5-2018

Computational Learning for Hand Pose Estimation

Chiho Choi
Purdue University

COMPUTATIONAL LEARNING FOR HAND POSE ESTIMATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Chiho Choi

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Karthik Ramani, Chair

School of Mechanical Engineering

School of Electrical and Computer Engineering (by courtesy)

Dr. Mireille Boutin

School of Electrical and Computer Engineering

Dr. Stanley H. Chan

School of Electrical and Computer Engineering

Dr. Jeffrey M. Siskind

School of Electrical and Computer Engineering

Approved by:

Dr. Jay P. Gore

Head of the School Graduate Program

To my family.

ACKNOWLEDGMENTS

First, I would like to thank my advior Dr. Karthik Ramani for his dedicated support and insightful guidence throughout the duration of this work. Dr. Ramani stimulated my interest in the field of computer vision and machine learning, in particular for 3D hand pose estimation. I could have never formed my research area without his wholehearted endorsement. I am very grateful to my committee members, Dr. Mireille Boutin, Dr. Stanley H. Chan, and Dr. Jeffrey M. Siskind for their invaluable support and feedback on my research.

I also would like to thank all friends and collegues in the C-Design lab (past and present), in particular Sujin, Sang, Wei, Yunbo, Hairong, Ke, Vinayak, Cecil, Ayan, Chin-Ning, Min Gon, Young Chun, and Jae Hyun for their casual but most helpful discussion on issues of my life and in shaping my researh. Moreover, I am truly grateful to Joon Hee Choi for sharing his knowldege in depth.

Next, I would like to acknowldege the National Science Foundation (No. 1235232 from CMMI, No. 1329979 from CPS, and No. 1637961 from NRI) and the Donald W. Feddersen Chaired Professorship for supporting this work.

I am sincerely grateful to my parents (Byung Kil Choi and Bok Dong Son) for their help, dedication, support, encourgement, and unconditional love which make me shape my aspiration. I also would like to express my gratitude for my late grandma, Deok Soon Lee, who loved me more than anyone in her life. I thank my lovely son, Anders, for his sweet smile and for giving me happiness during the last ten months of my studies. Finally, I want to express my deepest love and thanks to my dear wife, Jeong, who has been with me all these years and has made them the best years of my life.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

SYMBOLS

$\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\phi})$	Synthetic hand model
$\boldsymbol{\theta}$	joint angle parameters
$\boldsymbol{\phi}$	joint position parameters
$\mathbf{s}$	hand skeleton vertex coordinates
$\mathbf{v}$	visible point cloud for a hand pose
$\mathbf{d}$	Euclidean distance to poses in basis
$\mathbf{c}$	shape descriptor for a pose
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	latent factor matrices
$\mathbf{D}$	distance matrix
$\mathbf{P}$	parameter matrix
k	number of nearest neighbors
Δ	Laplace-Beltrami operator
J	heat sources
T	time steps
P	number of vertices

ABBREVIATIONS

HCI	human computer interaction
AR	augmented reality
VR	virtual reality
ConvNet	convolutional neural network
JMFC	joint matrix factorization and completion
PSO	particle swarm optimization
IK	inverse kinematics
FPS	frames per second
JL	Johnson-Lindenstrauss
DOF	degrees of freedom
DIP	distal interphalangeal joint
PIP	proximal interphalangeal joint
MCP	metacarpophalangeal joint
IP	interphalangeal joint
TM	trapeziometacarpal joint
ALS	alternative least squares
NN	nearest neighbor
GT	ground truth
PQ	product quantization
RF	random forest
SVM	support vector machine
DN	depth network
HDN	heat distribution network
HN	hallucination network
RN	refinement network

GLOSSARY

depth camera	RGB-D camera sensor
depth map	each pixel contains a measure of distance in mm between a sensor and reflector
depth image	each pixel is normalized in the range of [0 - 255]
confidence map	heatmap
3D keypoints	hand joint locations
end-effector	coordinates of the fingertips

ABSTRACT

Choi, Chiho Ph.D., Purdue University, May 2018. Computational Learning for Hand Pose Estimation. Major Professor: Dr. Karthik Ramani, School of Mechanical Engineering, School of Electrical and Computer Engineering (by courtesy).

Rapid advances in human–computer interaction interfaces have been promising a realistic environment for gaming and entertainment in the last few years. However, the use of traditional input devices such as trackballs, keyboards, or joysticks has been a bottleneck for natural interactions between a human and computer as two points of freedom of these devices cannot suitably emulate the interactions in a three-dimensional space. Consequently, a comprehensive hand tracking technology is expected as a smart and intuitive option to these input tools to enhance virtual and augmented reality experiences. In addition, the recent emergence of low-cost depth sensing cameras has led to their broad use of RGB-D data in computer vision, raising expectations of a full 3D interpretation of hand movements for human–computer interaction interfaces. Although the use of hand gestures or hand postures has become essential for a wide range of applications in computer games and augmented/virtual reality, 3D hand pose estimation is still an open and challenging problem because of the following reasons: (i) the hand pose exists in a high-dimensional space because each finger and the palm is associated with several degrees of freedom, (ii) the fingers exhibit self-similarity and often occlude to each other, (iii) global 3D rotations make pose estimation more difficult, and (iv) hands only exist in few pixels in images and the noise in acquired data coupled with fast finger movement confounds continuous hand tracking.

The success of hand tracking would naturally depend on synthesizing our knowledge of the hand (*i.e.*, geometric shape, constraints on pose configurations) and latent features about hand poses from the RGB-D data stream (*i.e.*, region of interest, key

feature points like finger tips and joints, and temporal continuity). In this thesis, we propose novel methods to leverage the paradigm of *analysis by synthesis* and create a prediction model using a population of realistic 3D hand poses. The overall goal of this work is to design a concrete framework so the computers can learn and understand about perceptual attributes of human hands (*i.e.*, self-occlusions or self-similarities of the fingers) and to develop a pragmatic solution to the real-time hand pose estimation problem implementable on a standard computer.

This thesis can be broadly divided into four parts: learning hand (i) from recommendiations of similar hand poses, (ii) from low-dimensional visual representations, (iii) by hallucinating geometric representations, and (iv) from a manipulating object. Each research work covers our algorithmic contributions to solve the 3D hand pose estimation problem. Additionally, the research work in the appendix proposes a pragmatic technique for applying our ideas to mobile devices with low computational power. Following a given structure, we first overview the most relevant works on depth sensor-based 3D hand pose estimation in the literature both with and without manipulating an object. Two different approaches prevalent for categorizing hand pose estimation, model-based methods and appearance-based methods, are discussed in detail. In this chapter, we also introduce some works relevant to deep learning and trials to achieve efficient compression of the network structure. Next, we describe a synthetic 3D hand model and its motion constraints for simulating realistic human hand movements. The section for the primary research work starts in the following chapter. We discuss our attempts to produce a better estimation model for 3D hand pose estimation by learning hand articulations from recommendations of similar poses. Specifically, the unknown pose parameters for input depth data are estimated by collaboratively learning the known parameters of all neighborhood poses. Subsequently, we discuss deep-learned, discriminative, and low-dimensional features and a hierarchical solution of the stated problem based on the matrix completion framework. This work is further extended by incorporating a function of geometric properties on the surface of the hand described by heat diffusion, which is robust to capture both

the local geometry of the hand and global structural representations. The problem of the hands interactions with a physical object is also considered in the following chapter. The main insight is that the interacting object can be a source of constraint on hand poses. In this view, we employ pose dependency on the shape of the object to learn the discriminative features of the hand–object interaction, rather than losing hand information caused by partial or full object occlusions. Subsequently, we present a compressive learning technique in the appendix. Our approach is flexible, enabling us to add more layers and go deeper in the deep learning architecture while keeping the number of parameters the same. Finally, we conclude this thesis work by summarizing the presented approaches for hand pose estimation and then propose future directions to further achieve performance improvements through (i) realistically rendered synthetic hand images, (ii) incorporating RGB images as an input, (iii) hand perseonalization, (iv) use of unstructured point cloud, and (v) embedding sensing techniques.

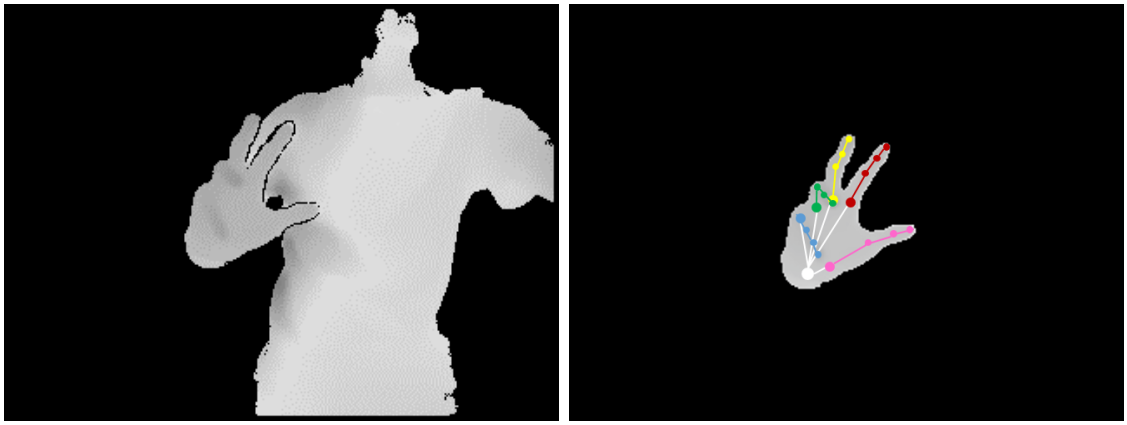
1. INTRODUCTION

For human beings, hand has been used as a most intuitive and natural way to interact with the outside world. This tendency towards communicating with computers has become essential for interactions in a three-dimensional (3D) space from a wide range of human computer interaction (HCI) interfaces. In addition, the real-time depth data acquisition from commercial sensors further accelerated a need for accurate 3D hand pose estimation to recognize finger movements for augmented reality (AR) and virtual reality (VR) applications.

As the development of reliable and low-cost sensing technologies has helped to simplify the tasks of hand pose estimation, extensive and lengthy research [1–9] has been conducted on finding a robust and efficient solution for kinematic pose estimation in the literature. However, a comprehensive hand tracking technology that would enhance virtual and augmented reality experiences still does not exist. The current approaches have been partially directed toward identifying (i) the articulation complexity of the hand, (ii) self-similarity and self-occlusion of the fingers, and (iii) data acquisition artifacts such as depth noise. Therefore, these solutions do not work consistently with general human-computer interaction interfaces and augmented/virtual reality applications.

1.1 Research Goals

The problem addressed in this thesis is to find an efficient and robust solution that aims to estimate complex kinematic poses of the articulated hand using a single depth camera. To achieve this, this thesis introduces a supervised learning method to build a prediction model using a population of realistic 3D hand poses. We leverage the paradigm of *analysis by synthesis* and generate synthetic depth maps by imposing



(a) (b)

Figure 1.1. The input and output of the proposed system. (a) The input depth data acquired from a RGB-D camera, and (b) the estimated 3D hand pose after background removal.

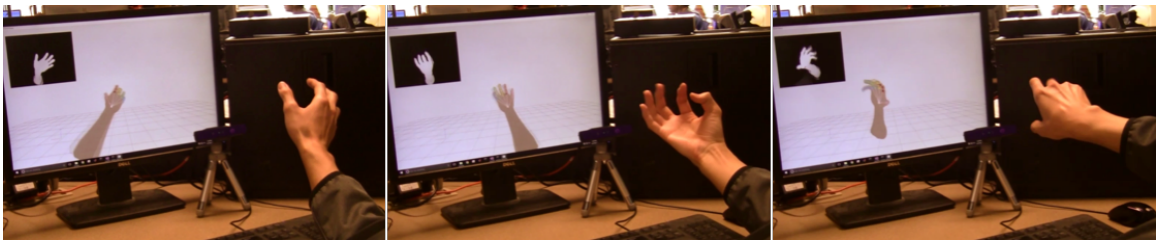


Figure 1.2. System setup for real-time hand pose estimation. A depth map is obtained from a single depth sensor such as Microsoft Kinect or Intel RealSense. Our prediction model estimates a set of joint angle paramters that can be used to reconstruct the given hand pose.

both static (*e.g.*, range of motion, joint length, location) and dynamic (*e.g.*, among joints and fingers) constraints. The uniformly sampled joint angle parameters render joint configurations and finger movement in a configuration space restricted by functional constraints of the hand. Our prediction model is then trained using the created depth maps that are reflective of real poses to efficiently and effectively learn a wide range of hand articulations and their representations. Along this line, three pragmatic solutions are continuosly studied and presented in the context of *computational*

learning and consequently for hand pose estimation. In addition to the solutions of an isolated hand, the problem of the hands interactions with a physical object is proposed as an extension in a 3D hand pose estimation domain.

1.2 Inspiration

In this section, we briefly describe our main insights that enabled us to understand kinematic hand poses and naturally motivated us to develop a novel pose estimation system.

1.2.1 Recommender systems

Our first insight is that a recommender system is very similar to a pose tracking system as shown in Figure 1.3. Both systems have some intrinsic and extrinsic information about its constituent objects, the user in a recommender system and individual poses in a tracking system. The intrinsic knowledge of the hand in a tracking system corresponds to a known user ratings in a recommender system. Similarly, the extrinsic RGB-D point cloud information corresponds to the metadata available about users. Specifically, the hand pose estimation problem is analogous to the cold-start problem in recommender systems.

The cold-start problem in recommender systems is to suggest personalized items to a new user with unknown preferences [10]. In analogy to a tracking system, the hand pose estimation problem is to evaluate the unknown pose parameters of the kinematic hand model for a new point clouds appearing at every instant of time via a RGB-D sensor. A common technique to alleviate the cold-start problem is to suggest items to a new user based on recommendations available for like-minded users [11]. The like-mindedness or similarity between users is evaluated using metadata such as age, gender, geographical location, interests, etc [12]. Following a similar approach, the nearest neighbors to an arriving point cloud with known parameter values are efficiently found using local shape descriptors from a large database of hand poses.

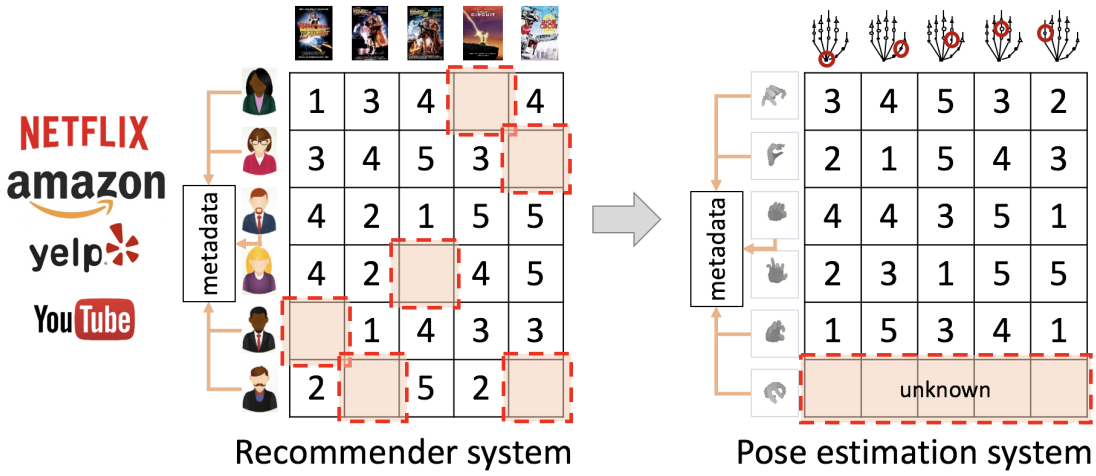


Figure 1.3. A recommer system is analoguous to a pose estimation system. In recommender systems, the unknown ratings can be predicted using the information of similarly behavioring users. Whereas, the unknown pose parameters can be estimated by analyzing the information of similar hand poses in the pose estimation system.

Subsequently, the unknown pose parameters for this point cloud are estimated by *collaboratively* regressing the known parameters of all neighborhood poses.

1.2.2 Biological neural networks

The other insight is to follow biological processes of the animal visual cortex to implicitly learn about visual representation of similar hand poses. Convolutional neural network (ConvNet) consists of multiple layers of small neuron collections, which respond to overlapping regions of the input image for extracting better feature representations.

ConvNets have achieved ground-breaking performance in image classification [13, 14] and video recognition [15, 16]. With the boom of interest in deep learning, 3D hand pose estimation is increasingly becoming a part of the learning and development processes of mid-level features learned from a large dataset. However, a naive strategy to replace the classification layer in a deep neural net with a regression layer leads to errors, as the objective function often gets stuck in a local minima. Pervious

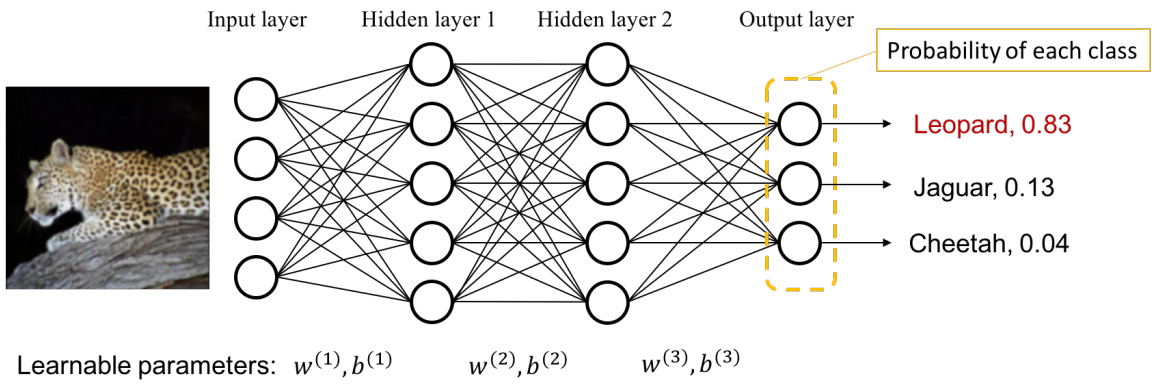


Figure 1.4. Deep neural network. A neural network is trained for the image classification task and outputs the probability of each class.

approaches have been proposed for estimating hand poses to decrease errors and find a global minima by incorporating a prior model [6], regressing heatmap features from a single view [3] and multiple views [8], and synthesizing a hand pose in a closed loop [7] using a convolutional neural network architecture. Different from these approaches, ConvNets are trained to output a discriminative low dimensional *activation feature* in the penultimate fully connected layer. This activation vector represents either the global hand orientation or the local articulations of the five fingers, given a depth map. The main insight is that a pool of (spatially or temporally) nearby activation features to an input activation feature can better represent the hand pose. The ConvNets automatically learn the scope of training (local or global), the type of the finger (thumb, index, middle, ring, or little), and prevalent occlusions by simply inputting the discretized class of the pose parameter values.

1.2.3 Geometric representation

The behavior of heat diffusion on the surface of a shape has generally been considered to be geometric features by analyzing a shape operator computed from the heat kernel matrix. The operator investigates the local geometry of the shape at

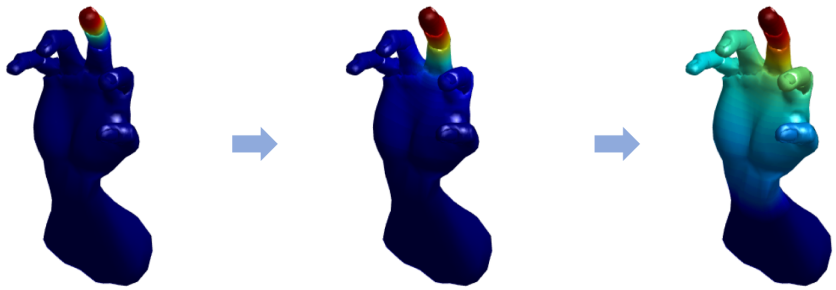


Figure 1.5. The behavior of heat diffusion. The point heat source is placed at the tip of the middle finger at time 0, and after some amount of time, the heat is diffused to the neighboring points.

small time scales and captures the global structure at large scales to be insensitive to non-rigid deformation, topological changes, and noise present in 3D models. Consequently, the shape signatures/descriptors built on such descriptive representations have been extensively studied in the geometry community [17–19] for shape matching and retrieval. The robustness for identifying the points on the mesh surface naturally motivates us to pursue 3D keypoint retrieval (*i.e.,* hand joint positions) in the hand pose estimation problem. Having it in our mind, we build a heat distribution descriptor that incorporates the deformation invariant properties of heat diffusion over an articulated hand at multiple scales. Therefore, our method is robust to the changes of the topology of the hand and noise present in input data.

1.2.4 Modality hallucination

The concept of modality hallucination has been previously presented in [20, 21] to produce a more informed model on visual recognition tasks. Our work shares analogies with [22] which transfers mid-level depth features extracted from an RGB image across domains. The potential for modality hallucination motivates us to consider learning an additional representation which is informed by analysis of the multi-scale heat distribution property, in the form of the articulated hand. Our main insight is that

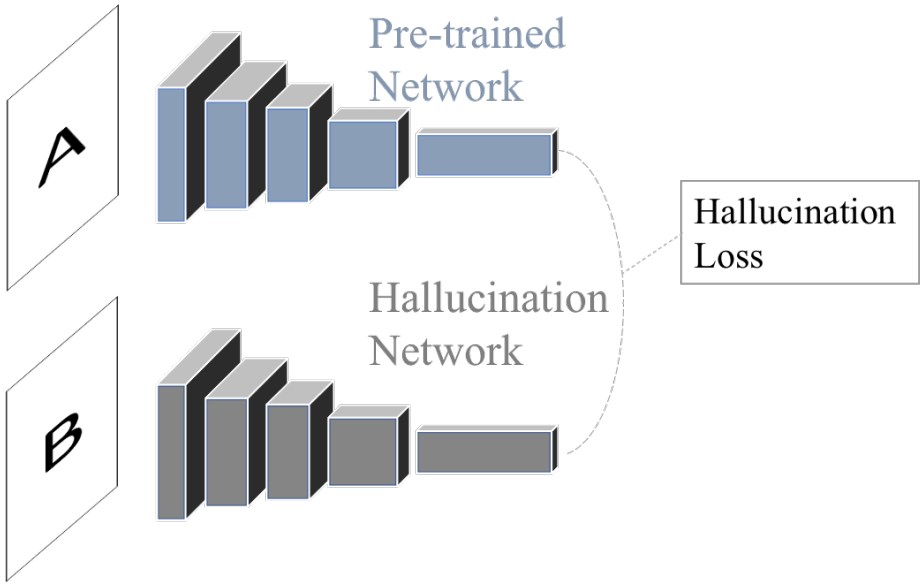


Figure 1.6. Modality hallucination. The hallucination newtwork is trained to mimic same feature representations that are learned from a dataset A using a different input modality B.

a geometrically consistent representation of the heat distribution modality can be learned from a single depth image, in addition to mid-level depth features. We use the resulting geometric responses together with depth features to further enhance the regression accuracy of the system. In practice, we found this step implicitly penalizes the initial estimates to be more effective and robust than the depth-alone framework.

1.2.5 Pose dependency on the shape of an object

Our fundamental observation from earlier work [23, 24] is that the interacting object can be a source of constraint on hand poses (see Figure 1.7). In this view, we employ pose dependency on the shape of the object to learn discriminative features of the hand-object interaction. The input images are used to extract grasp features encoded in pairs – one from a hand perspective and the other from an object perspective.

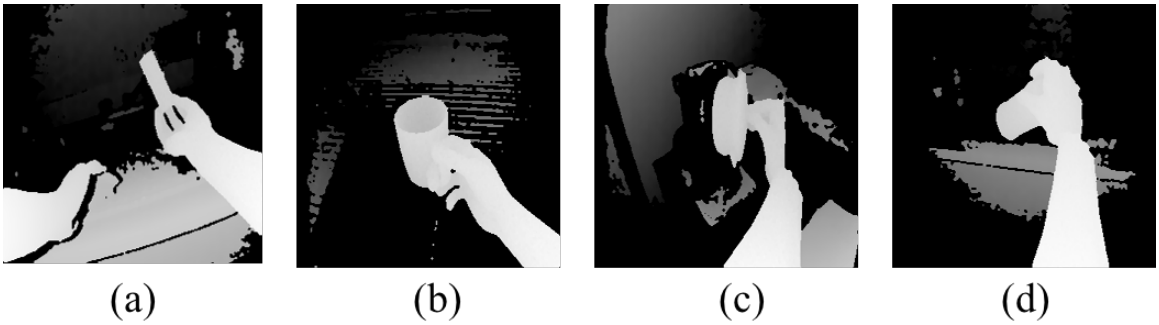


Figure 1.7. Pose dependency on the shape of an object. The shape of an object – (a) cylinder, (b) mug, (c) lid, and (d) cup – causes a configuration of the hand in the form of a hand grasp.

The partial or full loss of hand information during the interaction with hands cannot be recovered particularly when unknown objects are introduced. Instead of processing low-level data to recover or remove the region of object occlusions, we draw a ConvNet framework to extract informative expressions of grasps from those regions. We assume that there is a strong relation between the shape of the object and the configuration of the hand poses in the context of hand grasp. Thus, our model collaboratively learns the convolutional features about grasps from a hand and object perspective in pairs by sharing intermediate representations between two networks in the feature space.

1.3 Overview

This section states the contributions of the thesis and presents a detailed outline of the following sections.

1.3.1 Contributions

The main contributions are summarized as follows:

- A joint matrix factorization and completion (JMFC) model to collaboratively assess auxiliary information of nearest neighbors for regressing unknown pose parameters.
- A construction of a massive synthetic pose population using a 3D meshed hand based on the kinematic constraints, which mimics real hand gestures.
- Efficient nearest neighbor retrieval from the pose population using image feature descriptors applied on 3D depth map.
- Use of discriminate activation features of deep convolutional neural networks (ConvNets) in the penultimate fully connected layer.
- A hierarchical pipeline for hand pose estimation that combines the global pose orientation and finger articulations in a principled way.
- Pixel-wise segmentation of an articulated hand using a ConvNet architecture which is robust to the cluttered background and efficient to compute in real-time.
- Multi-scale geometric representations of the hand as a heat distribution descriptor which compactly encodes the information of hand articulations.
- Modality hallucination using a single depth image, which transfers additional feature representations to produce a more informed estimation model.
- The penalization of the initially predicted joint angle parameters with the guidance of the end-effectors (*i.e.*, the coordinates of the fingertips) in a feature space.
- Localization of an articulated hand and unknown object using a ConvNet architecture that directly regresses the heatmaps corresponding to the center position of the targets.

- Use of object shape information as a latent cue to estimate a hand pose in the form of grasp classification.
- Pixel-wise recreation of input data to correct the error of the sensor and mimic the attributes of synthetic data, which makes the system more robust.
- A multi-channel pipeline to encode the grasp representations in pairs from an unknown object along with an observed hand.
- A compressive learning architecture which is flexble to desgn more complicated structure by adding layers while preserving the amount of parameters the same.
- A pragmatic solution to the real-time hand pose estimation problem, implementable on a standard computer.

1.3.2 Thesis outline

The rest of this paper is organized as follows. In Chapter 2, we review the most relevant literature on 3D hand pose estimation and hand-object interaction (model-based and appearance-based approaches using a single RGB-D camera) as well as compressive neural networks. Chapter 3 describes a synthetic 3D hand model and generation of pose population base on hand constraints for natural and realistic poses. The novel framework of collaborative filtering for hand pose estimation is discussed in Chapter 4. Subsequently in Chapter 5, deep convolutional neural networks based pose parameter regression is presented followed by comparison to the state-of-the-art approaches. In Chpater 6, we discuss a multi-scale heat distribution descriptor and present a detailed explanation of the proposed hallucination framework. Chapter 7 describes our novel architecture for hand pose estimation during the interaction with an unknown object. In Appendix A, we present our effort to solve the memory and computational efficiency problem of a deep neural network. Finally in Chapter 8, future research directions are discussed in detail.

2. RELATED WORK

A variety of approaches have been proposed over the last decade for hand pose estimation. These include, without claim of exhaustivity, wearable (e.g., camera, gloves) and marker based approaches, techniques reliant on RGB input from single or multiple cameras, and more recently depth camera or RGB-D input based approaches. We review some work relevant to our depth-camera based approach and readers are referred to [25] for a comprehensive review of literature.

Approaches for hand-pose estimation can be categorized into either model-based (*generative*) methods, or appearance-based (*discriminative*) methods. An explicit hand model guides model-based methods to recover the hand pose. Current model-based approaches use particle swarm optimization (PSO) [1] or a Gauss-Seidel solver [26] to resolve the hand configuration. Although straight forward to implement, these methods depend on prior motion for initializing the solvers and have high computational complexity. As a result, the pose estimates from these methods are poor for non-contiguous data and they often do not run in real-time even with a GPU acceleration. Contrary to these works, appearance-based methods establish a map between image features and a library of hand pose configurations. Although these menthods do not explore model drift and achieve real-time processing of pose estimation, they are susceptible to self-occlusions and self-similarities of the fingers.

2.1 Hand Pose Estimation

2.1.1 Pose estimation of an isolated hand

Appearance-based approaches A system for 3D hand pose estimation has been developed through the use of a large database. Following the pioneering work in human-pose estimation [27], similar appearance based methods are proposed for hand

pose estimation in [28–30]. This group of approaches provides a trained classifier or regressor [31, 32] to find a mapping between image features and corresponding hand configurations. Compared to a human body, however, the human hand is smaller, more flexible, and severely affected by self-occlusion. Consequently, these methods lose track under low-resolution, output kinematically invalid solutions, and lack robustness against occlusion. In [33, 34], local pose regression methods are presented, demonstrating the efficacy of their approach against occlusions. While successful in many cases, they may experience jitters between frames when image features are insufficient to discriminate different poses. Recently, a convolutional neural network framework has been employed to improve the robustness to occlusions and jitters replacing hand-crafted features. Hand poses are estimated by incorporating a prior model [6], regressing the heatmaps from a single view [3] and multiple views [8], and synthesizing a hand pose in a closed loop [7]. However, these methods either require a comprehensive training dataset which is manually annotated by different individuals to ensure robust tracking or does not provide a complete framework for an interactive environment as they assume the hand is localized and preprocessed. To our knowledge, we present the first work for 3D hand pose estimation that (i) provides a prediction model completely trained using a synthetic dataset, (ii) assumes the input scene is more realistic by adding localization of the hand, and (iii) avoids the use of heuristic initialization.

Model-based approaches The optimization of an objective function has been a mainstream approach to recover the hand configurations using a deformable 3D hand model. Initially, particle swarm optimization (PSO) was successfully applied in [1,35] to find a best fit model from a population of candidate solutions. In addition, gradient-based optimization was considered in [2, 36] to achieve faster convergence. While straightforward to implement, they iteratively update the initial pose parameters toward the local best solution. Hence, these methods may fail to track the hand when a prior estimate is inaccurate or to provide real-time performance. More recently, hybrid approaches [4, 37–39] have been introduced to recover loss of tracking

using a per-frame reinitializer. Although these methods avoid model drift, the system achieves low frame rates [40], requires clear fingertip detection [4], or is heavily dependent on random forest [38] which shows relatively lower performance.

2.1.2 Pose estimation during hand-object interaction

Previous approaches for hand pose estimation in hand-object interaction have mainly focused on model-based pose optimization [24, 41–44], similar to generative methods in hand tracking. Some of these approaches aim to track the interacting hands from a multi-camera input with a manual initialization of a hand and object [24, 41, 44]. Even though a dynamics simulator [42] and an ensemble of collaborative trackers [43] are presented to handle multiple object tracking from a single RGB-D sensor, all these methods assume that the accurate 3D models of the manipulated objects are given. In [45], tracking hands in interaction with unknown objects is proposed for model reconstruction. However, their use of temporal information from a model-based hand tracker may cause a model drift and limit the functional range of hand-object interaction. Although our method also focuses on interaction with unknown objects, we do not explicitly track the object but try to learn a discriminative cue for hand pose estimation.

Besides these studies, our work shares similarities with [23, 24] in terms of pose dependency on the shape of the object. However, the method in [24] does not explicitly extract shape information from the object. In [23], a set of synthetic hand templates is used to find a similar pose while searching the nearest neighbor. However, the small number of examples in the database and the search complexity of this method are the major bottlenecks. Even though our method shares a similar insight, the search complexity is remedied by reducing the search space based on the grasp type and the orientation of the hand. Recently, [46, 47] have used hand-crafted features for pose estimation while interacting with an object. They first segment the hand and object regions using RGB data, and then run either an SVM classifier [46] or pixel-wise part

classification [47] for hand pose estimation. However, these methods oversimplify the pose estimation problem by transferring a grasp template [46] or require a simple primitive as a manipulating object [47]. Even though a convolutional neural network framework is subsequently employed to replace the hand-crafted features [48], this approach only aims for grasp classification. In contrast, our method introduces a new ConvNet architecture effectively designed to handle the hand-object interaction for pose estimation that learns discriminative grasp features from both perspectives (*i.e.*, of both the hand and the object). The pipeline overview is presented in Figure 7.1.

2.2 Compressive Network

There has been a great effort to solve memory and computation efficiency problem of ConvNet in recent literature. The main stream of this category is focused on quantizing the network parameters into bins. In [49], the weights are first converted to the frequency domain using a discrete cosine transform and then quantized into hash buckets to group frequency parameters. By sharing a single value for the parameters in the same bucket, the size of model can be reduced. However, the compressed model may significantly lose accuracy [50] mainly because of hashing and training procedure. Also in [51], the weights are compressed using vector quantization techniques, but this method may result in the reduction of the predictive performance.

In contrast to these works, [52] tries to compress the network parameters using random matrix projection without dropping accuracy. In the training process, they try to learn the weight matrix which is split into a set of matrices based on the sparsity of the Johnson-Lindenstrauss (JL) transform. Our compressive ConvNet is somewhat close to this work in spirit of using the JL transform. However, we aim to embed more layers and go deeper in between fully connected layers while preserving the original ConvNet structure by compressing the embedded layers. We do not explicitly reduce the network parameters but do compress the inbuilt network layers, and therefore we keep the same amount of parameters after all. The embedded layers only appear in

the training process to learn implicative representations by backpropagation based on the fixed JL transform, instead of updating its variables [52]. In this way, we are able to effectively train our ConvNet model from random projection and efficiently add more layers into the ConvNet architecture, reducing the dimensionality of embedded layers.

3. 3D HAND MODEL

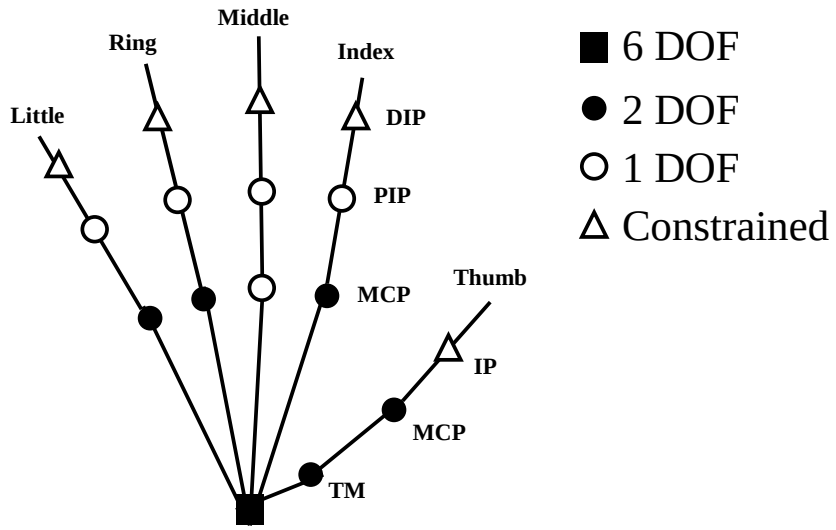


Figure 3.1. Our 21 DOFs hand model.

In this section, we first describe the 3D hand model and the procedure used to create a large library of hand poses. The pose library is annotated with labels we use for determining the hand pose from a depth map. We cluster the poses in the library to generate a set of pose exemplars useful for efficient nearest neighbor retrieval. Nearest neighbors are retrieved at runtime by evaluating the shape descriptor distance between the arriving depth data and simulated depth data of the pose exemplars.

3.1 Skeletal Hand Mesh

We statistically generate hand poses using a synthetic 3D hand model. The size of our synthetic hand model represents the median quartile of male hand sizes [53].

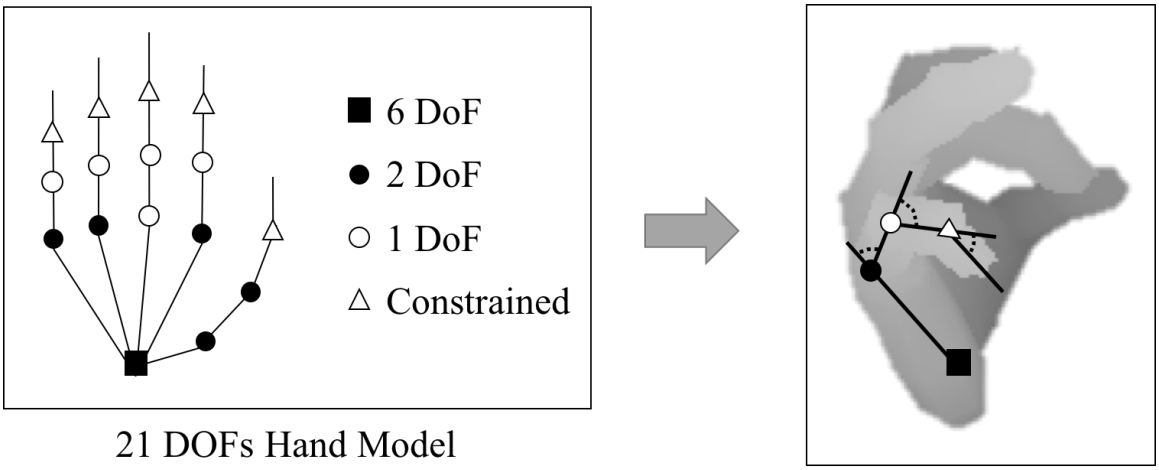


Figure 3.2. The joint angle parameters are a measure of angles between two bones.

Our hand model $\mathcal{M}$ is a compact Riemannian manifold without boundaries, which consists of 3,869 mesh vertices and 7,734 triangular faces. This model is explicitly scaled for individual subjects. We adopt a kinematic hand model with 21 degrees of freedom (DOF), $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\phi})$, as standard in hand pose estimation problems $\boldsymbol{\theta}$ denotes the set of 18 joint angle parameters and $\boldsymbol{\phi}$ is the set of 3 global translation parameters (x, y and z) of the hand.

3.2 Hand Constraints

We set limits on the configuration space of the pose parameters in order to automatically generate realistic hand poses using our 3D synthetic hand model. This ensures natural hand configurations mimicking real hand gestures. A comprehensive study on the functional ranges of joint movement is conducted in [54] and [55]. We employ the Type I and II constraints articulated in these papers on our kinematic hand model with 21 DOFs. The kinematic hand model and DOFs for each joint are shown in Figure 3.1. The acronyms DIP, PIP, MCP, IP and TM represent distal interphalangeal joint, proximal interphalangeal joint, metacarpophalangeal joint, in-

terphalangeal joint and trapeziometacarpal joint type, respectively. The joints with two degrees of freedom are a consequence of flexion and abduction motion.

Type I constraints set static ranges for tangible joint angle movement guided by the physical anatomy of the human hand. The angular ranges associated with the DOFs for each of the four fingers are listed in the first three rows of Table 3.1. Type II constraints are dynamic constraints dependent on Type I constraints. They are further subdivided into intra- and inter-finger constraints, representing the interdependence between joint angles in each finger and adjacent fingers, respectively. The intra-finger Type II joint angle constraints for all fingers, except the thumb, are listed in the last row of Table 3.1. The inter-finger Type II constraints limit the flexion of MCP joints in the little, ring, middle, and index fingers. For example, MCP-Flexion of the middle finger is dependent on MCP-Flexion of the index finger. Equation (3.1) iteratively governs the joint angle determination.

$$\theta_{MCP-F}^{Middle} = min(max(dmin, \theta_{MCP-F}^{Middle}), dmax), \tag{3.1}$$

where $dmin = max(\theta_{MCP-F}^{Index} - 25, \theta_{MCP-F}^{Ring} - 45, 0)$ and $dmax = min(\theta_{MCP-F}^{Index} + 54, \theta_{MCP-F}^{Ring} + 20, 90)$ are dynamic ranges as explained in [55]. We refer the reader to [55] for a complete list of inter-finger Type II constraints.

Table 3.1. Type I and II (intra-finger) constraints for index, middle, ring, and little finger.

Index	Middle	Ring	Little
$\theta_{MCP-Flexion}$			
$[0°, 90°]$	$[0°, 90°]$	$[0°, 90°]$	$[0°, 90°]$
$\theta_{MCP-Abduction/Adduction}$			
$[-15°, 15°]$	$0°$	$[-15°, 15°]$	$[-15°, 15°]$
θ_{PIP}			
$[0°, 110°]$	$[0°, 110°]$	$[0°, 110°]$	$[0°, 110°]$
θ_{DIP}			
$\frac{2}{3}\theta_{PIP}$	$\frac{2}{3}\theta_{PIP}$	$\frac{2}{3}\theta_{PIP}$	$\frac{2}{3}\theta_{PIP}$

We now list the constraints for the thumb. The Type I ranges for θ_{MCP-F} and $\theta_{MCP-Ab/Ad}$ are $[0, 60]$ and $[-5, 5]$ respectively, whereas the ranges for θ_{TM-F} and $\theta_{TM-Ab/Ad}$ are $[0, 60]$ and $[-15, 15]$ respectively. The intra-finger Type II constraint governing θ_{IP} in the thumb is:

$$\theta_{IP} = \frac{7}{5}\theta_{MCP-F}. \tag{3.2}$$

The inter-finger Type II constraints for the thumb are listed in [55].

4. LEARNING HAND FROM RECOMMENDATIONS OF SIMILAR POSES

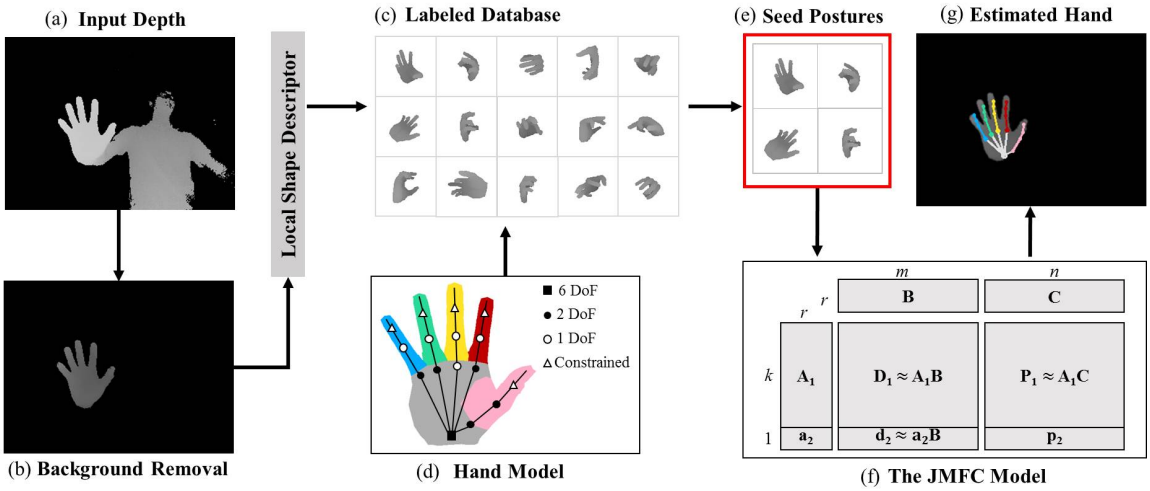


Figure 4.1. An overview of algorithm pipeline. Background noise in depth map is removed ((a) – (b)). We use a local shape descriptor to retrieve nearest neighbors from the labeled database of various hand configurations ((c) – (d)). The extracted neighbors serve as seed postures to a JMFC model, and unknown joint parameters are estimated using a matrix factorization and completion process ((e) – (g)).

4.1 Database Creation

In this section, we first describe the procedure used to create a large library of hand poses. The pose library is annotated with labels we use for determining the hand pose from a depth map. We cluster the poses in the library to generate a set of pose exemplars useful for efficient nearest neighbor retrieval. Nearest neighbors are retrieved at runtime by evaluating the shape descriptor distance between the arriving depth data and simulated depth data of the pose exemplars.

4.1.1 Synthetic data generation

Manually creating a library of hand poses using different individuals is a tedious task. Instead, we (1) impose constraints for joint configurations and finger movement as discussed in [54] and [55]; and (2) uniformly sample each of the 18 joint parameters in this restricted configuration space, in order to automatically simulate 118K realistic hand poses. These hand poses are effectively mesh modeled with corresponding skeletal information. In order to synthetically generate point clouds consistent with those visible to a depth camera under occlusion, we process these mesh models using a hidden point removal [56] strategy. Thus, each pose instance in the database is a mesh model with labels $(\boldsymbol{\theta}, \mathbf{s}, \mathbf{v})$, where $\mathbf{s}$ are the coordinates of the skeletal vertices and $\mathbf{v}$ are coordinates of the visible vertices from the viewpoint of a depth camera.

4.1.2 Pose exemplars and basis

In order to reduce redundancy of poses in the library, we cluster the poses and extract pose exemplars. Density based approaches can automatically detect arbitrary shaped clusters in high dimensional data. To identify pose clusters, we use a combination of two density-based clustering approaches, OPTICS [57] and DBSCAN [58], on the shape descriptor distance described below. The OPTICS algorithm does not explicitly generate clusters, but instead provides an ordering of all hand poses based on their similarities. The density parameters (minimum number of cluster members and maximum cluster radius) are estimated by investigating the output of OPTICS, and these parameters serve as input to DBSCAN. We then extract clusters using DBSCAN, and set the pose with minimum average distance to other cluster members to be the pose exemplar. We identify 1,030 exemplars among the 118K poses in the library, thus greatly improving the efficiency of nearest neighbor retrieval while maintaining accuracy (see Figure 4.3b).

Additionally, we evaluate $(\boldsymbol{\theta}, \mathbf{s}, \mathbf{v})$ for a set of 15 poses from the alphabets of American Sign Language (see Figure 4.2). A 15 dimensional vector, $\mathbf{d}$, is calculated

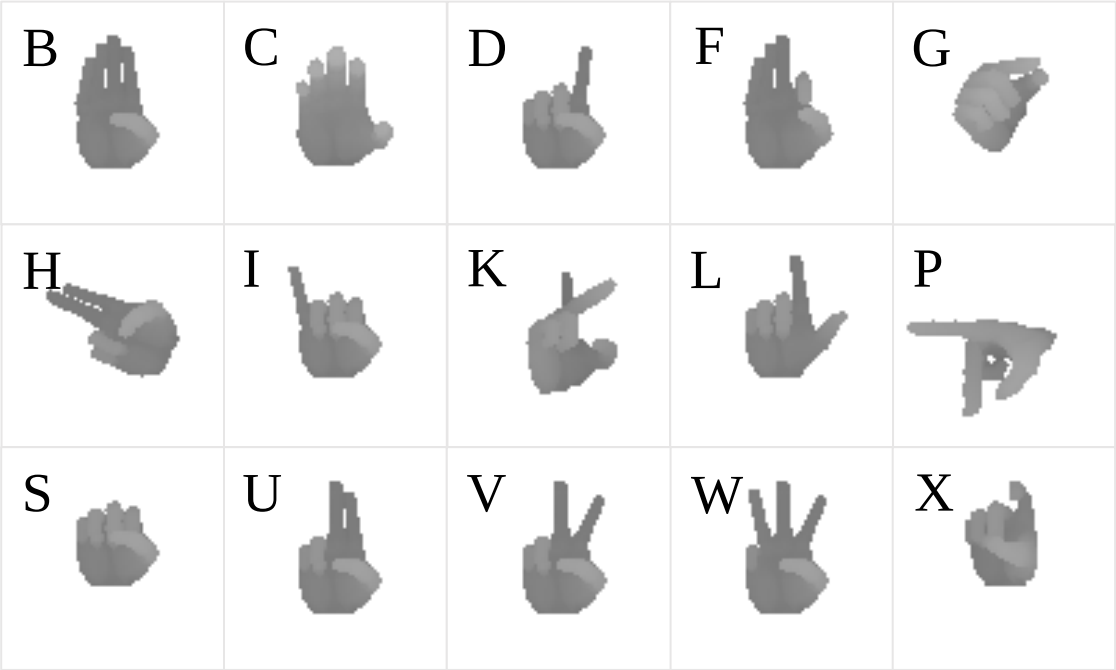


Figure 4.2. Illustration of 15 hand models used as basis adopted from American Sign Language.

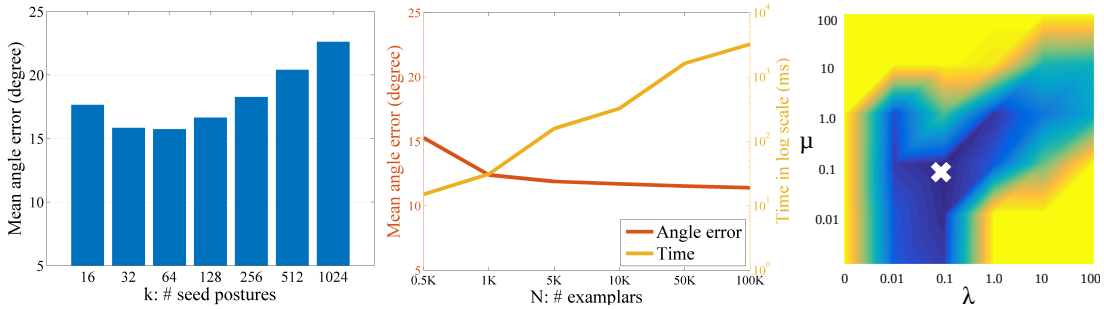
for each pose exemplar, wherein each element is the sum of all pairwise Euclidean distances between **v** of a pose in the basis and **v** of a pose exemplar. This vector serves as metadata for pose exemplars, akin to a feature vector for users in a recommender system.

4.1.3 Shape descriptor distance

We associate a local shape descriptor, **c**, to each pose exemplar. Nearest neighbor retrieval at runtime, proceeds by first determining the shape descriptor of the arriving point cloud, calculating its shape descriptor distance of all pose exemplars, and then selecting the nearest neighbors less than a threshold. The computation of the shape descriptor distance between two depth maps is described next.

We use the FAST feature point detectors on a depth map to identify corner points [59]. For each detected FAST feature point, a BRIEF descriptor [60] is computed, which encodes information about surrounding regions. Details of FAST and BRIEF computation are skipped for brevity. Correspondences are established between FAST feature points of two depth maps by iteratively (1) finding the pair with minimum Hamming distance (bitwise XOR operation) between their corresponding BRIEF descriptors, and (2) removing this matched pair for evaluating subsequent correspondences. The shape descriptor distance is then the average Hamming distance between BRIEF descriptors of all matched pairs of FAST feature points. Note that this distance varies with the hand's orientation, and hence outputs similarly oriented hand poses from the library as nearest neighbors. This feature is desirable in our approach as the in-plane rotation angles can then be robustly estimated using these nearest neighbors in the JMFC algorithm. Also, the descriptors for all pose exemplars are pre-computed to reduce computational overhead and only the descriptor for the input depth map is evaluated at runtime for nearest neighbor computation.

We get a set of 1,030 pose exemplars with labels $\mathbf{r} = (\boldsymbol{\theta}, \mathbf{s}, \mathbf{v}, \mathbf{d}, \mathbf{c})$ after the above pre-processing steps. Next we discuss the steps of our solution at runtime.



(a) (b) (c)

Figure 4.3. (a) Choice of nearest neighbor, k. Joint angle error is minimum for $32 < k < 64$. (b) Choice of number of exemplars, N. $N \approx 1000$ optimally trades off between accuracy and computational time. (c) Choice of regularization parameters, μ, λ. Joint angle error color coded with blue denoting low error and yellow denoting high error. Best choice is $\mu = 0.1, \lambda = 0.1$ indicated by $\times$.

4.2 Joint Matrix Factorization and Completion

The pipeline of our approach is demonstrated in Figure 7.1. The input depth is first processed to remove the background and only contains the depth pixels of the hand. The global parameters, $\boldsymbol{\phi}$ are directly estimated from this processed depth map. Next, the local shape descriptor of this depth map is evaluated and the nearest neighbors are retrieved from the labeled database using the shape descriptor distance. These neighbors serve as seed postures to the JMFC model and the joint angle parameters, $\boldsymbol{\theta}$, are estimated, followed by some final post-processing to output the tracked hand skeleton.

4.2.1 Model initialization

Background removal and estimation of $\boldsymbol{\phi}$: We use a simple heuristic to estimate the global translation parameters, $\boldsymbol{\phi}$. The depth map is pruned to exclude the background by only including points within the distance range of $(15, 50)$ cm to the depth camera, under the assumption that the hand lies in this region of interest. We determine the points corresponding to the hand in the depth map by considering the pixels enclosed in the longest continuous contour [61]. Extraneous noise in the detected blob is mitigated by using a median filter [62]. The translation parameters $\boldsymbol{\phi}$, are then set equal to the centroid of the remaining points in the depth map. Our experimental results suggest that this heuristic is fast and works well in practice. We propose to develop more sophisticated algorithms to estimate the translation parameters in future work.

Nearest neighbor retrieval and distance matrix: The k nearest neighbors [63] to depth map are calculated at each instant of time using the shape descriptor distance described in the previous section. The choice of parameter k is critical to the JMFC model. A small k compromises the robustness of the $\boldsymbol{\theta}$ estimation, whereas too large a k increases computational complexity making the model infeasible for real-

time applications. Hence, we determine the $\hat{k}$ nearest neighbors below a threshold for the shape descriptor distance and set k equal to:

$$k = min(max(32, \hat{k}), 64); \tag{4.1}$$

This is because k between $[32, 64]$ ensures fast and robust parameter estimation (see Figure 4.3a). The distance threshold for the shape descriptor distance is set at 15 for all our experiments. Next, we impute two matrices $\mathbf{P_1}$ and $\mathbf{D_1}$ of dimensions $k \times n$ and $k \times m$ respectively, with the known joint angles, $\boldsymbol{\theta}$ ($n = 18$) and Euclidean distance vector, $\mathbf{d}$ ($m = 15$) for the k indexed neighbors in the preprocessed database. We also calculate the 15-dimensional distance vector, $\mathbf{d_2}$, as the sum of all pairwise Euclidean distances between $\mathbf{v}$ of each pose in the basis and points on the refined depth map. Our algorithm for estimating the joint angle parameters, $\mathbf{p_2}$, using $\mathbf{P_1}, \mathbf{D_1}, \mathbf{d_2}$ independently for each frame is discussed next.

4.2.2 The JMFC model

As discussed previously, we use a joint matrix factorization and completion (JMFC) approach to estimate the unknown joint angles for a given depth map. Our rationale for using the JMFC model in analogy to a recommender system described in parenthesis is as follows: We have a matrix $\mathbf{P_1}$ with joint angles (known ratings) for a set of similar poses to the input depth (like-minded users to a new user). Additionally, matrix $\mathbf{D_1}$ contains auxiliary information about nearest neighbor poses relative to a basis (metadata about like-minded users) and vector $\mathbf{d_2}$ which contains the same auxiliary information about the new pose (metadata about new user) whose parameters $\mathbf{p_2}$ (unknown personalized ratings) are to be estimated. Our task is then to uncover the latent factors, $\mathbf{a_2}$ governing the parameters, $\mathbf{p_2}$ by determining the latent factors for (1) nearest neighbor poses, $\mathbf{A_1}$ (2) known joint angles, $\mathbf{C}$ and (3) known distances to basis models, $\mathbf{B}$. Mathematically, we find a factorization of matrices $\mathbf{P_1}, \mathbf{D_1}$ and vector $\mathbf{d_2}$ in terms of the latent factors $\mathbf{A_1}, \mathbf{a_2}, \mathbf{B}, \mathbf{C}$, and use these information to

impute the unknown vector, $\mathbf{p_2}$. In other words, we simply find low rank approximations of known matrices in order to estimate the unknown pose parameters. Using the above intuition, our JMFC model is succinctly expressed as:

$$\underset{\mathbf{A}_1,\mathbf{a}_2,\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{D}_1 \\ \mathbf{d}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{a}_2 \end{bmatrix} \mathbf{B} \right\|_F^2 + \frac{\mu}{2} \left\| \mathbf{P}_1 - \mathbf{A}_1\mathbf{C} \right\|_F^2 . \tag{4.2}$$

where $\mathbf{B}$ and $\mathbf{C}$ are r-dimensional latent factors for the distances ($\mathbf{D}$) and joint angle parameters ($\boldsymbol{\theta}$), respectively; $\mathbf{A_1}$ and $\mathbf{a_2}$ are the r-dimensional latent factors for the k-nearest neighbors and input depth map respectively, and μ is regularization parameter which trades off the losses due to matrix factorization and accuracy of matrix completion. $\mathbf{P_1}$ decomposes as a product of latent factors $\mathbf{A_1}$ and $\mathbf{C}$, ($\mathbf{P_1} \approx \mathbf{A_1C}$), $\mathbf{D_1}$ decomposes as a product of latent factors $\mathbf{A_1}$ and $\mathbf{B}$, ($\mathbf{D_1} \approx \mathbf{A_1B}$), whereas the row $\mathbf{d_2}$ decomposes as $\mathbf{a_2B}$ (see Figure 7.1f). To prevent overfitting, we add a regularization term, λ to the Frobenius norms of $\mathbf{A_1}, \mathbf{a_2}, \mathbf{B}$ and $\mathbf{C}$ which gives us the following minimization problem:

$$\begin{aligned} \underset{\mathbf{A}_1,\mathbf{a}_2,\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{D}_1 \\ \mathbf{d}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{a}_2 \end{bmatrix} \mathbf{B} \right\|_F^2 + \frac{\mu}{2} \left\| \mathbf{P}_1 - \mathbf{A}_1\mathbf{C} \right\|_F^2 \\ + \frac{\lambda}{2} \left(\|\mathbf{A}_1\|_F^2 + \|\mathbf{a}_2\|_F^2 + \|\mathbf{B}\|_F^2 + \|\mathbf{C}\|_F^2 \right) . \end{aligned} \tag{4.3}$$

We use the Alternative Least Squares (ALS) [64] to solve the above minimization problem, and it is summarized in Algorithm 1. Additional details about the objective function and the derivation of the algorithm are discussed in subsection 4.2.3.

The parameters λ and μ are empirically set to 0.1 and 0.1, respectively (see Figure 4.3c). The rank r of latent factors is set to 5 as it optimally trades off between accuracy and efficiency. The ALS procedure in Algorithm 1 repeats until the difference between output values of equation 5.7 for subsequent iterations is less than 10^{-6} or the number of iterations exceed 600. As a final step, the pose parameters $\mathbf{p_2}$ are estimated as

$\mathbf{p_2} \approx \mathbf{a_2C}$ and further refined by imposing the pose constraints mentioned in Section 3.1. This ensures that the final solutions comply with kinematically feasible hand configurations.

4.2.3 Algorithmic details

In this section, we present mathematical elements of the JMFC model which factorizes the distance matrix $\mathbf{D}$ in order to complete the parameter matrix $\mathbf{P}$. We briefly review the meaning of symbols used in the main manuscript. We first retrieve k similar hand poses to the input depth map from the database using the local shape descriptor, and additionally, m hand models serve as a basis of prototype poses. Using these information, we compute the distances between the hand models in basis and the k hand postures and set these values in matrix $\mathbf{D}_1$. Also, vector $\mathbf{d}_2$ is evaluated as the distance between the models in basis and an input depth map. Next, matrix $\mathbf{P}_1$ is imputed with joint angle parameters of the k hand poses. Our goal is to estimate the unknown parameters of the input depth map, by solving the optimization equation:

$$\begin{aligned} \underset{\mathbf{A}_1,\mathbf{a}_2,\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{D}_1 \\ \mathbf{d}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{a}_2 \end{bmatrix} \mathbf{B} \right\|_F^2 + \frac{\mu}{2} \left\| \mathbf{P}_1 - \mathbf{A}_1\mathbf{C} \right\|_F^2 \\ + \frac{\lambda}{2} \left(\|\mathbf{A}_1\|_F^2 + \|\mathbf{a}_2\|_F^2 + \|\mathbf{B}\|_F^2 + \|\mathbf{C}\|_F^2 \right). \end{aligned} \tag{4.4}$$

where λ, μ are regularization terms. Figure 4.4 shows the matrix framework of the JMFC model.

We use the Alternative Least Squares (ALS) to solve the minimization problem as follows: Let the argmin of equation (4.4) be f. Then, the gradient of f with respect to $\mathbf{A}_1$ is:

$$\frac{\partial f}{\partial \mathbf{A}_1} = \mathbf{A}_1 \left(\mathbf{B}\mathbf{B}^T + \mu \mathbf{C}\mathbf{C}^T + \lambda \mathbf{I} \right) - \left(\mathbf{D}_1\mathbf{B}^T + \mu \mathbf{P}_1\mathbf{C}^T \right) \tag{4.5}$$

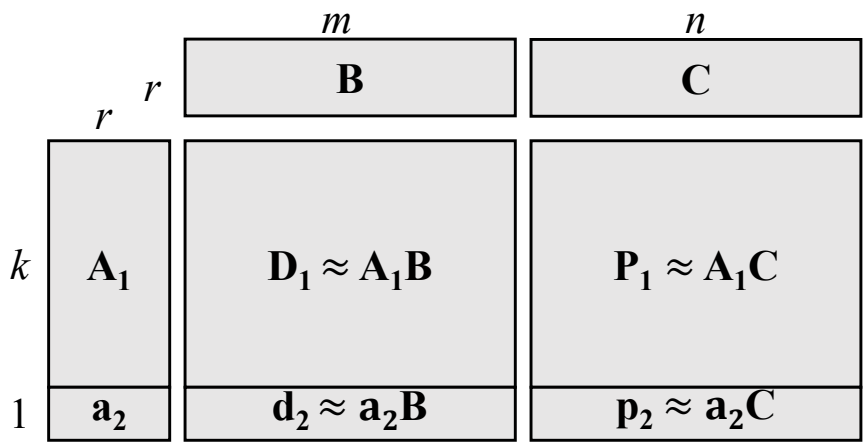


Figure 4.4. The matrix framework of JMFC.

Equating equation (4.5) to zero outputs the optimal solution of (4.4) with respect to $\mathbf{A_1}$, and is given by

$$\mathbf{A}_1 = \left(\mathbf{D}_1\mathbf{B}^T + \mu\mathbf{P}_1\mathbf{C}^T\right)\left(\mathbf{B}\mathbf{B}^T + \mu\mathbf{C}\mathbf{C}^T + \lambda\mathbf{I}\right)^{-1}. \tag{4.6}$$

Similarly, we can obtain the optimal solution of (4.4) with respect to $\mathbf{a}_2$, $\mathbf{B}$ and $\mathbf{C}$ are:

$$\mathbf{a}_2 = \left(\mathbf{d}_2\mathbf{B}^T\right)\left(\mathbf{B}\mathbf{B}^T + \lambda\mathbf{I}\right)^{-1} \tag{4.7}$$

$$\mathbf{B} = \left(\mathbf{A}_1^T\mathbf{A}_1 + \mathbf{a}_2^T\mathbf{a}_2 + \lambda\mathbf{I}\right)^{-1}\left(\mathbf{A}_1^T\mathbf{D}_1 + \mathbf{a}_2^T\mathbf{d}_2\right) \tag{4.8}$$

$$\mathbf{C} = \left(\mu\mathbf{A}_1^T\mathbf{A}_1 + \lambda\mathbf{I}\right)^{-1}\left(\mu\mathbf{A}_1^T\mathbf{P}_1\right). \tag{4.9}$$

We iteratively calculate (4.6), (4.7), (4.8) and (4.9) until f converges. Once f converges, we can obtain parameters of the input depth map using the equation $\mathbf{p_2} = \mathbf{a}_2\mathbf{C}$. Hence, the algorithm of our joint model is given by Algorithm 1.

The latent representations (matrix $\mathbf{A}, \mathbf{B}, \mathbf{C}$) are randomly initialized by uniformly sampling between 0 and 1. We contend the accuracy of JMFC will improve when initialized with PCA, albeit with a computational overhead. Instead, we propose to

Algorithm 1: The JMFC algorithm

Input: $\mathbf{D}_1$, $\mathbf{d}_2$, $\mathbf{P}_1, \mu, \lambda$
Initialize: $\mathbf{A}_1$, $\mathbf{a}_2$, $\mathbf{B}$, $\mathbf{C}$
while *stopping criterion not met* **do**
  $\mathbf{A}_1 \leftarrow \left(\mathbf{D}_1\mathbf{B}^T + \mu\mathbf{P}_1\mathbf{C}^T\right)\left(\mathbf{B}\mathbf{B}^T + \mu\mathbf{C}\mathbf{C}^T + \lambda\mathbf{I}\right)^{-1}$
  $\mathbf{a}_2 \leftarrow \left(\mathbf{d}_2\mathbf{B}^T\right)\left(\mathbf{B}\mathbf{B}^T + \lambda\mathbf{I}\right)^{-1}$
  $\mathbf{B} \leftarrow \left(\mathbf{A}_1^T\mathbf{A}_1 + \mathbf{a}_2^T\mathbf{a}_2 + \lambda\mathbf{I}\right)^{-1}\left(\mathbf{A}_1^T\mathbf{D}_1 + \mathbf{a}_2^T\mathbf{d}_2\right)$
  $\mathbf{C} \leftarrow \left(\mu\mathbf{A}_1^T\mathbf{A}_1 + \lambda\mathbf{I}\right)^{-1}\left(\mu\mathbf{A}_1^T\mathbf{P}_1\right)$
$\mathbf{p}_2 \leftarrow \mathbf{a}_2\mathbf{C}$

use improved initialization methods such as Random Acol [65] to improve JMFC in future work. The difference between PCA and our method is that our method simultaneously uses **D1** and $\mathbf{d_2}$ to obtain **B** (as opposed to PCA using only $\mathbf{D_1}$). Because the effect of $\mathbf{d_2}$ on B is small as $\mathbf{d_2} << \mathbf{D_1}$, the obtained solution is comparably robust to PCA. The orthonormal constraint imposed by PCA is unnecessary because, it only leads to scaling the rows of $\mathbf{A_1}$ not affecting the final outcome. Meanwhile, our method is much faster than PCA because only a few iterations (< 100) are required for convergence.

4.3 Experiments

In this section, we evaluate our approach for synthetic hand poses as viewed from a depth camera and real depth data. We perform quantitative analysis on a synthetic dataset of hand poses generated by uniformly sampling in the constrained hand configuration space. This ensures adequate coverage, and hence an unbiased evaluation of our approach. Further, we perform the same quantitative analysis using realistic hand pose data captured from a commercial depth camera. The prime difference between real and synthetic data is the presence of noise in real depth streams. We first describe the datasets and set baselines before proceeding to the performance evaluation. All our experiments are performed on Intel Xeon E3-1240 CPU with 16GBs RAM.

4.3.1 Datasets

We generate a synthetic dataset of 1,000 randomized hand postures following the procedure in [38] as follows. The 18 joint angle parameters and 3 global translation parameters are uniformly sampled in the constrained hand configuration space to generate a synthetic hand configuration, and the depth map of this pose is rendered within the view frustum. All constraints for this configuration space simulating realistic hand poses are listed in the supplementary material. Consequently, we get varied poses with corresponding ground truth. Note that we can use this approach to evaluate performance because our algorithm does not depend on temporal information and re-initializes at every frame.

We capture depth streams using the SoftKinetic's DepthSense DS325 and use this information for evaluating our algorithm on real datasets. Four sequences are captured, each from a different person, and each sequence contains 300 frames ($\approx$ 10 seconds) of hand movement. The ground truth is first roughly initialized using FORTH [1] with 256 particles and 75 generations, followed by manual refinement. Even with the large number of particles and generations, FORTH contains subtle errors in the hand pose which we manually remove.

Furthermore, we evaluate ours against two state-of-the-art approaches [1, 33] on the large and challenging dataset released with [33] in order to demonstrate that our method is applicable in a general setting. The dataset consists of 76,500 depth images captured from 9 subjects, using a Intel's Creative Senz3D camera compatible with DepthSense camera resolution. The depth maps comprise of 17 hand gestures under large viewpoint changes and span diverse finger articulations and hand configurations.

4.3.2 Evaluation metrics and baselines

Metric Four standard metrics are used for our quantitative evaluation: (1) *individual joint angle error* averaged over all frames, (2) *individual joint distance error* averaged over all frames, (3) *proportion of correct frames* as a function of maximum

allowed joint angle error, and (4) *proportion of correct frames* as a function of maximum allowed joint distance error described in [30, 38]. Metrics 1 and 2 indicate the estimation errors for individual joints whereas metrics 3 and 4 are indicative of overall robustness of an algorithm.

Baselines We demonstrate the efficacy of our overall algorithm by comparing our method to the following baselines: (a) *NN-only* wherein we estimate pose parameters using a single nearest neighbor among the pose exemplars and (b) *JMFC-full* wherein all 1,030 pose exemplars are used for pose estimation (nearest neighbors are not retrieved). We compare our algorithm to real-time implementation of FORTH on the realistic datasets by setting the parameters equal to 64 particles and 25 generations.

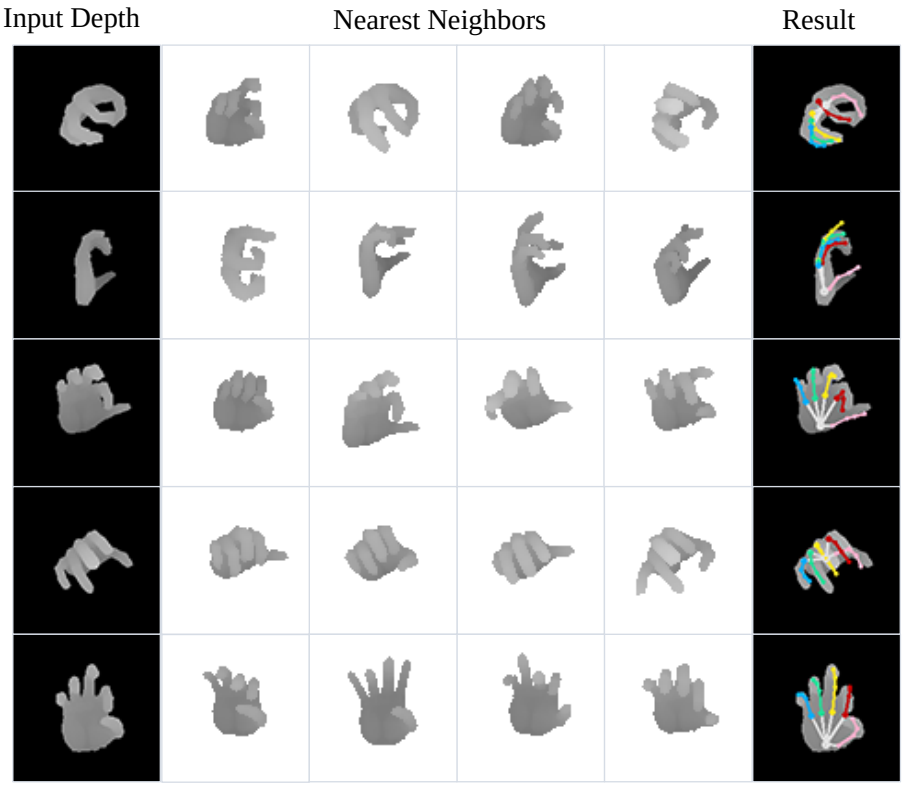


Figure 4.5. Qualitative analysis on the synthetic dataset. Left: randomly generated input poses. Middle: selected nearest neighbors (including outliers) from our pose exemplars. Right: the estimated hand pose.

4.3.3 Experiments on synthetic dataset

Quantitative Analysis We evaluated our approach on the generated synthetic poses. Figure 5.5 shows the quantitative evaluation of our algorithm in terms of the accuracy metrics, relative to the two baselines.

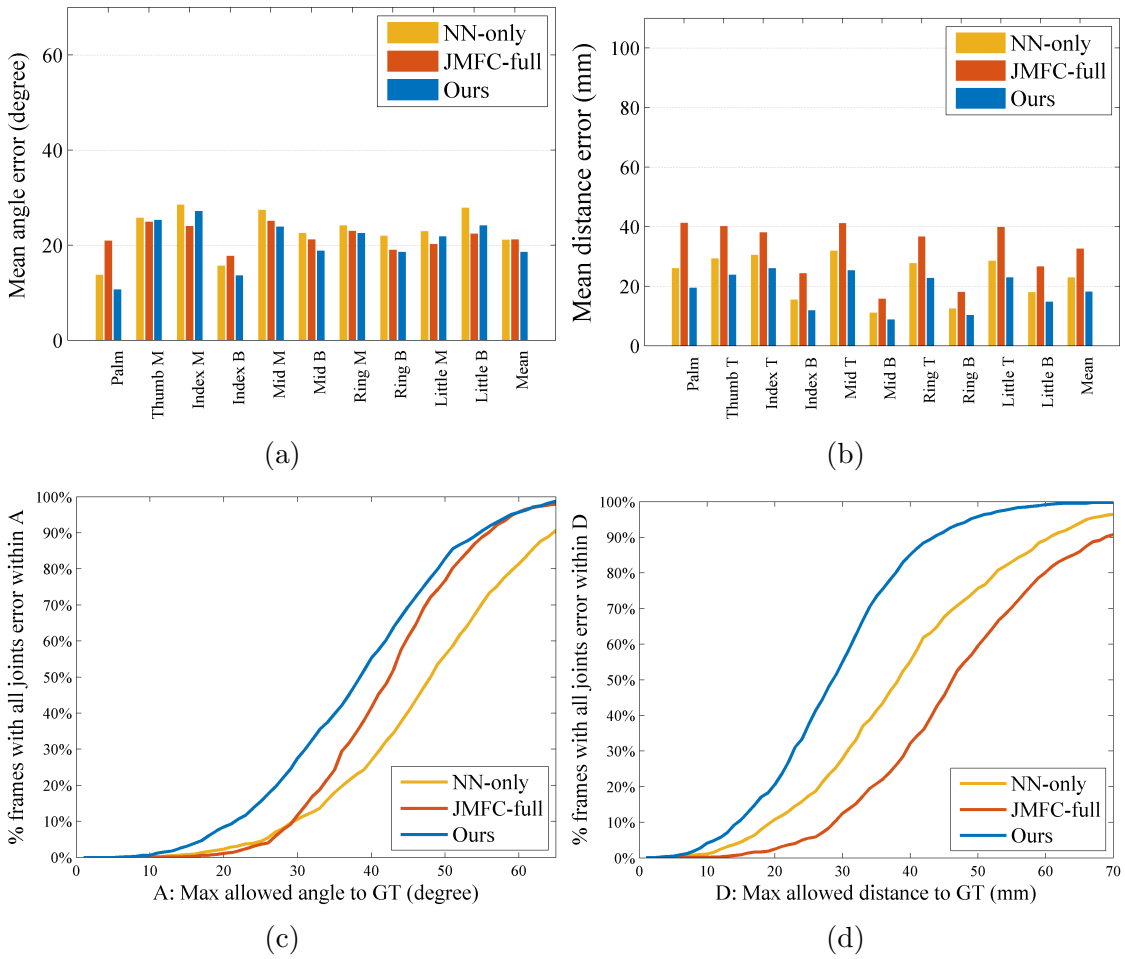


Figure 4.6. Quantitative analysis on the synthetic dataset with respect to four metrics, relative to baselines (T: tip, M: mid, and B:base). (a) The average joint angle error in degrees. (b) The average joint distance error in millimeters. (c) and (d) show the proportion of depth maps (y-axis) with joint angle and distance error less than a threshold (x-axis).

Figure 4.6a and 4.6b show the average error of estimated joint angles and distances relative to the ground truth. Our algorithm performs better than the two baselines

with respect to both metrics. In Figure 4.6a we see that the errors in joint angles for JMFC-full are generally less than NN-only, except for the palm angle, meaning that the joint angles are robustly estimated by the JMFC model even in the presence of extraneous poses not similar to the input depth map. However, the high error in palm angle for JMFC-full makes the estimated pose very different from the ground truth. This error in JMFC-full propagates to other joints leading to large distance errors relative to NN-only as seen in Figure 4.6b. Figure 4.6c and 4.6d show that our algorithm performs better than NN-only and JMFC-full at all thresholds for maximum allowed joint angle and distance error. The proportion of correctly identified frames is about 90 percent when the threshold for the joint distance error is set to 40 mm as seen in Figure 4.6d. The comparative result can be found in [38] (figure 9c). Although we do not have access to their datasets, this qualitative comparison to their state-of-the-art method under the same experimental settings is very promising. Also unlike their approach, we do this without considering temporal information and without a GPU.

4.3.4 Experiments on realisitic dataset

We perform a qualitative analysis of our approach in Figure 4.5. The central sub-figures indicate the nearest neighbors retrieved from the pose library. We observe that even though some nearest neighbors share very little similarity to the input depth map, the final solution is robustly estimated. This robustness against outliers is attributed to the vector $\mathbf{d_2}$ (the vector of distances to basis models) in the JMFC model, which implicitly mitigates the effect of faulty nearest neighbors. Intuitively, the incorrect pose parameter values of these faulty neighbors are weighed less in the collaborative assignment of pose parameters to the unknown pose.

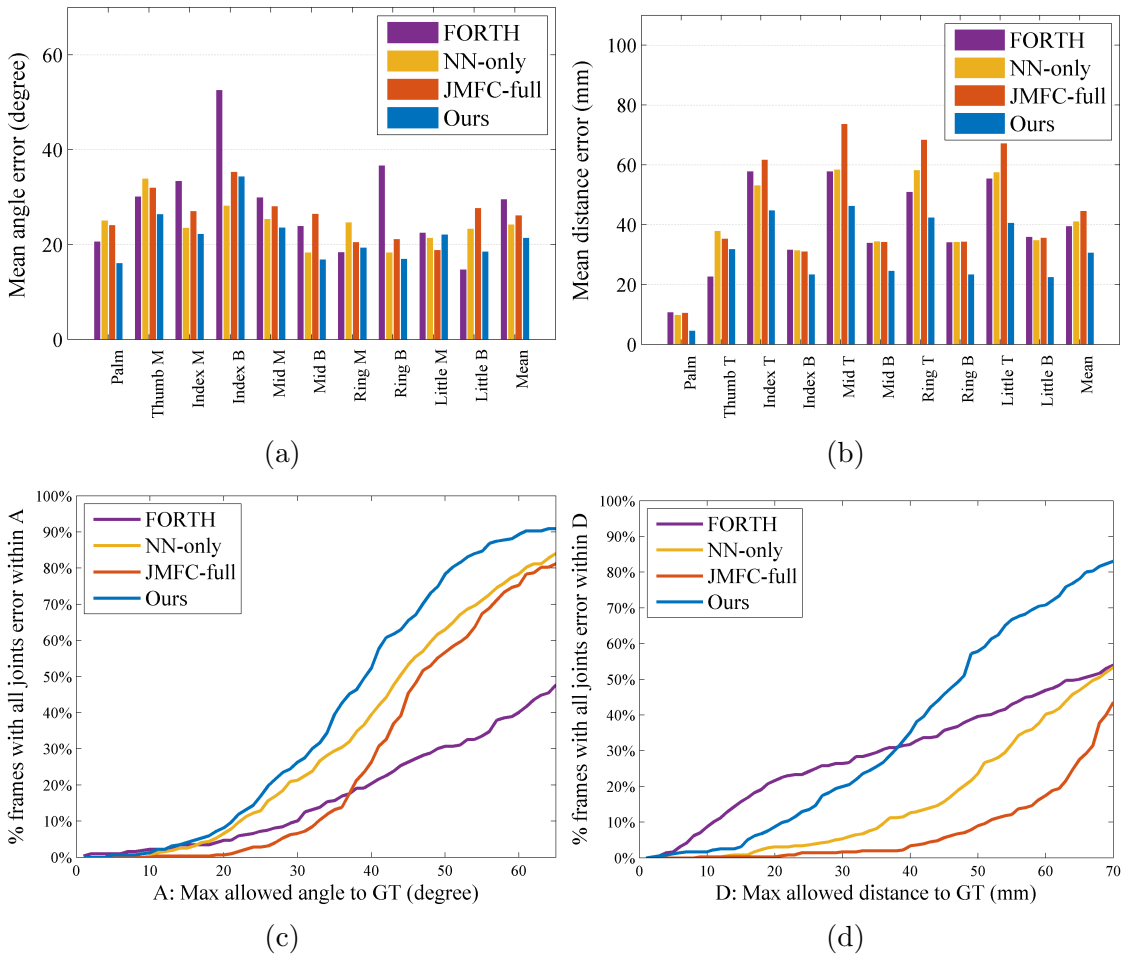


Figure 4.7. Quantitative analysis on the realistic dataset with respect to four metrics, relative to baselines (T: tip, M: mid, and B:base). (a) The average joint angle error in degrees. (b) The average joint distance error in millimeters. (c) and (d) show the proportion of depth maps (y-axis) with joint angle and distance error less than a threshold (x-axis).

Quantitative Analysis We evaluate our approach on the generated realistic dataset affected by noise with respect to three baselines, NN-only, JMFC-full and FORTH[1].

Figure 4.7a and 4.7b show the average error of estimated joint angles and distances relative to the manually refined ground truth over all four sequences. We observe

[1]The algorithm in [1] is reimplemented using our depth camera.

that overall our method is superior to all baselines with respect to all four error metrics. Unlike FORTH, our model does not need any temporal information, and hence, avoids errors accumulating over time. It is also interesting to note that noise in real datasets confounds nearest neighbor estimation leading to poorer performance than synthetic datasets. One solution to reduce the effect of noise is to use training for accurately generating pose hypothesis as done in [38] instead of using nearest neighbors, a possible direction for future work.

We observe that the performance of our algorithm to estimate joint angles on realistic dataset (Figure 4.7c) is very similar to the synthetic dataset. However, the performance as measured by error metric (d) deteriorates relative to synthetic dataset (Figure 4.7d). This hints at a compounded effect of poor nearest neighbor estimation and incorrect estimation of global translation parameters. The latter problem, however, is easily solvable by replacing our heuristic based method by methods implemented in [38, 66] for accurate region of interest detection. However, the thrust of our contribution is the JMFC model for joint angle estimation which is effectively validated.

Qualitative Analysis Figure 4.8 qualitatively evaluates our approach against the baselines. All depth maps are centered for effective visualization. The top column shows the input depth map and each row corresponds a baseline method. We observe that our approach is robust to the various types of hand configurations under occlusion.

The average frame rate of our complete algorithm for hand pose estimation on the realistic datasets is $\approx$ 29Hz, and hence applicable in a real-time environment. In comparison, our implementation of FORTH with NVIDIA Quadro K4000 GPU resulted in an average frame rate of 16Hz. Additionally, we do not require temporal information as our algorithm proceeds on a per frame basis.

Quantitative Analysis on Public Dataset We compare our algorithm on the dataset of [33] with FORTH and the *Holistic*, *Hierarchical* and *HPR-2D+Rot* regression methods proposed in [33]. We indirectly compare our method with [67] as

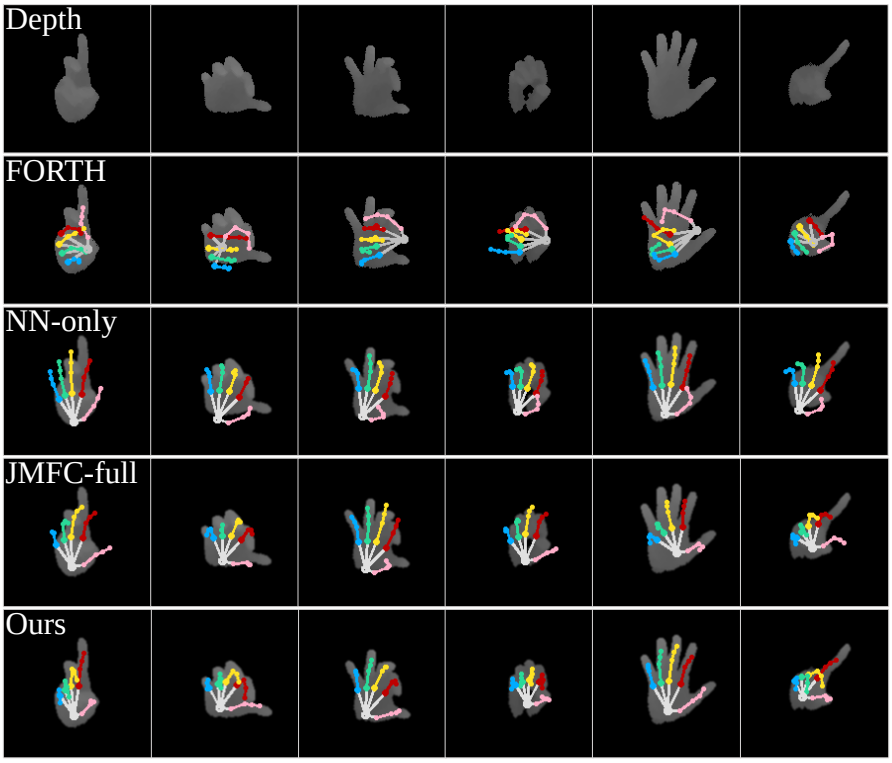


Figure 4.8. Qualitative comparison of our method with 3 baselines: FORTH, NN-only, JMFC-full in that order.

Hierarchical pose regression [33] has been shown to be better than [67] in [33] and with [37] which is similar in spirit to HPR-2D+Rot [33]. Figure 4.9 displays the proportion of depth maps (y-axis) with joint distance error less than a threshold (x-axis) for the 5 methods[2]. We see that our approach achieves better accuracy than FORTH and comparable performance to *Hierarchical* pose regression method of [33]. Our method has the highest fraction of frames with maximum allowed distance to ground truth in the $[0, 15]$ mm and $[40, 80]$ mm domain, validating that our approach is overall more robust to finger articulations and applicable to hand pose estimation in a general setting.

[2]Performance of *Holistic*, *Hierarchical* and *HPR-2D+Rot* methods are estimated from figure 5a in [33] which displays the same error metric.

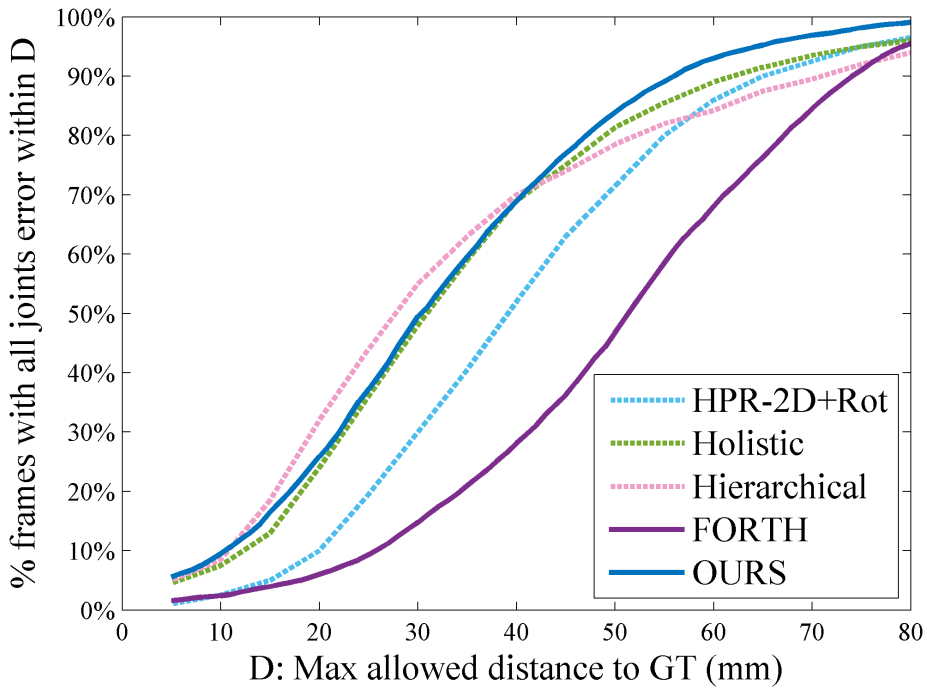


Figure 4.9. Quantitative comparison of our method with [1, 33] on a public dataset released with [33] with respect to proportion of depth maps (y-axis) with joint distance error less than a threshold (x-axis).

4.4 Conclusion and Future Work

In this chapter we present a novel approach for the hand pose estimation problem based on a joint matrix factorization and completion model. We present strong evidence of the applicability of our approach for hand tracking in a real-time environment. Although we demonstrate the efficacy of our approach for estimating joint angle parameters of the human hand, the overall idea is also applicable to the human pose estimation problem. More generally, our approach conclusively validates that advances in collaborative filtering approaches for recommender systems can be effectively synergized with pose estimation and tracking problems. This opens up several avenues for future work. One promising direction is the use of nuclear norm regularization instead of the Frobenius norm in the JMFC objective function to get low rank factors. We also wish to explore techniques for determining the best basis

and effectively integrating RGB information in our future work. Overall, we believe our JMFC model based approach for hand pose estimation opens up new avenues for real-time solutions in computer vision.

5. LEARNING HAND FROM LOW-DIMENSIONAL VISUAL REPRESENTATIONS

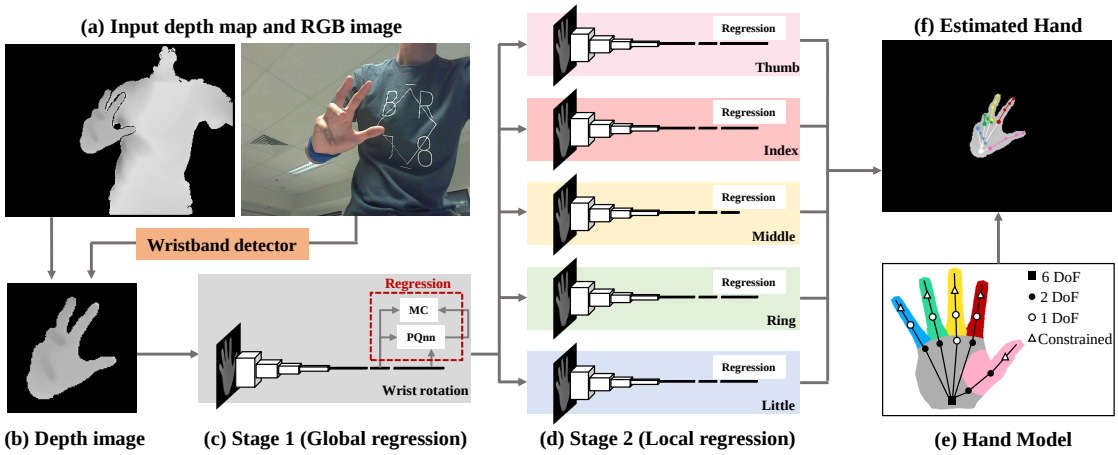


Figure 5.1. An overview of the proposed approach. In a real-setting, we extract region of interest using depth map and RGB-based wrist band detector (a)-(b). The obtained depth image is fed into a ConvNet which outputs an activation feature. This activation feature synchronizes with other features in a population database using our matrix completion method and the global pose parameters are estimated(c). Based on this global pose initialization, we estimate the rest of the local joint parameters in the same recursive manner (d). The final hand pose is displayed on a multimedia screen (f).

5.1 Preliminaries

In this section, we briefly describe our 3D hand model and discuss our method to extract the region of interest corresponding to the hand which serves as input to our hand pose estimation method.

Hand model We use a kinematic hand model with 21 degrees of freedom (DOF), represented as $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\phi})$, as standard in hand pose estimation literature (see Fig-

ure 7.1e). $\boldsymbol{\theta}$ denotes the set of 18 joint angle parameters and $\boldsymbol{\phi}$ is the set of 3 global translation parameters (x, y and z) of the hand.

Region of interest extraction Unlike the body, the hand occupies a relatively small region in the overall depth image obtained from the 3D depth camera. Hence, we preprocess the depth image to only include values that lie in the range of [50, 500] mm under the premise that the hand lies within this range. We then do a largest blob detection as an indicator of the hand segment, followed by median filtering for noise removal, depth normalization so that values lie in the range [0, 255], and finally resize the image while maintaining the aspect ratio to obtain a 64×64 depth image.The centroid of the blob in the original image marks the global position, $\boldsymbol{\phi}$. In more extreme settings (for ranges upto 2000 mm), we use a colored wristband as a simple indicator of the hand region as done in [4, 36]. Even in a close range scenario, the wristband helps removing extraneous pixels like those below the wrist, leading to better performance.

5.2 Dimensionality Reduction using Deep Learning

It is well known that the activation features from the intermediate hidden layers of a ConvNet can be re-purposed across domains [68, 69]. This suggests that the activation feature of a depth image itself contains discriminative cues about its overall shape and form of the hand, in the context of hand pose estimation. The thrust of our approach relies on the contention that a pool of nearby activation features is better able to reach consensus about the hand's orientation and shape. This introduces two challenges (1) The activation features in the population should conform to the activation features obtained from different individuals in diverse real settings. Additionally, they should be accurately annotated with their ground truth labels (joint angles or positions) (2) The population of activation features must be large enough to provide robust nearest neighbors to any input activation feature, however should be efficiently retrievable and consume limited memory. A straightforward approach

is to directly use the depth data gathered from 3D sensors to train a ConvNet and store the corresponding activation features. However, creating a such database of hand poses to cover full range of hand articulations with accurate ground truth labels is a tedious task. In this section, we describe how we generate such a population of activation features from synthetic dataset, reflective of real data.

5.2.1 Synthetic population of realistic hand poses

We generate synthetic depth maps by first imposing static (*e.g.*, range of motion, joint length, location) and dynamic (*e.g.*, among joints and fingers) constraints listed in [70][1]. We then uniformly sample each of the 18 joint parameters in this restricted configuration space. This ensures that the depth maps are reflective of real poses covering a wide range of hand articulations. However, data from 3D sensors are prone to noise, distortion and additional artifacts. Hence, we add gaussian noise $N(0, \sigma^2)$ to the synthetic depth maps wherein the standard deviation σ is chosen from a range of [0, 2] by uniform sampling. We empirically validated the inclusion of Gaussian noise by testing the classification accuracy of the global rotation angles in the correct bin (total 144) for a real hand depth sequence captured using SoftKinect DS325 (2500 frames). The drastic improvement of classification accuracy in Table 5.1 highlights that our noise model if fairly reflective of real sensor noise. Our training dataset covers an entire camera viewpoint (coverage due to the 3 wrist rotation angles $\boldsymbol{\theta^W} = \{\theta_r^W, \theta_p^W, \theta_y^W\}$, where $\theta_r^W \in [-45, 135], \theta_p^W \in [-45, 180], \theta_y^W \in [-45, 180]$). Our large coverage ensures the robustness our method to camera viewpoint changes and not restricted to near frontal poses. We discuss the size of the synthetic population in context to ConvNets in the next subsection.

[1]The availability of rigourous constraints in terms of joint angles is the main reason we choose angles over joint position in our hand pose method.

Table 5.1. The classification accuracy for the global rotation.

Gaussian noise	Classification accuracy
Yes	77.00%
No	44.88%

Table 5.2. Overall architecture of our convolutional networks. (Conv: convolutional layer, Pmax: max pooling layer, ReLU: rectified linear units layer, Smax: softmax layer)

	Layers	# Kernels	Filter size	Stride	Pad
1	Conv	16	5×5×1	1	2
2	Pmax			2	0
3	ReLU				
4	Conv	32	5×5×16	1	2
5	ReLU				
6	Pmax			2	0
7	Conv	32	5×5×32	1	2
8	ReLU				
9	Pmax			2	0
10	Conv	64	5×5×32	1	2
11	ReLU				
12	Pmax			2	0
13	Conv	128	4×4×64	1	0
14	ReLU				
15	Conv	32	1×1×128	1	0
16	ReLU				
17	Conv	144	1×1×32	1	0
18	Smax				

5.2.2 Activation features using ConvNet

ConvNet and its variants are the current state of the art architecture for numerous classification tasks such as object detection, scene recognition, texture recognition and fine grained classification. However, hand tracking is effectively a regression task. Our preliminary experiments with deep learning indicated that ConvNets do not adapt to regression as well as they do for classification as shown in Figure 7.7d. Consequently, our activation features are computed using ConvNet for classification instead of regression. These activation features feed into our matrix completion method which implicitly regresses and outputs the estimated joint angle parameters. The classification of joint angles into quantized bins, and hence, calculation of the activation feature in the penultimate layer, is performed by the ConvNet architecture displayed in Table **??**. Observe that the penultimate layer corresponding to the activation feature is a 32 dimensional vector of the sixth convolutional layer so as to reduce memory usage in storing the population of activation features. We use these activation features in a collaborative spatio-temporal fashion to estimate pose parameters using efficient nearest neighbor search and out novel matrix completion model.

There are two extremal strategies for quantization. The first strategy is to quantize each joint angle separately for a total of 21 ConvNets. However, this is inefficient both in terms of speed and memory. The second is to use an all-in-one strategy to train all joint angle parameters simultaneously. However, it would be impossible to learn an accurate classifier in such a high dimensional space even with a nominal number of bins. Hence, we use a 2-stage hierarchical strategy which satisfactorily balances computational time, memory requirement and classification accuracy.

In *Stage 1* the activation feature associated with the 3 global rotation angles, $\boldsymbol{\theta^W} = \{\theta_r^W, \theta_p^W, \theta_y^W\}$ is calculated and input into the matrix completion method along with a pool of nearest neighbors. The output of the matrix completion method is used to infer the correct rotation bin. For each rotation bin, five ConvNets are trained to output the activation feature associated with each of the five fingers. The

Table 5.3. Accuracy and memory comparison of global pose initialization.

Model	Accuracy	Memory	Settings
RF	57.45 %	1.30 GB	22 Depth, 70 Trees
	59.04 %	1.87 GB	22 Depth, 100 Trees
ConvNet	71.01 %	2.12 MB	20 Epochs
	72.30 %	2.12 MB	25 Epochs
PCA	5.72 %	None	

ConvNets in *Stage 2* are trained on images within the bin to simplify learning and also on images in adjacent bins to prevent boundary errors. We used 200K images for *Stage 1* global regression (see Figure 7.1c) wherein the roll, pitch, yaw angles were quantized into 144 bins. Subsequently, 5 Convnets for each of the 144 bins were trained on 10K images within the bin and 10K randomly chosen images in adjacent bins. Training converged after 20 Epochs for the global bin and approximately 10 Epochs for the local rotation bins. The discrete quantization over the joint angle values for each finger is as follows: thumb (144), index (144), middle (36), ring (144), and little (144).

The activation feature associated with the global rotation is critical to the overall accuracy of our approach because this step influences all subsequent ones. To demonstrate the efficacy of ConvNet relative to other approaches, we detail the classification accuracy of ConvNet for global rotation relative to PCA [33] and random forest (RF) [38]. We used 100K depth images because of RF's memory constraints. Table **??** shows that ConvNet achieves a very high accuracy with minimal memory requirement.

5.3 Matrix Completion for Regression

The matrix completion algorithm runs 6 times: once for the 3 global rotation angles and 5 times for estimating the 15 joint angle parameters associated with the fingers. An iterative approach as the one in [71] is inefficient. Instead we evaluate

the unknown parameters in a single shot by assuming a low rank matrix. We discuss the details of our nearest neighbor retrieval to create a pool of activation features followed by the matrix completion method below.

5.3.1 Extracting pool of activation features

Our matrix completion method takes spatio-temporal nearest neighbors as input. Acquiring temporal nearest neighbors are trivial as they are simply the activation features from the previous frames. However, brute force nearest neighbor evaluation from say the 200K global activation vectors introduces a computational bottleneck unsuitable for realtime application. Our solution to alleviate this problem is to use the top classes predicted by the softmax function in ConvNet to first reduce the search space. We then use highly efficient product quantization based nearest neighbor approximation [72] with 8 subquantizers to retrieve the desired number of nearest neighbors. Details of product quantization are skipped for brevity. In practice, we found retrieving a higher fraction of approximate nearest neighbors by product quantization and then selecting the desired number of nearest neighbors using brute force search from this reduced subset to be more robust than direct retrieval.

5.3.2 Matrix completion

Let n be number of spatial nearest neighbors, $\mathbf{D_1} \in \mathbb{R}^{n \times r}$ be the r dimensional activation vectors and $\mathbf{P_1} \in \mathbb{R}^{n \times m}$ be the m desired joint angle parameters being estimated of the n neighbors. In addition, let vector $\mathbf{d_2} \in \mathbb{R}^{1 \times r}$ be the r dimensional activation feature output from ConvNet. Let vector $\mathbf{p_2} \in \mathbb{R}^{1 \times m}$ be the unknown parameters.

$$\mathbf{M} = \begin{bmatrix} \mathbf{D_1} & \mathbf{P_1} \\ \mathbf{d_2} & \mathbf{p_2} \end{bmatrix} \tag{5.1}$$

Our task is to estimate $\mathbf{p_2}$ given the other 3 block matrices. Assuming a low rank structure of matrix M this reduces ro solving:

$$\mathbf{p_2} = \mathbf{d_2}(\mathbf{D_1})^{-1}\mathbf{P_1}, \tag{5.2}$$

The proof of the above result is detailed in subsection 5.3.3.

In practice, we observed that kernelizing the feature matrix and regularizing it by adding a small constant, c to the diagonal, in the spirit of ridge regression makes the output more robust. This parameter c is set to 0.001 in all our experiments. We use the RBF kernel with sigma equal to the variance of the dataset ($\sigma = 200$).

A straightforward extension beyond including just the spatial neighbors is to also include t temporal neighbors from previous frames. This reduces jitter and improves the final quality of our solution. We use 60 nearest neighbors and 16 temporal neighbors for the global parameter estimation. For the 15 local angles, we use 24 nearest neighbors and 4 temporal neighbors. The choice of these parameters is empirically validated in the supplementary material.

5.3.3 Algorithmic details

In this section, we mathematically derive the final equation of our matrix completion model which estimates the unknown pose vector $\mathbf{p_2} \in \mathbb{R}^{1\times m}$ from the activation features $\mathbf{D} = [\mathbf{D_1}; \mathbf{d_2}]$ and the known parameter values $\mathbf{P_1}$.

Figure 5.2 shows the imputation of the block matrices corresponding to equation (1) in the main manuscript, inclusive of spatial and temporal neighbors. First, n nearest neighbors to the input activation feature are retrieved from the database. These activation features and corresponding annotated parameter values are filled into the matrix block corresponding to $\mathbf{D}$ and $\mathbf{P}$, respectively. Additionally, the activation features corresponding to the t previous frames along with the estimated

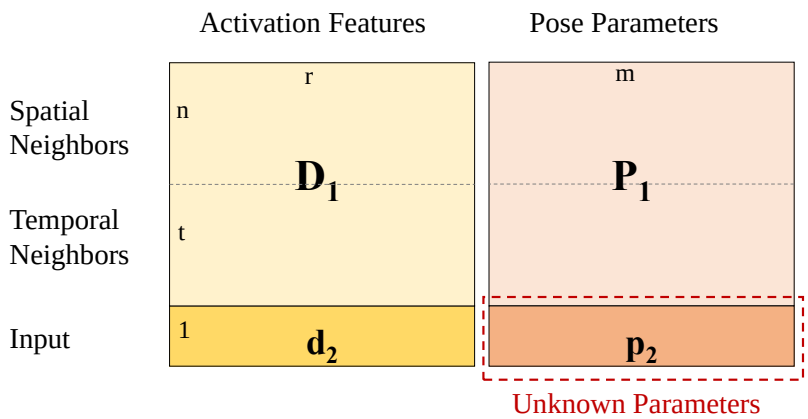


Figure 5.2. The individual block matrices in matrix completion imputed with deep features.

parameter values serve as temporal neighbors in the matrix blocks, $\mathbf{D}$ and $\mathbf{P}$. Suppose $\mathbf{p_2}$ is a submatrix of the matrix $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{D_1} & \mathbf{P_1} \\ \mathbf{d_2} & \mathbf{p_2} \end{bmatrix} \tag{5.3}$$

where $\mathbf{D_1} \in \mathbb{R}^{(n+t)\times r}$, $\mathbf{d_2} \in \mathbb{R}^{1\times r}$, and $\mathbf{P_1} \in \mathbb{R}^{(n+t)\times m}$, and r is the dimensionality of the feature vector.

Lemma 5.3.1 *Suppose that the matrix* $\mathbf{X}$ *is of rank* k *and partitioned as shown in equation 7.1. We assume that the matrix* $\mathbf{D_1}$ *also has rank* k*. Then*

$$\mathbf{p_2} = \mathbf{d_2}(\mathbf{D_1})^{+}\mathbf{P_1}, \tag{5.4}$$

where + denotes the Moore-Penrose pseudo-inverse.

Proof The matrix $\mathbf{X}$ is decomposed using SVD to rank k as $\mathbf{X} = U\Sigma V'$ where $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_k)$, $U \in \mathbb{R}^{(n+t+1)\times k}$, and $V \in \mathbb{R}^{(r+m)\times k}$. Assume $U_1 \in \mathbb{R}^{(n+t)\times k}$

and $U_2 \in \mathbb{R}^{1\times k}$. Consequently, $V_1 \in \mathbb{R}^{r\times k}$ and $V_2 \in \mathbb{R}^{m\times k}$. Then, we can rewrite $\mathbf{D_1} = U_1\Sigma V_1', \mathbf{P_1} = U_1\Sigma V_2', \mathbf{d_2} = U_2\Sigma V_1'$, and $\mathbf{p_2} = U_2\Sigma V_2'$. Let $\mathbf{D_1} = LR$, where $L = U_1 S$ and $R = SV_1'$ for $S = \sqrt{\Sigma}$. Using MacDuffee's theorem as done in [73],

$$\begin{aligned} \mathbf{D_1}^+ &= R^+L^+ \\ &= R'(RR')^{-1}(L'L)^{-1}L' \\ &= V_1S(SV_1'V_1S)^{-1}(SU_1'U_1S)^{-1}SU_1' \\ &= V_1(V_1'V_1)^{-1}\Sigma^{-1}(U_1'U_1)^{-1}U_1' \end{aligned} \tag{5.5}$$

As a result, $\mathbf{d_2}(\mathbf{D_1})^+\mathbf{P_1}$ equates to

$$\begin{aligned} \mathbf{d_2}(\mathbf{D_1})^+\mathbf{P_1} &= (U_2\Sigma V_1')V1(V_1'V_1)^{-1}\Sigma^{-1} \\ &\qquad (U_1'U_1)^{-1}U_1'U_1\Sigma V_2' \\ &= U_2\Sigma V_2' \\ &= \mathbf{p_2}. \end{aligned} \tag{5.6}$$

This completes the proof. ■

In practice, we kernelize the feature matrix $\mathbf{D}$ as radial basis functions (RBF):

$$\mathbf{K}(\mathbf{D},\mathbf{D}) = \exp\left(-\frac{\|\mathbf{D}^T\mathbf{D}\|^2}{2\sigma^2}\right), \tag{5.7}$$

where σ denotes the variance of the database (σ=200). The auxiliary knowledge about nearest neighbors is implicitly accounted for in the kernelized similarity matrix $\mathbf{K}$, making the estimation more robust to outliers and noise. Note that the kernelized matrices $\mathbf{K_1}$ and $\mathbf{k_2}$ replace matrices $\mathbf{D_1}$ and $\mathbf{d_2}$ in equation 7.1 with appropriate dimensions.

$$\mathbf{X} = \begin{bmatrix} \mathbf{K_1} & \mathbf{P_1} \\ \mathbf{k_2} & \mathbf{p_2} \end{bmatrix} \tag{5.8}$$

where $\mathbf{K_1} \in \mathbb{R}^{(n+t)\times(n+t)}$, $\mathbf{k_2} \in \mathbb{R}^{1\times(n+t)}$, and $\mathbf{P_1} \in \mathbb{R}^{(n+t)\times m}$. We ensure invertibility of matrix $\mathbf{K_1}$ by adding a diagonal matrix, $c\mathbf{I}$ to $\mathbf{K_1}$ where $c = 0.001$. Consequently, the kernelized version of equation 7.2 can be solved directly without resorting to an intermediary SVD of $\mathbf{K_1}$ which is computationally expensive. This diagonal matrix also acts as a regularizer and prevents overfitting similar in spirit to kernel ridge regression. The final solution is given by:

$$\mathbf{p_2} = \mathbf{k_2}\left(\mathbf{K_1} + c\mathbf{I}\right)^{-1}\mathbf{P_1}, \tag{5.9}$$

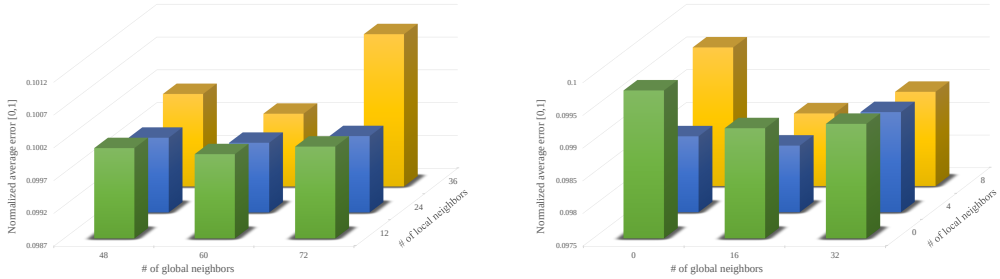
5.4 System Specifications

5.4.1 Running and training times

Our hierarchical framework for hand pose estimation takes advantage of multi-threading (OpenMP). The pose parameters in *Stage 2* corresponding to the five finger articulations are evaluated in parallel using five threads. Our system runs at 32 FPS ($\approx 31ms$ per frame) on an Intel Xeon E3-1240 CPU with 16GBs RAM. The computation time for each frame is split as $2ms$ for preprocessing (*i.e.*, region of interest extraction, and resizing the depth image to dimension 64×64), $9ms$ for *Stage1* which estimates the global orientation parameters, and $20ms$ for *Stage2* which estimates the local finger articulations. In order to speed up training, the ConvNets were trained with the aid of GPU (NVIDIA Quadro K4000 Graphics card). The training for global orientation parameters took about four hours and local parameters for the 144 bins took about 30 hours.

5.4.2 Justification of design choices

We use 60 spatial neighbors with 16 temporal neighbors for global parameter estimation and 24 spatial neighbors with 4 temporal neighbors, respectively, for all quantitative evaluations in Section 6.3 and 6.4 in the main manuscript. In this subsec-



(a) (b)

Figure 5.3. Design choices. Joint angle error is normalized between 0 and 1. (a) Choice of spatial neighbors n. Minimum joint angle error is achieved when the number of neighbors for global pose estimation are 60 and for local estimation are 24. (b) Choice of temporal neighbors t. The system shows highest accuracy with 16 neighbors for global estimation and 4 neighbors for local estimation.

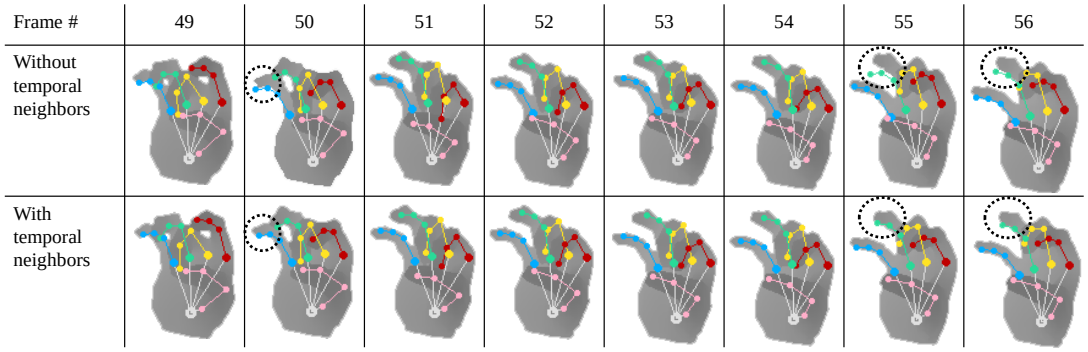


Figure 5.4. The effect of temporal neighbors on final hand pose estimation. Top row shows the result of our method without temporal neighbors and bottom row shows the result with temporal neighbors on continuous frames from our synthetic dataset. The dashed circles highlight the increased robustness and reduced jitter of final hand pose by incorporating temporal frames into matrix completion.

tion, we empirically validate the choice of these parameters. Figure 5.3a compares the accuracy achieved by using different number of spatial neighbors from the database for global and local parameter estimation on the synthetic database described in the main manuscript. The minimum mean joint angle error is achieved when we use 60 and 24 spatial neighbors for global and local parameter estimation, respectively.

A higher number of neighbors is inefficient both in terms of accuracy and time in our matrix completion model, whereas lesser number of neighbors may result the estimation of joint parameters to be stuck in a local minima. We also conducted experiments to find the balance between spatial and temporal neighbors. In order to reduce jitter in the pose estimates, we add t number of temporal neighbors in the matrix block (*i.e.*, matrix $\mathbf{D}$ and $\mathbf{P}$) as shown in Figure 5.2. In Figure 5.3b, we see that 16 temporal neighbors for global and 4 temporal neighbors for local parameter estimation is optimal in terms of achieved accuracy. The lower number of temporal nearest neighbors compared to spatial nearest neighbors indicates that the activation features contain implicit information about adjacent hand poses. The lower number of temporal nearest neighbors also makes our method robust to rapid hand movements, severe occlusion and other scenarios for which temporal information may not be reliable. However, including temporal nearest neighbors reduces jitter. This effect is displayed in Figure 5.4. The top row displays the result on continuous frames without incorporating temporal neighbors and the bottom row corresponds to the result by including temporal neighbors. We observe that the resulting hand pose by incorporating temporal neighbors is more robust (see dashed circles), and reduces jitter in a real-time setting.

5.5 Experiments

We conduct a comprehensive evaluation with state-of-the-art approaches as well as self-generated baselines on the synthetic and real datasets to demonstrate the efficacy of our solution. We first describe the datasets and baselines.

5.5.1 Datasets

We split our evaluation into two stages. First, we use synthetic data to compare our method to baselines. This comparison validates the rationale of our specific approach against other choices. This data is generated using the same approach as

described in Section 3 to generate our database, albeit continuity constraints are enforced. Two synthetic sequences are generated which are 2.5K frames long at standard rates (approximately 80 seconds each). The advantage of these synthetic sequences are that they are already labeled, avoiding tedious ground-truth assignment.

Next, for fair comparison to other methods, we evaluate the performance of our method on two publicly available datasets: Dexter1 [74] and NYU [3]. The Dexter1 dataset consists of seven gestures (*i.e.*, adbadd, flexex1, pinch, fingercount, tigergrasp, fingerwave, and random) with high inter-gesture verifiability, however, mostly from frontal viewpoints. Hence we use the NYU dataset for a more thorough evaluation of the method. As we shall shortly show, our method remarkably achieves state-of-art performance without fine-tuning on their training dataset.

Although the authors are aware of other datasets like ICVL [67], MSRA14 [4], or MSRA15 [33] in the literature, we do not use them for one or more of the following reasons: (1) the depth pixels of the body are included with the hand depth map. Recall we use a heuristic method for segregating the hand from the rest of the body and a wrist band under more extreme conditions. We did not find a straightforward way to segregate the data without incurring loss. (2) The hand poses are enforced using muscular labor, i.e., hand configurations wherein one or more finger applies pressure on another. These configurations are not accounted for in our joint angle modeling framework to render synthetic depth maps, however, modeling additional constraints to account for such hand poses is plan of future work. Also note that we use the SoftKinetic's DethSense DS325 for all our real demonstrations.

5.5.2 Baselines for method validation

There are three salient features of our approach which we rigorously validate. First, a hierarchical approach is justified in spite of the computational overload it introduces. Second, a pool of activation features is better at estimating the hand pose than a single activation feature or a direct regression based approach using

ConvNets. Third, our choice of imputing the matrix with spatio-temporal neighbors and kernelizing the features provides superior performance. We naturally perform this validation by comparing to the following three baselines: (a) *Holistic* which evaluates all parameters in an all-in-one approach using a single activation feature. We also compare it to *JMFC* which also performs a matrix update using a single feature vector, although using computationally expensive iterations in [71] (b) *Conv-PQ* which directly estimates the pose parameters to be the nearest neighbor and *Regression* which directly regresses pose parameters using ConvNets with L2 loss are used to validate our choice of pool of activation feature, and finally (c) *No-temporal* which contains only spatial neighbors for matrix completion, *Non-kernel* which uses feature matrix without kernelization, and *Weighted* which finds pose parameters using Gaussian similarity between activation features as weights are used to validate our matrix completion approach. The validation is done in terms of one or more of the following standard error metrics popular for pose estimation problems: (a) the average joint angle error in degrees, (b) the average joint distance error in millimeters, (c) the maximum allowed joint angle error in terms of a threshold ε_A, and (d) the maximum allowed joint distance error in terms of a threshold ε_D. Broadly speaking, the first two metrics evaluate performance at a local joint level whereas the the other measure global robustness of an approach. We employ the appropriate metric based on the context of the evaluation. Although our angle based method is particularly effective in minimizing joint angle errors, yet we choose joint distances as our error metric on public datasets to demonstrate the overall robustness of our approach.

5.5.3 Comparison to baselines

In this section, we quantitatively evaluate our method with respect to the baselines on the synthetic datasets. Figure 5.5 shows that our method significantly outperforms the proposed baselines both in terms of local as well global error metrics. The performance markup over the *Conv-PQ* approach as seen in Figure 7.7c indicates that

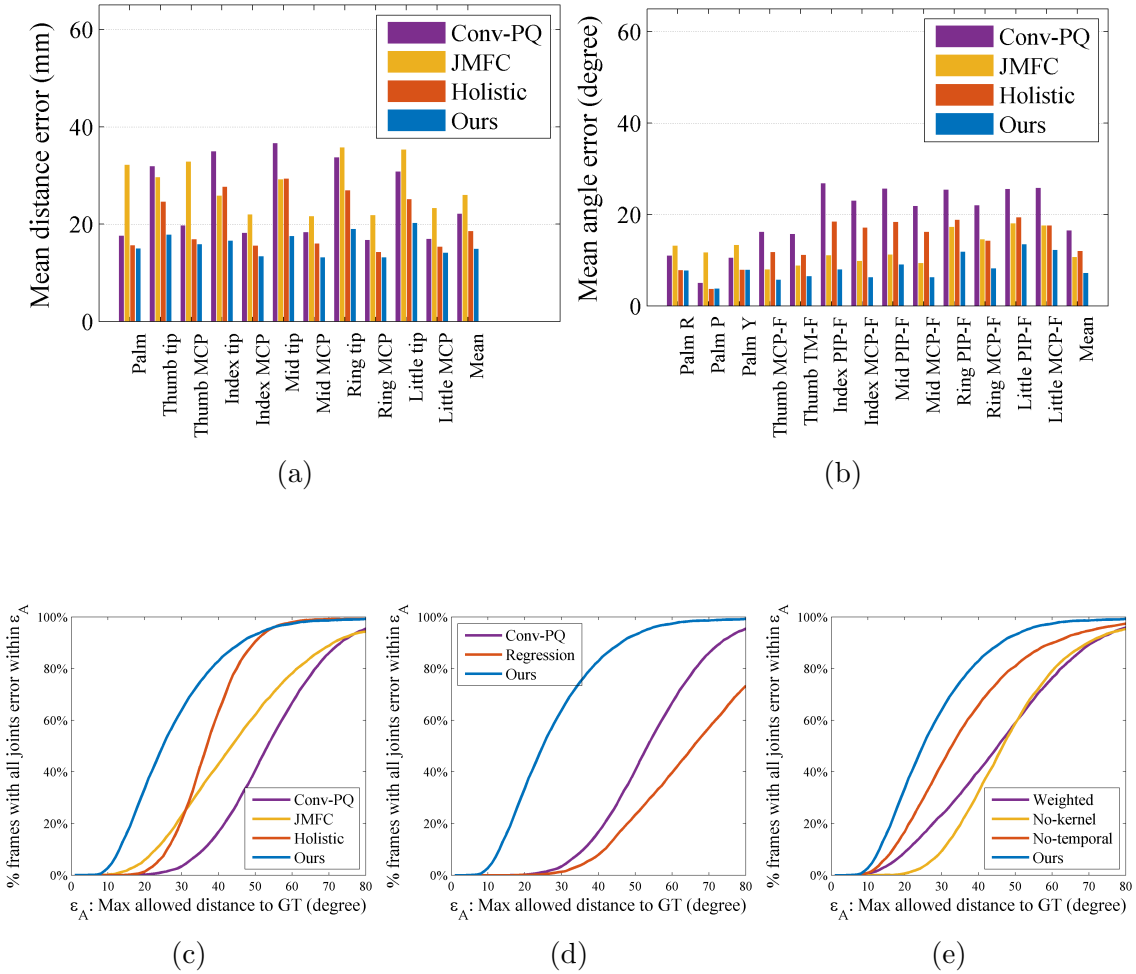


Figure 5.5. The results of quantitative evaluation on the synthetic dataset.

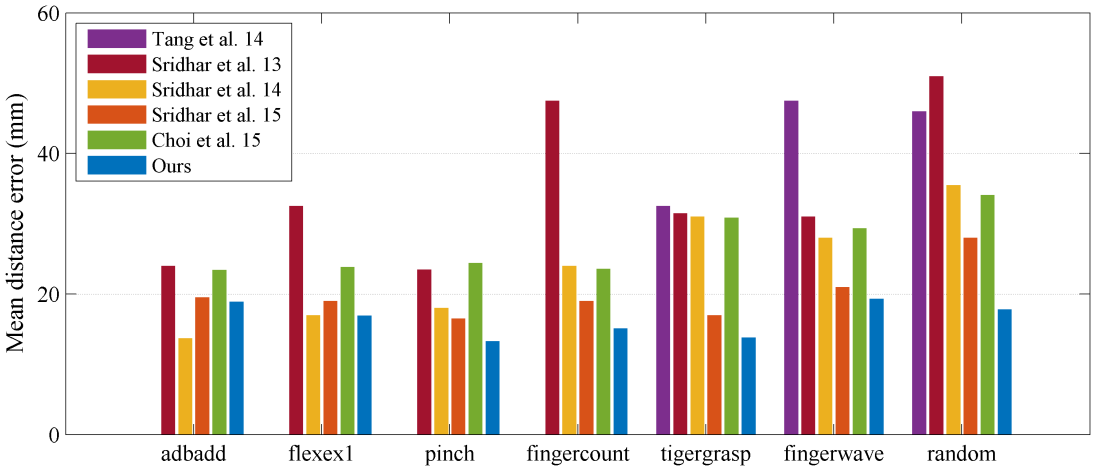
a ConvNet by itself would do a poor job of inferring a complex articulated structure such as the hand. The performance improvement over Holistic in the zone of small angles is also intuitive. It indicates that the global activation feature contains some latent information about the local joint angles, but this information is better revealed by a hierarchical estimation procedure. This is also validated in Figure 7.7a and 7.7b where we see a significant performance improvement in terms of joint angles for finger portions that are frequently occluded such as the middle finger. It is also noteworthy to note that the similarity of these plots in terms of error ranges to plots on real hand sequences implicitly validate our data creation process. *Regression* [2] for joint angle prediction resulted in worse performance than even *Conv-PQ* baseline (nearest activation feature) as shown in Figure 7.7d. We adopted different approaches, *e.g.,* fine-tuning our ConvNets, L1 loss, etc.to ensure that direct regression is indeed suboptimal. We contend that as joint angles are a function of relative joint points,learning joint angles is harder compared to joint positions, and hence, resulted in inferior performance. Figure 5.5e shows the performance of matrix-completion baselines relative to our proposed approach. The figure validates that constructing a kernel, incorporating temporal information and using matrix completion instead of simple weighted regression are all critical to good performance.

5.5.4 Comparison with the state-of-the-arts

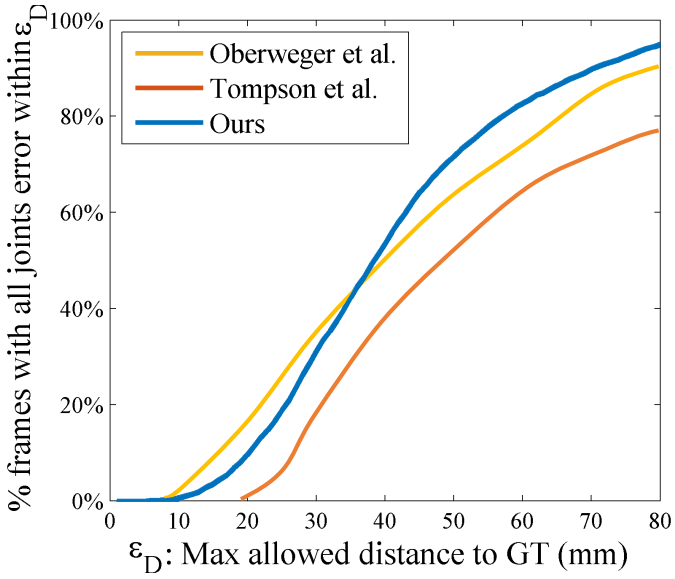
Having validated the rationale of our approach, we now compare our method to other state-of-the-art approaches [3, 5, 6, 67, 71, 74, 75] on the Dexter1 and NYU datasets.

Quantitative Analysis We measured the average distance error of five fingertips (in mm) on the Dexter1 dataset to evaluate the overall robustness of our approach. Figure 5.6a shows the comparison of our approach to other methods which include both discriminative [67, 71] as well as generative [74, 75] methods. Not only does our

[2]the penultimate layer is of dimension 2048 as we do not need nearest neighbor retrieval



(a)



(b)

Figure 5.6. The results of quantitative evaluation on the public dataset. Note that the accuracies are directly estimated from corresponding figures (*i.e.*, figure 4 in [5] and figure 3a in [6]).

method achieve the lowest overall error rate (see Table **??**), we also achieve the lowest individual error rates for all but one gesture *i.e.adbadd*. This is because the particular gesture is especially hard to model in terms of joint angle constraints.

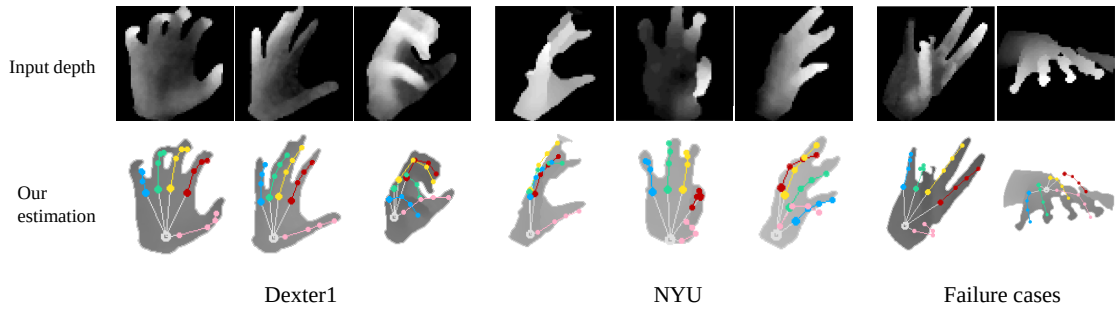


Figure 5.7. Qualitative evaluations are conducted on two public datasets, Dexter1 and NYU. The first row shows the input depth image, and corresponding estimation is presented in the second row.

We evaluated our approach directly on the 8.2K of test depth maps from the NYU dataset. Figure 5.6b illustrates the maximum allowed error with respect to the distance threshold. The fact that our method performs better than [6] over a long range indicates the activation features we get from ConvNet can be used across domains and sensor types [3], and hence the activation features can potentially be made general purpose. This is encouraging in the context of progressively fine-tuning ConvNets with more information such as when new joint angle constraints or dynamic constraints become available. Furthermore, simulating principled noise models such as [76] corresponding to true sensor noise can further enhance the generality of these features in the context of hand pose estimation.

Qualitative Analysis We do a qualitative evaluation of our algorithm with the state-of-the-art methods on some public datasets. The top row of Figure 7.9 shows cropped 64x64 depth images which are used as input to our system, and the second row shows corresponding estimates with our matrix completion method (without temporal neighbors). All estimated poses are kinematically valid and follow a natural sequence. For the sake of completion, we also show some failure cases in the last two columns of Figure 7.9. In our system this happens when some unnatural pose (driven

[3]NYU dataset use PrimeSense to capture their data

Table 5.4. The overall average error (mm) of the five fingertip positions on Dexter1. Ours shows the lowest error rate compared to the state-of-the-art methods.

Methods	[67]	[74]	[75]	[5]	[71]	Ours
Error	42.4	31.8	24.1	19.6	25.27	16.35

by muscular force) appears in front of the camera or when the image is severely affected by noise or has missing parts.

5.6 Conclusion and Future Work

We present a novel framework for hand pose estimation using a deep convolutional neural network. Instead of using a single activation feature, we use a pool of activation features to synchronize and collectively estimate the hand configuration, all in real time. This pool is derived by training a deep ConvNet with a large database of synthetic hand poses and efficiently storing the activation feature corresponding to the penultimate fully connected layer. Careful thought was placed so that this database is reflective of real data. At runtime the pool of activation features in the spatial domain and temporal domain combine together in a hierarchical way to robustly estimate the hand pose. The derived activation features can be applied across domains and sensor types as demonstrated in our experiments. Furthermore, our method achieves state of the art performance. Although our approach is general, one limitation of our activation features is that the estimations are only valid in the joint angle domain. Future work will focus on ways such that people working in the joint angle or joint position domain can seamlessly fuse their models together to create even deeper and more robust models. Another line of future work is to investigate our matrix completion approach in a more general setting. The simplicity combined with its efficiency makes a promising alternative to standard regression techniques for a wide array of machine learning tasks.

6. LEARNING HAND BY HALLUCINATING GEOMETRIC REPRESENTATIONS

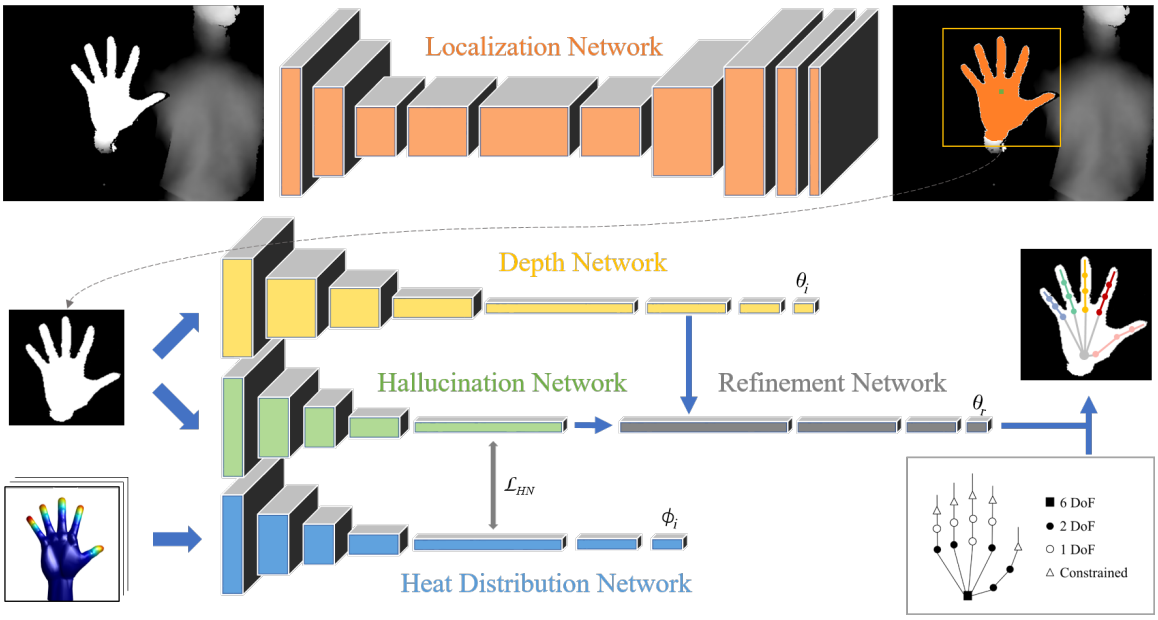


Figure 6.1. The pipeline overview. At training time, the hallucination network is trained to mimic heat distribution features using depth data. At testing time, the localization network takes as input a depth image to localize the hand. The identified hand is used to extract complementary features from the depth and hallucination network. The refinement network regularizes an initial pose estimate using the given feature representations.

Although extensive research efforts have provided a coarse interpretation of hand movements, the current hand pose estimation approaches do not include: (i) an understanding of the geometric consistency of complex kinematic poses of the articulated hand and (ii) an additional input modality (besides a single depth image) to produce a better estimation model. In this chapter, we demonstrate that better hand pose estimation can be attainable when these gaps are addressed.

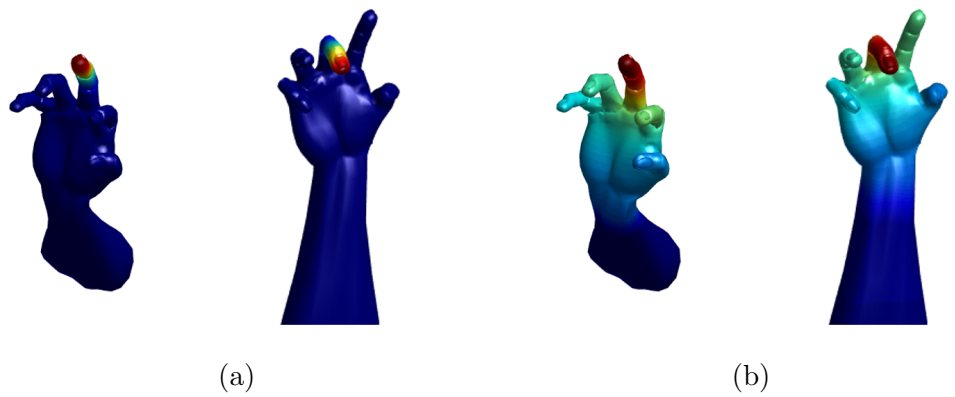


Figure 6.2. Visualization of the heat distribution descriptor on different hand poses over time. (a) The point heat source (red-colored) is placed at the tip of the middle finger at time $t = 0$. (b) For a large value $t = 40$, the behavior of heat distribution is geometrically consistent on both poses.

We propose a promising method for 3D hand pose estimation that achieves performance higher than or comparable to the state-of-the-arts. Specifically, we exploit a convolutional neural network (ConvNet) model which can extract the property of heat distribution over a 3D hand mesh model from a single depth image. The proposed method incorporates a heat distribution network to learn a geometrically informative representation of hand articulations as an additional modality. At training time, our modality hallucination network takes as input a depth image and is trained to capture the corresponding heat distribution modality. Thus, our method produces both the depth and heat distribution features from a single depth image at test time.

6.1 Heat Distribution

We briefly discover a heat operator derived in [17] and introduce the heat distribution descriptor to be used to train our hallucination network.

6.1.1 Heat flow on the hand surface

Our hand model $\mathcal{M}$ is a compact Riemannian manifold without boundaries, which consists of 3,869 mesh vertices and 7,734 triangular faces. Thus, we can write the heat diffusion equation on the surface of the hand:

$$\left(\Delta + \frac{\partial}{\partial t}\right) u(i,t) = 0, \tag{6.1}$$

where Δ is the Laplace-Beltrami operator and $u(i,t)$ is heat distribution at vertex i at time t. In addition, let H_t be the heat operator which satisfies $\mathrm{H}_t = e^{-t\Delta}$. Then the solution to Eqn. 6.1 is $u(i,t) = \mathrm{H}_t(f)$, where $f : \mathcal{M} \to \mathbb{R}$ denotes the amount of heat available at $t = 0$. Therefore, the heat flowing through the mesh surface from source vertex j to i at a given diffusion time t for all $i, j \in \mathcal{M}$ can be denoted by the heat kernel $\mathrm{H}_t(i,j)$:

$$\mathrm{H}_t(i,j) = \sum_k e^{-\lambda_k t} \mathrm{v}_{ki} \mathrm{v}_{kj}, \tag{6.2}$$

where λ_k and v_k is the k-th eigenvalue and the k-th eigenfunction of the Laplace-Beltrami operator Δ, respectively.

6.1.2 Heat distribution descriptor

Figure 6.2 illustrates heat distribution on the hand surfaces over time. A unit heat source is given at the tip of the middle finger (marked in red) at time $t = 0$, and the amount of diffused heat to the rest of the surface is visualized in Figure 6.2b. At small time scales, the local geometry of the hand can be investigated, while the global structure can be encoded at large scales. Note that the analogy of heat distribution on different hand poses validates the geometrically consistent property of the diffusion process. This property motivates us to design a heat distribution descriptor which is invariant to shape deformation and topological changes.

We employ the characterization of heat distribution at each point $i \in \mathcal{M}$ heat transferred from a set of key sources $J = \{j_1, ..., j_5\}$ in $T = \{t_1, t_2, t_3\}$ time steps. Let

P be the number of vertices of $\mathcal{M}$, then the proposed heat distribution descriptor $\mathbf{d}_t \in \mathbb{R}^{P\times 1}$ is as follows:

$$\begin{aligned} \mathbf{d}_t^j &= [\mathrm{H}_t(i_1, j), ..., \mathrm{H}_t(i_p, j), ..., \mathrm{H}_t(i_P, j)]^{\mathrm{T}} \\ & \qquad\qquad\qquad\qquad \forall j \in J \quad \text{and} \quad \forall t \in T, \\ \mathbf{d}_t &= \sum_j \mathbf{d}_t^j \quad \forall t \in T. \end{aligned} \tag{6.3}$$

Here each entry of the P-dimensional vector $\mathbf{d}_t$ corresponds to the cumulative amount of heat available at each vertex i at time $t \in T$ diffused from source vertices $j \in J$. Consequently, we compute the heat distribution matrix $\mathbf{D} = [\mathbf{d}_{t_1}, \mathbf{d}_{t_2}, \mathbf{d}_{t_3}] \in \mathbb{R}^{P\times T}$,

where $\{t_1, t_2, t_3\} = \{10, 30, 50\}$ in practice[1]. We further process the descriptor matrix $\mathbf{D}$ by rendering each column vector as an image format. A hidden point removal [56] strategy determines the visible vertices from the viewpoint of a camera. Our pose simulator investigates the visibility of the vertices and linearly interpolates the amount of heat distribution between neighboring vertices using the Phong interpolation method. As a result, we generate T-channel descriptors to feed into our hallucination network described in the following section.

Note that we use a heuristic to determine the heat sources J. We uniformly sample each of the source points from P vertices. These indices are fixed while generating our training dataset so that every hand poses share consistent geometric representations. Also, note that we do not find a significant difference in regression accuracy when we choose another set of J. At test time, the input point cloud is not indexed for the heat sources, and this is the main reason we hallucinate heat distribution features from a depth image.

[1]We observe that the behavior of heat diffusion is local (finger-level) at $t = 10$ and becomes global (hand-level) at $t = 50$.

6.2 Learning Hand Articulations

Our system follows the approach of [34, 71, 77] that estimates the *joint angle parameters* on a per-frame basis. Unlike the other pose estimation methods, this approach directly employs the motion constraints guided by the physical anatomy of the hand. In this setting, all estimated poses are kinematically valid and follow a natural sequence, and this is why we choose the angle parameters over the joint positions. Now we discuss how the proposed method learns hand articulations from depth and auxiliary modality features, in the form of the joint angles.

6.2.1 Localization network

Hand localization (*i.e.*, hand segmentation and region of interest extraction) has been heuristically solved in the literature [4, 32, 33, 71, 77, 78] by assuming (i) the hand appears largest in front of the sensor or (ii) the wristband can be identified by color segmentation. However, the underlying assumptions would be further from real scenarios, such as those at far-range or with a cluttered background. To achieve robust performance for localization, we divide the problem into two sub-tasks: hand segmentation and hand center regression.

We present a ConvNet architecture specifically designed to solve these tasks at one go. The graph of the network architecture is visualized in the supplementary material. Our main insight is that the deep neural network effectively identifies pixel-wise class labels through the convolution process [79]. To achieve this from our hand segmentation problem, the first three convolutional layers with a following max pooling layer down-sample the input 240×240 image to be the size 30×30. The next four convolutional layers capture the low-level image features in depth to distinguish the hand and background. Then we perform two unpooling operations in between convolutions to up-sample the given depth features (to be the size 120×120). The unpooling uses the original activations stored from the previous max pooling layers, which is critical for our system for the following reasons: (i) the unpooling process

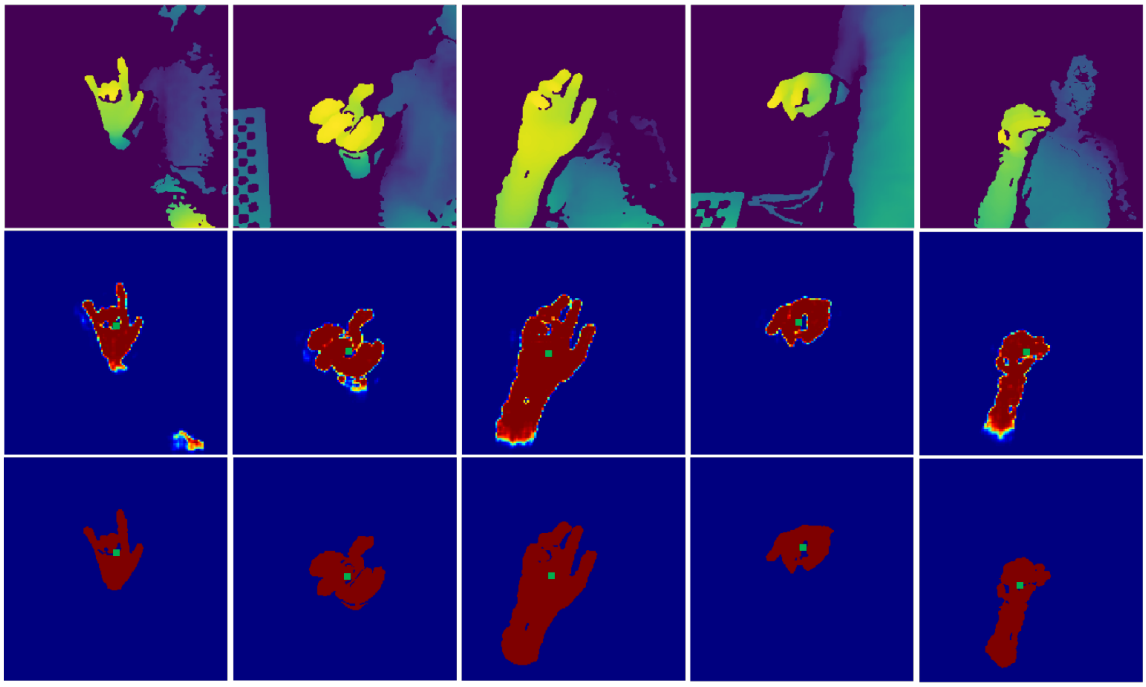


Figure 6.3. Visual analysis of hand localization. First row: input 240×240 depth images cherry-picked from the HandNet [81]. Second row: estimated hand probability map and centroid (green square). Third row: ground truth labels.

consistently increases the spatial size of the feature map to reconstruct the detailed hand segment, and (ii) it balances computational time and segmentation accuracy by generating sparse representations. Note that the deconvolution method [80] was also considered, which showed similar accuracy but required higher processing time because of its convolution operation. In addition, we employ intermediate convolutional features to regress the hand center. This branch is comprised of four additional convolutions and one inner product, estimating the centroid of the hand $\{u_c, v_c\}$[2]. It is further converted into the triplet ϕ_c to draw the bounding box around the hand. The result of hand localization is visualized in Figure 6.3.

[2]In practice, we achieved the mean distance error of 14.56 *pixels* in an image of size 320×240 on the HandNet dataset [81].

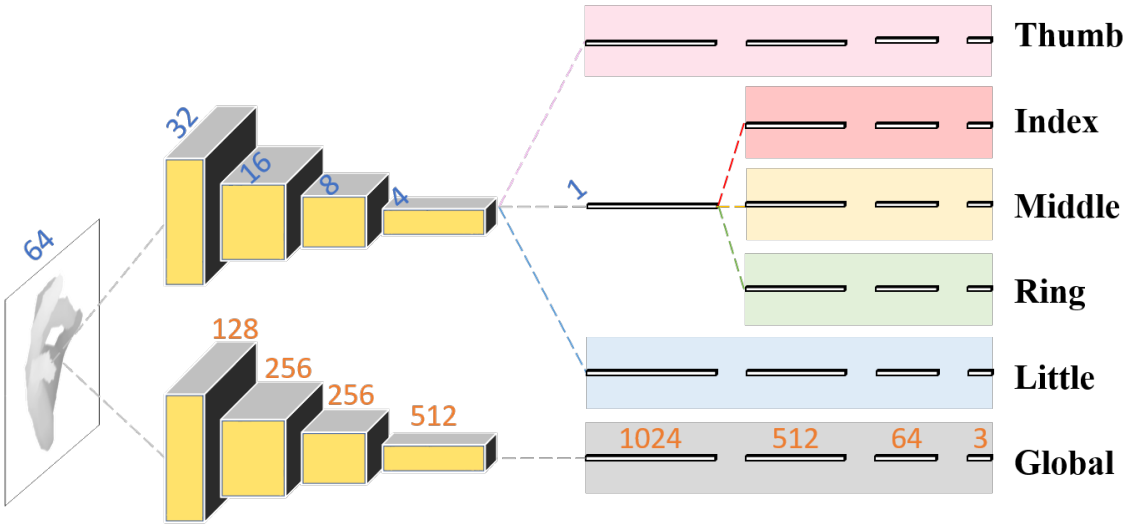


Figure 6.4. The proposed depth network consists of two streams: the top stream for the five fingers and the bottom stream for the global orientation parameters. Numbers in blue indicate the width & height of the feature map, and those in orange represent the number of kernels.

6.2.2 Multi-modal learning

Our system learns complementary features about hand articulations from different modalities. We train the depth network with the joint angle labels, taking into account the process of knowledge transfer across fingers. Moreover, multi-scale convolutional features are encoded through the heat distribution network to identify the fingertip positions.

Depth network (DN) The success of multi-task learning in [82] has caused immense effects on the deep learning models (*e.g.*natural language processing in [83, 84], face detection in [85, 86], and human pose estimation in [87, 88]). These works all aim to achieve improved performance and prevent overfitting by transferring shared knowledge. Aligned with these works, we estimate the joint angle parameters of five fingers θ_i from a single network. The architecture of our multi-task depth network is shown in Figure 6.4. The first four convolutional layers share knowledge of the hand. This is crucial for learning a perceptual set of attributes, such as self-occlusions or

self-similarities of the fingers across domains and hence leads to further improvements in the regression performance (see Section 6.3). For our specific operation, we group the fingers according to the anatomical position (*i.e.*, three groups: thumb, index-middle-ring, little) before passing the fifth convolutional layer. This insight allows us to achieve higher regression accuracy by learning structural representations from a correlation of adjacent fingers. In addition, we explore the global orientation of the hand from a separate network initiated in parallel using the same network configuration. For the proposed depth network (DN), we introduce the loss weights α, β, and γ to properly scale the loss function:

$$\mathcal{L}_{DN} = \alpha\mathcal{L}_T + \beta(\mathcal{L}_I + \mathcal{L}_M + \mathcal{L}_R + \mathcal{L}_L) + \gamma\mathcal{L}_G, \tag{6.4}$$

where the subscript denotes each finger (T: thumb, I: index, M: middle, R: ring, L: little, G: global). In practice, we observe that the thumb finger contributes less to the total loss $\mathcal{L}_{DN}$. Thus, we set the loss weights $\alpha = 3$, $\beta = 1$, and $\gamma = 1$ which balance the optimization process.

Heat distribution network (HDN) Our heat distribution network is trained with the fingertip position labels using T-channel descriptors that represent the multi-scale heat distribution property of the hand. The bottom of Figure 6.5 illustrates the overall architecture. Each of the parallelized networks independently learns hand articulations from local to global geometric features through the first five convolutional layers. Then we utilize a feature concatenation step to aggregate three convolutional features into a single composition along the depth dimension. This step allows us to encode both the finger-level local geometry and global hand structure into more informative representations generated across the time scales. As a result, our network is capable of learning a better mapping function between input hand poses and the corresponding fingertip positions ϕ_i.

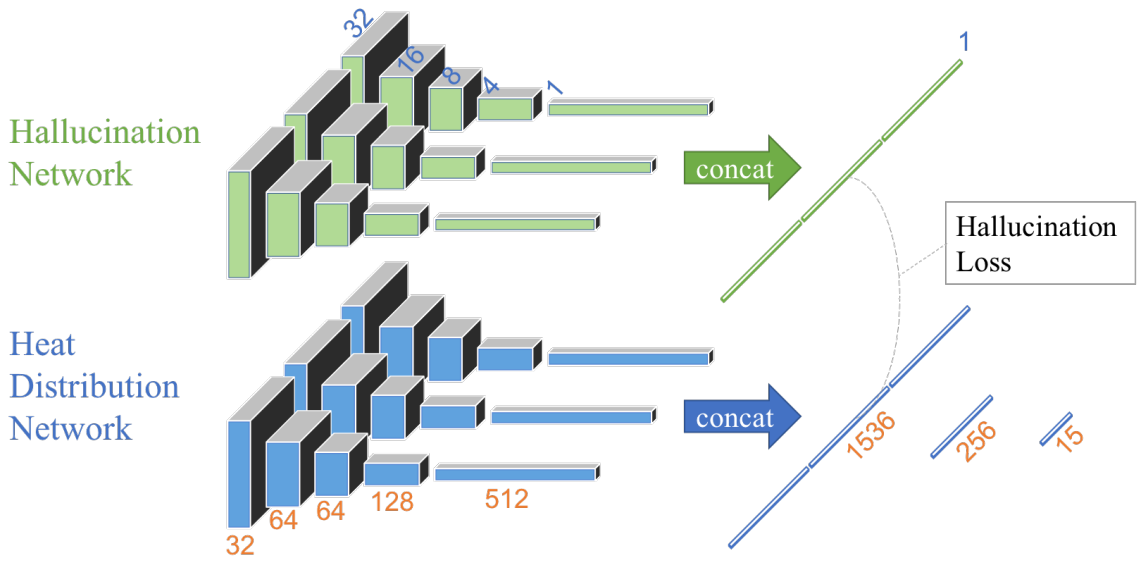


Figure 6.5. The architecture of the hallucination network (top) and heat distribution network (bottom). The concat layer concatenates multiple features to one blob. Numbers in blue indicate the width & height of the feature map, and those in orange represent the number of kernels.

6.2.3 Modality hallucination and refinement

Hallucination network (HN) The parameter values (i.e., weights and bias) of the hallucination network (HN) are initialized using the network parameters of the pre-trained heat distribution network (HDN). We then fine-tune these values with a Euclidean loss $\mathcal{L}_{HN}$ between the intermediate feature vectors, similarly to [22]. However, we do not use the whole structure of the HDN from our HN. Instead, our HN has only the first five convolutional layers as illustrated at the top of Figure 6.5. Note that the choice of the number of layers is empirically determined in the next part. As a result, our hallucination network outputs the geometrically descriptive responses learned from the heat distribution descriptors using a corresponding depth image.

Refinement network (RN) Conceptually, the resulting triplets $\boldsymbol{\phi}_i$ together with the joint angles $\boldsymbol{\theta}_i$ estimated from the DN are used to regularize the angle parameters in

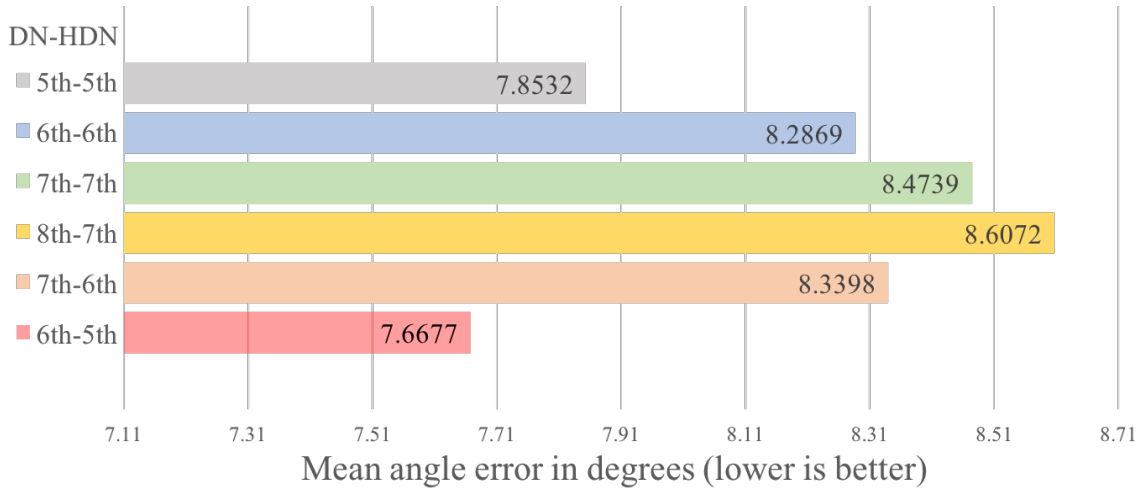


Figure 6.6. The mean angle error is used to evaluate different combinations of feature concatenation on our synthetic dataset. A number associated with each colorbar denotes layers of the DN (top) and the HDN (bottom), respectively. The best accuracy is achieved when we concatenate depth features extracted after 6th conv layer and heat distribution features extracted after 5th conv layer (red bar).

the refinement network (RN). In practice, however, the direct use of $\boldsymbol{\theta}_i$ and $\boldsymbol{\phi}_i$ does not achieve performance improvements. Alternatively, our RN takes as input a feature vector that is well-informed to predict the joint angle parameters $\boldsymbol{\theta}_i$ and fingertip positions $\boldsymbol{\phi}_i$. Hence, we generate a concatenated vector of depth feature F^l_{DN} and mimicked hallucination feature F^l_{HN}. The input feature maps for concatenation can be extracted from any layer l in the network, so we empirically determine where to extract these features with respect to regression accuracy. Figure 6.6 compares the performance of various combinations of feature concatenation. Note that we conduct these experiments using the depth activations and heat distribution activations F^l_{HDN} to eliminate the effect of hallucination error. It shows that the concatenation of depth features extracted after the sixth convolutional layer and heat distribution features extracted after the fifth convolutional layer achieves the highest performance (red bar). The RN consists of the four inner product layers with a following non-linear

(ReLU) layer. We progressively reduce the dimension of the vector as a factor of 4, that is 2048-512-128-n (where $n = 18$ is the number of angles).

Network optimization Finally, we have three sets of network parameters independently learned from the depth network (DN), hallucination network (HN), and refinement network (RN). We further fine-tune the given networks using depth data and the corresponding angle labels θ. Then the total loss can be drawn as follows:

$$\mathcal{L}_{Optimize} = \zeta \mathcal{L}_{DN} + \eta \mathcal{L}_{RN}. \tag{6.5}$$

We set the loss weights $\zeta = 1$ and $\eta = 5$ so that the depth network and refinement network to be properly optimized with input depth data without updating the heat distribution network. Note that the same loss weights (α, β, γ) are used for the depth network as discussed previously.

6.3 System Specifications

In this section, we present details of our localization network and the system specifications.

6.3.1 Architecture of the localization network

We visualize the graph of our localization network in Figure 6.8. The proposed network solves two sub-tasks for hand localization: hand segmentation and hand center regression. The segmentation stream identifies pixel-wise class labels through a series of the convolution process, so the resulting probability map can effectively reconstruct the detailed hand segment. Also, the regression stream robustly estimates the centroid of the hand and thus enable us to draw the bounding box around the segmented hand.

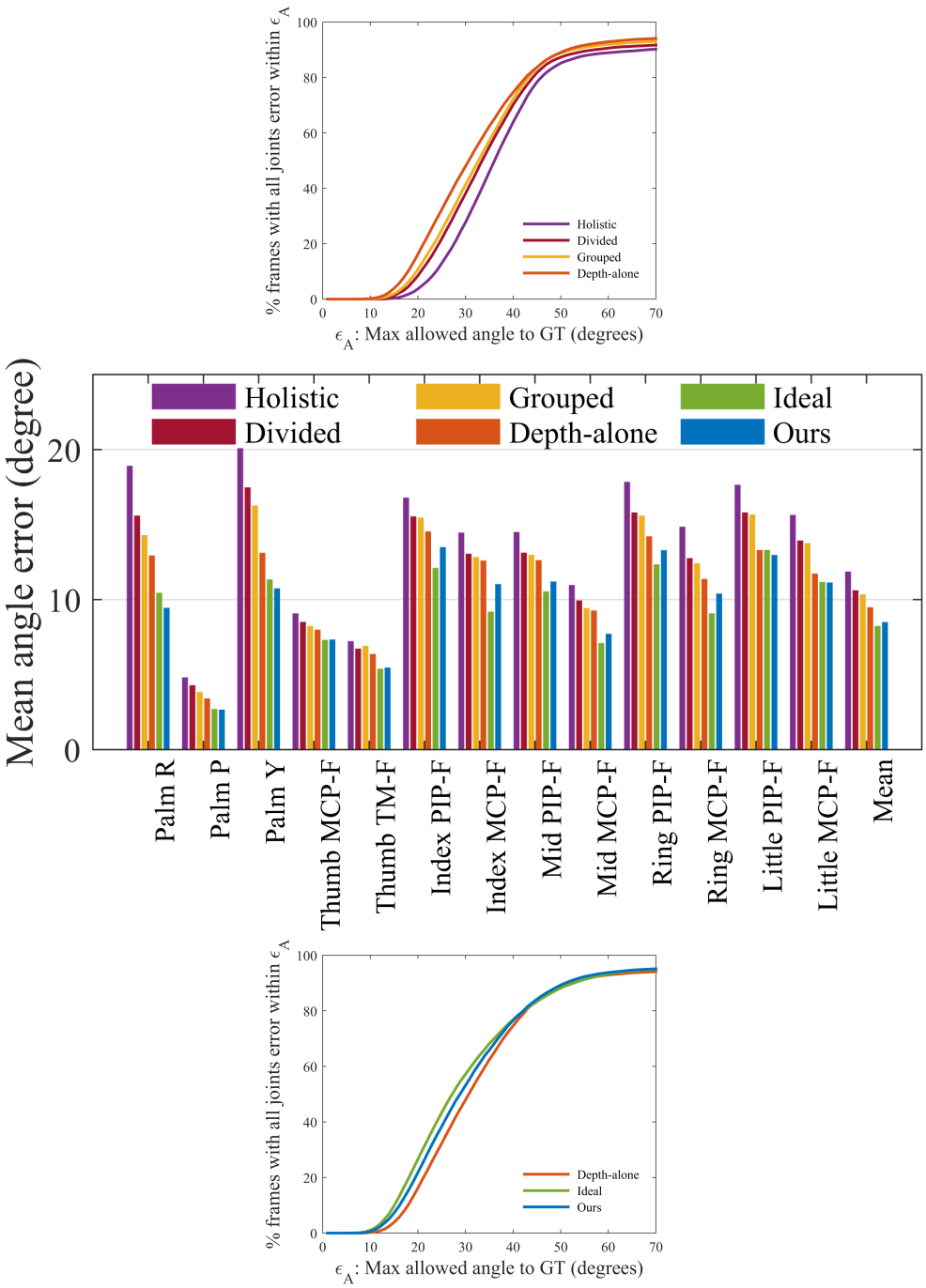


Figure 6.7. Quantitative evaluation of our method with repect to the self-generated baselines.

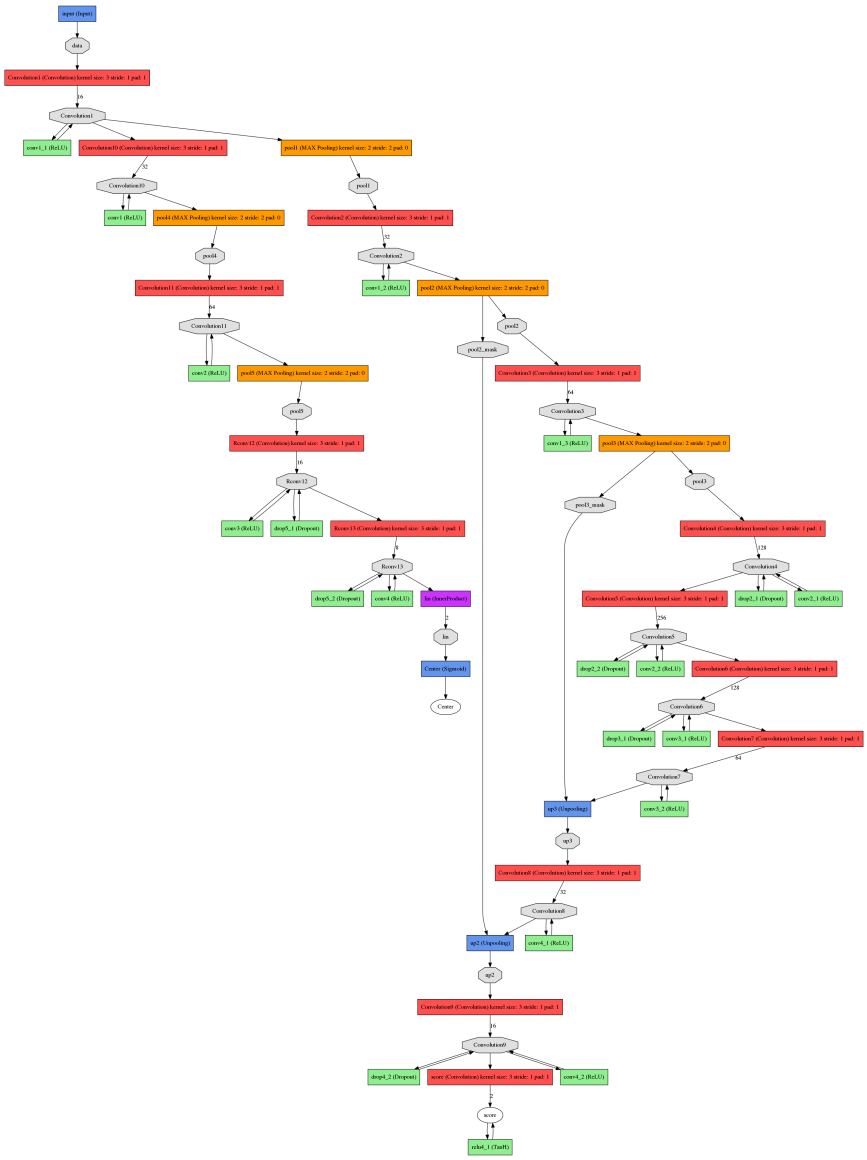


Figure 6.8. The graph of the proposed localization network architecture. There are two streams: the segmentation stream for pixel-wise hand segmentation, and the regression stream for hand center regression.

6.3.2 Implementation specifications

The proposed system was trained with GPUs using the Caffe framework [89]. We trained the localization network by setting the learning rate to 0.0005 and the number of epochs to 250 using the Adam optimizer[3] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. In addition, our depth network was converged after 80 epochs with the learning rate 0.01 for 60 epochs and 0.001 afterward. Our heat distribution network converged after 250 epochs. Here, we used the learning rate 0.01 for 200 epochs and then dropped it by a factor of 0.1. The hallucination network converged after 100 epochs with the learning rate 0.01. Lastly, the refinement network converged after 170 epochs by dropping the initial learning rate (0.01) in every 60 epochs by a factor of 0.1. At runtime, the computation time for each frame is split as 2 ms for processing data (*i.e.*, depth normalization, resizing, and bounding box cropping), 0.7 ms for hand localization, 1.8 ms to estimate the joint angle parameters from the proposed system. The additional hardware specifications are as follows: Intel's Core i5-4690K, 32GBs RAM, NVIDIA's Geforce GTX 1070, and Intel's RealSense SR300.

6.4 Experiments

We conduct evaluations using a synthetic and public dataset to respectively validate our design choices and the performance compared to the state-of-the-arts.

6.4.1 Datasets

We first introduce a self-generated synthetic dataset. This dataset is mainly used to evaluate our design choices. As discussed in Section 3, we use a hand model with 21 DOFs to render realistic hand poses with motion constraints. In the same manner, we collect the other 30K depth images with ground truth labels $\mathcal{Y}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{D})$. Note

[3]The rest of the networks used the SGD optimizer.

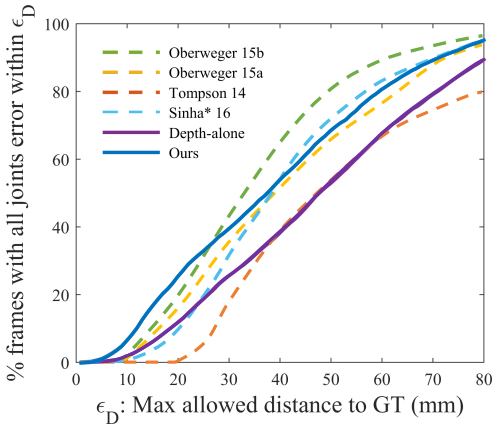
that we do not generate heat distribution descriptors **D** because the heat distribution responses are extracted from depth data at test time.

Additionally, we use public datasets (NYU [3] and MSRA14 [4]) to compare the performance of our approach to the state-of-the-art methods. Two datasets are collected from different camera types across contexts. Specifically, the NYU dataset involves a continuous sequence of hand movements acquired at far-range, whereas the MSRA14 dataset contains various gesture types of 6 individuals captured from close viewpoints. Note that our system requires the joint angle parameters for training and testing. Thus, we compute the ground truth angles of these datasets using the inverse kinematics as proposed in [34].

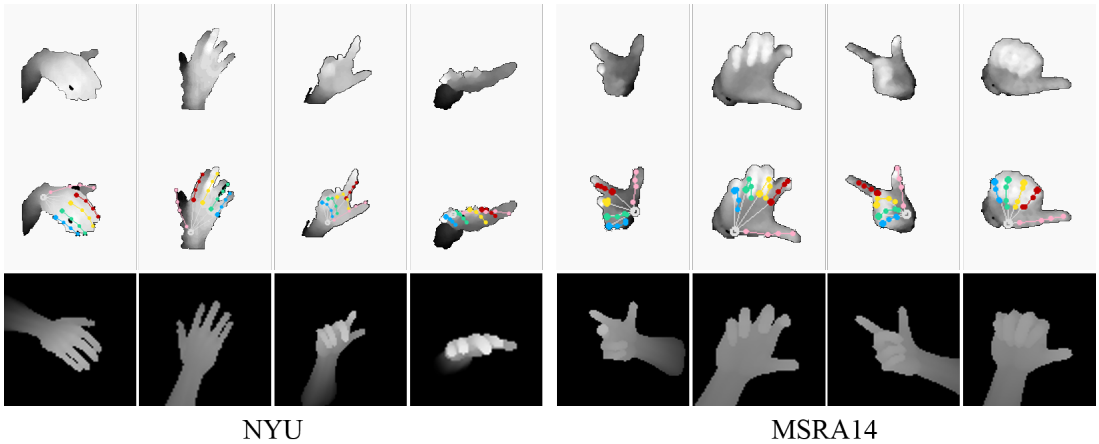
6.4.2 Comparison to baselines

Why multi-task learning? We demonstrate the rationale for using the multi-task approach to estimate the joint angle parameters from the depth network. We first define four baselines: (i) *Holistic*, which estimates all joint parameters using a single network; (ii) *Divided*, which divides the *Holistic* baseline into six sub-tasks (Thumb-Index-Middle-Ring-Little-Global) after the fifth convolutional layer; (iii) *Grouped*, which groups the fingers (T-IMR-L) according to their anatomical position and also separates the global network from the finger network; and (iv) *Depth-alone*, where we set the loss weights of the *Grouped* baseline as discussed. Figure 6.7 (top) quantitatively compares these baselines on a synthetic dataset, where we measure the robustness of each baseline. The performance of the *Holistic* baseline is dramatically improved by simply adopting the multi-task learning approach, and it is further enhanced by grouping fingers together, as the *Grouped* achieves higher regression accuracy than does the *Divided* baseline. This indicates that the network model learns a structural correlation across fingers to anatomically constrain hand configurations. While training the network, the loss of the thumb finger fluctuated more and converged faster than did the other losses. We thus scale the loss function of the thumb

by setting $\alpha = 3$ so that the contribution of the thumb will be 3 times larger than that from the *Depth-alone*. In this way, we achieve even better performance, as also demonstrated from the individual mean angle error in Figure 6.7 (middle). This comparison validates the rationale of our use of the multi-tasking approach as opposed to other choices.



(a)



(b)

Figure 6.9. Performance evaluations on the overall robustness. (a) Quantitative evaluation is conducted using the NYU dataset [3]. (b) Qualitative evaluations using the NYU and MSRA14 [4] dataset. The first row shows the input depth image, and the estimated poses are visualized in the second row. The third row shows pose reconstruction based on our estimates.

Why regularize the initial estimation? Furthermore, we explore the efficacy of the proposed refinement process. Figure 6.7 (bottom) shows the quantitative evaluation of our method with respect to the following baselines: (i) *Depth-alone* estimates of the joint angles with the aforementioned settings and (ii) *Ideal* estimates where geometric features are directly extracted from the HDN to eliminate the effect of hallucination error. The fact that our method performs better than the *Depth-alone* baseline validates the efficacy of the RN. The individual mean angle error (see Figure 6.7 [middle]) shows consistent results. These results also indicate that the fingertip positions guide the initially estimated angle parameters to be more accurate. Overall, our approach resulted in comparable performance to *Ideal* or even higher accuracy for the global orientations and the little finger in terms of the mean joint angle error, demonstrating that our hallucination network is well-trained to mimic heat distribution features in detail.

6.4.3 Comparison with the state of the arts

Quantitative evaluation Figure 6.9a quantitatively compares the performance of our approach with the state-of-the-art methods using a publicly available NYU dataset [3]. The maximum allowed joint distance error is examined in terms of the distance threshold ϵ_D. Here we observe that the overall performance of the *Depth-alone* baseline (purple line) is greatly improved in *Ours* (blue line) by hallucinating geometric features and penalizing the initial predictions. Moreover, our approach achieves performance higher than that of the state-of-the-art methods [3, 6, 77] over all the ranges. It further demonstrates that the better estimation model can be built by learning complementary information from a different input modality. Our method also shows comparable performance to the generative approach [7] with a higher fraction of frames that have Euclidean error less than 27 mm. It indicates that our approach performs better with a smaller error tolerance.

Table 6.1. Quantitative comparison (in mm) of our approach with the state-of-the-arts (generative methods [1, 4] and discriminative method [8]) on the MSRA14 dataset [4].

Sub.	1	2	3	4	5	6	Avg.
[1]	35.4	19.8	27.3	26.3	16.6	46.2	28.6
[4]	8.6	7.4	9.8	10.4	7.8	11.7	9.2
[8]	30.1	19.7	24.3	19.9	21.8	20.7	22.7
Ours	17.6	15.2	26.4	16.9	26.6	17.5	20.0

We additionally show the comparison of our approach to the generative methods [1, 4] and discriminative [8] method using the MSRA14 dataset [4]. For this, we follow the cross-dataset experiment proposed in [8]. We finetune our network models using the MSRA15 dataset [33] to measure the averaged distance error (in mm) of the palm and five fingertips from the MSRA14 dataset. In Table 7.6, we observe that the discriminative methods (ours and [8]) show lower accuracy than that of the generative method [4]. For this, we share similar insights with [8] as follows: (i) the discriminative methods neither incorporate temporal information between frames nor use a manual initialization in the first frame and (ii) the hand is not calibrated or scaled for each subject, which is crucial to reduce errors. However, the proposed method mostly outperforms [1] and [8] as it achieves a lower error rate. Thus, we conclude that the use of geometric representations as an additional modality results in more robust hand pose estimation.

Qualitative evaluation We conduct qualitative evaluations of our method using the NYU and MSRA14 dataset. The second row in Figure 6.9b illustrates hand poses estimated from the depth images in the first row. In addition, we provide the corresponding hand reconstructions in the third row, demonstrating that our approach enforces kinematically valid hand configurations. Although the fourth column of the NYU input image has missing pixels (see the fingertips), our method robustly predicts the hand pose without using temporal information.

6.5 Conclusion and Future Work

We address two important elements that have been missing in the current hand pose estimation approaches: (i) the understanding of geometric properties of the articulated hand and (ii) the use of an additional input modality to produce more informative representations. To incorporate these factors into the pose estimation system, we present a multi-scale heat distribution descriptor specifically designed to encode the local geometry as well as the global structural features of the hand. This descriptor is used to learn the convolutional responses, and our system hallucinates them using a corresponding depth image. Consequently, we use the geometrically informed features together with the discriminative depth representations extracted from the depth network to accurately estimate hand articulations. The extensive evaluations conducted using both the synthetic and real dataset validate the robustness of the proposed approach as we achieve performance higher than or comparable to the state-of-the-art methods.

7. LEARNING HAND FROM A MANIPULATING OBJECT

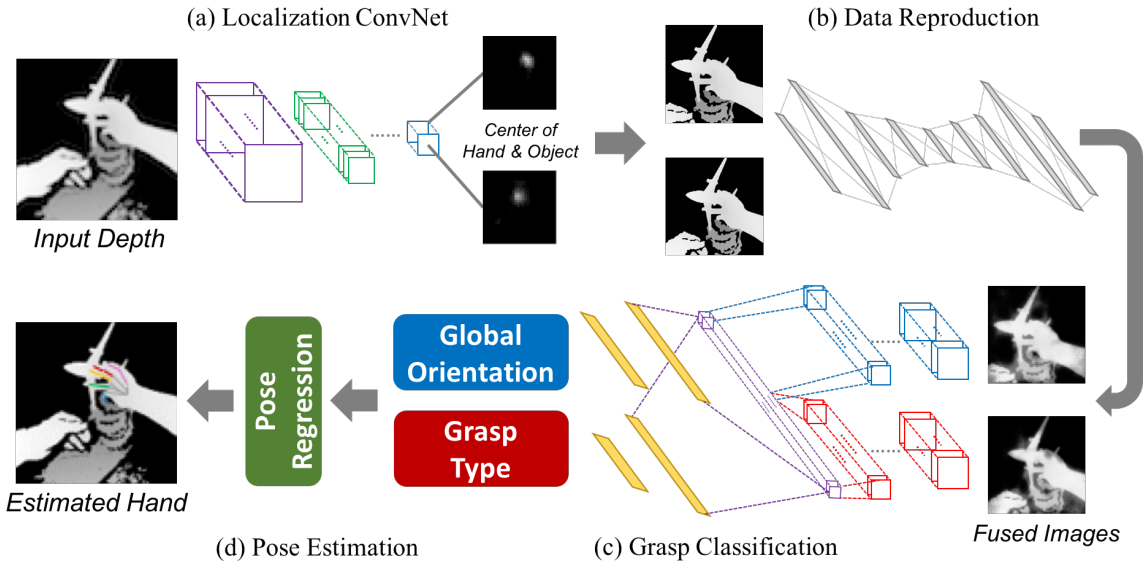


Figure 7.1. An overview of the proposed approach. (a) The localization ConvNet takes a depth image as input to predict the heatmaps of the hand and object center. (b) The reproduction network generates the informative fused images for grasp classification. (c) Our system collaboratively classifies both the global orientations of the hand and grasp type using the paired images. (d) Then, pose regression is applied to estimate the pose parameters of the hand.

Real-time depth data acquisition from commercial sensors has helped to simplify the tasks for hand pose estimation over the last decade. Although extensive research has been conducted on finding a robust and efficient solution for kinematic pose estimation of an isolated hand [1–4, 31, 33, 38, 67, 71, 77, 90], the problem of the hand's interactions with a physical object is barely considered in the literature. The current approaches allow the user to manipulate a known object and a simple primitive shape such as a cylinder or cuboid. Therefore, these solutions do not work consistently with

general human-computer interaction interfaces and augmented reality applications during natural interactions.

Hand pose estimation during the interaction with an unknown object is a challenging problem due to (i) the loss of hand information caused by partial or full object occlusions, (ii) the complicated shape of the unknown object and articulated nature of the hand, (iii) global 3D rotations, and (iv) the noise in acquired data, which confounds continuous estimation. In this paper, we present a new framework to effectively resolve these issues by collaboratively learning deep convolutional features from a hand and object perspective. Our fundamental observation from earlier work [23, 24] is that the interacting object can be a source of constraint on hand poses. In this view, we employ pose dependency on the shape of the object to learn discriminative features of the hand-object interaction.

The traditional approaches for pose estimation start with segmenting hand and object regions using RGB data followed by running an SVM classifier [46] or pixel-wise part classification [47] using hand-crafted features. A convolutional neural network (ConvNet) has recently been adopted to replace the hand-crafted features in [48], but this approach only aims for grasp classification. In contrast to these methods, we introduce a simultaneous training of deep neural networks for hand pose estimation. As a first step, we localize both the hand and object position using a ConvNet architecture. Specifically, we show that predicting the positions in the form of the heatmaps is an efficient way of overcoming the use of simple heuristics such as color-based segmentation or known object initialization.

We leverage the paradigm of *analysis by synthesis* and create a population of everyday human grasps. Similar to [48], the scope of hand-object interactions includes daily activities captured from an *egocentric* viewpoint. We adopt a 33-class taxonomy [91] to focus more on the shape of the hand grasp rather than the grasping motion [92]. The hand-object interactions are effectively mesh modeled with the corresponding hand pose parameters and grasp class labels. Although these synthetic depth images are easily simulated and accurately annotated, they do not explore artifacts (*e.g.*,

noise and distortion) of real data captured from 3D sensors [77]. Thus, we design a fully unsupervised learning architecture to generate reconstructed data based on the idea of signal reconstruction in autoencoders. The output images are used to extract grasp features encoded in pairs - one from a hand perspective and the other from an object perspective. To this end, we validate the use of two input sources (*i.e.*hand and object), in the context of grasp classification and consequently for hand pose estimation.

7.1 Hand–Object Localization

In this section, we first discuss a creation of our synthetic dataset that simulates the hand interacting with an object. Then we present our pragmatic solution to extract a center position of both the hand and the object for later use.

7.1.1 Synthetic dataset

3D hand Our hand model has a structure similar to that of the 21 DOFs kinematic mesh broadly used for hand pose estimation [71, 77]. We additionally construct the 2 DOFs lower arm to independently model the arm segment rotations, which helps to identify the global hand orientation, thus regularizing the jitter of the estimated pose [38]. Our training dataset simulates hand-object interaction from an entire egocentric viewpoint by rotating 3 wrist angles $\boldsymbol{\theta^W} = \{\theta_r^W, \theta_p^W, \theta_y^W\}$ where $\theta_r^W \in [-60, 60]^\circ, \theta_p^W \in [-90, 90]^\circ, \theta_y^W \in [-10, 50]^\circ$. These rotational ranges are further quantized into the 48 orientation classes ($4 \times 6 \times 2$).

3D CAD models We collect 3D mesh models of 600 daily objects that can be easily obtained online[1] and are freely downloadable. Our object models are all rigid shapes and we only explicitly determine the contact points of each object for the specific grasp.

[1]3D ContentCentral (https://www.3dcontentcentral.com) and GrabCAD (https://grabcad.com)

Dataset creation Manual simulation of hand-object interaction from different individuals is an unsupervised and time-consuming task that cannot even guarantee the annotation quality of the grasps. Along this line, we employ a model fitting method to optimize hand grasps with respect to the shape of the target objects. For this, particle swarm optimization is used to minimize the distance error between the observed object and our 3D hand model. Although this generative method guides the objective function to best fit the observed data, it might be susceptible to a collision of two geometric shapes (*i.e.*, the intersection of two triangular meshes). Therefore, we adopt a technique of collision detection to quickly determine if the grasp state is invalid. Details of collision detection are skipped for brevity, and we refer the readers to [93]. In practice, our approach reaches realistic object grasps and outputs the corresponding joint angle parameters of the hand with the grasp class label. We then insert these rendered depth maps into the cluttered background captured in-the-wild using Intel's RealSense F200, very similarly to [46]. This process is used not only to mimic an everyday environment for our simulated interaction but also to generalize our deep neural network - in particular, to handle the sensitiveness to diverse background perturbations. In total, we generate 330K synthetic depth maps. They are rendered from 33 grasps in terms of 40 objects (on avg.), 48 wrist rotations, and 5 populations per grasp[2].

7.1.2 Localization network

A heuristic method [4, 36, 77] to extract the region of interest cannot work consistently with general human-computer interaction applications. Hence, we train a ConvNet model to regress the confidence map (*i.e.*, the heatmap) of the center for the hand and object model (see Figure 7.2). Our fully convolutional network is comprised of six convolutional layers followed by a nonlinear layer. Furthermore, a final Euclidean loss layer computes the sum of squares of differences between the predicted

[2] 33 grasps $\times$ 40 objects $\times$ 48 rotations $\times$ 5 populations $\approx$ 330K

Table 7.1. The design of a ConvNet for heatmap regression. (Conv: convolutional layer, Pmax: max pooling layer, ReLU: rectified linear units layer, L2: Euclidean loss layer)

	Layers	# Kernels	Filter size	Stride	Pad
1	Conv	16	5×5×1	1	2
2	ReLU				
3	Pmax			2	0
4	Conv	32	5×5×16	1	2
5	ReLU				
6	Pmax			2	0
7	Conv	64	5×5×32	1	2
8	ReLU				
9	Pmax			2	0
10	Conv	128	5×5×64	1	2
11	ReLU				
12	Conv	256	5×5×128	1	2
13	ReLU				
14	Conv	2	5×5×256	1	2
15	ReLU				
16	L2				

Table 7.2. Accuracy comparison of hand localization on our synthetic dataset.

Model	Error	Settings
Ours	6.7 *pixels*	9 Epochs
RF [3]	27.6 *pixels*	22 Depth, 70 Trees

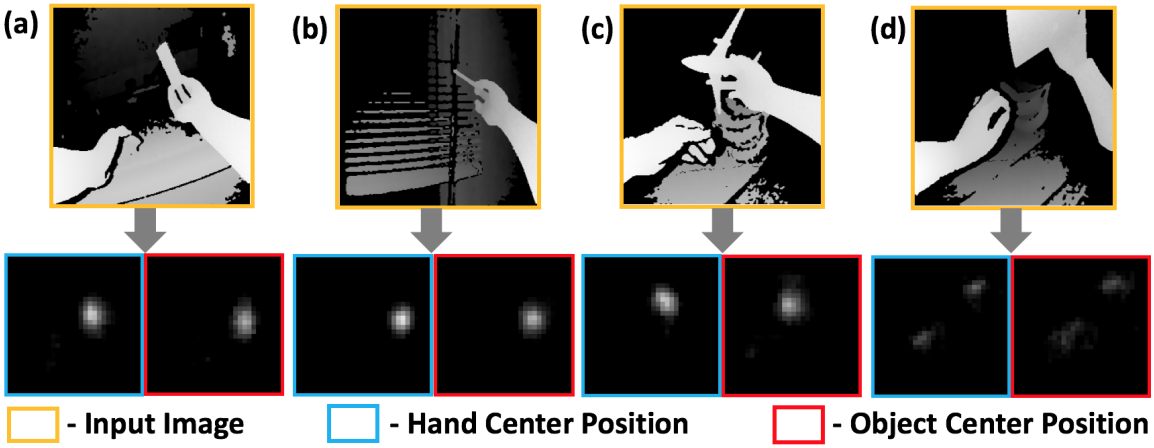


Figure 7.2. The heatmap regressor successfully segments the center points within contact regions for the hand and the object respectively (a)∼(c). The performance is lower in special cases such as introducing another hand in the scene (d).

heatmap and ground truth, as shown in Table 7.1. Even though the use of additional layers slightly increases estimation accuracy, the performance improvement is trivial compared to a significant increase in computation requirement.

Table 7.2 shows the quantitative comparison with a random forest (RF) classifier used in [3] which performs pixel-wise hand segmentation. Here, we first compute a centroid of segmented hand pixels and calculate the error in *pixels* from a centroid of ground truth. In contrast, our heatmap regressor directly outputs the position of the hand center and significantly outperforms the RF-based approach from localization accuracy.

Data Processing The depth values of input depth map D_m are first normalized to the range of [0, 255] to generate depth image D_i, and then we rescale D_i to

width of 240. The rescaled depth image D_r of size 240×240 is fed into localization ConvNet. The network outputs two 30×30 heatmaps corresponding to the centroid of the hand and the object, respectively. Next, we up-sample these heatmaps with a scaling factor 8 and then rescale to width of 320 so that the size to be the same as the original depth map D_m. The maximum value in each heatmap marks the hand centroid $\{u_h, v_h\}$ and the object centroid $\{u_o, v_o\}$. Note that the depth value of these points $d_m^h = \{u_h, v_h, d_h\}$ and $d_m^o = \{u_o, v_o, d_o\}$ can be obtained from the original depth map D_m. We use d_m^h and d_m^o to generate 64×64 depth images D_i^h and D_i^o centered at the hand/object centroid. The above process is detailed in the supplementary material.

7.2 Reproduction of Realistic Dataset

One observation obtained from quantitative evaluations from earlier work [77] is that the system of *analysis by synthesis* showed different aspects depending on the type of dataset. They evaluated their approach using synthetic and realistic datasets for self-comparison and comparison with the state-of-the-art, respectively. However, the system showed much better performance using a synthetic dataset. Even though [77] tried to mimic the actual sensor image by adding a Gaussian noise, there exists a gap between the two to be further improved. To address it, we propose a framework that allows the datasets to learn the attributes across domains instead of heuristically adding artifacts to the datasets or removing artifacts from them.

7.2.1 Synthesizing data by reconstruction

Our system is trained on a synthetic dataset that is virtually simulated with 3D mesh models. Although this approach is attractive because it allows the system to be applied to a range of sensor types, we might lose a certain degree of accuracy compared to the case when the same dataset type is used for both training and testing. Therefore, we generate synthesized real data based on the idea of signal

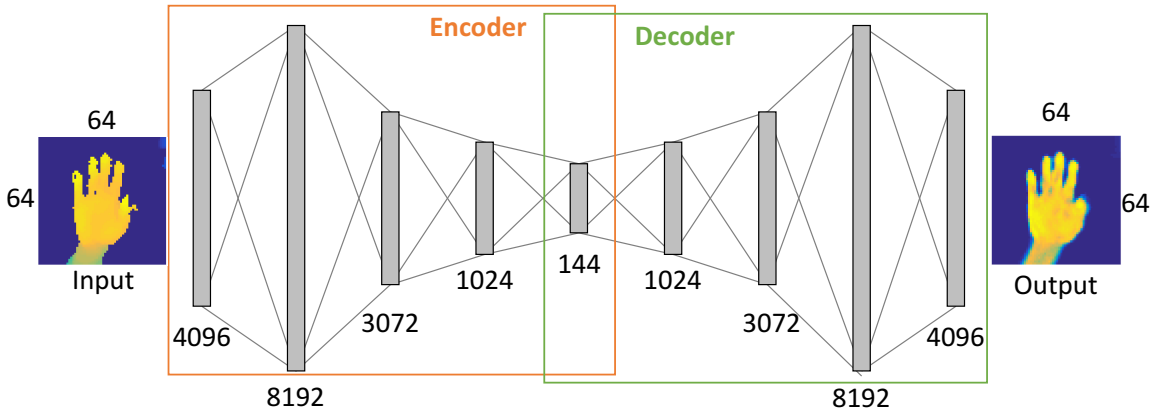


Figure 7.3. Overall architecture of the proposed data reproduction network.

reconstruction in autoencoders. The autoencoders try to predict the missing part from the non-missing values to recover original data. Our insight is that the loss of real data can be better represented by imposing the *repairing* process of an autoencoder. For this, we train our model to reconstruct pixel-level artifacts of the input depth, D_i^h and D_i^o.

In hand tracking literature, a synthesizer is proposed to correct the error of initial estimation in [94]. The initial pose estimation is used to generate a synthesized hand depth image, and the updater predicts an updated hand pose using both input data and the synthesized model in a closed loop. For this, they trained three different ConvNet models using a set of annotated training pairs. In contrast to this work, our approach differs as follows: (i) our method is unsupervised and we do not require any training pairs between real and synthetic data; (ii) we re-generate the synthesized depth image in a single shot without using inefficient iterations; and (iii) pixel-level noise and artifacts are tractable by encoding the input data and mapping back to the original data.

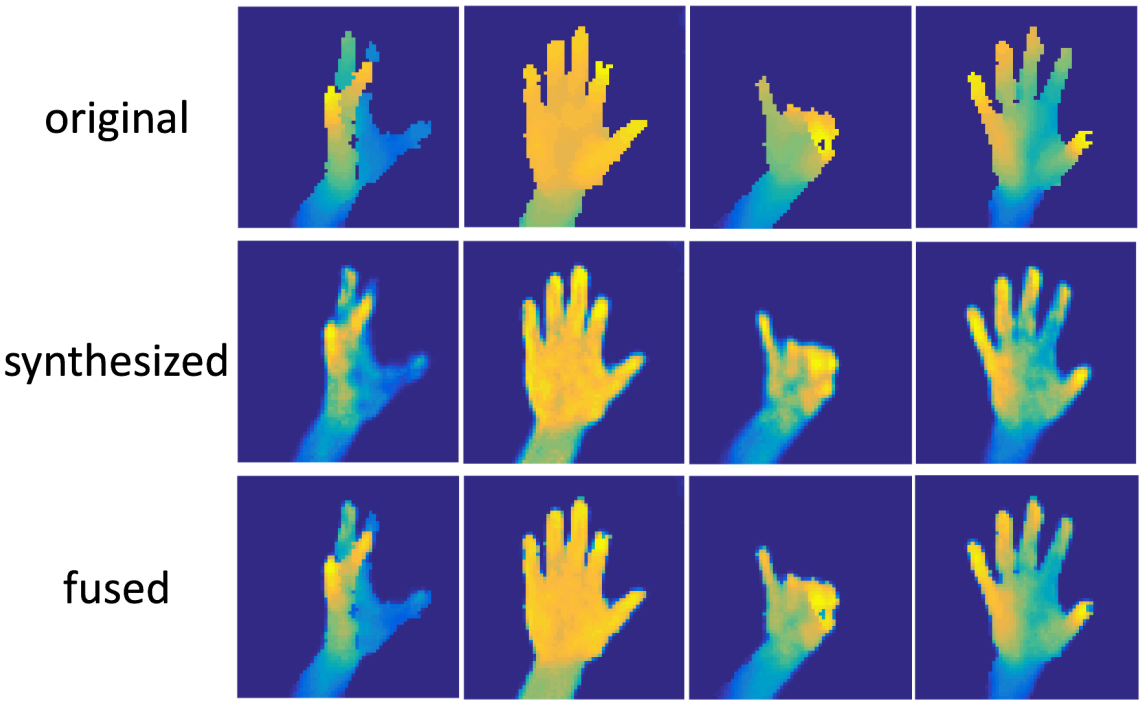


Figure 7.4. Visual comparison for data synthesis on the selected depth images of NYU dataset [3]. First row: the original depth images. Second row: the synthesized images using our framework. Third row: spatially fused images.

7.2.2 Reproduction network

Our system follows the traditional autoencoder framework which consists of two components, an *encoder* and a *decoder*. The encoder tries to reduce the dimensionality of the input by mapping high-dimensional data into a lower dimensional feature space, whereas the decoder recovers the original input by mapping back the learned representation into a high-dimensional space. The overall specification of our data reproduction network is displayed in Figure 7.3. We impose four hidden layers followed by a nonlinear function (sigmoid layer) for both the encoder and the decoder. The proposed network is trained on the 240K depth images captured across sensor types[3] and converged after 20 epochs.

[3]80K synthetically rendered images + 160K real depth images (80K captured from PrimeSense & 80K from Intel's RealSense F200.)

In Figure 7.4, three data types are visually compared. The top row shows the *original* 64×64 depth images (D_i^h and D_i^o) selected from an NYU dataset [3] for proof of concept. The second row shows the corresponding *synthesized* images generated using our reproduction network. We note that pixel-wise artifacts (*e.g.*, holes or missing pixels) of the *original* images are eliminated from the *synthesized* images by the reconstruction process of the network. However, a new compression distortion is observed from the palm regions of the *synthesized* images. To further eliminate such distortions, we spatially *fuse* the depth images by averaging the input (*original*) and output (*synthesized*) images. This is a simple yet effective strategy to improve the overall performance. The improvement of classification accuracy (37.75% to 41.00% in Table 7.4) on the *fused* images (D_f^h and D_f^o) demonstrates the impact of the averaging process. We discuss more details with empirical validation in Section 6.

7.3 Hand Pose Estimation

In this section, we discuss the importance of grasp classification for hand pose estimation and introduce our robust estimation framework of hand poses during the interaction with an unknown object.

7.3.1 Grasp classification

The partial or full loss of hand information during the interaction with hands cannot be recovered particularly when unknown objects are introduced. Instead of processing low-level data to recover or remove the region of object occlusions, we draw a ConvNet framework to extract informative expressions of grasps from those regions. We assume that there is a strong relation between the shape of the object and the configuration of the hand poses in the context of hand grasp. Thus, our model collaboratively learns the convolutional features about grasps from a hand and object perspective in pairs by sharing intermediate representations between two networks in the feature space.

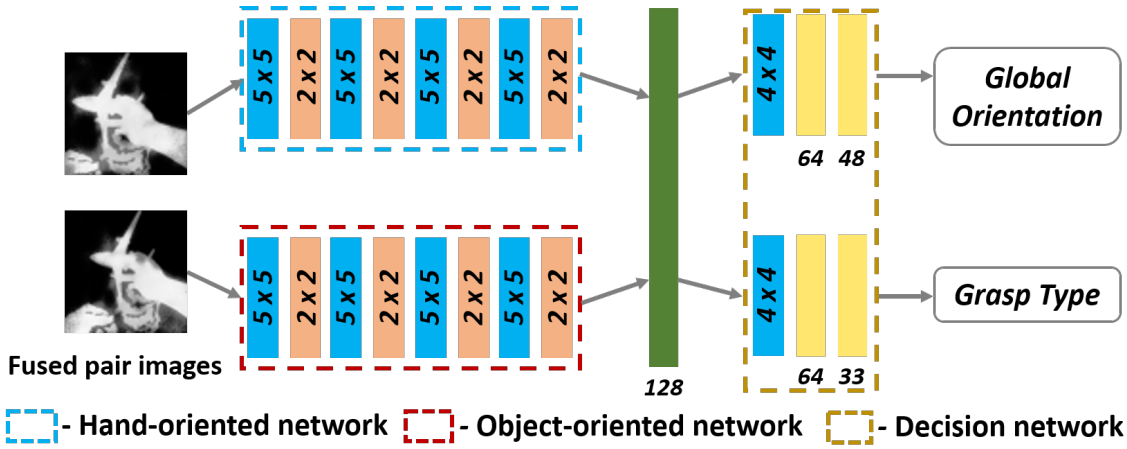


Figure 7.5. The architecture of proposed grasp classification network. Given fused pair images, each image is passed through distinctive networks to classify both hand's global orientation as well as grasp type. Color codes: Blue = Conv+ReLu, orange = Pmax, green = concatenation, yellow = Fully connected layer (ReLU exists between fully connected layer).

Grasp classification is a key factor for hand pose estimation in the presence of an external object interacting with hands. We achieve a good initialization of the hand pose for per-frame pose estimation, leading to robustness to occlusions and flexibility to unknown objects.

Details of our network structure are shown in Figure 7.5. The *fused* 64×64 image pair (D_f^h and D_f^o) from the previous step is now used as input to this model. Each network independently learns discriminative representations from different perspectives: the hand-oriented network focuses on the loss of hand information caused by occlusions due to the object, while the object-oriented network extracts potential pose information even from the unseen object. Each feature map of size $4 \times 4 \times 64$ independently extracted after the fourth convolutional layer is then concatenated as a tensor of size $4 \times 4 \times 128$. This step is important to transfer knowledge about a perceptual set of attributes such as hand/object occlusions, shape, or silhouette learned from different domains. This vector is further used to estimate the pose parameters in the next subsection.

7.3.2 Pose estimation

Although the ConvNet-based hierarchical classification strategy is effective for finding unknown pose parameters [77], it is computationally inefficient to train every five networks corresponding to each of the 144 global bins. Our pose estimation method is inspired by a 2-stage hierarchical strategy, but we do not estimate the global parameters from stage 1. Instead, we only constrain the pose configuration space using the possible hand orientations and a grasp type likely to be a set of good initializations. Once we identify the reduced subset, then we evaluate all the pose parameters in an all-in-one approach in stage 2 from this space.

The decision network (5th convolutional layer and the following fully connected layers in Figure 7.5) first classifies the top 5 orientations using the softmax function. Our rationale for classifying the orientation of the hand is as follows: the overall performance of hand pose estimation becomes deterministic based on the robustness of pose initialization [33,38], and the majority of the pose error is associated with the global orientation of the hand in practice. We subsequently classify the top 1 grasp type from the same network. Then we identify a reduced subset (*i.e.*, 1 grasp$\times$40 objects$\times$5 orientations$\times$5 populations $\approx$ 1K) from our 330K training images. An additional 64-dimension feature vector $\mathbf{f_2}$ is extracted in the penultimate layer of the orientation decision network, which contains discriminative cues sufficient to classify the global orientation of the hand. Finally, we perform a nearest neighbor search from the restricted space to retrieve l poses similar to the input hand pose. In practice, we observe that the use of more neighbors does not effectively increase the overall performance but introduces a computational bottleneck.

Our regression method aligns with the collaborative learning approach [71,77] to predict the pose parameters. Let $n = 64$ be a dimensionality of the feature vector, $m = 18$ be a number of joint angles, and $l = 32$ be a number of nearest neighbors, then

the matrices $\mathbf{F_1} \in \mathbb{R}^{l \times n}$, $\mathbf{f_2} \in \mathbb{R}^{1 \times n}$, $\mathbf{P_1} \in \mathbb{R}^{l \times m}$, and $\mathbf{p_2} \in \mathbb{R}^{1 \times m}$ are the submatrices of M:

$$\mathbf{M} = \begin{bmatrix} \mathbf{F_1} & \mathbf{P_1} \\ \mathbf{f_2} & \mathbf{p_2} \end{bmatrix}, \tag{7.1}$$

where $\mathbf{F_1}$ is the feature vectors of neighboring poses, $\mathbf{f_2}$ is the feature vector of the current pose, $\mathbf{P_1}$ is the joint angles of neighboring poses, and $\mathbf{p_2}$ is the unknown angles to be regressed. We compute $\mathbf{p_2}$ using MacDuffees theorem:

$$\mathbf{p_2} = \mathbf{f_2}(\mathbf{F_1})^{+}\mathbf{P_1}, \tag{7.2}$$

where + denotes the Moore-Penrose pseudo-inverse. The proof of the above process is detailed in [77].

7.4 System Specifications

We present details of our network design and the system specifications.

7.4.1 Architecture of localization ConvNet

We visualize neuron activations of our localization ConvNet in Figure 7.6 to validate our network structure. Our network is mainly comprised of six convolutional layers. Through Conv1 to Conv6, neurons are activated nearby edges of foreground (*i.e.*the hand and object). As shown in Figure 7.6, the network shows higher confidences to the center of the hand and object (see Conv4-conv5). Then, it outputs two clear heamaps corresponding to their centers after Conv6. Note that the left and right image of Conv6 shows the center position of the hand and the object, respectively.

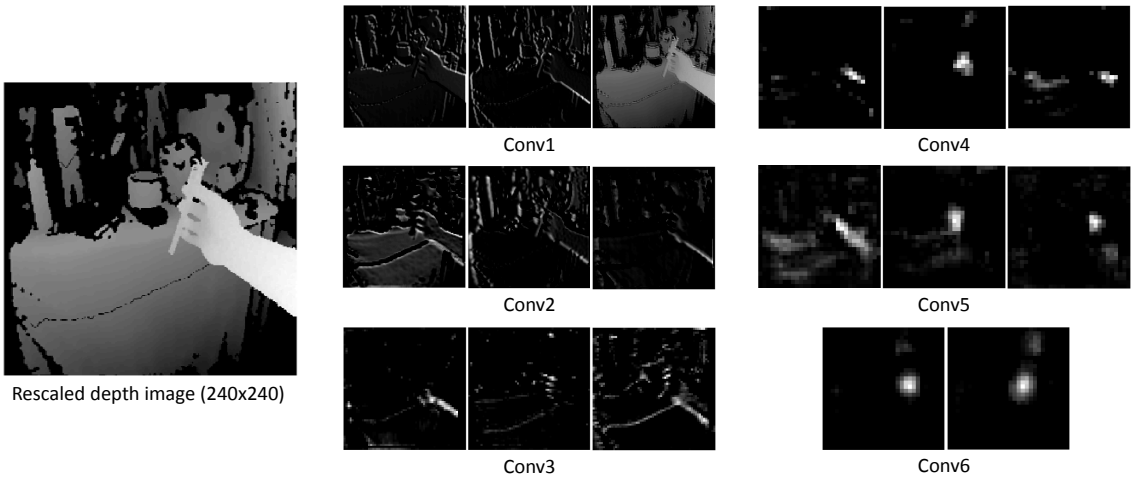


Figure 7.6. Neuron activations are presented for each convolutional layer. We randomly select three feature maps and resized for only visualization purpose. The outputs of Conv6 are two-channel 30×30 heatmaps that represents a confidence of the hand/object center position.

Table 7.3. Details of ConvNet architectures. We present four different configurations which differ in the order of feature concatenation (ConvNet B and D) and the number of FC layers (ConvNet C and D) from our grasp classification network (ConvNet A). Hand: hand-oriented network, Object: object-oriented network, Orientation: orientation decision network, Grasp: grasp decision network.

ConvNet Architecture							
ConvNet A		ConvNet B		ConvNet C		ConvNet D	
Hand	Object	Hand	Object	Hand	Object	Hand	Object
conv5-16	conv5-16	conv5-16	conv5-16	conv5-16	conv5-16	conv5-16	conv5-16
conv5-32	conv5-32	conv5-32	conv5-32	conv5-32	conv5-32	conv5-32	conv5-32
conv5-32	conv5-32	conv5-32	conv5-32	conv5-32	conv5-32	conv5-32	conv5-32
conv5-64	conv5-64	conv5-64	conv5-64	conv5-64	conv5-64	conv5-64	conv5-64
		conv4-128	conv4-128			conv4-128	conv4-128
Concatenation							
conv4-128	conv4-128			conv4-128	conv4-128	FC-64	
FC-64	FC-64	FC-64	FC-32	FC-64	FC-32	FC-32	FC-32
FC-48	FC-33	FC-48	FC-33	FC-48	FC-32	FC-48	FC-33
					FC-33		
Softmax							
Orientation	Grasp	Orientation	Grasp	Orientation	Grasp	Orientation	Grasp

7.4.2 Architectures for grasp classification

To find a best ConvNet configuration for classifying global orientations and grasp types, we evaluate different ConvNet architectures as outlined in Table 7.3. Our grasp classification network is named as ConvNet A, and its structure consists of five convolutional layers followed by a max pooling and nonlinear (ReLU) layer and two fully connected (FC) layers with a nonlinear layer at the end.

We test three distinctive configurations (ConvNet B-D) which differ in the order of feature concatenation and the number of FC layers from ConvNet A. Figure 7.7 shows the effect of the proposed variations in terms of loss and accuracy. ConvNet B concatenates the feature maps after the fifth convolutional layer. As a result, both the hand-oriented network and object-oriented network independently process the data by convolving 128 4×4 kernels with the output feature map of the fourth convolutional layer. However, this step makes our decision function less discriminative. In ConvNet C, we expected better performance in grasp classification with additional FC-32 layer for the grasp type decision network. However, we observed slower loss convergence and lower accuracy comparing to ConvNet A (Figure 7.7c and 7.7d). Still, ConvNet C exhibits higher performance than ConvNet B and D because the convolutional layer after concatenation locally extracts more better representations. For ConvNet D, although we put an additional FC-64 layer for more expressive feature extraction in the decision network, the performance drops due to reduced dimensionality of learned features.

7.5 Experiments

We conduct extensive evaluations to verify our design choices for localization and grasp classification as well as hand pose estimation. To demonstrate the efficacy of our approach, we compare the results of testing our method and a state-of-the-art method using a public dataset and of testing our self-generated baselines using a synthetic dataset.

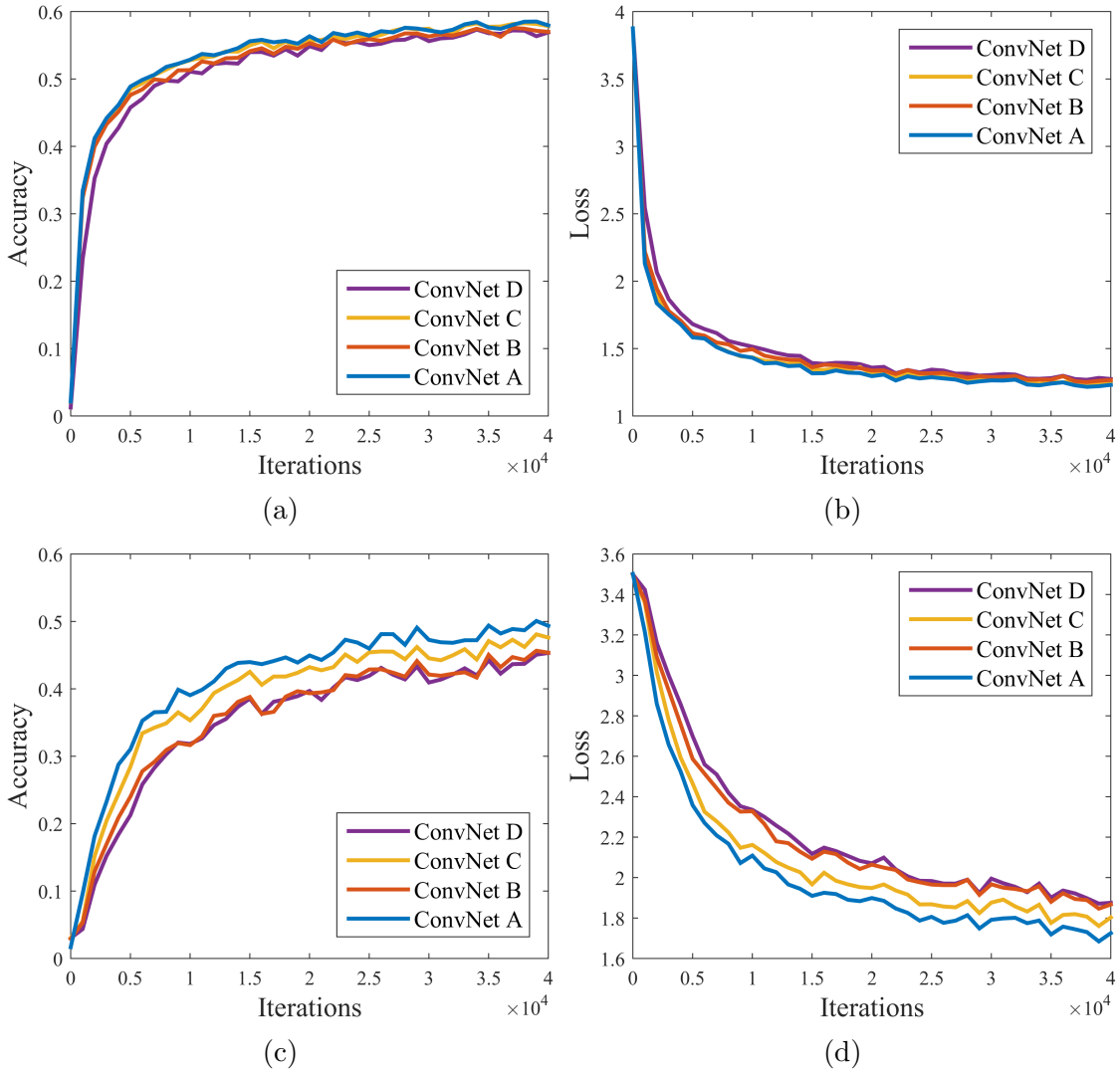


Figure 7.7. Loss and accuracy while training the network models. (a) and (b): Accuracy and loss of the orientation decision network. (c) and (d): Accuracy and loss of the grasp decision network.

7.5.1 Datasets for comparison

The size of our synthetic dataset is 16.5K; it is comprised of 500 depth maps per grasp randomly rendered from different objects, orientations, and backgrounds. This dataset is used for comparison with self-generated baselines (described below) to validate our design choices. Since we aim to achieve 3D hand pose estimation, our dataset is fully annotated with the grasp numbers, orientation labels, joint angle parameters, and joint positions in 3D.

For localization and grasp classification, we additionally evaluate using a publicly available GUN-71[4] dataset [48]. It was captured in-the-wild from eight subjects covering 28 everyday objects per grasp with various *egocentric* views. Since the grasp type is labeled on a per-frame basis, it is suitable to evaluate the performance of the proposed reproduction network and grasp classification approach with respect to grasp recognition accuracy.

Although we are aware of the publicly available hand-object datasets in the literature [47, 95], we do not use them for evaluations. As we discussed in Section 3.1, our hand-object interactions are simulated from an *egocentric* viewpoint which differs from their interaction ranges. In addition, their software is not publicly available so we evaluate the quantitative/qualitative performance of our method using the synthetic dataset and the publicly available GUN-71.

7.5.2 Analysis of design choices

Experiments on public dataset To demonstrate the efficacy of the proposed data reproduction process, we individually train nine models using different types of dataset from scratch. We first define three types of training and test datasets: (i) *Original* denotes the original depth images (D_i^h and D_i^o) obtained as a result of localization; (ii) *Synthesized* is a set of images outputted from the reproduction net-

[4]Although GUN-71 dataset contains 71 grasps, we only use the common 33 grasps ($\approx$ 6K depth maps).

Table 7.4. Grasp classification results for 33 grasps evaluated on GUN-71 dataset [48]. The use of reproduction network (spatially fused) improves overall classification results. Note that *Train* denotes the type of training dataset used to train our model and *Test* denotes the format of GUN-71 dataset used for testing our networks.

Train set \ Test set	Original GUN-71	Synthesized GUN-71	Fused GUN-71
Original	39.75%	16.87%	31.71%
Synthesized	32.86%	37.75%	36.51%
Fused	36.43%	29.31%	**41.00%**

Table 7.5. Accuracy comparison of grasp classification on GUN-71 dataset.

Model	Classification accuracy
Rogez et al. [48]	20.50 %
Original	39.75 %
Synthesized	37.75 %
Ours (*Fused*)	**41.00 %**

Table 7.6. Classification accuracy for the orientation of the hand and the grasp type. *Hand only* achieves higher performance to orientation classification than *Object only* but has less impact on grasp classification.

Network	*Hand-only*	*Ojbect-only*	**Ours**
Orientation Acc.	59.31%	51.12%	**60.50%**
Grasp Acc.	43.87%	49.12%	**55.56%**

work; (iii) *Fused* indicates the images (D_f^h and D_f^o) obtained by spatially averaging the *Original* and *Synthesized* data. The experimental result is shown in Table 7.4. The best performance (accuracy of 41.00 %) is achieved when the network is trained using the spatially fused images and tested on the same type of dataset. It validates that training and testing with *Fused* data allows the extraction of more expressive representations of data while minimizing depth artifacts. Interestingly, the model that is trained and tested using the *Synthesized* data shows poorer performance than the model that is trained and tested using the *Original* data. Here we observe that the higher accuracy may not be accomplished by simply synthesizing the depth images because the reproduced dataset could explore a new distortion, as also shown in Figure 7.4 (second row). Subsequently, Table 7.5 compares the performance of our grasp classification method to that of [48]. Note that the accuracy of [48] is directly captured from their paper. All our methods significantly outperform their deep feature-based SVM grasp classifier by a huge margin. This comparison validates the rationale of our specific approach against other choices.

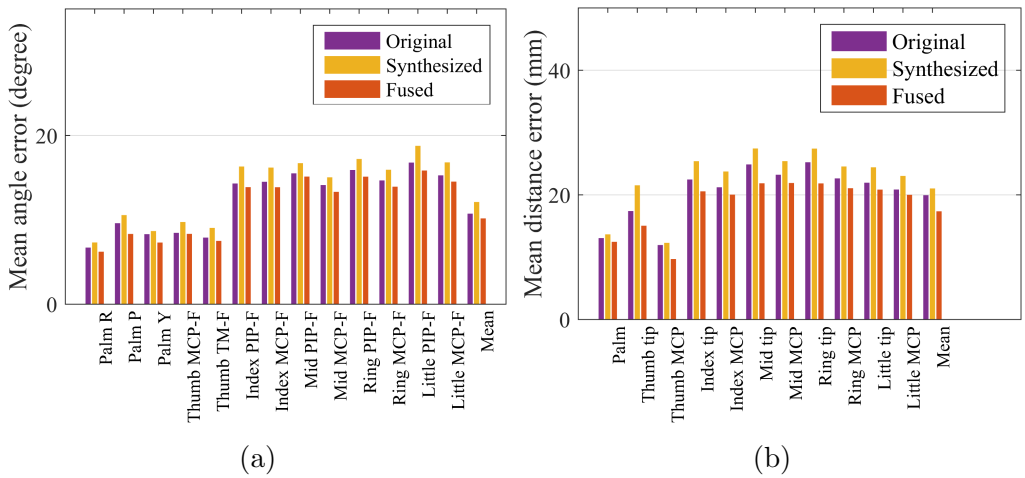
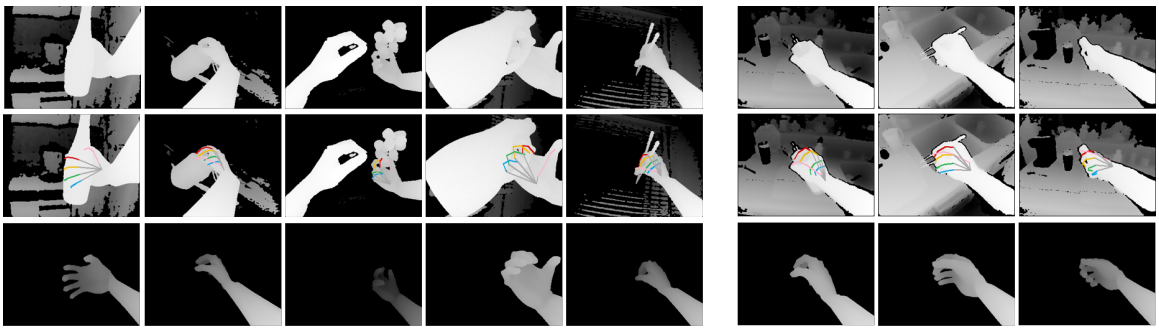


Figure 7.8. Quantitative evaluation on the overall robustness. (a) The individual mean joint angle error is used to compare the performance of the proposed method and baselines (in degrees). (b) Accuracy of hand pose estimation is examined as a function of the averaged joint distance (in mm) error.

Experiments on synthetic dataset We conduct more ablative tests that demonstrate the efficacy of our two-stream (the hand and object stream shown in Figure 7.5) orientation/grasp classification network. For this, we compare our two-stream network to two additional baselines by conducting tests with a synthetic dataset: (i) with only the hand stream (*Hand-only*) and (ii) with only the object stream (*Object-only*). Table 7.6 shows the performance of these baselines relative to our proposed approach. As expected, the *Hand-only* stream performs better to classify the orientation of the hand, whereas the *Object-only* stream achieves higher accuracy for grasp type classification relative to the *Hand-only* stream. It implies that the *Hand-only* stream extracts more beneficial information about the configuration of the hand. The *Object-only* stream focuses more on the shape of the object, which infers hand grasp. The proposed two-stream strategy outperforms these two baselines by extracting informative representations from both streams. It validates that constructing the two-stream network is critical to good performance.



(a) (b)

Figure 7.9. Qualitative evaluations are conducted on (a) our synthetic dataset and (b) publicly available GUN-71 dataset. The first row shows the input depth image, and estimated hand skeletons are presented in the second row. The third row shows the reconstructed hand mesh model from skeleton estimation.

7.5.3 Evaluation for pose estimation

Quantitative evaluation We validate the proposed framework for hand pose estimation using our own synthetic dataset. Figure 7.8a shows the averaged angle error (in *degrees*) over all frames for each joint position. We observe that the error of the *Synthesized* (12.11) and *Original* (10.73) data is higher than that of the *Fused* (10.17) data all over the joint positions. It validates the rationale of the proposed data reproduction process. The consistent result is drawn in Figure 7.8b which presents the averaged distance error for each joint. Again, the use of the *Fused* images outperforms the others over an entire range, validating our choice is overall more robust for pose estimation. In particular, the fact that the distance error of our palm position is less than average indicates our localization network well performs on the cluttered background in the presence of unseen objects.

Qualitative evaluation We conduct a qualitative evaluation of our approach using our synthetic dataset and publicly available GUN-71 dataset [48]. The top row of Figure 7.9a shows the input depth frames rendered using our 3D hand and object models. Note that the cluttered background was captured in-the-wild using a commercial depth camera. The second row shows the hand pose estimates using our

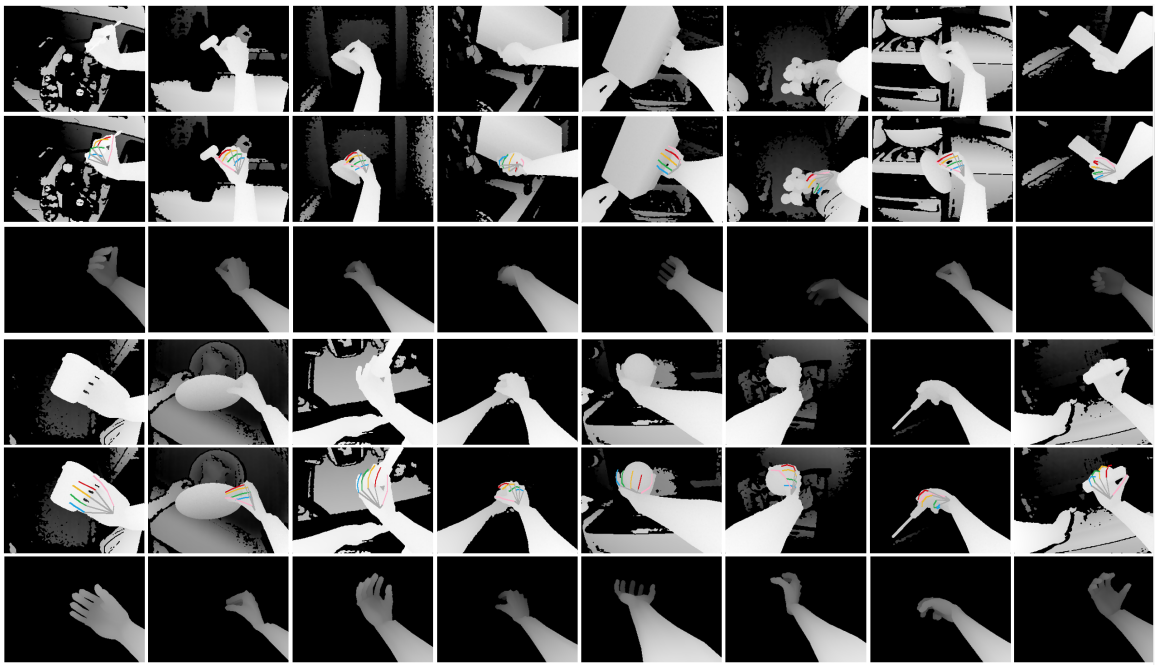


Figure 7.10. Additional qualitative evaluations using a synthetic dataset. The first (fourth) row shows the input depth image, and estimated hand skeletons are presented in the second (fifth) row. The third (sixth) row shows the reconstructed hand mesh model from skeleton estimation.

framework. Finally, the reconstructed hand models are displayed in the third row. We observe that the proposed approach robustly estimates the valid and natural hand configurations against the severe object occlusions, various global orientations, and the cluttered background. Subsequently, the first row of Figure 7.9b shows the selected depth images of the GUN-71 dataset. Note that we use the first 33 classes of the GUN-71 dataset, which share the same grasp types with our dataset. The second and third row, respectively, shows the estimated poses and corresponding reconstruction based on our estimates. Figure 7.9 demonstrates that our approach performs robustly across input sources (*i.e.*, the data type and noise in acquired data)

We conduct additional qualitative evaluations of our approach using the synthetic dataset. The first and fourth row of Figure 7.10 shows the input depth images, and

the corresponding hand estimates are presented in the second and fifth row. Then we reconstruct hand models based on our estimation in the third and sixth row.

7.6 Conclusion and Future Work

We present a learning framework for hand pose estimation while interacting with an unknown object. Our main insight is that the shape of the object can be used to better represent the hand pose in the form of interactive grasps. By exploring their intimate relationship, more discriminative cues can be collaboratively derived from both perspectives. To generate a large database of the synthetic human grasps, we simulate 3D hand and CAD models. Using the dataset along with a ConvNet, we localize the center of the hand and object to create a pair of images. This pair is processed through the reproduction network to learn attributes of the synthetic images. We then classify the hand orientations and grasp type from the multi-channel network to reduce the search space for pose estimation. Finally, we compute the angle parameters from this subset. The evaluation results show that we achieve robust performance for both grasp classification and hand pose estimation. Future work will focus on varying attributes (*e.g.*, transparency) of the 3D object models and covering an entire camera viewpoint to reflect more realistic factors to our system.

8. CONCLUSION

In this chapter, we first summarize the presented computational learning frameworks designed for hand pose estimation. Then, we conclude the thesis by discussing future research directions that would further improve the performance of overall system.

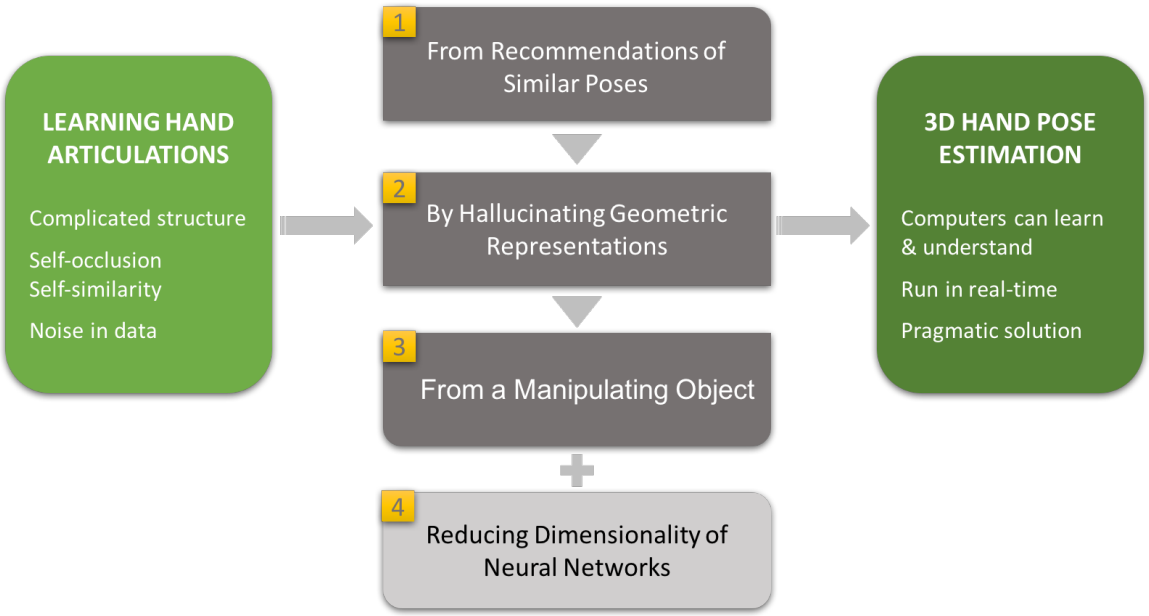


Figure 8.1. Computational learning for hand pose estimation at a glance.

8.1 Summary

Robust hand tracking is central to human-computer interaction interfaces and virtual/augmented reality applications. Although there exists robust and accurate methods for full body pose estimation, hand pose estimation is far more challenging due to (i) the articulation complexity of the hand, (ii) self-similarity and self-occlusion of the fingers, and (iii) data acquisition artifacts such as depth noise. Recent re-

searches have been directed toward identifying these issues, but these research efforts have provided a coarse interpretation of hand movements. Thus, we propose this thesis work to provide a robust and efficient solution to the 3D hand pose estimation problem, which enables us to employ a full 3D interpretation of hand movements for human–computer interaction interfaces.

This thesis starts from an introduction to our main insights that lead us to come up with novel ideas in Chapter 1. Subsequently in Chapter 2, we review relevant literature on 3D hand pose estimation, hand-object interaction, and additional machine learning techniques that we adopted for solving a pose estimation problem. A description for our 3D hand mesh model and our effors to render natural and realistic hand poses is explained in Chapter 3.

Collaborative filtering aims to predict unknown user ratings in a recommender system by collectively assessing known user preferences. In Chapter 4, we first drew analogies between collaborative filtering and the pose estimation problem. Specifically, we recasted the hand pose estimation problem as the cold-start problem for a new user with unknown item ratings in a recommender system. Inspired by fast and accurate matrix factorization techniques for collaborative filtering, we developed a real-time algorithm for estimating the hand pose from RGB-D data of a commercial depth camera. First, we efficiently identified nearest neighbors using local shape descriptors in the RGB-D domain from a library of hand poses with known pose parameter values. We then used this information to evaluate the unknown pose parameters using a joint matrix factorization and completion (JMFC) approach. Our quantitative and qualitative results suggested that our approach is robust to variation in hand configurations while achieving real time performance ($\approx$ 29 FPS) on a standard computer.

We proposed DeepHand in Chapter 5 to estimate the 3D pose of a hand using depth data from commercial 3D sensors. We discriminatively trained convolutional neural networks to output a low-dimensional activation feature given a depth map. This activation feature vector is representative of the global or local joint an-

gle parameters of a hand pose. We efficiently identified spatial nearest neighbors to the activation feature, from a database of features corresponding to synthetic depth maps, and store some temporal neighbors from previous frames. Our matrix completion algorithm useed these spatio-temporal activation features and the corresponding known pose parameter values to estimate the unknown pose parameters of the input feature vector. Our database of activation features supplemented large viewpoint coverage and our hierarchical estimation of pose parameters was robust to occlusions. We showed that our approach compares favorably to state-of-the-art methods while achieving real time performance ($\approx$ 32 FPS) on a standard computer.

In Chapter 6, we proposed a robust hand pose estimation method by learning hand articulations from depth features and auxiliary modality features. As an additional modality to depth data, we presented a function of geometric properties on the surface of the hand described by heat diffusion. The proposed heat distribution descriptor is robust to identify the keypoints on the surface as it incorporates both the local geometry of the hand and global structural representation at multiple time scales. Along this line, we trained our heat distribution network to learn the geometrically descriptive representations from the proposed descriptors with the fingertip position labels. Then the hallucination network was guided to mimic the intermediate responses of the heat distribution modality from a paired depth image. We used the resulting geometrically informed responses together with the discriminative depth features estimated from the depth network to regularize the angle parameters in the refinement network. To this end, we conducted extensive evaluations to validate that the proposed framework is powerful as it achieves state-of-the-art performance.

Chapter 7 proposed a robust solution for accurate 3D hand pose estimation in the presence of an external object interacting with hands. Our main insight is that the shape of an object causes a configuration of the hand in the form of a hand grasp. Along this line, we simultaneously trained deep neural networks using paired depth images. The object-oriented network learns functional grasps from an object perspective, whereas the hand-oriented network explores the details of hand configu-

rations from a hand perspective. The two networks shared intermediate observations produced from different perspectives to create a more informed representation. Our system then collaboratively classified the grasp types and orientation of the hand and further constrained a pose space using these estimates. Finally, we collectively refined the unknown pose parameters to reconstruct the final hand pose. To this end, we conducted extensive evaluations to validate the efficacy of the proposed collaborative learning approach by comparing it with selfgenerated baselines and the state-of-the-art method.

In Appendix A, we proposed a compressive technique for deep neural networks to learn more informative representations of data. Our approach was flexible to add more layers and go deeper in the deep learning architecture while keeping the number of parameters the same.

8.2 Future Directions

Although the presented frameworks produce a better prediction model for given hand articulations, there still exists an opportunity to further improve the performance of hand pose estimation. We provide future directions in this section to address some limitations in the current framework.

8.2.1 Realistically rendered synthetic hand images

The presented hand pose estimation approaches use synthetically rendered depth images to train the network models. These synthetic depth images can be easily simulated and accurately annotated, enabling us to avoid human efforts. However, they do not explore artifacts (e.g., noise and distortion) of real data captured from 3D depth sensors and hence experience data dependent performance. Although we developed a reproduction network to ease these issues as shown in Chapter 7, our approach do not directly create real images using synthetic images. Instead, we generate a new type of images, which we call *Fused Images*, using both real and

synthetic images. Along this line, the first step of future work will be to further improve this algorithm so the system can directly find noise aspects of input data and implicitly handle the given issues without creating explicit depth images.

8.2.2 Incorporating RGB images as an input

The current state-of-the-art methods including our approaches only use depth images to train the prediction models. This is because RGB images have more variations in terms of the color of the hand, intensity of the scene, and light changes. Creating the dataset which includes all possible variation has been considered as an impossible mission in the computer vision area. However, we showed a possibility of using synthetic images to build a state-of-the-art model throughout this thesis work. In this view, we can create as many training images as possible using our 3D pose simulator by incorporating different colors of the hand and light and intensity of the scene. The system will be more robust if we train a deeper network using this additional input modality as RGB images help to ease ambiguity of low-resolution depth images.

8.2.3 Hand personalization

We revealed one of the limitations of our approaches, that is scalability of the hand as shown in Chapter 6. Although our method achieved performance higher than that of the state-of-the-art discriminative method, it showed lower accuracy than that of the generative method. As we discussed our insights, the major issue is that the hand is not calibrated or scaled for each subject. Our prediction models were trained using the synthetically rendered depth images, and synthetic images were created using a 3D virtual hand model. Thus, one of the potential directions on hand personalization is that we create a database using multiple hand models by statistically generating them. Furthermore, we can directly personalize the hand model that is used for visualization of the pose estimates. In the beginning of the system, we use few frames

to measure a hand shape and the rest frames for hand pose estimation. This setup will enable to develop a more robust estimation framework.

8.2.4 Use of unstructured point cloud

The presented approaches do not fully take advantage of the given depth data as we first convert depth maps to depth images. Although each pixel of depth maps contains a measure of a distance between the camera and the reflector in mm, we discretize and quantize these values to align a range of $[0, 255]$ and finally $[0, 1]$. It is convenient to feed depth images into the network with this step but resulting in the loss of information that seems critical to good performance. Although we can think of using the original depth values to fit in a range of $[0, 1]$, then the depth changes on the surface of the hand will be trivial. Thus, we can take into account the process of unstructured point cloud in the deep neural network framework. Although treating a point cloud as a voxel grid is straightforward to implement, these methods are limited with low-resolution. Therefore, we can consider to adopt an idea of 2D geometric images [98] flattened from a 3D structure to create one-to-one mappings between the 2D renderings and given poind cloud.

8.2.5 Embedding sensing techniques

Commercial virtual reality devices use a physical controller as their input tool to help interactions between a human and computer. This is because (i) the current hand pose estimation techniques are not robust enough to use in a practical scenario and (ii) the physical input devices are more comfortable for people to conduct certain interacting tasks in a virtual environment as they provide a feedback such as haptics. By focusing on these factors, we can embed sensing capability to improve the performance of 3D hand pose estimation. This concept is a lot different with the current hand pose estimation techniques that follow the markerless vison-based tracking framework. However, the actual sensors on top of the hand can ease these practical

issues for commercial devices. One way we can imagine is to collect a coordinate of hand joints using magnetic sensors [99] to regularize the estimates of the vision-based prediction model. The other way is to wear a soft, flexible, and stretchable sensor materials [100] to provide expanded interactions like physical input devices.

8.3 Closing Statement

In recent years, 3D hand pose estimation has become essential for people to interact with computers in a more natural and intuitive way. The goal of this thesis work is to design a concrete framework so the computers can learn and understand about perceptual attributes of human hands (i.e., self-occlusions or self-similarities of the fingers) and to develop a pragmatic solution to the real-time hand pose estimation problem implementable on a standard computer. In this way, human–computer interaction experiences will be further enhanced promising a realistic environment for gaming and entertainment. Along this line, we simultaneously developed a 3D hand pose estimation system by focusing on robustness, efficiency, and performance and solving the challenging issues on hand pose estimation. We introduced new and novel techniques to our domain and seamlessly applied to the given task, which could potentially open up new avenues in the computer vision community. In the future, we envision an efficient and robust framework building on these foundations where machines entertain and help humans.

REFERENCES

REFERENCES

[1] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, pages 1–11, 2011.

[2] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013*, pages 63–70. Canadian Information Processing Society, 2013.

[3] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (TOG)*, 33(5):169, 2014.

[4] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1106–1113. IEEE, 2014.

[5] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

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

[8] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3d hand pose estimation in single depth images: From single-view cnn to multi-view cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] C. Wan, A. Yao, and L. Van Gool. Direction matters: hand pose estimation from local surface normals. In *European Conference on Computer Vision*. Springer, 2016.

[10] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *International Conference on Intelligent User Interfaces*, pages 127–134, 2002.

[11] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *ACM RecSys*, pages 165–172, 2013.

[12] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100, December 2008.

[13] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[15] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, 2013.

[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[17] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[18] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas. One point isometric matching with the heat kernel. In *Computer Graphics Forum*, volume 29, pages 1555–1564. Wiley Online Library, 2010.

[19] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1, 2011.

[20] C. M. Christoudias, R. Urtasun, M. Salzmann, and T. Darrell. Learning to recognize objects from unseen modalities. In *European Conference on Computer Vision*, pages 677–691. Springer, 2010.

[21] L. Spinello and K. O. Arras. Leveraging RGB-D data: Adaptive fusion and domain adaptation for object detection. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4469–4474. IEEE, 2012.

[22] J. Hoffman, S. Gupta, and T. Darrell. Learning with side information through modality hallucination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 826–834, 2016.

[23] J. Romero, H. Kjellström, and D. Kragic. Hands in action: real-time 3d reconstruction of hands in interaction with objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 458–463. IEEE, 2010.

[24] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *2011 International Conference on Computer Vision*, pages 2088–2095. IEEE, 2011.

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

[26] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3d skeletal hand tracking. In *Graphics Interface*, pages 63–70, 2013.

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

[28] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision*, pages 852–863. 2012.

[29] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[30] D. Tang, T. H. Yu, and T. K. Kim. Real-time articualted hand pose estimation using semi-supervised transductive regression forests. In *IEEE International Conference on Computer Vision*, pages 3224–3231, 2013.

[31] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *CDC4CV*, pages 119–137. Springer, 2013.

[32] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3224–3231. IEEE, 2013.

[33] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun. Cascaded hand pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 824–832, 2015.

[34] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3325–3333, 2015.

[35] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1862–1869. IEEE, 2012.

[36] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly. Robust articulated-icp for real-time hand tracking. Technical report, 2015.

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

[38] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. K. C. R. I. Leichter, A. V. Y. Wei, D. F. P. K. E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust, and flexible real-time hand tracking. In *SIGCHI*, 2013.

[39] P. Krejov, A. Gilbert, and R. Bowden. Guided optimisation through classification and regression for hand pose estimation. *Computer Vision and Image Understanding*, 155:124–138, 2017.

[40] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3456–3462. IEEE, 2013.

[41] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*, pages 640–653. Springer, 2012.

[42] N. Kyriazis and A. Argyros. Physically plausible 3d scene tracking: The single actor hypothesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16, 2013.

[43] N. Kyriazis and A. Argyros. Scalable 3d tracking of multiple interacting objects. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3430–3437. IEEE, 2014.

[44] Y. Wang, J. Min, J. Zhang, Y. Liu, F. Xu, Q. Dai, and J. Chai. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)*, 32(4):43, 2013.

[45] P. Panteleris, N. Kyriazis, and A. A. Argyros. 3d tracking of human hands in interaction with unknown objects. In *BMVC*, pages 123–1, 2015.

[46] G. Rogez, J. S. Supancic, and D. Ramanan. First-person pose recognition using egocentric workspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4325–4333, 2015.

[47] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *European Conference on Computer Vision*, pages 294–310. Springer, 2016.

[48] G. Rogez, J. S. Supancic, and D. Ramanan. Understanding everyday hands in action from RGB-D images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3889–3897, 2015.

[49] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. *CoRR, abs/1504.04788*, 2015.

[50] L. Shi, S. Feng, et al. Functional hashing for compressing neural networks. *arXiv preprint arXiv:1605.06560*, 2016.

[51] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[52] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[53] T. M. Greiner. Hand anthropometry of US army personnel. Technical report, DTIC Document, 1991.

[54] J. Lin, Y. Wu, and T. S. Huang. Modeling the constraints of human hand motion. In *Proceedings of the Workshop on Human Motion*, pages 121–, Washington, DC, USA, 2000.

[55] J. Lee and T. Kunii. Model-based analysis of hand posture. *Computer Graphics and Applications, IEEE*, 15(5):77–86, Sep 1995.

[56] S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. *ACM Trans. Graph.*, 26(3), July 2007.

[57] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. *ACM Conference on Management of data*, pages 49–60, 1999.

[58] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM Conference on Knowledge discovery and data mining*, pages 226–231, 1996.

[59] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, pages 1508–1515. IEEE Computer Society, 2005.

[60] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *In ECCV*, pages 778–792. 2010.

[61] S. Suzuki and K. be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985.

[62] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(1):13–18, Feb 1979.

[63] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[64] R. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *IEEE International Conference on Data Mining*, pages 43–52, Oct 2007.

[65] A. N. Langville, C. D. Meyer, R. Albright, J. Cox, and D. Duling. Initializations for the nonnegative matrix factorization. In *Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 23–26. Citeseer, 2006.

[66] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graph.*, 33(5):169:1–169:10, September 2014.

[67] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3786–3793. IEEE, 2014.

[68] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of The 31st International Conference on Machine Learning*, pages 647–655, 2014.

[69] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[70] J. Lee and T. L. Kunii. Model-based analysis of hand posture. *Computer Graphics and Applications, IEEE*, 15(5):77–86, 1995.

[71] C. Choi, A. Sinha, J. Hee Choi, S. Jang, and K. Ramani. A collaborative filtering approach to real-time hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2336–2344, 2015.

[72] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1704–1716, 2012.

[73] A. B. Owen and P. O. Perry. Bi-cross-validation of the svd and the nonnegative matrix factorization. *The annals of applied statistics*, pages 564–594, 2009.

[74] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[75] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *Proceedings of the International Conference on 3D Vision (3DV)*, December 2014.

[76] M. J. Landau, B. Y. Choo, and P. A. Beling. Simulating kinect infrared and depth images. 2015.

[77] A. Sinha, C. Choi, and K. Ramani. DeepHand: Robust hand pose estimation by completing a matrix imputed with deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4150–4158, 2016.

[78] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3221, 2015.

[79] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[80] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[81] A. Wetzler, R. Slossberg, and R. Kimmel. Rule of thumb: Deep derotation for improved fingertip detection. In M. W. J. Xianghua Xie and G. K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 33.1–33.12. BMVA Press, September 2015.

[82] R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[83] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[84] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4460–4464. IEEE, 2015.

[85] C. Zhang and Z. Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1036–1041. IEEE, 2014.

[86] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

[87] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision*, pages 332–347. Springer, 2014.

[88] S. Li, Z.-Q. Liu, and A. B. Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 482–489, 2014.

[89] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[90] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, pages 852–863. 2012.

[91] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77, 2016.

[92] J. Liu, F. Feng, Y. C. Nakamura, and N. S. Pollard. A taxonomy of everyday grasps in action. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 573–580. IEEE, 2014.

[93] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical report, Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[94] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a Feedback Loop for Hand Pose Estimation. In *Proceedings of the International Conference on Computer Vision*, 2015.

[95] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016.

[96] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563. ACM, 2006.

[97] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015.

[98] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.

[99] S. H. Yoon, Y. Zhang, K. Huo, and K. Ramani. Tring: Instant and customizable interactions with objects using an embedded magnet and a finger-worn device. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 169–181. ACM, 2016.

[100] S. H. Yoon, K. Huo, Y. Zhang, G. Chen, L. Paredes, S. Chidambaram, and K. Ramani. isoft: A customizable soft sensor with real-time continuous contact and stretching sensing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 665–678. ACM, 2017.

APPENDIX

A. REDUCING THE DIMENSIONALITY OF NEURAL NETWORKS BY COMPRESSION

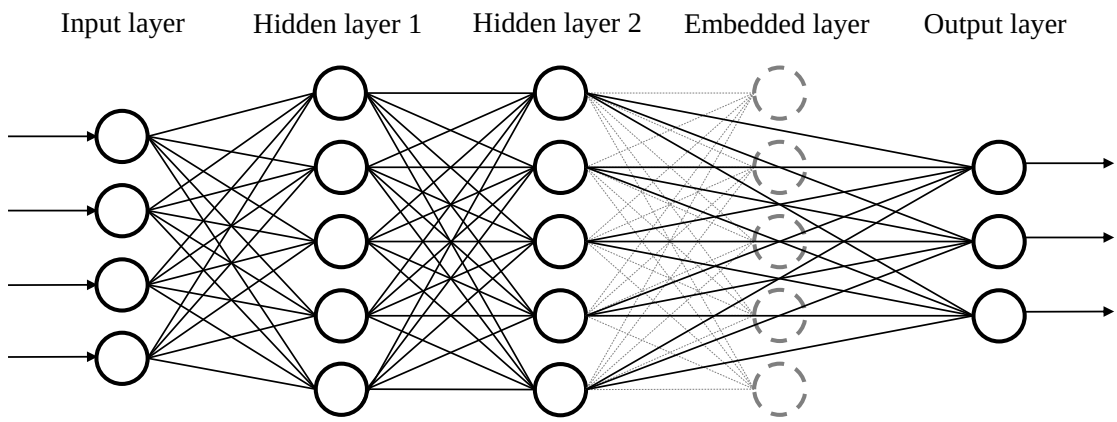


Figure A.1. The proposed compressive technique. At training time, the network parameters learned from the embedded layers are compressed using ramdom projection to preserve the number of parameters the same.

In this chapter, we propose a compressive technique for deep neural networks to learn more informative representations of data. Our approach is flexible to add more layers and go deeper in the deep learning architecture while keeping the number of parameters the same.

A.1 Compressive Neural Network

In random projection, the size or dimension of the random variables can be effectively reduced from a size m to k where $k \ll m$ by mapping high dimensional data into a lower dimensional space using a random matrix $\mathbf{J} \in \mathbb{R}^{k \times m}$. Let $\mathbf{E} \in \mathbb{R}^{m \times n}$ be

a set of n m-dimensional data. Then, the projection to a lower dimensional vector space can be $\mathbf{L} = \mathbf{JE}$, where $\mathbf{L} \in \mathbb{R}^{k \times n}$ is a lower dimensional matrix.

A.1.1 Johnson-Lindenstrauss lemma for dimensionality reduction

The JL lemma and its variants have shown a linear mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^k$ while preserving the pairwise distances of the data in the Euclidean space. Our compressive technique is highly inspired by the fact that the distance of a pair $\mathbf{e}_i, \mathbf{e}_j \in \mathbf{E}$ can be preserved with high probability and small distortion ϵ after projecting on a lower dimensional space,

$$\sqrt{\frac{k}{m}}\|\mathbf{e}_i - \mathbf{e}_j\|_2(1 - \epsilon) \leq \|f(\mathbf{e}_i) - f(\mathbf{e}_j)\|_2 \leq \sqrt{\frac{k}{m}}\|\mathbf{e}_i - \mathbf{e}_j\|_2(1 + \epsilon). \quad \text{(A.1)}$$

We extend this insight to embed a compressive layer that ouputs higher dimensional neurons and then compress the embedded layer to preserve the identical network architecture to the original network.

Assume the original network reduces k neurons to n neurons from the i-th hidden layer with a weight matrix $\mathrm{W}_i \in \mathbb{R}^{k \times n}$. Also assume that we are able to achieve better performance by embedding the hidden layers with two weight matrices $\mathrm{W}_i^{\mathrm{e}_1} \in \mathbb{R}^{k \times l}$ and $\mathrm{W}_i^{\mathrm{e}_2} \in \mathbb{R}^{l \times n}$, where $l > n$. Then, the computational cost increases from $\mathcal{O}(kn)$ to $\mathcal{O}(kl + ln)$ at test time. In practice, however, we might not be able to design such network from the embedded systems and platforms as they run with limited computational power and memory. Thus, we compress the embedded layer with $\mathrm{W}_i^{\mathrm{e}_1}$ into a matrix $\mathrm{W}_i^{\mathrm{c}} \in \mathbb{R}^{k \times n}$ (the same size as W_i) using the JL lemma to reduce the computational cost to $\mathcal{O}(kn)$ while achieving performance higher than the original network with W_i. The detailed process is as follows: (i) reshape $\mathrm{W}_i^{\mathrm{e}_1}$ into a matrix $\mathbf{E} \in \mathbb{R}^{m \times n}$ where m is an integer equal to $\frac{l}{n} \times k$; and (ii) calculate $\mathrm{W}_i^{\mathrm{c}} = \mathbf{JE}$ where $\mathbf{J}$ is the fixed transform matrix of size $k \times m$ where the pairwise distances of $\mathbf{E}$ are preserved in $\mathrm{W}_i^{\mathrm{c}}$ with high probability based on the JL lemma (Eqn. A.1).

Although the storage and computational costs of this process during training are both $\mathcal{O}(km)$ with the JL transform, they can be further reduced, in particular, to $\mathcal{O}(m \log m + \epsilon^{-3} \log^2 n)$ for computational time by the fast Johnson-Lindenstrauss transform (FJLT) [96]. The FJLT picks three real-valued matrices **PHD** of the form $\mathbf{J} = \mathbf{PHD}$, where $\mathbf{P} \in \mathbb{R}^{k \times m}$ is a sparse matrix with Gaussian random numbers, $\mathbf{H} \in \mathbb{R}^{m \times m}$ is a Hadamard matrix, and $\mathbf{D} \in \mathbb{R}^{m \times m}$ is a diagonal matrix with random ± 1 entries.

A.1.2 Backpropagation of compressive networks

Our compressive method learns the weight matrix $\mathbf{W}_i^{\mathrm{e}_1}$ and a bias term $\mathbf{b}_i \in \mathbb{R}^{1 \times n}$ during backpropagation. Let $C(\mathbf{W}, \mathbf{b})$ be a cost function (*i.e.*average sum of squared error) for the i-th layer. The bias is first updated using the gradient of C with respect to $\mathbf{b}_i$,

$$\mathbf{b}_i \leftarrow \mathbf{b}_i - \alpha \nabla_{\mathbf{b}_i} C(\mathbf{W}, \mathbf{b}), \tag{A.2}$$

where α is the learning rate. In contrast, we cannot directly update $\mathbf{W}_i^{\mathrm{e}_1}$ from the gradient of C because of its dimensionality. Instead, we find the low dimensional gradient of C with respect to $\mathbf{W}_i^c$ and then update the matrix $\mathbf{W}_i^{\mathrm{e}_1}$ as:

$$\mathbf{W}_i^{\mathrm{e}_1} \leftarrow \mathbf{W}_i^{\mathrm{e}_1} - \alpha \mathbf{J}^T \nabla_{\mathbf{W}_i^c} C(\mathbf{W}, \mathbf{b}). \tag{A.3}$$

We note that our technique guarantees the computational cost $\mathcal{O}(kn)$ with W_i^c at test time, despite the embedded layers are added into the architecture. The time complexity can be further decreased by incorporating the parameter compression methods such as [52, 97] into our approach.

Table A.1. The evaluations using the MNIST dataset. *Ref* is a model trained using the LeNet configuration of Caffe and *JL* is where we apply the proposed compressive (Comp) technique. The prefix *D* shows if we use the dropout technique to avoid overfitting. The output size denotes either the number of kernels (Conv) or the number of outputs (FC) embedded in between layers.

Model	Comp. layer	Output size	Error
Ref	-	-	0.87 %
JL-C2	$\text{Conv}_2^{e_1}$	250	0.85 %
JL-FC1	$\text{FC}_1^{e_1}$	6000	**0.75 %**
JL-C2-FC1	$\text{Conv}_2^{e_1}$ & $\text{FC}_1^{e_1}$	250 & 6000	1.02 %
D-Ref	-	-	0.70 %
D-JL-C2	$\text{Conv}_2^{e_1}$	250	**0.51 %**
D-JL-FC1	$\text{FC}_1^{e_1}$	6000	0.64 %
D-JL-C2-FC1	$\text{Conv}_2^{e_1}$ & $\text{FC}_1^{e_1}$	250 & 6000	0.57 %

A.2 Experiment on MNIST dataset

We evaluate the proposed technique using an MNIST dataset for proof of concept. We train a reference model (*Ref*) from scratch using the LeNet configuration implemented on the Caffe framework [89]. The network is comprised of two convolutional layers (Conv) with a max pooling layer and two subsequent fully connected layers (FC). The error rate of the reference model is 0.87 %. For comparison, we train two sets of network models using different configurations; the models with a label *JL* where we embed additional layers and reduce dimensionality by compression, and the prefix *D* denotes whether we use the dropout technique to prevent overfitting. The experimental results are shown in Table A.1. Note that all of network models preserve the same computational complexity with the *Ref* model at test time. The best performance is achieved from the model *JL-FC1* (the number of output neurons is 6000) with an error rate of 0.75 % without using dropout. It demonstrates that classification with the compressed parameters effects on extracting more expressive representations of data, and thus the proposed compressive network is able to achieve a fundamental goal of designing the deeper network architecture. Interestingly, we observe that the higher accuracy may not be accomplished by simply increasing the size of the embedded feature map or the number of layers, as the network architecture experiences overfitting (see *JL-C2-FC1*). Thus, we use the dropout layers to regularize the network in the following experiments. With an aid of the dropout layers, the model *D-JL-C2* (the number of kernels is 250) shows a minimum error rate of 0.51 %. It validates the rationale of our compressive technique to further improve the performance while preserving the same computational power.

A.3 Conclusion and Future Work

We propose a new technique to design a deeper network by compressing the embedded layers. Our main insight is that the high dimensional data can be mapped into the lower dimensional space while preserving the pairwise distances of the elements

using the JL transform. The evaluation results validate the efficacy of the proposed approach as we improve the classification accuracy.

VITA

VITA

Chiho Choi received a B.S. from Hanyang University, Korea in 2011, and a M.S. from University of Southern California, in 2013. His research interest lies at the intersection of machine learning and computer vision. He broadly builds machine learning algorithms in a practical way for computer vision systems. Choi worked as a research assistant in the Computational Design and Innovation Lab (C Design Lab) led by Professor Karthik Ramani, focusing on human shape interaction.

Chiho Choi will join Honda Research Institute in Mountain View, CA as a Scientist. His future research will include 3D computer vision, machine learning, deep learning, and human-computer interactions.

LIST OF PUBLICATIONS

LIST OF PUBLICATIONS

This thesis includes the following published work by the author (in reverse chronological order):

- C. Choi, S. Kim, and K. Ramani, Learning Hand Articulations by Hallucinating Heat Distribution. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp.3104-3113), 2017.
- C. Choi, S. H. Yoon, C. N. Chen, and K. Ramani, Robust Hand Pose Estimation during the Interaction with an Unknown Object. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp.3123-3132), 2017.
- C. Choi, S. Kim, J. H. Choi, and K. Ramani, Embedding Compressive Layers in Deep Neural Networks. *Technical Report*, 2017.
- C. Choi*, A. Sinha*, and K. Ramani, DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp.4150-4158), 2016.
- C. Choi, A. Sinha, J. H. Choi, S. Jang, and K. Ramani, A Collaborative Filtering Approach to Real-time Hand Pose Estimation. *n Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp.2336-2344). 2015.