Single Shot 3D Hand Pose Estimation Using Radial Basis Function Networks Trained on Synthetic Data

Vassilios - Clitos Nicodemou

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Antonis Argyros*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

Single Shot 3D Hand Pose Estimation Using Radial Basis Function Networks Trained on Synthetic Data

Thesis submitted by
Vassilios - Clitos Nicodemou
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Vassilios - Clitos Nicodemou

Committee approvals: _____
Antonis Argyros
Professor, Thesis Supervisor

_____
Constantine Stephanidis
Professor, Committee Member

_____
Xenophon Zabulis
Principal Researcher, Committee Member

Departmental approval: _____
Antonis Argyros
Professor, Director of Graduate Studies

Heraklion, February 2018

Ευχαριστίες

Οι εμπειρίες που συγκέντρωσα και οι γνώσεις που απέκτησα κατά τη διάρκεια των μεταπτυχιακών μου σπουδών είναι αμέτρητες. Κυρίαρχο ρόλο όλα αυτά τα χρόνια, κατείχε η στήριξη και συμπαράσταση ανθρώπων που με βοήθησαν και πίστεψαν σε εμένα, πριν ακόμη επιλέξω την πορεία των σπουδών μου.

Θα ήθελα να ευχαριστήσω τον επόπτη μου, καθηγητή Αντώνη Αργυρό, που από τα προπτυχιακά μου χρόνια κατάφερε να μου κινήσει το ενδιαφέρον για τον κλάδο της υπολογιστικής όρασης. Μου έδωσε τα μέσα, την κατάλληλη ώθηση και την δυνατότητα να γνωρίσω και να εκτιμήσω τόσο τον ίδιο τον κλάδο, όσο και την ερευνητική κατεύθυνση. Τον ευχαριστώ για όλες τις γνώσεις που μου μετέφερε και που με έκανε μέλος της ομάδας του. Όντας εξαίρετος καθηγητής αλλά και άνθρωπος, έπαιξε καθοριστικό ρόλο στην επιλογή μου να παραμείνω στον κλάδο και στην ομάδα.

Θα ήθελα να ευχαριστήσω πάρα πολύ τον Ιάσονα Οικονομίδη, για την τεράστια βοήθεια, στήριξη και συμπαράστασή του. Χωρίς αυτόν, η μεταπτυχιακή μου εργασία θα μου φαινόταν ένα ακατόρθωτο επίτευγμα. Η υπομονή του και η εμπειρία του συνέβαλε θετικά και παραδειγματικά σε όλη την φάση των σπουδών μου. Με αποτέλεσμα εκτός από έναν καλό συνεργάτη να αποκτήσω και έναν πολύ καλό φίλο.

Θα ήθελα να αναφέρω επίσης το εργαστήριο Υπολογιστικής Όρασης του Ινστιτούτου Πληροφορικής του ΙΤΕ, καθώς και το Πανεπιστήμιο Κρήτης για τους σημαντικούς πόρους που μου παρείχαν έτσι ώστε να εκπληρώσω τους στόχους μου.

Τέλος θα ήθελα να ευχαριστήσω πολύ τους γονείς μου για την τεράστια στήριξη, υπομονή και πίστη που μου έδειξαν. Τους φίλους μου που ήταν πάντα στο πλευρό μου. Και κυρίως την Αναστασία που μου έδωσε θάρρος και κουράγιο στα εύκολα αλλά και στα δύσκολα που περάσαμε μαζί.

Τα καλύτερα έπονται...

Όλους εσάς, σας ευχαριστώ πολύ!

Single Shot 3D Hand Pose Estimation Using Radial Basis Function Networks Trained on Synthetic Data

Abstract

Human motion tracking and analysis forms an important category of problems in the field of Computer Vision. Within this category, the class of problems that deal with the estimation of the full pose of a human hand are especially interesting. This thesis treats the problem of estimating in real time the full pose of a human hand, using only visual input. Many approaches have been proposed to solve this problem, including applying machine learning techniques. The recent success of deep neural networks for computer vision tasks has resulted in new advancements in this area. Despite the significant effort that has been devoted to the problem of 3D hand pose estimation, no method has succeeded to tackle the problem in its full generality.

Machine learning approaches and, in particular, deep learning ones require large annotated datasets for training. The annotation process in real-world data is human labor intensive and time consuming. Therefore, an automatic way of creating and annotating training data is preferable. The use of synthetic data provides an easy way to obtain large volumes of accurately annotated data. On the negative side of using synthetic data, details of the real data may not be accurately simulated. Existing machine learning techniques are sensitive to the distribution of input data and may fail to generalize to real-world data when trained on synthetic data.

In this thesis we present a novel framework to perform single shot 3D hand pose estimation from depth maps. More specifically, the input is assumed to be a single depth map, depicting a single hand in isolation, that is, not occluded by its surroundings. The depth map is acquired using a depth sensor, and no visual aids (e.g., markers) are used to facilitate the task of localizing the hand or parts of it. The method follows a coarse-to-fine strategy, employing Radial Basis Function Networks (RBFNs) that are trained on a large synthetic dataset.

In order to synthesize the dataset that is used to train the RBFNs, we capture a real-world sequence of a human hand performing a set of diverse hand gestures. We proceed to estimate the hand pose for each frame of the sequence using an offline hand tracking method with high computational budget, achieving accurate estimations. Given the set of all the recovered hand poses, we proceed to select the most diverse of them. We use this representative set, along with a dense sampling of all possible rotations as a seed to generate the synthetic training set.

An initialization RBFN and multiple specialized RBFNs are trained on parts of this large synthetic dataset. There are two classes of specialized RBFNs. One class is appropriately trained to recover the global hand rotation given the hand articulation and the second one to recover the global hand articulation given the hand rotation. Given an input depth map, we use the trained models to recover a hand pose. Towards this end, the initialization RBFN is used to provide a

rough pose estimation. Subsequently, the specialized RBFNs are employed in an iterative refinement scheme in order to improve the initial estimation. This iterative refinement scheme is repeated for a predetermined number of repetitions, after the completion of which the final estimation is retrieved.

The overall computational cost of the proposed approach is dominated by the computation of several RBFNs, yielding in practice a system that achieves close to real-time performance. Furthermore, the proposed method is parallelizable, taking advantage of the inherent data-parallelism of RBFNs. The method requires few real-world data and virtually no manual annotation, and it has few hyper-parameters that are experimentally investigated to identify their optimal values. We perform a quantitative evaluation of our method on a test sequence of our own. Additionally, we present quantitative results on a public dataset that is commonly used to evaluate hand pose estimation and tracking methods. Qualitative results are also presented for both datasets. We show that our approach achieves promising results in all cases. Conclusively, this work shows that the proposed RBFNs-based approach can generalize quite well when learning from synthetic data.

Εκτίμηση της 3Δ Πόζας του Ανθρώπινου Χεριού από Μια Εικόνα Χρησιμοποιώντας Δίκτυα Συναρτήσεων Ακτινικής Βάσης Εκπαιδευμένα σε Συνθετικά Δεδομένα

Περίληψη

Η παρακολούθηση και ανάλυση της ανθρώπινης κίνησης αποτελεί μια σημαντική κατηγορία προβλημάτων στον τομέα της Υπολογιστικής Όρασης. Μέσα σε αυτή την κατηγορία, τα προβλήματα που αφορούν στην εκτίμηση της 3Δ πόζας ενός ανθρώπινου χεριού είναι ιδιαίτερα ενδιαφέροντα. Αυτή η εργασία στοχεύει στην επίλυση του προβλήματος της εκτίμησης της πόζας ενός ανθρώπινου χεριού σε πραγματικό χρόνο, χρησιμοποιώντας μόνο οπτική πληροφορία. Πολλές προσεγγίσεις έχουν προταθεί για την επίλυση αυτού του προβλήματος, μεταξύ των οποίων και η εφαρμογή τεχνικών μηχανικής μάθησης. Η πρόσφατη επιτυχία των βαθέων νευρωνικών δικτύων σε προβλήματα υπολογιστικής όρασης έχει οδηγήσει σε σημαντική πρόοδο σε αυτόν τον τομέα. Ωστόσο, παρά τις έντονες προσπάθειες που έχουν αφιερωθεί στην επίλυση προβλημάτων αυτής της κατηγορίας, καμία μέθοδος δεν έχει καταφέρει να επιλύσει το πρόβλημα στη γενική του μορφή.

Οι τεχνικές μηχανικής μάθησης και ιδιαίτερα αυτές που βασίζονται σε βαθιά νευρωνικά δίκτυα απαιτούν μεγάλα επισημασμένα σύνολα δεδομένων για τη εκπαίδευσή τους. Η επισήμανση σε πραγματικά δεδομένα είναι κοπιαστική και απαιτεί χρόνο και άλλους ανθρώπινους πόρους. Επομένως, προτιμάται ένας αυτόματος τρόπος δημιουργίας και επισημείωσης των δεδομένων εκπαίδευσης. Η χρήση συνθετικών δεδομένων παρέχει έναν εύκολο τρόπο για τη δημιουργία μεγάλου όγκου επισημασμένων δεδομένων υψηλής ακρίβειας. Ένα μειονέκτημα αυτής της προσέγγισης είναι ότι οι λεπτομέρειες των πραγματικών δεδομένων μπορεί να μην προσομοιωθούν με ικανοποιητική ακρίβεια κατά τη δημιουργία των συνθετικών δεδομένων. Οι υπάρχουσες τεχνικές μηχανικής μάθησης είναι ευαίσθητες στην κατανομή των δεδομένων εισόδου και συνεπώς, μπορεί να αποτύχουν να γενικεύσουν σε πραγματικά δεδομένα όταν εκπαιδεύονται σε συνθετικά δεδομένα.

Σε αυτή την εργασία παρουσιάζουμε μια νέα προσέγγιση για την εκτίμηση πόζας χεριού από δεδομένα βάθους από μία μόνο όψη. Πιο συγκεκριμένα, υποθέτουμε ότι η είσοδος είναι ένα μεμονωμένο καρέ δεδομένων βάθους, που απεικονίζει ένα απομονωμένο χέρι, δηλαδή ένα χέρι που δεν επικαλύπτεται από αντικείμενα στο περιβάλλον του. Το καρέ βάθους προσλαμβάνεται από έναν αισθητήρα βάθους και δεν χρησιμοποιούνται οπτικά βοηθήματα για να διευκολυνθεί η εργασία εντοπισμού του χεριού ή τμημάτων του. Η μέθοδος ακολουθεί μια στρατηγική εξειδίκευσης (coarse to fine) χρησιμοποιώντας δίκτυα συναρτήσεων ακτινικής βάσης (Radial Basis Function Networks, RBFNs) που εκπαιδεύονται σε ένα μεγάλο σύνολο συνθετικών δεδομένων.

Η δημιουργία των συνθετικών δεδομένων που απαιτούνται για την εκπαίδευση των δικτύων ξεκινάει καταγράφοντας μια πραγματική ακολουθία ενός ανθρώπινου χεριού που εκτελεί διαφορετικές χειρονομίες. Στη συνέχεια εκτιμάται η πόζα του χεριού

για κάθε καρέ της ακολουθίας χρησιμοποιώντας μια μέθοδο παρακολούθησης χεριών με υψηλό υπολογιστικό φόρτο, επιτυγχάνοντας ακριβείς εκτιμήσεις. Από αυτό το σύνολο όλων των ανακτημένων ποζών, επιλέγουμε αυτές που διαφέρουν περισσότερο μεταξύ τους. Χρησιμοποιούμε αυτό το αντιπροσωπευτικό σύνολο, μαζί με μια πυκνή δειγματοληψία όλων των πιθανών περιστροφών για να δημιουργήσουμε το συνθετικό σύνολο εκπαίδευσης.

Ένα δίκτυο αρχικοποίησης και πολλαπλά εξειδικευμένα δίκτυα εκπαιδεύονται σε τμήματα του συνόλου συνθετικών δεδομένων. Υπάρχουν δύο ειδών εξειδικευμένα δίκτυα. Η μία κατηγορία, περιλαμβάνει δίκτυα που είναι κατάλληλα εκπαιδευμένα για να ανακτήσουν τον προσανατολισμό των χεριών με δεδομένη την άρθρωση των δακτύλων. Η δεύτερη κατηγορία ανακτά την άρθρωση δεδομένου του προσανατολισμού του χεριού. Λαμβάνοντας σαν είσοδο ένα καρέ βάθους, χρησιμοποιούμε τα εκπαιδευμένα μοντέλα για να ανακτήσουμε την πόζα του χεριού. Για το σκοπό αυτό, το δίκτυο αρχικοποίησης χρησιμοποιείται για την εκτίμηση μιας αρχικής πόζας. Στη συνέχεια, τα εξειδικευμένα δίκτυα χρησιμοποιούνται σε ένα επαναληπτικό σχήμα που έχει σκοπό να βελτιώσει την αρχική εκτίμηση. Αυτό το επαναληπτικό σχήμα βελτίωσης εκτελείται για έναν προκαθορισμένο αριθμό επαναλήψεων, μετά την ολοκλήρωση του οποίου ανακτάται η εκτίμηση πόζας.

Το συνολικό υπολογιστικό κόστος της προτεινόμενης προσέγγισης καθορίζεται από τις υπολογιστικές απαιτήσεις ενός μικρού πλήθους δικτύων RBFN, επιτυγχάνοντας επιδόσεις σχεδόν πραγματικού χρόνου. Επιπλέον, η προτεινόμενη μέθοδος επιδέχεται επιτάχυνσης μέσω παραλληλοποίησης, χάρη στον εγγενή παραλληλισμό των δικτύων RBFN. Η μέθοδος απαιτεί λίγα πραγματικά δεδομένα και σχεδόν καθόλου επισημείωση. Επιπλέον, έχει λίγες υπερ-παραμέτρους οι οποίες διερευνούνται πειραματικά για να προσδιοριστούν οι βέλτιστες τιμές τους. Η ποσοτική αξιολόγηση της μεθόδου βασίστηκε σε μια πραγματική ακολουθία για την οποία γνωρίζουμε τις πραγματικές πόζες του χεριού (ground truth). Επιπλέον, παρουσιάζουμε ποσοτικά αποτελέσματα σε ένα κοινά διαθέσιμο σύνολο δεδομένων (public dataset) που χρησιμοποιείται για την αξιολόγηση των μεθόδων εκτίμησης και παρακολούθησης πόζας χεριών. Ποιοτικά αποτελέσματα παρουσιάζονται και για τα δύο σύνολα δεδομένων τα οποία δείχνουν ότι η προσέγγισή μας επιτυγχάνει ικανοποιητικά αποτελέσματα σε όλες τις περιπτώσεις. Συμπερασματικά, η εργασία αυτή δείχνει ότι προτεινόμενη προσέγγιση για εκτίμηση της 3Δ πόζας του χεριού που βασίζεται σε δίκτυα RBFN μπορεί να γενικεύσει αρκετά καλά σε πραγματικά δεδομένα, ενώ έχει εκπαιδευθεί σε συνθετικά δεδομένα.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The human hand is one of the most important means of interaction, communication and manipulation, making it a very important subject of research. It is therefore desirable to build interaction systems that take advantage of the motion of the human hand. The task of estimating the full pose of a human hand observed using visual input comprises a very interesting and useful problem in the field of Computer Vision [13]. From a theoretical point of view, this area of problems is interesting because they are a part of a more general group of problems that try to estimate the pose of arbitrary articulated objects.

The significant importance of solving this problem can be seen by the vast number of applications on which the hand pose estimation is applicable. Effective methods to solve the problem can be used as building blocks in the domain of human-computer interaction (HCI). Applications of such approaches include robotic teleoperation, game control, medical rehabilitation, sign language recognition and more. As the technological evolution pushes the real-world even deeper into the existence of the virtual one, hand pose recovery enables Virtual- and Augmented-Reality scenarios. The number of applications that are in need of such solutions is increasing exponentially as every industry wants to exploit the products of the arising virtual evolution.

This problem is accompanied by many difficulties making it, in its full generality, an unsolved problem. Firstly, the search space of human hand poses is very large because of the high number of degrees of freedom, a fact that still holds even after accounting for correlations of natural hand motion. Furthermore, the human hand is very dexterous, meaning that this search space is not redundant. Another complicating factor is the fact that the hand can move very fast, breaking the temporal continuity assumption and thus making purely tracking approaches prone to failure. Lastly, the uniform, almost texture-less hand appearance leads to visual ambiguities that must be resolved. All these difficulties make the problem challenging, and the ways of resolving it more demanding.

Machine learning methods have been widely used for hand pose estimation [60]. In these methods, a model learns a mapping between observations and the

poses. In order to teach the model, one must acquire a large number of training observations from a dataset. Usually a real-world manual creation and annotation is needed for these datasets, by a human subject. This work though is time and recourse consuming. Hence, a more autonomous way is preferable by creating synthetic datasets that reflect the properties of real-world datasets.

Recent approaches commonly rely on deep artificial neural networks for estimating hand poses [47, 48, 15]. Learning from synthetic data though (such as images/depth maps), can be problematic due to a gap between synthetic and real data distributions [68], leading the deep neural networks (such as a Convolutional Neural Networks CNNs) to learn details only present in synthetic data and failing to generalize well on real data. We propose a different approach that uses an Artificial Neural Network (ANN) compared to CNNs, achieving an appealing result.

In this thesis a method is presented for single-shot hand pose estimation of an isolated (i.e. not interacting with the environment) hand observed using a depth sensor. Only the problem of pose estimation is treated, assuming a bounding box is provided by a hand detector such as the methods proposed in [41, 30]. The use of regressor Radial Basis Function Networks (RBFNs) is proposed for hand pose recovery. A creation of a synthetic dataset from captured sequences is used in order to train the networks.

The proposed approach is split in a training and an 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 [78]. In contrast to that work however, we discard the real-world depth maps and only keep the hand poses. These poses seed the 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 is 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. This rough estimation will be use to initialize an iterative search that uses 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.

The rest of this thesis is organized as follows: an outline of the relevant literature is presented in Chapter 2. A detailed overview of existing methods, techniques and tools that we will use in the development of the proposed method are provided

in Chapter 3. The design and the implementation of the proposed methodology is described in Chapter 4. In Chapter 5 are presented the experiments conducted to tune and evaluate the proposed method, along with the obtained results. Finally, Chapter 6 concludes the thesis with a discussion over the impact of this method and the future work.

Chapter 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 attempted to tackle it [55]. The computational load of the method proposed in that work 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 recent introduction of commodity depth sensors helped renew the interest in the problem [49, 54]. Many of the challenges and methods are presented in [73], where the authors state that pose estimation may appear roughly solved, still it remains a challenging category of problems. More recently, the success of deep learning in computer vision has also given new interest in this problem [48, 15].

A comprehensive overview on the subject, the interest for the problem and the difficulties in solving it, can be found in Erol *et al.*[13]. In this survey work, the authors propose to 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 DoF). Another categorization discriminates between methods that can perform single-shot pose estimation, or Single Frame, and those that perform tracking (termed Model-Based Tracking methods). In an evolution of these categorizations, we can separate all methods into three categories, discriminative methods, generative methods and hybrid 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 [48]. As presented in [44], discriminative methods learn a direct mapping from input to target values, while generative methods fit a model to increase to joint probability of the input and the target values. Hybrid methods use both types of methods to a part, in order to overcome the disadvantages of each other. In the following sections we will describe methods following the generative, discriminative and hybrid category respectively.

2.1 Generative Methods

Generative methods are model-based approaches. This means that, methods of this type make use of a hand model that is used during the estimation process to compute image features. These features are compared to respective features extracted from the observed image. The aforementioned features may be edges, depth information, skin color, or an estimated appearance constructed by the hand model. A quantification of the feature comparison for varying poses of the hand model serves as an objective function to an optimization routine. 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. Therefore most of the methods optimize their estimation based on previous estimations. On the other hand, 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 of configuration space and the development of fast and reliable tracking algorithms to estimate the hand posture.

The first works of generative methods appear from 1994 and after [55, 21]. In both works the authors constructed a primitive model of a hand in order to achieve tracking, coping with the multiple degrees of freedom (DoF) of the hand. Some methods use wearable devices or gloves to aid tracking like in [83, 61, 28]. The most recent means of feature acquisition is through depth images [86, 64, 27], especially after the recent growth of the depth sensor capabilities. Other methods like in [4] and [37] use variants of particle filtering to track predetermined hand models, or shape invariant hand models [38] from depth information.

Not all cases use depth information to track or fit a model of the hand. In [10] the authors make use of temporal texture continuity and shading information in order to track from a monocular video. In [2] the authors provide a 3D estimation from cluttered images. Other works try to retrieve depth information from RGB images through stereo matching like in [90, 51], where the acquired depth information may be treated like in depth images' scenarios.

For optimizing the objective functions defined by the methods, it was postulated that the use of particle swarm optimization (PSO) [1, 12] was beneficial towards estimation [49, 50]. The authors in [49] used PSO to minimize the discrepancy between the 3D structure of hypothesized instances of a hand model and actual hand observations. In [54], the authors propose a generative method combining the PSO and Iterative Closet Point (ICP) algorithms to speed-up the search of the hand pose space. Other works that use ICP are [14, 74] which employ the articulated variant of this technique.

A very interesting and challenging group of problems copes with hands interacting with other objects. In [57] the authors emphasize the difficulties of hand interacting with objects. The authors in [20] use a local tracker for each segment

of the hand to overcome the challenge of occlusions between the hand and the object. In [50] the authors enforce physical constrains to the model interacting with the object to improve the tracked estimations. In [52] the authors track hands in interaction with unknown objects, creating the model of the objects during the tracking procedure. Again, depth information is not always necessary as in [3] the authors make use of salient points, like finger tips, by taking into account edges, optical flow and collisions.

2.2 Discriminative Methods

Discriminative methods estimate hand configurations directly from 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 holds the input to the method, usually an image, and the target values that correspond to the input, the hand configuration represented on the input. 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, one must acquire large training dataset to cover the large hand space, thus it will take usually a large amount of time to train properly the whole method. On the positive side, discriminative methods are in general fast at runtime, 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, however in order to cover the full set of possible hand poses, a very large dataset must be employed. The research areas of discriminative methods include the selection of appropriate training algorithms, the suitable learning techniques used by these algorithms, the creation and annotation of large datasets and the effort 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. In [83] the authors use a glove to track a 3D hand and employ nearest neighbor approach to achieve tracking at interactive rates. The use of random decision forest is also widely used with some variations, as seen in [31, 82, 75], for regressing to a 3D hand estimation. In [9] the authors used Relevance Vector Machine (RVM) [77] based learning method to estimate hand pose from multiple cameras.

Since Convolutional Neural Networks (CNN) architectures started being used widely, many works have included these types of techniques and variations of them from RGB images [40] and depth images [8, 17, 70, 22]. In [42] the authors casted

the 3D hand estimation problem from depth images into a voxel-to-voxel prediction with the use of a 3D voxelized grid. The authors in [78] combined a random forest classifier for image segmentation and a CNN to regress to a hand pose, whereas in [15] the authors create 2D heat-maps from depth images to perform the pose estimation. Some methods employ multiple CNNs, as in [36] where the authors use a tree-like structured multiple CNNs to regress to a final hand pose. In [48] a CNN architecture is used combined with a prior on the 3D pose to improve the accuracy of the model. Later the authors improved their architecture by using the best methods for training the networks [45]. The best architectural approaches and training strategies for CNNs where examined in [19].

Similar to the generative methods, for the discriminative methods the task of estimating hand poses that interact with objects is very interesting and challenging. In [58] the authors used a nearest neighbor based approach. The use of CNN is also significant for hand interacting with object for detecting and estimating the pose of both [43], even for the cases when the object is unknown [63].

Another very interesting category of approaches are the ones that use synthetic data for training the discriminative models. The creation of synthetic data is automated making the procedure of annotation instant. Creating such synthetic datasets is presented in [46]. Some works rely on synthetic data training in order to classify similarities between synthetic and real data [67, 56]. Others, like in [26] train random decision forest and support vector machines to regress on a 3D hand poses trained on synthetic data. The most challenging methods train CNNs architectures on synthetic data [69, 11, 16] by augmenting the used training set, partially addressing the fact that CNNs are problematic on generalizing when trained on synthetic data. Moreover, the authors in [92] used a deep network learning technique to learn from synthetic RGB images and regress a 3D hand pose estimation.

2.3 Hybrid Methods

Hybrid methods try to exploit the advantages of both previously mentioned methods, generative and discriminative, trying to avoid their respective disadvantages. Researchers try to achieve this by employing a technique that uses components of generative and discriminative methods. Research focuses on what methods to use from generative and discriminative aspects, as on what algorithms and optimization procedures to follow.

Some methods acquire small discriminative aspects [71, 34, 84] of fixing estimations that are returned by a model-based method for added robustness, while others try to leverage the advantages of both like in [29] where the authors use random decision forest to correct instances of optimization failure. In [76] the authors employ a discriminative objective function and a joint optimization of the model used and the observed data. The authors in [91] use a model based deep learning approach by adopting forward kinematics based layer that ensures the

geometric validity of the estimations. In [79] the authors combine a generative model with discriminatively trained salient points and with collision detection and physics simulation to increase accuracy. The authors in [53] use a regressor to learn and deliver multiple initial hypotheses, then a 3D model is fitted by deliberately exploiting the inherent uncertainty of the proposed joints. Using CNNs in [87] a hierarchical PSO is integrated into a CNN framework forcing constrains to the output results. Again using deep architectures, the authors in [81] propose a method using two deep generative models. To further improve the generalization of their model they train a generator that synthesizes depth maps and a discriminator that benefits from a augmented training set synthetic samples.

2.4 Proposed Work Categorization

The proposed work is closely related to [59] and [78]. In [59], Romero *et al.*present a system to recover the hand pose from monocular RGB input using Histogram of Oriented Gradients features. Similarly to our approach, Romero *et al.*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. In [78], Tompson *et al.*propose the use of an early generative RGBD-based approach [49] 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 [78], 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 [46]. A deep convolutional architecture is presented in [48] 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. The iterative refinement of specialized networks that we use in this work has a close resemblance.

Chapter 3

Preliminaries

This chapter provides a detailed overview of the methods and techniques that will be used in this thesis. These include previous work in related areas that is essential for the methodology presented in Sec. 4. More specifically, this chapter provides a description of the type of the artificial neural network (Radial Basis Function Network), the hand model and the algorithms used subsequently.

3.1 Radial Basis Function Network (RBFN)

A Radial Basis Function Network (RBFN) is a particular type of artificial neural network, introduced firstly by Broomhead *et al.* [5], that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the input and neuron parameters.

RBFNs are commonly used in various of classification and regression problems. Function approximation is a frequently tackled task for RBFNs as in [85, 24, 89], also other regression problems can extended to cases that deal with images as well [7, 88]. Classification problems that use RBFNs can also regard images as in [23, 32].

3.1.1 Network Architecture

Typically RBFNs have three layers: an input layer, a hidden layer with non-linear Radial Basis activation function and a linear output layer.

The input layer can be modeled as a vector of real numbers $\mathbf{x} \in \mathbb{R}^n$. The output of the network is then a scalar function of the input vector, $\varphi : \mathbb{R}^n \to \mathbb{R}$, and is given by

$$\varphi(\mathbf{x}) = \sum_{i=1}^{N} w_i \rho(\|\mathbf{x} - \mathbf{c}_i\|) \tag{3.1}$$

also called *activation value*, where N is the number of neurons in the hidden layer, $\mathbf{c}_i$ is the center vector for neuron i, and w_i is the weight of neuron i in the linear output neuron.

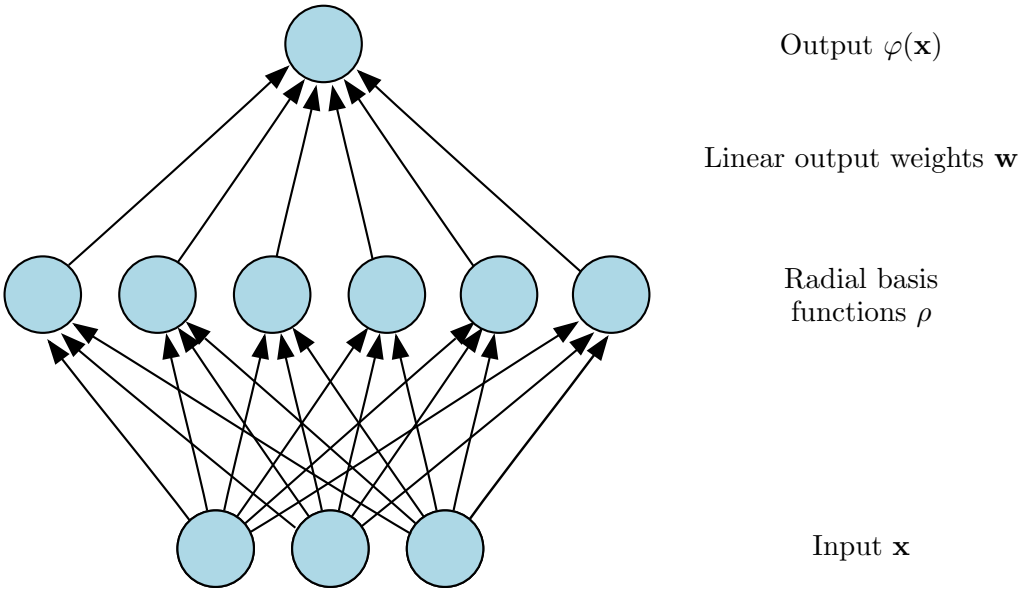


Figure 3.1: Architecture of a radial basis function network. An input vector $\mathbf{x}$ is provided to all radial basis functions. Each radial basis function computes a different response according to its parameters. The output of the network is a linear combination of the outputs from the radial basis functions.

In this work we use a normalized architecture of RBFNs. The mapping in this architecture is

$$\varphi(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i \rho(\|\mathbf{x} - \mathbf{c}_i\|)}{\sum_{i=1}^{N} \rho(\|\mathbf{x} - \mathbf{c}_i\|)} \tag{3.2}$$

where the denominator $\rho(\|\mathbf{x} - \mathbf{c}_i\|)$ is used to normalize the response of each center of the network for every center c_i where $i = 1, \ldots, N$. Normalized RBFNs relax the localized characteristics of standard RBF networks and exhibit excellent generalization properties, as explained in [6].

Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name radial basis function. In the basic form of RBFNs, all inputs are connected to each hidden neuron. The distance from the center vector is typically taken to be the Euclidean norm and the radial basis function is commonly taken to be Gaussian

$$\rho(\|\mathbf{x} - \mathbf{c}_i\|) = \exp[-\beta \|\mathbf{x} - \mathbf{c}_i\|^2] \tag{3.3}$$

where $\mathbf{x}$ is the input and $\mathbf{c}_i$ is again the center vector for neuron i. The β parameter is defined as

$$\beta = \frac{1}{2\sigma^2} \tag{3.4}$$

where σ is the standard deviation of the Gaussian.

During learning, the goal is to determine the tunable parameters w_i, c_i and σ in a manner that optimizes the fit between the output ϕ of the network and the data.

3.1.2 Training the Network

The training process of an RBFN consists of: choosing the hyper-parameters, centers $\mathbf{c}$ and the standard deviation σ of the RBFN neurons, and learning the output weights $\mathbf{w}$ between the RBFN neurons and the output nodes. The work of Schwenker *et al.* [65] provides an overview of common approaches on training RBFNs.

RBF networks are trained from pairs of input and target values $\mathbf{x}_t$,y_t for $t = 1, \ldots, |T|$, indexing a training set T. The vector $\mathbf{x}_t \in \mathbb{R}^n$ is the input as already mentioned, and $y_t \in \mathbb{R}$ is the annotated target output corresponding to the input $\mathbf{x}_t$. Training is achieved by a two-step algorithm. In the first step, the center vectors $\mathbf{c}_i$ of the RBFN in the hidden layer are chosen. The second step fits a linear model with coefficients w_i to the hidden layer's outputs with respect to some objective function. A common objective function, at least for regression, is the least squares function:

$$K(w) = \sum_{t=1}^{|T|} K_t(w) \tag{3.5}$$

where,

$$K_t(w) = [y_t - \varphi(\mathbf{x}_t, w)]^2 \tag{3.6}$$

There are many possible approaches for selecting the centers of the network also studied in [18, 80]. In this work we will mainly focus on two alternatives for center initialization. The first approach is to select k centers from the training set by performing a k-Means clustering. The second approach is to create a center for every training sample. This increases the computation time but also increases the accuracy of the network.

For selecting the first option, that is, using k-Means clustering technique for the centers selection, we may set the parameter σ to be the average distance between all points in a cluster from the cluster's centroid

$$\sigma_j = \frac{1}{|T_j|} \sum_{i=1}^{|T_j|} \|\mathbf{x}_{j,i} - \mathbf{c}_j\| \tag{3.7}$$

where $\mathbf{c}_j$ is the centroid of cluster j, $|T_j|$ is the number of training samples belonging to the cluster j and $\mathbf{x}_{j,i}$ is the ith training sample in the cluster j. As a second option, we can chose one center for every single training sample. In this case we may define the parameter σ as the mean value of distances between all training

samples of the training set T.

$$\sigma = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{3.8}$$

In both cases, the value of σ is not necessarily the most optimal, that is, the one that achieves the best performance of the network. Therefore, an experimental evaluation can be performed in order to determine the optimal value of hyper-parameter σ.

The final set of tunable parameters to be determined are the output weights $\mathbf{w}$. These can be trained using gradient descent. First, for every data point in our training set, we compute the activation values of the RBFN neurons. These activation values become the training inputs to gradient descent. We also add a fixed value of 1s as a bias term to the beginning of the vector of activation values. Gradient descent must be run separately for each output node.

Another possible training technique, and the one used in this work, is applying the linear pseudoinverse solution. This solution arises by formulating the problem as a least squares minimization between the responses of the radial basis function and the desired outputs after fixing the centers $\mathbf{c}_i$:

$$minimize \|\theta^T \mathbf{w} - \mathbf{b}\|. \tag{3.9}$$

The weights that minimize the error at the output are computed using the Moore-Penrose inverse solution

$$\mathbf{w} = \theta^+ \mathbf{b} \tag{3.10}$$

where θ^+ is the pseudoinverse of θ the values of which are the radial basis functions evaluated at the points $\mathbf{x}_t$

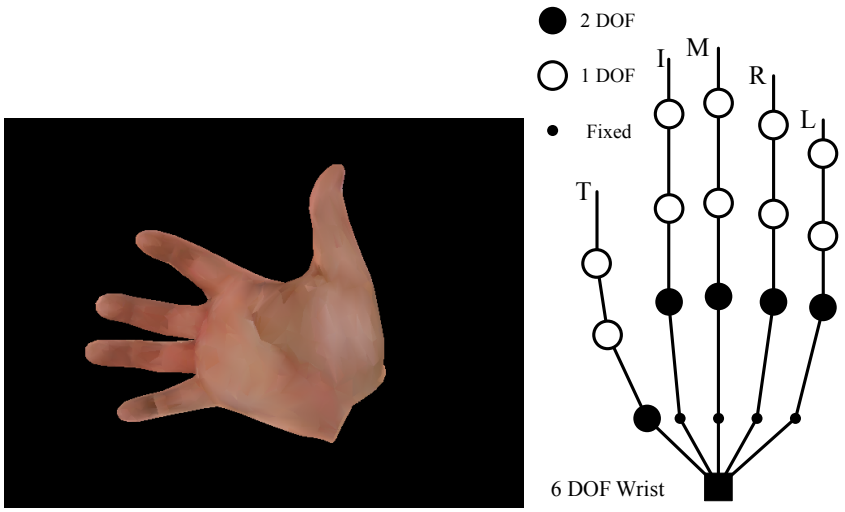
$$\theta_{ti} = \rho(\|\mathbf{x}_t - \mathbf{c}_i\|) \tag{3.11}$$

and $\mathbf{b}$ is a vector with the target values y_t for $t = 1, \ldots, |T|$. The existence of this linear solution means that unlike multilayer perceptron (MLP) networks [62], RBFNs have a unique local minimum when the centers are fixed.

In contrast to the definitions above, it is common to have a training set that maps input vectors $\mathbf{x}_t$ to output vectors $\mathbf{y}_t \in \mathbb{R}^m$, instead of scalars y_t. This is easy to accommodate by training m separate RBF networks, each targeting a scalar of the vector $\mathbf{y}$, and concatenating their output to regress the vector $\mathbf{y}$. In practice, these RBFNs share a large part of the computations and so it is beneficial to reuse them. Thus, this concatenation of RBFNs can be considered as a single RBFN that regresses to a vector $\mathbf{y} \in \mathbb{R}^m$.

Having computed the centers $\mathbf{c}$, the parameter β and the entries of θ from a trained RBFN, we can evaluate any given test input. We denote them as parameters of a trained RBFN model.

The implementation of the RBFN architecture found in [39] is used and modified for the purposes of this work. The RBFN is used in this work to estimate hand poses using regression. The input to the RBFN is a depth map image and the target value is the pose of the depicted hand.



(a) (b)

Figure 3.2: The skinned hand model used in this work: (a) the rendered skinned hand; (b) the 20 joints of the model with their corresponding degrees of freedom (DOF);

3.2 Hand Model

In this work we use a 27-parameter skinned hand model for representation as seen in Fig. 3.2a. Similar to Oikonomidis *et al.* [49], the hand model kinematics are defined by 20 joints that have up to two degrees of freedom, as shown in Fig. 3.2b.

A hand pose $\mathbf{p}$ is a 27-parameter vector defined as

$$\mathbf{p} = (x, y, z, q_x, q_y, q_z, q_w, \phi_1, \phi_2, \ldots, \phi_{20}) \tag{3.12}$$

The first 3 values $\mathbf{p}_{xyz} = (x, y, z)$ define the global 3D position of the hand model. The next 4 values $\mathbf{p}_q = (q_x, q_y, q_z, q_w)$ define a normalized quaternion that determines the global rotation of the hand model in the 3D space with respect to the center of the model. The remaining 20 parameters $\mathbf{p}_\phi = (\phi_1, \ldots, \phi_{20})$ define the angle of each joint of the hand, fully determining the articulation of the hand.

The 27 parameters are also used to render the hand model. The rendering is used to obtain the depth map of the hand model given a hand pose $\mathbf{p}$. We denote the rendering as a function $Ren(\mathbf{p})$ that is implemented in a similar way as in [49]. In contrast to [49] where the hand shape is constructed as a set of appropriately transformed cylinders and spheres, in our work we use a linear-blend skinned hand model.

Given a hand pose $\mathbf{p}$ we set some key points on the joints and the fingertips of the hand, then using the model's forward kinematics with the 3D positions of the key points, we obtain the model's landmarks L. These landmarks are 3D positions that can be used for comparing two hand poses, by computing distances between

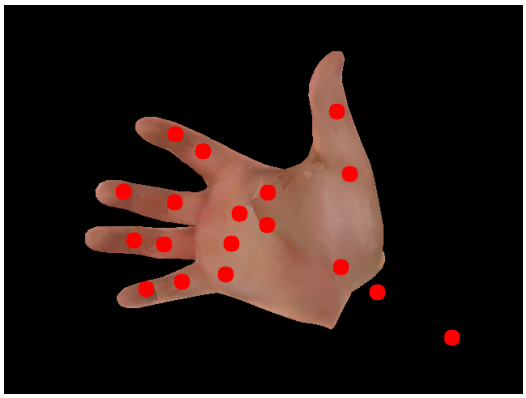


Figure 3.3: The skinned hand model used in this work with the annotated landmarks, visualized as red discs.

corresponding landmarks of compared hand poses. In Fig. 3.3 the aforementioned landmarks are visualized overlayed the rendered hand model.

3.3 KKZ Algorithm

A new initialization technique was introduced by Katsavounidis *et al.* [25] for Generalized Lloyd iteration in [33], which exploited the fact that vectors that are most far apart from each other based on a distance metric, are more likely to belong to different classes.

Given a sequence of vectors $\mathbf{v}_i$, with $i = 1, ..., N$, we obtain a sorted vector array $\mathbf{s}$ as follows:

1. Calculate the norms of all vectors in the sequence. Choose the vector with the maximum norm as the first item of your sorted array $\mathbf{s}$.

2. Calculate the distance of all vectors $\mathbf{v}_k$ from the first vector of the sorted array $\mathbf{s}$, and choose the vector $\mathbf{v}_k$ with the largest distance as the second item of the sorted array. After this step, we have a sorted array $\mathbf{s}$ of size 2.

3. Generally, with a sorted array $\mathbf{s}$ of size i, $i = 2, 3, ...$, we compute the distance between any remaining vector $\mathbf{v}_k$ and all existing vectors of the sorted array and call the smallest value the distance between $\mathbf{v}_k$ and the sorted array $\mathbf{s}$. Then, the vector with the largest distance from the sorted array $\mathbf{s}$ is chosen to be the $(i + 1)$th item of $\mathbf{s}$. The procedure stops when we obtain an array of size N.

When the procedure finishes, we end up with a sorted array of vectors with the property that each element of the array is the furthest from the elements above it.

Given a large set of hand poses, we would like to be able to identify an arbitrary number of the most diverse ones. Doing this offers us a more sparse and diverse

set holding representative of the original set. The KKZ algorithm is used to sort hand poses with respect to their articulations and rotations. This offers the most diverse poses to be located at the beginning of the sorted array of poses.

Chapter 4

Method Description

This chapter describes in detail the design and the implementation of the proposed method. Our goal is to recover the global position, orientation and full articulation of a human hand observed by a depth sensor. More specifically, given a single depth map of the observed hand, we aim to recover the pose parameters listed above, without using any information provided by previous depth maps from a sequence. Towards this end, we parameterize the pose space of the human hand as a 27-dimensional vector. This vector defines a pose for the employed hand model (Sec. 3.2). The task is therefore reduced to one of parameter estimation, where the parameters of interest are the full hand pose vector. To achieve this we train Radial Basis Function Networks (RBFN) (Sec. 3.1) to regress directly from an input depth map to this pose vector.

We start with a real-world captured dataset to acquire a large number of articulations. By sampling densely the rotation space, we combine them with the acquired articulations to create a synthetic dataset. We use the synthetic dataset to train an initial RBFN and multiple specialized RBFNs. For an input depth map we use the initial RBFN to retrieve a rough approximation. Then, the specialized RBFNs try to refine iteratively this approximation.

The method is divided in two phases, the training phase and the estimation phase. The training phase (Sec. 4.1) consists of preparing the training and training the initialization RBFN and specialized RBFNs. The estimation phase (Sec. 4.2) uses the parameters learned from training in order to estimate the hand pose parameters given a single depth frame. Figure 4.1 illustrates the pipeline of the system during the training phase and provides an example during the estimation phase.

4.1 Training Phase

In order to proceed with the training of the system, some preparation steps are necessary. More specifically, the training dataset has to be adjusted and augmented so it can be used as input in the training process. We use two sets $\mathcal{A}$ and $\mathcal{R}$ that

hold all the articulations and all the rotations respectively of a training dataset. This dataset is synthetically generated using the tracking of a real hand performing multiple gestures (this procedure is outlined in Sec. 5.1.2). For specific articulations and orientations we have to train separately specialized RBFNs using a subset of the initial training dataset, plus one extra RBFN that we are going to use as an initialization step of the Iterative Refinement Algorithm (Sec. 4.2.1). The input needed for training each RBFN is pairs of rendered depth maps of hand poses $Ren(\mathbf{p})$ and their respective hand pose $\mathbf{p}$ which is the desired output of the network.

4.1.1 Training Set Preparation

Our method uses a synthetically generated training set, as described in Sec. 5.1.2, in order to train the networks. Having prior knowledge of the ground-truth hand poses $\mathbf{p}$ during the training phase, we extract all the articulations that are defined by the set $\mathcal{A}$. Also the $\mathcal{R}$ set is composed from densely sampling the quaternion space. The hand poses were obtained by tracking a real world sequence that captures diverse hand poses. Training our system in all the available data is not preferable, due to the fact that some of the samples are repeated (not necessarily identical, but similar poses) and the increase of the training time. Towards this end, we employ KKZ Algorithm (Sec. 3.3) to select a subset of diverse hand poses and hand articulations. More specifically using the KKZ Algorithm we sort all articulations in $\mathcal{A}$ and orientations in $\mathcal{R}$ and select the first $|A|$ articulations and the first $|R|$ orientations to create the new sets A and R that are going to be used for training, where $|A| \leq |\mathcal{A}|$ and $|R| \leq |\mathcal{R}|$ are preselected sizes of sets $A \subseteq \mathcal{A}$ and $R \subseteq \mathcal{R}$. The KKZ Algorithm rearranges the sets in such order that any ith element, from the articulation set A for instance, $i = 1, 2, \ldots, |A|$, of the sorted set is the farthest from the set of elements 1 to $i-1$. Thus we achieve a dense, evenly distributed training set that contains representatives of all the articulations in the captured sequence and of all rotations.

During the sorting procedure of KKZ, a distance metric between the compared vectors is required. 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:

$$DQ(\mathbf{q_1}, \mathbf{q_2}) = 1 - (\mathbf{q_1} \cdot \mathbf{q_2}) \tag{4.1}$$

where q_1 and q_2 are the two compared quaternions. To sort the set A of articulation, the employed distance metric is more complex. More specifically, we calculate the distance between the landmarks (Sec. 3.2) that are defined by the articulations in A. To determine the landmarks of a hand pose, we also need the orientation and the global position of the hand. Given the fact that we are only interested on the articulation of the hand, we fix the rotation and the position of the compared hand poses, so that the comparison will not affect our distances. The distance

function between two articulations a_1 and a_2 is defined as

$$D(a_1, a_2) = \frac{1}{n} \sum_{l=1}^{n} ||L_{1_l} - L_{2_l}||_2 \tag{4.2}$$

where L_1 and L_2 are the landmarks obtained by the articulations a_1 and a_2 respectively for the fixed rotation and n is the number of landmarks defined by the hand model.

The RBFNs must also regress directly from the depth map to the global position p_x, p_y, p_z. Learning the global position of each training sample is not desirable because it is independent of any given instance of depth map input. We could use the 3D position of the center of the bounding box $\mathbf{b}$ that encloses the hand model to denote 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 the 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.

For creating the input data used for training our networks, we render a pose $\mathbf{p}$ using the rendering function $Ren(\mathbf{p})$. Therefore $Ren(\mathbf{p})$ is a depth image of the given pose $\mathbf{p}$.As seen in Sec. 4.1.3, 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 we employ comparison between them. In order to compare depth maps it is necessary to perform depth normalization. Each training depth map is normalized by subtracting the median of the non-zero 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.

4.1.2 Training Initialization RBFN

An RBFN must be trained that is going to be used as an initialization step of the Iterative Refinement Algorithm (Sec. 4.2.1). We select the z first articulations and rotations from our sets A and R that will be used to train the initialization RBFN, due to their diversity by exploiting the property of the KKZ ordering. We create the z^2 training poses by combining the articulations and orientations of this training set. Having the pairs of each depth maps $Ren(\mathbf{p})$ with their respective hand poses $\mathbf{p}$, corresponding inputs $\mathbf{x}$ and targets $\mathbf{y}$ in Sec. 3.1.2, we train the network and store the three network's parameters $C_{init}, \beta_{init}, \theta_{init}$.

The selected articulations and rotations are limited to the number z which is smaller than the $|A|$ and $|R|$. Apart from practical limitations such as memory capacities, 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.

4.1.3 Training Specialized RBFNs

Having a single RBFN trained on every possible articulation and orientation is not feasible, thus the specialized networks ease the refinement by learning for a fixed articulation all rotations 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.

4.1.3.1 Articulation-Specialized RBFNs

Every $RBFNa_i$ is trained separately, where $i = 1, \ldots, |A|$. 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

$$\begin{bmatrix} Ren(\mathbf{p}_{i,1}) \\ \vdots \\ Ren(\mathbf{p}_{i,j}) \\ \vdots \\ Ren(\mathbf{p}_{i,|R|}) \end{bmatrix}, \begin{bmatrix} \mathbf{p}_{i,1} \\ \vdots \\ \mathbf{p}_{i,j} \\ \vdots \\ \mathbf{p}_{i,|R|}) \end{bmatrix} \tag{4.3}$$

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.

4.1.3.2 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 with the pairs of training input and their corresponding targets

$$\begin{bmatrix} Ren(\mathbf{p}_{1,j}) \\ \vdots \\ Ren(\mathbf{p}_{i,j}) \\ \vdots \\ Ren(\mathbf{p}_{|A|,j}) \end{bmatrix}, \begin{bmatrix} \mathbf{p}_{1,j} \\ \vdots \\ \mathbf{p}_{i,j} \\ \vdots \\ \mathbf{p}_{|A|,j}) \end{bmatrix} \tag{4.4}$$

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

4.2 Estimation Phase

The focus of this work is the estimation of the hand pose having as input a crop of the hand within 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. We follow an iterative refinement scheme, starting from a pose estimation that is regressed from the initialization RBFN. To refine this estimation, we proceed to alternatively refine the orientation and the articulation part of the pose by finding the specialized network that can refine the proposed solution by using its specialized knowledge.

4.2.1 Iterative Refinement Algorithm

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 iteratively it finds an articulation from set A to use the according RBFN that was trained on that articulation to return a new estimation. This estimation is used to find a rotation from set R to use the corresponding rotation specialized RBFN for a new estimation, returning to the start of the iteration.

More specifically, to evaluate a single test depth image t we start by finding a rough approximation $\tilde{\mathbf{p}} = RBFN_{init}(t; C_{init}, \beta_{init}, \theta_{init})$ using our initialization RBFN. Subsequently, for a number of epochs we find the closest articulation a_i, from set A that was used for training, to the articulation $\tilde{a}$ of the estimated $\tilde{\mathbf{p}}$. We input the test image t to the $RBFNa_i$ that was trained upon this closest articulation to obtain a new estimation $\tilde{\mathbf{p}} = RBFNa_i(t; Ca_i, \beta a_i, \theta a_i)$. 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 $\tilde{\mathbf{p}} = RBFNr_j(t; Cr_j, \beta r_j, \theta r_j)$. We repeat these steps as mentioned, starting from the beginning of the iteration. Iteratively we lookup 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 describes the Iterative Refinement Algorithm described above.

The function $FindClosestArticulation$ 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

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 $\tilde{\mathbf{p}}$.

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

Eq. 4.2. 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 as explained in Sec. 3.2, 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. 4.1.

Since at each iteration we search the space of our training data to find the closest articulations and rotations, the runtime 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 runtime of the estimation phase.

The orientation of our hand model in Sec. 3.2 is defined as a normalized quaternion. Therefore we normalize the rotation quaternion of our estimation $\tilde{\mathbf{p}}$ in order to proceed with the comparison with other hand poses, or with the rendering of the hand's image. The normalization of a quaternion vector q is performed as:

$$q_{norm} = \frac{q}{|q|}. \tag{4.5}$$

For a description of the methods pipeline, we can refer again to the Fig. 4.1. 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.

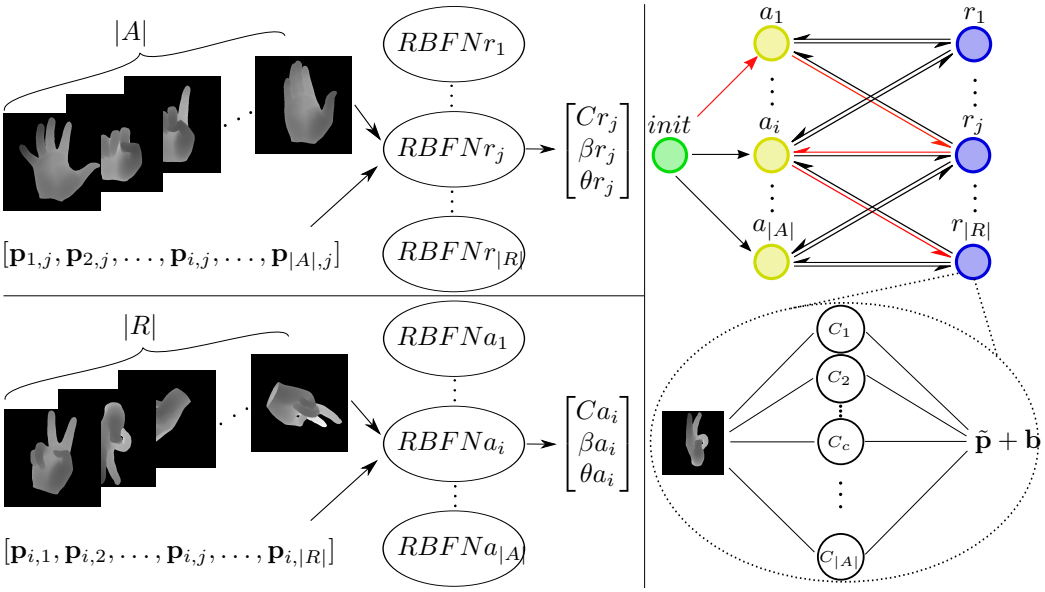


Figure 4.1: (Top-Left) Training $|R|$ rotation-specialized RBFNs, by fixing the rotation r_j each time and learning the pairs $(Ren(\mathbf{p}_{i,j}), \mathbf{p}_{i,j})$ for $i = 1, \ldots, |A|$. (Bottom-Left) Similarly, training $|A|$ articulation-specialized RBFNs, by fixing the articulation a_i each time and learning the pairs $(Ren(\mathbf{p}_{i,j}), \mathbf{p}_{i,j})$ for $j = 1, \ldots, |R|$. (Top-Right) Sample execution of the IRA for 2 epochs. Firstly, $RBFN_{init}$ estimates an approximation $\tilde{\mathbf{p}}$. Epoch 1: We find the articulation a_1 to be the closest articulation from the set A to the approximated articulation $\tilde{a}$. $RBFNa_1$ estimates a pose $\tilde{\mathbf{p}}$. We find the rotation r_j as the closest rotation from the set R to the estimated rotation $\tilde{r}$. Then, $RBFNr_j$ estimates a pose $\tilde{\mathbf{p}}$. Epoch 2: We repeat the previous actions, finding this time the articulation a_i and rotation $r_{|R|}$ as the closest. Therefore $RBFNr_{|R|}$ estimates the final hand pose $\tilde{\mathbf{p}}$. (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 $\mathbf{b}$.

Chapter 5

Experimental Evaluation

This chapter provides the evaluation procedure for this method, giving a description of the datasets used during all phases and throughout the whole procedure of evaluating the method. Also, details of the method's configuration and parameterization are presented with quantitative feedback. Last but not least,the results of the proposed method are provided in both quantitative and qualitative forms.

Firstly, we describe the procedure we followed to generate the synthetic data that we used to train the RBFNs. Specifically, two sequences were captured for creating a training and testing dataset respectively. In particular, the first sequence is needed to construct a synthetic dataset that is entirely used in the training phase, while the second sequence is exclusively used for testing. For testing our method we also used the publicly available MSRA hands dataset in [72].

Each dataset used by our method has to be preprocessed before entering the pipeline. This is done to make all input uniform with regard to the image size, image pixel values and the hand positioning. This preprocessing allows us to treat all input images in a uniform manner.

We define a metric for comparing hand poses in order to evaluate our method. This metric is also used for the hyper-parameter tuning, to determine the best parameterization of our model.

In order to tune the hyper-parameters we use a subset of the synthetic training set to find the best parameters for our method. The tuning is performed to optimize the method with respect to the number of centers, the value of the standard deviation σ used in the RBFNs (Sec. 3.1.2) and the number of epochs used by IRA in Sec. 4.2.1. The chosen parameters are assessed using the respective pose estimation errors of the models, created using these parameters. This procedure allows us to select the hyper-parameter values that result to the best pose estimation error.

Finally, we present the results of evaluating the proposed method. The evaluation is performed on two datasets, the second captured sequence and the MSRA hands dataset. We evaluate on each dataset two equally trained models. The first one is the model of the proposed method as describes in Sec. 4, and the second one

is only the initial network $RBFN_{init}$, without the use of the iterative refinement of the specialized RBFNs. For both datasets and models we present quantitative and qualitative results showing the performance of our method.

5.1 Datasets

In order to employ the training phase, a dataset must be used to train the implemented networks. This dataset was created synthetically using a prerecorded sequence. The evaluation of the method is performed on different datasets. In this work we provide experimental evaluation on two datasets, one created from a captured sequence, and the other being a publicly available dataset, the MSRA hand dataset [72]. All datasets are preprocessed before being used as input for both training and evaluation phases.

5.1.1 Captured Sequences

For creating the captured dataset used in this work, we recorded two sequences of human hand poses. Overall, we recorded the color and the depth information (RGBD) of a human hand performing a large variety of finger articulations on a limited range of hand pose rotations. The hand actions that are performed in the second sequence are identical to the actions of the first sequence. They were recorded in this manner with the prospect of using the first sequence as a training information, and the second sequence as a test.

The first sequence is about 2 minutes long, and the second sequence is about 1 minute and 30 seconds long. Specifically, the first sequence contains 3180 frames of color and depth maps individually, and the second sequence contains 2710 frames of color and depth information. Sample frames of the captured sequences with their depth maps are illustrated in Fig. 5.1. All images are of size 640×480, captured from a RGBD depth sensor. The cropped images shown in Fig. 5.1 are only for visualization purposes, and not the actual input to the system.

We employ the implementation of the method presented in [49] to track the sequences with a large budget, yielding accurate estimation of each of the hand poses, which is a 27-parameter vector $\mathbf{p}$ for each frame. The manual initialization required by this method is the only manual annotation used in the proposed pipeline. The tracked information returns accurately hand poses that are used as target values $\mathbf{y}$ for training the networks. Also, these hand poses are used as ground truths for evaluating estimations of depth maps.

The 20 last elements, $\phi_1, \ldots, \phi_{20}$, of every hand pose $\mathbf{p}$ from the first sequence, define the articulation set $\mathcal{A}$. These pairs of depth maps and hand poses of the first sequence are used in the creation of the synthetic dataset.

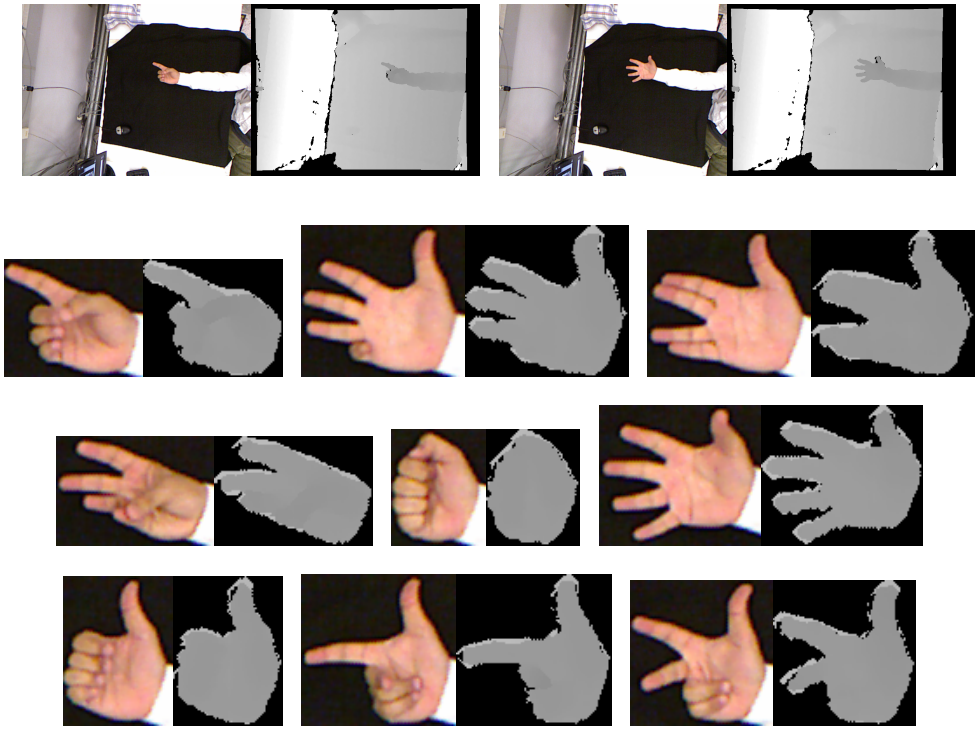


Figure 5.1: Samples of the captured dataset: First row original 640×480 RGB images with their corresponding depth images, second to fourth row cropped RGB images of hands with their corresponding depth images.

5.1.2 Synthetic Dataset

A synthetic dataset is created that is used during the training phase in Sec. 4.1.1 for the creation of the training set. For the creation of the synthetic dataset we use the 3180 hand poses that were tracked from the first captured sequence.

As mentioned, the first captured sequence has a limited range of hand pose rotations. This makes the dataset less diverse as it does not cover a large space of hand poses. This is not desirable because we want to train the RBFNs so that they generalize to any given depth map of a hand pose. Capturing a large sequence of hand poses with more rotations is also not desirable, because it can become a time consuming and tedious process. Thus, we need to create a more dense augmented training set that covers a large space of hand poses in an autonomous fashion.

To augment the training set, we sample densely the quaternion space computing a set of unique and diverse quaternion vectors. Towards this end, we follow an approach that largely resembles the KKZ approach. Specifically, we create a huge set of random normalized quaternion vectors $\mathcal{Q}$ that represent the quaternion space. Then we start by randomly sampling without replacement one quaternion from this space and add it to a new set $\mathcal{R}$. Next, we find for all quaternions in $\mathcal{Q}$

Algorithm 2 Generating rotation set $\mathcal{R}$

Input: A set of random normalized quaternion vectors $\mathcal{Q}$.
Output: The rotation set $\mathcal{R}$.

$\mathcal{R} \leftarrow \mathbf{q}$;
// where $\mathbf{q}$ is a randomly sampled quaternion from $\mathcal{Q}$
for $j = 1 \dots \#quaternions$ **do**
  // where $\#quaternions$ is the number of quaternions we want to insert in $\mathcal{R}$
  $\mathbf{q} \leftarrow \arg\max_{\mathbf{q}}(min(DQ(\mathcal{Q}, \mathcal{R})))$
  //DQ returns the distance matrix between all quaternions from set $\mathcal{Q}$ to $\mathcal{R}$
  $\mathcal{Q} \leftarrow \mathcal{Q} - \{\mathbf{q}\}$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{q}\}$
end for
return ($\mathcal{R}$);

the closest quaternions to the ones in $\mathcal{R}$ by calculating the distances between the set $\mathcal{Q}$ and $\mathcal{R}$. The quaternion with the maximum shortest distance from $\mathcal{Q}$ will be added to $\mathcal{R}$. We repeat the above steps for a desired number of rotations. The distance between two quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$ is defined in Eq. 4.1:

$$DQ(\mathbf{q_1}, \mathbf{q_2}) = 1 - (\mathbf{q_1} * \mathbf{q_2}).$$

Algorithm 2 describes the generation of set $\mathcal{R}$.

The idea of creating the set of rotations $\mathcal{R}$ is similar to the described KKZ algorithm in Sec. 3.3. In this work we set the number of desired quaternions to be equal to 1024. Selecting a higher number would benefit towards the increased density of the dataset, but we are confined to the memory space as we will explain below.

Having the two sets $\mathcal{A}$ and $\mathcal{R}$ we can proceed with the creation of the synthetic dataset. We create hand pose vectors $\mathbf{p}$ by taking all possible combinations of articulations from $\mathcal{A}$ and rotations from $\mathcal{R}$. That means that $\mathbf{p}_\phi = a, \forall a \in \mathcal{A}$ and $\mathbf{p}_q = r, \forall r \in \mathcal{R}$. The global positions of the hand poses are set to $p_x = 0$, $p_y = 0$ and $p_z = 1000$. Setting those numbers to any other value will affect negatively the generation of depth images below. In conclusion we will end up with $3180 \times 1024 = 3256320$ different hand poses for our dataset.

Having the generated poses of our dataset, we need to generate the depth maps that correspond to the hand poses. From the implementation of [49] we also use the module that synthesizes candidate depth maps of our skinned articulated hand model. We refer to the functionality of this module as *Ren* in Sec. 4. By applying the rendering function *Ren* to every hand pose $\mathbf{p}$ of the synthetic dataset we generate the corresponding 3256320 depth maps $Ren(\mathbf{p})$.

Overall, after this procedure we have in our possession the input depth images $\mathbf{x} = Ren(\mathbf{p})$ and their target values $\mathbf{y} = \mathbf{p}$ required by the training procedure.

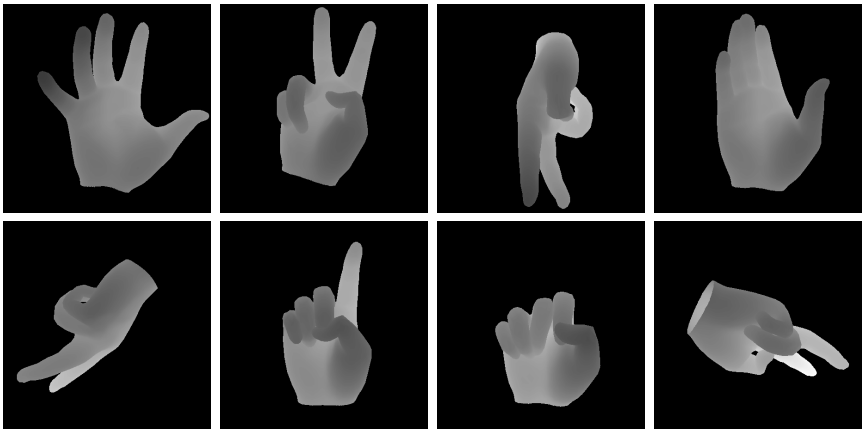


Figure 5.2: Samples of the synthetic dataset.

For the purposes of this work we select a synthetic dataset to be the one that is used for training, as annotating real-world data is time and recourse consuming compared to synthetic data creation.

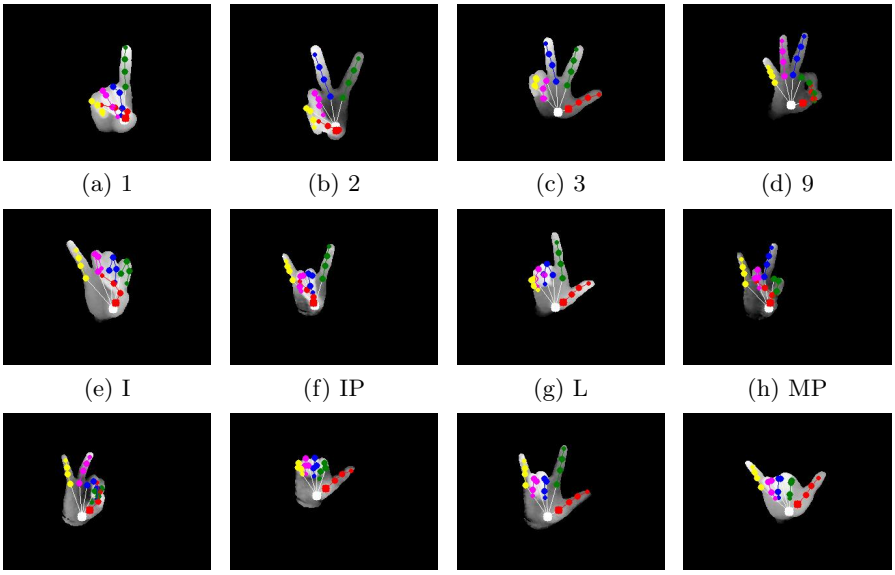
Samples of the synthetic dataset are presented in Fig. 5.2. The images of the poses presented are cropped, only for visualization purposes.

5.1.3 MSRA Hands Dataset

A publicly available dataset was also used for evaluating the performance of this work. The publicly available dataset, MSRA hands dataset, presented in [72] is a large dataset of hand poses. The dataset consists of 17 different hand gestures performed from 9 different subjects. Each gesture is recorded for about 500 frames, for a total of 76500 frames. The provided depth maps are annotated with 3D joint locations. In particular, there are 21 annotated 3D joints for each depth map. We manually mapped the joints of this dataset to the corresponding landmarks of our hand model in order to compare the estimated hand poses with the annotated hand poses of this dataset.

The hand shapes and sizes differ from subject to subject making the dataset diverse. The performed gestures are mostly chosen from the American Sign Language. Specifically, the gestures consist of numbers 1 to 9 and 8 more letters from the American Sign Language. Each gesture is repeated for many rotations spanning as much as possible the finger articulation and hand rotation space. This dataset was used to evaluate the proposed method for all the varying articulations rotations and hand shapes captured by this dataset.

Samples of depth images with their ground truth joint annotations are shown in Fig. 5.3.



(a) 1 (b) 2 (c) 3 (d) 9

(e) I (f) IP (g) L (h) MP

(i) RP (j) T (k) TIP (l) Y

Figure 5.3: Samples of depth images of MSRA dataset with the annotated joints for some of the numbers (Fig. 5.3a-5.3d) and letters of the American Sign Language (Fig. 5.3e-5.3l).

5.1.4 Input Preprocessing

Every dataset has its differences regarding the input depth maps. The differences include, the variety of image sizes, the depth values where the hands were captured, the background depths and the models of the hands used. We pass the input through a preprocessing step to achieve a fair comparison of the inputted data of the network. All depth maps are preprocessed from every dataset.

For every depth image, used for training or testing, we apply the following procedure. We mask the region of the depth map where the hand is located. This region contains all the projected depth pixels that are part of the hand from the wrist to the fingertips. All the pixels that are not located in the masked region have their value set to 0. We find the bounding box that contains the masked hand. Using this bounding box we crop the depth image. The cropped image is padded with zeros along its smallest dimension in order to make it a square image with equal height and width. The padded zeros are inserted symmetrically on the two opposite sides of the smallest dimension so that the hand will be located in the center of the image. Each depth map is normalized by subtracting the median value of the non-zero values and adding to them a fixed depth value of 1000. Finally we resize the image to 64 by 64 pixels using Nearest-neighbor interpolation.

After applying this procedure to a test image, we store the subtracted median so that we can add it later in order to estimate the actual position of the hand.

This procedure is used to make the comparison between depth maps possible, as RBFNs compare centers, that in our case are the preprocessed depth images.

5.2 Metric Definition

To evaluate the performance of the method we define a metric for comparing two hand poses. 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

$$D(\mathbf{p_1}, \mathbf{p_2}) = \frac{1}{n} \sum_{l=1}^{n} ||L_{1_l} - L_{2_l}||_2 \tag{5.1}$$

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. It is easy to see that, since this distance is defined as the average of point distances, its unit of measurement is the same, in our case millimeters (mm). As mentioned before, we use a similar distance function (Eq. 4.2) in Sec. 4.1.1 for measuring distances between articulations. In contrast to Eq. 4.2, Eq. 5.1 does not imply anything to be fixed on the hand poses, giving a complete comparison between two distinct hand poses.

We will refer to this function as the distance error function. We use this function to evaluate and measure the performance of individual parts of our method, such as specialized RBFNs or the initial RBFN, as also the whole method itself. This function is also used to tune the hyper-parameters, optimizing the performance of our method.

5.3 Hyper-parameter Tuning

As stated before, our method uses three hyper-parameters that must be chosen beforehand. These are the centers of the RBFNs, the parameter σ used by the radial basis functions and the epochs used during the evaluation phase of the method. To select the best parameters we must investigate possible parameter initializations and validate our method trying to find the best parameters.

For each investigation we train our method on a subset of the training set, and validate on a separate validation set to find the ones that increase the accuracy of the method.

5.3.1 Centers Selection

The alternative procedures for the selection of centers in a RBFN differ not only on the desired number that specifies the quantity of centers in the network, but also on the way we select these centers. Specifically, variations exist on the way a center can be selected and formed.

In our case, centers correspond to vectors with 4096 elements derived from the input images (64×64 image size). As described in Sec. 3.1.2 we try two different approaches for center selection. The first approach is to select k centers from the training set by performing a k-Means clustering, while the second approach is to create a center for every training sample.

For selecting the centers we use a smaller part of the synthetic training set as created in Sec. 4.1.1 to train the networks and a validation set taken from the same synthetic dataset. Specifically, the 40 first articulations and 40 first rotations from the sets A and R respectively are chosen to train the networks. We train 40 articulation specialized RBFNs, 40 rotation specialized RBFNs and the $RBFN_{init}$ for $z = 40$, forming a total of 1600 training samples. The validation set contains 1000 random samples taken from the same synthetic dataset, different from the ones selected for training.

5.3.1.1 Mini-batch k-Means Center Clustering

To achieve clustering of the centers we use a variation of the k-Means algorithm [35]. The Mini-batch k-Means algorithm is introduced in [66]. This variation of the k-Means algorithm reduces computation cost by orders of magnitude compared to the classic algorithm while achieving comparable results to the regular k-Means algorithm. This is achieved by using mini-batch stochastic gradient descent to solve the optimization problem of finding the nearest cluster of the data.

We train our networks with the training set described for validation. As σ we assign the value 4016, and estimate the final poses for 6 epochs. This value does not affect our final validation, as long as we train all networks equally with the two hyper-parameters fixed. During the training of each RBFN, including the initial one, we perform the Mini-batch k-Means algorithm for various numbers of k. The number k denotes the number of centers we want to select. We repeat the training procedure of all our networks, specialized and initial, for 12 different numbers of centers. Specifically, we train our method for 128, 256, 384, 512, 640, 768, 896, 1024, 1152, 1280, 1408 and 1536 centers. We evaluate each trained model on the previously described validation set and present the results in Fig. 5.4.

The graph in Fig. 5.4 shows the percentage of hand poses of the validation set having average distance error less than the threshold in the horizontal axis. We observe that as we increase the number of centers we have a significant gain in the performance of our method. Even though for 1536 centers, that is the highest number of centers of this graph, the minimum distance error is not the lowest. Still, for higher percentage of the validation set we obtain lower distance errors. Overall, it is clear to assume that as the number of centers increases and gets closer to the number of the training samples, which is 1600, the performance of our method improves.

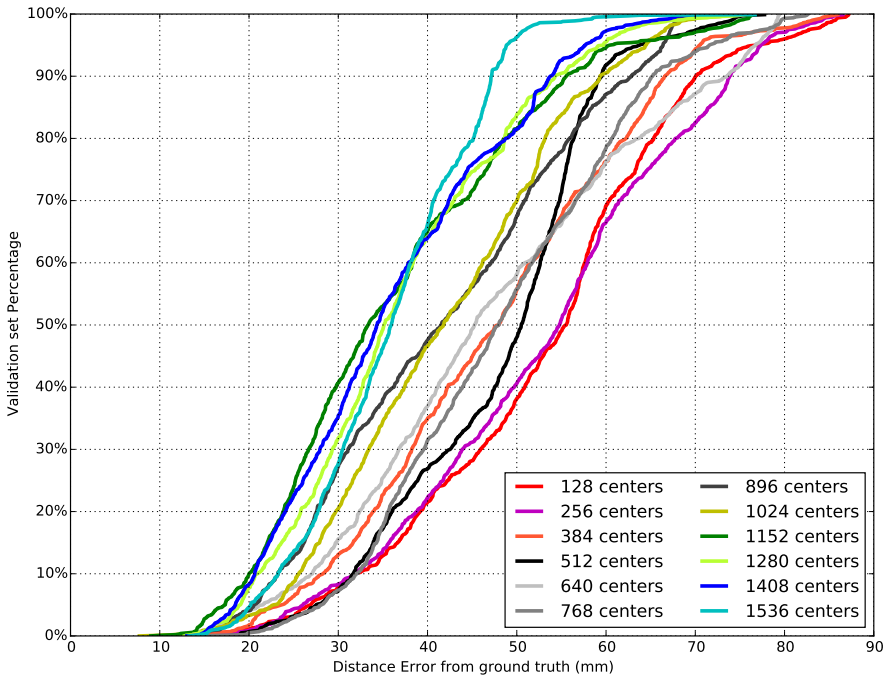


Figure 5.4: Validating RBFNs for different number of centers.

5.3.1.2 Training Set Center Assignment

Following the intuition of the previous experiment, we discard the clustering method and train a new model using as centers each training sample separately, that is 1600 centers. Again the other two hyper-parameters are fixed with values $\sigma = 4016$ and 6 epochs. Figure 5.5 shows the results of the previous experiment, plus the new trained model that uses as centers all the training samples without clustering.

As expected the new model has the best accuracy overall, proving the assumption of the previous experiment, that the performance of the model increases when the number of centers reach the number of samples used for training. Intuitively this means that the best performance is achieved when memorizing the training dataset, interpolating between its samples.

5.3.2 Selection of σ

The value of parameter σ can be initialized as defined in Eq. 3.8:

$$\sigma = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i\|.$$

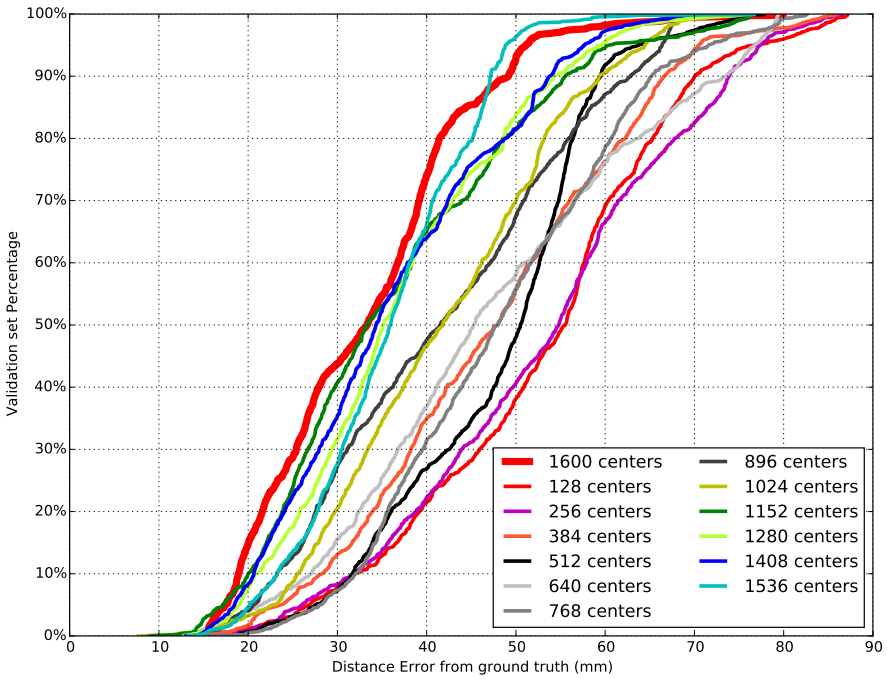


Figure 5.5: Validating RBFNs for different number of centers plus a model trained on all training samples.

As stated though, this may not be the optimal value, thus we need to perform a validation to determine the best value for the parameter.

To validate for the best value of the parameter σ, we use the $RBFN_{init}$ network trained for $z = 100$ on the 100 first articulations and 100 first rotations of sets A and R respectively from the training set in Sec. 4.1.1, having a total of 10000 training samples. And as validation set we use the same 1000 random samples used for the center selection that are different from the training samples. We select as centers of the trained models, to be the training samples. Since we use only the initial RBFN, we do not have to specify the number of epochs.

We train multiple individual $RBFN_{init}$ models for different values of σ. Specifically, we range the value of σ for 11 different values: 1600, 2536, 4019, 6370, 10095, 16000, 25358, 40190, 63697, 100953 and 160000. These values are created using the mean of our training data, which in our case is 1600. We select those values around the mean that are formed as

$$\sigma = \lceil 1600 * 10^{s} \rceil \tag{5.2}$$

where s takes the values: 0 to 2 with step 0.2.

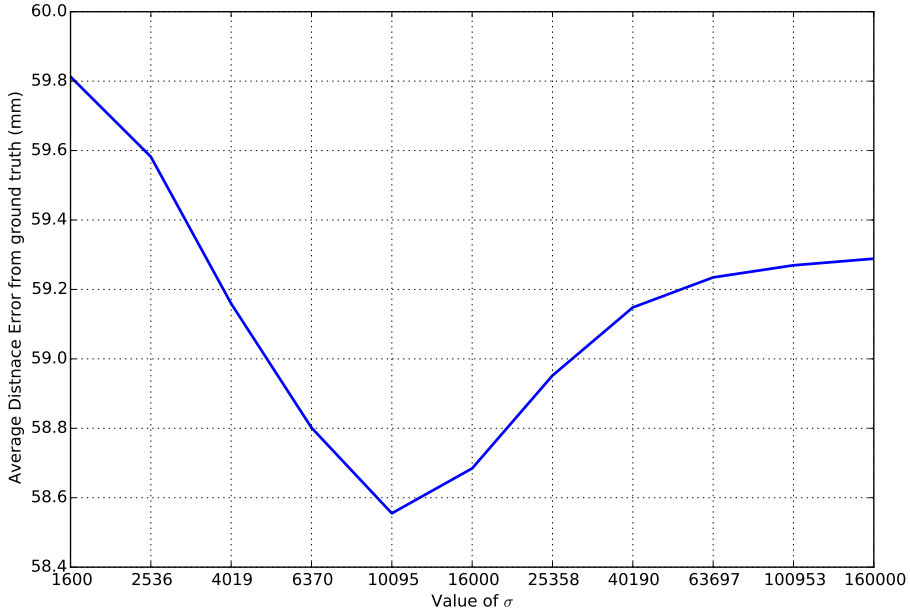


Figure 5.6: Validating RBFNs for different number of σ values.

We assess the performance of those 11 networks for the same validation set of 1000 random samples used for the center selection. The results are presented in Fig. 5.6.

We observe that even though the most optimal value for σ is 10095, the average distance error decreased only by $1.2mm$ from the the highest average distance error. Although the reduction is negligible, we prefer $\sigma = 10095$ for the slightest increase of accuracy.

5.3.3 Number of Epochs Selection

Finally, to select the number of epochs we use the obtained method after trained on the same training set of 1600 samples that we use in center selection and with the same validation set of 1000 test samples. We train again 40 articulation and 40 rotation specialized RBFNs as also the $RBFN_{init}$ for $z = 40$ using the 1600 training samples. The networks are trained with the training samples selected as centers and parameter $\sigma = 10095$.

We perform the estimation phase on our validation set using 2, 4, 6, 8 and 10 epochs and report the average distance error for each evaluation. The results are presented in Fig. 5.7.

We can observe that for 8 number of epochs we achieve the lowest average distance error, and beyond 8 epochs the accuracy appears to degrade and the error

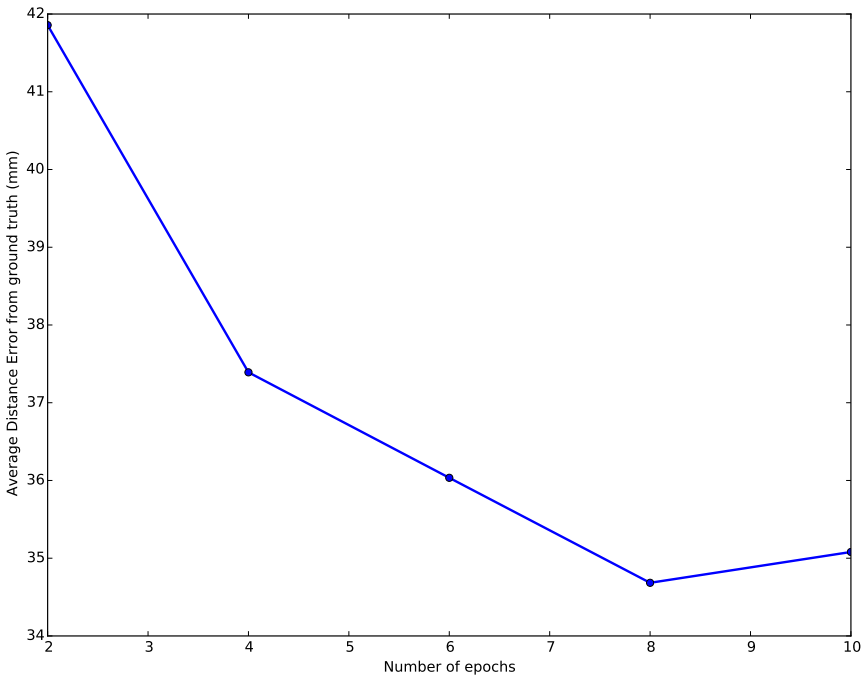


Figure 5.7: Validating RBFNs for different number of epochs.

to increase. While there is an important difference of the average error between 2 and 8 epochs, we notice that from 6 to 8 epochs the distance error improves hardly by $1mm$. Also, for 6 epochs it took about 1 second on average to conclude on an estimation, while for 8 epochs the method needed 1.25 seconds on average. This might not be noticeable on our validation experiment, but for larger training sets this might affect our evaluation time (see Sec. 4.2.1). Having said the above, we prefer 6 number of epochs over 8 since the increase in computational time is disproportional to the accuracy gain.

5.4 Results

We conducted four experiments to evaluate the performance of our method. The first experiment evaluated the proposed method on our own captured test sequence. The second experiment evaluated the proposed method on the publicly available MSRA hands dataset. The third and fourth experiments evaluated only the initial network $RBFN_{init}$ on the captured test sequence and the MSRA dataset respectively, to experimentally verify the validity of the iterative refinement scheme. All the experiments ran on a computer equipped with Intel Core i7-4790 CPU at 3.60GHz $\times$ 8 and 16GB of RAM.

For conducting the first two experiments we started by training the entire pipeline of the proposed method on the synthetically created training dataset using the previously identified hyper-parameters. Specifically, as described in Sec. 4.1.1 we used $|A| = 1024$ articulations to form the set A and $|R| = 1024$ rotations to form the set R that combined constitute our training set. Thus, we trained 1024 articulation specialized RBFNs and 1024 rotation specialized RBFNs. For training the initial RBFN we selected $z = 100$ to form a total of 10000 training samples to train the $RBFN_{init}$. We set for all RBFNs the centers as the respective training samples of each network. All networks are trained with $\sigma = 10095$. The estimation phase iterates for 6 epochs.

For the third and fourth experiments we used only the one initial network $RBFN_{init}$ with $z = 100$ to perform hand pose estimation. Therefore, combinations of 100 articulations and 100 rotations were used, forming a total of 10000 training samples. Again we set as centers of the trained network the samples used for training. The training is performed for $\sigma = 10095$. Since we do not use any refinement in this model, no number of epochs needs to be specified.

The sizes of the sets A and R as well as the value z that corresponds to the training size of the $RBFN_{init}$, were selected purely for practical reasons as larger sizes are limited by our computer's memory. Because we do not favor articulations over rotations and vice versa, we select both sizes $|A|$ and $|R|$ to be equal. Note that if the density of the test articulation or rotation space is known beforehand then by adjusting these parameters we can easily emphasize more towards one or the other. More generally, the proposed method can easily incorporate constraints on the rotation or articulation space.

5.4.1 Quantitative Results

For the first experiment we used our own real-world sequence to perform the evaluation on the proposed method. The sequence is composed out of 2710 frames containing depth and color information as well as the bounding boxes that contain the hands. Alongside we had the accurately tracked hand poses that we used as ground truth in order to compare them with the estimated ones.

Using the testing sequence we measure the distance error of the estimated poses and plot the number of samples that correspond to each distance. In Fig. 5.8 we present the histogram with 250 bins for the distance errors for these testing samples.

The proposed method achieves an average distance error of $44.37mm$ on the captured test sequence and with standard deviation of $17.74mm$. Even though the average error is greater than $2cm$ we can observe that many of the tested samples have distance error close to $2cm$.

In the second experiment we evaluated our method on the MSRA dataset. The dataset is composed out of 76500 frames from 9 subjects performing 17 gestures. We are provided with depth information and the bounding boxes containing the hand for each frame. Again we are given the ground truth of the poses for

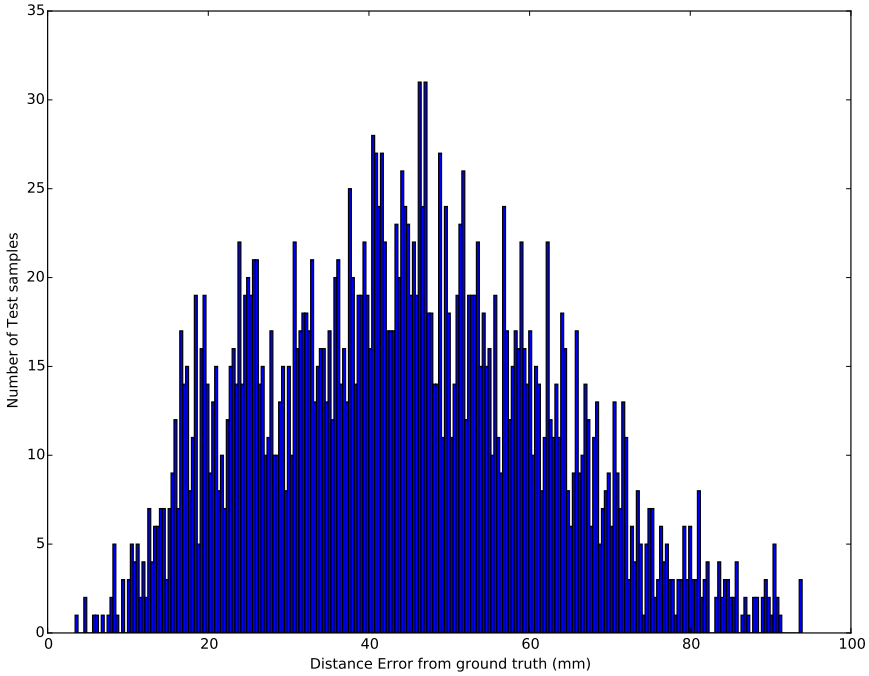


Figure 5.8: Histogram of distance errors for the estimations on the captured sequence of the proposed method.

computing the distance errors.

We measure the error of the poses that our method estimated for the MSRA dataset, plotting the number of samples that have a certain distance. The resulting histogram is presented in Fig. 5.9 with 250 bins.

On the MSRA dataset, our method achieves an average distance error of $49.60mm$ and standard deviation of $12.60mm$. Even though the standard deviation is quite low, we have some testing samples that reach distance errors of $430mm$. Still, more than 99% of our samples have distance error less than $100mm$.

For the third experiment we used the trained model that uses only the $RBFN_{init}$ for estimation. In this experiment we evaluated this model on our captured test sequence. In Fig. 5.10 we present the histogram that shows the number of test samples having a certain distance error, for 250 bins.

Having an average distance error of $61.13mm$ and standard deviation $6.46mm$ we can observe that the iterative refinement that we impose in the previous experiments, improves the performance of our method. The deviation of the distances may increase with the use of IRA, in the contrast the accuracy of the method is almost doubled.

For our last experiment we used again the model of $RBFN_{init}$ to evaluate the

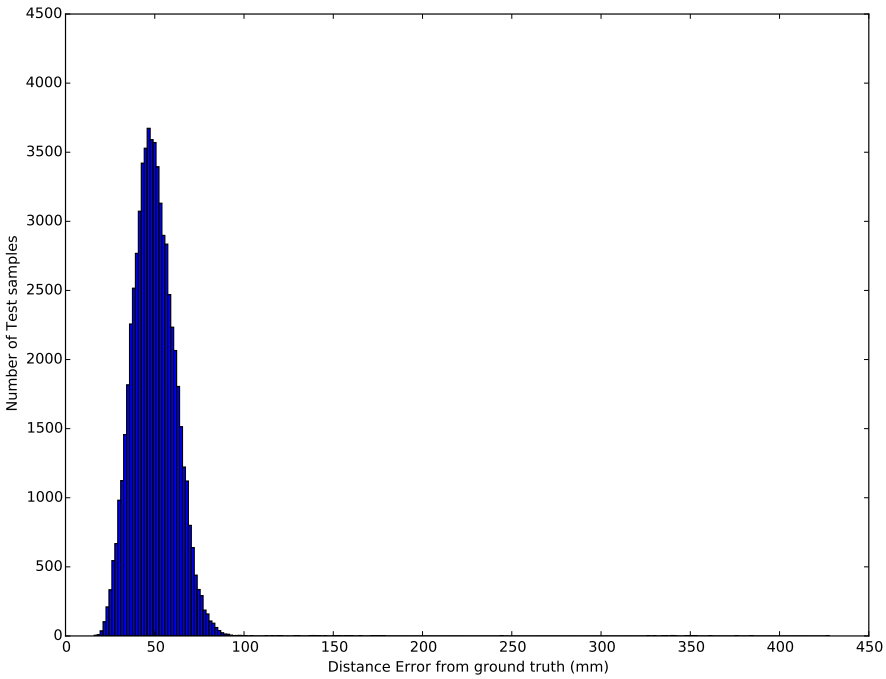


Figure 5.9: Histogram of distance errors for the estimations on the MSRA dataset of the proposed method.

poses of MSRA dataset. Figure 5.11 shows the histogram of distance errors for 250 bins.

With average distance of $61.45mm$ and standard deviation of $10.02mm$ we can see that the absence of IRA degrades the accuracy of our model. Again on the same dataset we observe a few large distances reaching $370mm$, but as previously more than 99% of our test samples have distance error less than $100mm$.

To ease the comparison between the use and the absence of the iterative refinement scheme in our method, we present two plots that show the percentage of poses, on the vertical axis, for which the distance error is below a certain threshold, on the horizontal axis, for both datasets. In Fig. 5.12 the plot for the captured test sequence is presented, while in Fig. 5.13 the same plot is shown for the MSRA dataset.

We observe now that the use of IRA improves the initial estimation provided by the $RBFN_{init}$. Thus, the accuracy of the method increases in both datasets. We can also observe that for some difficult poses the $RBFN_{init}$ estimates poses that the specialized RBFNs can not further refine, on the contrary specialized RBFNs worsen the final estimation. Nevertheless, it is noteworthy to say that the method performs better on the captured test sequence, than on the MSRA dataset. We

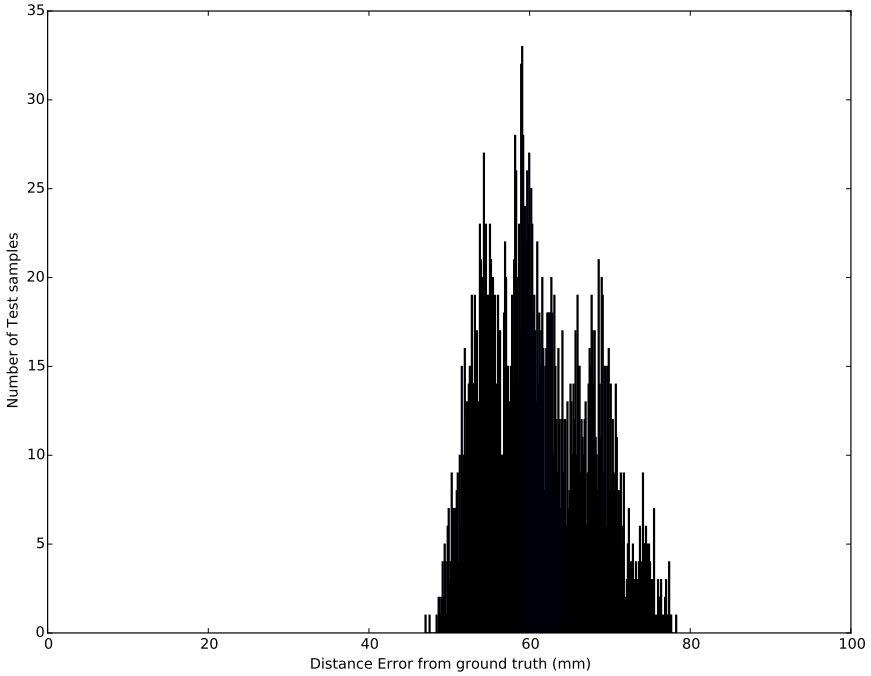


Figure 5.10: Histogram of distance errors for the estimations on the captured sequence of $RBFN_{init}$ model.

Table 5.1: Average distance errors (mm) for the four experiments.

	Captured test sequence	MSRA dataset
$RBFN_{init}$ and IRA	**44.37**	**49.60**
$RBFN_{init}$ only	61.13	61.45

hypothesize that the different hand shapes and sizes as well as the varying depth ranges contribute to this degradation in performance. Still, the performance is very appealing given the fact that the method achieves a generalization from synthetic to real data.

Summarizing, we collect all average distances in Table 5.1, and all deviations of our experiments in Table 5.2.

The use of IRA is significant for the performance of our method, proving that the specialized RBFNs manage to refine the initial estimations that they are provided by the $RBFN_{init}$.

Table 5.3 presents average distance errors for 4 methods that were tested on the MSRA dataset and our method. All of the 4 methods were trained on 8 subjects of the MSRA dataset, and tested on the held out subject.

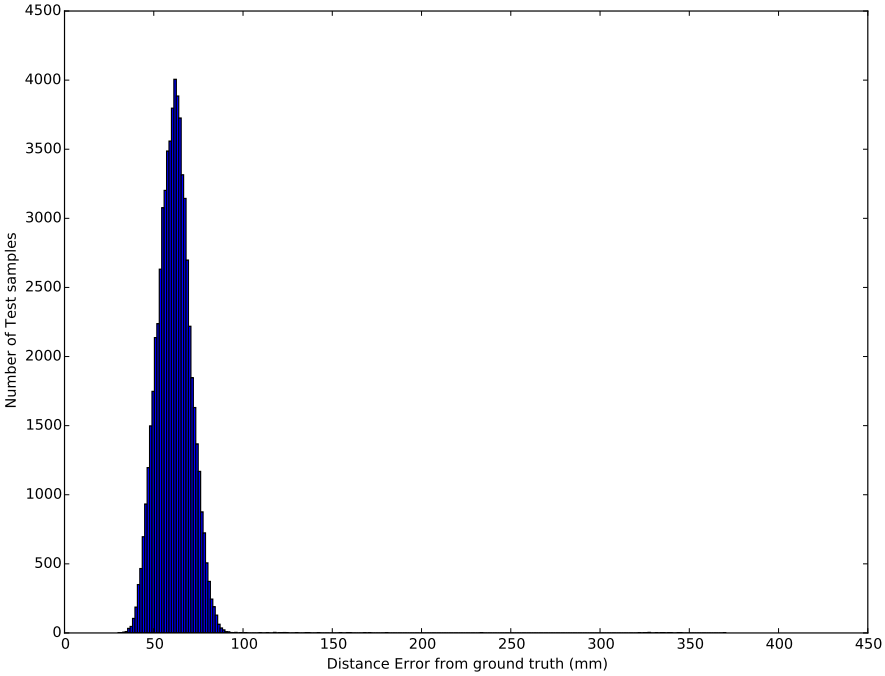


Figure 5.11: Histogram of distance errors for the estimations on the MSRA dataset of $RBFN_{init}$ model.

Table 5.2: Standard deviation (mm) of testing samples for the four experiments.

	Captured test sequence	MSRA dataset
$RBFN_{init}$ and IRA	17.74	12.60
$RBFN_{init}$ only	6.46	10.02

Table 5.3: Comparison of our method with other methods from the literature on the MSRA dataset.

Method	**Median Distance Error**
Wan *et al.* [81]	$25 \pm 2mm$
Sun *et al.* [72]	$28 \pm 2mm$
Wan *et al.* [82]	$32 \pm 2mm$
Ge *et al.* [16]	$20 \pm 2mm$
Ours	$48.93mm$

Note that our method was tested on the whole dataset of 9 subjects and not only on a single (not specified) subject as was done by the other 4 methods.

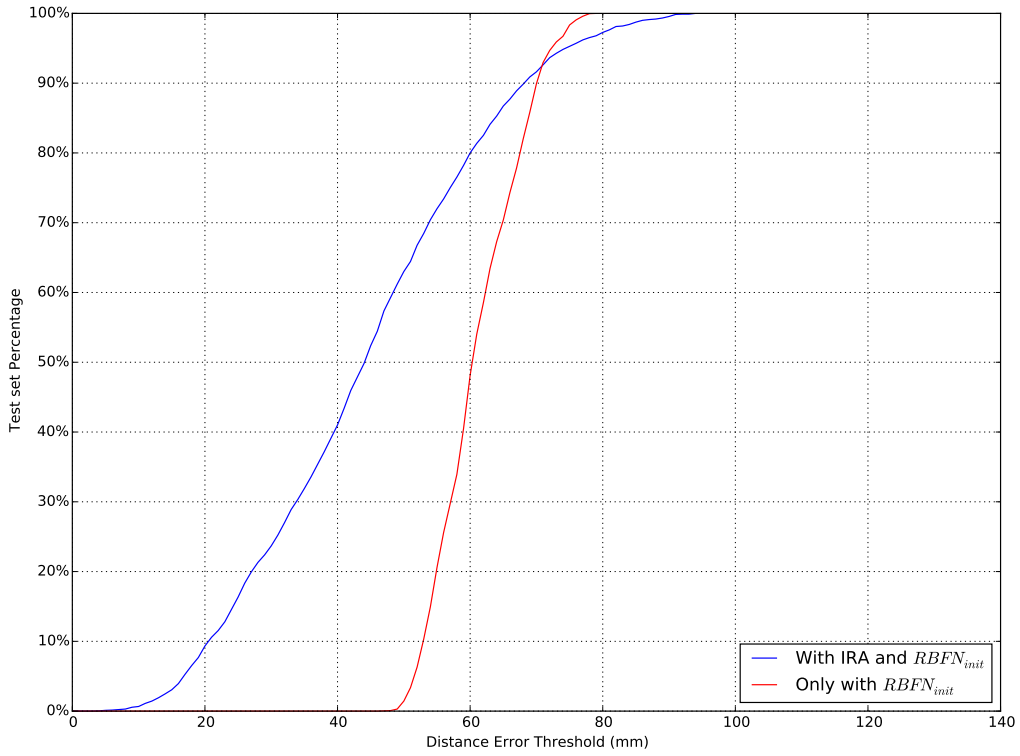


Figure 5.12: Percentage of poses that are under a certain average distance error threshold for the captured test sequence.

We observe that our method has the largest error compared to the other methods. However, we trained our method on a synthetic dataset, meaning that our approach has never seen any frame from the test dataset, or any other real-world dataset for that matter. On the contrary, the compared methods were trained on a part of the dataset that was used for testing. Even though the error we achieve is higher than the other approaches, it is satisfactory as seen by the qualitative results in Sec. 5.4.2, illustrating that our method is capable of generalization.

5.4.2 Qualitative Results

Figure 5.14 shows representative frames of estimated hand poses on the captured test sequence with their respective manually annotated ground truths. And in Fig. 5.15 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.

For the captured test sequence we can observe that the results are quite appealing. For some cases it is clear that the manual annotation was not accurate enough, thus the estimation has a closer similarity to the annotation rather than

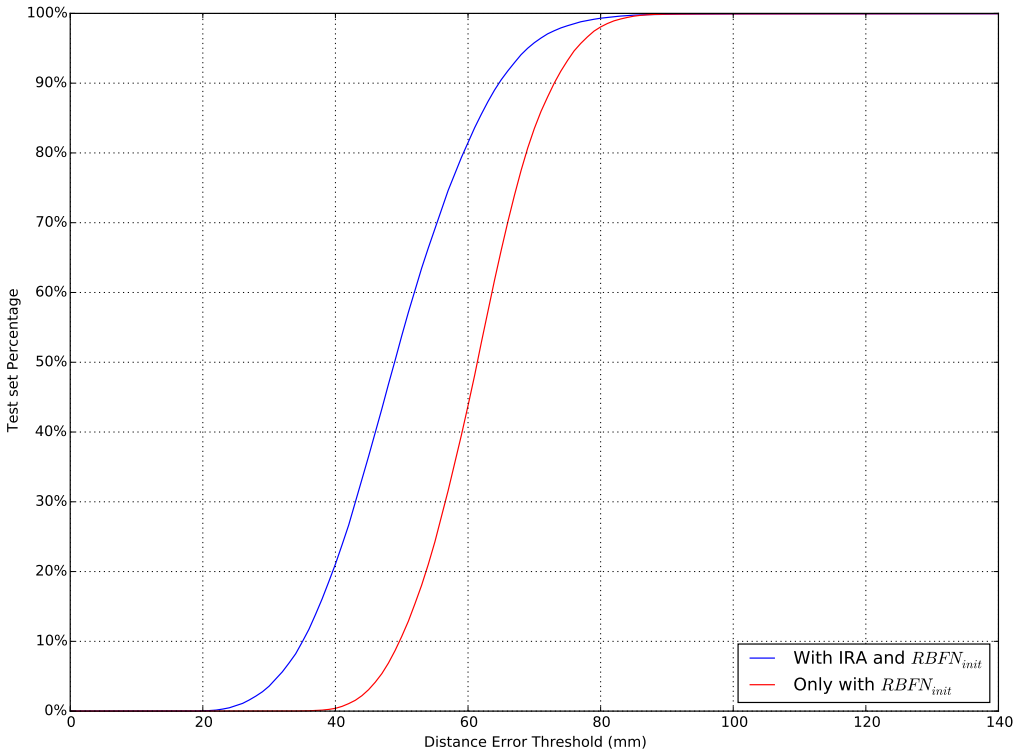


Figure 5.13: Percentage of poses that are under a certain average distance error threshold for MSRA dataset.

to the original RGB image.

For the MSRA dataset we observe that the estimations are quite accurate as well, and the method manages to generalize well on the public dataset 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. 5.16 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 projecting wrong finger articulations that can generate similar depth maps to ones that we insert as input. Other fail 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. While other cases show how for difficult poses where are minor depth changes (as in the case of closed palm) the method fails to estimate accurately the rotation of the hand.

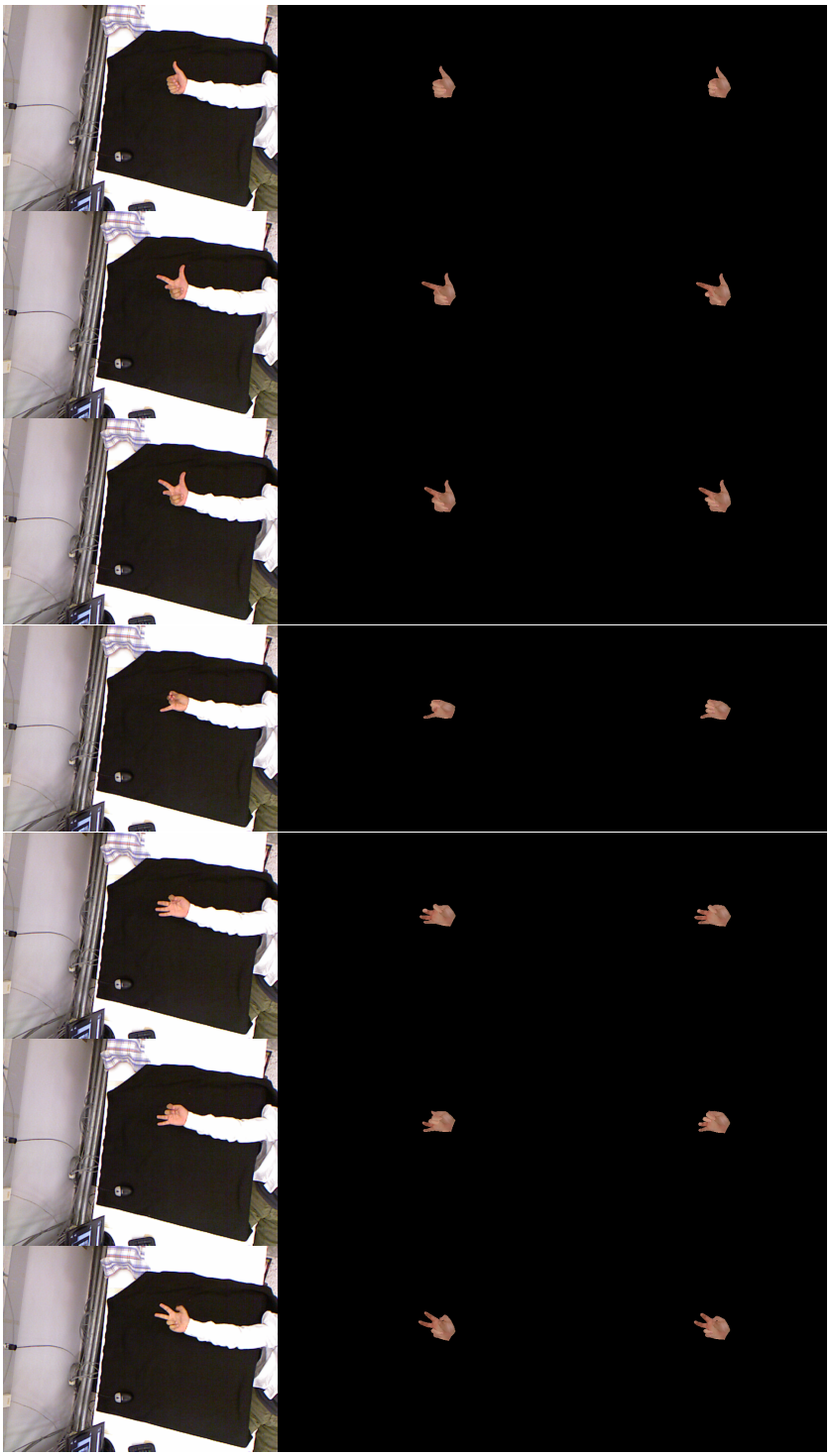


Figure 5.14: Qualitative results on captured test sequence: (Left) is the RGB test image, (Middle) is the manually annotated ground truth, (Right) is the final estimation.

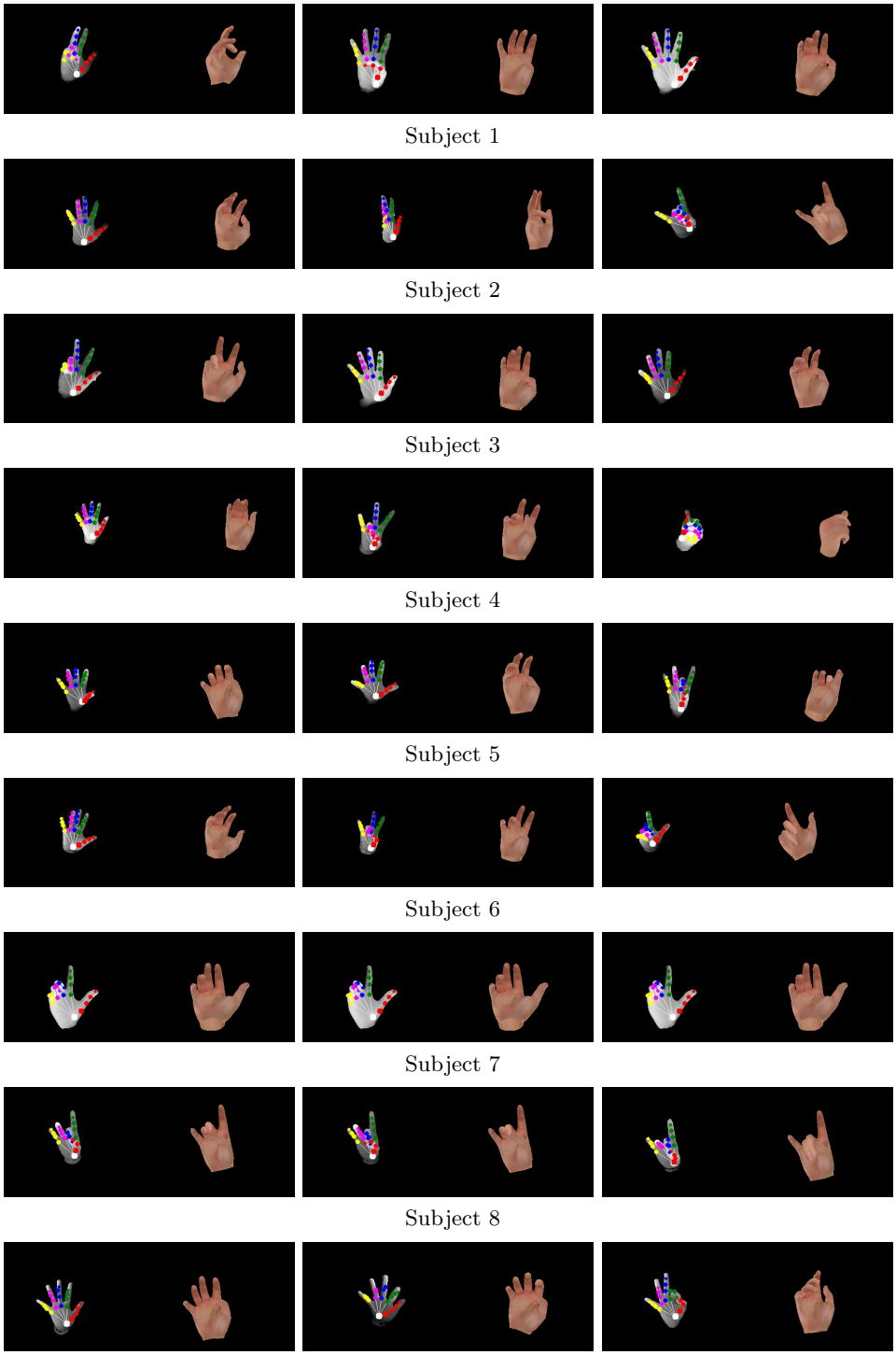


Figure 5.15: Qualitative results on MSRA dataset: Left part of each image sample is the test depth image with the annotated ground truth, Right part of each image sample is the final estimation.

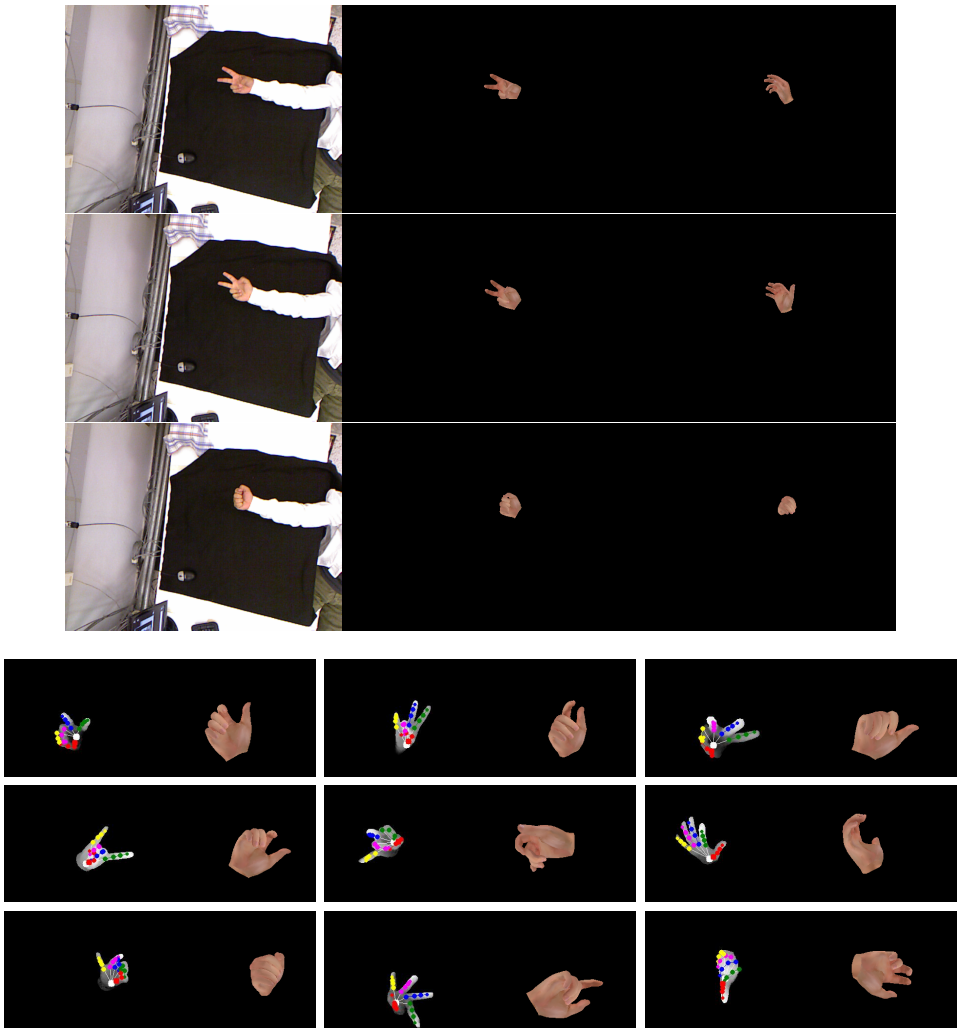


Figure 5.16: Fail cases for both datasets.

Chapter 6

Conclusions

In this thesis 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 give a rough estimation given a depth image. Then with the use of specialized RBFNs we managed to iteratively refine this approximation arriving on a final estimation. The use of RBFNs allows our method to make use of a large synthetic dataset, generalizing well to real-world data. We validated our method in order to tune the hyper-parameters of the proposed model. We justify the selection of these hyper-parameters, by presenting quantitative results. Finally, we tested our method on two different testing datasets. The first one is a captured sequence, and the second is a publicly available dataset, that is used to assess hand pose estimation methods. To draw our conclusions, we presented quantitative results of our proposed method, accompanied also by qualitative feedback.

From a theoretical point of view, the significance of the proposed method is that it shows that RBFNs can generalize quite well when learning synthetic data. A RBFN does not search for patterns in local partitions of the data, and tries to associate the data with the defined distances from its centers. A stand alone RBFN might not perform satisfactory, but with the refinement of its initial suggestion, multiple specialized RBFNs can improve the final estimation.

Future Work

Future work includes the investigation of the effects of parallax distortion and the generalization across human hand sizes and shapes. The comparison with other works and datasets would increase the perspective view on the concluded matter. Specifically, experimental comparison with deep learning approaches trained on synthetic data would give clearer support over the assumption that RBFNs generalize better than other deep learning architectures.

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.

Bibliography

[1] Peter J Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. *Lecture Notes in Computer Science: Evolutionary Programming VII*, 1447:601–610, 1998.

[2] Vassilis Athitsos and Stan Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR*, volume 2, page 432, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[3] Luca Ballan and Aparna Taneja. Motion Capture of Hands in Action using Discriminative Salient Points. In *ECCV*, 2012.

[4] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3d hand tracking. *IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, pages 675–680, 2004.

[5] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[6] Guido Bugmann. Normalized gaussian radial basis function networks. *Neurocomputing*, 20(1-3):97–110, 1998.

[7] Inhyok Cha and Saleem A Kassam. Rbfn restoration of nonlinearly degraded images. *IEEE Transactions on Image Processing*, 5(6):964–975, 1996.

[8] Xinghao Chen, Guijin Wang, Hengkai Guo, and Cairong Zhang. Pose Guided Structured Region Ensemble Network for Cascaded Hand Pose Estimation. *arXiv preprint arXiv:1708.03416*, 2017.

[9] T.E. de Campos and D.W. Murray. Regression-based Hand Pose Estimation from Multiple Cameras. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, volume 1, pages 782–789. Ieee, 2006.

[10] Martin de La Gorce, D.J. Fleet, and Nikos Paragios. Model-based 3d hand pose estimation from monocular video. *IEEE Trans. on PAMI*, pages 1–15, February 2011.

[11] Xiaoming Deng, Shuo Yang, Yinda Zhang, Ping Tan, Liang Chang, and Hongan Wang. Hand3D: Hand Pose Estimation using 3D Neural Network. *arXiv preprint arXiv:1704.02224*, 2017.

[12] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[13] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based Hand Pose Estimation: A Review. *CVIU*, 108(1-2):52–73, Oct. 2007.

[14] Shachar Fleishman, Mark Kliger, Alon Lerner, and Gershom Kutliroff. ICPIK: Inverse Kinematics based articulated-ICP. In *Computer Vision and Pattern Recognition Workshops*, volume 2015-Octob, pages 28–35, 2015.

[15] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 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*, pages 3593–3601, 2016.

[16] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 3D Convolutional Neural Networks for Efficient and Robust Hand Pose Estimation from Single Depth Images. In *Proceedings IEEE CVPR*, pages 1991–2000, 2017.

[17] Francisco Gomez-Donoso, Sergio Orts-Escolano, and Miguel Cazorla. Robust Hand Pose Regression Using Convolutional Neural Networks. In *Iberian Robotics conference*, pages 591–602. Springer, 2017.

[18] Alberto Guillén, Ignacio Rojas, Jesús González, Héctor Pomares, Luis Javier Herrera, Olga Valenzuela, and Alberto Prieto. A possibilistic approach to rbfn centers initialization. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 174–183. Springer, 2005.

[19] Hengkai Guo, Guijin Wang, Xinghao Chen, and Cairong Zhang. Towards Good Practices for Deep 3D Hand Pose Estimation. *arXiv preprint arXiv:1707.07248*, 2017.

[20] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *ICCV*, Oct 2009.

[21] Tony Heap and David Hogg. Towards 3D Hand Tracking using a Deformable Model. In *Ieee*, volume 9, pages 140–145, 1996.

[22] Sina Honari, Jason Yosinski, Pascal Vincent, and Christopher Pal. Recombinator Networks: Learning Coarse-to-Fine Feature Aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5743–5752, 2015.

[23] Su Hongtao, David Dagan Feng, and Zhao Rong-Chun. Face recognition using multi-feature and radial basis function network. In *Selected papers from the 2002 Pan-Sydney workshop on Visualisation-Volume 22*, pages 51–57. Australian Computer Society, Inc., 2003.

[24] Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67, 2005.

[25] I. Katsavounidis, C.-C. Jay Kuo, and Z. Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1:144–146, October 1994.

[26] Cem Keskin, Furkan Kirac, Yunus Emre Kara, and Lale Akarun. 3D hand pose estimation and classification using depth sensors. In *Signal Processing and Communications Applications Conference (SIU), 2012 20th*, pages 1–4, 2012.

[27] S Khamis, J Taylor, J Shotton, C Keskin, S Izadi, and A Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 2540–2548, 2015.

[28] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. Digits: Freehand 3D Interactions Anywhere Using a Wrist-Worn Gloveless Sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, page 167, 2012.

[29] Philip Krejov, Andrew Gilbert, and Richard Bowden. Guided optimisation through classification and regression for hand pose estimation. *Computer Vision and Image Understanding*, 155:124–138, 2017.

[30] T Hoang Ngan Le, Kha Gia Quach, Chenchen Zhu, Chi Nhan Duong, Khoa Luu, and Marios Savvides. Robust hand detection and classification in vehicles and in the wild. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1203–1210. IEEE, 2017.

[31] Peiyi Li, Haibin Ling, Xi Li, and Chunyuan Liao. 3D hand pose estimation using randomized decision forest with segmentation index points. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 819–827, 2015.

[32] Daw-Tung Lin. Facial expression classification using pca and hierarchical radial basis function network. *Journal of information science and engineering*, 22(5):1033–1046, 2006.

[33] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[34] Ziyang Ma and Enhua Wu. Real-time and robust hand tracking with a single depth camera. In *Visual Computer*, volume 30, pages 1133–1144, 2014.

[35] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[36] Meysam Madadi, Sergio Escalera, Xavier Baro, and Jordi Gonzalez. End-to-end Global to Local CNN Learning for Hand Pose Recovery in Depth data. *arXiv preprint arXiv:1705.09606*, 2017.

[37] A Makris, N Kyriazis, and A A Argyros. Hierarchical Particle Filtering for 3D Hand Tracking. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*, pages 8–17, 2015.

[38] Alexandros Makris and Antonis Argyros. Model-based 3D Hand Tracking with on-line Shape Adaptation. In *British Machine Vision Conference*, pages 77.1–77.12, 2015.

[39] Chris McCormick. Radial basis function network (rbfn) tutorial. `http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial`, Aug 2013.

[40] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. In *ACM Transactions on Graphics (SIGGRAPH 2017)*, 2017.

[41] Arpit Mittal, Andrew Zisserman, and Philip HS Torr. Hand detection using multiple proposals. In *BMVC*, pages 1–11, 2011.

[42] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map. pages 29–31, 2017.

[43] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. *arXiv preprint arXiv:1704.02201*, 2017.

[44] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[45] Markus Oberweger and Vincent Lepetit. DeepPrior++: Improving Fast and Accurate 3D Hand Pose Estimation. In *ICCV workshop*, volume 840, 2017.

[46] Markus Oberweger, Gernot Riegler, Paul Wohlhart, and Vincent Lepetit. Efficiently creating 3d training data for fine hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4957–4965, 2016.

[47] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

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

[49] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Efficient Model-based 3D Tracking of Hand Articulations using Kinect. In *BMVC*, Dundee, UK, Aug. 2011.

[50] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Full DOF Tracking of a Hand Interacting with an Object by Modeling Occlusions and Physical Constraints. In *ICCV*, pages 2088–2095. IEEE, Nov. 2011.

[51] Paschalis Panteleris and Antonis Argyros. Back to RGB: 3D tracking of hands and hand-object interactions based on short-baseline stereo. *arXiv preprint arXiv:1705.05301*, 2017.

[52] Paschalis Panteleris, Nikolaos Kyriazis, and Antonis A Argyros. 3D Tracking of Human Hands in Interaction with Unknown Objects. In *British Machine Vision Conference (BMVC)*, pages 123.1–123.12, 2015.

[53] Georg Poier, Konstantinos Roditakis, and Samuel Schulter. Hybrid One-Shot 3D Hand Pose Estimation by Exploiting Uncertainties. *Bmvc2015*, 33(5):2014, 2015.

[54] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1113, 2014.

[55] J. Rehg and T Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. *ECCV*, vol:35–46, 1994.

[56] Grégory Rogez, James S. Supančič, and Deva Ramanan. First-person pose recognition using egocentric workspaces. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 4325–4333, 2015.

[57] Grégory Rogez, James S Supancic, and Deva Ramanan. Understanding everyday hands in action from rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3889–3897, 2015.

[58] Javier Romero, H. Kjellström, and Danica Kragic. Hands in action: Real-time 3D reconstruction of hands in interaction with objects. In *IEEE International Conference on Robotics and Automation*, pages 3–8, 2010.

[59] Javier Romero, Hedvig Kjellstrom, and Danica Kragic. Monocular real-time 3d articulated hand pose estimation. *IEEE-RAS Int'l Conf. on Humanoid Robots*, Dec 2009.

[60] Romer Rosales, Vassilis Athitsos, Leonid Sigal, and Stan Sclaroff. 3D hand pose reconstruction using specialized mappings. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 378–385. IEEE, 2001.

[61] Romer Rosales, Vassilis Athitsos, Leonid Sigal, and Stan Sclaroff. 3D Hand Pose Reconstruction Using Specialized Mappings. In *ICCV*, 2001.

[62] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.

[63] Chiho Choi Sang, Ho Yoon, Chin-Ning Chen, and Karthik Ramani. Robust Hand Pose Estimation during the Interaction with an Unknown Object. In *Iccv 2017*, pages 3123–3132, 2017.

[64] T Schmidt, R Newcombe, and D Fox. {DART}: Dense Articulated Real-Time Tracking. *Proc.˜of Robotics: Science and Systems (RSS)*, (1), 2014.

[65] Friedhelm Schwenker, Hans A Kestler, and Günther Palm. Three learning phases for radial-basis-function networks. *Neural networks*, 14(4-5):439–458, 2001.

[66] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[67] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. In *CVPR*. IEEE, 2011.

[68] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 6, 2017.

[69] Ayan Sinha, Chiho Choi, and Karthik Ramani. DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features. In *Computer Vision and Pattern Recognition*, pages 4150–4158, 2016.

[70] Shuran Song and Jianxiong Xiao. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2015.

[71] Srinath Sridhar, Franziska Mueller, Michael Zollhöfer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9906 LNCS, pages 294–310, 2016.

[72] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 824–832, 2015.

[73] James S Supancic, Grégory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. Depth-based hand pose estimation: data, methods, and challenges. In *Proceedings of the IEEE international conference on computer vision*, pages 1868–1876, 2015.

[74] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust Articulated-ICP for Real-Time Hand Tracking. In *Computer Graphics Forum*, 2015.

[75] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3786–3793, 2014.

[76] Jonathan Taylor, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, Jamie Shotton, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, and Julien Valentin. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics*, 35(4):1–12, 2016.

[77] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[78] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33, August 2014.

[79] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing Hands in Action Using Discriminative Salient Points and Physics Simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016.

[80] Zekeriya Uykan, Cuneyt Guzelis, M Ertugrul Celebi, and Heikki N Koivo. Analysis of input-output clustering for determining centers of rbfn. *IEEE transactions on neural networks*, 11(4):851–858, 2000.

[81] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation. In *CVPR*, 2017.

[82] Chengde Wan, Angela Yao, and Luc Van Gool. Direction matters: hand pose estimation from local surface normals. In *European Conference on Computer Vision*, pages 554–569. Springer, 2016.

[83] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3):1, July 2009.

[84] Ying Wu, J.Y. Lin, and T.S. Huang. Capturing natural hand articulation. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, pages 426–432, 2001.

[85] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *ISRN Applied Mathematics*, 2012, 2012.

[86] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3456–3462, 2013.

[87] Qi Ye, Shanxin Yuan, and Tae Kyun Kim. Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9912 LNCS, pages 346–361. Springer, 2016.

[88] Işık Yilmaz and Oguz Kaynar. Multiple regression, ann (rbf, mlp) and anfis models for prediction of swell potential of clayey soils. *Expert systems with applications*, 38(5):5958–5966, 2011.

[89] Lu Yingwei, Narasimhan Sundararajan, and Paramasivan Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural computation*, 9(2):461–478, 1997.

[90] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3D Hand Pose Tracking and Estimation Using Stereo Matching. *arXiv:1610.07214*, 2016.

[91] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*, 2016.

[92] Christian Zimmermann and Thomas Brox. Learning to Estimate 3D Hand Pose from Single RGB Images. *arXiv preprint arXiv:1705.01389*, 2017.