3D Hand Pose Regression with Variants of Decision Forests

Danhang Tang

Electrical and Electronic Engineering

Imperial College London

Supervised by: Dr. Tae-Kyun Kim

This dissertation is submitted for the degree of

Doctor of Philosophy

October 2015

Abstract

3D hand pose regression is a fundamental component in many modern human computer interaction applications such as sign language recognition, virtual object manipulation, game control, etc. This thesis focuses on the scope of 3D pose regression with a single hand from depth data. The problem has many challenges including high degrees of freedom, severe viewpoint changes, self-occlusion and sensor noise.

The main contributions of this work are to propose a series of decision forest-based methods in a progressive manner, which improves upon the previous and achieves state-of-the-art performance is achieved in the end. The thesis first introduces a novel algorithm called semi-supervised transductive regression forest, which combines transductive learning and semi-supervised learning to bridge the gap between synthetically generated, noise-free training data and real noisy data. Moreover, it incorporates a coarse-to-fine training quality function to handle viewpoint changes in a more efficient manner. As a patch-based method, STR forest has high complexity during inference. To handle that, this thesis proposes latent regression forest, a method that models the pose estimation problem as a coarse-to-fine search. This inherently combines the efficiency of a holistic method and the flexibility of a patch-based method, and thus results in 62.5 FPS without CPU/GPU optimisation. Targeting the drawbacks of LRF, a new algorithm called hierarchical sampling forests is proposed to model this problem as a progressive search, guided by kinematic structure. Hence the intermediate results (partial poses) can be verified by a new efficient energy function. Consequently it can produce more accurate full poses. All these methods are thoroughly described, compared and published. In the conclusion part we discuss and analyse

their differences, limitations and usage scenarios, and then propose a few ideas for future work.

Keywords

decision forest, random forest, classification, regression, 3D pose estimation, human body, human hand, semi-supervised learning, transfer learning, transductive learning, latent tree model, generative model, discriminative model, holistic, patch-based, part-based, inverse kinematic.

Declaration of Originality

This thesis is submitted to the Electrical and Electronic Engineering Department of Imperial College London, in fulfilment of the requirements for the degree of Doctor of Philosophy. The contributions contained in this thesis are my own work. Main contents have been published in the following listed papers. For all these works I made the major contribution in design, implementation and experiments.

- **Danhang Tang, Tsz-Ho Yu and Tae-Kyun Kim**, "Real-time Articulated Hand Pose Estimation using Semi-supervised Transductive Regression Forests", in Proc. of International Conference on Computer Vision (ICCV), Sydney, Australia, 2013 (oral)

- **Danhang Tang, Hyung Jin Chang*, Alykhan Tejani*, Tae-Kyun Kim**, "Latent Regression Forest: Structured Estimation of 3D Hand Posture", in Proc. of Computer Vision and Pattern Recognition (CVPR), Columbus, Ohio, USA, 2014 (oral), *indicates equal contributions

- **Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, Jamie Shotton**, "Opening the Black Box: Hierarchical Sampling Optimization for Estimating Human Hand Pose", in Proc. of International Conference on Computer Vision (ICCV), Santiago, Chile, 2015 (oral)

- **Dang Tang, Hyung Jin Chang, Alykhan Tejani, Tae-Kyun Kim**, "Latent Regression Forest: Structured Estimation of 3D Hand Poses", submitted to Transactions on Pattern Analysis and Machine Intelligence (TPAMI), under review

Apart from the above, these following publications that I co-authored are out of the main scope, but also briefly mentioned in literature review and future directions.

- **Danhang Tang, Yang Liu and Tae-Kyun Kim**, "Fast Pedestrian Detection by Cascaded Random Forest with Dominant Orientation Templates", in Proc. of British Machine Vision Conference (BMVC), Surrey, UK, 2012.
- **Mohammad Rouhani, Danhang Tang, Alykhan Tejani and Tae-Kyun Kim**, "Multi-Class Object Detection Using Random Forest", Technical Report
- **Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, Tae-Kyun Kim**, "Latent-Class Hough Forests for 3D Object Detection and Pose Estimation", in Proc. of European Conference on Computer Vision (ECCV), Zurich, Switzerland, 2014
- **Mang Shao, Danhang Tang, Yang Liu, Tae-Kyun Kim**, "A comparative study of video-based object recognition from an egocentric viewpoint", in Neurocomputing, 2015
- **Chao Xiong, Xiaowei Zhao, Danhang Tang, Shuicheng Yan, Tae-Kyun Kim**, "Conditional Convolutional Neural Network for Modality-aware Face Recognition", in Proc. of International Conference on Computer Vision (ICCV), Santiago, Chile, 2015

Danhang Tang
01/09/2015

Copyright Declaration

Danhang Tang
01/09/2015

Acknowledgements

I would not have been able to finish my PhD study without the help of many people. Before commencing this thesis, I would like to express my sincere gratitude to them.

First I wish to thank my adviser, Tae-Kyun Kim, for his patience and hard work in guiding me through this journey. From research 101 to high level reasoning, what I have learned from him will immensely influence my future career.

I would also like to thank my parents for their unconditional, never-ending love and encouragement. Without their support, emotionally and financially, I would not have even begun my PhD, not to mention made it through the hard time. Thanks for always being there.

There were ups and downs over the past few years, yet I was lucky to have wonderful experience collaborating and co-authoring with a number of brilliant researchers. Youngkyoon Jang and Yang Liu, showed me the door when I was wandering outside the world of research. During our collaboration, Tsz-Ho Yu demonstrated his infectious devotion and the standard of excellent academic writing. Together with HyungJin Chang and Alykhan Tejani, we divided, conquered and had fun. I would also like to thank Mang Shao, Chao Xiong, Rigas Kouskouridas and Guillermo Garcia and everyone else in the Imperial College Vision Lab, for the working experience and their friendship. It has been a great pleasure to be part of this group. Outside the college, I was fortunate to have the chance of collaborating with researchers from Microsoft Research Cambridge, including my mentor Jamie Shotton, Pushmeet Kohli, Jonathan Taylor and Cem Keskin.

Last but not the least, I would like to thank Mike Brookes, Antonis Argyros and Ste-

fanos Zafeiriou for not only examining my thesis, but also pointing me in the direction of future work.

Danhang Tang
01/09/2015

Contents

List of Figures

List of Algorithms

List of Tables

Glossary

D maximum tree depth.

E energy function.

F a decision forest.

G latent tree model.

H Shannon entropy.

I depth image.

$L(\cdot)$ predictor function.

$Q(\cdot)$ quality function.

Δ predictor function.

Φ a set of features.

$\boldsymbol{\theta}$ joint angle label.

δ the distance function for building latent tree model.

$\mathbb{H}$ Hough space.

$\mathbf{D}$ the distance matrix for building latent tree model.

$\mathbf{j}$ joint location offset label.

$\mathbf{x}$ input vector.

$\mathbf{y}$ output vector.

$\mathcal{D}$ a training set.

$\mathcal{E}$ edges of a latent tree model.

$\mathcal{G}$ a Gaussian mixture model.

$\mathcal{J}$ a set of joints.

$\mathcal{L}$ the labeled training set.

$\mathcal{O}$ hand parts, or observable nodes of a latent tree model - these two are conceptually different but happen to be the same thing under the context of this thesis..

$\mathcal{R}$ the realistic training set.

$\mathcal{S}$ the synthetic training set.

$\mathcal{U}$ latent (unobservable) nodes of a latent tree model.

$\mathcal{Y}$ a set of pose hypotheses.

$1(\cdot)$ the indicator function.

CoM centre of mass of the foreground pointcloud.

Ψ association term.

cov covariance matrix.

tr trace of a matrix.

μ mean.

ϕ feature function, also known as weak learner model.

a viewpoint label.

h split function.

lc left child node.

n a tree node.

p hand part classification label.

rc right child node.

t a decision tree.

Acronyms

FPS frame per second.

AAM active appearance model.

ANN artificial neural network.

CLNJ Chow-Liu neighbor-joining.

CNN convolutional neural network.

DA discriminant analysis.

DF decision forest.

DIP distal Interphalangeal.

DoF degrees of freedom.

HCI human computer interaction.

HF Hough forest.

HSF hierarchical sampling forests.

HSO hierarchical sampling optimisation.

LDA linear discriminant analysis.

LRF latent regression forest.

LRT latent regression tree.

LTM latent tree model.

MCP metacarpophalangeal.

MLP multi-layer perceptron.

MSHD Microsoft Research Synthetic Hand Dataset [Sharp et al., 2015].

NYU New York University Dataset [Tompson et al., 2014].

PCA principle component analysis.

PIP proximal Interphalangeal.

PSO particle swarm optimisation.

QDA quadratic discriminant analysis.

RBF radial basis function network.

RDA regularised discriminant analysis.

SL structured-light.

STR semi-supervised transductive regression.

SVM support vector machine.

ToF time-of-flight.

CHAPTER 1

Introduction

Contents

1.1 Problem Definition

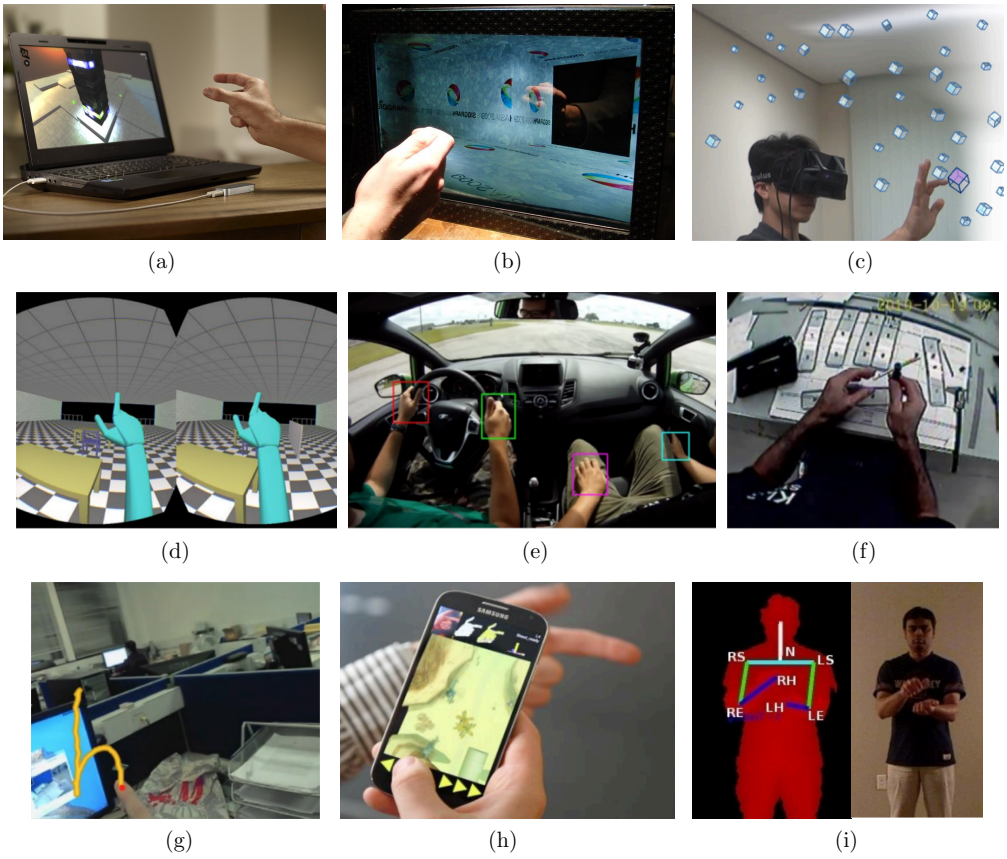
Hand pose regression is a fundamental component in many computer vision applications. Differing from gesture recognition where semantic expressions of hand poses or motions are recognized, its target is to locate the finger joints in individual images or video frames. Early works utilise monocular cameras as input devices to capture 2D informations, and then apply either discriminative [Athitsos and Sclaroff, 2003, Guan et al., 2006] or generative approaches [Chua et al., 2002, Stenger et al., 2006] to obtain

poses. Readers can refer to [Erol et al., 2007] for a thorough survey. Despite their efforts, the ambiguity of monocular 2D data render the task of full degrees of freedom (DoF) 3D hand pose regression infeasible.

Since 2011, the introduction of consumer depth sensors such as Microsoft Kinect has led to the widespread success of real-time human body pose estimation [Shotton et al., 2011]. Following that, the area of 3D hand pose estimation receives more and more attention within the computer vision community. Blooming starts when short-range depth sensors that are designed specifically for hand applications become available, including Primesense™ Carmine, Leap Motion and Intel® RealSense™, etc. In this thesis, we investigate the problem of 3D hand pose regression using discriminative approaches, in particular the decision forest. The term '3D' implies not only are the input data captured from depth data, but the outputs are also in 3D Euclidean space. Also, 'regression' indicates that instead of classifying data into discrete poses such as 'open', 'fist' and 'pointing', etc., this thesis focuses on producing continuous, full DoF poses in parameter space. And the scope is restricted to only one hand, without interaction with the other hand or objects.

Accurate and efficient 3D hand pose regression is beneficial to many human computer interaction (HCI) tasks. For instance, in the case of VR/AR, to enable interaction with virtual objects, 3D hand joint locations need to be recovered in real-time [Boussemart et al., 2004, Melax et al., 2013, Jang et al., 2015]. With the precedent of Kinect, it will come as no surprise to see video games that require fine-grained level of hand control emerging in the future. For some scenarios that are inconvenient for users to interact via other means, or hearing impaired people, gestures become a set-forward communication choice, e.g. , navigation [Hirsch et al., 2009], driver interaction [Jacob et al., 2015]. Moreover, semantic meanings built on top of gestures provide a more sophisticated communication with machines. For instance, in sign language recognition, not only 3D gestures, but the dynamic also plays a significant roles [Zafrulla et al.,

2011]. Recent works on in-air writing take a further step to free users from learning a sign language [Chang et al., 2015]. And of course the applications of controlling robot hands, especially to fulfil delicate tasks such as surgery, require high precision of 3D hand pose recovery. Examples are shown in Fig. 1.1.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 1.1: Applications of 3D hand pose regression: (a) 3D gaming with Leap Motion; (b) navigation [Hirsch et al., 2009]; (c) virtual object manipulation in AR [Jang et al., 2015] and (d) VR [Melax et al., 2013]; (e) driver vehicle interaction [Jacob et al., 2015]; (f) grasping in egocentric view [Cai et al., 2015]; (g) in-air writing [Chang et al., 2015]; (h) interaction with mobile phone [Song et al., 2015]; (i) American Sign Language recognition [Zafrulla et al., 2011].

Formally the task can be defined as a general prediction problem: given an observation point (a 3D position projected on the input image) $\mathbf{x}$[1], our algorithm produces 3D joint locations $\mathbf{y}$. Note that depending on the specific algorithm, $\mathbf{x}$ can be the centre of either a local patch or the whole hand region. $\mathbf{y}$ is restricted by the degrees of freedom of different joints. For instance, wrist and MCPs are 3 degrees of freedom (DoF). PIPs have 2 DoF whilst DIP and finger tips have only 1 DoF.[2] In total approximately 26 DoF are used thoughout this thesis[3]. To produce $\mathbf{y}$, three different types of intermediate output parameters are often used, as shown in Fig. 1.2: (a) hand part label [Shotton et al., 2011, Keskin et al., 2012, Tang et al., 2013], denoted as a scalar p, which means to classify each foreground pixel into one of the hand parts. Joint locations $\mathbf{y}$ can be estimated afterwards given these classification results. (b) 3D offsets for voting joint locations [Girshick et al., 2011, Tang et al., 2014], denoted as an offset vector $\mathbf{j}$, such that $\mathbf{y} = \mathbf{x} + \mathbf{j}$. Although all joint positions are in $\mathbb{R}^3$, they are still restricted by the aforementioned DoF. (c) Joint rotation angles [Oikonomidis et al., 2011a, Sharp et al., 2015], denoted as a vector $\boldsymbol{\theta}$. This requires a ready hand skeleton model, and $\boldsymbol{\theta}$ is represented as angles between bones. $\mathbf{y}$ can be calculated by applying forward kinematics to $\boldsymbol{\theta}$ and the hand model [McCarthy, 1991].[4] These 3 types of intermediate parameters are all applied and analysed in the main chapters of this thesis.

The following sections in this chapter are organised as this. In Section 1.2, we discuss the challenges of 3D hand pose regression, which set up the motivation of this thesis. In Section 1.3 we give an brief overview and summarise the main contributions.

[1] In fact, if we denote the input image be I, the input for our algorithms should be a tuple $(I, \mathbf{x})$. However, for notation simplicity, we will only use $\mathbf{x}$ as input throughout this thesis.

[2] MCP, PIP, and DIP stand for the metacarpophalangeal, proximal interphalangeal, and distal interphalangeal joints, respectively.

[3] In some literatures [Lee and Kunii, 1993, Lin et al., 2000] one more DoF for thumb (27 in total) is considered to represent more natural movements.

[4] Note that in this case, the 3D position of wrist is needed as global translation.

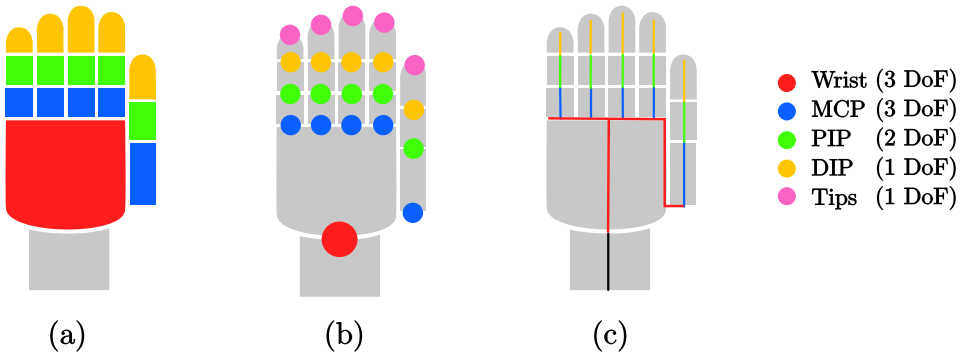


Figure 1.2: Different types of pose parameters: (a) hand part segmentation p (b) 3D offsets for joint locations $\mathbf{j}$ (c) joint angles $\boldsymbol{\theta}$.

1.2 Challenges

From a general machine vision point of view, one can treat this task as estimating the locations of a set of distinctive points from appearance data. Similar paradigm can be found in other vision problems such as face landmarking, scene registration or human body pose regression. Therefore, articulated hand pose estimation shares quite a few challenges, whilst having some unique difficulties.

High degrees of freedom

It has been shown in previous section that the human hand is an articulated object with 26 DoF. Comparing to tasks like scene registration [Guzman-Rivera et al., 2014] or face landmarking [Çeliktutan et al., 2013], one needs to consider the underlying kinematic rules that connect and restrict hand parts. Due to these constraints, natural hand movements do not span the whole 26 DoF pose space. However the parameters needed to be estimated are still large. For instance, the first 6 degrees of freedom, $\{x, y, z, pitch, yaw, roll\}$ which indicate the global location and palm orientation, can span the whole 6D parameter space, as shown in [**?**]. This makes it not only hard to optimise during testing, but also difficult to generate enough amount of representative

training data.

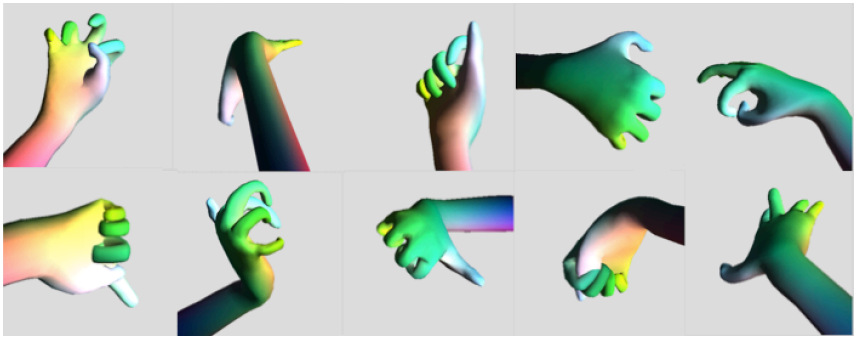


Figure 1.3: Demonstration of high degrees of freedom and self-occlusion with examples from the MSHD dataset [Sharp et al., 2015]. This dataset is synthetically generated to cover the most parameter space of all available datasets.

Self occlusion

Differing from regular occlusion, when the target is visually blocked by other objects, self-occlusion occurs when some parts of the target overlaps itself. Whilst being not so much a problem in object detection tasks, self-occlusion is a substantive threat in pose estimation problems, and especially prevalent with articulated objects like human body or hand. The complex anatomy of hand and high DoF makes it difficult to not only inference, but also labelling occluded joints. Unlike rigid objects or deformable objects with small DoF, e.g. , face, when a part is self-occluded, its location is not unique and has to be inferred via kinematic rules. In some extreme cases, e.g. , egocentric view (row 1, column 2 in Fig. 1.3), most of the fingers are occluded by the forearm. With only one monocular sensor we have no way of knowing the exact locations of those fingers. However, reasoning with kinematic rules will give us some logical guesses.

Size and shape variances

Unlike human body, hand size and shape variances are less discussed in related studies. However, from the field of medicine, [Lee et al., 2009] provides some statistics showing that the variances are too significant to be ignored. The study involves 46 male subjects between 20 and 39 years of age, which apparently only covers a limited demographic. Despite that, hand length spans from 173.5 mm to 208 mm, and hand width spans from 72.1 mm to 97.3 mm. This amount of variation inevitably leads to differences in appearance space, which may cause failure especially with model-fitting based methods.

Noisy hand pose data.

Body poses usually occupy larger and relatively static regions in the depth images. Hands, however, are often captured in a lower resolution. As a result, sensor noise level in hand pose regression tasks is relatively higher. In the case of structured-light sensors, this leads to artefacts that cannot be repaired or smoothed easily. Consequently, a large discrepancy is observed between synthetic (noise-free) and realistic data.

Moreover, manually labelled realistic data are extremely costly to obtain. Existing state-of-the-arts resort to synthetic data [Keskin et al., 2012], or model-based optimisation [de La Gorce et al., 2011, Oikonomidis et al., 2011a]. Nonetheless, such solutions do not consider the realistic-synthetic discrepancies, their performances are hence affected. Besides, the noisy realistic data make joint detection difficult, whereas in synthetic data joint boundaries are always clean and accurate.

Rapid motion

Natural hand movements can be rather fast. For instance, wrist movements can be up to 5 m/s in translation and 300°/s in rotation [Erol et al., 2007]. Modern consumer

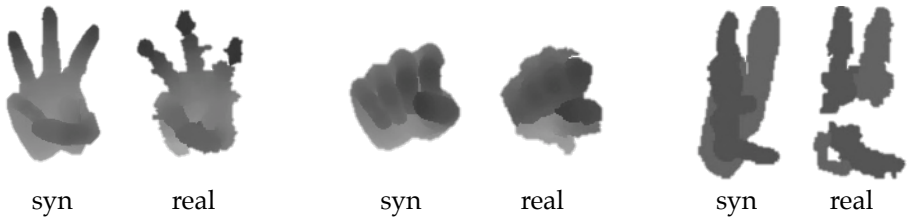


Figure 1.4: Three pairs of synthetic-real data from the ICVL dataset [Tang et al., 2013] showing the discrepancy between them.

depth sensors only run at a frame-rate up to 30 Hz. This inevitably causes motion blur within frames and reduces smoothness between frames, which often leads to failure in tracking-based methods. Setting aside the sensor frame-rate, rapid motions also give rise to the efficiency requirement for both discriminative and generative types of methods. Furthermore, similar to human body, it has been discussed in [Forsyth et al., 2005] that smaller parts tend to move faster, which make them harder to be located.

Segmentation

3D body pose estimation methods often consider a typical gaming scenario, where the targets are detached from background, e.g. , walls, furnitures. In this case simply setting a threshold on depth values can segment the targets. Related literatures thus often treat segmentation as provided. However, in the case of human hands, it is very difficult to obtain a clean segmentation. For one thing, hands and arms are connected and share the same skin colour. For another, the relative distance between hands and body can vary significantly and rapidly. Therefore simply applying traditional image processing algorithms like region growing with depth and/or colour cues would not work well. Despite that this thesis does not have contribution regarding segmentation, how we use off-the-shelf algorithms to solve this non-trivial task will be discussed in main chapters.

1.3 Thesis Overview and Contributions

This thesis covers a line of work that starts by migrating the successful decision forest (DF)-based algorithms on 3D human pose estimation [Shotton et al., 2011, Girshick et al., 2011]. As the aforementioned hand-specific challenges discovered, we address them by leveraging cheap unlabeled data, exploiting the inherent hierarchical structure of DF, and enforcing kinematic constraints, etc. Chapter 2 covers a general literature review in the field. Chapter 3 provides basic concepts and notations of DF and its variants. Chapter 4 introduces our own datasets, labelling techniques, as well as several publicly available datasets. Evaluation criteria are also discussed. Main contributions are covered in Chapter 5, Chapter 6, and Chapter 7. Apart from the reviews in Chapter 2, each of these chapters also explores some works that are closely related to the motivation and contributions of that chapter. A discussion of conclusion and future works is in Chapter 8.

Highlights of main chapters are listed as below:

Chapter 4. Datasets and Evaluation Protocols

This chapter proposes two datasets, as well as capturing/generation methods and how to annotate them. The first dataset (ICVL v1) targets specifically the discrepancy between realistic and synthetic training data. Real data are captured with a structured-light sensor (Primesense) and thus rather noisy, whilst noise-free synthetic data are rendered from a mesh model. To accommodate the contributions in Chapter 5, a linking method between synthetic and real data is introduced. The second dataset (ICVL v2) uses a time-of-flight (ToF) sensor which has much lower noise comparing to a structured-light one. Also an automatic annotation method is adopted to make it possible to label large amount of real data.

Apart from our own datasets, 2 publicly available datasets (NYU and MSHD) used in our experiments are also introduced. Pros and cons of data and annotations are discussed.

Furthermore, experiment protocols and evaluation criteria are also discussed in this chapter.

Chapter 5. Semi-supervised Transductive Regression Forest

This chapter addresses the problem of lacking labelled real training data. Unlike human body pose estimation, due to the small size of human hands and sensor noises, there is usually a discrepancy between synthetic and realistic data, especially with the structured-light based sensors like Primesense™ . Hence training with only synthetic data does not yield a satisfying result. On the other hand, realistic training data are costly via manual labelling, because of the complex hand articulation.

In this work, the semi-supervised transductive regression forest (STR forest) is proposed. STR forest performs various recognition tasks, including viewpoint classification, joint classification and joint regression. It utilises both synthetic and real training data via transductive learning, which creates associations between the two training datasets. In addition, semi-supervised learning is employed in the STR forest training algorithm, in order to utilise the unlabelled real data. Furthermore, a data-driven inverse kinematic technique is also presented to recover the occluded joints from the STR forest.

This work has been published in the IEEE International Conference of Computer Vision, 2013 [Tang et al., 2013].

Chapter 6. Latent Regression Forests: a coarse-to-fine search for poses

This chapter aims to reduce the complexity of forest-based methods. Categorized by the type of input data, there are two types of discriminative methods: holistic and patch-based. Holistic methods are efficient but less generalizable, whilst patch-based methods have strong generalization power, but the complexity depends on the amount of input patches (usually thousands).

In order to combine the advantages of both types of methods, we propose latent regression forest (LRF), which can be viewed as a coarse-to-fine search. LRF is structured by a latent tree model (LTM), which decomposes the high-dimensional pose estimation problem into different layers of low-dimensional targets. Starting from the centre of segmented foreground, LRF gradually approach the locations of each joint layer by layer. Furthermore, we propose to use Chow-Liu neighbor-joining (CLNJ) [Choi et al., 2011] to automatically learn the desired LTM in an unsupervised manner, i.e. , without knowing the kinematic structure of human hand. Which means this method can be easily applied to other articulated objects.

This work has been published in the IEEE Conference on Computer Vision and Pattern Recognition, 2014 [Tang et al., 2014].

Chapter 7. Hierarchical Sampling Forests: a progressive search for poses

A typical method that combines discriminative and generative methods, such as [Sharp et al., 2015] will regress candidate full poses and then iteratively update them based on the evaluation of the energy function. The output full poses of the discriminative part are inaccurate and therefore making the generative part difficult to converge. Since the discriminative regression part knows little about either the relationships between the parameters or the form of the energy function, we call this black box optimization.

One set-forward idea is to replace the discriminative part with LRF. However, there are a few weak points of LRF. (1) Since results of latent nodes are unobservable, there is no way to verify them. (2) The outputs of LRF are 3D locations, whilst joint angles are more desirable in many applications. (3) Though implicitly encoded with kinematic information, results are not bounded by kinematic constraints. (4) It is not straightforward to incorporate tracking.

Targeting these, we propose a novel method that improves upon black box optimization by exploiting high-level knowledge of the structure of the parameters. In particular, our method regresses partial poses in a layered progressive manner, guided by kinematic structure. A surrogate energy function is proposed to keep only the best partial poses. Our approach can achieve higher accuracy than black box optimization, even without iteration. Due to the hierarchical structure, our method is called hierarchical sampling optimization.

This work has been published in the IEEE International Conference of Computer Vision, 2015 [Tang et al., 2015].

CHAPTER 2

Literature Review

Contents

2.1 Overview

As mentioned in previous chapter, from a broader perspective, regressing 3D hand pose can be considered as a general process of mapping from appearance space to parameter space (Euclidean, angle, shape, etc.). This has been studied in other computer

vision areas such as facial landmark or human pose regression. Therefore many algorithms or techniques we use in 3D hand pose regression are rooted in earlier algorithms in those areas. In this chapter, in addition to recent literature of 3D hand pose, we will also review some related work in facial landmarking and human pose regression. For convenience of readers, we will focus on literature in a general way, i.e. , the work that related to all the main chapters in this thesis; in addition to that, each main chapter also has its own related work section, where the works that only related to that chapter are discussed. The following sections are organised as this. In Section 2.2 and Section 2.3 we go over facial landmark detection and human body pose estimation respectively. And then in Section 2.4, related works of hand pose estimation are classified and discussed. Due to the importance and universality to all articulated objects, kinematics and tracking are discussed in Section 2.5 and Section 2.6, respectively.

2.2 Facial Landmarking

Facial landmarks are prominent features on face such as eye corners, ear lopes and nose tip. Detecting them is a fundamental process for many higher level face applications like expression understanding, face recognition and face registration. From the examples shown in Fig. 2.1, one can see that the landmark points are globally structured and locally deformable, which is similar to our hand pose regression problem. The difference lies in the level of deformation, unlike hand, face has less degrees of freedom (DoF), hence less self-occlusion. Moreover, the deformation in hand is articulated, i.e. , a set of rigid parts grouped together by some kinematic rules. Whereas in face, the deformation is usually done via local warping of templates, without using any kinematic rule. In this section, we choose to review a few papers that are closely related to our problem, so that readers can quickly see the connection. For earlier face landmarking papers, readers can refer to [Çeliktutan et al., 2013] for a review; for more

recent work, [Yang et al., 2015] provide an empirical study.

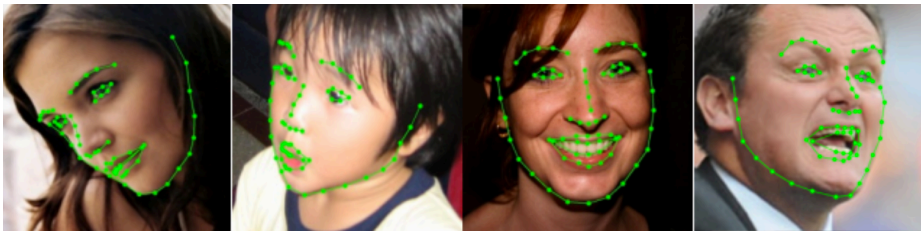


Figure 2.1: Facial landmark detection examples adapted from [Zhao et al., 2014]. Similar to 3D hand pose, these landmark points are globally structured and locally deformable.

In the field of facial landmarking, the most popular work is probably the active appearance model (AAM) and its variants. Since its beginning, the fitting of AAM has developed a generative [Cootes et al., 1998] and a discriminative [Edwards et al., 1998] lines of work. [Cootes et al., 1998] is a generative method which iteratively tuning the model parameters while minimising the $l2$-difference between the observation (shape and texture) and synthesised data generated with current parameters (landmarks). In contrast, [Edwards et al., 1998] turn AAM fitting into a discriminative linear regression of the relationship between observation and parameter updates. Ever since then, numerous contributions has been made from these two perspectives. One drawback of the original AAM, generative or discriminative, is the assumption that the shape and texture are linearly separable with mean and variance-in-subspace learned from a training set via principle component analysis (PCA). Such a model does not generalise well to unseen samples, due to the limited representational power of the eigenspace [Gross et al., 2004, Liu, 2009]. Another weakness is that the discriminative AAM [Edwards et al., 1998] only utilise one step of regression, with smaller weight of the update on each iteration.

To address these, [Saragih and Göcke, 2007] propose to discriminatively train a boosting classifier with non-linear weak-learners for each iteration. Similarly,

[Liu, 2009] turns the non-linear regression problem into a binary classification of appearance-parameter pairs (ground-truth pairs are labelled as positive whilst other pairs are negative). On top of that, [Tresadern et al., 2010] introduce a two-stage method which use non-linear models at first, and linear models when getting close to ideal solution to improve efficiency without sacrificing accuracy. All these work develop some tricks in how to perform an iterative non-linear regression with boosted classifiers, which was originally designed for classification tasks. To avoid the necessity, [Dantone et al., 2012] adopt the regression forest, but going back to the one-step regression scheme makes it rather greedy.

State-of-the-art landmarking algorithms follow a 'cascaded regression' paradigm. The seminal work of supervised descent method (SDM) [Xiong and De la Torre, 2013] formulates facial landmarking as a general non-linear least square (NLS) problem which can be solved with optimisation techniques. A cascaded linear regressor is then constructed to approximate the steps of Newton's method, avoiding the differentiation of SIFT features. In parallel, [Cao et al., 2012] also propose a boosted cascaded regression method. However, unlike previous boosting-based methods [Saragih and Göcke, 2007, Liu, 2009, Tresadern et al., 2010], they treat all landmarks as a holistic template, which enforces the shape constraints. Their following up work [Ren et al., 2014] jointly learn the regressor and a fairly efficient form of binary feature representation, which results in the speed of 3000 FPS.

2.3 3D Body Pose Regression

Since human body and hand are both articulated objects, studies of them share a lot of similar ideas, methodologies and research directions. Five years back, when researchers had achieved commercial success in 3D human pose estimation, 3D hand pose was still under-investigated. Ever since many hand pose algorithms have been

inspired by the the body pose community. Hence it is sensible and necessary to explore body pose literatures before tackling hand pose problems. In this section, we have limited the contents to studies using depth sensors, i.e. , 2.5D data to address 3D human body pose estimation.

2.3.1 Discriminative, Generative and Hybrid

From a machine learning perspective, existing methods can be also be classified into discriminative, generative and hybrid methods. Dating back to 2011, the seminal work of Kinect [Shotton et al., 2011] treat 3D body pose estimation as a classification problem by classifying each foreground pixels into one of the body parts, followed by a mean-seeking method to locate body part centres and hence skeletons. On top of that, [Girshick et al., 2011] propose to use regression forest to vote for self-occluded joints. To decouple torso orientation and body pose, [Sun et al., 2012] introduce a 2-layer structure called Conditional Regression Forest. Later on, [Kontschieder et al., 2013] combines regression forest with Auto-context to improve the pixel classification accuracy.

Generative methods use optimization techniques to fit a 3D mesh model into observed data. [Zhu et al., 2008b] apply Cartesian tracking control to minimize the discrepancy between current configuration and observed features. [Deutscher and Reid, 2005], on the other hand, propose a particle-based stochastic search algorithm to track the poses. To improve accuracy, [Gall et al., 2010] design a 2-layer optimization, where the first layer performs global optimization in a stochastic manner, and the second layer refines the results locally.

Whilst generative methods can provide more refined results, they require an articulated mesh model, and rely on tracking for a good initialization to avoid local minima. Discriminative methods provide a direct mapping from observation to a configuration.

Thus are fast and tracking-free. It is natural to combine them and achieve the advantages of both. [Salzmann and Urtasun, 2010] use constrained discriminative regression as initialization, followed by a generative refinement stage. These two stages are considered as forward/inverse kinematic mappings and thus can be formulated into the same framework. To reduce tracking error, [Baak et al., 2013] use a voting scheme to combine final pose from the previous frame and the discriminative result from current frame and then locally fit a skinned model to depth data. [Ye et al., 2011] take one step further by not using tracking at all, i.e. , one-shot optimization. They first perform a nearest neighbour search in a large database to find the closest pose that matches current data in a low-dimensional space, and then refining the pose by a mesh warping optimization. Similarly, [Taylor et al., 2012] also perform one-shot optimization, by first discriminatively inferring dense pixel-to-model correspondences.

2.3.2 Holistic and Patch-based

Judging by the type of input data, one can also categorize prior arts into holistic or patch-based. Assuming the target body region is segmented, holistic predictors take the whole region as input and give one full pose result. Since there is only one input sample, it is usually quite efficient. However, holistic methods are usually implemented as nearest neighbour search, which heavily relies on the parameter coverage of training samples. In other words, they have less generalization power and thus are less accurate. [Ye et al., 2011] use PCA to reduce the dimensions of holistic samples, whilst [Baak et al., 2013] use a sparse feature representation for holistic matching. In both cases, although efficiency of discriminative part is further improved, to have satisfying accuracy, a local refinement step is applied afterwards, which becomes the runtime bottleneck.

State-of-the-art methods often fall into the patch-based category. In this case, each

foreground pixel is considered as the centre of a local patch. Predictors make decision for each pixel based on its surrounding local appearance. This allows combination between training data and thus can generalize to unseen samples to some degree. The Poselets [Bourdev and Malik, 2009] define multiple discriminative parts that contain a subset of body joints in order to predict 3D body poses. Examples using depth sensors include [Shotton et al., 2011, Girshick et al., 2011, Sun et al., 2012, Kontschieder et al., 2013]. A downside of these methods is the amount of input samples, depending on the number of foreground pixels, is usually hundreds of thousands. Since both training and testing time are proportional to the amount of input samples, patch-based methods apparently have higher complexity than holistic ones.

2.4 3D Hand Pose Regression

Earlier approaches for articulated hand pose estimation are diversified. For instance, [Chua et al., 2002] recognises hand poses from colored markers on hand to analyse articulations. [Athitsos and Sclaroff, 2003] estimates articulated hand and viewpoint from a database using probabilistic line matching. Pose ambiguities of hand poses are resolved using multiple cameras in [Guan et al., 2006]. [Stenger et al., 2006] infer hand poses using a Bayesian filter based on Chamfer matching. We refer the readers to [Erol et al., 2007] for a detailed survey of early hand pose estimation algorithms.

Similar to the body pose problem, modern hand pose algorithms can also be classified into these 3 categories: discriminative, generative and hybrid. Discriminative and generative approaches have their pros and cons, whilst a hybrid solution will try to combine them to achieve the benefits of both. For a detailed survey, recently [Riegler et al., 2015] propose a framework and a dataset for evaluating 3D hand pose methods.

2.4.1 Discriminative Approaches

Discriminative approaches directly learn the posterior $p(\mathbf{y}|\mathbf{x})$, i.e. , a mapping from visual features $\mathbf{x}$ to a value $\mathbf{y}$ in the target parameter space.

Instead of using a predefined visual model, discriminative methods learn a pose estimator from a labelled training dataset. [Romero et al., 2009, Wang and Popović, 2009] both apply approximate nearest neighbour search for matching database samples in a holistic manner. [Keskin et al., 2012] propose a solution to the data-explosion problem, by first clustering the training data followed by training multiple experts on each cluster using the method of [Shotton et al., 2011]. Furthermore, due to the increased variation in the hand, capturing ground-truth annotated real data is a problem in its own right. Recently, inspired by the aforementioned cascaded regression in facial landmarking [Xiong and De la Torre, 2013, Cao et al., 2012], [Sun et al., 2015] successfully combine cascaded error regression with a multiple-layered hierarchical regression forest, achieving good accuracy and real-time speed in 300 FPS.

Discriminative methods rely heavily on the quality of training data. A large labelled dataset is necessary to model a wide range of poses. The performance is therefore highly correlated to the size and diversity training dataset. Even if taking global translation out of the equation, a 3D hand pose still has 23 DoF, which is already impossible to exhaustively generate all training samples, not to mention the shape and size variance. Therefore a generative refinement step is usually needed to have satisfying accuracy.

2.4.2 Generative Approaches

Also known as 'analysis by synthesis' , generative methods are popular among recent state-of-the-arts. Hypotheses are generated from a visual model, e.g. a 3D hand

mesh. Hand poses are tracked by fitting the hypotheses to the test data. For example, [de La Gorce et al., 2011] use a hand mesh with detailed simulated texture and lighting. [Hamer et al., 2009] address strong occlusions using local trackers at separate hand segments. [Ballan et al., 2012] infer finger articulations by detecting salient points. [Oikonomidis et al., 2011b] estimate hand poses from RGB-D images using particle swarm optimisation (PSO). Model-based approaches inherently handle joint articulations and viewpoint changes. However, their performances depend on the previous pose estimations, output poses may drift away from groundtruth when error accumulates over time. Moreover, the dense error function they use requires rendering and thus becomes speed bottleneck. Recently, [Melax et al., 2013] proposed a tracker based on physical simulation which achieves 60 FPS.

It is worth mentioning that most generative methods do not consider shape variations of hands due to the additional complexity involved [Erol et al., 2007], which jeopardises their accuracies in realistic applications. [Melax et al., 2013] require manual adjustment on some hand shape parameters (finger length, palm width, etc..). [Taylor et al., 2014] propose a solution to personalize a 3D hand mesh model to user hand shapes. Their follow-up work [Khamis et al., 2015] reduces the size and shape parameters by decomposing the 3D hand mesh model into a few basis. Hence the shape of hand can be adjusted easily. [Qian et al., 2014], on the other hand, discard the 3D mesh model and adopt an approximate ball model, which not only make it less sensitive to size and shape variance, but also improve the run-time speed.

2.4.3 Hybrid Approaches

When tracking is eventually lost, the incorrect poses cannot be easily recovered without re-initialisation. Modern pose estimation algorithms attack the problem with a 'hybrid' discriminative/generative approach. [Qian et al., 2014] use fingertip detection

results as initialization for PSO. Also they replace the dense error function in [Oikonomidis et al., 2011b] with an approximate but more efficient version. [Tompson et al., 2014] apply convolutional neural network (CNN) to location 14 joint positions as initialization for PSO. [Sridhar et al., 2015] use decision forest to classify pixel into parts, in order to provide an energy function for a later optimization step. One difficulty in such a hybrid solution is the *interface* between the discriminative and generative parts, since the discriminative part usually produces either part classification p or joint location $\mathbf{j}$ results, whereas the generative part requires joint angle $\boldsymbol{\theta}$ for rendering pose hypotheses (previously defined in Section 1.1). Hence the above methods all have an extra step converting between these two iteratively. [Sharp et al., 2015], on the other hand, design a discriminative part that can produce joint angle $\boldsymbol{\theta}$ directly. However, unlike joint locations $\mathbf{y}$, joint angles $\boldsymbol{\theta}$ can not be divided into independent local regression problems, because $\boldsymbol{\theta}$ is a kinematic chain. Also regressing on $\boldsymbol{\theta}$ as a whole is difficult due to the high DoF. Hence [Sharp et al., 2015] take a different path by classifying the input into one of 7 discrete poses (open, fist, pointing, etc..), and randomise the angles according to the kinematic constraints in order to generate a set of hypotheses for the generative part (PSO). This unfortunately sacrifices the accuracy of the discriminative results and thus increases the burden for PSO.

2.5 Kinematics

Kinematics is a set of mechanic rules which describes the anatomical connectivity, degrees of freedom and motion range of each part of an entity [McCarthy, 1991], usually in the form of joint angle ranges. It is the defining feature of articulated objects (human body, hand, robot arms, etc.), comparing to rigid objects or deformable objects (face, cloth, etc.). In the field of computer graphics, kinematic rules are used for rendering articulated objects, which is done by applying a sequence of rigid transformation from

the base part (e.g. , wrist or forearm) to end parts (e.g. , finger tips). This process is termed *forward kinematic*. Conversely, if one tries to recover joint angles $\boldsymbol{\theta}$ given all or some of the joint locations $\mathbf{y}$ (previously defined in Section 1.1), the process is called *inverse kinematic*.

The generative methods for 3D hand pose regression usually require rendering a set of hypotheses via off-the-shelf forward kinematics, hence the final outputs are always anatomical correct. Examples can be found in human body pose regression [Yao et al., 2012, Pons-Moll et al., 2011], as well as hand pose regression [de La Gorce et al., 2011, Oikonomidis et al., 2012, Stenger et al., 2006]. The discriminative methods, on the other hand, need to be considered in three-fold: 1) If the method classifies pixels into hand parts, p, we can only infer visible part/joint locations. In this case we must use inverse kinematics to recover all joint locations $\mathbf{y}$. 2) If one choose to regress joint angels $\boldsymbol{\theta}$ directly, and then calculate $\mathbf{y}$, the angle constraints can be applied to $\boldsymbol{\theta}$ and hence $\mathbf{y}$ are kinematically correct. [Sharp et al., 2015] follow this scheme by constructing a set of discriminatively trained decision forest, which directly regresses full poses in the format of joint angle $\boldsymbol{\theta}$. However, as mentioned in the previous section, this is less accurate. 3) If we directly regress the offset votes $\mathbf{j}$ and then obtain absolute locations $\mathbf{y}$, there is no guarantee that the results are anatomically correct. At this point an inverse kinematic process is also needed as a global optimisation to correct the mistakes.

Inverse kinematic has been extensively studied in the robotic field [DSouza et al., 2001, Oyama et al., 2001, de Angulo and Torras, 2008, Neumann et al., 2010]. In general there are two different types: learning-based and model-based. The learning based methods generate a training set $\{(\mathbf{y}, \boldsymbol{\theta})\}$ with a skeleton model. And then train a discriminative method to predict $\boldsymbol{\theta}$ from $\mathbf{y}$. Note that unlike the aforementioned discriminative methods, the input here are not the appearance data $\mathbf{x}$ but predicted joint locations $\mathbf{y}$. If $\mathbf{y}$ is not complete, e.g. , with only visible joint locations, the inverse might be ambiguous. Even in the case of complete $\mathbf{y}$, considering not all joint locations are kine-

matically correct, this process must be robust to prediction errors. To disambiguate the inverse, [DSouza et al., 2001] train a regressor to jointly consider the appearance data $\mathbf{x}$ and $\mathbf{y}$. [Oyama et al., 2001] find it hard to train a single artificial neural network (ANN) to invert kinematics, instead they divide the parameter space into subspace and train an ANN each. In the field of hand pose, [Tang et al., 2013] generate a kinematic dataset and use a simple nearest neighbour search to find the closest match (see Chapter 5 for more details).

Model-based approaches are basically adapted from aforementioned generative methods for pose regression. The only modification is to replace the energy function that measures appearance difference, with a new energy function that measures the $l2$ difference between $\mathbf{y}$ and the locations calculated from $\boldsymbol{\theta}$. One example is the inverse kinematic step in [Tompson et al., 2014], which is essentially an adaption of [Oikonomidis et al., 2011a] that based on PSO.

2.6 Articulated Pose Tracking

Although obtaining good accuracy with individual frames are important, tracking can significantly improve the performance as well. Regular tracking collapses the target objects into bounding boxes or even points, and then associates them with trajectories across video frames. Pose tracking takes a further step by also keeping track of the object poses, for instance, the global rotation (pitch, roll and yaw) for a rigid object. In the case of articulated pose tracking, also termed kinematic tracking, locally each part moves along a trajectory, while being grouped by the global configuration which follows certain kinematic rules. Due to the high degrees of freedom with articulated objects, the amount of possible states are exponentially higher.

From a tracking perspective, hand is generally considered a more difficult target

than body. First of all, hands generally appear smaller in the field of view. Secondly, motions of hand are faster. It is quite often that the speed of finger tips exceeds the sensor frame-rate, which makes them difficult to track in Euclidean space. Moreover, although clothing makes it more challenging for body segmentation, it is actually in favour of body pose tracking due to the distinctive features. Unfortunately hand pose tracking cannot benefit from that.

Adapted from a body pose survey [Forsyth et al., 2005], we classify articulated pose tracking methods into *tracking-using-flow*, *tracking-by-update* and *tracking-by-detection*. Tracking-using-flow utilises the pixel-wise correspondence between frames. It is of great use to face or body pose tracking. However, due to the low efficiency, less robust to rapid motions, and dependency on distinctive features, it is seldom used in hand pose tracking. Tracking-by-update treats the pose tracking problem as an iteratively updating process, either probabilistic or non-probabilistic. In the case of probabilistic update, the tracked poses are considered as *hidden states*, whose dependency are usually assumed Markov. Examples include hidden Markov model [Chen et al., 2003], Kalman filter [Stenger et al., 2001] and particle filter [Shan et al., 2004]. Among them particle filter is rather popular owing to no assumption for distribution. However, the efficiency quickly drops in high dimensional space for its sampling nature. An evolutionary-based update method, particle swarm optimisation (PSO), is then applied to body pose [Ivekovič et al., 2008] and later hand pose problem [Oikonomidis et al., 2011a]. Instead of sampling, particles in PSO efficiently move around the parameter space, with the guidance of an energy function that is not necessarily differentiable. Hence it quickly dominates the pose tracking field. Despite its success, PSO is known for being easily trapped in local minima. Using a discriminative detector to 'reinitialise' the tracking every now and then is therefore a good idea [Sharp et al., 2015]. This scheme falls into the category of tracking-by-detection. Using PSO, one does not need to decide when to track or detect, but simply mix the poses from

previous frame(s) and the detector into the particles[1]. The heuristic part is the ratio between these two, which is usually decided empirically.

[1]Each particle is a full pose hypothesis.

CHAPTER 3

Decision Forest

Contents

3.1 Overview

Since the main contributions of this thesis are based on decision forest (DF), for self-containedness, the background knowledges of DF are introduced in this chapter. DF, also known as random forest or randomised trees, was originally proposed

by [Breiman, 2001] and extensively studied in [Criminisi and Shotton, 2013]. Conceptually, a DF is based on the concept of decision tree [Quinlan, 1986], whereby at test time each internal tree node routes data to its left or right child-node by applying a threshold to a projection of the input features. Each input ultimately ends up at a *leaf node* , where a prediction function is stored during training and applied during testing. On top of that, [Breiman, 2001] combined the idea of bagging [Breiman, 1996] with an ensemble of decision trees and termed it as random forest (decision forest).

DF for classification and regression were first introduced in [Breiman, 2001]. Over the years, many variants have been proposed, including Hough forest [Gall and Lempitsky, 2009], clustering forest [Schölkopf et al., 2006], manifold forest [Bonde et al., 2010], semi-supervised forest [Leistner et al., 2009], conditional forest [Sun et al., 2012], and so forth. These variants usually modify one or several of the following components of DF: (1) Quality function, a criterion for selecting the best feature. (2) Predictor, a function that is used for predicting output $\mathbf{y}$. (3) Feature, the function for splitting data $\mathbf{x}$. (4) Tree structure, which inherently has a hierarchical architecture.

The following sections are organised as these. Section 3.2 briefly gives a lateral review of modern classifiers and Section 3.3 lists the reasons why we choose DF for this problem. The quality functions and predictors for classification and regression tasks are discussed in Section 3.4. A few choices of feature functions are introduced in Section 3.5. Finally in Section 3.6, the generic training and testing procedure are provided as algorithm templates for following chapters.

3.2 Related Work

Numerous classifiers have been proposed to fulfil machine learning tasks. In a rather recent and comprehensive survey [Delgado et al., 2014], 179 modern machine learning

classifiers are grouped into 17 families by the fields they originated from. This seems a bit overly categorised-for instance, boosting, decision trees, bagging and random forest belong to 4 different families-but is also useful when performing fine-grained comparison. For a conceptual discussion, we select the families with top-tier result from [Delgado et al., 2014] and simplify the categorisation by merging some of them. Eventually we end up with four families: discriminant analysis, support vector machines, neural networks and ensemble classifiers.

Discriminant analysis

Discriminant analysis (DA) based methods have been used for feature extraction and classification purposes. They project input data to a lower dimensional space, attempting to maximise the between-class difference whilst minimising the intra-class difference. Among them the linear discriminant analysis (LDA), which uses a linear projection, has caught much attention since the successful application Fisherface [**?**]. Since this method performs the principle component analysis (PCA) to reduce the dimension before apply LDA, which may have discarded some discriminant information, quite a few direct LDA (D-LDA) methods without PCA are proposed, such as [Yu and Yang, 2001]. On the other hand, when the input data has more degrees of freedom, for instance, introducing variations like lighting, viewpoint, age, gender. into the face recognition problem, will significantly lower the performance of LDA, since the input data are much more difficult to be separated linearly in the appearance space. Hence the quadratic discriminant analysis (QDA) is proposed to allow non-linear decision boundaries. However, QDA suffers from the 'small sample size' problem [Wahl and Kronmal, 1977]. Targeting that, [Friedman, 1989] proposed the regularised discriminant analysis (RDA) to allow QDA training even when the sample size is smaller than feature dimension.

Support vector machine

The standard support vector machine (SVM) [Vapnik, 1995] is a non-probabilistic classifier, which takes a set of input data and classify them into two classes by finding the decision boundary for which the margin is maximized. The original version was a linear, binary classifier. [Boser et al., 1992] combined it with the kernel trick and proposed a non-linear version. For multi-class problem, [Vapnik, 1998] proposed an intuitive *one-versus-the-rest* scheme, which composes of K SVMs trained using data from K classes respectively. Another approach called *one-versus-one* is to train $K(K-1)/2$ different 2-class SVMs on all possible pairs of classes. Although these methods are not difficult to implement, due to the limitation of speed and size in training, the optimal design for multi-class SVM classifiers is yet a further area for research. Note that it has been proved that SVM with standard kernels can approximate any continuous function up to any desired accuracy, if the parameters are chosen appropriately [Hammer and Gersmann, 2003].

Artificial neural network

Artificial neural network (ANN) is among the earliest machine learning techniques and can be traced back to the 40s. Being overtaken by later methods such as SVM for a few decades, ANN has regained much interest due to the recent outburst of deep learning. Since a deep structure of ANN is more for feature representation learning, as in [Delgado et al., 2014], we focus on the ANN as a classifier in this chapter. As to the comparison between decision forest and deep learning (in particular convolutional neural network), we refer the readers to Chapter 7 for an empirical result.

Based on the idea of perceptron, which is essentially a linear model, the multi-layer perceptron (MLP) consists of several fully-connected layers of neurons. According to the universal approximation theorem, MLP, even with one hidden layer and a finite number of neurons, is a universal function approximator. In 1988, [Broomhead and

Lowe, 1988] proposed radial basis function network (RBF), a type of ANN that uses radial basis functions as activation functions. Unlike the back-propagation in MLP, each layer of RBF can be trained separately while also being a universal function approximator [Park and Sandberg, 1991].

Ensemble learning

An ensemble model, also known as an additive model, is achieved by grouping a set of weak learners to have a stronger classifier. In [Dietterich, 2000], three reasons for using an ensemble method are listed: *statistical*-better parameter coverage, *computational*-less likely to be trapped in local minima, and *representational*-stronger function approximation power than a single predictor. The ensemble classifier family includes decision forest (DF), boosting, bagging and ferns, etc. Apparently there are usually two different ways of constructing an ensemble: by boosting or by bagging. A boosted classifier [Viola and Jones, 2004] makes a fast decision by aggregating simple weak-learners, whose computations are accelerated on an integral image. To achieve real-time performance, a cascade of boosted classifiers, which can be seen as a degenerated tree, has been widely exploited. A tree-structured system consisting of multiple boosting classifiers [Torralba et al., 2007, Wu and Nevatia, 2007, Grossmann, 2004] has been studied for accelerating the classification time for multi-view or multi-category object detection. The tree hierarchy speeds up the decision-making by filtering out those easy examples at the tree roots. In [Sochman and Matas, 2005, Zhou, 2005], it takes an early exit when the boosting sum reaches a certain value whose sign cannot be altered by the remaining weak-learners. In [Kim et al., 2011], the state-of-the-art object detector was significantly speeded up by converting a boosting classifier into a decision tree by Boolean optimisation. Despite their success in real-time performance, they often resort to a large amount of training time and effort for high detection accuracy. A solution more conveniently scalable-up to a large train data set is highly required for better

recall rates and a wider scope of applications.

Owing to its scalability and real-time performance, DF has made an enormous success in human pose estimation [Shotton et al., 2011] for console game user interfaces. Numerous variants has been proposed and proved very successful in semantic image segmentation, key-point tracking, and object categorisation problems as a fast discriminative codebook. Among them HF is a successful case [Gall and Lempitsky, 2009, Girshick et al., 2011]. The key difference from DF is that HF incorporates a new split function to measure the offset uncertainty between current patches and the object centre, which is inspired by the classification and regression tree (CART) [Breiman et al., 1984]. Working along with the appearance term, it clusters patches close to each other both in appearance and spatial space. At a testing stage, a Hough voting scheme is performed with these offset vectors to predict object centres.

Similar to SVM and ANN, the function approximation capability of decision tree has been proved, but mostly under the context of reinforcement learning [Pyeatt et al., 2001], where the input space is divided into different subspace, and local functions are evaluated. As a boosted ensemble of decision trees, the gradient boosting trees [Friedman, 2001] was designed as a greedy function approximator, under the formulation of steepest gradient descent. As a bagging ensemble, the DF also inherits the function approximation capability. It has been discussed in [Dietterich, 2000] that theoretically, ANN or a single decision tree can explore the whole function space, if given enough training data. However in practice, with a finite training set, one should try to avoid finding a function that only fits the training data. Hence the ensemble and randomness of DF can significantly reduce over-fitting.

3.3 Why Decision Forest?

According to previous studies [Breiman, 2001, Criminisi and Shotton, 2013], DF has quite a few desired properties as a classifier. In this section, we discuss the motivation of choosing DF over other classifiers, with respect to the 3D hand pose regression problem.

Among modern classifiers, it has been empirically proved that DF has the top-tier performance. For instance, in the comparison study conducted by [Delgado et al., 2014], ensemble classifiers out-perform all other families, in both 2-class and multi-class cases. Also a DF-based method achieves the best accuracy among all 179 classifiers. Furthermore, in the field of 3D human pose estimation, DF has been the most popular classifier since the seminal work of [Shotton et al., 2011]. Given the similarity, it is nature to expect its good performance in our problem.

With the tree structure, DF inherently supports multi-class/multi-variate problems. And similar to SVM, the decision boundary it generates has the maximum-margin property, even in the case of multi-class [Criminisi and Shotton, 2013]. Compared to binary classifiers like SVM or adaboost, DF doe not require a one-vs.-one or one-vs.-rest scheme for multi-class problems, hence more efficient. Since the 3D hand pose regression problem is often treated as classifying multiple parts or regressing multiple joint locations, this property is definitely favourable.

The injected randomisation prevents over-fitting without parameter regularisation, conveniently offering good generalisation performance on new unseen samples. Note that this point is a bit subtle, for whether over-fitting is good or bad becomes somewhat debatable lately. This is due to the recent outburst of big data, which also leads to the advent of deep learning. However, for a 26 DoF problem like 3D hand pose estimation, it is almost impossible to generate *enough* training data to cover the whole function

space. Therefore DF is a more advantageous choice.

DF can nicely scale up to a large training set. For a high DoF problem like 3D hand pose, a large (though never large enough) dataset is mandatory for parameter space coverage. Plus the ability of training with unbalanced, biased or missing training data is fairly useful with real-life dataset.

The node-splitting scheme can efficiently handle high dimensional input data, without a dimension-reduction preprocessing step. This is rather handy when the inputs are images or depth maps. Even when the input data have multiple modalities (RGB and depth), DF can efficiently handle them without extra computation, which provides a prerequisite for real-time speed.

With a little modification, DF can be applied to a series of different tasks like classification, regression or clustering. This opens up quite a few doors for us to exploit its potential, resulting in some of the main contributions in this thesis.

The disadvantage of DF is the lack of theoretical support, which is probably due to the difficulties in formulating the whole algorithm in an analytical form. As a result, sometimes one can not avoid choosing the parameters (number of trees, maximum depth, or number of split trials) empirically and heuristically.

3.4 Classification and Regression

In this section, we discuss the quality function and predictors under the context of classification and regression tasks.

Classification

Whether a DF serves as a classifier or regressor is determined by the quality function $Q(\cdot)$ and predictor $L(\cdot)$. In the case of classification forest, information gain is often used as the quality function:

$$Q_{cla}(\mathcal{D}) = H(\mathcal{D}) - \sum_{k}^{\{lc,rc\}} \frac{|\mathcal{D}^k|}{|\mathcal{D}|} H(\mathcal{D}^k), \tag{3.1}$$

where H stands for the Shannon entropy, such that $H = -\sum_{c \in \mathcal{C}} \log(p(c))$. And $p(c)$ is the probability of $\mathbf{x}$ belongs to class c. Note that in practice, since $\mathcal{D}$ is the same for all feature candidates, we can skip computing the entropy $H(\mathcal{D})$. For notation completeness, we will keep this term in formulas throughout this thesis.

At each *leaf node* , a predictor model is stored. In the case of classification, the predictor just need to aggregate the probability histogram $p(c|\mathbf{x})$ from all trees. The simplest, also the most popular way is to average the prediction from all trees, as defined below,

$$L(c|\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|\mathbf{x}), \tag{3.2}$$

where T is the number of trees.

Regression

When performing regression, each sample $\mathbf{x}$ is labelled with an offset vote $\mathbf{j} = \mathbf{y} - \mathbf{x}$. Hence $\mathcal{D}$ consists of a set of training tuple $\mathcal{D} = \{(\mathbf{x}, \mathbf{j})\}$. In this case, the training objective is to minimise the label variance of $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$. For training efficiency, it is often assumed that the dimensions in $\mathbf{j}$ are independent. Thus trace of the covariance matrix is commonly used as an approximation, such that,

$$Q_{reg}(\mathcal{D}) = \mathrm{tr}(\mathrm{cov}(\mathcal{D})) - \sum_{k}^{\{lc,rc\}} \frac{|\mathcal{D}^k|}{|\mathcal{D}|} \left(\mathrm{tr}\left(\mathrm{cov}(\mathcal{D}^k\right)\right)), \tag{3.3}$$

where $\text{cov}(\mathcal{X})$ is the covariance matrix of $\mathcal{X}$ and tr is the trace function. In practice, the trace can be efficiently calculated via

$$\text{tr}(\text{cov}(\mathcal{X})) = \sum_{(\mathbf{x}_i, \mathbf{j}_i) \in \mathcal{X}} \|\mathbf{j}_i - \bar{\mathbf{j}}\|^2 , \tag{3.4}$$

where $\bar{\mathbf{j}}$ is the mean vector of all the offsets in $\mathcal{X}$. Similar to the classification case, the calculation of the first term $\text{tr}(\text{cov}(\mathcal{D}))$ can be skipped, but we keep it in formulas for notation completeness.

In pose regression problems, prediction is usually done via Hough voting [Gall et al., 2011]. After the input samples propagated down each tree, we can gather the offset votes at each arrived leaf node and form a location vote set $\mathcal{Y}_j$ for each joint j, such that $\mathcal{Y}_j = \{\mathbf{y}_j = \mathbf{j}_j + \mathbf{x}_j\}$. A Gaussian Parzen density estimator is then applied to $\mathcal{Y}_j$ as our final predictor.

$$L_j(\mathbf{y}') = \sum_{\mathbf{y}_j \in \mathcal{Y}_j} \frac{1}{n b_j} \exp\left(- \left\| \frac{\mathbf{y}' - \mathbf{y}_j}{b_j} \right\|_2^2 \right) , \tag{3.5}$$

where $\mathbf{y}'$ is all possible 3D locations, n is the amount of training samples that compose $\mathcal{Y}_j$, and b_j is the bandwidth for estimator. There are two problems with (3.5): 1) Exhaustively check all possible locations is too time-consuming. 2) If provided a rather big training set, one will end up with large amount of offset vectors in each leaf node. This not only demands memory but also slows down the voting process. To address these, [Girshick et al., 2011] employ Meanshift [Cheng, 1995] during testing to speed up this mode-seeking process. During training, they also use Meanshift to cluster $\{\mathbf{j}\}$, and then take the first 1 or 2 modes (depending on confidence) as leaf model, which significantly reduces the number of Hough votes. In this thesis, we follow this scheme.

3.5 Feature

At each internal node of a decision tree, a binary split function $h(\cdot)$ is required to route the data to its left or right child node. Formally it is defined as,

$$h(\mathbf{x}) = \begin{cases} 1 & \phi(\mathbf{x}) > \tau \\ 0 & \text{Otherwise,} \end{cases} . \tag{3.6}$$

where ϕ is a feature function which extracts useful informations in the form of a scalar from $\mathbf{x}$ and compares it with a threshold τ. This comparison results in $\{0, 1\}$, indicating $\mathbf{x}$ going to the left or right child node.

Now the question left is the choice of ϕ. Different types of feature have been extensively discussed in [Criminisi and Shotton, 2013]. In general there are two types: linear and non-linear.

Linear

The linear feature function can be formulated as,

$$\phi(\mathbf{x}|\mathbf{w}) = \mathbf{w} \cdot \rho(\mathbf{x}) \, , \tag{3.7}$$

where $\rho(\mathbf{x})$ represents the homogeneous vector form of an $N \times N$ patch centred at $\mathbf{x}$, and $\mathbf{w}^T$ is a $|N^2 + 1|$ scalar weight vector in homogeneous coordinates, whose value is decided during training. By comparing with τ, (3.7) linearly separates the data into left and right child nodes.

Non-linear

If not linearly separable, a non-linear feature function defined below can be used,

$$\phi(\mathbf{x}|\mathbf{K}) = \rho^T(\mathbf{x})\mathbf{K}\rho(\mathbf{x}) + b \, , \tag{3.8}$$

where $\mathbf{K} \in \mathbb{R}^{(N^2+1)\times(N^2+1)}$ is a matrix representing a conic section in homogeneous coordinates. The conic section acts like a non-linear hyper-surface separating the input data.

In practice, since $\mathbf{x}$ often need to pass multiple tree nodes (feature functions), it is empirically proved that a weak linear feature has already work well. Furthermore, the following simplified version of linear features are quite popular.

Axis-aligned

An axis-aligned feature only selects one dimension of the input $\mathbf{x}$. Geometrically it looks like a hyperplane parallel to all axes except for the chosen one, hence the name. More formally, we define,

$$\begin{aligned} &\phi(\mathbf{x}|\mathbf{w}) = \mathbf{w} \cdot \rho(\mathbf{x}) \,, \\ &w_i \in \{0,1\} \text{ for } w_i \in \mathbf{w}, \\ &|\mathbf{w}| = 1, \end{aligned} \tag{3.9}$$

which forces $\mathbf{w}$ to be a binary vector with only one bit is 1. In some literature, the axis-aligned feature is also formulated as,

$$\phi(\mathbf{x}|\mathbf{u}) = I(\mathbf{x} + \mathbf{u}) \,, \tag{3.10}$$

where $\mathbf{u}$ is a 2D offset within the aforementioned patch $\rho(\mathbf{x})$, and $I(x)$ retrieves the value at a specific position x. This 'offset' version of formulation is equivalent to (3.9) but in a simpler form.

Two-pixel difference

A linear feature that is lightly more complex than the axis-aligned one is called two-pixel difference, which is first proposed by [Lepetit et al., 2005], applied to 2D segmentation problems [Shotton et al., 2008]. Formally, we define the 'offset' version of

two-pixel difference as,

$$\phi(\mathbf{x}|\mathbf{u},\mathbf{v}) = I(\mathbf{x}+\mathbf{u}) - I(\mathbf{x}+\mathbf{v})\,, \tag{3.11}$$

where $\mathbf{u}$, and $\mathbf{v}$ are two offsets within the patch $\rho(\mathbf{x})$. Not only does this feature use one more dimension than the axis-aligned one, the different signs of the two terms is actually a discrete differentiation operator, which is an *approximation of the gradient*. Later on in [Shotton et al., 2011], (3.11) is extended to a 3D version as,

$$\phi(\mathbf{x}|\mathbf{u},\mathbf{v}) = I(\mathbf{x}+\frac{\mathbf{u}}{I(\mathbf{x})}) - I(\mathbf{x}+\frac{\mathbf{v}}{I(\mathbf{x})}), \tag{3.12}$$

where the offsets are normalised by the depth value of input (or patch centre) $\mathbf{x}$. This makes the feature scale invariant and thus even more effective. The good trade-off of two-pixel difference on efficiency and effectiveness has been proven by prior arts [Shotton et al., 2011, Girshick et al., 2011, Sun et al., 2012]. In this thesis, we have not investigated other form of features but simply adopted (with slight modification) of this 3D version of two-pixel difference feature. More thoughts regarding this will be discussed in the future work (see Chapter 8).

3.6 Training and Testing

In this section, we provide generic algorithms for training/testing DF, which will serve as starting templates for following chapters.

During training, a set of data samples $\mathcal{D}$ is partitioned into $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$ for left lc and right childnode rc respectively. This partitioning is repeated recursively till the stopping conditions are met. Each partition generates a *split node* (non-leaf node) by choosing a feature ϕ^* from a set of feature candiates Φ with a quality function Q [1], such that,

[1] Strictly speaking, we are not choosing ϕ but values of its parameters. For notation simplicity, ϕ is also used to represent its parameters throughout this thesis.

$$\phi* = \underset{\phi \in \Phi}{\operatorname{argmax}}\, Q(\mathcal{D}, \phi). \tag{3.13}$$

Note that Q can be (3.1) for classification or (3.3) for regression. When it is not possible to split further, i.e. , the stop condition has been met, a *leaf node* and a corresponding predictor (classification or regression) are created. The training process is described in Algorithm 1.

Testing procedure is relatively simple and basically following the standard decision tree propagation. At each *split node n*, the learned feature function ϕ decides the input $\mathbf{x}$ going to left or right branch. Upon reaching a *leaf node* , a stored function $L(\cdot)$ predicts a decision value (discrete for classification and continous for regression). And finally results from all trees are aggregated to make a final decision. This process is described in Algorithm 2.

Algorithm 1 Training a tree in DF

Require: A set of training samples $\mathcal{D}$; A quality function $Q(\cdot)$; Maximum depth D.
Ensure: A tree t

procedure Grow($\mathcal{D}$)
  Let $d = 0$ ▷ First stage of training
  Split($\mathcal{D}, d$)
end procedure

function Split($\mathcal{D}$, d)
  Randomly propose a set of features Φ.
  for all $\phi \in \Phi$ **do**
    Partition $\mathcal{D}$ into $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$ by ϕ.
  end for
  Use (3.13) to . For classification tasks, ; for regression, use (3.3).
  if classification **then**
    Use (3.1) as the quality function to select the optimal $\phi*$.
  else if regression **then**
    Use (3.3) as the quality function to select the optimal $\phi*$.
  end if
  if $d > D$ **then** ▷ Use maximum tree depth as the stop condition.
    if classification **then**
      Add a *leaf node* with a class distribution histogram into t.
    else if regression **then**
      Add a *leaf node* with a set of Hough votes $\{\mathbf{j}\}$ into t.
    end if
  else
    Partition $\mathcal{D}$ into $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$ by $\phi*$.
    Add a *split node* with $\phi*$ into t.
    Split($\mathcal{D}^{lc}$, $d + 1$)
    Split($\mathcal{D}^{rc}$, $d + 1$)
  end if
  Return
end function

Algorithm 2 Testing

Require: An input sample $\mathbf{x}$; A forest F.
Ensure:

procedure TEST($\mathbf{x}$)
  for all tree $t \in F$ **do**
    Let $n = t \rightarrow root$.
    PROPAGATE($\mathbf{x}$, n)
  end for
  if classification **then**
    Use (3.2) to aggregate results from all trees.
  else if regression **then**
    Use (3.5) to aggregate results from all trees.
  end if
end procedure

function PROPAGATE($\mathbf{x}$, n)
  if n is a *leaf node* **then**
    if classification **then**
      Return the probability histogram p_n.
    else if regression **then**
      Return the stored votes $\{\mathbf{j}\}$.
    end if
  else
    if $\phi(\mathbf{x}) == 1$ **then**
      Let $n = n \rightarrow lc$.
    else
      Let $n = n \rightarrow rc$.
    end if
    PROPAGATE($\mathbf{x}$, n)
  end if
end function

CHAPTER 4

Datasets and Evaluation Protocols

Contents

4.1 Proposed Datasets

When we first started investigating this problem in 2011, there were no public dataset available. Thus we had to collect our own datasets. Note that the main theme of this thesis is to discuss discriminative learning-based methods. Hence a training set is also needed. Regarding this, a few important issues are listed below.

Whether to use synthetic or realistic data is the first dilemma. For testing, it is definitely more appropriate to use real data. And since testing sets are usually smaller, it is acceptable to manually label them. However in the case of training, since we usually need a large quantity of data to capture different variances, it is quite tedious to manually annotate them. By generating synthetic data with mesh models, one can easily acquire a large amount of data and labels, but they do not neccesarily reflect real world scenarios.

How do we annotate if we choose to collect real data? Manual labelling is generally more accurate, but requires a lot of efforts. For one thing, how to tackle the ambiguity of occluded joints is not clear. For another, if the task is to perform pixel-wise classification, one needs to annotate each foreground pixel which is infeasible. In early days, glove-based or marker-based methods are used to provide annotations [Sturman and Zeltzer, 1994]. However, these devices not only hinders the natural movements, but also change the appearance of input data, let alone the high cost. Recently [Sharp et al., 2015] use a painted glove with 6 different colours for the palm and fingers. While this does not alter the depth data, it only gives pixel classification labels. Using model-fitting methods to acquire labels is an economic choice. However the accuracy is usually not guaranteed.

Since it is not easy to train and test across sensors, choosing one becomes a non-trivial problem. Different sensors exhibit different characteristics. We refer readers to a survey for detailed comparison between structured-light (SL) and time-of-flight (ToF) sensors. Since human skin is non-reflective and the hand size is relatively small, ToF sensors are a better choice. However, consumers-affordable ToF sensors were not available until recently. Hence datasets captured by both types of sensor are discussed in this chapter.

As discussed in Section 1.2, hand shape and size variations need to be considered too. For synthetically generated datasets, we only need to randomly vary the shape pa-

rameters before rendering, according to studies such as [Lee et al., 2009] . For realistic data, training and testing subjects need to be chosen carefully to cover different sizes, which is not easy.

4.1.1 Imperial College Vision Lab (version 1)

In our work [Tang et al., 2013], we propose the Imperial College Vision Lab version 1 (ICVL v1) dataset. Primesense™ Carmine was used as it was the most popular short-range sensor at that time. However, in trial runs we notice quite a few different types of sensor noises. They introduce a large discrepancy between synthetic and real data. Unlike the work from [Xu and Cheng, 2013], which explicitly models these noises and recreate them in synthetic data, we apply Transfer learning to fill this gap. This idea requires both synthetic and real data (only a small portion of real data need to be labelled), as well as a synthetic counterpart for each labelled real data. The detailed hand pose regression approach will be presented in Chapter 5. In this section we only introduce the collection and annotation process.

Realistic data

Subjects are required to perform different poses with their left hand in front of the sensor. Poses are not restricted but randomly chosen by the subjects themselves, hence cover quite a few everyday natural gestures. For testing sequences, depth images are captured with 30 FPS. For training data, frames are sampled at a rate of 5 FPS to reduce similar poses. To be able to compare with tracking-based methods such as [Oikonomidis et al., 2011a], subjects are requested to start with the frontal, open hand pose. Although this initialization pose is not required in our method, it does provide a 'starting pose' (see row 1 in Fig. 4.2) for us to measure sizes of subjects' hand, for in this case all joints are visible (not self-occluded).

When labelling, we develop a simple tool for us to manually click on 2D joint locations on the images in a fixed order. However, the annotators need to visually decide whether the current joint is visible or self-occluded. If it is visible, he/she should click with left mouse button, and Z-value of that 2D position will be retrieved to have a 3D position. If it is occluded, the annotator clicks with right button on an estimated location and only 2D position will be recorded.

After having 3D positions for visible joints and 2D positions for self-occluded joints, one can then fit a skeleton model to have 3D locations for the self-occluded joints. A more proper way of fitting is using a generative model, as in the inverse kinematic step of [Tompson et al., 2014]. In here we choose an approximate method:

1. A large pose database (65K different poses) is pre-generated, with 3D locations for all joints. Note that because this database is viewpoint-invariant (only 20 DoF), this amount of poses has already a good coverage of the pose space.
2. As mentioned before, the scale factor between the poses in database and the subjects' hand size is computed with the 'starting pose'.
3. For each frame, given the scale factor, use visible joint locations to match a pose in database with nearest neighbour search.
4. If multiple choices are returned, use 2D locations of occluded joints to choose one.
5. Replace the position of occluded joints with the corresponding ones in database.

Synthetic data

We choose SmithMicro Poser Pro as the tool for generating synthetic data. It provides skinned human models with adjustable shape and size. Also it has a built-in python

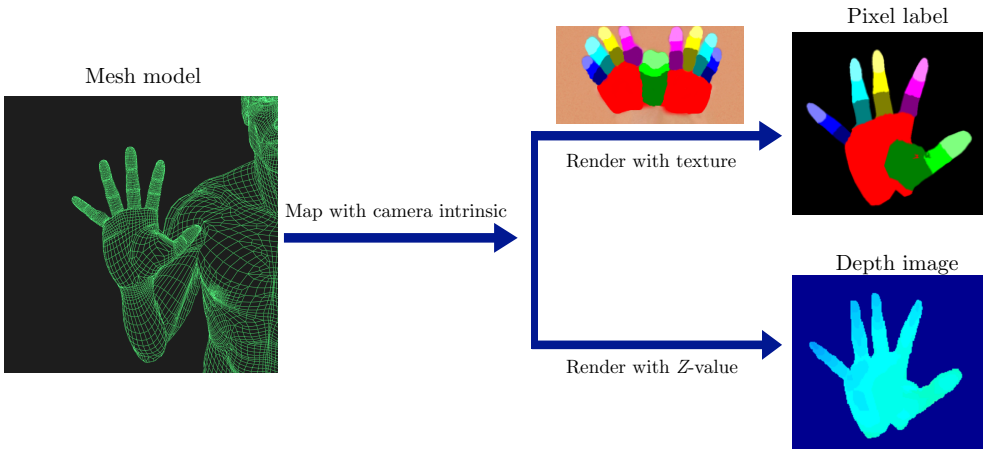


Figure 4.1: The process of generating synthetic data.

engine that we can easily use to generate poses. Similar to the real data, we also do not restrict the types of poses but randomly generate the joint angles, among which the metacarpophalangeal (MCP)/proximal Interphalangeal (PIP)/distal Interphalangeal (DIP) joints follows the kinematic constraints in [Pavlovic et al., 1997].

To render depth images, we simply modify the ray casting algorithm [Roth, 1982] to render with z-value for every pixel. Rendering pixel classification labels is a bit more tricky. We manually paint the corresponding part in texture map for each hand part with different colors. And then rendering without illumination will give us the pixel-wise label map. This process is illustrated in Fig. 4.1.

Pairing data

For each labeled real depth image, we can simply apply its pose parameters to a synthetic model and generate a synthetic counterpart depth image. This is useful in the method described in Chapter 5. A few pairs of synthetic-real images from the ICVL v1 dataset are shown in Fig. 4.2.

4.1.2 Imperial College Vision Lab (version 2)

In 2014, as ToF sensors become more and more popular, sensor noise is not a major problem anymore. Also obtainable generative methods provide an economic way to label real data. Hence in [Tang et al., 2014], we chose to collect a real dataset called 'Imperial College Vision Lab version 2' (ICVL v2).

We use Intel®'s *Creative Interactive Gesture Camera* as a depth sensor for capturing training and testing data. As a consumer ToF sensor, it captures depth images at a lower noise level than structured-light sensors at a close range, making it ideal for hand pose estimation. For labelling, we utilise [Melax et al., 2013] to obtain a preliminary pose for each frame. To improve annotation accuracy, we first manually adjust shape of the mesh model for every subject with the aforementioned 'starting pose'. And then for each frame, we increase the iteration by 60 times (1 FPS) to make the results converge better.

For training, we have collected sequences from 10 different subjects with varying hand sizes by asking each subject to make various hand poses with an illustration of 26 different postures shown as aid. Each sequence was then samples at 3 FPS producing a total of 20K images and by additionally applying in-plane rotations to this set, the final dataset contains 180K ground truth annotated training images. For testing, we have collected two sequences (denoted sequence A and B) each containing 1000 frames capturing a vast array of different poses with severe scale and viewpoint changes. Furthermore, as [Melax et al., 2013] is tracking based and requires initialisation ('starting pose'), in order to do a fair comparison both test sequences start in this way. Examples from the 'ICVL v2' datasets are shown in Fig. 4.3.

The downsides of this dataset are also obvious. The accuracy upper bound relies on the annotation tool. For those frames that [Melax et al., 2013] easily fails, e.g. , fist poses from different views, samples are scarce. Moreover, as shown in Fig. 4.3,

for some of those labeled frames, finger tip positions are a bit off. These are common issues shared by datasets that utilize automatic fitting tools.

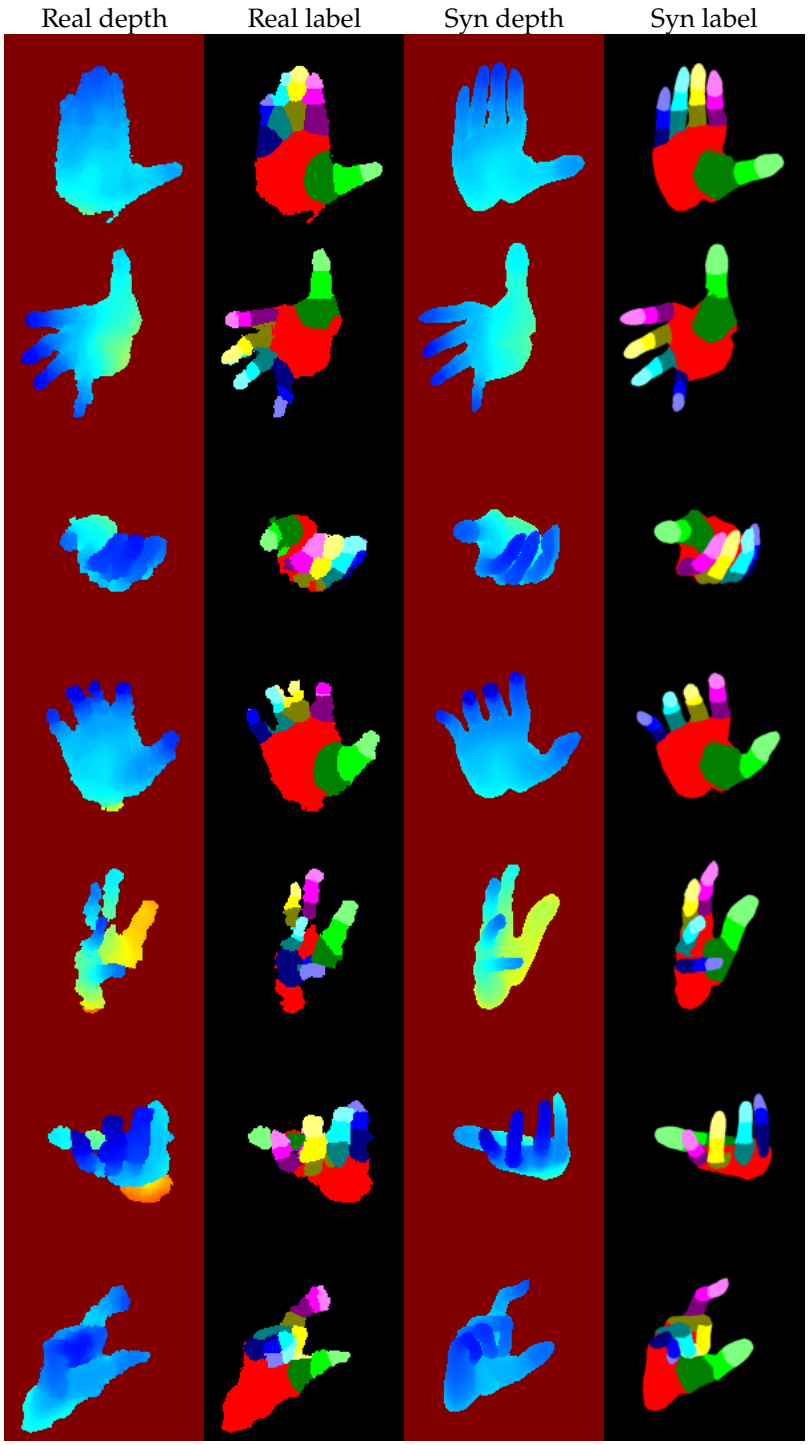


Figure 4.2: Examples from the ICVL v1 [Tang et al., 2013] dataset. Real and synthetic training pairs are shown.

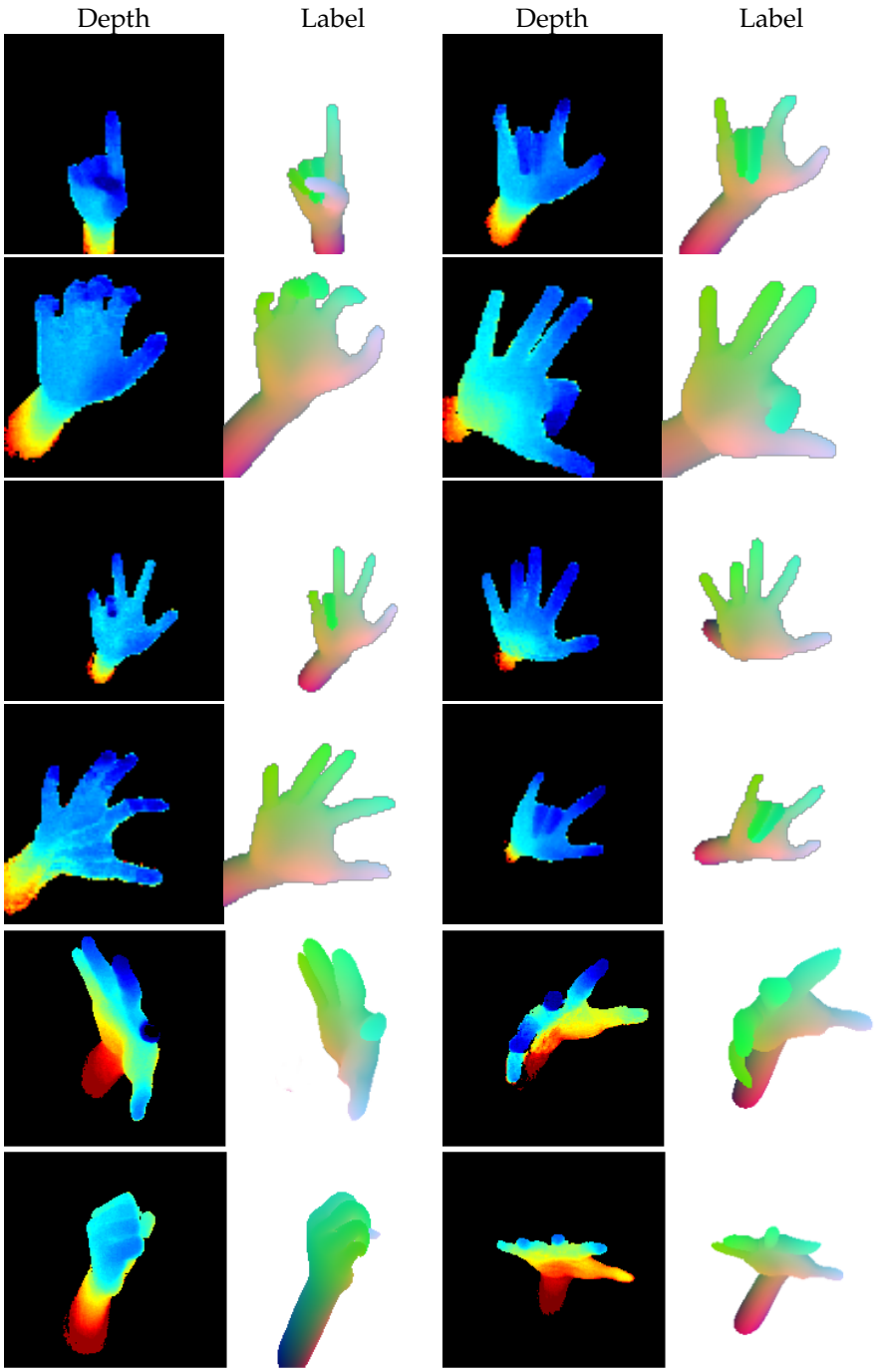


Figure 4.3: Examples from the ICVL v2 [Tang et al., 2014] dataset. Ground-truth poses are visualized as rendered synthetic images - same for all figures in this thesis.

4.2 Other Public Datasets

Two public datasets are also used in the experiments of this thesis.

4.2.1 New York University Dataset

[Tompson et al., 2014] from New York University propose a dataset called New York University Dataset [Tompson et al., 2014] (NYU) , with real data captured by Kinect. Since it is the first method that applies Deep learning to 3D hand pose regression problem, this dataset provides an important baseline for comparison. Given the sensor, it processes similar noise artifacts as ICVL v1. In fact, during annotation they utilize 3 Kinects, the interference makes the noise problem even more serious.

This method in [Tompson et al., 2014] first applies CNN to predict 14 keypoint locations, and then use PSO to fit a skeleton to these keypoints and retrieve the pose. In the dataset, only 3D locations of those 14 keypoints are provided as annotations. To be able to train with angular-based methods, as instructed in their paper, we also use PSO to retrieve the angular poses, as shown in Fig. 4.4.

For each sensor, the training set has 79K images and testing set has 8252 images. It is worth noted that the training set has only one subject, whilst the testing set has two (one of them is the same as training). Thus it is rather overfitting.

4.2.2 Microsoft Research Synthetic Hand Dataset

In [Sharp et al., 2015], a synthetic dataset was proposed, called Microsoft Research Synthetic Hand Dataset [Sharp et al., 2015] (MSHD). The dataset was generated with camera intrinsics of Kinect2, which is a ToF sensor. Training set has 100k images whilst

testing set has 1k images. They provide pose parameters in 3D Euclidean and angular forms, as well as pixel-wise classification labels. Since both training and testing data are synthetic, one might think it is an easy dataset. On the contrary, MSHD is very challenging because of the following aspects.

- It is the first hand dataset that covers full viewpoint (pitch, yaw and roll of palm rotation) space. The synthetic nature makes it easy to achieve so in both training and testing data. This is useful in many real world applications. However, the amount of training data becomes relatively sparse considering the covered parameter space, which makes a learning-based method rather difficult to achieve low error.
- When generating, it takes shape and size into consideration by randomly choosing one of the 13 personalized mesh models. These models have been fitted to hands of 13 different subjects using [Taylor et al., 2014]. This adds more variance into this already high-dimensional problem.
- All testing images are randomly generated without temporal information, which makes it not possible to use tracking.
- Last but not the least, the rendering process explicitly simulates the noise of ToF sensors.

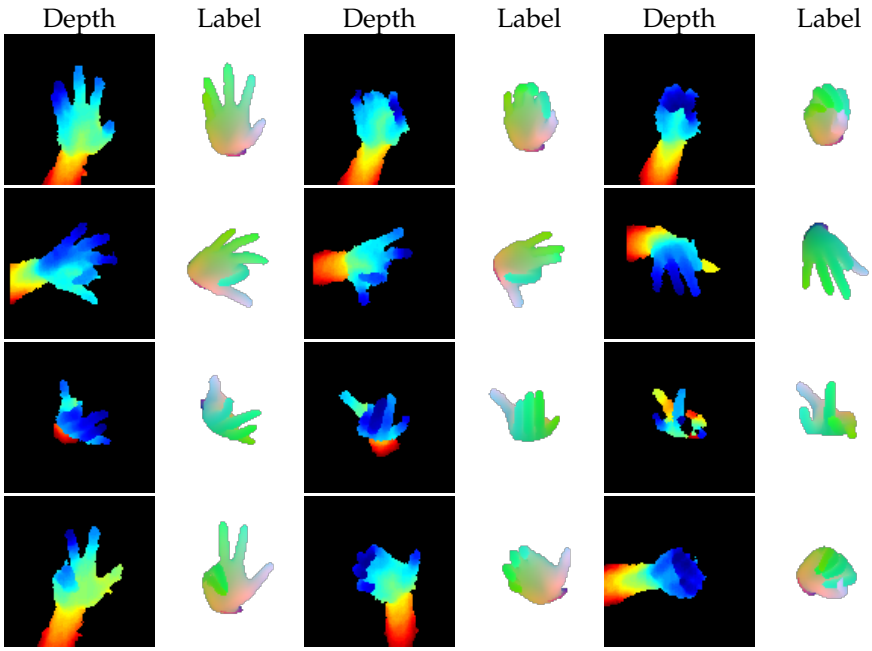


Figure 4.4: Examples from the NYU [Tompson et al., 2014] dataset.

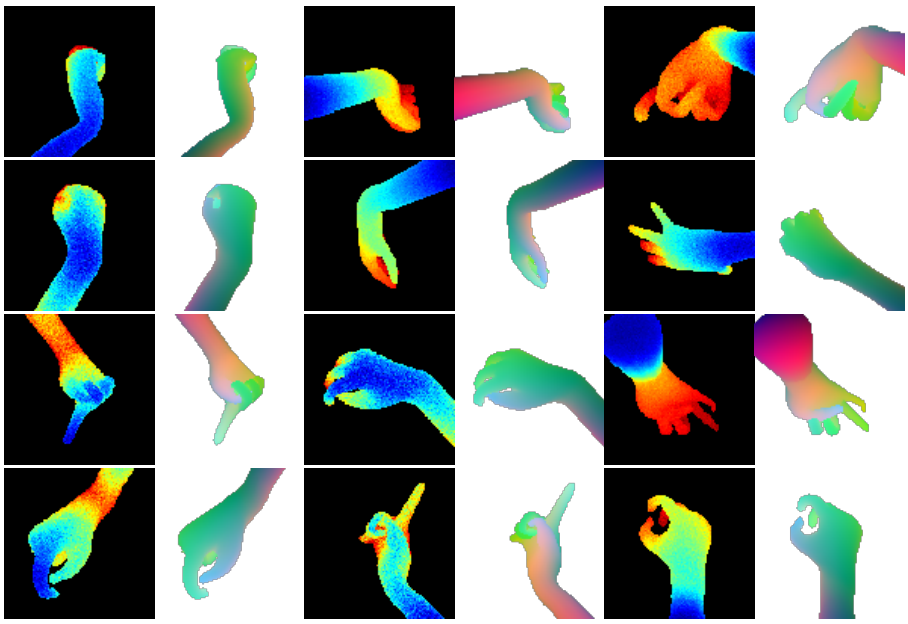


Figure 4.5: Examples from the MSHD [Sharp et al., 2015] dataset.

4.3 Evaluation Criterion

Quite a few different evaluation criteria have been proposed in both human body pose and hand pose communities. In this section we choose and discuss a few popular options that are adopted in the later chapters.

Despite there are two different types of output parameter $\mathbf{y}$, angular and 3D Euclidean, it is sensible to choose Euclidean as error metric because: (1) Angle can be easily converted to Euclidean, but it is not too easy the other way around for lack of kinematic constraints. (2) Joint angles are dependent-even if the predicted DIP angle is the same as ground-truth, if the PIP angle is different, the final finger tip position will be different. (3) Although joint bending can be in angular form, the Euclidean position of wrist has to be given as global translation. So it is technically a 'mixture' of two types, which is not straight-forward for computing a single error value. Therefore in all evaluations throughout this thesis, the implication is that $\mathbf{y}$ has been converted to *3D Euclidean form*.

Unfortunately there is little knowledge about what accuracies are needed in practice. Intuitively, for some applications that require only discrete poses, i.e. , sign language recognition, a few centimetre error in joint positions may not be a big issue. Whereas for applications like virtual object manipulation, recovering precise finger joints are crucial, hence even 1 cm error might not be acceptable. Since the HCI community has only begun to use 3D hand poses as input recently, we can expect the feedback of accuracy requirements from them before long.

4.3.1 Per-frame accuracy

We start by discussing how to evaluate the accuracy of each frame, under the assumption that there is only one hand in the image.

Pixel classification error

For some dataset that provides only classification labels (the *Finger paint* dataset in [Sharp et al., 2015]), or the method itself has a classification step [Shotton et al., 2011, Tang et al., 2013], it is necessary to evaluate the pixel classification error. Recall that in Section 1.1 we have defined the hand part classification label as p. For classification accuracy, we adopt the average per-class accuracy from [Shotton et al., 2011], which is given by the following equation,

$$\Delta_{cla} = \frac{1}{|\mathcal{O}|} \sum_{j \in \mathcal{O}} \frac{1}{N_j} \sum_{i=1}^{N_j} 1(p_i^j \neq \hat{p}_i^j), \tag{4.1}$$

where $\mathcal{O}$ is a set of hand parts, N_j is the amount of groundtruth pixels belongs to part j, p_i and $\hat{p}_j$ are the prediction and grouthtruth label of a sample respectively, and $1(\cdot)$ is the indicator function. This metric takes into account the varying amount of pixels of different parts. This metric has been used in classification methods such as [Tang et al., 2013] (see Chapter 5).

Pose error

Let $\mathcal{J}$ be the set of joints we need to predict. The final joint locations $\mathbf{y}$ can then be decomposed into $\{y_j \in \mathbf{y} : j \in \mathcal{J}\}$. Meanwhile we can denote the groundtruth as $\{\hat{y}_j \in \hat{\mathbf{y}} : j \in \mathcal{J}\}$

For a single frame, assuming only one hand, the error can be measure by average 3D Euclidean offsets, such that,

$$\Delta_{avg} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \left\| \mathbf{y}_j - \hat{\mathbf{y}}_j \right\|_2 \tag{4.2}$$

A more challenge measurement is to consider the maximum joint error only,

$$\Delta_{max} = \max_{j \in \mathcal{J}} \left\| y_j - \hat{y}_j \right\|_2 \tag{4.3}$$

4.3.2 Set accuracy

In practice we need to evaluate a method on a set of frames, either with or without temporal information. A set-forth choice is to plot either Δ_{avg} or Δ_{max} against time t. However, for single frame-based methods, such as [Tang et al., 2013, Tang et al., 2014], the fluctuation may be too severe to visualise. A clearer way is to plot the *cumulative moving average error* against time t,

$$\Delta_{\delta}^{t} = \frac{\Delta + |t-1|\Delta_{\delta}^{t-1}}{|t|} \tag{4.4}$$

where Δ_{δ} can be either Δ_{avg} or Δ_{max}. In this way the trend can be easily seen. This metric has been used in [Tang et al., 2014] (see Chapter 6).

A much more challenging criterion called *worst case accuracy* was proposed by [Taylor et al., 2012], which measures the proportion of test frames that have error Δ_{δ} within a threshold, such that,

$$\Delta(\tau) = \frac{|\{\Delta_{\delta}(I)|\Delta_{\delta}(I) < \tau, I \in \{I\}\}|}{|\{I\}|} \tag{4.5}$$

where τ is the threshold, Δ_{δ} can be either Δ_{avg} or Δ_{max}. By varying the threshold τ, one can generate a curve indicating the proportion of easy and difficult frames. This metric has been adopted in [Tang et al., 2014] (see Chapter 6) and [Tang et al., 2015] (see Chapter 7).

4.4 Summary

In this chapter, we proposed two datasets, ICVL v1 and v2, serving different purposes. Collection and annotation methods are explained in detail. In addition to that, two publicly available datasets, NYU and MSHD are also introduced. Each dataset has its

pros and cons summarized in Table 4.1. Apparently none of them covers all the aspects for the time being. Hence one may consider experiments with multiple datasets for a proper comparison. Apart from the datasets, several evaluation criteria for different scenarios are also discussed.

Table 4.1: Description of datasets

Dataset	ICVL v1	ICVL v2	NYU	MSHD
Sensor	Primesense	Intel	Primesense	Kinect2
Depth resolution	640×480	320×240	640×480	512×424
Distance	near	near	far	far
Label	pixel classification & 3D locations	3D locations	3D locations	pixel classification & joint angles
Source	synthetic & real	real	real	synthetic
Viewpoint	medium	restricted	medium	full
Temporal	slow	fast	slow	individual frames
Subject	3	10	2	13
Training images	418,500	180,000	79,000	100,000
Testing images	2,000	2,000	8,252	1,000

CHAPTER 5

Semi-supervised Transductive Regression Forest

Contents

5.1 Overview

As discussed in Section 1.2, if a structured-light depth sensor is used for capturing depth images, the discrepancy between synthetic and realistic training data is inescapable. Shown in Fig. 5.1, missing parts and quantisation error is common in real data, especially at small, partially occluded parts such as finger tips. Regardless of the object distance, missing values are created when a complete structured light patterns fail to project on the occlusion boundaries. Unlike sensor noise and depth errors in [Shotton et al., 2011, Girshick et al., 2011, Baak et al., 2011], these discrepancies between synthetic and realistic data cannot be repaired or smoothed easily. Consequently, one cannot simply train a classifier with synthetic data and expect it to work well on real data. For instance, if only trained with synthetic data, the conventional decision forest (DF) training process may choose a feature as in Fig. 5.1 (e), which does not work on real data.

Moreover, due to noise, occlusions and the inherent complex structure of human hand, manually labelled realistic data are extremely costly to obtain. Existing state-of-the-arts often resort to synthetic data, e.g. [Keskin et al., 2012], or model-based optimisation, e.g. [de La Gorce et al., 2011, Oikonomidis et al., 2012]. Nonetheless, such solutions do not consider the realistic-synthetic discrepancies, their performances are hence affected. Besides, the noisy realistic data make joint detection difficult, whereas in synthetic data joint boundaries are always clean and accurate.

In this chapter, addressing the above challenges, we present a novel semi-supervised transductive regression (STR) forest to incorporate the relationship between realistic and synthetic data into forest training. This process is known as *transductive transfer learning* [Pan and Yang, 2010]: A transductive model learns from a *source domain*, e.g. synthetic data ; on the other hand, it applies *knowledge transform* to a dif-

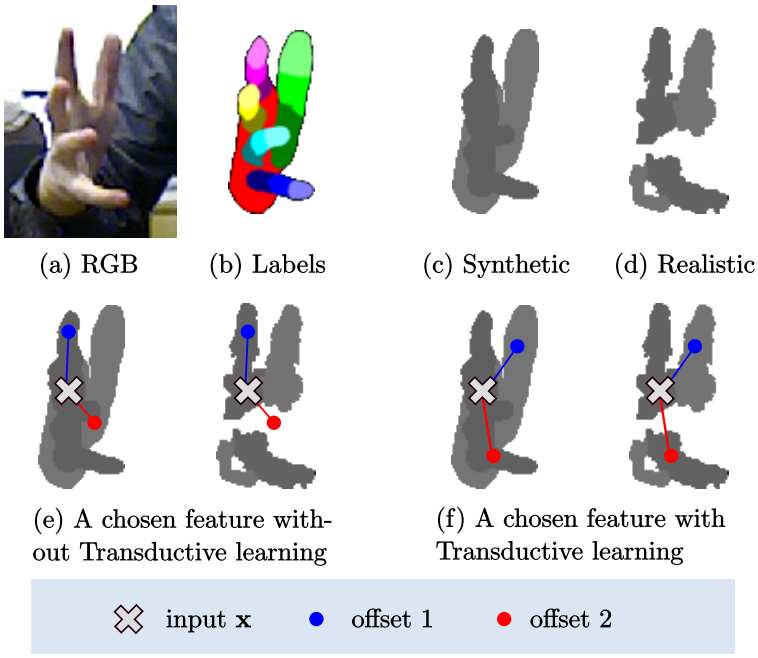


Figure 5.1: Example of training data and effect of Transfer learning: The ring finger is missing due to occlusions in (d), and the little finger is wider than the synthetic image in (c). The second row: examples of feature (3.12). Without Transfer learning (e), the feature that trained on synthetic data may not work on real data; with Transfer learning (f), the training procedure will favour the features that work well on both domains.

ferent but related *target domain*, e.g. realistic data, in the testing stage.

Since the fingers and palm of a human hand share very similar texture, automatic segmentation of the hand is very difficult in traditional videos. In addition, due to occlusions and viewpoint changes, Traditional video-based approaches rely on low-level silhouettes and edge features [Rosales et al., 2001, Chua et al., 2002, Athitsos and Sclaroff, 2003, Stenger et al., 2006], which require extra computational effort in hand detection and segmentation and limited amount of recognisable poses. Depth images, on the other hand, provide straightforward hand and finger segmentation by providing 2.5D information of the scene. Although range sensors or stereo cameras are required to capture depth images, they are currently leveraged to optimise the performance of

hand pose estimation.

The STR forest learning algorithm is also semi-supervised. It learns the noisy appearances of real data from both labelled and unlabelled data points. As a result, it benefits from the characteristics of both domains. The STR forest not only captures a wide range of poses from synthetic data, it also achieves robust performances in challenging environments by learning the noisy, irregular appearances of real data.

In addition to the proposed STR forest, a pseudo-kinematic joint refinement algorithm is proposed to handle occlusions and noisy articulations efficiently. Hypotheses of 3D hand pose estimates from the STR forest are verified by this data-driven technique, which models the structure of human hands, without computing sophisticated inverse kinematics.

5.1.1 Contributions

The main contributions of the proposed 3D hand pose estimation framework are as follows.

- **Real-Synthetic fusion**

  Handling the issue of noisy inputs, a transductive learning algorithm for 3D hand pose estimation, namely the STR forest, is proposed. The STR forest relates the characteristics of real and synthetic training data by integrating the idea of transductive learning into Hough forest (HF), improving robustness and pose coverage in real testing environments.

- **Semi-supervised learning**

  Training a discriminative hand pose recogniser requires a large and extensive training dataset. Hand-labelling of hand pose data, however, is costly and ineffective due to occlusions and complicated articulations of human hands. The

STR forest utilises both labelled and unlabelled data in its learning algorithm, improving estimation accuracy while maintaining a low labelling cost.

- **Coarse-to-fine training**

  The original training objective for pose regression is computationally demanding. We propose a coarse-to-fine hierarchy by switching among viewpoint classification, joint part classification and pose regression. The first two objectives are more efficient than the final pose regression objective. Moreover, an adaptive switching technique is employed to activate only one objective at a time.

- **Data-driven pseudo-kinematics**

  Traditional Hough forest algorithms do not model occlusions and structural consistency of articulations [Gall and Lempitsky, 2009]. A data-driven, pseudo-kinematic technique is therefore introduced to verify the feasibility of hand poses inferred from the preceding STR forest.

5.2 Related Work

Sensor noise

Published at the same time of this work, [Xu and Cheng, 2013] also tackles sensor noises. They first analyze and classify noises into two different types: (1) shadow around boundaries, caused by occlusion due to the monocular depth sensor setting; (2) missing pixels, as demonstrated in Fig. 5.1, for the size of that area is smaller than the structured-light pattern, which happens quite often around finger tips. And then they propose a solution to recreate these noises in synthetic training images. Similarly, [Sharp et al., 2015] also explicitly models the flying pixel [Hansard et al., 2012] of time-of-flight (ToF) sensors in their synthetic dataset. Unlike their approaches, in-

stead of explicitly model the source of noise, we choose to automatically learn feature functions that work well on both synthetic and real data.

Transfer learning

Transductive transfer learning is often employed when training data of the target domain is too costly to obtain. It has seen various successful applications [Pan and Yang, 2010], nonetheless it has not been applied to articulated pose estimation. In this work, real-synthetic fusion is realised by extending the idea of cross-modality learning by [Bronstein et al., 2011] to the proposed STR forest, where the training algorithm preserves the predefined associations between cross-domain data pairs, which are defined in Section 5.4.1.

Semi-supervised learning

Various semi-supervised forest learning algorithms have been proposed. [Criminisi and Shotton, 2013] measured data compactness to relate labelled and unlabelled data points. [Leistner et al., 2009] designed a margin metric to evaluate unlabelled data. In this chapter, a STR forest adaptively combines the aforementioned semi-supervised and regression forest learning techniques in a single frame work.

5.3 Approach Overview

The proposed 3D hand pose estimation framework is illustrated in Fig. 5.2. The training dataset used in the STR learning algorithm is explained in Section 5.4.1. Training instances are collected from a target domain (real depth images) and a source domain (synthetic depth images). Whilst data points in the fully labelled source domains are generated automatically with ground truth, only a small portion of the data points in

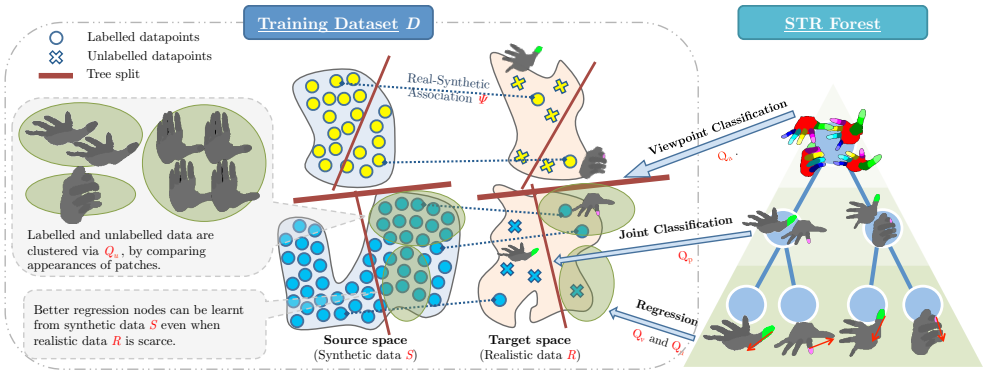


Figure 5.2: The proposed 3D hand pose estimation framework. The training dataset contains both real and synthetic depth images. The proposed STR forest is constructed from the training dataset to perform three tasks sequentially: viewpoint classification, hand part classification and joint regression.

the target domain are labelled manually and the remaining points are left unlabelled. The synthetic dataset and the real dataset are related by establishing associations from the labelled target data points to their corresponding source data points.

The proposed STR learning algorithm is presented in Section 5.4.2. Two new techniques are introduced to the traditional Hough forest (HF) [Gall et al., 2011]. Transductive real-synthetic associations is preserved throughout the learning process, such that the paired data points pass down to the same node. Furthermore, the distribution of labelled and unlabelled real data are described jointly in the proposed STR forest using an unsupervised learning term.

The STR forest is a hybrid random forest that performs classification and regression tasks adaptively within a unified algorithm, as shown in Fig. 5.2. At the top levels of each decision tree, it first classifies training patches of depth images, according to their view points. Subsequently, the second classification task takes place to classify patches with respect to their joint labels. Once the patches have been successfully classified, regression models are learned to estimate joint locations through a voting scheme.

Since the STR forest does not consider the physiological structure of human hands,

a data-driven joint refinement technique is introduced to refine the joint locations from the STR forest. The joint refinement algorithm is explained in Section 5.4.3.

5.4 Learning

5.4.1 Training datasets

To target the discrepancy between synthetic and realistic data, we propose a dataset (ICVL v1) that contains both kinds. The collection and annotation techniques, as well as some example images have been covered in Section 4.1.1. In this section, we define some dataset related concepts for following sections.

The training dataset $\mathcal{D}$ is a combination of real data $\mathcal{R}$ and synthetic data $\mathcal{S}$, as shown in (5.1):

$$\begin{aligned} \mathcal{D} &= \{\mathcal{R}, \mathcal{S}\} = \{\mathcal{R}_l, \mathcal{R}_u, \mathcal{S}\} \qquad \text{(all training data)} , \\ \mathcal{L} &= \{\mathcal{R}_l, \mathcal{S}\} \qquad\qquad\qquad\quad \text{(labelled training data)} . \end{aligned} \tag{5.1}$$

A small potion of $\mathcal{R}$ is labelled, where the labelled and the remaining unlabelled parts are defined as $\mathcal{R}_l$ and $\mathcal{R}_u$ respectively. All synthetic data $\mathcal{S}$ are labelled with ground truth. In addition, the collection of labelled data $\mathcal{D}$ is denoted by $\mathcal{L}$.

All data points in $\mathcal{D}$ are represented by vectorised local patches extracted from depth images. In this work, patches are sampled randomly from the foreground pixels in the training images $\mathcal{D}$. Each patch is 64×64 in size, which is comparable to the patch size in [Girshick et al., 2011]. The total number of training features roughly equals 5% of foreground pixels in the training images.

Every labelled data point in $\mathcal{R}_l$ or $\mathcal{S}$ is assigned to a tuple of labels $(a, p, \mathbf{j})$. Viewpoint of a patch is represented by the roll, pitch and yaw angles, which are quantised

into 3, 5 and 9 steps respectively. The view label $a \in \{1 \dots 135\}$ indicates one of the 135 quantised viewpoints. An articulated hand model consists of 16 regions, with three regions per finger and one palm region. A data point is also given the class label of its region, $p \in \{1 \dots 16\}$. Finally, every labelled data point contains 16 vote vectors $\mathbf{j} \in \mathbb{R}^{3\times 16}$ from the patch's centroid to the 3D locations of all 16 joints, similar to [Gall et al., 2011].

Realistic-synthetic associations are established through matching datapoints in $\mathcal{R}_l$ and $\mathcal{S}$, according to their 3-D joint locations. The realistic-synthetic association $\Psi : \mathcal{R}_l, \mathcal{S} \rightarrow \{1, 0\}$ is defined as below:

$$\Psi(r \in \mathcal{R}_l, s \in \mathcal{S}) = \begin{cases} 1 \text{ when } r \text{ matches } s \\ 0 \text{ otherwise} \end{cases} \tag{5.2}$$

Two patches r and s, by definition, are describing the same hand pose when their association function $\Psi(r, s)$ equals one.

5.4.2 STR Forest

The proposed STR forest performs classification, clustering and regression on both domains in one pose estimator, instead of performing each task in separate forests. We grow N_t decision trees by recursively splitting and passing the current training data to two child nodes. The feature of a node is represented by a simple two-pixel test as (**??**). A group of candidate features are generated at each node, the best one is chosen from the candidates by maximising a quality function. Instead of using a typical metric such as information gain or label variance [Criminisi and Shotton, 2013], we propose two new quality functions. The quality function is selected at random between $Q_{ap\mathbf{j}}$

and Q_{tss} for training in (5.3).

$$\begin{cases} Q_{ap\mathbf{j}} = \alpha Q_a + (1-\alpha)\beta Q_p + (1-\alpha)(1-\beta)Q_{\mathbf{j}} \\ Q_{tss} = Q_t^{\omega} Q_u \end{cases} \tag{5.3}$$

where $Q_{ap\mathbf{j}}$ is a hybrid quality function for learning classification-regression decision trees, and Q_{tss} enables transductive and semi-supervised learning. Given the training data $\mathcal{D} = \{\mathcal{R}_l, \mathcal{R}_u, \mathcal{S}\}$, the quality functions are defined as below.

5.4.2.1 Viewpoint classification term Q_a

Traditional *information gain* is used to evaluate the classification performance of all the viewpoint labels a in dataset $\mathcal{L}$ [Breiman, 2001]. Since this term is applied on the top of the hierarchy, a large amount of training samples needs to be evaluated. Inspired by [Girshick et al., 2011], reservoir sampling is employed to avoid memory restriction and speed up training. Each decision tree is trained on a subset of the training dataset $\mathcal{D}$ created by reservoir sampling, which chooses data points randomly from $\mathcal{D}$ without replacement [Vitter, 1985]. We adopt the quality function (3.1) to classify viewpoints and term it as Q_a.

5.4.2.2 Patch classification term Q_p

Similar to Q_a, Q_p is the information gain of the joint labels p in $\mathcal{L}$. It measures the performance of classifying individual patch in $\mathcal{L}$. We simply adopt (3.1) and denote it as Q_p for notation convenience.

Consequently, Q_a and Q_p optimises the decision trees by classifying $\mathcal{L}$ by their viewpoints and joint labels respectively. Whilst Q_a handles the classification of global viewpoints of the whole depth image, Q_p deals with the classification of individual local patches.

5.4.2.3 Regression term Q_{j}

This term optimises the regression aspect of the proposed STR forest, by maximising the compactness of vote vectors in a tree node. Similar to Hough forest (HF), each patch in $\mathcal{L}$ is associated with a vote, which is a 3D vector from the patch's centroid to the centre of a joint in $\mathcal{P}$. For the quality function, we simply adopt (3.3), denoted as Q_{j} for notation convenience. Q_{j} increases with compactness in vote space.

5.4.2.4 Unsupervised term Q_u

The appearances in the target domain, i.e. real data, are modelled in an unsupervised manner. Assuming appearances and poses are correlated under the same viewpoint, pose similarities between a pair of data points can be roughly estimated by the resemblance of their appearances. The unsupervised term Q_u evaluates the similarities among all real patches $\mathcal{R}$ within a node, such that

$$Q_{\mathrm{j}} = \mathrm{tr}(\mathrm{cov}(\mathcal{R})) - \sum_{k}^{\{lc,rc\}} \frac{|\mathcal{R}^k|}{|\mathcal{R}|} \left(\mathrm{tr}\left(\mathrm{cov}(\mathcal{R}^k)\right)\right). \tag{5.4}$$

The difference between (5.4) and (3.3) is, instead of minimising label variance, (5.4) minimises the variance of input data. Since the realistic dataset is sparsely labelled, i.e. $|\mathcal{R}_u| \gg |\mathcal{R}_l|$, $\mathcal{R}_u$ are essential for modeling the target distribution. In order to speed up the learning process, Q_u can be approximated by down-sampling the patches in $\mathcal{R}$.

5.4.2.5 Transductive term Q_t

The relationship between sparse real data and dense synthetic data is learned via *transductive learning*. Inspired by cross-modality boosting [Bronstein et al., 2011], the transductive term Q_t preserves the cross-domain associations Ψ as the training data pass down the trees:

$$\begin{aligned} Q_t &= \frac{|\{r,s\} \subset \mathcal{L}^{lc}| + |\{r,s\} \subset \mathcal{L}^{rc}|}{|\{r,s\} \subset \mathcal{L}|} \\ \forall\, \{r,s\} \subset \mathcal{L} &\text{ where } \Psi(r,s) = 1 \end{aligned} \tag{5.5}$$

The transductive term Q_t is the ratio of preserved association. It gives a maximum value of one when all the linkages are kept after a split.

5.4.2.6 Adaptive switching $\{\alpha, \beta, \gamma, \omega\}$

Parameters $\{\alpha, \beta, \omega\}$ are the weightings of quality terms within the chosen quality function. The STR forest adopts a three-phase learning strategy, such that the node learning objectives shifts adaptively from viewpoint classification to hand part classification to joint regression. At the early stage of testing, coarse viewpoint classification is preferred to finer hand part classification and regression. After the global viewpoint labels have been classified successfully, learning focus switches from viewpoint classification to hand part classification. Once the joint labels have been classified, learning objective then switches to joint regression where input data are clustered with respect to their poses. Fig. 5.2 illustrates the coarse-to-fine structure of trees in the proposed approach, a tree performs mainly classifications at the top levels, its behaviour changes adaptively to hand part classification at the middle levels. Finally, joint regression gives more accurate pose estimation at the bottom levels.

To control the learning phases, margin function $\Delta(\cdot)$ is defined to measure classification performance of a tree node:

$$\Delta_{\mathcal{A}}(\mathcal{L}) = \max_{a \in \mathcal{A}} p(a|\mathcal{L}) - \max_{\hat{a} \in \mathcal{A}, \hat{a} \neq a} p(\hat{a}|\mathcal{L}), \tag{5.6}$$

$$\Delta_{\mathcal{P}}(\mathcal{L}) = \max_{p \in \mathcal{P}} p(p|\mathcal{L}) - \max_{\hat{p} \in \mathcal{P}, \hat{p} \neq p} p(\hat{p}|\mathcal{L}). \tag{5.7}$$

The margin function $\Delta(\cdot)$ indicates the purity of a node, which is the difference between the most probable class and the second most probable class in a node. When a node contains data from the same class, its margin function returns zero. On the contrary, if a node contains evenly distributed class labels, its margin function tends to one. (5.6) and (5.7) are responsible for the classification of viewpoint labels a and joint labels p in $\mathcal{L}$ respectively. They measure the purity of a node with respect to viewpoint and patch label:

$$\alpha = \begin{cases} 1 & \text{if } \Delta_{\mathcal{A}}(\mathcal{L}) < t_{\alpha} \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

$$\beta = \begin{cases} 1 & \text{if } \Delta_{\mathcal{P}}(\mathcal{L}) < t_{\beta} \\ 0 & \text{otherwise} \end{cases} . \tag{5.9}$$

At top levels, the viewpoint margin $\Delta_{\mathcal{A}}(\mathcal{L})$ tends to one as the viewpoint labels are evenly distributed. As a result, according to (5.3) and (5.8), the learning algorithm favours viewpoint classification, which is the information gain of viewpoint labels Q_a. As more nodes are constructed down the decision tree, the viewpoint margin $\Delta_{\mathcal{A}}(\mathcal{L})$ decreases as more viewpoint labels are classified. At the middle levels, when α becomes zero, the learning objective Q_p changes from viewpoint classification to joint label classification. Finally, when joint labels have been classified at the bottom levels, the weightings α and β are both zero. According to (5.8), the learning objective $Q_{ap\mathbf{j}}$ switches to joint regression $Q_{\mathbf{j}}$, grouping training data with respect to their poses. Two

tunable thresholds t_α and t_β are used to control switching between different learning objectives during the training process.

The tunable parameter ω controls the relative importance of transductive term Q_t to unsupervised term Q_u.

Algorithm 3 Transductive training with DF

Require: Training dataset $\mathcal{D} = \{\mathcal{R}_l, \mathcal{R}_u, \mathcal{S}\}$; the maximum tree depth D.
Ensure: A classification-regression tree t trained on both domains that can test on realistic domain.

procedure GROW($\mathcal{D}$)
  Let $d = 0$.
  Let $\alpha = 1, \beta = 0$. ▷ Start with viewpoint classification
  SPLIT($\mathcal{D}, d$)
end procedure

function SPLIT($\mathcal{D}$, d, α, β)
  Randomly propose a set of features Φ.
  Randomly select $Q_{ap\mathbf{j}}$ or Q_{tss} in (5.3).
  for all $\phi \in \Phi$ **do**
    Partition $\mathcal{D}$ into $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$ by ϕ.
    Use selected quality function to score current feature ϕ.
  end for
  Select the optimal feature $\phi*$ by scores.
  if $d < D$ **then**
    Add a *leaf node* with a set of Hough votes $\{\mathbf{j}\}$ into t.
  else
    Partition $\mathcal{D}$ into $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$ by $\phi*$.
    Add a *split node* with $\phi*$ into t.
    Update α and β with current partition. ▷ Decide whether switch or not.
    SPLIT($\mathcal{D}^{lc}$, $d+1$, α, β)
    SPLIT($\mathcal{D}^{rc}$, $d+1$, α, β)
  end if
  Return
end function

5.4.3 Data-driven kinematic joint refinement

Since the proposed STR forest considers joints as independent detection targets, structural information is essential to recover poorly detected joints when they are occluded or missing from the depth image. It is also necessary to discard or correct anatomically impossible pose hypotheses. As a result, the proposed framework employs a data-driven, kinematic-based method to refine joint locations, without having an explicit hand model as in many generative tracking-based methods. In order to obtain the maximum pose coverage, a large hand pose database $\mathcal{K}$ is generated from a synthetic articulated model, such that $|\mathcal{K}| \gg |\mathcal{S}|$. The pose database $\mathcal{K}$ is generated using the same hand model as in the synthetic dataset $\mathcal{S}$. Different from $\mathcal{S}$, $\mathcal{K}$ only contains the joint coordinates generated from the articulated model, while the corresponding depth images are not rendered. Thus it is view point invariant and we can afford to generate many poses. Training data $\mathcal{K}$ is first categorised with respect to their viewpoints,

$$\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_i, \ldots, \mathcal{K}_{|\mathcal{A}|}\}. \tag{5.10}$$

For each $\mathcal{K}_i$, a N-component, axis-aligned Gaussian mixture model $\mathcal{G}$, is used to describe the viewpoint-specific spatial distributions of 3D joint locations:

$$\begin{aligned} \mathcal{G} &= \{\mathcal{G}_1, \ldots, \mathcal{G}_i, \ldots, \mathcal{G}_{|\mathcal{A}|}\}, \\ \mathcal{G}_{\mathbf{i}} &= \{\mu_{\mathbf{i}}^1 \ldots \mu_{\mathbf{i}}^n \ldots \mu_{\mathbf{i}}^N; \mathrm{cov}_{\mathbf{i}}^1 \cdots \mathrm{cov}_{\mathbf{i}}^n \cdots \mathrm{cov}_{\mathbf{i}}^N\}. \end{aligned} \tag{5.11}$$

The n-th mean and variance in the $\mathbf{i}$-th viewpoint are denoted by $\mu_{\mathbf{i}}^n$ and $\mathrm{cov}_{\mathbf{i}}^n$ respectively. In Section 5.6, a mixture model of 50 Gaussian distributions is constructed for joint refinement, i.e. $N = 50$. The procedures for computing the data-driven kinematic model $\mathcal{G}$ is summarised in algorithm Algorithm 4.

Algorithm 4 Data-driven Kinematic Models.

Require: A joint dataset $\mathcal{K} \subset \mathbb{R}^{3\times 16}$ that contains synthetic joint locations, where $|\mathcal{K}| \gg |\mathcal{S}|$.

Ensure: A set of viewpoint-dependent distributions $\mathcal{G} = \{\mathcal{G}_{\mathbf{i}} | \forall \mathbf{i} \in \mathcal{A}\}$ of global poses.

1: Split $\mathcal{K}$ with respect to viewpoint label $\mathcal{A}$, such that $\mathcal{K} = \{\mathcal{K}_1 \dots \mathcal{K}_{|\mathcal{A}|}\}$

2:

3: **for all** $\mathbf{i} \in \mathcal{A}$ **do**

4: Learn a N-part GMM $\mathcal{G}_i$ of the dataset $\mathcal{K}_i$:

5: $\mathcal{G}_{\mathbf{i}} = \{\mu_{\mathbf{i}}^1 \dots \mu_{\mathbf{i}}^n \dots \mu_{\mathbf{i}}^N; \mathrm{cov}_{\mathbf{i}}^1 \cdots \mathrm{cov}_{\mathbf{i}}^n \cdots \mathrm{cov}_{\mathbf{i}}^N\}$, where $\mu_{\mathbf{i}}^n$ and $\mathrm{cov}_{\mathbf{i}}^n$ denote the mean and *diagonal* variance of the n-th Gaussian component in $\mathcal{G}_{\mathbf{i}}$ of viewpoint $\mathbf{i}$.

6: **end for**

5.5 Testing

5.5.1 Hand part classification and detection

Patches are extracted densely given a testing depth image. They pass down the STR forest to obtain their viewpoint $\hat{a}$ and vote vectors $\hat{\mathbf{j}}$. Similar to HF [Gall et al., 2011], the patches vote for all 16 joint locations according to $\hat{\mathbf{j}}$.

The objective of kinematic joint refinement is to compute the final joint locations $\mathbf{y} = \{y_1 \dots y_j \dots y_{16} | \forall y \in \mathbb{R}^3\}$ given an input depth image. In order to reject the outlying votes received, the meanshift technique in [Girshick et al., 2011] is applied. The set of votes received by the j-th joint is fitted to a 2-part GMM $\hat{\mathcal{G}}_j$.

$$\hat{\mathcal{G}}_j = \left\{ \hat{\mu}_j^1, \hat{\mathrm{cov}}_j^1, \hat{\rho}_j^1, \hat{\mu}_j^2, \hat{\mathrm{cov}}_j^2, \hat{\rho}_j^2 \right\}, \tag{5.12}$$

where $\hat{\mu}$, $\hat{\mathrm{cov}}$, $\hat{\rho}$ denote the mean, variance and weight of the Gaussian components respectively. Fig. 5.3 depicts the two Gaussian components obtained from fitting the voting vectors of a joint.

The true joint usually forms one compact cluster of votes, which leads to a high weighting and low variance in one of the Gaussians. On the contrary, a weak detection

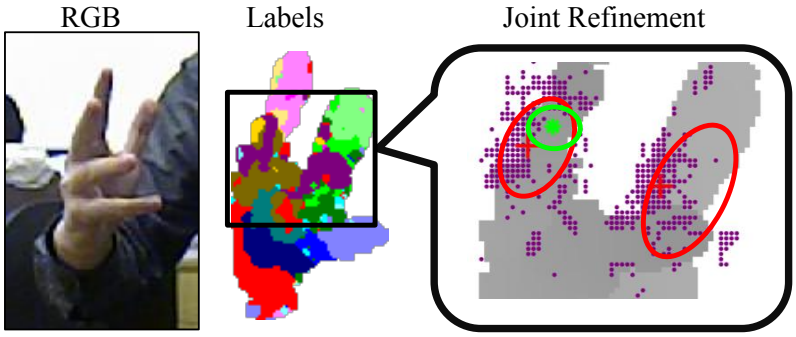


Figure 5.3: Pseudo-kinematic joint refinement algorithm. Two Gaussian distributions are fitted to the votes from the STR forest. Joint refinement is performed by combining the Gaussians between the STR forest (red Gaussian on the left) and the data-driven kinematic model (green Gaussian).

usually contains scattered votes, indicated by separated means with similar weights. A y_j is considered as a high-confidence joint when the Euclidean distance between $\hat{\mu}_j^1$ and $\hat{\mu}_j^2$ is smaller than a predefined threshold t_q. The output joint location y_j from the STR forest is computed as

$$y_j = \begin{cases} \hat{\mu}_j^1 & \text{if } ||\hat{\mu}_j^1 - \hat{\mu}_j^2||_2 < t_q \text{ and } \hat{\rho}_j^1 \geq \hat{\rho}_j^2, \\ \hat{\mu}_j^2 & \text{if } ||\hat{\mu}_j^1 - \hat{\mu}_j^2||_2 < t_q \text{ and } \hat{\rho}_j^1 < \hat{\rho}_j^2, \\ \text{undefined } ^{\dagger} & \text{otherwise .} \end{cases} \tag{5.13}$$

†: Estimated by kinematic joint refinement algorithm

For a high-confident j-th joint, the final output location y_j is represented by the mean of the dominating Gaussian in $\hat{\mathcal{G}}_j$. Otherwise, low-confidence joints, i.e. undefined joints in (5.13), are computed using the kinematic joint refinement algorithm in Section 5.5.2.

5.5.2 Kinematic joint refinement

Whilst locations of high-confidence joints are finalised from the regression forest, the joint refinement process is performed to estimate the remaining low-confidence joints. The high-confidence joints are matched to the means $\{\mu_{\hat{a}}^1 \dots \mu_{\hat{a}}^N\}$ in the kinematic model $\mathcal{G}_{\hat{a}}$, where their nearest neighbours are found by least squares fitting, with a direct similarity transformation $\mathbf{T}$. Only the high-confident joint locations are used in the above nearest neighbour matching, and the low-confident joint locations are masked out. Given the nearest Gaussian $\{\mu_{\hat{a}}^{nn}, \text{cov}_{\hat{a}}^{nn}\}$ of the high-confidence joints, one of the two Gaussians in Section 5.5.1 is selected:

$$\{\tilde{\mu}, \tilde{\text{cov}}\} = \underset{\{\mu,\text{cov}\}\in\{\hat{\mu}_j^1,\hat{\text{cov}}_j^1\},\{\hat{\mu}_2\,\hat{\text{cov}}_2\}}{\text{argmin}} \left|\left|\mathbf{T}\mu - \mu_{\hat{a}}^{nn}[j]\right|\right|_2^2. \tag{5.14}$$

The distribution $\{\tilde{\mu}, \tilde{\text{cov}}\}$ is one of the Gaussian components in $\hat{\mathcal{G}}_j$ that is closer to the corresponding j-th joint location in $\{\mu_{\hat{a}}^{nn}[j] \in \mathbb{R}^3, \text{cov}_{\hat{a}}^{nn}[j] \in \mathbb{R}^{3\times3}\}$ taken from $\{\mu_{\hat{a}}^{nn}, \text{cov}_{\hat{a}}^{nn}\}$. However, when a joint is fully occluded, its detection result in (5.14) become unreliable because the regression forest does not consider complete occlusion. As a result, the final output of a low-confidence joint y_l is recovered by merging the Gaussians in equation (5.15):

$$y_j = \left(\tilde{\text{cov}}^{-1} + (\text{cov}_{\hat{a}}^{nn}[j])^{-1}\right)^{-1}\left(\tilde{\text{cov}}\mu_{\hat{a}}^{nn}[j] + \text{cov}_{\hat{a}}^{nn}[j]\tilde{\mu}\right). \tag{5.15}$$

Fig. 5.3 illustrates the process of refining a low-confidence joint. The middle proximal joint is occluded by the index finger as seen in the RGB image; the Gaussians components in $\hat{\mathcal{G}}_j$ is represented by the red crosses (mean) and ellipses (variance). The final output is computed by merging the nearest neighbour obtained from $\mathcal{G}$, i.e. $\{\mu_{\hat{a}}^{nn}[j], \text{cov}_{\hat{a}}^{nn}[j]\}$ (the green Gaussian), and the closer Gaussian in $\hat{\mathcal{G}}_j$ (the left red Gaussian). The procedures of refining output poses $\mathbf{y}$ are stated in Algorithm 5.

Algorithm 5 Joint detection and pose refinement.

Require: Vote vectors obtained from passing down the testing image to the STR forest.
Ensure: The output pose $\mathbf{y} : \mathbb{R}^{3\times 16}$.

```
Extract patches $\hat{\mathcal{X}}$ from depth image $I$.
for all $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$ do
    Compute the viewpoint $\hat{a}$ and joint $\hat{p}$.
    Each $\hat{\mathbf{x}}$ votes at the leaf nodes.
end for
for all Set of voting vectors for the $j$-th joint do
    Learn a 2-part GMM $\hat{\mathcal{G}}_j$ of the voting vectors.
    if $||\hat{\mu}_j^1 - \hat{\mu}_j^2||_2^2 < t_q$ then
        The $j$-th joint is a high-confidence joint.
        Compute the $j$-th joint location with (5.13).
    else
        The $j$-th joint is a low-confidence joint.
    end if
end for
Find the nearest neighbour $\{\mu_{\hat{a}}^{nn}, \mathrm{cov}_{\hat{a}}^{nn}\}$ by matching the high-confidence joints with $\mathcal{G}_{\hat{a}}$.
Update the remaining low-confidence joint with (5.14) and (5.15).
```

5.6 Experiments

5.6.1 Evaluation dataset

Synthetic training data $\mathcal{S}$ was rendered using an articulated hand model as shown in Fig. 5.4. Instead of adjusting individual joints, each finger was controlled by a bending parameter, such that only realistic articulations, i.e. poses that can be performed by a real hand, were recorded in the training dataset. In order to handle the shape variations in real environments, when generating the depth images in dataset $\mathcal{S}$, for each synthetic image, we randomly adjust the finger length by $(-5\,\mathrm{mm}, +5\,\mathrm{mm})$ and palm width/height $(-1\,\mathrm{cm}, +1\,\mathrm{cm})$ in Poser Pro. A more ideal way is to learn the shape range from real hands as in [Khamis et al., 2015]. Subsequently, a synthetic dataset $\mathcal{S}$ was generated by projecting the articulated model, of different poses, to depth im-

ages. In this work, $\mathcal{S}$ contained 2500 depth images per viewpoint, the size of $\mathcal{S}$ was $2500 \times 3 \times 5 \times 9 = 337.5K$.

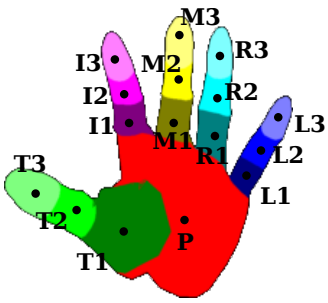


Figure 5.4: Colour codes and labels of the 16 hand regions. The joint labels are used in Fig. 5.5. The colour codes are used to visualise the hand part classification results in Fig. 5.8.

Meanwhile, real data $\mathcal{R}$ was captured using an Asus Xtion depth sensor. There were about 600 frames per viewpoint, thus the size of $\mathcal{R}$ is $81K$. Within $\mathcal{R}$, about 20% of the frames in $\mathcal{R}$ are labelled. The number of labelled sample $|\mathcal{R}_l|$ is around $10K$. Since labels can be reused for the in-plane rotationally symmetric images (same yaw and pitch, but different roll), only around $1.2K$ of data are actually hand-labelled.

Visible joints in $\mathcal{R}_l$ were annotated manually using 3D coordinates. However, annotation of occluded joints were labelled using the (x, y) coordinates initially. Real-synthetic associations Ψ and the remaining z-coordinates in $\mathcal{R}_l$ were computed simultaneously by matching visible joint locations with $\mathcal{S}$, using least squares and a direct similarity transform, similar to kinematic joint refinement in Section 5.5.2. Consequently, each data point in $\mathcal{R}_l$ was paired with its closest match $\mathbf{x}_{syn} \in \mathcal{S}$, and its occluded z coordinates were approximated by the corresponding z coordinates of $\mathbf{x}_{syn}$. A joint was modelled as a 3D truncated Gaussian distribution centred at the joint location, while its variance was defined empirically according to the structure of human hands. Foreground pixels were clustered into one of these distributions and therefore assigned with labels p. For example, the palm label usually contributed a larger region

than the fingertip labels, due to the large variance in the palm joint.

Three different sequences (sequence A,B and C) were recorded and labelled needs 50, 1000 and 240 frames respectively. Sequence A was captured with a static viewpoint for the single-view experiment. Sequence B features various viewpoint variations and sequence C contains abrupt changes in both viewpoint and scale. In the experiments, 3 trees were trained with maximum depth varying from 16 (single-view experiment) to 24 (multi-view experiment), depending on the amount of training data.

5.6.2 Single view experiment

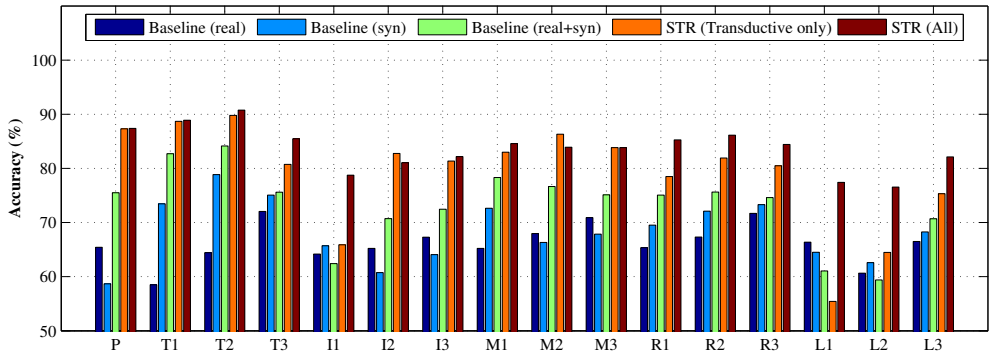


Figure 5.5: Hand part classification accuracy of the single view sequence. Classification accuracy is defined as the percentage of correct foreground pixels. In the "Transductive only" STR forest, the unsupervised term Q_u is fixed and always equals one, therefore unlabelled data are not used in training the "Transductive only" forest. Data-driven joint refinement is not performed in the baseline and the STR forest approaches. Please refer to Fig. 5.4 for code names of joints.

The proposed approach was evaluated under a single view scenario, comparing with the traditional regression forest algorithm of [Gall et al., 2011] as a baseline. Since there was only one viewpoint in testing sequence A, Q_a in (5.3) did not affect the experimental results. In other words, the proposed method and the baseline algorithm only differed in joint regression and classification, hence only Q_p,Q_j, Q_t and Q_u were

actually used in this experiment.

Performances of candidate algorithms were measured by their pixel-wise classification accuracy per joint [Shotton et al., 2011]. Fig. 5.5 shows the classification accuracy plots of the algorithm evaluated in this experiment. It demonstrates the strengths of real-synthetic fusion and semi-supervised learning. Accuracy of baseline method was improved by simply including both domains in training without any algorithmic changes.

The transductive learning term Q_t improved the accuracy substantially, particularly for the finger tips which are less robust in the baseline algorithms. By coupling real data with synthetic data, the transductive term Q_t effectively learns the discrepancies between the domains. From Fig. 5.5, some joints are often mislabelled as other more dominating joints after transductive learning, e.g. joints in little finger tip (L3) and index proximal (I1). Nevertheless, semi-supervised learning significantly corrects those joints after transductive learning. To summarise, semi-supervised learning complements transductive learning by modelling the intermediate poses between the sparsely labelled real data.

5.6.3 Multiple view experiment

In the multi-view experiment, the proposed approach was compared with the state-of-the-art by [Oikonomidis et al., 2011a] (FORTH algorithm, a tracking-based generative model) under two challenging multi-view scenarios. Quantitative and qualitative evaluations were performed to provide a comprehensive comparison of the methods.

Hand articulations were estimated from the multi-view testing sequences, i.e. sequence B and C, by both of the methods. Sequence B contains long and continuous pose and viewpoint changes, while abrupt viewpoint changes and strong occlusions

are observed in sequence *C*. Since FORTH requires manual initialisation, the testing sequences are designed such that they start with a fixed initialisation pose and position, in order to facilitate a fair comparison. In this experiment, pose estimation accuracy was measured by joint localisation error [Oikonomidis et al., 2011a].

5.6.3.1 Quantitative results

Fig. 5.6 and Fig. 5.7 show the average localisation errors of testing sequence *B* and *C* respectively. It also shows the same error graphs from a stable joint (palm, *P*) and a difficult joint (index finger tip, *I*3). The proposed STR forest, together with the data-driven kinematic joint refinement, outperform FORTH in all aspects, especially for the finger tip joints that are noisy and frequently occluded. Since the proposed approach is frame-based, even though a few large estimation errors are observed, they can be recovered quickly without drifting.

Experiments with sequence *C* justify the advantages of the proposed system over tracking-based methods. In the first 200 frames, STR forest with joint refinement perform just slightly better than FORTH. However, localisation errors in FORTH accumulate after an abrupt change in the testing sequence, and the incorrect pose does not recover. Since tracking-based approaches rely on previous results to optimise the current hypothesis iteratively, estimation errors amass and drifting issue worsens over time. On the contrary, frame-based discriminative approaches consider each frame as an independent input, enabling fast error recovery at the expense of a smooth and continuous output, hence small jitters are observed from the hand pose estimation results.

In this experiment, the proposed joint refinement scheme improves the joint estimation accuracy in general, as illustrated in Fig. 5.6 and Fig. 5.7. Some large classification errors, e.g. Fig. 5.6b, are corrected after applying joint refinement. It implies that the joint refinement process not only improves the accuracy of joint, but also prevents

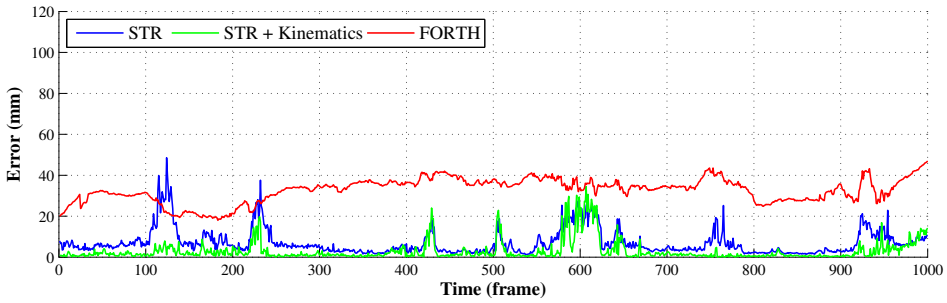
incorrect detections by validating the output of STR forest with kinematic constraints.

5.6.3.2 Qualitative analysis

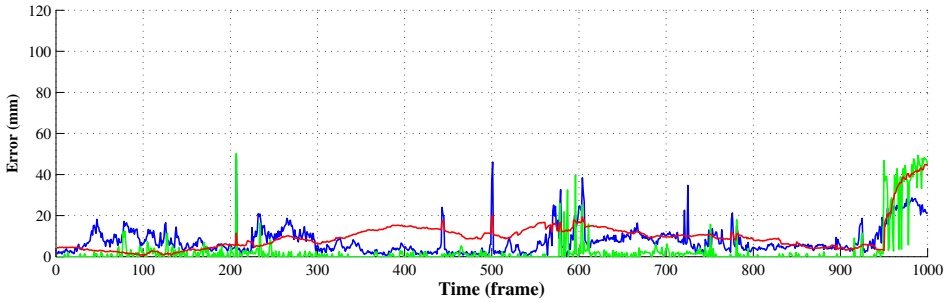
The proposed pose estimation system was also evaluated qualitatively, Fig. 5.8 displays some of the experimental results extracted from testing sequence *B* and *C*. In Fig. 5.8, hands are cropped from the original images for better visualisation (135×135 pixels for sequence *B*, 165×165 pixels for sequence *C*). During the experiment, the size of the original images (RGB and depth) is 640×480 pixels. Fig. 5.8(a) and (b) show the pose estimation results of the same articulation from different view points. Fig. 5.8(d) shows a frame at the beginning of test sequence *B*, both approaches obtain accurate hand articulations initially. However, the performance of FORTH declined rapidly in the middle of the sequence when its tracking algorithm lost target and failed to recognise a correct pose in Fig. 5.8(e), yet the STR forest approach still gives correct results. In addition, strong occlusions are shown in Fig. 5.8(f), where FORTH did not work well but the STR forest gave the correct pose. In Fig. 5.8, however, both approaches did not produce very accurate results due to strong motion blur, which is indicated by a large blank region in the corresponding depth image.

In Fig. 5.9 more qualitative images from 2 other subjects have been shown, including success and failure cases. These are screenshots from 2 unlabeled video sequences, which can be found in our video demo.

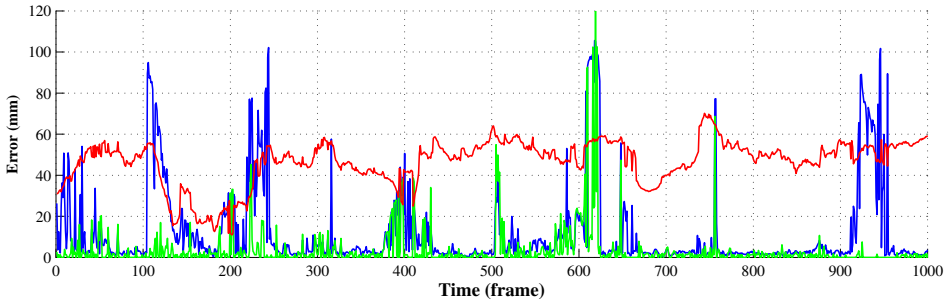
With respect to run-time performance, the proposed STR forest estimated one hand pose at about 25 FPS on an Intel I7 PC without GPU acceleration, whilst the FORTH algorithm ran at 6 FPS on the same hardware configuration plus GPU acceleration (NVidia GT 640).



(a) Test sequence B (Average error)

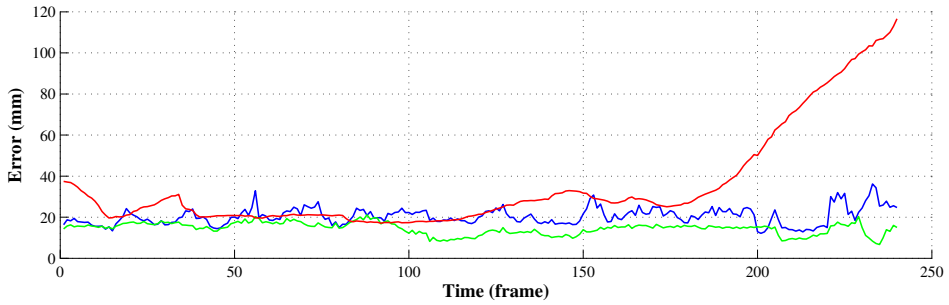


(b) Test sequence B (Palm)

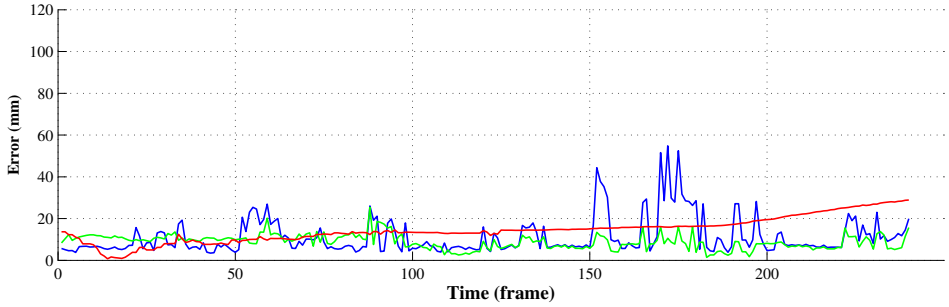


(c) Test sequence B (Index finger tip)

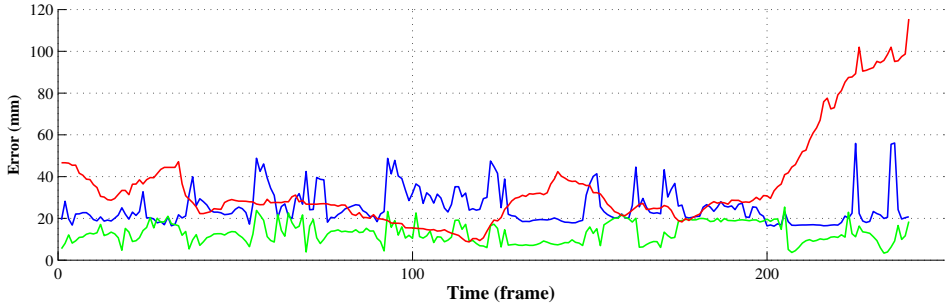
Figure 5.6: Quantitative hand pose estimation results — Sequence B. These graphs shows the 3D hand pose localisation errors (in mm) against time (in frames). Localisation error is defined as the distance between the estimated joint location and its corresponding ground truth location. In addition to the proposed STR forest and the data-driven joint refinement extension, the FORTH algorithm [Oikonomidis et al., 2011a] is also included as a reference.



(a) Test sequence *C* (Average error)



(b) Test sequence *C* (Palm)



(c) Test sequence *C* (Index finger tip)

Figure 5.7: Quantitative hand pose estimation results — Sequence *C*. Different from sequence *B*, the testing sequence *C* contains more abrupt changes to viewpoint and pose than sequence *B*.

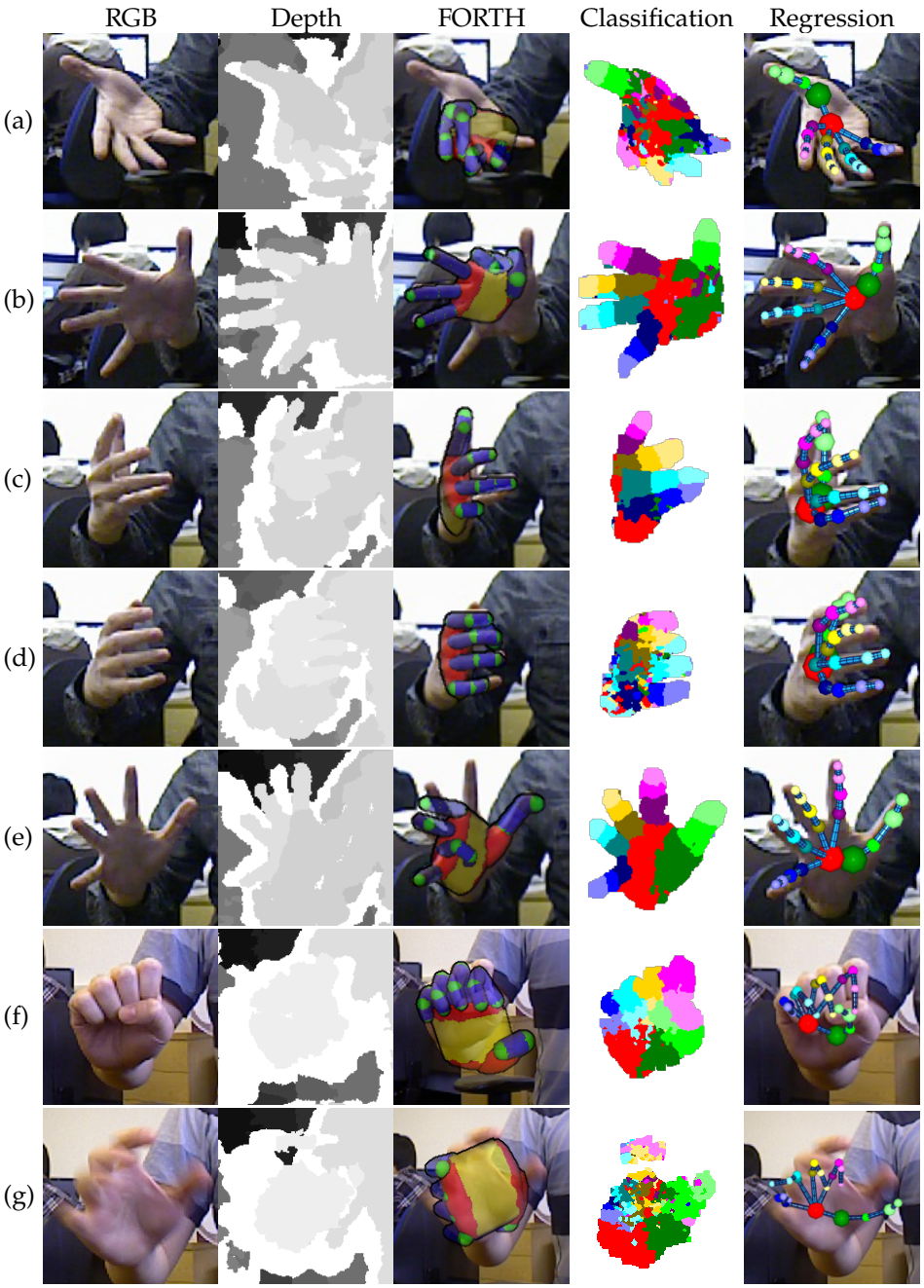
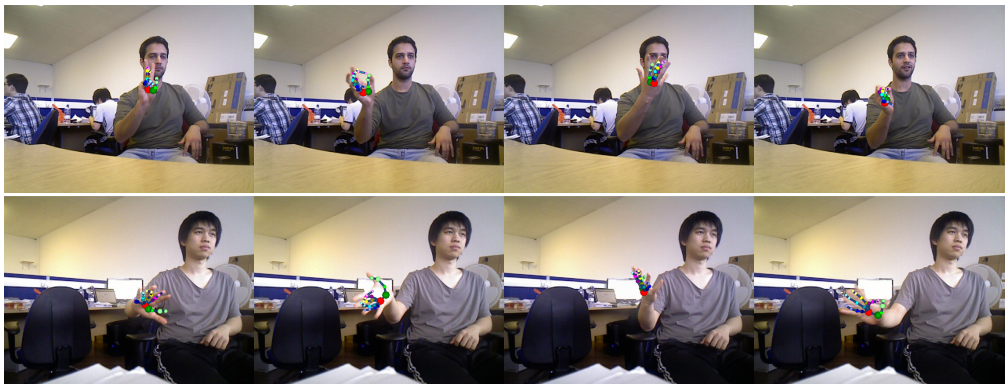


Figure 5.8: Qualitative results of the multi-view experiment. (a)-(e) are taken from sequence *B* and (f)-(g) are from sequence *C*. Hand regions are cropped from the original images for better visualisation (135 $\times$ 135 pixels for sequence *B*, 165 $\times$ 165 pixels for sequence *C*); the size of the origin images (RGB and depth) is 640 $\times$ 480 pixels. Colour scheme of joint labels refer to Fig. 5.4.

(a) Success cases



(b) Failure cases

Figure 5.9: Qualitative results of different testing subjects.

5.7 Summary

The chapter investigates the problem of pose estimation of human hand pose from depth image sequences. Despite sharing many similarities with body pose estimation, practical 3D hand pose estimation is still far from mature, primarily due to the characteristics of hand poses, in which occlusion and noise prevail. Furthermore, the discrepancies between real and synthetic data also undermine the performances of existing systems.

Dealing with the above issues, this work introduces a semi-supervised transductive approach for articulated hand pose estimation. A novel discriminative pose estimator, namely STR forest, is proposed to infer hand poses using both real and synthetic training data. With transductive learning, the STR forest recognises a wide range of poses from a small labelled real dataset by linking its data points to a large synthetic dataset. Semi-supervised learning is also applied to fully utilise the sparsely labelled real dataset. To improve the estimation accuracy of heavily occluded hand poses, a data-driven pseudo-kinematic technique is used to correct the occluded joints.

The proposed hand pose estimation system was evaluated with respect to different challenging environments. From the quantitative and qualitative analyses, it demonstrated promising results in estimating articulated hand poses from noisy and strongly occluded data. It also achieved superior accuracy and run-time performance compared with current state-of-the-art approaches.

On the other hand, there are several limitations that affect the performance of the proposed pose estimation system. For instance, the STR forest does not model the anatomical constraints of human hands, e.g. the constant distances between joints, and the limited ranges of joint angles. The data-driven joint refinement scheme (Section 5.4.3) is an ad hoc extension to rectify anatomically infeasible hand poses from

the STR forest. In addition, its similar to other appearance-based approaches, the proposed pose estimator does not recognise hand poses that are not present in the training data. Expanding the training dataset greatly increases the computational time of the learning process. Furthermore, the proposed system estimates only one hand pose at a time, which is based on the assumption that one foreground object is presented in each input depth image.

CHAPTER 6

Latent Regression Forests

A coarse-to-fine search for poses

Contents

6.1 Overview

In the previous chapter we described semi-supervised transductive regression (STR) forest, a hierarchical model which adaptively switches among viewpoint classification, hand part classification and pose regression. A few drawbacks have been discussed. Firstly, since no kinematic constraints are involved during forest prediction, an extra refinement step is needed, such as enforcing dependency between local outputs [Taylor et al., 2012] or kinematic constraints [Tang et al., 2013]. Without such procedures, highly unlikely or even impossible poses can be produced as output. Secondly, due to the learning-based nature, a significant size of training data is required, which not only makes the training very difficult, but also results in large tree sizes.

In Section 2.4 we discussed two types of discriminative methods: holistic and patch-based. Holistic methods are efficient but less flexible due to its nearest neighbour nature [Romero et al., 2009, Wang and Popović, 2009]. A patch-based method, such as STR forest can generalise to unseen samples by consider local appearance only. However, the complexity is high, because during testing each pixel need to be classified or regressed. Fig. 6.3 illustrates this difference in a forest-based manner.

On the other hand, state-of-the-art patch-based methods can often run in real-time. The reasons behind are three-fold: (1) Depth data provides a clean segmentation, leaving only foreground pixels - and yet there are still thousands. (2) Depth data provides scale information. Thus no need to construct a scale pyramid as with RGB data. (3) Feature function (3.12) is efficient.

Apparently, these reasons have not yet considered the aforementioned differences between holistic and patch-based methods. Which means there is room to further improve in terms of efficiency. And that is what we aim to achieve in this part.

In this chapter, we show that the hand pose estimation problem can be decomposed

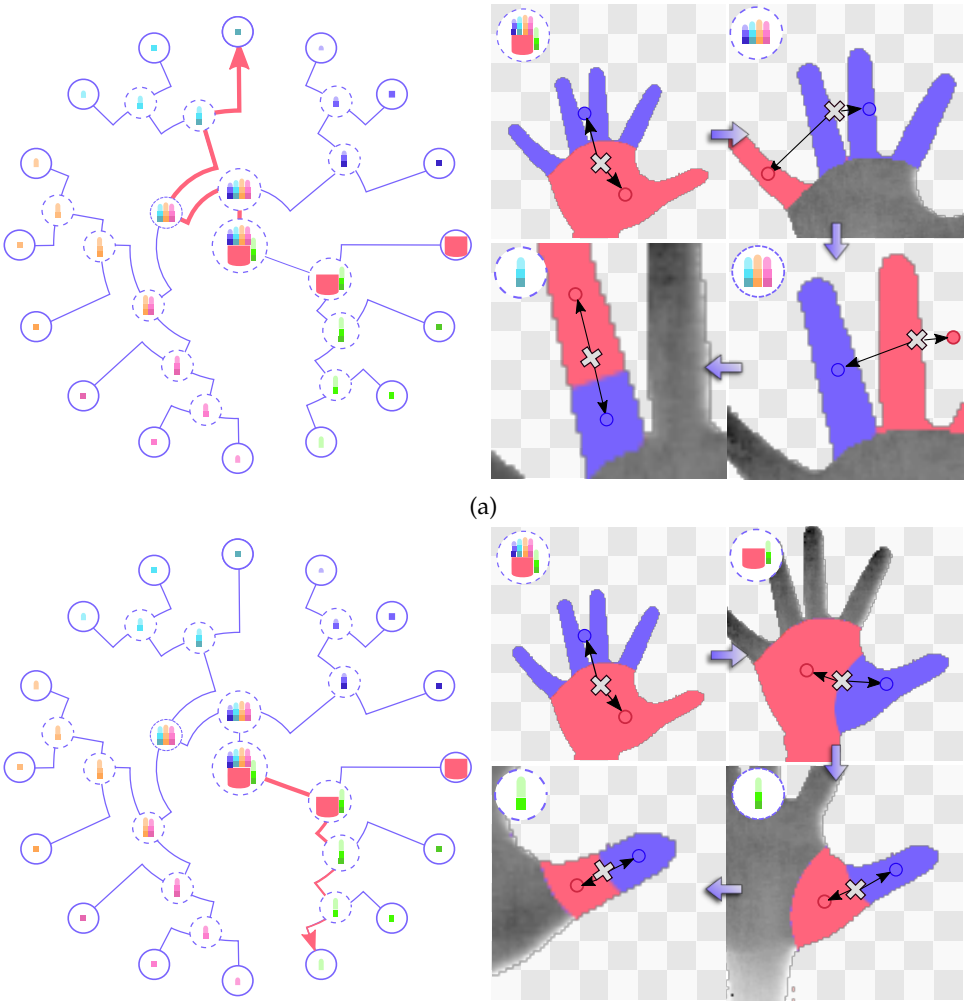
into estimating the location of a discrete set of parts on the hand skeleton model. To this end, we propose latent regression forest (LRF) to formulate the problem as a dichotomous divide-and-conquer search for skeletal parts, guided by a learnt topological model of the hand. For the topological model, we choose to use latent tree model (LTM) as its tree structure is similar to the hand topology. As illustrated in Fig. 6.1, each latent node in the LTM (left) corresponds to one forest which predicts the centres of two sub-regions (right). Furthermore, the topological model is used to enforce implicitly learnt global kinematic constraints on the output. Additionally, by training in a discriminative manner using our new diverse hand pose dataset, our approach is able to generalise to hands of various shapes and sizes as demonstrated in our experiments. Our experiments show that the LRF outperforms state-of-the-art methods in both accuracy and efficiency. The main contributions of our work can be summarised as follows:

1. **Unsupervised learning of the hand topology.** We represent the topology of the hand by an LTM [Choi et al., 2011] which is learnt in an unsupervised fashion. This topological model is used while training the LRF to enable a structured coarse-to-fine approach.

2. **Latent regression forest.** We introduce a framework for structured coarse-to-fine search in depth images. Guided by the learnt LTM, we learn binary decision trees that iteratively divide the input image into sub-regions until each sub-region corresponds to a single skeletal part. Furthermore, an error regressor is embedded into each stage of the framework, in order to avoid error accumulation.

3. **A new multi-view hand pose dataset.** We present a new hand pose dataset containing 180K fully 3D annotated depth images from 10 different subjects (see Section 4.1.2).

The following sections are organised in this way. In Section 6.4 we discuss how we can learn the hand topology in an unsupervised fashion. Following this, in Section 6.5, we discuss how this topology is used to build an LRF to perform a structured, coarse-to-fine search in the image space.



(a)

(b)

Figure 6.1: The testing procedure of LRF. It can be viewed as a search process, guided by a binary LTM; starting from root of the LTM, we minimise the offset to its children at each level until reaching a leaf node which corresponds to a skeletal part position. An example of our LTM is visualised as a circular tree on the left column. For visualization simplicity, we only show the searching process for locating (a) MCP of ring finger, (b) MCP of thumb. Each search path in the left corresponds to the searching process on the right. In practice, all parts are located in one go. White cross: centre position of current latent node. Dotted circle: latent node. Solid circle: observable node.

6.2 Related Work

6.2.1 Visual Search

For object detection, in terms of extracting feature candidates, the top performing methods are all based on sliding windows [Viola and Jones, 2004, Torralba et al., 2007,Gall and Lempitsky, 2009,Dalal and Triggs, 2005,Chum and Zisserman, 2007,Dollár et al., 2009, Felzenszwalb et al., 2010], which takes the form of cropping different sizes of sub-images at different positions. However, this is a very computationally expensive process and in fact is the bottleneck of these frameworks. According to [Dollár et al., 2009,Dollár and Perona, 2010,Felzenszwalb et al., 2010], generating gradient histograms alone over a finely sampled pyramid takes about 1 second per image (size 640×480). One clue to speed up this process is via Visual Search, which can help restrict the scanning space down to a relatively small attention area.

Visual search (visual saliency or attention) has long been studied with biological motivations in the cognitive robotic field [L. Itti, 2001, Peters and Itti, 2007, Gelenbe, 2010, Begum and Karray, 2011]. It tackles attentional selection of scene regions worthy of further analysis by higher-level processes. While earlier works have addressed bottom-top saliency e.g. Itti-Koch model, which predicts interesting locations based on low-level visual features, recent works have explored a way to incorporate both bottom-up and top-down saliency [Peters and Itti, 2007]. The top-down model accounts task-dependent influences. In the literature of computer vision, works have been more computationally motivated. The traditional yet very standard sliding-window approach has been accelerated by a branch-and-bound parameter selection scheme [Lampert et al., 2008,Sznitman and Jedynak, 2010]. Method to control an active PTZ camera by maximising scene information gain [Sommerlade and Reid, 2008] is another relevant work. However, all these works are task-independent, i.e. missing top-down models.

In [Vijayanarasimhan and Kapoor, 2010], the object categorisation method guides the feature acquisition process in the way that maximises class separation, i.e. the top-down mechanism. Also relevant are the branch of studies about contextual information to help object detection. The hierarchy of scenes, objects and parts are modeled in [Sudderth et al., 2005, Choi et al., 2010].

6.2.2 Graphical Models

Graphical models, especially tree-based models have recently been used for estimating human body pose. A latent structured model is described by [Ionescu et al., 2011] to estimate 3D human body poses from silhouettes. For deformable object detection, [Zhu et al., 2008a] propose to combine a top-down and a bottom-up processe to learn a hierarchical model that leveraging parts. [Tian et al., 2012] build a hierarchical tree models to examine spatial relationships between body parts. Whereas [Wang and Li, 2013] use a latent tree model (LTM) to approximate the joint distributions of body part locations.

LTMs, in particular, are an interesting type of graphical model as they are able to represent complex relationships in the data [Mourad et al., 2013] and, furthermore, recent methods for constructing these models ([Choi et al., 2011, Harmeling and Williams, 2011]) enable us to learn consistent and minimal latent tree models in a computationally efficient manner.

6.3 Baselines

We decompose the hand pose estimation problem into estimating the location of 16 skeletal parts on the hand model. More formally, we are given a depth image I of a segmented hand and trying to predict a set of 3D positions for 16 skeletal parts $\mathcal{O}$ (see Fig. 6.2 for the visualisation). To this end, a training set $\mathcal{D} = \{(I, \mathcal{O})\}$, where each training sample can be represented as a tuple $(I, \mathcal{O})$ is used to train a classifier.

In this section, we describe a holistic regression forest and a patch-based regression forest, respectively in Section 6.3.1 and Section 6.3.2. They will serve as baselines in the experiment part (see Section 6.6).

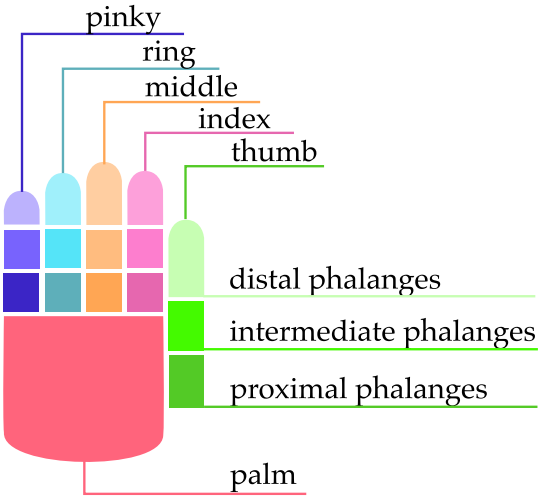
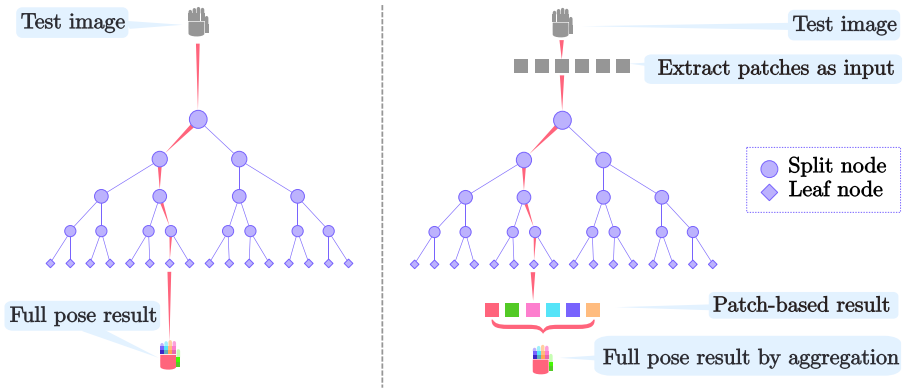


Figure 6.2: Colour coding of the 16 skeletal parts in this chapter.

6.3.1 A holistic method

We now design a simple holistic regression forest, which takes the entire hand region as input, and regresses directly on 16 hand part locations. To this end, we define the input as a tuple $(I, \mathbf{x}_0)$, where $\mathbf{x}_0$ is the centroid of segmented pointcloud, and the output as



(a) Holistic Regression (b) Patch-based Regression Tree

Figure 6.3: Comparison between holistic and patch-based regression trees. (a) a holistic regression tree takes the whole hand as input and gives one pose result; (b) a patch-based regression tree takes each patches as input and aggregates results together.

$\mathbf{j} = \| \ (\mathbf{y}_k - \mathbf{x}_0 | \mathbf{y}_k \in \mathcal{O})$, where each part location is normalised with the centroid and hence becomes an offset, and $\|$ is the concatenation operator resulting $\mathbf{j} \in \mathbb{R}^{16\times 3}$. The training data for this holistic regression forest hence becomes $\mathcal{D}_H = \{(I, \mathbf{x}_0, \mathbf{j})\}$.

During training, for each internal node, the quality function (3.3) is used to find a sub-optimal split. Note that in this case (3.3) need to minimise the variance of 48D vectors, which complexity will be reduced by our method. For the feature function, we simply adopt (3.12).

6.3.2 A patch-based method

We introduce a patch-based baseline which is based on the work of [Keskin et al., 2012]. It has a two-layer structure: first classify hand images into different shapes and then classify each pixel (or patch) into one of the hand parts. Such a structure saves the memory and can usually improve the accuracy. However, this classification-based method can not predict self-occluded parts. We then modify it by integrating regression voting

as in [Tang et al., 2013].

In this case, each input sample is defined as $(I, \mathbf{x}_x)$, where $\mathbf{x}_x$ is the location of a foreground pixel x. And the corresponding output is the 3D offset $\mathbf{j}_x = \|(\mathbf{y}_k - \mathbf{x}_x | \mathbf{y}_k \in \mathcal{O})$, which has already been defined in Section 1.1. The training data is then given as $\mathcal{D}_P = \{(I, \mathbf{x}_x, \mathbf{j}_x)\}$.

The same quality (3.3) and feature (3.12) functions can be adopted here. However, since $|\mathcal{D}_H| \ll |\mathcal{D}_P|$ due to the size of $\{x\}$, it is much more efficient to train a holistic regressor.

The results from all patches and trees are then aggregated together and a robust mean-seeking method, in our case *Meanshift* [Cheng, 1995], is applied to locate the final hand part positions. The differences between a holistic regression tree and a patch-based regression tree are illustrated in Fig. 6.3.

6.4 Learning the Hand Topology

To guide the search process, we desire to define a coarse-to-fine, hierarchical topology of the hand, where the coarsest level of the hierarchy is given by the input to the search, i.e. , the entire hand, and the finest level by the outputs, i.e. , the skeletal parts. Using a latent tree model (LTM) to represent this structure is a sensible choice. More importantly, unlike previous works that use LTM [Wang and Li, 2013], our method aims to leverage this hierarchical model to reduce the training samples and testing complexity.

Wang and Li applied LTM to represent the articulation of human body [Wang and Li, 2013]. However, their method is not an ideal choice for our problem because: 1) The default LTM they learnt contains no latent node, i.e. , all nodes represent skeletal parts. 2) They then defined 10 combined parts (or poselets) to mimic latent nodes, resulting in

an LTM that does not seem to be the most representative topology of human body. Both cases do not support our coarse-to-fine search paradigm and hence do not improve the efficiency.

In this section, we introduce a method to automatically learn an LTM that not only captures the hand topology, but also is a coarse-to-fine manner. This method requires neither prior knowledge of physical joint connections, nor predefined combined parts, which means it can be applied to any other articulated objects.

6.4.1 Model definition

An LTM is a tree-structured graphical model, $G = (\mathcal{O} \cup \mathcal{U}, \mathcal{E})$, where the vertices are composed of *observable vertices* (skeletal parts), O, and *latent vertices* (combination of parts), $\mathcal{U} = \{l\}$, where $l \subseteq O$; and $\mathcal{E}$ denotes edges (see Fig. 6.7).

Given an LTM of the hand topology, G, for each vertex $k \in G$, $k = 0...|G|$, its parent is defined by $p(k)$ and its 2 children by $lc(k)$ and $rc(k)$. For each training depth image, I, a 3D position, $\mathbf{x}_i^I$, is associated with k. For each observable vertex, $k \in \mathcal{O}$, this is simply the position of the associated skeletal part; for each latent vertex, $k \in \mathcal{E}$, the position is represented by the mean position of the observable nodes they are composed of.

6.4.1.1 Unsupervised learning

A neighbour-joining strategy called Chow-Liu neighbor-joining (CLNJ) for learning LTMs efficiently was proposed by [Choi et al., 2011]. The method starts by constructing an information distance matrix $\mathbf{D}$ of all random variables. And then a Chow-Liu tree is constructed with $\mathbf{D}$. For each internal node in the Chow-Liu tree, a recursive joining method scheme is applied by identifying its neighbourhood. This method can produce consistent LTMs without redundant latent nodes. Please refer to [Choi et al., 2011] for more details.

Recall that we are given a training set $\mathcal{D} = \{(I, \mathcal{O})\}$, where $\mathcal{O}$ which represents 16 skeletal part positions can now be related to observable vertices, such that $\mathcal{O} = \{\mathbf{x}_k | k \in O\}$. Therefore we do not assume any knowledge of physical part connections and do not define any combined parts, which our algorithm will adaptively learn.

At the beginning of the search, we have no knowledge of any hand part location but a region of interest containing a segmented pointcloud. Hence in additional to the 16 part locations, we define the holistic hand region as one more variable and the centroid of the segmented pointcloud as its location. Note that this should be considered as an observable node rather than a latent node, since its location is calculated from the pointcloud rather than other part positions. This change simply provides a 'starting point' for our algorithm.

6.4.2 Distance Function

A distance matrix, $\mathbf{D}$, of all 17 vertices (1 centroid and 16 parts) is needed for Chow-Liu neighbor-joining (CLNJ). The key part of constructing $\mathbf{D}$ is to define a distance function. In [Choi et al., 2011] the *information distance* between two random variables x and y is defined as below,

$$d_{xy} = -\log \left| \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)\,\text{var}(y)}} \right| \tag{6.1}$$

This results in a distance matrix and an LTM as shown in Fig. 6.4, which are consistent with the LTM in [Wang and Li, 2013], i.e. , no latent nodes. As mentioned before, we desire to learn a graphical model that represents a coarse-to-fine structure, where predictions of each level are not the final results but a closer approximation. Hence we explore two other different metrics that better represent the physical connections.

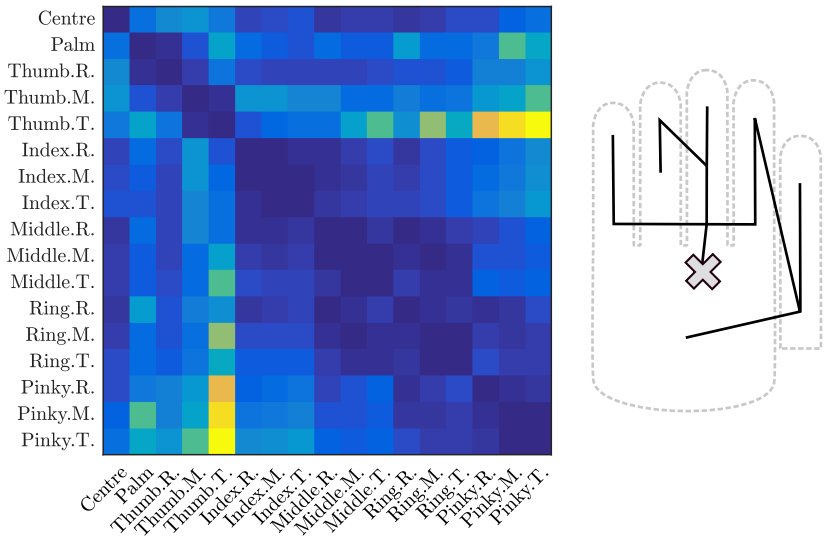


Figure 6.4: The distance matrix (left) and latent tree model (right) generated with the information distances defined in [Choi et al., 2011].The cross symbol indicates the centroid. (P: proximal phalanx, I: intermediate phalanx, D: distal phalanx, the same in other figures.)

Using the training set $\mathcal{D}$, we define the distance d between two observable vertices x and y as:

$$d_{xy} = \frac{\sum_{I \in \mathcal{D}} \delta\left(I, x, y\right)}{|\mathcal{D}|}, \tag{6.2}$$

where $\delta\left(I, x, y\right)$ is a function measuring the distance between vertices x and y in image I. An intuitive choice for δ is Euclidean distance. However, we find that the learnt LTM can not represent the anatomical structure well (see Fig. 6.7). To better exploit the training data, we propose to use an geodesic-like function for measuring pair-wise part distance.

Formally, the process of calculating the geodesic distances of a sample $(I, \mathcal{O})$ is given as followed,

1. We first construct a fully connected, undirected graph of all 17 vertices in $\mathcal{O}$, using Euclidean distance.

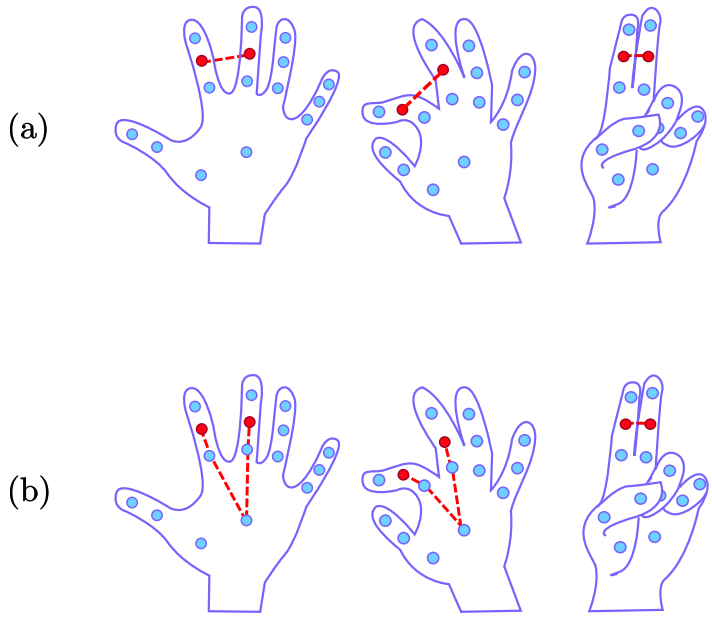
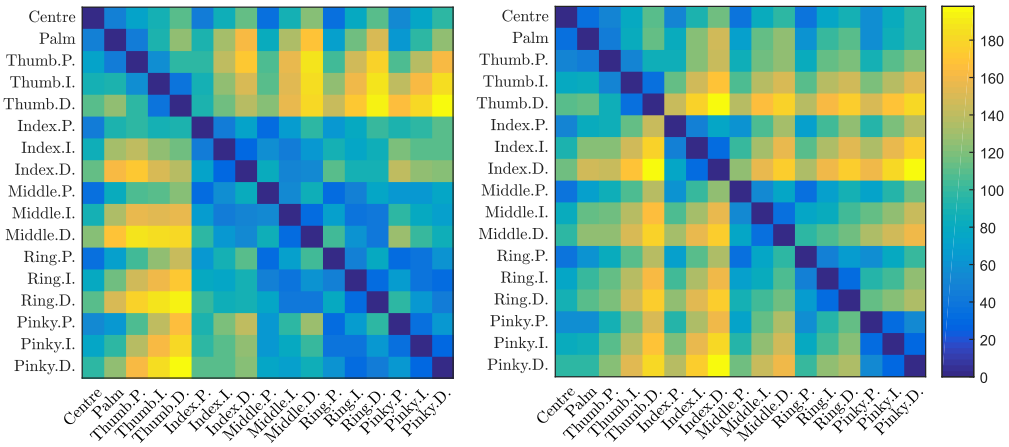


Figure 6.5: Different distance functions for CLNJ. Row (a) Euclidean distance between two parts. Row (b) Geodesic distance between two parts. The Euclidean distances between two parts are different from pose to pose. The geodesic distance, however, are consistent in most cases, as in (b) left and middle, although it is still different in the case of (b) right. Since (6.2) is averaged across all training samples, the final distance matrix with geodesic distance can still reflect the physical connection.

2. And then for each pair of vertices x, y in this graph, a projected 2D Bresenham line is drawn on I and all the depth values along this line are averaged.

3. If the averaged depth value is larger than $\max(I(x), I(y))$, the edge between x and y are then removed. This means there is a large depth discontinuity along this edge in image space, i.e. , the Bresenham line passes both foreground and background.

4. What remains is a graph in which the edges all lie along a smoothly transitioning depth path. The geodesic distance between two vertices can then be calculated as the shortest path connecting them in this graph.



(a) (b)

Figure 6.6: Comparison between using Euclidean and geodesic distance: (a) distance matrix generated with Euclidean distance, (b) distance matrix generated with geodesic distance. The distance matrix in (b) represents the physical connections better than (a).

In Fig. 6.5 we demonstrate the difference between Euclidean and geodesic distance on a toy hand model. Through different poses the Euclidean distance between two hand parts can change drastically while the geodesic one remains largely unchanged; this robustness of the geodesic distance to pose variance is preferable to capturing the topology of human hand. This is also reflected in the distance matrices shown in Fig. 6.6 . The distances between parts from different fingers are larger using geodesic distance.

In Fig. 6.7, we illustrate two LTMs generated using the Euclidean and geodesic metrics respectively. There are two interesting observations comparing to Fig. 6.4: 1) A coarse-to-fine structure is represented with latent nodes. 2) The model is a binary tree - this is relatively trivial, since it is an outcome of the distance function and our method do not have this constraint. As highlighted by the dashed lines, the Euclidean-generated model groups sub-parts of fingers with different fingers e.g. proximal phalanx of middle finger is grouped with index finger, whereas in the geodesic-generated model the fingers are all separated. This is consistent to the anatomical structure,

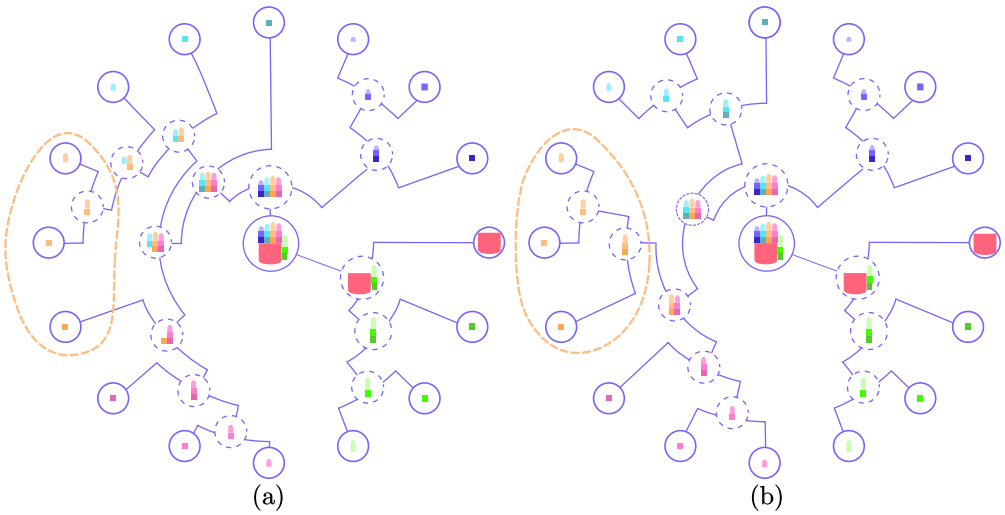


Figure 6.7: LTMs learnt with different distance functions. (a) LTM generated using the Euclidean distance metric. (b) LTM generated using the geodesic distance metric. Solid circles represent observable vertices and dashed ones latent vertices. Yellow dash lines encircle the three parts of middle finger. They are separated In the case of Euclidean distance and grouped together in the case of geodesic distance.

which we assume to have no knowledge during training. This means our method can be applied to any other articulated objects for recovering topology and pose estimation. These two different structures results in the different structure of our regressor, which is covered in Section 6.5. The impact on accuracy is empirically demonstrated in Section 6.6.

6.5 Latent Regression Forest

The aim of an LRF is to perform a search of an input image for several sub-regions, each corresponding to a particular skeletal part. Searching is preformed in a dichotomous divide-and-conquer fashion where each division is guided by the learnt LTM representing the topology of the hand.

An LRF is an ensemble of randomised binary decision trees, each trained on a boot-

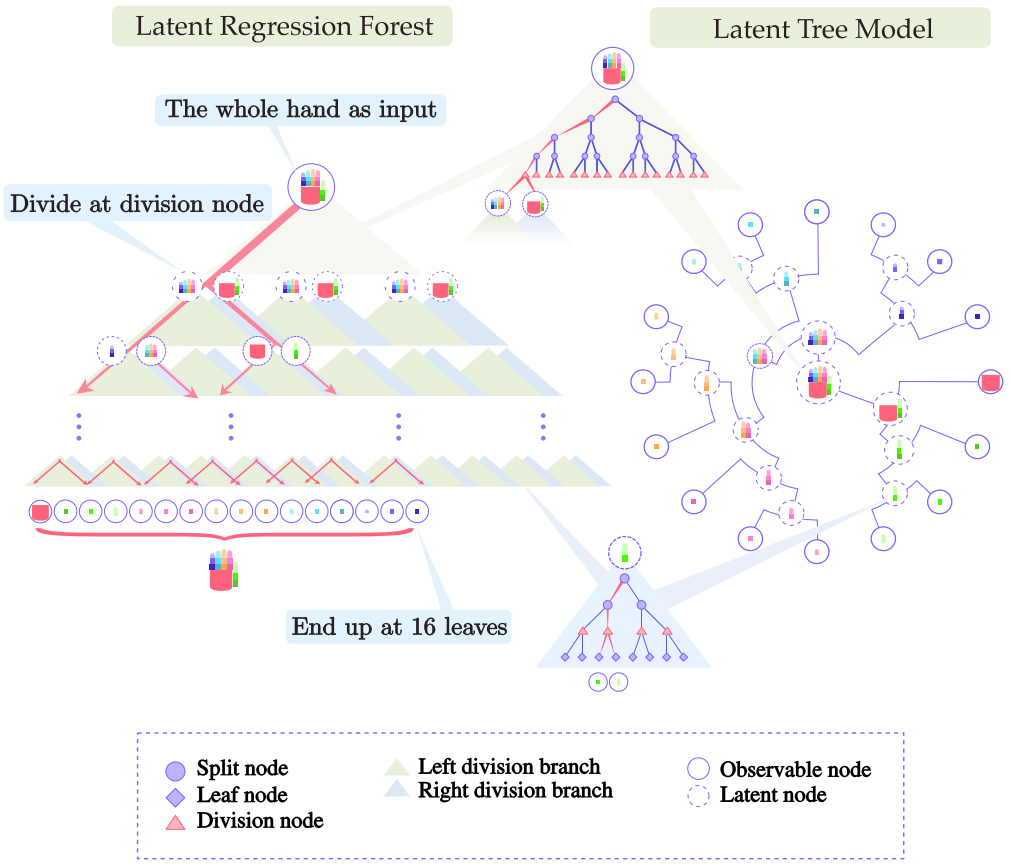


Figure 6.8: Structure of a latent regression tree. Left: an LRF; Right: an LTM. During training, we grow a regression forest for each internal node of the LTM. During testing, the whole point cloud is propagated down the tree and keep dividing until ending up at 16 leaf nodes.

strap sample of the original training data. Each latent regression tree (LRT) contains three types of nodes: *split*, *division* and *leaf* (see Fig. 6.8). Split nodes perform a test function on input data and decides to route them either left or right. Division nodes divide the current search objective into two disjoint objectives and propagate input data down both paths in parallel. Finally, leaf nodes are terminating nodes representing a single skeletal part and store votes for the location of this part in 3D space.

In Section 6.5.1 we discuss how to build the LRF followed by a discussion of the

testing procedure in Section 6.5.2.

6.5.1 Training

Given an LTM of the hand topology, G, for each vertex $k \in G$, $k = 0...|G|$, its parent is defined by $p(k)$ and its 2 children by $lc(k)$ and $rc(k)$. For each training depth image, I, a 3D position, $\mathbf{x}_k^I$, is associated with k. For each observable vertex, $k \in \mathcal{O}$, this is simply the position of the associated hand part; for each latent vertex, $k \in \mathcal{U}$, the position is represented by the mean position of its child nodes.

Each LRT in the Latent Regression Forest is trained as follows: the LRT is trained in stages, where each stage corresponds to a non-leaf vertex in the LTM, G. Starting with the root vertex, $k = 0$, of G we grow the LRT with the objective of separating the image into two cohesive sub-regions which correspond to the vertices, $lc(k)$ and $rc(k)$, which are the children of the root node.

This separation is achieved by growing a few layers of LRT. At each node, we randomly generate splitting candidates, $\Phi = \{(\phi_k, \tau_k)\}$, consisting of a function, ϕ_k, and threshold, τ_k, which splits the input data, $\mathcal{D}$, into two subsets, $\mathcal{D}^{lc}$ & $\mathcal{D}^{rc}$, s.t. $\mathcal{D}^{lc} = \{\mathbf{x} | \phi_k(\mathbf{x}) < \tau_k\}$ and $\mathcal{D}^{rc} = \mathcal{D} \setminus \mathcal{D}^{lc}$. A function, ϕ_k, for a splitting candidate, whilst at the stage represented by the LTM vertex k is defined as:

$$\phi_k(\mathbf{x}) = I\left(\mathbf{x}_k + \frac{\mathbf{u}}{I_I(\mathbf{x}_0^I)}\right) - I\left(\mathbf{x}_k + \frac{\mathbf{v}}{I(\mathbf{x}_0)}\right), \tag{6.3}$$

where $I_I(\cdot)$ is the depth at an image position, $\mathbf{x}_k$ is the position of the LTM vertex, k, in the image, I and vectors $\mathbf{u}$ and $\mathbf{v}$ are random offsets. Similarly to (3.12), the offsets are normalised to make them depth-invariant. However, in order to avoid error accumulation in depth values, the normalisation factor is always the *centre of mass*, $\frac{1}{I_I(\mathbf{x}_0)}$.

The optimal splitting candidate, ϕ_k^*, that gives the largest information gain is stored at the LRT node, which is a *split node* as in the standard DF. The information gain whilst at the stage represented by the LTM vertex k is defined as:

$$Q_k(\mathcal{D}) = \text{tr}(\text{cov}(\mathcal{D}_k)) - \sum_i^{lc,rc} \frac{|\mathcal{D}^i|}{|\mathcal{D}|} \text{tr}\left(\text{cov}(\mathcal{D}^i{}_k)\right), \tag{6.4}$$

which is very similar to (3.3) except for the definition of $\text{cov}(\mathcal{X}_k)$: it is the covariance matrix of the random variables given by $\left\{ \left(\mathbf{x}^I_{lc(k)} - \mathbf{x}^I_k\right) \parallel \left(\mathbf{x}^I_{rc(k)} - \mathbf{x}^I_k\right) | I \in \mathcal{X} \right\}$, where $\parallel$ is the concatenate operator. The offset vectors indicate the offsets from the current centre to each centre of the left ($lc(k)$) and right ($rc(k)$) subregions, hence the random variables are 6 dimensional vectors.

This process is then repeated recursively on each split of the data, $\mathcal{D}^{lc}$ and $\mathcal{D}^{rc}$, until the information gain falls below a threshold.

At this point we introduce a *division node* which divides the current search objective into two finer ones and enters the next search stage. The division node duplicates the training data and continues to grow the tree along two separate paths, each corresponding to to one of the two children of the current LTM vertex, k. Additionally, for each training image, I, reaching this division node we store the vectors $\mathbf{j}_m = (\mathbf{x}_m - \mathbf{x}_k)$ corresponding to the 3D offsets of k and its children $m \in \{lc(k), rc(k)\}$.

This process of split followed by division is then repeated until the LTM vertex, k, to be considered is a leaf; at which point we create a leaf node in the LRT corresponding to the skeletal part represented by k. The leaf node stores information about the 3D offset of k from its parent $p(k)$, that being $\left(\mathbf{x}_k - \mathbf{x}_{p(k)}\right)$.

As previously mentioned, the hand has many complex articulations and self-occlusions, thus, in order to fully capture this variation the training set used is too large to fit into memory. To retain training efficiency we make use of the fact that we train in coarse-to-fine stages based on the learnt LTM. An intuition is that coarse stages require

less training data than the fine ones, therefore we can gradually add more training data at each stage of the training procedure.

For an LTM of maximum depth d, we split the training data, $\mathcal{D}$, into d equally sized random, disjoint subsets $\mathcal{D}_0, ...\mathcal{D}_{d-1}$. We start training an LRT with $\mathcal{D}_0$ for the first stage, and for each stage after we add an additional subset to the training data. That is, for stage s the training set is composed of $\mathcal{D}_s \cup \mathcal{D}_{s-1}$. The training procedure to grow a single LRT is described in Algorithm 6.

As explained in previous sections, a multi-stage coarse-to-fine structured search is efficient. However, an underlying risk is that the dependency between stages can lead to error accumulation throughout the search. To compensate for this, we embed an *error regressor* inspired by [Saffari et al., 2009] into each stage of LRF. After training stage s with set $\mathcal{D}_s$ and before creating a *division node* , we use $\mathcal{D}_{s+1}$ to validate the trained forest so far. For each sample $\mathbf{x}_i \in \mathcal{D}_{s+1}$, an error offset $\Delta\mathbf{j}$ between the ground truth and the estimation is measured. Similar to the previously described method of splitting, the forest is further grown for a few layers in order to minimise the variance of $\Delta\mathbf{j}$. Once the information gain falls below a threshold, a *division node* is generated and the forest training enters next stage, $s+1$.

6.5.2 Testing

At test time, pose estimation is performed on an image I as follows; we define the starting position for the search, $\mathbf{x}^I_{k=0}$ as the centre of mass of the segmented depth image, which corresponds to the root vertex of the LTM. Starting at the root of the LRF, the image traverses the tree, branching left or right according to the split-node function, until reaching a division node. For each offset, $\mathbf{j}_j$ stored at the division node, 3D votes are accumulated in two Hough spaces, $\mathbb{H}^{lc}$ and $\mathbb{H}^{rc}$, where the votes for $\mathbb{H}^{lc}$ are defined as $\left\{\mathbf{x}_k + \frac{\mathbf{j}_j}{\mathbf{x}_0} | \mathbf{j}_j \in \mathbf{j}_l\right\}$ and similarly for $\mathbb{H}^{rc}$. The modes of these two Hough spaces now

Algorithm 6 Growing an LRT

Require: A set of training samples $\mathcal{D}$; a pre-learned LTM $G = (\mathcal{O} \cup \mathcal{U}, \mathcal{E})$ with a maximum depth D.
Ensure: An LRT t

procedure Grow($\mathcal{D}$, M)
  Equally divide $\mathcal{D}$ into random subsets $\mathcal{D}_0, ...\mathcal{D}_D$ by the maximum depth of G
  Let $k = 0$, $n = 0$ ▷ Initialise kth node of LTM and nth node of LRT
  Let $s = 0$ ▷ First stage of training
  Split($k, n, \mathcal{D}_0, s$)
end procedure

function Split(k, n, $\mathcal{D}$, s)
  Randomly propose a set of split candidates Φ
  for all $\phi \in \Phi$ **do**
    Partition $\mathcal{D}$ into $\mathcal{D}^{lc}$ and S^{rc} by ϕ with (6.3).
  end for
  Use the quality function in (6.4) to find the optimal $\phi*$
  if $Q_k(\mathcal{D})$ is sufficient **then**
    Save n as a *split node* into t. ▷ Create a *split node*
    Split(k, $lc(n)$, $\mathcal{D}^{lc}$, s)
    Split(k, $rc(n)$, $\mathcal{D}^{rc}$, s)
  else if $k \in \mathcal{U}$ **then**
    Save n as a *division node* into t. ▷ Create a *division node*
    Let $\mathcal{D} = \mathcal{D} \cup \mathcal{D}_{s+1}$ ▷ Augmenting the training set
    Split($l(k)$, $l(n)$, $\mathcal{D}$, $s+1$)
    Split($r(k)$, $r(n)$, $\mathcal{D}$, $s+1$)
  else
    Save n as a *leaf node* into t. ▷ Create a *leaf node*
  end if
  Return
end function

represent the two new positions, $\mathbf{x}_{lc(k)}$ and $\mathbf{x}_{rc(k)}$, from which the next search stage begins. This process is then repeated recursively until each path terminates at a leaf node.

This process will result in the image reaching *multiple leaf nodes*, one for each terminating node in the LTM. Using the stored offsets at the leaf nodes, each leaf node votes for its corresponding skeletal part in a corresponding 3D Hough space. Aggregating votes of all trees, we locate the final positions of the hand parts by a structured search in the Hough space, for which the structure is dictated by the learnt LTM as follows. For each skeletal part, we assign to it a dependent observable vertex in the LTM which corresponds to the vertex with the smallest geodesic distance as calculated in the matrix, $\mathbf{D}$ (6.2). The location of each part in the Hough space is then defined as the maxima which is closest to the location of its dependent vertex.

6.5.3 Complexity

In contrast to the patch-based baseline that takes dense pixels as input [Keskin et al., 2012] our algorithm takes the entire hand region. Thus, while both methods are constrained in complexity by the depth of the trees d, ours processes much smaller amount of samples. This is because the number of pixels to be evaluated in patch-based approaches are usually in the order of thousands for a standard VGA image; whereas, in contrast, we only evaluate one sample per image. The complexity is then similar to a holistic method - though a bit higher due to the division. Since there are too many different factors, it is hard to compare the complexity formally. Instead we report the run-time speed in Section 6.6.

On the other hand, recall that the training of holistic and patch-based methods require to minimise the variance of 48 dimension offset vectors. In LRF, since the offset vectors are 6D, it is much faster for training one split node.

Algorithm 7 Testing with LRF

Require: A segmented depth image I; a forest F; an LTM G.
Ensure:

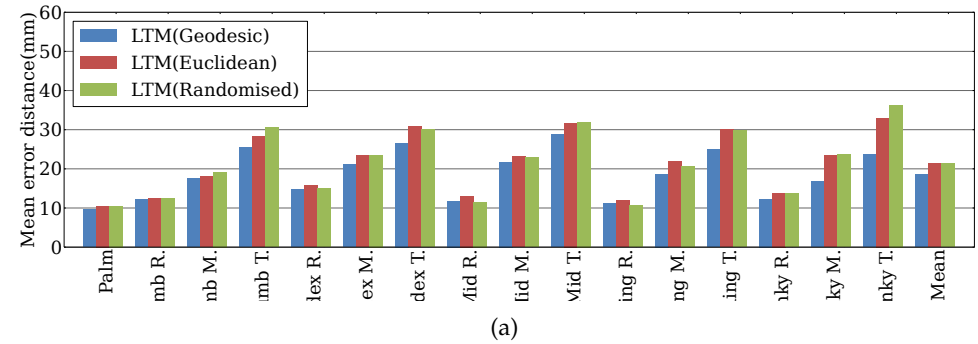
procedure TEST(I)
  Calculate centroid $\mathbf{x}_0$ of I.
  for all $t \in F$ **do**
    Let $k = G \rightarrow root$.
    Let $n = t \rightarrow root$.
    PROPAGATE($\mathbf{x}_0, k, n$)
  end for
  Aggregate results from all trees using *Meanshift*.
end procedure

function PROPAGATE($\mathbf{x}$, k, n)
  if n is a *leaf node* **then** ▷ *leaf node*
    Vote for the hand parts indicated by k.
  else if n is a *division node* **then** ▷ *division node*
    Predict centres of two subregions ($\mathbf{x}_{lc}, \mathbf{x}_{rc}$) with stored votes in n.
    PROPAGATE($\mathbf{x}_{lc}, k \rightarrow lc, n$) ▷ lc is the root node of left division branch.
    PROPAGATE($\mathbf{x}_{rc}, k \rightarrow rc, n$) ▷ rc is the root node of right division branch.
  else ▷ *split node*
    if $\phi(\mathbf{x}) == 1$ **then**
      Let $n = n \rightarrow lc$. ▷ lc is left child node.
    else
      Let $n = n \rightarrow rc$. ▷ rc is right child node.
    end if
    PROPAGATE($\mathbf{x}$, k, n)
  end if
end function

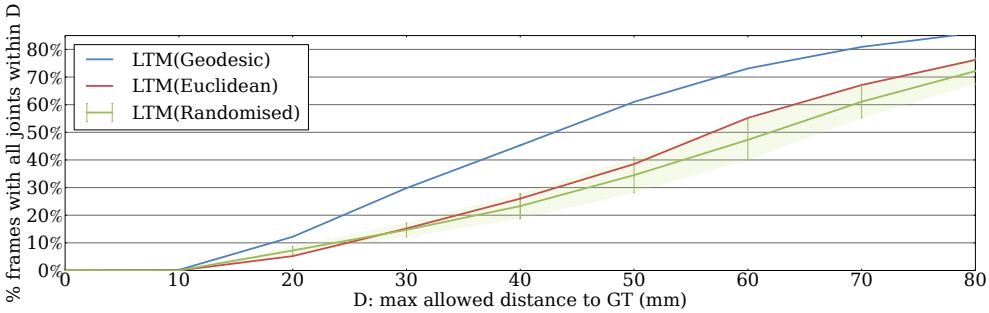
6.6 Experiments

To conduct experiments, we collect and annotate a new dataset with the Intel®'s *Creative Interactive Gesture Camera* sensor. Regarding this, we refer the readers to Section 4.1.2 for detailed descriptions.

In all experiments we train each LRF by evaluating 2000 splitting candidates at each node as in [Gall et al., 2011]. The threshold used to stop growing the tree at a particular



(a)



(b)

Figure 6.10: Effect of different LTMs. (R:MCP, M:PIP, T:DIP)

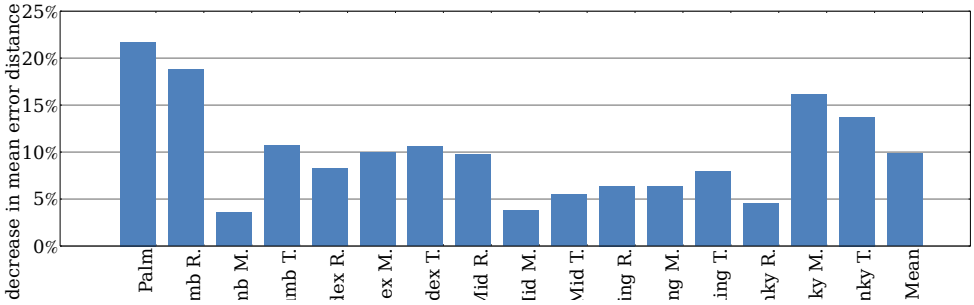


Figure 6.11: Error regression.

stage is chosen based on the size of a finger joint, which was set to $(10mm)^2$.

In Section 6.6.1 we conduct a self comparison of the different components in the LRF. Following this, In Section 6.6.2 we perform a thorough evaluation against other

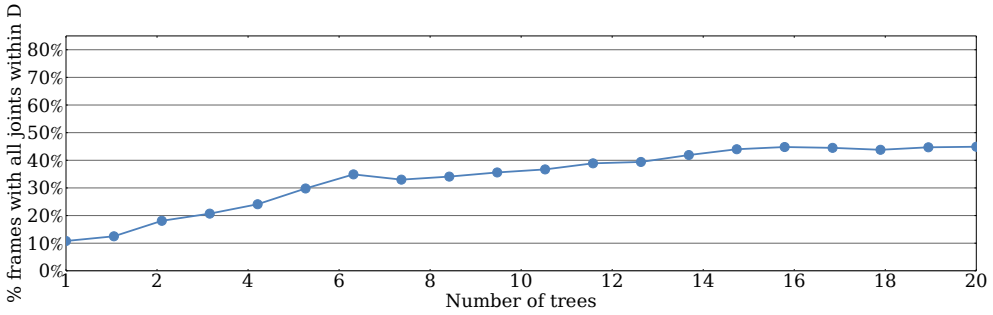


Figure 6.12: Worst case accuracy [Taylor et al., 2012] against number of trees. (The distance threshold $D = 35mm$)

state-of-the-art methods. Finally, in Fig. 6.19 and Fig. 6.20 we present some qualitative results.

6.6.1 Self Comparisons

To evaluate the impact of different distance metrics used when constructing the LTM we quantitatively measure the impact of the different topologies on performance. We compare LTMs generated using the Euclidean and geodesic distance as well as 5 randomly generated LTMs. For each of these 7 topologies, an LRF is trained on a subset of the training data and evaluated on sequence A.

Fig. 6.10(a) shows the standard evaluation metric of mean error, in *mm*, for each joint across the sequence. As shown, the Euclidean-generated LTM performs slightly better than the random ones, whereas the geodesic-generated LTM achieves the best performance on all hand parts except for two. In addition to this, we also employ the challenging metric *worst case accuracy* (see (4.5)). The results using this metric can be seen in Fig. 6.10(b). As shown, the Euclidean-generated LTM achieves the same performance as the upper-bound of performance from the random LTMs, whereas the geodesic-generated LTM significantly outperforms all of them showing a 20% improve-

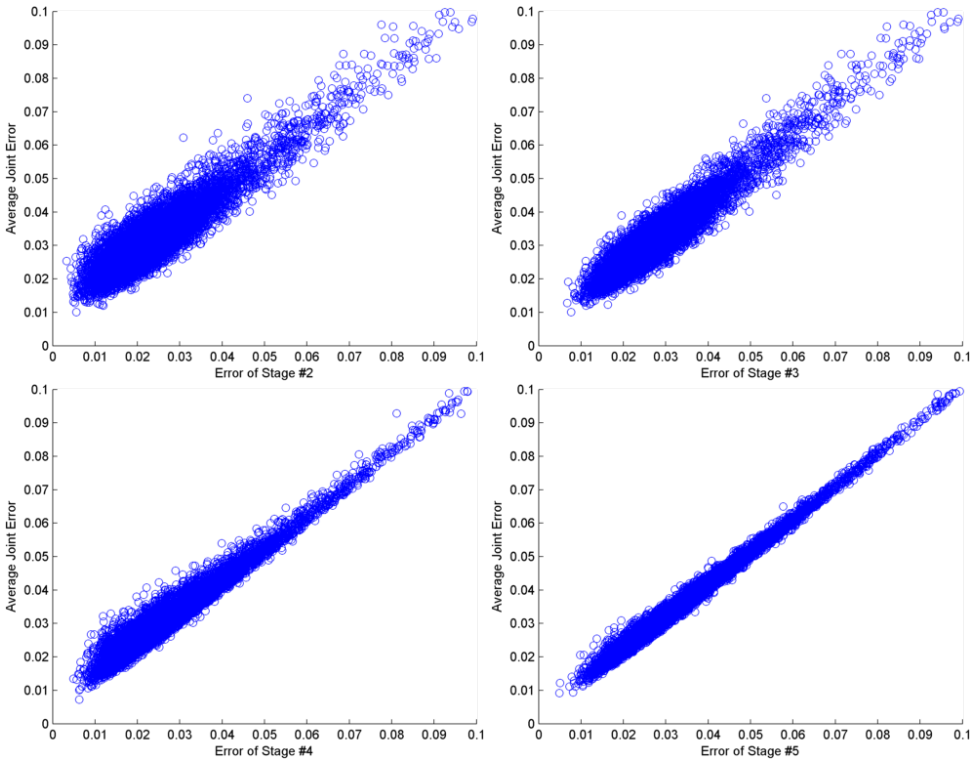


Figure 6.13: Error correlation between each stage (x axis) and final error (y axis). Δ_{avg} (4.2) is used as error measurement (unit: meter).

ment at a threshold of $40mm$.

Additionally, we evaluate the impact of the cascaded error regressor. In Fig. 6.11 we show the decrease in mean error distance for each part across the whole sequence. As can be seen, we achieve up to a 22% reduction in mean error for one hand part and and improvement of 10% on average.

In principle, since each tree generates much less votes comparing to traditional regression tree, more trees are needed in order to produce robust results. Fig. 6.12 shows the accuracy impact from different number of trees. A reasonable choice considering the trade-off between accuracy and efficiency is 16 trees, which is the setting we use in all experiments.

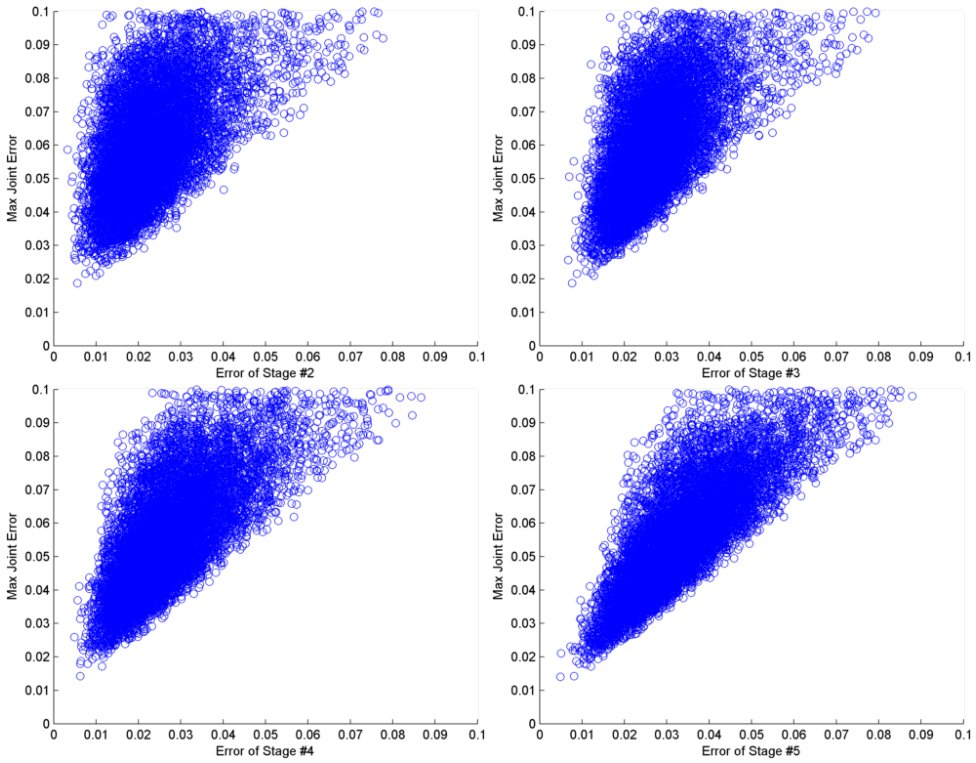


Figure 6.14: Error correlation between each stage (x axis) and final average error (y axis). Δ_{max} (4.3) is used as error measurement (unit: meter).

Fig. 6.13 shows a stage-wise experiement, measuring the error correlation between each stage and the final error (Δ_{avg}). We can see that the errors at early stages of LRF are generally larger than later stages. Similar observations can be found in Fig. 6.14 when using Δ_{max} as measurement.

6.6.2 Comparison on the ICVL dataset

We compare a 16-tree LRF with two state-of-the-art methods. The first is a regression version of [Keskin et al., 2012], for which we use our own implementation using the training parameters as described in [Keskin et al., 2012]. The second method we compare to is the model-based tracker of [Melax et al., 2013], for which we use a compiled

binary version provided by the authors. As this method is model based it requires calibration of the hand structure (width and height). Therefore, in order to do a fair comparison we compare to two versions of this method, one which has been calibrated and one which has not.

In Fig. 6.15 (a) and Fig. 6.16 (a) we show the cumulative moving average of the mean of hand part location error. As can be seen, our approach maintains a low average error throughout both sequences, and as expected the tracking based approaches reduce in error over time. In Fig. 6.15 (b) and Fig. 6.16 (b) we show the cumulative moving average of the palm prediction error, a relatively stable part. In Fig. 6.15 (c) and Fig. 6.16 (c) we show the cumulative moving average of the index fingertip error, a relatively unstable part. Notice, that after approximately the 500th frame the tracking based methods continuously decrease in accuracy for this part, indicating the tracking has failed and could not recover. This further highlights the benefit of using frame-based approaches. Additionally, in Fig. 6.17, we compare all methods using the more challenging metric proposed in [Taylor et al., 2012]. As can be seen our method largely outperforms the other state-of-the-arts.

6.6.3 Comparison on the MSHD dataset

In this section we first compare LRF with the baselines on the MSHD dataset. To best visualise the trade-off between accuracy and efficiency, we plot the mean error against the time cost. Unlike the experiments with the ICVL dataset, the patch-based method has better accuracy than the holistic one, at the cost of much lower efficiency. This is because of the aforementioned sparse distribution of samples in this dataset, which requires more generalisation. However, LRF still outperforms both, and yet maintaining a fast speed.

Furthermore, we also compare with the method from [Sharp et al., 2015] (kindly

Table 6.1: Efficiency comparison.

Method	Time (FPS)	Parallelism
LRF	62.5	None
Holistic	75	None
Patch-based [Keskin et al., 2012]	8.6	None
Melax et al. [Melax et al., 2013]	60	CPU (i7 2.2GHz)
Oikonomidis et al. [Oikonomidis et al., 2011b]	6	GPU (GT 640)
Xu et al. [Xu and Cheng, 2013]	12	None
Sridhar et al. [Sridhar et al., 2013]	10	GPU (NVS 300)
Qian et al. [Qian et al., 2014]	25	CPU (i7 3.4GHz)

provided by the authors), which is a discriminative and generative hybrid method. Its generative part is a tracking-based method that built on PSO. Since the MSHD test set has no temporal coherence, to compare fairly, we only compare LRF with their discriminative part. Similar to the patch-based baseline, its discriminative part is also forest-based and requires patches as input. But rather than regressing on the full pose directly, it maps the input to certain pose distributions (close, open, fist, etc.) and draws samples from them. These samples are supposed to be optimised by PSO. But here we only use their energy function to choose the best sample and measure its error. Usually the error reduces as more samples are drawn, but converges at some point. Thus the results appear to be a curve on the plot Fig. 6.18. Again, LRF can achieve better accuracy with lower time cost.

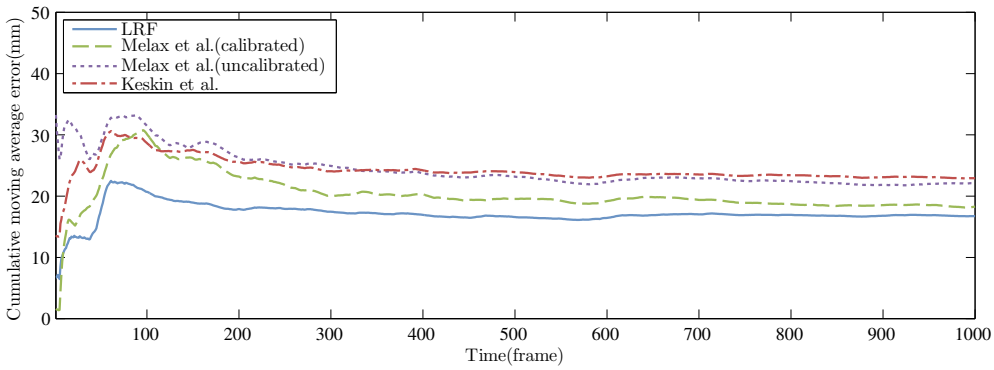
6.6.4 Efficiency

Regarding the run-time speed, our method runs in real-time at 62.5 FPS which is comparable to the holistic baseline (75 FPS) and much faster than [Keskin et al., 2012]

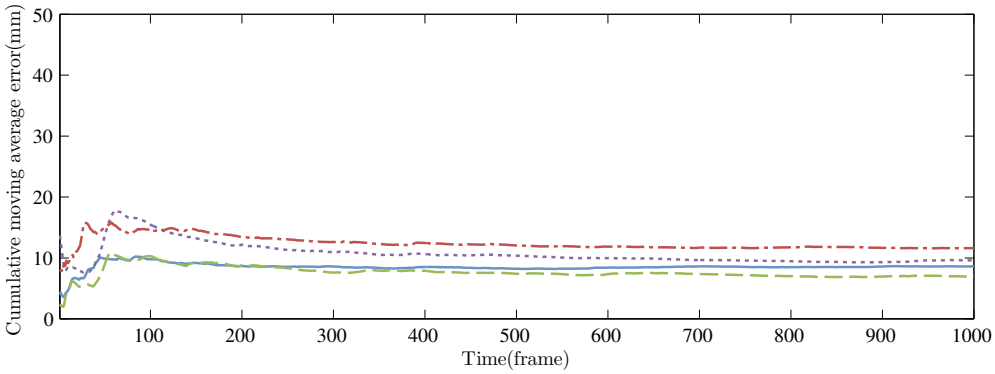
(8.6 FPS). Note that our method and the baselines are unoptimised—single threaded, without any CPU/GPU parallelism. In addition, we also report the speed of other methods such as [Melax et al., 2013] (60 FPS), [Oikonomidis et al., 2011b] (6 FPS), [Xu and Cheng, 2013] (12 FPS), [Sridhar et al., 2013] (10 FPS) and [Qian et al., 2014] (25 FPS). Most of them are either CPU or GPU optimised, as indicated in Table 6.1.

6.6.5 Qualitative Results

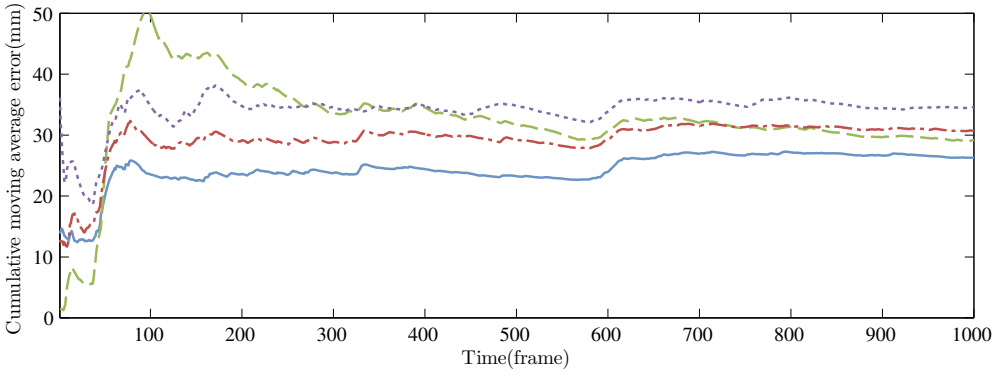
Qualitative results from the ICVL dataset are shown in Fig. 6.19 and Fig. 6.20. In the failure cases, despite that some of the part locations are wrong, especially distal phalanges which have larger depth in the LTM, the kinematic structure is preserved to some degree. However, due to lack of collision constraints, predictions of different parts are sometimes in the same location. In Fig. 6.21, examples from the MSHD dataset are shown. We order the result by mean error and visualise 5 images from the best, middle and worst cases respectively. The mid results tell us that errors usually appear at the distal phalanges. The worst results show that this method often struggles in the *egocentric* setting, where most of the hand parts are self-occluded. Finally a few shots from our demo video are shown in Fig. 6.22, which reflect the real-time performance of LRF. Due to the 30Hz frame-rate limit of the sensor, we can not fully demonstrate the efficiency advantage in real scenario. Note that in the $7th$ frame, the target hand moves out of the camera view and moves back in the $8th$ frame. This is usually the point where a tracking-based method fails. However our method is single-frame based and will not be affected. For more details readers can refer to our demo video.



(a) Test sequence A (average error)

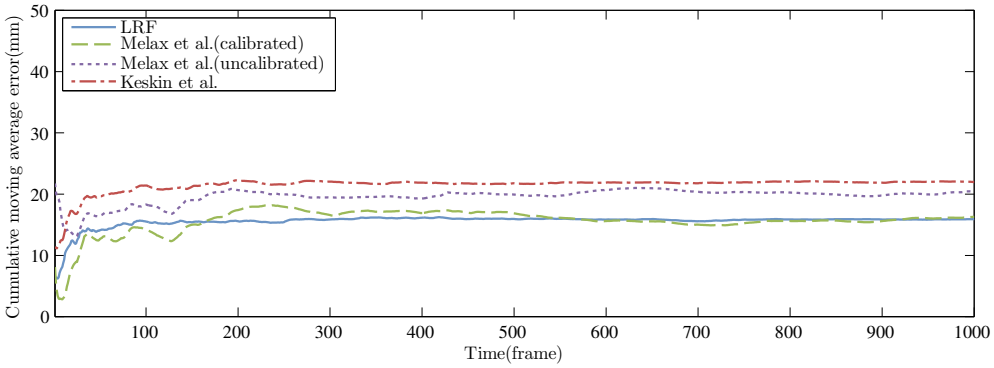


(b) Test sequence A(palm)

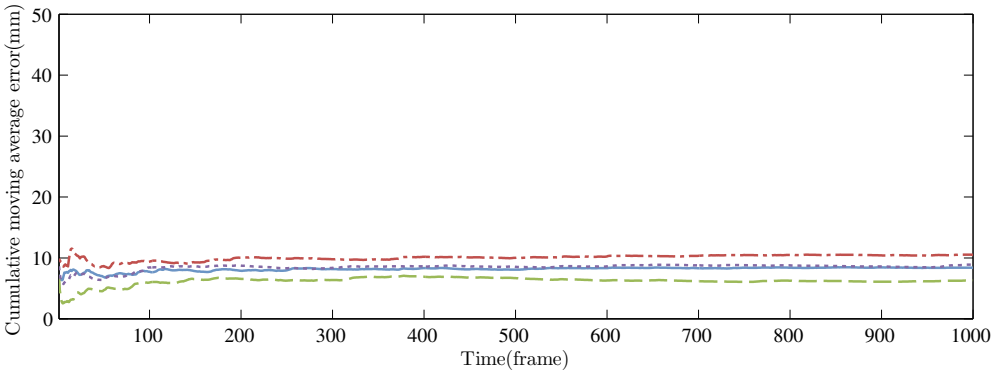


(c) Test sequence A(index tip)

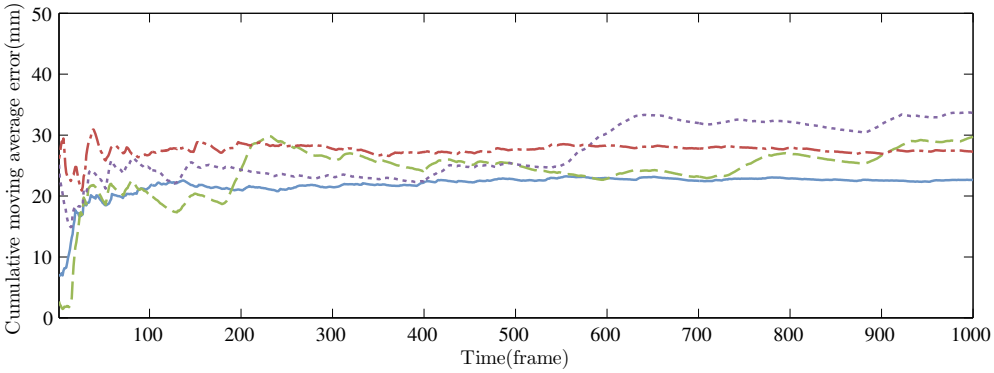
Figure 6.15: Quantitative comparison against state-of-the-art methods.



(a) Test sequence *B*(average error)

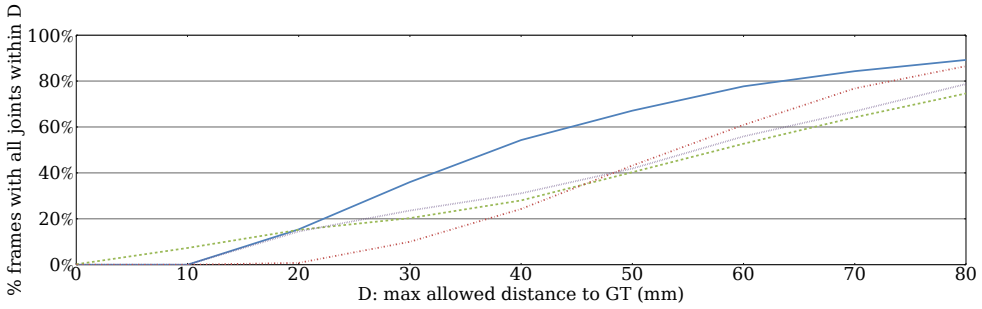


(b) Test sequence *B*(average error)

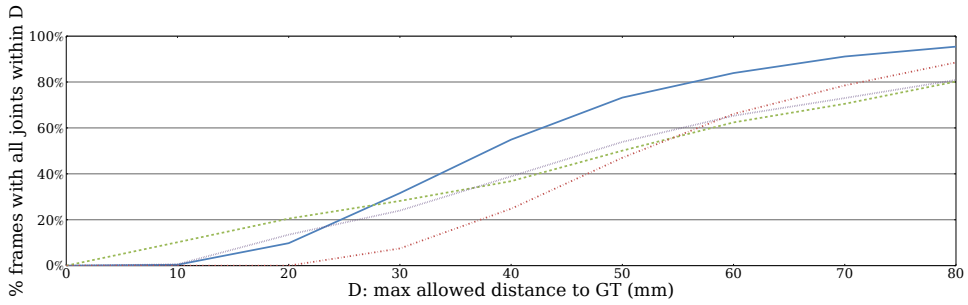


(c) Test sequence *B*(index tip)

Figure 6.16: Quantitative comparison against state-of-the-art methods.



(a) Worst case accuracy [Taylor et al., 2012] of sequence *A*



(b) Worst case accuracy [Taylor et al., 2012] of sequence *B*

Figure 6.17: Quantitative comparison against state-of-the-art methods on the ICVL v2 dataset.

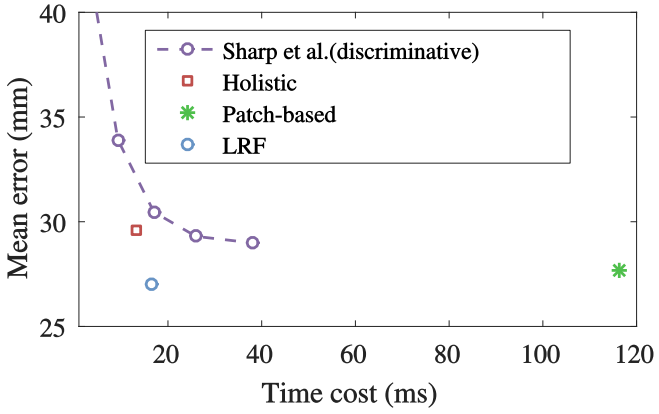


Figure 6.18: Trade-off comparison on the MSHD dataset. Note that we only compare with the discriminative part of [Sharp et al., 2015], which is CPU-optimised. Implementations of LRF and the other baselines do not use any CPU/GPU parallelism. Note that [Sharp et al., 2015] predicts 21 joint locations. To be fair we do not consider the 5 finger tip positions when calculating the error of [Sharp et al., 2015].

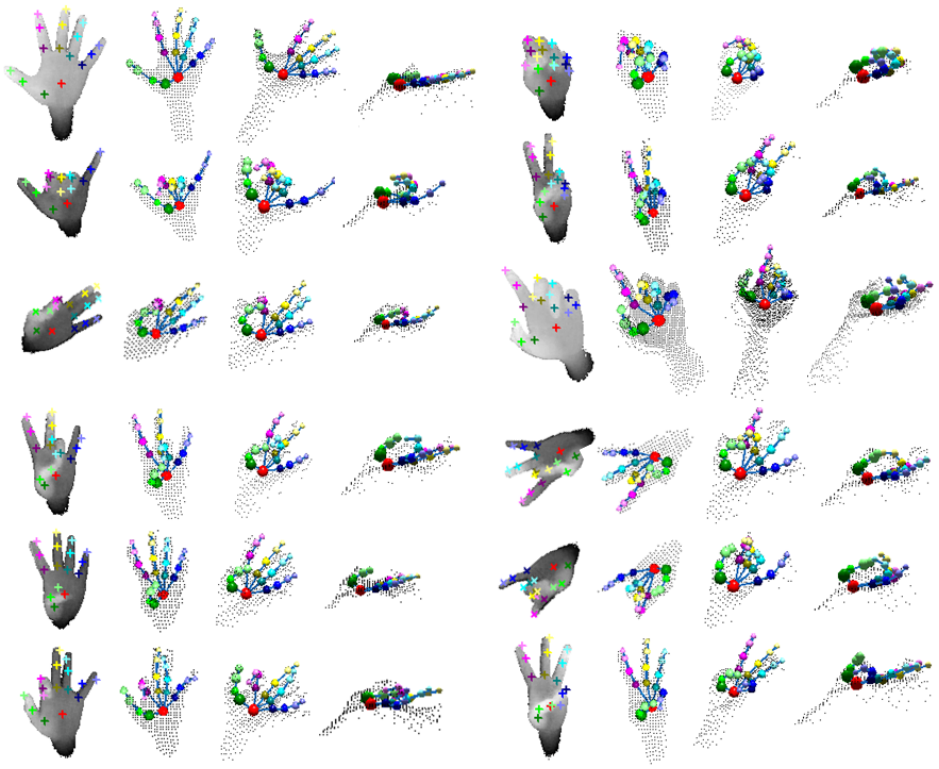


Figure 6.19: Successful results of LRF. We show the localisation on the depth image followed by a visualisation of the estimated 3D hand part locations from multiple angles.

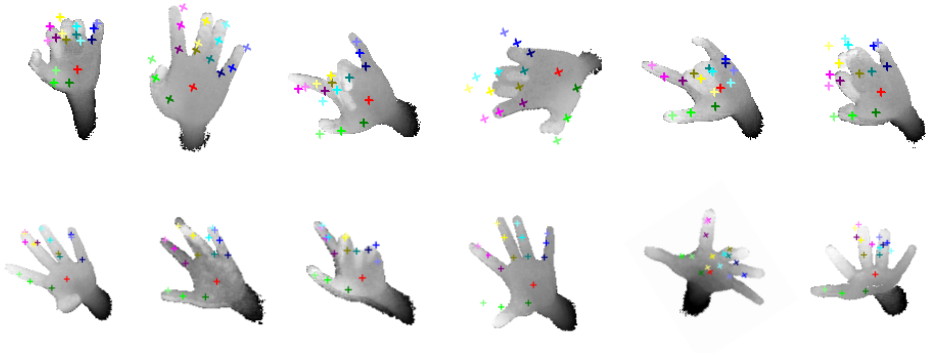
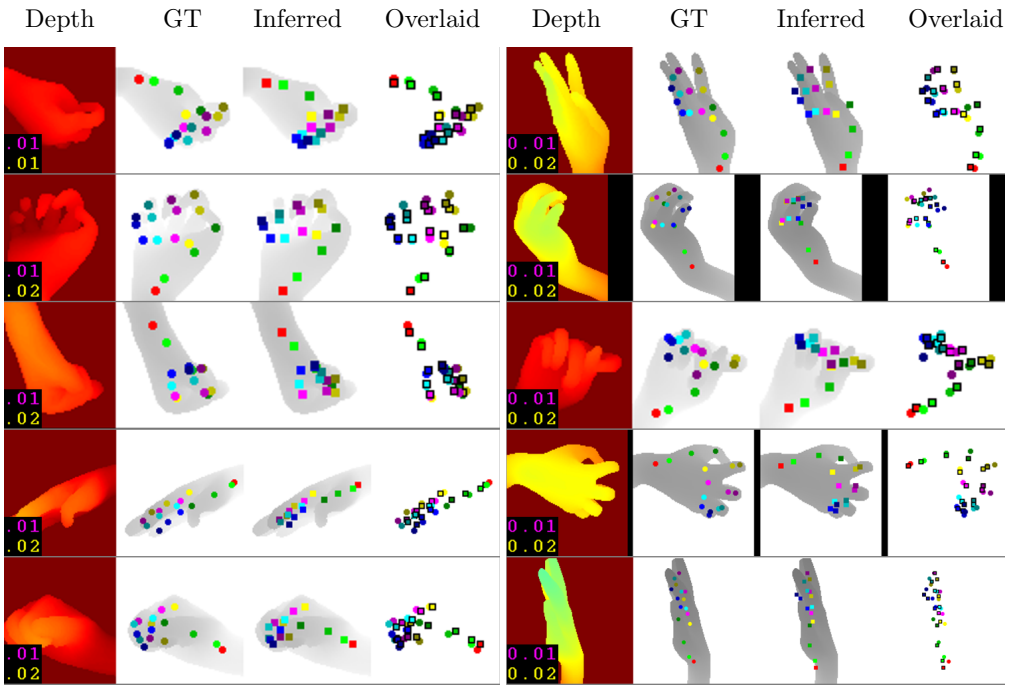
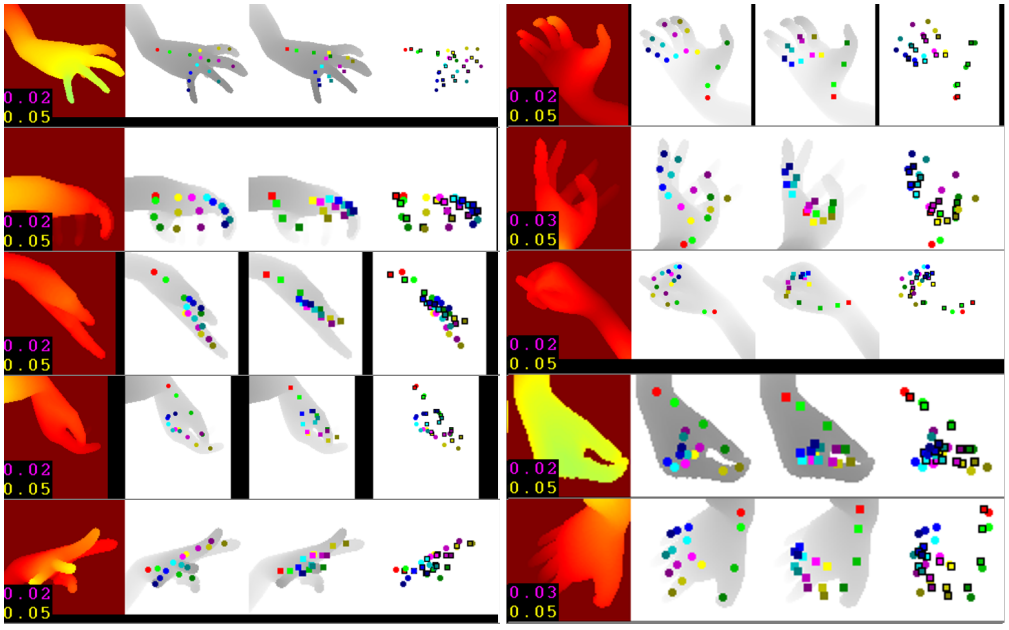


Figure 6.20: Failure cases of LRF. Note however that the structure of the output is still in line with the hand topology.

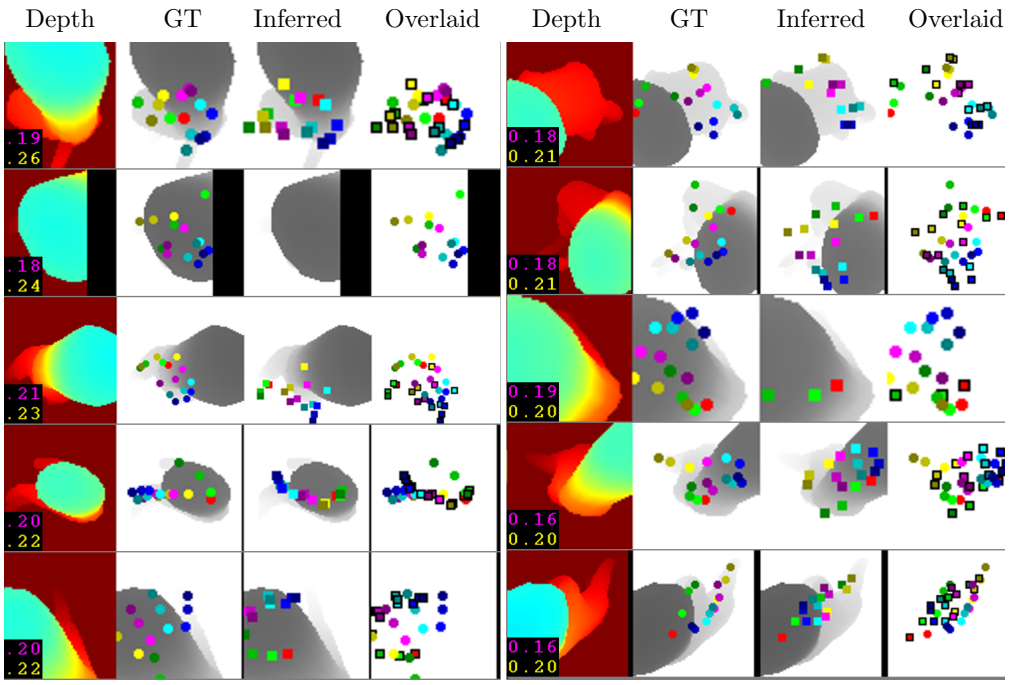


(a) 10 best results



(b) 10 middle results

Figure 6.21: Qualitative results from the MSHD dataset ordered by the error Δ_{avg}. Depth image, groundtruth, inferred results, and inferred overlaid on groudtruth are shown. Numbers at bottom-left indicates the errors. Pink: Δ_{avg}, yellow: Δ_{max}.



(c) 10 worst results

Figure 6.21: Qualitative results from the MSHD dataset.

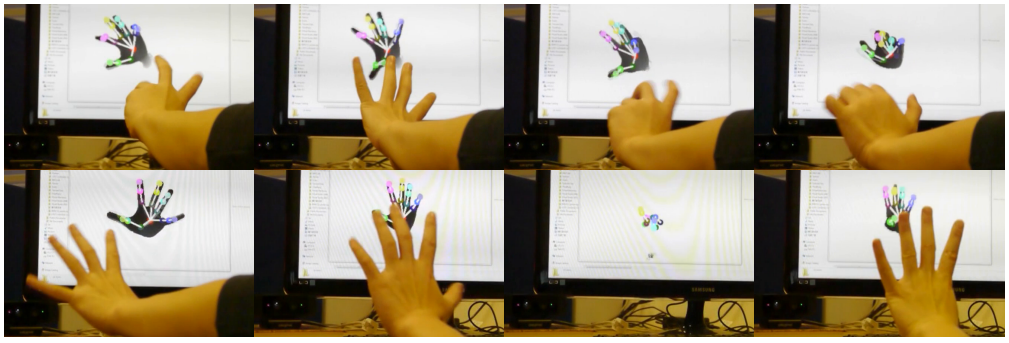


Figure 6.22: Screen shots from the demo video that shows real-time performance. In frame #7, the target hand moves out of frame and moves back in later. And the accuracy is not affected in frame #8. For more details readers can refer to the supplementary video of [Tang et al., 2014].

6.7 Summary

In this chapter we present the latent regression forest (LRF), a method for real-time estimation of 3D articulated hand pose. Current discriminative methods in the field are either holistic or patch-based. Although patch-based methods achieve state-of-the-art performance, we manage to improve its efficiency by incorporating holistic concepts whilst keeping its generalization power.

We first represent the topology of human hand by a latent tree model (LTM). Its tree-based structure not only implicitly preserves kinematic informations to some degree, but also naturally turns this problem into a structured coarse-to-fine search for skeletal parts as we desire. Furthermore, we propose to use CLNJ to learn the structure of LTM in an unsupervised manner. After some trials, we find that using geodesic distance as metric for CLNJ better preserves the kinematic structure. Compared to other forest-based methods that take dense pixels as input, our method is applied on the whole image as opposed to individual pixels, greatly increasing the run-time speed. As samples propagated down an LRF, the algorithm focuses on local appearance and thus more flexible.

To the best of our knowledge this is the first work combining LTM and decision forest. Since both have tree-based hierarchical structure, the integration is natural and easy to implement. This allows us to apply the LRF to many vision problems that need a structured search, either spatially or temporally. Moreoever, LRF can potentially be applied to many existing topics of LTM from other fields, spanning from marketing to medicine. Readers may refer to [Mourad et al., 2013] for more applications.

There are also a few drawbacks with LRF. Firstly, starting from holistic is efficient, but also makes LRF vulnerable for occlusions, e.g. , when the hand is holding an object or two hands are interacting. As mentioned in introduction, these cases are out

of the scope of this thesis. Ideas from [Oikonomidis et al., 2011b, Oikonomidis et al., 2012, Tzionas et al., 2014, Tzionas and Gall, 2015] can be combined in future work. Secondly, although globally the results are constrained by the topology, locally it is not guaranteed to be kinematically correct. Thirdly, correlated with the previous point, the outputs are 3D locations. Angular forms are more desired as they are kinematically constrained and can be rendered. Fourthly, although we incorporate an error regression into training to minimise error accumulation, there is no such strategy during testing. Last but not the least, although LRF has multiple stages, due to the latent nature of intermediate results, they can not be verified easily. For instance, an intermediate result can be outside the pointcloud silhouette and still be legitimate. We can not verify the results until having the final full pose.

CHAPTER 7

Hierarchical Sampling Forests

a progressive search for poses

Contents

7.1 Overview

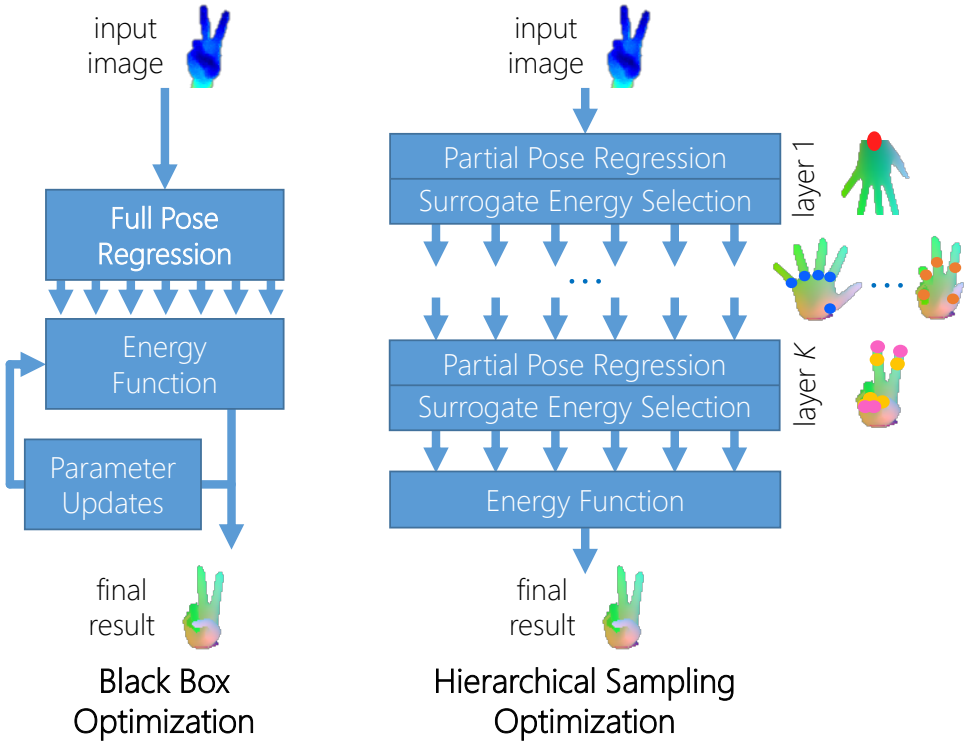


Figure 7.1: Opening the black box. A typical black box optimization approach (left) will regress candidate full poses and then iteratively update them based on the evaluation of the energy function. Our approach (right) instead regresses partial poses in a layered kinematic hierarchy, using a surrogate energy function to keep only the best partial poses. Our approach can achieve higher accuracy than black box optimization, even without iteration.

By 2015, numerous methods have been proposed for the 3D hand pose regression. Most discriminative methods [Romero et al., 2009, Wang and Popović, 2009, Keskin et al., 2012, Tang et al., 2013] treat this problem as a direct inverse-rendering problem, such that $f : \mathbf{x} \mapsto \mathbf{y}$, where a predictor f regresses full pose $\mathbf{y}$ on input $\mathbf{x}$, and $\mathbf{x}$ can be considered rendered from $\mathbf{y}$. A few methods, such as LRF [Tang et al., 2014] and [Sun et al., 2015], decompose the regression of $\mathbf{y}$ into multiple stages. However,

they do not exploit the intermediate results. Generative methods such as [Oikonomidis et al., 2011a, Melax et al., 2013, Qian et al., 2014] renders full poses from hypothesis and measure the error between rendered appearance and observation. State-of-the-art method [Sharp et al., 2015], though combining discriminative and generative, also regresses on full pose candidates and pass them on to the generative part. We call all these methods 'black box optimization', since they do not exploit the intermediate steps, i.e. , partial poses, of the inverse-rendering problem.

Apart from that, the discriminative part of [Sharp et al., 2015] utilizes an interesting strategy. To serve the purpose of feeding full pose hypotheses to the generative part, i.e. , particle swarm optimisation (PSO), it has to meet a few requirements. First of all, for scoring the pose hypotheses, PSO requires an energy function representing the L1 difference between a rendered hypothesis and the input pointcloud. Recall that in Section 1.1 we have defined three types of intermediate parameters: part label$p$, offset vote $\mathbf{j}$ and joint angle $\boldsymbol{\theta}$. Although the final joint locations $\mathbf{y}$ can be calculated from all three types, the angular parameter $\boldsymbol{\theta}$ is a necessity for rendering. Also, if one would like to design a coarse-to-fine multiple stage predictor, there are two existing options. The first one is LRF described in the previous chapter. Unfortunately, joint angles do not comply with the latent structure. Another option is to utilise a kinematic structure [Sun et al., 2015]. In this case we can decompose $\boldsymbol{\theta}$ into multiple meaningful joint angles. However, due to the ambiguity of joint angles, as shown in Fig. 7.2, a discriminative regressor which predicts the angle of a finger joint does not work well. As a result, [Sharp et al., 2015] choose to directly regress full poses.

Secondly, since PSO takes diverse particle samples[1] as input, the discriminative part needs to generate a set of full pose hypotheses $\mathcal{Y} : \{\mathbf{y}\}$. And more importantly, the minimum set error given by the following equation is highly correlated to the per-

[1] In this case, each particle is a full hand pose hypothesis.

formance of PSO.

$$\Delta* = \min_{\mathbf{y}\in\mathcal{Y}} \Delta_\delta(\mathbf{y}), \tag{7.1}$$

where Δ_δ is the choice of error measurement that can be either (4.2) or (4.3). In other words, we hope $\mathcal{Y}$ can be diverse and at least one hypothesis is as close to ground-truth as possible. This motivates [Sharp et al., 2015] to adopt a sampling scheme, where the forest-based regressors map each input to a distribution $\mathcal{G}$, from which a set of full pose hypotheses are sampled. This idea has its root from the multiple-output framework [Guzman-Rivera et al., 2014]. However, the way [Sharp et al., 2015] performs sampling is to classify input into one of 7 predefined poses, such as fist, pointing, open, etc.., and then generate hypotheses by perturbing from these discrete poses. Since these hypotheses have high uncertainty (large $\Delta*$), we typically need to generate a large number of them so that PSO can have satisfying accuracy.

In this chapter, we propose to open up the black box and exploit our high-level knowledge about (i) the rendering process, and (ii) the relationship between the pose parameters. To this end, we propose to decompose the hand pose by kinematic structure. Our approach has a tight coupling between the generative and discriminative aspects, and results in high quality pose estimates at very high efficiency. As shown in Fig. 7.1, instead of directly regressing and optimizing the full hand pose (e.g. [Sharp et al., 2015]), our new framework, termed *hierarchical sampling optimization*, builds up the pose parameters layer by layer, guided by the kinematic hierarchy. At each layer, a new set of candidate sample partial poses is regressed, conditioned on the result at the previous layer. We describe a surrogate energy function that, based on a high-level view of the rendering process, can quickly cull the bad samples.

To demonstrate the efficacy of our approach, we perform an exhaustive evaluation of our method on three publicly available datasets. The experimental results show that our method outperforms six state-of-the-art methods on these datasets while considerably improving the efficiency compared to black box approaches.

To the best of our knowledge, this is the first discriminatively trained pipeline for inverting the graphic rendering procedures that is guided by the relationships between the arguments of the generative procedure. We expect that our work can be more broadly applied to other vision tasks that can be formulated as inverse problems.

7.1.1 Contributions

The main contributions of this chapter are as follows.

- Propose hierarchical sampling forests (HSF) to decompose the problem by kinematic structure into several stages. Comparing to latent structure such as LRF, the intermediate results are observable and thus verifiable.
- Combine the hierarchical structure forest with a multiple-output framework, which will significantly reduce the error accumulation.
- To avoid the ambiguity of joint angles, we propose a cross-modality technique to train on 3D Euclidean locations and predict joint angles.
- Propose an efficient surrogate energy function which can be applied to partial poses. Therefore we can perform optimization and tracking in between the hierarchical structure.

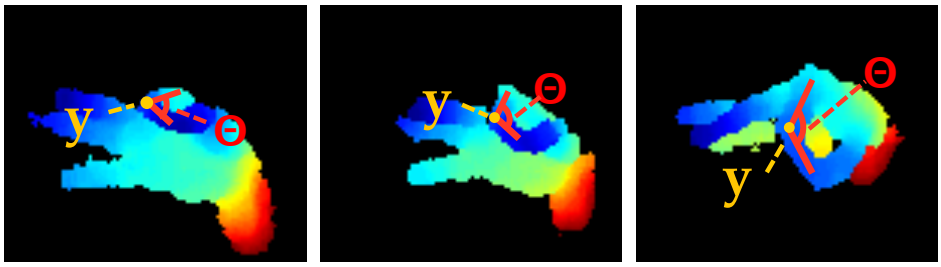


Figure 7.2: The ambiguity of local joint angles. 3 frames from NYU dataset [Tompson et al., 2014]. Due to global rotation, the same gesture looks rather different across these frames. Although the 3D location $\mathbf{y}$ changes, joint angle $\boldsymbol{\theta}$ stays the same (in 3D) and trackable.

7.2 Related Work

7.2.1 Kinematic Hierarchy

There have been numerous works leveraging the kinematic structure of human body. [Yang and Ramanan, 2011] propose a mixture model to capture contextual relations between body parts. Similar to LRF, [Wang and Li, 2013] also utilise latent tree model to represent body structure. However, the LTM they learn only has observable nodes, thus is more like a kinematic tree. The closest literature to our work is [Yub Jung et al., 2015], which performs random walk and sampling along kinematic structure.

In the field of 3D hand pose regression, [Sun et al., 2015] also has a hierarchical kinematic structure. However, unlike our proposal, they do not refine the predictions and also do not reason about the likelihood of partial hypotheses as they are built. This prevents them from pruning bad hypotheses quickly. Also their method work with 3D Euclidean locations only, which can not be utilised and refined by a generative method as in [Sharp et al., 2015].

7.2.2 Multiple Output

Multiple output prediction is a set of machine learning methods which can trace to as early as 70s'. Early methods often take an maximum a-posteriori (MAP) view by finding the top M most probable hypothesis [Lawler, 1972]. [Fromer and Globerson, 2009] formulate the MAP problem as a linear program (LP) on a particular polytope, whilst [Flerova et al., 2012] provide a dynamic programming view. Recently the multiple output concept has been applied to many computer vision problems such as body pose estimation [Yang and Ramanan, 2011], segmentation [Batra et al., 2012, Guzman-Rivera et al., 2012], scene relocation [Guzman-Rivera et al., 2014], etc. Among them, [Guzman-Rivera et al., 2012] and [Guzman-Rivera et al., 2014] propose to use discriminative model for learning how to generate multiple choices.

7.3 The Pose Estimation Inverse Problem

This section formulates hand pose estimation as an inverse problem. We denote the input depth image I, viewed as a function of pixel location $\mathbf{x}$, as $I(\mathbf{x}) \in [0, \infty)$. To that end, we explicitly assume that a hand with pose $\mathbf{y}$, or its angular form $\boldsymbol{\theta}$ gives rise to a depth image I, via a computer graphics rendering process. Treating pose estimation as an inverse problem explicitly assumes that a hand with pose $\boldsymbol{\theta}$ gives rise to a rendered depth image $R_{\boldsymbol{\theta}}(\mathbf{x})$ with the same range as $I(\mathbf{x})$. The goal is to then invert the process by finding the parameters $\boldsymbol{\theta}$ such that $R_{\boldsymbol{\theta}} \simeq I$. As our model of the true process which gave rise to I will be imperfect, we cannot expect perfect equality, and instead aim to minimize an energy function that measures the error in reconstructing I with $R_{\mathbf{y}}$ (see Section 7.5.1 for more details).

The following sections are organized as this: we start by describing how we parametrise the human hand, highlighting the inherent structure in the pose parame-

ters that our approach exploits. And then we outline the reconstruction error function and the partial reconstruction error (henceforth referred to as the 'silver' energy) functions. After that we propose the hierarchical sampling forests (HSF), a new form of hybrid generative-discriminative model that hierarchically constructs a full hand pose parameter vector in parts.

7.4 Pose Parametrisation

The full pose vector $\mathbf{y}$ is arranged in a standard kinematic tree defined by the skeletal structure of the hand (see Fig. 7.3 right for an illustration). The tree has four layers: layers 1 and 2 respectively predict wrist position and rotation; layer 3 contains one forest per finger, each of which predicts its respective metacarpophalangeal (MCP) rotations (flexion and abduction); layer 4 also specializes per finger, and jointly predicts the (highly correlated) flexions for the proximal Interphalangeal (PIP) and distal Interphalangeal (DIP) joints. Finally, we concatenate the partial poses resulting in a hypothesis of the full hand pose. The kinematic tree structure then approximates the human anatomical hand skeleton, with each joint's parameters specifying its *rotation* relative to its parent. As is standard we assume a fixed hand shape that specifies the *translations* of each joint relative to its parent. Algorithm 8 defines the kinematic model in the form of pseudocode.

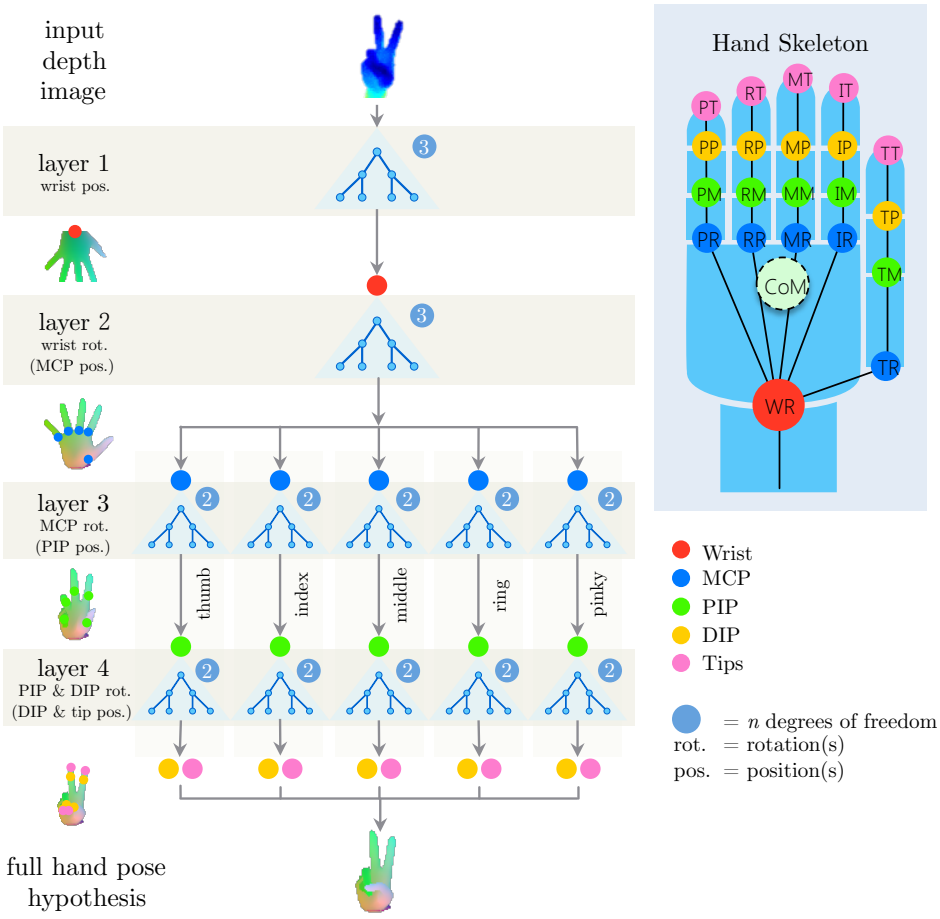


Figure 7.3: Layers in the pose hierarchy. Our approach regresses a full hand pose hypothesis in four layers. At each layer, we predict *particular subsets* of pose parameters, conditioned on all the previous layers' results. Samples from these distributions are filtered using a low-cost 'silver' energy function, and the best solution is passed to the kinematic children. The layers are aligned with the kinematic tree, shown top right: layers 1 and 2 respectively predict wrist position and rotation; layer 3 contains one forest per finger, each of which predicts its respective MCP rotations (flexion and abduction); layer 4 also specializes per finger, and jointly predicts the (highly correlated) flexions for the PIP and DIP joints. Finally, we concatenate the partial poses resulting in a hypothesis of the full hand pose.

Algorithm 8 Kinematic tree model (pseudocode in matlab style).

```
%Fields of a kinematic tree node:
%   Id
%   Input: input joint location (with codenames as in Fig. 7.3).
%   Output: predicted joints (with codenames as in Fig. 7.3).
%   Children: pointing to a set of kinematic tree node Ids.

%The kinematic tree
KinematicTree = [
struct('Id': 0, 'Input': [CoM], 'Output': [WR], 'Children': [1]),
struct('Id': 1, 'Input': [WR], 'Output': [TR, IR, MR, RR, PR],'Children': [2, 3, 4, 5, 6]),
struct('Id': 2, 'Input': [TR], 'Output': [TM], 'Children': [7]),
struct('Id': 3, 'Input': [IR], 'Output': [IM], 'Children': [8]),
struct('Id': 4, 'Input': [MR], 'Output': [MM], 'Children': [9]),
struct('Id': 5, 'Input': [RR], 'Output': [RM], 'Children': [10]),
struct('Id': 6, 'Input': [PR], 'Output': [PM], 'Children': [11]),
struct('Id': 7, 'Input': [TM], 'Output': [TT, TP], 'Children': []),
struct('Id': 8, 'Input': [IM], 'Output': [IT, IP], 'Children': []),
struct('Id': 9, 'Input': [MM], 'Output': [MT, MP], 'Children': []),
struct('Id': 10, 'Input': [RM], 'Output': [RT, RP], 'Children': []),
struct('Id': 11, 'Input': [PM], 'Output': [PT, PP], 'Children': []) ];
```

7.5 Energy Functions

Given a model of the hand, its pose (and sometimes shape [Khamis et al., 2015]) parameters are optimized to minimize an energy function.m An ideal energy function is the reconstruction error: the distance between the observed image and a synthetic rendering of the model. This rises from other machine vision areas which also requires fitting a deformable model to appearance data, usually combined with analysis by synthesis techniques. One example is the active appearance model (AAM for facial landmark detection [Cootes et al., 1998], which is extended to a 3D face model (yet still work with 2D appearance data) later [Blanz and Vetter, 1999]. Note that differentiating this reconstruction error is non-trivial. [Blanz and Vetter, 1999] use sum-of-square difference and explicitly compute the derivative. [Loper and Black, 2014] adopt a differentiable approximation at the 2D boundaries.

With an articulated model like hand, not only is the error differentiation difficult, the model parameters are also non-convex. [de La Gorce et al., 2011] tackle this by first designing an energy approximation with antialiasing, and then using a sequential quadratic programming technique to optimise it. A more popular way, which avoids the differentiation is to employ 'black box' optimization strategies such as PSO [Oikonomidis et al., 2011a,Qian et al., 2014,Sharp et al., 2015] that only require function evaluations, without gradients. While black box optimization can lead to high-quality results, it typically requires a large number of function evaluations and is therefore computationally expensive. To reduce the complexity, one can reduce the aforementioned min set error in (7.1), or the complexity of each function evaluation. Both ideas are highly related to the design of energy functions. In [Oikonomidis et al., 2012] and [Sharp et al., 2015], a rendering-based energy function is used, which can be applied to full pose only. This energy function has been termed as the 'golden energy' in [Sharp et al., 2015]. On the other hand, [Qian et al., 2014] propose an approximation of the golden energy, which does not require rendering and thus is efficient. In this section, we will revisit the golden energy, and then propose an efficient approximate that works on partial pose, termed as the 'silver energy'.

7.5.1 The Golden Energy

Given a full pose vector $\mathbf{y}$, how can we evaluate how well it 'fits' an observed test image? We exploit an energy function that proposed by [Oikonomidis et al., 2011a] and later dubbed the 'golden' energy [Sharp et al., 2015], which uses $\boldsymbol{\theta}$ to render a synthetic depth image of the hand and then compares each pixel using a robust L1 term:

$$\begin{aligned} E_{\text{Au}}(\boldsymbol{\theta}) &= \sum_{u} e(u, \boldsymbol{\theta}) \ , \\ e(u, \boldsymbol{\theta}) &= \min(|I(u) - R_{\boldsymbol{\theta}}(u)|, \tau) \ , \end{aligned} \tag{7.2}$$

Input depth $Z(u)$ | Rendered hypothesis $R_\theta(u)$ | L1 difference map $e(u, \theta)$

Figure 7.4: The golden energy (7.2): the input depth image, a rendered hypothesis and their L1 difference (white: high difference; red: low difference).

where $\boldsymbol{\theta}$ is a full hand pose parameter hypothesis, $I(u)$ is the observed depth value at pixel u, $R_{\boldsymbol{\theta}}(u)$ is the rendered depth value at pixel u under pose hypothesis $\boldsymbol{\theta}$, and τ is a truncation threshold, here set to 100mm.

The golden energy is effective at penalizing incorrect poses, including common problems such as 'model-over-background' and 'background-over-model'. However, computing E_{Au} is expensive: due to self-occlusions, a full rendering must be performed for each candidate value of $\boldsymbol{\theta}$ to obtain $R_{\boldsymbol{\theta}}$. Black box strategies to optimize E_{Au} therefore tend to require high-end GPU compute to render the possibly thousands of hypothesized poses that are needed per frame [Oikonomidis et al., 2011a, Sharp et al., 2015].

7.5.2 The Silver Energy

One of the main contributions of this paper is to show that we can achieve state-of-the-art results that optimize the golden energy (7.2), while requiring orders of magnitude fewer full golden energy evaluations. The key insight is that by exploiting the special kinematic structure of the pose parameter vector, we can decompose the problem into sub-problems which can be tackled independently using an efficient surrogate (or proxy) energy, which we dub the 'silver' energy. Our silver energy, E_{Ag}, is inspired by [Qian et al., 2014], but differs in that it is applied to each joint separately rather than

to a full pose vector.

Standard inverse approaches to pose estimation do not exploit the special structure of $\boldsymbol{\theta}$. In our approach, we decompose $\boldsymbol{\theta}$ into several *partial poses* that follow the four layers specified in Fig. 7.3. Layers 3 and 4 further decompose $\boldsymbol{\theta}$ by finger f, giving the complete set of partial poses as: $Y = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\} \cup \{\boldsymbol{\theta}_{3f}, \boldsymbol{\theta}_{4f} : f \in \{\underline{\text{t}}\text{humb}, \underline{\text{i}}\text{ndex}, \underline{\text{m}}\text{iddle}, \underline{\text{r}}\text{ing}, \underline{\text{p}}\text{inky}\}\}$.

The representation of partial poses is different for different layers of the framework. For instance, layer 1 predicts global translational offset (the vector from the image centroid to the wrist position), and thus $\boldsymbol{\theta}_1 = \mathbf{y}_1 \in \mathbb{R}^3$ is a 3D Euclidean vector, while layer 2's parameter set $\boldsymbol{\theta}_2$ encodes the global (palm) rotation as a unit quaternion [Oikonomidis et al., 2011a, Sharp et al., 2015], and so $\boldsymbol{\theta}_2 \in \mathbb{R}^4$. For the rest of bones, Euler angles are employed to represent bone rotations: the MCP joint has two DoF (flexion and abduction) so $\boldsymbol{\theta}_{3f} \in \mathbb{R}^2$, whilst the PIP and DIP joints have only 1 DoF each but are lumped together so that $\boldsymbol{\theta}_{4f}$ lies in $\mathbb{R}^2$ also. Note that while the above parametrisation has been presented specifically for the human hand, our approach could straightforwardly be extended to arbitrary tree structures.

For notational convenience we will use l to uniquely index any of the 12 partial poses, and 'layer l' to refer to the layer that contains the partial pose with index l. Additionally, we will use $\bar{\boldsymbol{\theta}}_l$ to denote partial pose $\boldsymbol{\theta}_l$ concatenated with the partial poses of layer l's ancestors. This is because we can not calculate the locations $\mathbf{y}_l$ from $\boldsymbol{\theta}_l$ alone.

The silver energy $E_{\text{Ag}}(\mathbf{y}_l)$ does not require rendering and works on each partial pose (locations) $\mathbf{y}_l$. We first use standard forward kinematic transformations (see e.g. [Taylor et al., 2012]) to compute the set of *positions* of the *children* bones C_l of partial angular pose $\boldsymbol{\theta}_l$ (the colored circles below each layer in the left column of Fig. 7.3). We write $\mathbf{y}_l$ to denote this set of positions. In more detail: Layer 1 predicts the wrist trans-

lation $\boldsymbol{\theta}_1$ from the image centroid, which, added to the observed centroid $\mathbf{x}_1 = \text{CoM}(I)$ gives the singleton set $\mathbf{y}_1 = \{\boldsymbol{\theta}_1 + \mathbf{x}_1\}$. Conditioned on $\mathbf{y}_1$ as input, such that $\mathbf{x}_2 = \mathbf{y}_1$, layer 2 predicts the wrist rotation $\boldsymbol{\theta}_2$, which uniquely determines the positions of the five MCP joints as the set $\mathbf{y}_2$. Layer 3 predicts the MCP rotation $\boldsymbol{\theta}_{3f}$ for each finger f, which uniquely determines the position of the PIP joint as the singleton set $\mathbf{y}_{3f}$. Finally, layer 4 jointly predicts PIP and DIP rotations $\boldsymbol{\theta}_{4f}$, giving the DIP and fingertip positions as set $\mathbf{y}_{4f}$. To summarise,

$$\mathbf{x}_l = \begin{cases} \text{CoM}(I) & \text{if } l = 1 \\ \mathbf{y}_{l-1} & \text{otherwise} \end{cases} , \tag{7.3}$$

The silver energy is then computed given layer l's predicted child positions $\mathbf{y}_l$ as

$$E_{\text{Ag}}(\mathbf{y}_l) = \sum_{x \in \mathbf{y}_l} \left[B(x) + D(x) \right] . \tag{7.4}$$

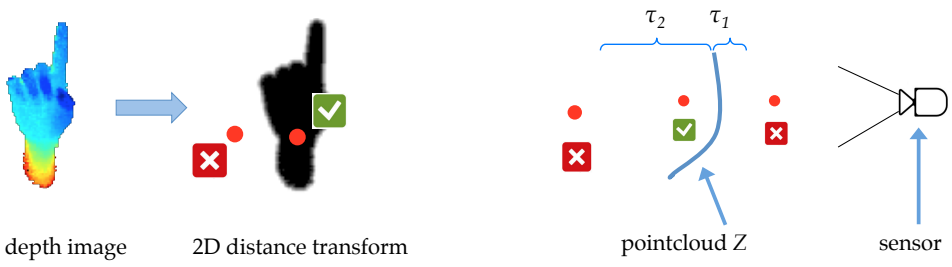
The first term encourages each child to lie inside the observed silhouette, and is defined as

$$B(x) = \min_{z \in I, z>0} (\| \pi(x) - z \|_2, \tau_0) , \tag{7.5}$$

where $\pi(x)$ projects a 3D position x into the image, and z represents all foreground data points. This term calculates the 2D truncated distance transform of the silhouette of hand input image I at x. The distance values are truncated to τ_0 (10 pixels outside). Then they are normalized to $[0, 1]$ outside and all 0 inside just to avoid weighting with the other term.

The second term in the silver energy aims to ensure the predicted depth at the child position roughly agrees with the observed depth. It is defined as

$$D(x) = \begin{cases} 0 & \text{if } B(x) > 0 \\ \max(1, \frac{I(x) - x_z - \tau_1}{\alpha}) & \text{if } I(x) - x_z > \tau_1 \\ \max(1, \frac{x_z - I(x) - \tau_2}{\alpha}) & \text{if } x_z - I(x) > \tau_2 \\ 0 & \text{otherwise} \end{cases} , \tag{7.6}$$



(a) $B(x)$ (b) $D(x)$

Figure 7.5: The silver energy: τ_1 and τ_2 in (7.6) form a 'safe zone' within a certain range of the depth value of each point. Red dots indicate the proposals for the wrist; tick means the proposal is accepted by the current term and cross means otherwise.

where x_z represents the z-coordinate of 3D position x. The thresholds τ_1 and τ_2 penalize the child positions living too far in front or behind the observed depth image. The whole silver energy can be considered as an efficient approximate of a 3D distance transform, forming a "safe zone" within $[\tau_1, \tau_2]$ behind the point cloud. If a joint is outside this region, a penalty is given. We set τ_1 to 15mm for the wrist and 5mm for other joints, whereas τ_2 is simply set to a common length of human hands (250mm) for accommodating self-occluded joints. α is set to $10mm$ to allow a soft penalty around the 'safe zone' border. The values of parameters are calculated from the 13 personalised mesh models in the MSHD dataset. As illustrated in Fig. 7.5, we generate three proposals (red dots), but only one of them is within the safe zone and thus accepted.

Note that in contrast to the golden energy, the silver energy can be computed extremely efficiently and can also be applied to partial poses $\mathbf{y}_l$. In the following sections we describe our algorithm for using the silver energy to quickly filter out bad partial pose hypotheses, greatly restricting the search space when we optimize the golden energy.

7.6 Hierarchical Sampling Optimisation

We now describe our main contribution, hierarchical sampling optimisation (HSO), a strategy for optimizing energies such as the golden energy. We start by providing a high-level overview of the method. We then describe how the method can efficiently sample partial poses using a regression forest, and finally move on to explain how the predictors used in the hypotheses generation phase are discriminatively trained.

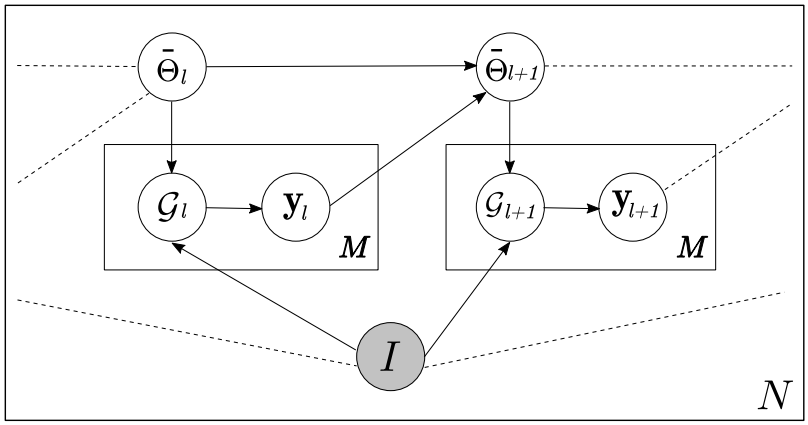


Figure 7.6: Graphical model of the prediction process.

7.6.1 Overview

The HSO approach is illustrated in Fig. 7.7. At a high-level, it looks much like a standard black box inverter: we generate N full hand pose hypotheses, and then select the pose $\boldsymbol{\theta}^*$ with the lowest golden energy $E_{\text{Au}}(\boldsymbol{\theta}^*)$. However, unlike a standard approach, each hypothesis is built up by following the kinematic structure of human hand as described in the previous section. Following many other approaches e.g. [Sharp et al., 2015], we assume that the foreground pixels are pre-segmented.

Each layer l takes an *evaluation position* $\mathbf{y}_l$ and generates M sample values for the

partial pose $\boldsymbol{\theta}_l$. Concatenating each partial pose sample with its ancestors' partial poses gives $\bar{\boldsymbol{\theta}}_l$, allowing us to compute the sample's child positions $\mathbf{y}_l$ and thereby the silver energy E_{Ag}. The sample with the minimal silver energy value is selected, giving $\boldsymbol{\theta}_l^*$. If multiple samples have the same energy values, one of them is selected randomly. Temporal information can optionally be incorporated by additionally evaluating the best partial pose $\boldsymbol{\theta}_l^*$ from the previous frame under E_{Ag}.

We then proceed to each child l' at the next layer in the hierarchy, conditioning on the result $\mathbf{y}_l^*$ just obtained from the relevant finger's output child position in $\mathbf{y}_l^*$. For layer 1, we use the image centroid for the evaluation position, i.e. $\mathbf{x}_1 = \text{CoM}(I)$. After layer four, the full pose vector $\mathbf{y}$ is reconstructed by simply concatenating each layer's best result. The graphical model in Fig. 7.6 depicts a fragment of this process.

7.6.2 Segmentation

As mentioned in Chapter 1, segmentation is non-trivial, although we have not properly tackled this so far. In previous chapters, we simply set a depth threshold to separate the foreground and background, which is infeasible in practice. In this chapter, we adopt the segmentation method from [Sharp et al., 2015], which turns this problem into a pixel classification. Rather classifying pixels into hand parts, we label all pixels with hand (positive) and non-hand (negative), and train a DF to deal with this binary classification problem.

7.6.3 Testing

A regression forest [Criminisi and Shotton, 2013] is simply an ensemble of decision trees, whereby at test time each internal tree node routes data to its left or right child-node by applying a threshold to a (possibly non-linear) projection of the input features.

Each input ultimately ends up in a leaf node, where a distribution is stored for sampling as described above. For the feature function, we simply employ the same modified two-pixel difference function (6.3).

The only remaining question is how to sample the partial poses $\boldsymbol{\theta}_l$. While in theory any sampling strategy could be used (e.g. sampling from a learned Gaussian, or from the prediction made by a convolutional neural network), we use a discriminatively trained regression forest. A single forest F_l is trained to predict each partial pose $\boldsymbol{\theta}_l$. The samples for $\boldsymbol{\theta}_l$ are generated by evaluating the forest *at just a single pixel*: the projection of 3D evaluation position $\mathbf{x}_l$ into the image.

Algorithm 9 The testing procedure of HSO

Require: A segmented depth image I; the kinematic model K as defined in Algorithm 8
Ensure: A full pose result $\mathbf{y}$
  procedure Test(I)
    Caculate $\mathbf{x}_0 = \text{CoM}(I)$. ▷ Use centre of mass as the starting point
    Let $k \leftarrow K[0]$ ▷ Root node
    for all $i \leftarrow 1$ to N **do**
      $\boldsymbol{\theta} = \text{DescendKinematicForest}(I, \mathbf{x}_0, k)$. ▷ Generate a full pose hypothesis $\boldsymbol{\theta}$.
      Calculate the golden energy E_{Au} of $\boldsymbol{\theta}$.
    end for
    Select the best $\boldsymbol{\theta}$ with lowest energy.
    Calculate $\mathbf{y}$ from the best $\boldsymbol{\theta}$ with forward kinematics.
    Return $\mathbf{y}$.
  end procedure

  function DescendKinematicForest(I, $\mathbf{x}$, k)
    Descend $\mathbf{x}$ down the forest that corresponds to k, which returns a GMM $\mathcal{G}$.
    Generate M samples from $\mathcal{G}$.
    Choose the best sample as $\boldsymbol{\theta}_k$ with the silver energy E_{Ag}.
    for all $c \in k \rightarrow$ Children **do**
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \bigcup \boldsymbol{\theta}_{kc}$
      Calculate the location of kc, $\mathbf{x}_{kc}$, with $\boldsymbol{\theta}_{kc}$ and forward kinematics.
      DescendKinematicTree($I, \mathbf{x}_{kc}, K[c]$).
    end for
    Return $\boldsymbol{\theta}$.
  end function

The forest itself is completely standard and uses 2-pixel comparison features. At each leaf node of forest F_l is stored a learned Gaussian mixture model over the parameters $\boldsymbol{\theta}_l$. We can thus quickly drawn M samples from this mixture model to generate the samples of $\boldsymbol{\theta}_l$ required for the hierarchical sampling optimization. Note that the forest prediction, the mixture model sampling, and the silver energy evaluation are all extremely efficient. Fig. 7.7 illustrates the testing procedure while Algorithm 9 provides a pseudocode.

7.6.4 Training

When training each tree, unlike previous methods that are applied to all foreground pixels [Keskin et al., 2011], in theory we only need 1 training examples corresponding to the evaluation position $\mathbf{x}_l$ per training image I, paired with its output label $\boldsymbol{\theta}_l$. With it we can calculate the final joint locations $\mathbf{y}_l$. In practice, to be more robust to sensor noise and to minimize error accumulation at test time, we jitter each $\mathbf{x}_l$ with random offsets within the range of ± 5 pixels to generate 5 training examples per image.

If one simply labels each training sample with angles and trains forest accordingly there can be problems because the local joint rotations we are trying to estimate do not necessarily correlated with changes in appearance (and thus the feature responses of (3.12)). This is illustrated in Fig. 7.2, where local appearance varies significantly in these 3 frames due to global rotation changes, but the index finger's joint angle that we are aiming to predict stays the same. Further, generating an output distribution $\mathcal{G}$ involves angle interpolation on a circular domain, which is known to have artefacts such as the *gimbal lock*. Using quaternions as in [Oikonomidis et al., 2011a,Sharp et al., 2015] can help to avoid this problem, but will also introduce new problems such as sign ambiguities in the interpolation. To avoid this we apply the following simple fix. In addition to the joint angles $\boldsymbol{\theta}_l$, we also use the 3D offset vector $\mathbf{j}_l$ from $\mathbf{x}_l$ to their

children positions $\mathbf{y}_l = \mathbf{x}_{l+1}$. In particular, each data point is labelled with a tuple $(\boldsymbol{\theta}_l, \mathbf{j}_l)$, where $\mathbf{j}_l = \text{vec}(\{y - \mathbf{x}_l : y \in \mathbf{y}_l\})$ indicates a vector of 3D offsets from the current evaluation point $\mathbf{x}_l$ to each of its children output positions in $\mathbf{x}_{l+1}$.

At training time, the 3D offset vector $\mathbf{j}$ are used to minimize the quality function (3.3). Since there are 12 nodes in the kinematic model (see Algorithm 8), we need to train 12 forests for an HSF. Algorithm 10 gives the training pseudocode for one tree with respect to one kinematic node k. When generating the training samples, input $\mathbf{x}_k$ is the position of the joint indicated by $k \rightarrow$ input. The output tuple $(\mathbf{j}_k, \boldsymbol{\theta}_k)$ is calculated given $k \rightarrow$ ouput.

Algorithm 10 Training a hierarchical sampling tree corresponding to a node in the kinematic tree model (Algorithm 8).

Require: A set of training samples $\mathcal{D} = \{(I, \boldsymbol{\theta})\}$, where I is a depth image and $\boldsymbol{\theta}$ is its parameter; A node k in the kinematic model; Maximum tree depth D.
Ensure: A Hierarchical Sampling Tree t_k.

procedure Train($\mathcal{D}$, k)
  Construct a training set $\mathcal{D}_k = \{(I, \mathbf{x}_k, \mathbf{j}_k, \boldsymbol{\theta}_k) | k, \boldsymbol{\theta}\}$ ▷ Generate the training set given k and $\boldsymbol{\theta}$ as in Sec. 3.3.
  Train a standard regression tree t_k with $\mathcal{D}_k$.
  for all leaf node $n \in t_k$ **do**
    $\mathcal{G} = \text{Fit}(\mathcal{D}_{kn})$. ▷ $\mathcal{D}_{kn}$ is the subset of $\mathcal{D}_k$ that arrives at n.
    Store $\mathcal{G}$ with n.
  end for
end procedure

function Fit($\mathcal{D}$)
  Fit a GMM $\mathcal{G}_{\mathbf{j}}$ to $\{\mathbf{j} | (\mathbf{j}, \boldsymbol{\theta}) \in \mathcal{D}\}$.
  ▷ Use the 3D offset $\mathbf{j}$ as a proxy to cluster sample tuples.
  for all component $c_{\mathbf{j}} \in \mathcal{G}_{\mathbf{j}}$ **do**
    Fit a Gaussian $c_{\boldsymbol{\theta}}$ to $\{\boldsymbol{\theta} | (\mathbf{j}, \boldsymbol{\theta}) \in c_{\mathbf{j}}\}$.
    ▷ Generate the actual GMM with samples that are close enough.
  end for
  Return $\mathcal{G}_{\boldsymbol{\theta}} = \{c_{\boldsymbol{\theta}}\}$.
end function

7.6.5 Generating Leaf Models

Likewise, we then use mean shift mode detection to cluster the 3D offsets (with a bandwidth 0.01m). After that data points of each cluster should be sufficiently close to each other so that interpolation on the angles in $\boldsymbol{\theta}_l$ is not a problem. This allows us to directly fit a GMM $\mathcal{G}_l$ to $\boldsymbol{\theta}_l$. In the case of a quaternion parameter, the sign of every $\boldsymbol{\theta}_l$ can be checked against the mean of its cluster, in order to avoid any sign ambiguities. As these 3D offset vectors were only used as a proxy for the true rotational parameters of interest during training, we can directly sample joint angles from $\mathcal{G}_l$ at test time without worrying about them.

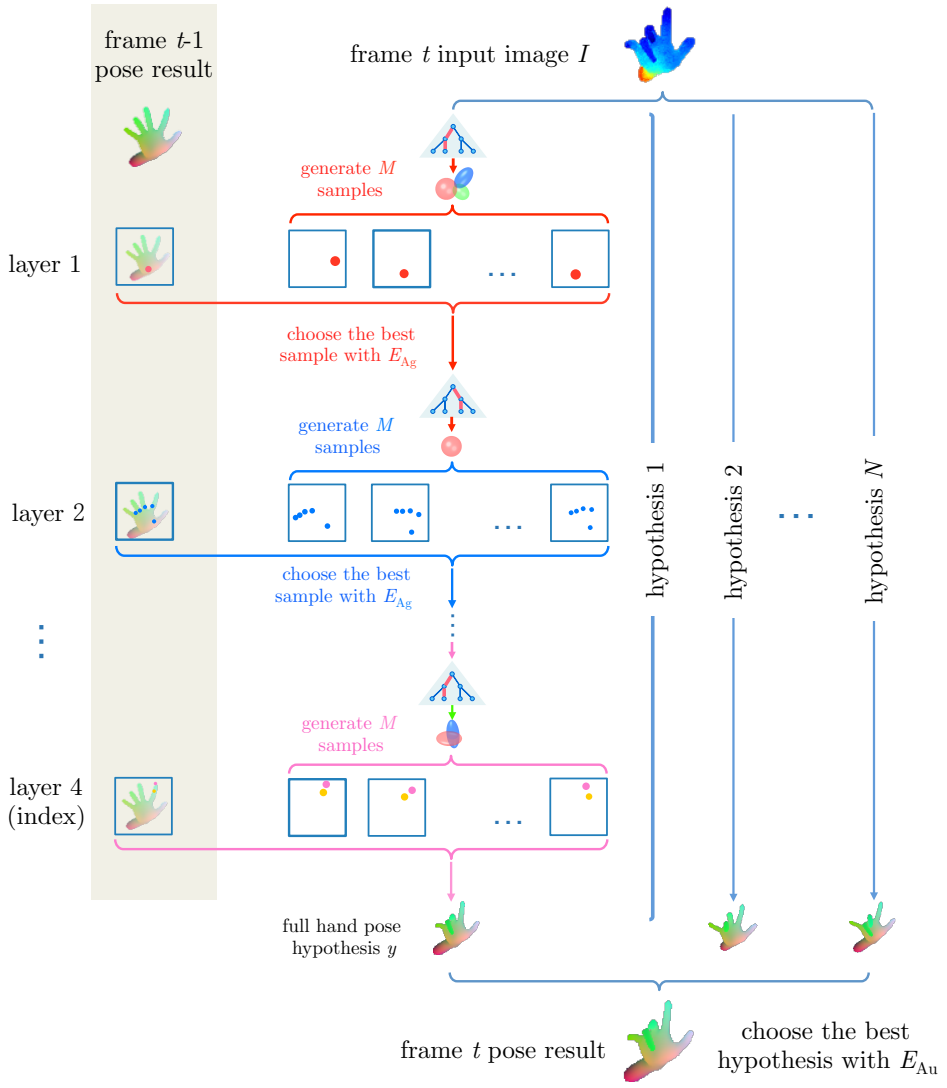


Figure 7.7: Hierarchical sampling optimization. At each layer, the input data $\mathbf{x}$ is mapped to a distribution $\mathcal{G}$, where M samples are drawn and selected. After four layers, we have one full pose hypothesis. This process is repeated N times to have N hypotheses. Finally the golden energy is applied to choose the best hypothesis as output.

7.7 Experiments

We conduct our experiments on 3 publicly available datasets: Microsoft Research Synthetic Hand Dataset (MSHD) [Sharp et al., 2015], ICVL v2 [Tang et al., 2014] and NYU [Tompson et al., 2014] , for they have spanned a wide range of perspectives (see Table 4.1). Note that the ICVL v2 and NYU datasets only provide 3D joint locations as label, whereas our method require joint angles for training. To tackle this, inspired by the inverse kinematic step in [Tompson et al., 2014], we use PSO to fit their ground-truth joint locations and recover angles. All experiments are conducted with Intel i7, 32GB RAM and an NVidia Quadro K600.

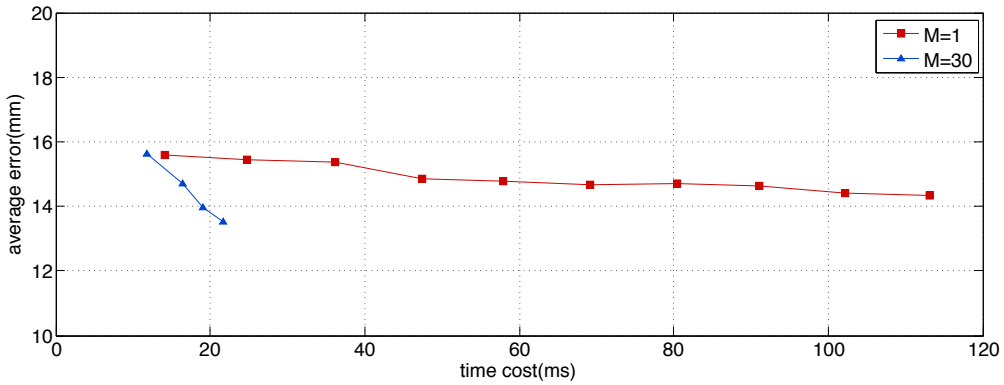
7.7.1 Self-comparison

To analyse the contributions of our method, a set of self-comparison experiments are conducted on the ICVL v2 dataset [Tang et al., 2014] for its relatively compact size. Each forest is empirically trained with 3 trees and maximum depth of 15.

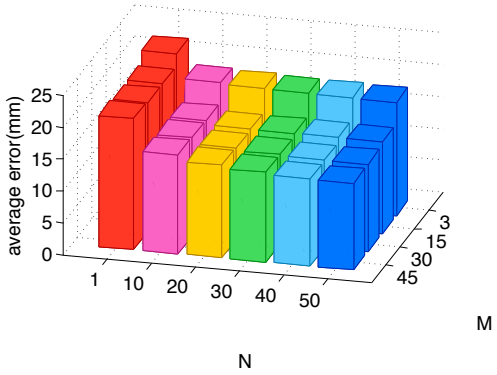
We first conduct a experiment to reveal the impact of of hierarchical optimization, by comparison between these two baselines: $M = 1$ (i.e. no per-joint optimization) and $M = 30$. To analyse the boost in both accuracy and efficiency, we vary N to obtain a curve showing the trade-off between them (Fig. 7.8a). For $M = 1$, N is varied from 100 to 1000, with a step size of 100; for $M = 30$, N can be much smaller. So we choose N from 5 to 20 with step size of 5. As Fig. 7.8a shows, the average error drops more drastically when $M = 30$, without increasing too much of time budget. From the accuracy prospective, to achieve the same error of 14.5mm, $M = 30$ only requires 17% time cost of $M = 1$.

Our second attempt is to perform a parameter analysis on M and N. To uniformly sample from all trees, M is always set to multiple of $T = 3$. As in Fig. 7.8b, the error

reduces $2mm$ and $4mm$ for the first step of N and M respectively. However, once they are combined ($N = 10, M = 15$), accuracy is improved by $10mm$. After $N = 40$ and $M = 30$, the error is still decreasing but converged.



(a) Comparing with and without per-joint optimization



(b) Analysis on M and N

Figure 7.8: Self comparison on the ICVL v2 dataset [Tang et al., 2014].

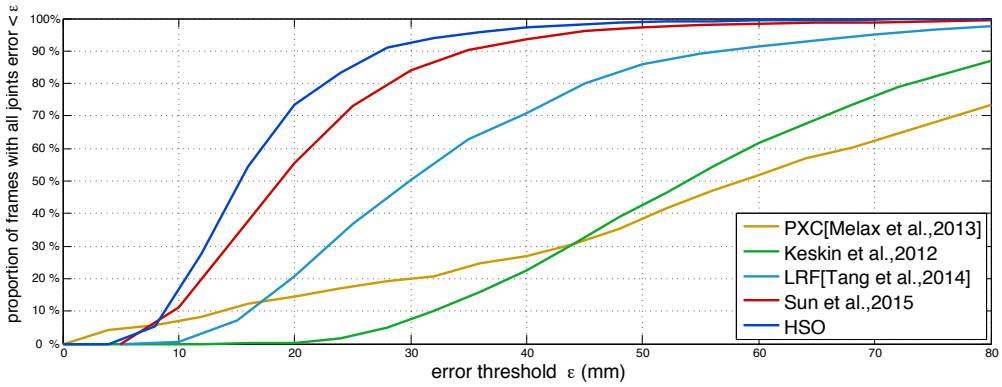


Figure 7.9: Comparing with State-of-the-arts on ICVL v2 dataset [Tang et al., 2014].

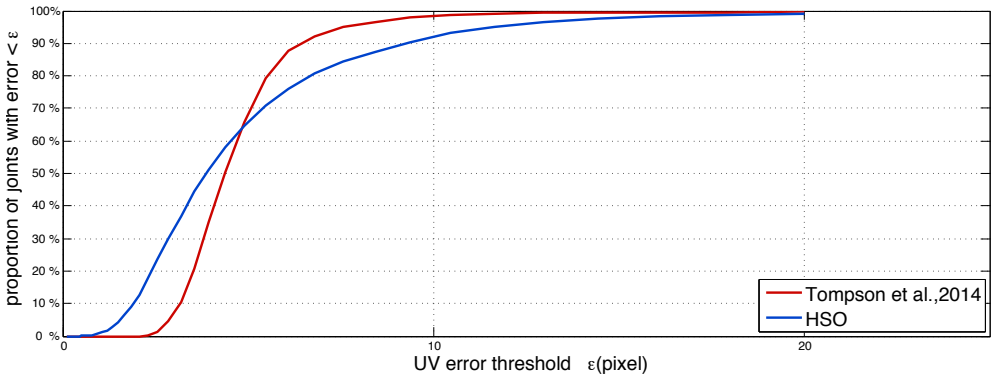
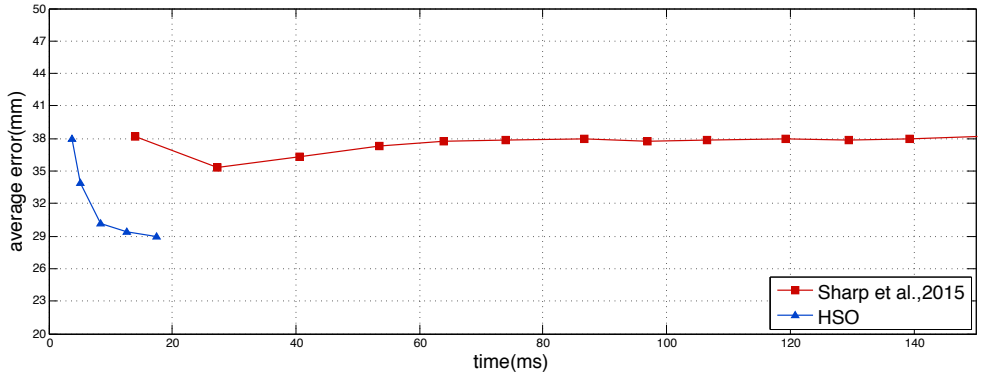


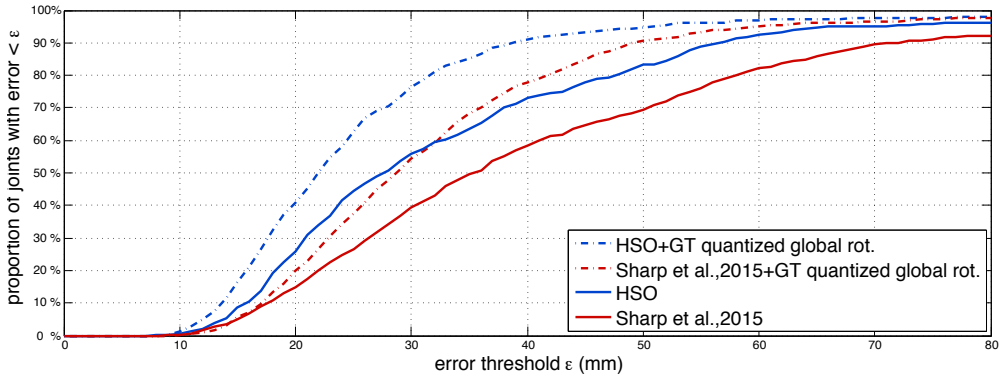
Figure 7.10: Results on NYU dataset [Tompson et al., 2014]. We performed this comparison in pixels because [Tompson et al., 2014] only provide 2D results.

7.7.2 Comparison with Prior Work

We then move on to compare HSO with 6 State-of-the-art methods including [Sharp et al., 2015], [Sun et al., 2015], [Tompson et al., 2014], LRF [Tang et al., 2014], [Keskin et al., 2012] and Intel® Perceptual Computing(PXC) [Melax et al., 2013]. These methods



(a) Comparing the accuracy and efficiency trade-off with [Sharp et al., 2015]



(b) Comparing with the discriminative part of [Sharp et al., 2015]

Figure 7.11: Results on MSHD dataset [Sharp et al., 2015].

cover a wide spectrum of sensors, perspectives of challenges and different methods from decision forest to deep neural networks. All their results are either obtained from the authors or implementations that appear in previous literature.

The ICVL v2 dataset consists of 2 sequences that with fast abrupt gestures, which often causes tracking to fail. We adopt the maximum allowed error as metric ϵ [Taylor et al., 2012]. All methods except PXC are individual-frame based. PXC is tracking-

based, but with a open hand pose detector as a reinitialiser, which explains why its curve shows it is particularly good at some poses but performs poorly on the overall sequences. Sun et al., due to its hierarchical structure and cascaded regressor, outperforms its prior arts. But HSO is able to further improve by about 15% of frames when $\epsilon = 20mm$.

On the NYU dataset we compare HSO with [Tompson et al., 2014] that based on Convolutional Neural Networks. Following their setting, the evaluation is done in UV space. Their result contains 14 keypoint locations. To be able to compare fairly, we find a common subset of 12 joint locations to compare (remove palm center and the hamate bone position). The slow hand movements is in favour for tracking-based methods like [Tompson et al., 2014]. Different from their approach, we incorporate temporal information hierarchically as described in Section 7.6.1. As shown in Fig. 7.10, our method performs better in most cases. However, due to the structured light sensor it uses, the sequences contain relatively high noise and sometimes significantly portion of missing pixels. This makes the silver energy fail and degrades our accuracy on the difficult cases, as shown in the failed cases of Fig. 7.12.

Finally we compare with [Sharp et al., 2015] on their MSHD dataset. Despite with synthetic data, this dataset is particularly challenging, since it exhibits full range of global rotations. As described in [Sharp et al., 2015], such a diverse data distribution requires very deep decision trees to have satisfying accuracy, which leads to considerable memory consumption. Following their solution, we utilize the same rotation space classifier (obtained from the authors) to quantize the space into 128 clusters, and train an hierarchical sampling forests (HSF) per each. Moreover, on some global rotations, for instance, egocentric view, the hand can be occluded by the forearm. Thus we have an extra step to randomize forearm positions and use golden energy to choose the best one.

Firstly, conditioned on ground-truth global rotation results, we compare HSO and

[Sharp et al., 2015]. In order to vary time cost, we change N of HSO, and PSO generations in [Sharp et al., 2015]. Note that to compare fairly, the starting pose in [Sharp et al., 2015] is discarded, so that both methods do not utilize any temporal information. As a result PSO becomes very difficult to converge just with particles from their discriminative part. And HSO outperforms in both accuracy and efficiency, as shown in Fig. 7.11a.

And then we replace the PSO part in [Sharp et al., 2015] with the same golden energy selector in our case, just to compare the discriminative parts. Here we consider two cases: using ground-truth global rotation cluster, or being probabilistically conditioned on the results of the global rotation classifier, as in [Sharp et al., 2015]. Again as shown in Fig. 7.11b, HSO substantially outperforms [Sharp et al., 2015] in both cases. Furthermore, the fact that the curves using groundtruth global rotation cluster is much better, indicates that the global rotation classifier (random ferns in this case), has hindered accuracy of the whole pipeline.

In terms of efficiency, from Fig. 7.8a and Fig. 7.11a we can gather that HSO runs 50fps with satisfying accuracy.

Qualitative results are demonstrated in Fig. 7.12 and Fig. 7.13. Despite the quantitative proof of reduction in error accumulation, the 5^{th} row of MSHD results gives an example of wrong palm orientation estimation (ground-truth is facing up whilst prediction is facing down), and inevitably all the rest joint predictions are wrong. Also, failed cases with NYU dataset shows how missing pixels affects the predictions. Finally Fig. 7.14 shows a few screen shots from a recorded demo video showing the real-time performance.

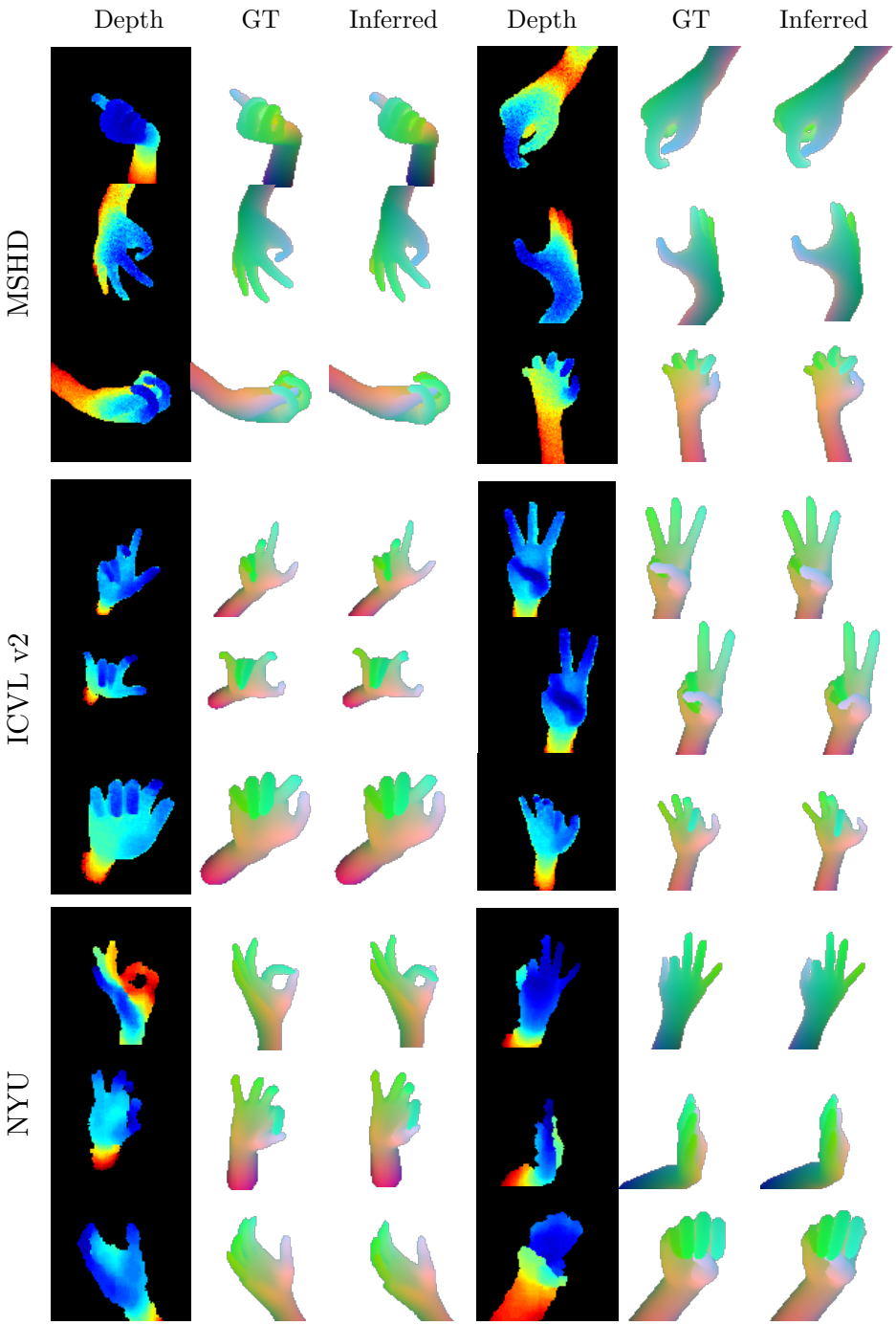


Figure 7.12: Success cases from MSHD [Sharp et al., 2015], ICVL v2 [Tang et al., 2014] and NYU datasets [Tompson et al., 2014].

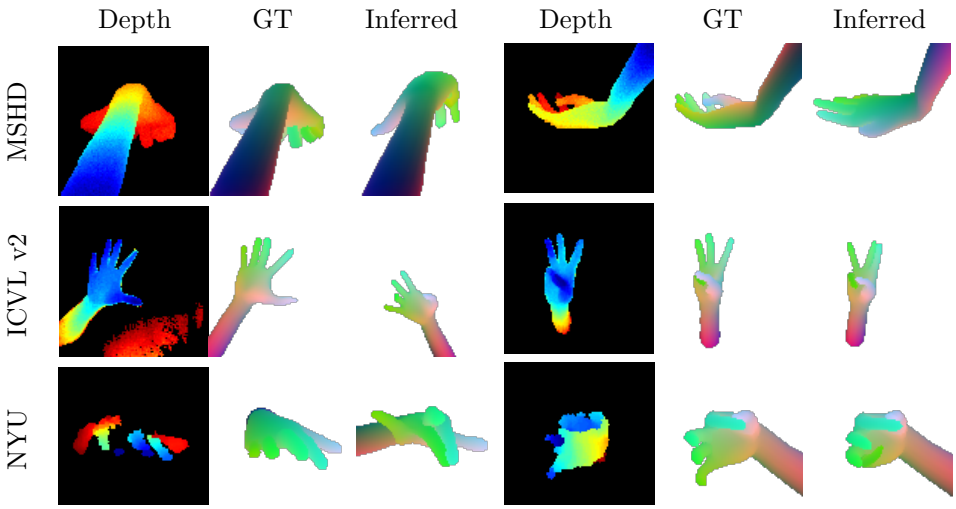


Figure 7.13: Failure cases due to difficult angle, unclean segment and sensor noise, etc.

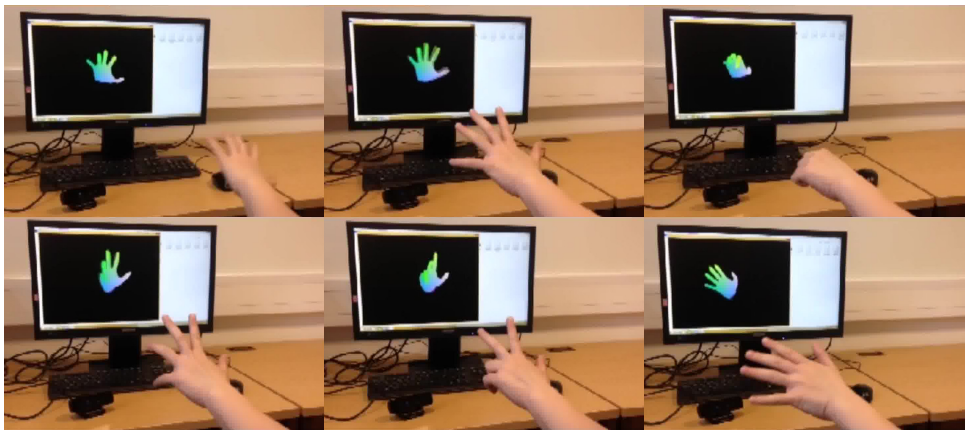


Figure 7.14: Screen shots from the demo video that shows real-time performance.

7.8 Summary

The golden energy, as introduced by [Oikonomidis et al., 2011a], is in many senses the ideal energy to evaluate the fit of a model to input depth data. The idea is simple, just render and compare. Unfortunately, this energy is notoriously hard to optimize, with elusive gradients and many local minima, making traditional black-box optimization techniques extremely brittle and inefficient. Indeed, [Sharp et al., 2015] go to considerable lengths engineering a system at a very low level in order to make black optimization work at real-time, albeit at the expense of complete saturation of a top of the line GPU.

In contrast, our work recognizes the fact that we are solving similar instances of the same problem repeatedly, and thus some knowledge of the problem domain can, and almost surely should be applied to the optimization procedure. In particular, we exploit the fact that subsets of the parameters have subtle relations that can be tested for, via our silver energy, and exploited to drastically reduce the number of hypotheses tested by the more expensive golden energy. This allows us to achieve state of the art results on markedly reduced computational budgets.

Regardless, we recognize the fact that there is more work to be done, and are optimistic by the fact that this method can be so easily modified by replacing the energies and regressors used. For example, although the golden energy is extremely well motivated from a modelling perspective, there continues to be a model mismatch due to inaccuracies in our hand model and non-Gaussian sensor noise. As we ultimately want to minimize the prediction of joint angles, which we might call the platinum energy, while being robust to these model defects we might consider actually learning a mapping from the depth image, rendered image pair to the platinum energy value.

In addition, there are many complementary and promising ideas in the literature.

For example, we could try to explicitly encourage the samples from each layer to be diverse, as is done in [Guzman-Rivera et al., 2014, Kulesza and Taskar, 2010, Park and Ramanan, 2011], in order to increase the chances of finding a good sample. Also the cascaded regressors in [Sun et al., 2015] can be naturally incorporated into our hierarchical framework and thus might further improve the accuracy.

CHAPTER 8

Conclusion and Future Works

Contents

8.1 Achievements

This thesis investigates the problem of 3D hand pose regression. In this chapter, we summarise our achievements on this topic so far, analyse our findings and discuss future directions.

Chapter 1 started by giving a definition and limiting the scope to a single hand pose regression using 3D depth data. A few application examples were listed to depict the

usage scenarios of this topic. And then the anatomical topology of human hand was analysed, laying down basic concepts and notations for following chapters. After that the challenges of this particular problem were discussed, which provided motivations for the contributions of this thesis.

Chapter 2 categorised and discussed existing literatures. Starting from a closely related topic, 3D human body pose, prior arts were classified into discriminative, generative and hybrid methods. From the perspective of input data, holistic and patch-based methods were also reviewed. Following that, literatures of 3D hand pose regression were also categorised in a similar way.

To be self-contained, in Chapter 3, we briefly discussed the reasons of choosing decision forest as our discriminative classifier. And then we reviewed the concept of DF, formulating its properties including feature, quality function, leaf predictor and tree structure, which defined basic notations. The standard training and testing algorithms were also given as templates for the following chapters.

In Chapter 4, the datasets used in this thesis, as well as evaluation criterion were introduced. Firstly we presented two of our own datasets, ICVL v1 and v2, along with their collection and annotation process. And then two more public datasets, NYU and MSHD were reviewed. And then Frame-based and set-based evaluation protocols were briefly introduced.

Chapter 5 proposed a novel discriminative framework called semi-supervised transductive regression forest, targeting the discrepancy between synthetic and real data, as well as the high dimension in pose regression. Synthetic data has been used by many existing hand pose estimation systems for training. The proposed STR forest captures the benefits of both real and synthetic data via transductive learning. The STR forest learns the implicit relationship between a small, sparsely labelled real dataset and a large synthetic dataset with generated ground truths. Besides, a data- driven

technique was also proposed to recover noisy and self-occluded joints. Experiments demonstrated not only the promising performance of this approach with respect to noise and self-occlusions, but also its superiority in accuracy, robustness and speed over existing methods.

Chapter 6 designed a new regression algorithm called latent regression forest, targeting the testing efficiency of patch-based methods such as STR forest. By representing the hand topology as an latent tree model, this method seamlessly combined the benefits of both holistic and patch-based methods, having the efficiency of holistic whilst keeping the flexibility of patch-based. Furthermore, an unsupervised learning method was adopted to automatically learn the structure of LTM. Experiments showed that it was 8 times faster than existing patch-based method whilst keeping comparable accuracy.

Chapter 7 aimed to resolve the drawbacks of LRF. By decomposing the pose regression problem into 4 layers according to the kinematic structure, a discriminative algorithm called hierarchical sampling forests was proposed to regress from partial poses to full poses, in a progressive manner. Moreover, an efficient energy function called 'silver energy' was designed to verify partial poses, such that invalid hypotheses can be discarded at early stage. With this idea, we showed superior accuracy and efficiency to 6 state-of-the-art methods on 3 public datasets.

8.2 Findings

STR forest utilises Transductive learning to make use of both synthetic and real data, especially unlabelled real data to bridge the gap caused mainly by sensor noises. However, as short-range ToF sensors prevails, noises have been significantly reduced. Also as more generative methods are available, the efforts of labelling real data are much

less. Nonetheless, we argue this idea can be applied to many other cross-modality cases, for instance, bridging the gap between T1 and T2-weighted MRI images [Bronstein et al., 2011].

STR forest provides a unified framework to adaptively switch among viewpoint classification, finger part classification and pose regression. It does reduce the training complexity by allowing simpler quality functions at upper part of the forest, taking advantage of the fact that the upper splits of DF do not need to be very accurate. However, the testing complexity of a tree-like algorithm is given by the average tree depth, which is highly correlated to the amount and diversity of training samples. Comparing to previous patch-based regression algorithms [Girshick et al., 2011, Keskin et al., 2012], STR forest does not improve this part.

LRF, on the other hand, has significantly improved the efficiency over STR forest. But it has a few limitations. (1) Starting from holistic makes it LRF vulnerable for occlusions. (2) Although globally the results are constrained by the topology, locally it is not guaranteed to be kinematically correct. (3) Its coarse-to-fine design is not compatible to angular form of parameters. (4) Due to the latent structure, intermediate results are not observable and hence can not be verified easily.

Rather than a coarse-to-fine search as in LRF, HSF approaches this problem by taking a progressive way. The benefits of doing so are three-fold: (1) While does not start from holistic sample, theoretically it may be more robust in terms of occlusion. (2) All intermediate results are part of final poses. Hence can be verified using the silver energy. (3) Angular form of parameters is allowed in this way. Consequently we can render the full pose hypotheses and verify them with the golden energy. (4) For the discriminative part, it maintains the low complexity property of LRF over STR forest. However, the golden energy verification still requires rendering and thus trades efficiency for accuracy.

While it is obvious that LRF and HSF are superior than STR forest, can we conclude that HSF is better than LRF? For this specific problem of 3D body/hand pose regression, which we have the prior knowledge of its kinematic structure, the answer is yes. However, in some other scenarios, when the underlying spatial structure of target is unknown, or even deformable, there is still an ample scope for LRF. Furthermore, if we move on to the temporal domain, e.g. , action recognition, LRF and HSF have their own areas of interest. Ultimately, LRF is coarse-to-fine and thus efficient when processing videos in an offline manner, whilst HSF is a progressive search and therefore suits for online action recognition.

8.3 Future Ideas

Although we have achieved state-of-the-art performance, there is still plenty of room to improve the performance for this 3D hand pose regression problem. A few ideas for future work is listed below:

Discriminative

With LRF, currently it is applied in a top-down, coarse-to-fine manner. We could also try the opposite way, i.e. , bottom-up. This way it becomes more robust against occlusion by staring from patch-based results and using holistic as verification. Moreover, it is possible to combine both top-down and bottom-up, possibly iteratively to further improve the accuracy. However, we need a smart way of combining in order to avoid losing efficiency, which is unclear at the moment.

HSF derives from the multiple output framework. It makes use of the sampling trick to improve accuracy. However, when training the forests, one aspect of multi-output, diversity, is not considered. One thing we could try is to explicitly encourage

the samples from each layer to be diverse by training different sets of forests with diverse samples, as is done in [Guzman-Rivera et al., 2014,Kulesza and Taskar, 2010,Park and Ramanan, 2011]. Or to go one step further, we could train only one set of forests, and create different diverse leaf models, in order to save memory and improve both training and testing efficiency.

From a graphical model point of view, current version of HSF utilises only one kinematic structure, going from the wrist to finger tips, which was decided intuitively. However, different types of structure, for instance, decomposing by fingers, might worth exploring. Bi-directional sweeps, passing messages from wrist to tips and then back, could also refine the results. Considering more contextual informations from already predicted joints encodes an implicit collision term, which is absent in the current Markov structure.

Within each layer, we could also apply the cascaded regressors in [Sun et al., 2015] to improve the quality of partial pose samples.

Deep learning, in particular CNN, has been applied to the 3D hand pose regression problem in [Tompson et al., 2014]. Although it has achieve good performance on the NYU dataset, it is worthy to conduct more thorough comparative experiments with other public datasets. Similar to LRF, a modality-aware CNN has been applied to face recognition in our previous work [Xiong et al., 2015], which could probably be applied to this problem.

Generative

Although the golden energy is the best at hand, model mismatch and non-gaussian sensor noise are degrading its effectiveness. As in [Taylor et al., 2014], we could improve the optimisation by learning a mapping from the depth image, rendered image pair to the platinum energy value.

We could also try decomposing PSO into a hierarchy, which has been applied in the field of human pose estimation [Ivekovič et al., 2008, John et al., 2010]. However, the golden energy, which requires full pose does not work in this case. Hence a more suitable choice is to apply the silver energy.

Hybrid

The common way of combining a discriminative and a generative method, as in [Sharp et al., 2015], is to create an interface between them, passing full pose as parameters. However, since we have already decompose the discriminative part into multiple layers, it is natural to combine a hierarchical discriminative methods HSF with a hierachical generative method [Ivekovič et al., 2008, John et al., 2010], such that we could refine the partial pose results in an evolutionary way, rather than sampling.

Interaction

From an application point of view, multiple-hand interaction and hand-object interaction are definitely interesting. The challenges behind are occlusion and more introduced DoF. One way to handle occlusion with patch-based methods is to estimate a probabilistic foreground mask, as in our previous work on rigid object pose regression [Tejani et al., 2014]. Other work on these problems include [Oikonomidis et al., 2011b, Oikonomidis et al., 2012, Tzionas et al., 2014, Tzionas and Gall, 2015], which could potentially be integrated with our algorithms in future work.

Bibliography

[Athitsos and Sclaroff, 2003] Athitsos, V. and Sclaroff, S. (2003). Estimating 3D hand pose from a cluttered image. *IEEE Conference on Computer Vision and Pattern Recognition*.

[Baak et al., 2011] Baak, A., Müller, M., Bharaj, G., Seidel, H.-P., and Theobalt, C. (2011). A data-driven approach for real-time full body pose reconstruction from a depth camera. In *International Conference on Computer Vision*.

[Baak et al., 2013] Baak, A., Müller, M., Bharaj, G., Seidel, H.-P., and Theobalt, C. (2013). A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Consumer Depth Cameras for Computer Vision*, pages 71–98. Springer.

[Ballan et al., 2012] Ballan, L., Taneja, A., Gall, J., Van Gool, L., and Pollefeys, M. (2012). Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*.

[Batra et al., 2012] Batra, D., Yadollahpour, P., Guzman-Rivera, A., and Shakhnarovich, G. (2012). Diverse m-best solutions in markov random fields. In *European Conference on Computer Vision*, pages 1–16. Springer.

[Begum and Karray, 2011] Begum, M. and Karray, F. (2011). Visual attention for robotic cognition: a survey. In *IEEE Trans. on Autonumous mental development*.

[Blanz and Vetter, 1999] Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces. In *SIGGRAPH*.

[Bonde et al., 2010] Bonde, U. D., Kim, T.-K., and Ramakrishnan, K. (2010). Randomised manifold forests for principal angle-based face recognition. In *Asian Conference on Computer Vision*, pages 228–242. Springer.

[Boser et al., 1992] Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Conference on Learning Theory*.

[Bourdev and Malik, 2009] Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3D human pose annotations. In *International Conference on Computer Vision*, pages 1365–1372. IEEE.

[Boussemart et al., 2004] Boussemart, Y., Rioux, F., Rudzicz, F., Wozniewski, M., and Cooperstock, J. R. (2004). A framework for 3D visualisation and manipulation in an immersive space using an untethered bimanual gestural interface. In *Virtual Reality Software and Technology*.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*.

[Breiman et al., 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). Classification and regression trees. In *Wadsworth*.

[Bronstein et al., 2011] Bronstein, M. M., Bronstein, E. M., Michel, F., and Paragios, N. (2011). Data fusion through crossmodality metric learning using similaritysensitive hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Broomhead and Lowe, 1988] Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document.

[Cai et al., 2015] Cai, M., Kitani, K. M., and Sato, Y. (2015). Hand grasp recognition from egocentric videos. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

[Cao et al., 2012] Cao, X., Wei, Y., Wen, F., and Sun, J. (2012). Face alignment by explicit shape regression. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Çeliktutan et al., 2013] Çeliktutan, O., Ulukaya, S., and Sankur, B. (2013). A comparative study of face landmarking techniques. *EURASIP J. Image and Video Processing*, 2013:13.

[Chang et al., 2015] Chang, H. J., Garcia-Hernando, G., Tang, D., and Kim, T.-K. (2015). Spatio-temporal hough forest for efficient detection-localisation-recognition of fingerwriting in egocentric camera. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

[Chen et al., 2003] Chen, F.-S., Fu, C.-M., and Huang, C.-L. (2003). Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21:745–758.

[Cheng, 1995] Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799.

[Choi et al., 2010] Choi, M. J., Lim, J. J., Torralba, A., and Willsky, A. S. (2010). Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Choi et al., 2011] Choi, M. J., Tan, V. Y. F., Anandkumar, A., and Willsky, A. S. (2011). Learning latent tree graphical models. *JMLR*, 12:1771–1812.

[Chua et al., 2002] Chua, C.-S., Guan, H., and Ho, Y.-K. (2002). Model-based 3D hand posture estimation from a single 2D image. *Image and Vision Computing*.

[Chum and Zisserman, 2007] Chum, O. and Zisserman, A. (2007). An exemplar model for learning object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Cootes et al., 1998] Cootes, T. F., Edwards, G. J., and Taylor, C. J. (1998). Active appearance models. In *European Conference on Computer Vision*.

[Criminisi and Shotton, 2013] Criminisi, A. and Shotton, J. (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Springer.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Dantone et al., 2012] Dantone, M., Gall, J., Fanelli, G., and Van Gool, L. (2012). Real-time facial feature detection using conditional regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[de Angulo and Torras, 2008] de Angulo, V. R. and Torras, C. (2008). Learning inverse kinematics: Reduced sampling through decomposition into virtual robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 38:1571–1577.

[de La Gorce et al., 2011] de La Gorce, M., Fleet, D., and Paragios, N. (2011). Model-based 3D hand pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Delgado et al., 2014] Delgado, M. F., Cernadas, E., Barro, S., and Amorim, D. G. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181.

[Deutscher and Reid, 2005] Deutscher, J. and Reid, I. (2005). Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Workshop on Multiple Classifier Systems*.

[Dollár and Perona, 2010] Dollár, P. and Perona, P. (2010). The fastest pedestrian detector in the west. In *British Machine Vision Conference*.

[Dollár et al., 2009] Dollár, P., Tu, Z., Perona, P., and Belongie, S. (2009). Integral channel features. In *British Machine Vision Conference*.

[DSouza et al., 2001] DSouza, A., Vijayakumar, S., and Schaal, S. (2001). Learning inverse kinematics. In *International Conference on Intelligent Robots and Systems*.

[Edwards et al., 1998] Edwards, G. J., Taylor, C. J., and Cootes, T. F. (1998). Interpreting face images using active appearance models. In *Automatic Face and Gesture Recognition, IEEE International Conference on*.

[Erol et al., 2007] Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. (2007). Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*.

[Felzenszwalb et al., 2010] Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

[Flerova et al., 2012] Flerova, N., Rollon, E., and Dechter, R. (2012). Bucket and mini-bucket schemes for m best solutions over graphical models. In *Graph Structures for Knowledge Representation and Reasoning*, pages 91–118. Springer.

[Forsyth et al., 2005] Forsyth, D. A., Arikan, O., Ikemoto, L., O'Brien, J., and Ramanan, D. (2005). Computational studies of human motion: part 1, tracking and motion synthesis. *Foundations and Trends® in Computer Graphics and Vision*, 1(2-3):77–254.

[Friedman, 1989] Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[Fromer and Globerson, 2009] Fromer, M. and Globerson, A. (2009). An lp view of the m-best map problem. In *Advances in Neural Information Processing Systems*, pages 567–575.

[Gall and Lempitsky, 2009] Gall, J. and Lempitsky, V. (2009). Class-specific hough forests for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Gall et al., 2010] Gall, J., Rosenhahn, B., Brox, T., and Seidel, H.-P. (2010). Optimization and filtering for human motion capture. *International journal of computer vision*, 87(1-2):75–92.

[Gall et al., 2011] Gall, J., Yao, A., Razavi, N., Van Gool, L., and Lempitsky, V. (2011). Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Gelenbe, 2010] Gelenbe, E. (2010). Search in unknown random environments. In *Phys Rev E Stat Nonlin Soft Matter*.

[Girshick et al., 2011] Girshick, R., Shotton, J., Kohli, P., Criminisi, A., and Fitzgibbon, A. (2011). Efficient regression of general-activity human poses from depth images. *International Conference on Computer Vision*.

[Gross et al., 2004] Gross, R., Matthews, I. A., and Baker, S. (2004). Generic vs. person specific active appearance models. In *British Machine Vision Conference*.

[Grossmann, 2004] Grossmann, E. (2004). Adatree: Boosting a weak classifier into a decision tree. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*.

[Guan et al., 2006] Guan, H., Chang, J. S., Chen, L., Feris, R., and Turk, M. (2006). Multi-view appearance-based 3D hand pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

[Guzman-Rivera et al., 2012] Guzman-Rivera, A., Batra, D., and Kohli, P. (2012). Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems*, pages 1799–1807.

[Guzman-Rivera et al., 2014] Guzman-Rivera, A., Kohli, P., Glocker, B., Shotton, J., Sharp, T., Fitzgibbon, A., and Izadi, S. (2014). Multi-output learning for camera relocalization. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

[Hamer et al., 2009] Hamer, H., Schindler, K., Koller-Meier, E., and Van Gool, L. (2009). Tracking a hand manipulating an object. In *International Conference on Computer Vision*.

[Hammer and Gersmann, 2003] Hammer, B. and Gersmann, K. (2003). A note on the universal approximation capability of support vector machines. *Neural Processing Letters*, 17(1):43–53.

[Hansard et al., 2012] Hansard, M., Lee, S., Choi, O., and Horaud, R. P. (2012). *Time-of-flight cameras: principles, methods and applications*. Springer Science & Business Media.

[Harmeling and Williams, 2011] Harmeling, S. and Williams, C. K. I. (2011). Greedy learning of binary latent trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1087–1097.

[Hirsch et al., 2009] Hirsch, M., Lanman, D., Holtzman, H., and Raskar, R. (2009). BiDi screen: A thin, depth-sensing LCD for 3D interaction using lights fields. *ACM Transactions on Graphics*.

[Ionescu et al., 2011] Ionescu, C., Li, F., and Sminchisescu, C. (2011). Latent structured models for human pose estimation. In *International Conference on Computer Vision*, pages 2220–2227.

[Ivekovič et al., 2008] Ivekovič, Š., Trucco, E., and Petillot, Y. R. (2008). Human body pose estimation with particle swarm optimisation. *Evolutionary Computation*, 16(4):509–528.

[Jacob et al., 2015] Jacob, Y., Manitsaris, S., Moutarde, F., Lele, G., and Pradere, L. (2015). Hand gesture recognition for driver vehicle interaction.

[Jang et al., 2015] Jang, Y., Noh, S.-T., Chang, H. J., Kim, T.-K., and Woo, W. (2015). 3D finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *Visualization and Computer Graphics, IEEE Transactions on*.

[John et al., 2010] John, V., Trucco, E., and Ivekovic, S. (2010). Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530–1547.

[Keskin et al., 2011] Keskin, c., Kirac, F., Kara, Y., and Akarun, L. (2011). Real time hand pose estimation using depth sensors. In *International Conference on Computer Vision Workshops*.

[Keskin et al., 2012] Keskin, c., Kirac, F., Kara, Y. E., and Akarun, L. (2012). Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision*.

[Khamis et al., 2015] Khamis, S., Taylor, J., Shotton, J., Çem Keskin, Izadi, S., and Fitzgibbon, A. (2015). Learning an efficient model of hand shape variation from depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Kim et al., 2011] Kim, T.-K., Budvytis, I., and Cipolla, R. (2011). Making a shallow network deep: Conversion of a boosting classifier into a decision tree by boolean optimisation. *International Journal of Computer Vision*, pages 1–13.

[Kontschieder et al., 2013] Kontschieder, P., Kohli, P., Shotton, J., and Criminisi, A. (2013). Geof: Geodesic forests for learning coupled predictors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 65–72. IEEE.

[Kulesza and Taskar, 2010] Kulesza, A. and Taskar, B. (2010). Structured determinantal point processes. In *NIPS*.

[L. Itti, 2001] L. Itti, C. K. (2001). *Computational Modelling of Visual Attention*.

[Lampert et al., 2008] Lampert, C., Blaschko, M., and Hofmann, T. (2008). Beyond sliding windows: object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Lawler, 1972] Lawler, E. L. (1972). A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management science*, 18(7):401–405.

[Lee and Kunii, 1993] Lee, J. and Kunii, T. L. (1993). Constraint-based hand animation. In *Models and techniques in computer animation*, pages 110–127. Springer.

[Lee et al., 2009] Lee, S.-J., Kong, Y.-K., Lowe, B. D., and Song, S. (2009). Handle grip span for optimising finger-specific force capability as a function of hand size. *Ergonomics*, 52(5):601–608.

[Leistner et al., 2009] Leistner, C., Saffari, A., Santner, J., and Bischof, H. (2009). Semi-supervised random forests. In *International Conference on Computer Vision*.

[Lepetit et al., 2005] Lepetit, V., Lagger, P., and Fua, P. (2005). Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition, IEEE Conference on*, volume 2, pages 775–781. IEEE.

[Lin et al., 2000] Lin, J., Wu, Y., and Huang, T. S. (2000). Modeling the constraints of human hand motion. In *Human Motion, 2000. Proceedings. Workshop on*, pages 121–126. IEEE.

[Liu, 2009] Liu, X. (2009). Discriminative face alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1941–1954.

[Loper and Black, 2014] Loper, M. M. and Black, M. J. (2014). OpenDR: An approximate differentiable renderer. In *ECCV*.

[McCarthy, 1991] McCarthy, J. M. (1991). Introduction to theoretical kinematics. *SIAM Review*, 33:302.

[Melax et al., 2013] Melax, S., Keselman, L., and Orsten, S. (2013). Dynamics based 3D skeletal hand tracking. In *I3D*.

[Mourad et al., 2013] Mourad, R., Sinoquet, C., Zhang, N. L., Liu, T., and Leray, P. (2013). A survey on latent tree models and applications. *JAIR*, 47:157–203.

[Neumann et al., 2010] Neumann, K., Rolf, M., Steil, J. J., and Gienger, M. (2010). Learning inverse kinematics for pose-constraint bi-manual movements. In *International Conference on Simulation of Adaptive Behavior*.

[Oikonomidis et al., 2011a] Oikonomidis, I., Kyriazis, N., and Argyros, A. (2011a). Efficient model-based 3D tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*.

[Oikonomidis et al., 2011b] Oikonomidis, I., Kyriazis, N., and Argyros, A. A. (2011b). Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *International Conference on Computer Vision*.

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

[Oyama et al., 2001] Oyama, E., Chong, N. Y., Agah, A., Maeda, T., and Tachi, S. (2001). Inverse kinematics learning by modular architecture neural networks with performance prediction networks. In *International Conference on Robotics and Automation*.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *TKDE*.

[Park and Ramanan, 2011] Park, D. and Ramanan, D. (2011). N-best maximal decoders for part models. In *ICCV*.

[Park and Sandberg, 1991] Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257.

[Pavlovic et al., 1997] Pavlovic, V., Sharma, R., Huang, T. S., et al. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):677–695.

[Peters and Itti, 2007] Peters, R. and Itti, L. (2007). Beyond bottom-up: Incorporating task-dependent influences into a computational model of spatial attention. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Pons-Moll et al., 2011] Pons-Moll, G., Baak, A., Gall, J., Leal-Taixe, L., Muller, M., Seidel, H.-P., and Rosenhahn, B. (2011). Outdoor human motion capture using inverse kinematics and von mises-fisher sampling. In *International Conference on Computer Vision*.

[Pyeatt et al., 2001] Pyeatt, L. D., Howe, A. E., et al. (2001). Decision tree function approximation in reinforcement learning. In *International Symposium on Adaptive Systems*, volume 1, page 2.

[Qian et al., 2014] Qian, C., Sun, X., Wei, Y., Tang, X., and Sun, J. (2014). Realtime and robust hand tracking from depth. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

[Ren et al., 2014] Ren, S., Cao, X., Wei, Y., and Sun, J. (2014). Face alignment at 3000 fps via regressing local binary features. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Riegler et al., 2015] Riegler, G., Ferstl, D., Rüther, M., and Bischof, H. (2015). A framework for articulated hand pose estimation and evaluation. In *Image Analysis*, pages 41–52. Springer.

[Romero et al., 2009] Romero, J., Kjellström, H., and Kragic, D. (2009). Monocular real-time 3D articulated hand pose estimation. In *Humanoids*.

[Rosales et al., 2001] Rosales, R., Athitsos, V., Sigal, L., and Sclaroff, S. (2001). 3D hand pose reconstruction using specialized mappings. In *International Conference on Computer Vision*.

[Roth, 1982] Roth, S. D. (1982). Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144.

[Saffari et al., 2009] Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009). On-line random forests. In *International Conference on Computer Vision Workshops*.

[Salzmann and Urtasun, 2010] Salzmann, M. and Urtasun, R. (2010). Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 647–654. IEEE.

[Saragih and Göcke, 2007] Saragih, J. M. and Göcke, R. (2007). A nonlinear discriminative approach to aam fitting. In *International Conference on Computer Vision*.

[Schölkopf et al., 2006] Schölkopf, B., Platt, J., and Hofmann, T. (2006). Fast discriminative visual codebooks using randomized clustering forests. pages 985–992.

[Shan et al., 2004] Shan, C., Wei, Y., Tan, T., and Ojardias, F. (2004). Real time hand tracking by combining particle filtering and mean shift. In *Automatic Face and Gesture Recognition, IEEE International Conference on*.

[Sharp et al., 2015] Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y., Freedman, D., Kohli, P., Krupka, E., Fitzgibbon, A., and Izadi, S. (2015). Accurate, robust, and flexible real-time hand tracking. In *SIGCHI*.

[Shotton et al., 2011] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Shotton et al., 2008] Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Sochman and Matas, 2005] Sochman, J. and Matas, J. (2005). Waldboost - learning for time constrained sequential detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Sommerlade and Reid, 2008] Sommerlade, E. and Reid, I. (2008). Information-theoretic active scene exploration. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Song et al., 2015] Song, J., Sörös, G., Pece, F., and Hilliges, O. (2015). Real-time hand gesture recognition on unmodified wearable devices.

[Sridhar et al., 2015] Sridhar, S., Mueller, F., Oulasvirta, A., and Theobalt, C. (2015). Fast and robust hand tracking using detection-guided optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*.

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

[Stenger et al., 2001] Stenger, B., MendonÃğa, P. R. S., and Cipolla, R. (2001). Model-based hand tracking using an unscented kalman filter. In *British Machine Vision Conference*.

[Stenger et al., 2006] Stenger, B., Thayananthan, A., Torr, P. H. S., and Cipolla, R. (2006). Model-based hand tracking using a hierarchical Bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Sturman and Zeltzer, 1994] Sturman, D. J. and Zeltzer, D. (1994). A survey of glove-based input. *Computer Graphics and Applications, IEEE*, 14(1):30–39.

[Sudderth et al., 2005] Sudderth, E. B., Torralba, A., Freeman, W. T., and Willsky, A. S. (2005). Learning hierarchical models of scenes, objects, and parts. In *International Conference on Computer Vision*.

[Sun et al., 2012] Sun, M., Kohli, P., and Shotton, J. (2012). Conditional regression forests for human pose estimation. *IEEE Conference on Computer Vision and Pattern Recognition*.

[Sun et al., 2015] Sun, X., Wei, Y., Liang, S., Tang, X., and Sun, J. (2015). Cascaded hand pose regression. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Sznitman and Jedynak, 2010] Sznitman, R. and Jedynak, B. (2010). Active testing for face detection and localisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Tang et al., 2014] Tang, D., Chang, H.-J., Tejani, A., and Kim, T.-K. (2014). Latent regression forest: Structured estimation of 3D hand posture. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Tang et al., 2015] Tang, D., Taylor, J., Kholi, P., Çem Keskin, Kim, T.-K., and Shotton, J. (2015). Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *International Conference on Computer Vision*.

[Tang et al., 2013] Tang, D., Yu, T.-H., and Kim, T.-K. (2013). Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *International Conference on Computer Vision*.

[Taylor et al., 2012] Taylor, J., Shotton, J., Sharp, T., and Fitzgibbon, A. (2012). The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Taylor et al., 2014] Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Hertzmann, A., and Fitzgibbon, A. (2014). User-specific hand modeling from monocular depth sequences. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 644–651. IEEE.

[Tejani et al., 2014] Tejani, A., Tang, D., Kouskouridas, R., and Kim, T.-K. (2014). Latent-class hough forests for 3D object detection and pose estimation. In *European Conference on Computer Vision*, pages 462–477. Springer.

[Tian et al., 2012] Tian, Y., Zitnick, C. L., and Narasimhan, S. G. (2012). Exploring the spatial hierarchy of mixture models for human pose estimation. In *European Conference on Computer Vision*, pages 256–269. Springer.

[Tompson et al., 2014] Tompson, J., Stein, M., LeCun, Y., and Perlin, K. (2014). Real-time continuous pose recovery of human hands using convolutional networks. In *Transactions on Graphics*.

[Torralba et al., 2007] Torralba, A., Murphy, K. P., and Freeman, W. T. (2007). Sharing visual features for multiclass and multiview object detection. 29(5):854–869.

[Tresadern et al., 2010] Tresadern, P. A., Sauer, P., and Cootes, T. F. (2010). Additive update predictors in active appearance models. In *BMVC*.

[Tzionas and Gall, 2015] Tzionas, D. and Gall, J. (2015). 3D object reconstruction from hand-object interactions. In *International Conference on Computer Vision*.

[Tzionas et al., 2014] Tzionas, D., Srikantha, A., Aponte, P., and Gall, J. (2014). Capturing hand motion with an rgb-d sensor, fusing a generative model with salient points. In *Pattern Recognition*, pages 277–289. Springer.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

[Vijayanarasimhan and Kapoor, 2010] Vijayanarasimhan, S. and Kapoor, A. (2010). Visual recognition and detection under bounded computational resources. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Viola and Jones, 2004] Viola, P. and Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision*.

[Vitter, 1985] Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57.

[Wahl and Kronmal, 1977] Wahl, P. W. and Kronmal, R. A. (1977). Discriminant functions when covariances are unequal and sample sizes are moderate. *Biometrics*, pages 479–484.

[Wang and Li, 2013] Wang, F. and Li, Y. (2013). Beyond physical connections: Tree models in human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Wang and Popović, 2009] Wang, R. Y. and Popović, J. (2009). Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*.

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Cluster boosted tree classifier for multi-view, multi-pose object detection. In *International Conference on Computer Vision*.

[Xiong et al., 2015] Xiong, C., Zhao, X., Tang, D., Yan, S., and Kim, T.-K. (2015). Conditional convolutional neural network for modality-aware face recognition. In *International Conference on Computer Vision*.

[Xiong and De la Torre, 2013] Xiong, X. and De la Torre, F. (2013). Supervised descent method and its applications to face alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*.

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

[Yang et al., 2015] Yang, H., Jia, X., Loy, C. C., and Robinson, P. (2015). An empirical study of recent face alignment methods. *arXiv:1511.05049*.

[Yang and Ramanan, 2011] Yang, Y. and Ramanan, D. (2011). Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1385–1392. IEEE.

[Yao et al., 2012] Yao, A., Gall, J., and Gool, L. (2012). Coupled action recognition and pose estimation from multiple views. *International Journal of Computer Vision*.

[Ye et al., 2011] Ye, M., Wang, X., Yang, R., Ren, L., and Pollefeys, M. (2011). Accurate 3D pose estimation from a single depth image. In *International Conference on Computer Vision*.

[Yu and Yang, 2001] Yu, H. and Yang, J. (2001). A direct lda algorithm for high-dimensional data - with application to face recognition. *Pattern Recognition*, 34:2067–2070.

[Yub Jung et al., 2015] Yub Jung, H., Lee, S., Seok Heo, Y., and Dong Yun, I. (2015). Random tree walk toward instantaneous 3D human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2467–2474.

[Zafrulla et al., 2011] Zafrulla, Z., Brashear, H., Starner, T., Hamilton, H., and Presti, P. (2011). American sign language recognition with the kinect. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, ICMI '11.

[Zhao et al., 2014] Zhao, X., Kim, T.-K., and Luo, W. (2014). Unified face analysis by iterative multi-output random forests. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Zhou, 2005] Zhou, S. (2005). A binary decision tree implementation of a boosted strong classifier. In *IEEE Workshop on Analysis and Modeling of Faces and Gestures*.

[Zhu et al., 2008a] Zhu, L. L., Lin, C., Huang, H., Chen, Y., and Yuille, A. (2008a). Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *European Conference on Computer Vision*, pages 759–773. Springer.

[Zhu et al., 2008b] Zhu, Y., Dariush, B., and Fujimura, K. (2008b). Controlled human pose estimation from depth image streams. In *2008*, pages 1–8.