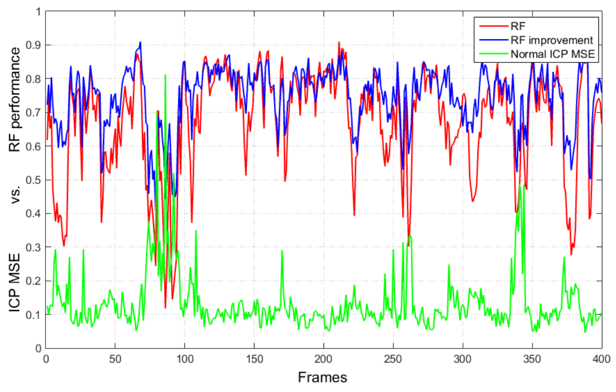


Figure 12: Quantitative RF performance. We improved RF performance by around 20%. We include ICP MSE (green line) as a function of the difficulty of each frame. ICP MSE normalized by a maximum threshold 200. ICP MSE shows how near each frame is to its nearest shapes.

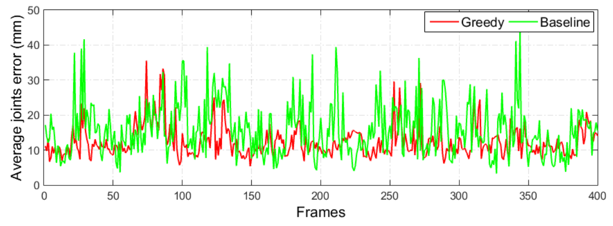


Figure 13: Greedy vs. baseline in a sequence of frames. The frames are the same as Fig. 12.

era and reprojecting it to the image plane. The results are shown in the Fig. 11(f). It can be seen that boundary noise can affect the accuracy more than the hand distance to camera. However, this effect appears to be minimal.

Time complexity. Our methodology has a high parallelization capability at any stage. It is GPU-friendly since fingers estimations are minimized separately.

Greedy finger minimization needs just evaluating function E over the selected candidates. Initial pose estimation is achieved in real time. Most of the processing time is consumed by PSO optimization over bilinear model coefficients C. We use 5 generations over 100 particles which is comparable to 30 and 100 in [21] respectively. We implemented the whole pipeline in Matlab and C++, which although not optimized, runs at 10 fps.

5. Conclusions

We have presented a novel top-down approach for the problem of hand pose recovery in depth images, jointly model-based and data-driven. We have introduced a new and large joint-annotated synthetic dataset with high degree of self-occlusion. We handled self-occlusions by separately extracting each hand component based on our proposed objective function in single frames. Then, we refined occluded joints recovery by including a bilinear model to optimize the parameters in a sequence of images. Evaluation on NYU dataset showed that uniform distribution of data in terms of pose and viewpoint is critical in the accuracy of nearest shape extraction and pose recovery. Given such uniform distribution on synthetic and MSRA datasets, we showed that the method is robust against highly-variable hand poses, while being able to recover occluded joints both efficiently and accurately.

Acknowledgements

This work has been supported by the H2020 project SEE.4C, TIN2015-65464-R and TIN2016-74946-P (MINECO/FEDER, UE) and CERCA Programme / Generalitat de Catalunya.

References

[1] I. Akhter, T. Simon, S. Khan, I. Matthews, and Y. Sheikh. Bilinear spatiotemporal basis models. *TOG*, 31(17), 2012.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24:509–522, 2002.

[3] C. Choi, A. Sinha, J. H. Choi, S. Jang, and K. Ramani. A collaborative filtering approach to real-time hand pose estimation. *ICCV*, 2015.

[4] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 135–142. ACM, 1993.

[5] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The cave: Audio visual experience automatic virtual environment. *Commun. ACM*, 35(6):64–72, June 1992.

[6] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(12):52 – 73, 2007.

[7] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E.-H. Zahzah. Human pose estimation from monocular images: A comprehensive survey. *Sensors*, 16(12), 2016.

[8] J. S. S. III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan. Depth-based hand pose estimation: methods, data, and challenges. *arXiv:1504.06378v1*, 2015.

[9] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[10] C. Keskin, F. Kra, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classication using multi-layered randomized decision forests. *ECCV*, 7577:852–863, 2012.

[11] F. Kondori, S. Yousefi, J.-P. Kouma, L. Liu, and H. Li. Direct hand pose estimation for immersive gestural interaction. *PRL*, 2015.

[12] J. Y. Liuhao Ge, Hui Liang and D. Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. *CVPR*, 2016.

[13] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. *CV Winter Workshop*, 2015.

[14] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3316–3324, 2015.

[15] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efcient model-based 3d tracking of hand articulations using kinect. *BMVC*, pages 101.1–101.11, 2011.

[16] M. Paladini, A. Bartoli, and L. Agapito. Sequential non-rigid structure-from-motion with the 3d-implicit low-rank shape model. *ECCV*, pages 15–28, 2010.

[17] X. Perez-Sala, S. Escalera, C. Angulo, and J. Gonzalez. A survey on model based approaches for 2d and 3d visual human pose recovery. *Sensors 14*, 3:4189–4210, 2014.

[18] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. *CVPR*, 2014.

[19] K. Roditakis, A. Makris, and A. Argyros. Generative 3d hand tracking with spatially constrained pose sampling. *BMVC*, 2017.

[20] G. Rogez, J. S. Supancic, and D. Ramanan. First-person pose recognition using egocentric workspaces, 2015.

[21] T. Sharp, C. Keskin, and e. a. Robertson. Accurate, robust, and flexible real-time hand tracking. In *ACM Human Factors in Computing Systems*, pages 3633–3642. ACM, 2015.

[22] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[23] X. Shu, F. Porikli, and N. Ahuja. Robust orthonormal subspace learning: Efcient recovery of corrupted low-rank matrices. *CVPR*, pages 23–28, 2014.

[24] A. Sinha, C. Choi, and K. Ramani. Deephand: robust hand pose estimation by completing a matrix imputed with deep features. pages 4150–4158, 2016.

[25] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *ICCV*, 2013.

[26] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun. Cascaded hand pose regression. In *CVPR*, 2015.

[27] D. Tang, H. Chang, A. Tejani, and T.-K. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. *CVPR*, 2014.

[28] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. *ICCV*, pages 3224–3231, 2013.

[29] L. Tao and B. J. Matuszewski. 3d deformable shape reconstruction with diffusion maps. *BMVC*, 2013.

[30] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics (TOG)*, 35(6):222, 2016.

[31] J. Tompson, M. Stein, Y. LeCun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *TOG*, 33(5), 2014.

[32] C. Wan, T. Probst, L. Van Gool, and A. Yao. Crossing nets: Dual generative models with a shared latent space for hand pose estimation. *CVPR*, 2017.

[33] L. Xia, C.-C. Chen, and J. Aggarwal. View invariant human action recognition using histograms of 3d joints. *HAU3D*, 2012.

[34] C. Xu and L. Cheng. Efcient hand pose estimation from a single depth image. *ICCV*, pages 3456–3462, 2013.

[35] Q. Ye, S. Yuan, and T.-K. Kim. Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In *European Conference on Computer Vision*, pages 346–361. Springer, 2016.

[36] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 13(2):119–152, 1994.

[37] F. Zhou and F. D. la Torre. Spatio-temporal matching for human detection in video. *ECCV*, 8694:62–77, 2014.

[38] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei. Model-based deep hand pose estimation. *IJCAI*, 2016.