Hand pose recognition using a consumer depth camera

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieurin
(abgekürzt: Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Inf. Alina Kuznetsova

geboren am 18.01.1989
in Raum Moskau, Russland (USSR)

2016

Referent:	Prof. Dr.-Ing. B. Rosenhahn
Korreferent:	Prof. Dr. A. Schilling
Vorsitzender:	Prof. Dr.-Ing. J. Ostermann
Tag der Promotion:	28.01.2016

Acknowledgement

Here I want to express endless gratitude to my adviser, Prof. Dr. Bodo Rosenhahn, for the freedom to pursue the research direction of my choice, as well as support, advice, encouragement, and a lot of provided opportunities. I would also like to thank Prof. Dr. Andreas Schilling and Prof. Dr. Joern Ostermann for being a part of my defense committee.

Special thanks to my Disney Research internship advisers, Dr. Leonid Sigal and Dr. Sung Ju Hwang, for the academic guidance and many things I learned during our joint work.

Thanks to all my institute colleagues, especially my office mate Roberto Henschel, as well as to Yasser Samayo and Dr. Michele Fenzi, for endless support (and pre-deadline food supplies) and countless kicker matches. Thanks to Leonid German for many interesting C++ discussions. Special thanks to my previous office colleague, Dr. Laura Leal-Taixé, for support, advice and collaboration.

I am grateful to the person who advised me in the beginning of my PhD, Dr. Gerard Pons-Moll. Big thanks to Dr. Michael Yang for some very interesting discussions and strategic advise.

Thanks to Mrs. Brodersen, Mrs. Bank, Mrs. Jaspers-Goeoring and Dr. Pahl, for processing my countless contract-related issues and non-trivial travel requests. Thanks to Matthias Schuh for very valuable assistance.

Thanks for all the people whom I met on the conferences and during the internships, for discussions, comments and advice on my work and the huge number of fun moments!

Big thanks to all my non-academic friends, especially to Kate Volkova, Oxana Khamidova and Dana Evseeva.

Finally, I thank my father for a lot of patience and my mother, for the invaluable support during the difficult times.

Abstract

Due to the emerging interest in virtual reality applications, as well as the existing interest in improving the means of human-computer interaction, understanding actions performed by human hands is a very important problem. One way to approach this challenge is through continuous hand pose estimation. Thanks to the recent developments in time-of-flight imaging, the general problem of hand pose recognition became much more tractable in a single-camera setting.

The classical approach to pose recognition is based on fitting a simplified non-rigid model, parametrized by a set of pose parameters, to the observation (generative approach). However, due to being only locally optimal, the found pose parameters heavily depend on the initialization. Therefore, it is common to rely on the temporal information, i.e. the assumption, that the pose parameters change continuously from one frame to another. Unfortunately, this assumption is fragile; therefore, it is desirable to estimate hand pose from a single frame, without relying on the temporal information, which is the topic of this thesis.

In the first part of the thesis, we describe the properties of the time-of-flight sensors and develop a calibration algorithm, specifically targeting the problem of time-of-flight sensor calibration. In the second part of the thesis we develop a generative pose estimation approach, based on the non-rigid iterative closest point (ICP) algorithm, that is adapted specifically to the problem of hand pose estimation from a single frame. Motivated by the problem of finding a good initialization, as well as to address the problem of gesture recognition itself, in the third part we investigate the performance of one hand pose classification approach, based on random forests (discriminative approach). Finally, we propose a way to combine both approaches to compensate for the susceptibility of the generative approach to the bad initialization and the inability of the discriminative approach to provide continuous pose estimates. We show in several experiments on the public datasets, that such a combination is beneficial and allows to obtain significant improvement in hand pose estimation performance.

Keywords: model-based pose estimation, gesture recognition, time-of-flight imaging

Zusammenfassung

Aufgrund neuester industrieller Entwicklungen in Virtual Reality Anwendungen und vielfältigen Möglichkeiten zur Verbesserung von Mensch-Maschine-Schnittstellen ist die Erkennung und Interpretation von Gesten menschlicher Hände ein sehr relevantes Problem. Eine mögliche Herangehensweise an diese Herausforderung ist die kontinuierliche Handposeschätzung in Bilddaten. Aufgrund der hohen Zahl der Freiheitsgrade und der homogenen Hautfarbe ist eine rein bildgetriebene Handposeschätzung bis heute nicht stabil in Echtzeit möglich. Dank neuester Entwicklungen in der Time-of-Flight (ToF) Bildgebung wird das Problem der Handposeschätzung jedoch handhabbar.

Der klassische Ansatz für die Handposeschätzung basiert auf der Anpassung von nicht-starren vereinfachten Handmodellen mit der Beobachtung (generativer Ansatz). Das Modell wird durch eine Menge von Pose-Parametern parametrisiert, die durch die Lösung eines nicht-konvexen Optimierungsproblems geschätzt werden. Dadurch, dass nur lokale Minima gefunden werden, hängt die Qualität der Lösung von der Initialisierung des Verfahrens ab. Daher ist eine übliche Annahme, dass sich die Pose-Parameter kontinuierlich über die Zeit ändern. Diese Annahme ist nicht immer gegeben; daher ist es wünschenswert, die Handpose aus nur einem Frame zu schätzen, ohne die Zeitinformation mit einzubeziehen. Dieser Lösungsansatz ist das Thema der vorliegenden Arbeit.

Im ersten Teil der Arbeit werden die Eigenschaften der ToF Sensoren untersucht. Auf Basis dieser Untersuchungen wird ein Kamerakalibrierungsalgorithmus entwickelt, der für den Fall der ToF Sensorkalibrierung geeignet ist. Im zweiten Teil der Arbeit wird ein generativer Ansatz vorgeschlagen, der auf dem Iterative Closest Point (ICP) Algorithmus für nicht-starre Objekte basiert und speziell für die Handposeschätzung aus einem ToF Frame adaptiert wurde. Motiviert durch das Problem, eine gute Initialisierung für den generativen Ansatz zu finden, sowie auch durch das allgemeine Problem der Gestenerkennung, wird im dritten Teil der Arbeit ein Einsatz für die Handposeklassifikation mittels Random Forests dargestellt (diskriminativer Ansatz). Schließlich werden beide Ansätze kombiniert, um die Anfälligkeit des generativen Ansatzes für eine schlechte Initialisierung zu kompensieren sowie das Problem, dass der diskriminative Ansatz keine kontinuierliche Poseschätzung ermöglicht, zu beheben. Wir zeigen in mehreren Experimenten

auf etablierten Datensätzen, dass durch die Kombination der beiden Ansätze die Handposeschätzung signifikant verbessert werden kann.

Stichworte: modelbasierte Poseschätzung, Gestenerkennung, Time-of-Flight Bildgebung

Contents

Chapter 1

Introduction

1.1 Motivation

Hand pose recognition is a very interesting and challenging problem. It has many applications in different areas, such as human-computer interaction, robotics, medical research, etc. One distinctive feature of all these tasks is that having a multi-camera setup is undesirable, while having a single sensor is the most reasonable set-up.

Hand pose estimation is often compared to full body pose estimation; however, it is a much harder problem, since unlike in the case of full body pose estimation, there is no large rigid part, such as human torso, about which certain assumptions can be made. Additionally, human hand has large number of degrees of freedom (26) and is prone to severe self-occlusion. Therefore, robust hand pose estimation from a single image for a long time remained unsolved, although some advances have been made in case of a multi-camera setup [10].

With the recent developments in the depth sensing technologies, new interest emerged to this problem, since the presence of depth data made the problem of hand pose estimation much more feasible.

1.2 The hand pose estimation problem

The main problems to deal with when solving the hand pose estimation task are the following:

- A human hand is very flexible and has many degrees of freedom.
- As mentioned previously, here is no large rigid part about which non-restrictive assumptions can be made.
- A hand is prone to severe self-occlusions.
- Its is not textured, therefore it is hard to find features to distinguish different hand parts, as well as robust edges.
- Hands vary in size and shape, and a suitable hand model is required.

Hand pose estimation has been studied for some years already. In early works, normal color sensors were used to estimate hand pose [118]. However, due to the complexity of the task only a partial hand pose recovery was done, or strong assumptions were made about the hand pose in the image. Later, different multi-camera set-ups were proposed to address the problem of self-occlusion and ambiguities, that occur when only RGB image data is available. To further reduce the complexity of the problem, tracking over several frames can be used instead of per-frame pose estimation, as shown by the authors of [10]. The proposed approach uses images collected by 8 synchronized and calibrated cameras to achieve robustness. Finally, great progress in this area happened after the appearance of depth sensors, such as Microsoft Kinect [3, 4] or Intel Creative Gesture Camera [2]. The additional depth data, delivered by the sensors mentioned above, allows to greatly reduce observation ambiguity. This, in its turn, permits a single-camera set-up, which is very desirable.

In general, the approaches for hand pose estimation (as well as human pose estimation in general) can be divided into two groups: generative [10, 94, 114] and discriminative approaches [59, 100]. Generative approaches usually solve the general hand pose estimation task, where the goal is to distinguish between all possible hand configurations, i.e., to determine hand pose parameters in the continuous space, while discriminative approaches are applied to various problems, for example sign language recognition, where a finite number of poses should be discriminated. However, pose parameters from the continuous pose space can be also estimated by firstly classifying hand parts and then fitting a full DoF model using obtained hand part correspondences.

Generative approaches The idea behind generative approaches is to find the parameters, that explain the observation in the best way, given a predefined

model. Firstly, a set of hypotheses is generated for different model parameters. The agreement between each hypothesis and the observation is then evaluated using an objective function. The goal is to find the parameters that correspond to the best value of the objective function.

Of course, the task of modeling the whole observation (an image of a hand in this case) with a high level of details is very complex and time consuming, and mainly addressed by photo-realistic rendering. Therefore, it is common to pick the most significant features of the image and model only these features, comparing them to the observation. For example, in case of human pose estimation, it can be silhouettes [97, 98], and in hand pose estimation silhouettes and edges [25, 118].

Furthermore, one can use different methods to sample candidate parameters from the parameter space in an efficient way. A conventional method is to use gradient descend and its modifications to directly optimize a given objective function. However, this approach tends to get caught in local minima, when the initial parameters are not properly selected. The parameters can also be sampled from a distribution, that reflects the prior knowledge about the observation, as in case of tracking, when the estimated pose in the previous frame is assumed to be close enough to the pose of the hand in the current frame. The sampling approach, such as particle filtering [40], simulated annealing [63] or particle swarm optimization [58], is usually used, when the cost function is hard to optimize using gradient-based methods (e.g. it does not have analytical expression, is not smooth or even not continuous) or when the prior knowledge is unreliable. Various ICP-based algorithms [14] are as well representatives of the group of generative approaches.

For hand pose estimation, the generative approach is the traditional one: firstly, a simplified geometrical hand model is created, parametrized by a set of parameters; it is then projected into the image and compared to the observation.

Discriminative approaches Discriminative approaches are targeted to distinguish different observations, instead of trying to model them. To discriminate between the observation, a classifier or a regressor is trained using training data, which is assumed to have the same feature distribution as the test data.

There are numerous methods in machine learning that were developed to solve both classification and regression tasks, such as linear models, kernel methods, random forests, etc. The size of the necessary training set and the performance of the algorithm largely depends on the chosen method.

In the recent works it has been shown, that random forests deliver particularly good results on hand pose estimation problem [59, 119], in case of hand pose estimation of a single non-interacting hand.

In general, each group of methods has its own strengths and weaknesses. We will discuss both in relation to hand pose estimation problem in the subsequent chapters.

Publicly available datasets In the past couple of years, several public datasets appeared to provide the ground for comparison of different hand pose estimation methods. Before that, however, there were no real-world datasets collected and annotated, for the reasons discussed below. It was common to evaluate the performance of the hand tracking methods in synthetic data, as done, for example, in [94].

Obtaining ground truth datasets for hand pose estimation is a difficult problem. The traditional way to obtain ground truth for human pose estimation is recording the data in marker-based Motion Capture Studio, such as Vicon Motion Capture System [5]. However, for hand tracking this approach is difficult to implement technically, since markers placed on a hand get occluded very often due to the high agility of a human hand. The other approach is to use a sensorized glove — however, it is quite expensive and was reported to be imprecise.

Nevertheless, recently several manually annotated datasets appeared: [116] presents a completely manually annotated dataset (called Dexter dataset), collected using both the Intel Creative Gesture Camera [2], multiple color cameras and the Microsoft Kinect [3]. Only finger tips and palm center are annotated in this case. We use this dataset for evaluation in the subsequent chapters.

Another dataset [119] is collected using the Intel Creative Gesture Camera and annotated using the hand pose estimation implementation, provided in Intel SDK [2]; since the resulting annotations are not always correct, the authors then manually corrected obtained tracks and used the resulting data for evaluation.

The third dataset [133] is collected using the Intel Creative Depth sensor and the ground truth joint angle annotations are obtained using a sensorized glove. The authors as well report their studies on the precision of the sensorized glove for different joints.

1.3 Contributions of the thesis

With the appearance of consumer depth cameras, the problem of hand pose estimation became much more tractable. However, it appears to be hard to solve it using just a discriminative or just a generative approach. We therefore propose to combine both approaches to achieve more robust and stable performance of the method.

In general, hand pose estimation starts with detecting and segmenting the hand in the image. In this work, we do not discuss the hand detection problem, since several very good algorithms exist to tackle it [88]. Provided a depth image, the hand segmentation task can be solved using, for example, simple region growing or depth thresholding; a more advanced algorithm is described in the Appendix.

Since in our work we work with a 3D point cloud and not a 2D depth image, the depth image should be converted into 3D using camera calibration parameters. When multiple cameras are used, for example, both a depth and a color sensor, it is also necessary to obtain the relative position of the sensors. Our first contribution solves this preliminary task: we develop a calibration algorithm to recover calibration parameters for the case of a time-of-flight depth camera (please see Chapter 1 for more details).

In this work, we firstly present a generative method for hand pose estimation, which is based on a traditional non-rigid iterative closest point (ICP) algorithm [43]. Due to its weaknesses which we discuss in the following chapters we then develop a discriminative approach. It turns out, however, that combining both methods can be beneficial, so that the combination compensates weaknesses of each approach.

1.4 Structure of the thesis

The overview of the thesis is presented in Figure 1.1.

Chapter 2 We discuss different types of depth sensors, the basics of their functioning and the challenges of depth imaging. Then an overview of the basic calibration approaches is given and finally a novel adaptation of the approach for calibration of a time-of-flight (ToF) depth sensor is presented. The proposed approach is evaluated using two recently appeared time-of-flight sensors: Intel Creative Gesture

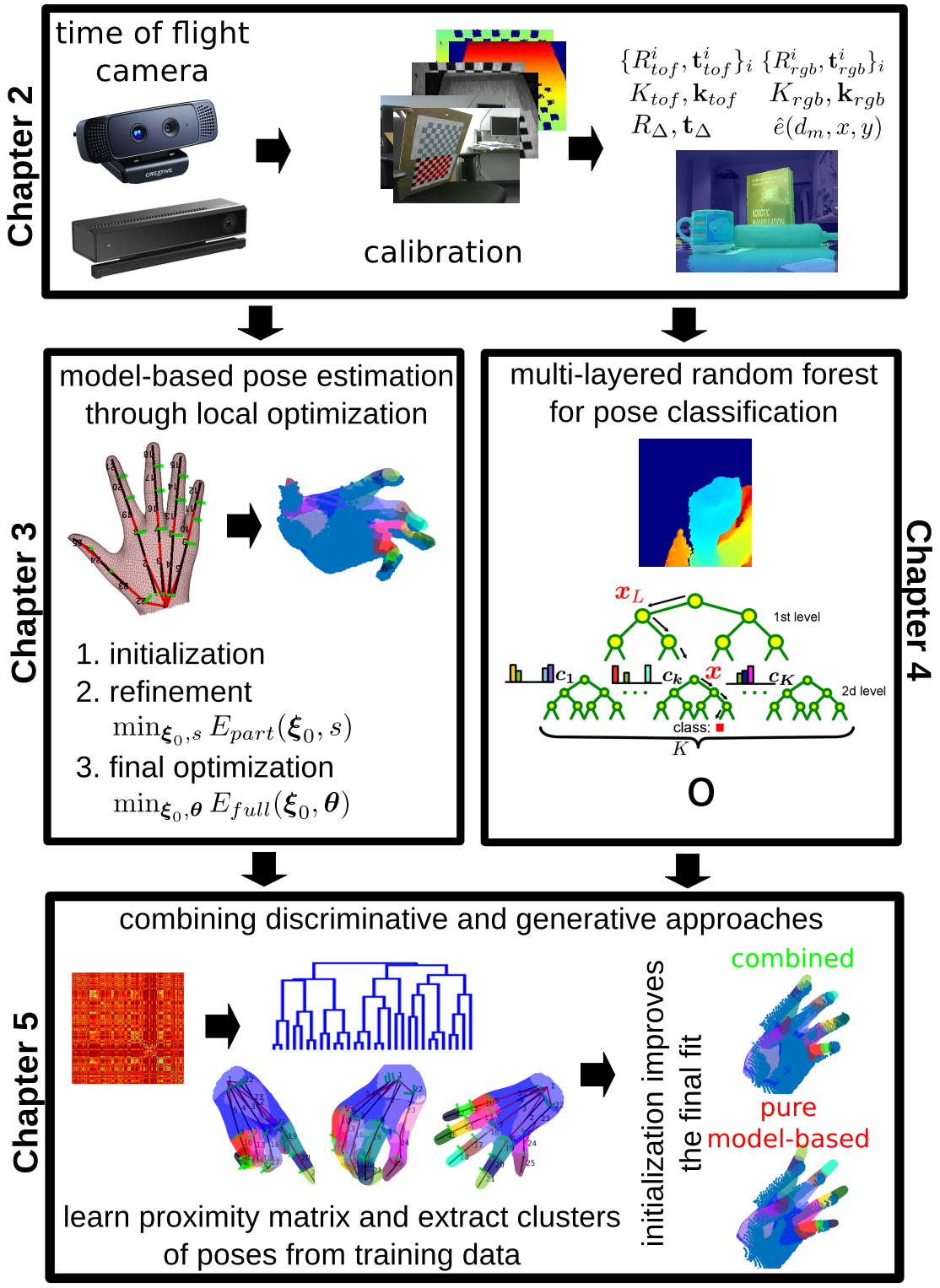


FIGURE 1.1: The full pipeline of the hand pose estimation algorithm.

Camera and Kinect2.

Chapter 3 We firstly give a theoretical introduction into generative approaches to pose estimation, and describe the hand model and its parametrization using exponential mapping, as well as the non-rigid iterative closest point (ICP) algorithm. An overview of generative approaches to hand pose estimation is presented, together with their advantages and the common drawbacks. We finally describe our approach, based on non-rigid ICP, evaluate it on the synthetic dataset and on Dexter dataset [116] and conclude the section with the discussion of the weak points of this approach.

Chapter 4 We give a theoretical introduction to random forests for classification and introduce a global point cloud descriptor. Afterwords, we present a discriminative approach for hand pose classification, based on a random forest. We evaluate this approach on the task of classification of the static signs of the ASL alphabet both on the publicly available dataset [100], collected using a Kinect, and on the new more challenging dataset, collected using an Intel Creative Depth Camera.

Chapter 5 Based on the discussion in the previous chapters, we combine the proposed discriminative and generative approaches into a single pipeline to compensate for the drawbacks of both of them. We evaluate this new approach on the synthetic database and on the Dexter dataset [116].

Chapter 6 We conclude the thesis with possible future work and open directions in the area of hand pose estimation.

1.5 Papers of the author

Below are the papers of the author and their abstracts. The papers relevant to the main contributions of the dissertation, are listed first:

[71] **Alina Kuznetsova** and Bodo Rosenhahn, *Hand pose estimation from a single RGB-D image*, 9th International Symposium on Visual Computing (ISVC), 2013

Hand pose estimation is an important task in areas such as human computer interaction (HCI), sign language recognition and robotics. Due to the high variability in hand appearance and many degrees of freedom (DoFs) of the hand, hand pose estimation and tracking is very challenging, and different sources of data and methods are used to solve this problem. In the paper,

we propose a method for model-based full DoF hand pose estimation from a single RGB-D image. The main advantage of the proposed method is that no prior manual initialization is required and only very general assumptions about the hand pose are made. Therefore, this method can be used for hand pose estimation from a single RGB-D image, as an initialization step for subsequent tracking, or for tracking recovery.

[**69**] **Alina Kuznetsova** and Laura Leal-Taixé and Bodo Rosenhahn, *Real-time sign language recognition using a consumer depth camera*, IEEE International Conference on Computer Vision Workshops (ICCVW), 3rd Workshop on Consumer Depth Cameras for Computer Vision (CDC4CV), 2013

Gesture recognition remains a very challenging task in the field of computer vision and human computer interaction (HCI). A decade ago the task seemed to be almost unsolvable with the data provided by a single RGB camera. Due to recent advances in sensing technologies, such as time-of-flight and structured light cameras, there are new data sources available, which make hand gesture recognition more feasible. In this work, we propose a highly precise method to recognize static gestures from a depth data, provided from one of the above mentioned devices. The depth images are used to derive rotation-, translation- and scale- invariant features. A multi-layered random forest (MLRF) is then trained to classify the feature vectors, which yields to the recognition of the hand signs. The training time and memory required by MLRF are much smaller, compared to a simple random forest with equivalent precision. This allows to repeat the training procedure of MLRF without significant effort. To show the advantages of our technique, we evaluate our algorithm on synthetic data, on publicly available dataset, containing 24 signs from American Sign Language(ASL) and on a new dataset, collected using recently appeared Intel Creative Gesture Camera.

[**72**] **Alina Kuznetsova** and Bodo Rosenhahn, *On calibration of a low-cost time-of-flight camera*, IEEE European Conference on Computer Vision Workshops (ECCVW), 2014

Time-of-flight (ToF) cameras are becoming more and more popular in computer vision. In many applications 3D information delivered by a ToF camera is used, and it is very important to know the camera's extrinsic and intrinsic parameters, as well as precise depth information. A straightforward algorithm to calibrate a ToF camera is to use a standard color camera calibration procedure, on the amplitude images. However, depth information

delivered by ToF cameras is known to contain complex bias due to several error sources. Additionally, it is desirable in many cases to determine the pose of the ToF camera relative to the other sensors used. In this work, we propose a method for joint color and ToF camera calibration, that determines extrinsic and intrinsic camera parameters and corrects depth bias. The calibration procedure requires a standard calibration board and around 20-30 images, as in case of a single color camera calibration. We evaluate the calibration quality in several experiments. The code for the calibration toolbox is made available online.

The papers of the author as a result of side research projects are listed below:

[93] Joachim Ohser, Claudio Ferrero, Oliver Wirjadi, **Alina Kuznetsova** and Jochen Düll and Alexander Rack, *Estimation of the probability of finite percolation in porous microstructures from tomographic images*, International Journal of Materials Research, 2012

Percolation is an important property of porous media, as it describes the connectivity of pores. We propose a novel, direction-dependent percolation probability which can efficiently be estimated from three-dimensional images obtained by microtomography. Furthermore, in order to describe the penetrability of the pore space by particles of a given diameter or a fluid of a given surface tension, we introduce a percolation probability depending on the width of the pores, from which we may also derive a measure of the mean pore channel width. As application examples, we consider the penetrability of porous berillium pebbles, the connectivity of pores in arctic firn, the percolation of the pore space of aluminium foams and the mean width of the percolating space between the fibers in a laminate's percolating pore space.

[70] **Alina Kuznetsova**, Gerard Pons-Moll and Bodo Rosenhahn, *PCA-enhanced stochastic optimization methods*, 34th Annual Symposium of the German Association for Pattern Recognition (DAGM), 2012

In this paper, we propose to enhance particle-based stochastic optimization methods (SO) by using Principal Component Analysis (PCA) to build an approximation of the cost function in a neighborhood of particles during optimization. Then we use it to shift the samples in the direction of maximum cost change. We provide theoretical basis and experimental results showing that such enhancement improves the performance of existing SO methods significantly. In particular, we demonstrate the usefulness of our method

when combined with standard Random Sampling, Simulated Annealing and Particle Filter.

[**73**] **Alina Kuznetsova**, Nikolaus F. Troje and Bodo Rosenhahn, *A statistical model for coupled human shape and motion synthesis*, 8th International Conference on Computer Graphics Theory and Applications (GRAPP), 2013

Due to rapid development of virtual reality industry, realistic modeling and animation is becoming more and more important. In the paper, we propose a method to synthesize both human appearance and motion given semantic parameters, as well as to create realistic animation of still meshes and to synthesize appearance based on a given motion. Our approach is data-driven and allows to correlate two databases containing shape and motion data. The synthetic output of the model is evaluated quantitatively and in terms of visual plausibility.

[**49**] Helga Henseler, **Alina Kuznetsova**, Peter Vogt and Bodo Rosenhahn, *Validation of the Kinect Device as a New Portable Imaging System for Three-Dimensional Breast Assessment*, Journal of Plastic, Reconstructive & Aesthetic Surgery, 2014

The aim of this study was the evaluation of a new, simple, touchless, low-cost and portable three-dimensional (3D) measurement system for objective breast assessment. The Kinect Recording System by Microsoft was used. RGB and depth images were captured of nine silicone breast implants of known volumes. The data were processed using MATLAB® software. Volume measurements were obtained in a blinded calculation on the 3D images. For further comparison, implant volumes were assessed with the Arthur Morris device, a manual measurement tool. Four tests revealed that the true breast implant volumes were calculated within an error margin of 10%. Reproducibility of measurements was satisfactory. Overall, the accuracy and reproducibility of the measurements of the Kinect System were better than those of the Arthur Morris device. Accuracy of volume assessments with the Kinect System was satisfactory for clinical application. Our new portable 3D imaging system was successfully validated. The results obtained with the Kinect System were sufficiently accurate and reproducible for application in 3D breast capture. We successfully validated the portable 3D imaging system for the first ever use in 3D breast assessment.

[77] Laura Leal-Taixé, Michele Fenzi, **Alina Kuznetsova**, Bodo Rosenhahn and Silvio Savarese, *Learning an Image-based Motion Context for Multiple People Tracking*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014

We present a novel method for multiple people tracking that leverages a generalized model for capturing interactions among individuals. At the core of our model lies a learned dictionary of interaction feature strings which capture relationships between the motions of targets. These feature strings, created from low-level image features, lead to a much richer representation of the physical interactions between targets compared to hand-specified social force models that previous works have introduced for tracking. One disadvantage of using social forces is that all pedestrians must be detected in order for the forces to be applied, while our method is able to encode the effect of undetected targets, making the tracker more robust to partial occlusions. The interaction feature strings are used in a Random Forest framework to track targets according to the features surrounding them. Results on six publicly available sequences show that our method outperforms state-of-the-art approaches in multiple people tracking.

[76] Aron Larsson, **Alina Kuznetsova**, Ola Caster and Love Ekenberg, *Implementing Second-Order Decision Analysis: Concepts, Algorithms, and Tool*, Advances in Decision Sciences, 2014

We present implemented concepts and algorithms for a simulation approach to decision evaluation with second-order belief distributions in a common framework for interval decision analysis. The rationale behind this work is that decision analysis with interval-valued probabilities and utilities may lead to overlapping expected utility intervals yielding difficulties in discriminating between alternatives. By allowing for second-order belief distributions over interval-valued utility and probability statements these difficulties may not only be remedied but will also allow for decision evaluation concepts and techniques providing additional insight into a decision problem. The approach is based upon sets of linear constraints together with generation of random probability distributions and utility values from implicitly stated uniform second-order belief distributions over the polytopes given from the constraints. The result is an interactive method for decision evaluation with second-order belief distributions, complementing earlier methods for decision evaluation with interval-valued probabilities and utilities. The method has been implemented for trial use in a user oriented decision analysis software.

[67] **Alina Kuznetsova**, Sung Ju Hwang, Bodo Rosenhahn and Leonid Sigal, *Expanding Object Detector's Horizon: Incremental Learning Framework for Object Detection in Videos*, IEEE Conference on Computer Vision and Pattern Recognition, 2015

Over the last several years it has been shown that image-based object detectors are sensitive to the training data and often fail to generalize to examples that fall outside the original training sample domain (e.g., videos). A number of domain adaptation (DA) techniques have been proposed to address this problem. DA approaches are designed to adapt a fixed complexity model to the new (e.g., video) domain. We posit that unlabeled data should not only allow adaptation, but also improve (or at least maintain) performance on the original and other domains by dynamically adjusting model complexity and parameters. We call this notion *domain expansion*. To this end, we develop a new scalable and accurate incremental object detection algorithm, based on several extensions of large-margin embedding (LME). Our detection model consists of an embedding space and multiple class prototypes in that embedding space, that represent object classes; distance to those prototypes allows us to reason about multi-class detection. By incrementally detecting object instances in video and adding confident detections into the model, we are able to dynamically adjust the complexity of the detector over time by instantiating new prototypes to span all domains the model has seen. We test performance of our approach by expanding an object detector trained on ImageNet to detect objects in egocentric videos of Activity Daily Living (ADL) dataset and challenging videos from YouTube Objects (YTO) dataset.

[68] **Alina Kuznetsova**, Sung Ju Hwang, Bodo Rosenhahn and Leonid Sigal, *Exploiting View-Specific Appearance Similarities Across Classes for Zero-shot Pose Prediction: A Metric Learning Approach*, Conference on Artificial Intelligence (AAAI), 2016

Viewpoint estimation, especially in case of multiple object classes, remains an important and very challenging problem. First, objects under different views undergo extreme appearance variations, often making within-class variance larger than between-class variance. Second, obtaining precise ground truth for real-world images, necessary for training supervised viewpoint estimation models, is extremely difficult and time consuming. As a result, annotated data is often available only for a limited number of classes. Hence it is desirable to share viewpoint information across classes. To address these problems, we propose a metric learning approach for joint class prediction

and pose estimation. Our metric learning approach allows us to circumvent the problem of viewpoint alignment across multiple classes, and does not require precise or dense viewpoint labels. Moreover, we show, that the learned metric generalizes to new classes, for which the pose labels are not available, and therefore makes it possible to use only partially annotated training sets, relying on the intrinsic similarities in the viewpoint manifolds. We evaluate our approach on two challenging multi-class datasets, 3DObjects and PASCAL3D+.

Chapter 2

Calibration of depth cameras

The major part of this thesis is devoted to computer vision algorithms, that take advantage of depth data, that is transformed into point clouds. Consequently, additional knowledge about properties of the depth sensors, such as their calibration parameters and typical artifacts, is required.

Therefore, in this chapter, we firstly introduce physical and algorithmic principles of functioning for different types of depth sensors; we then give the basics of a general camera calibration algorithm of a color sensor and finally present a new calibration algorithm for a subclass of depth sensors, the time-of-flight sensors. Additionally, we introduce an artifact correction algorithm for the case of continuous wave (CW) modulation technology [45]. We evaluate the proposed calibration algorithm on the two time-of-flight sensors, Intel Creative Gesture Camera· [2] and Microsoft Kinect2 [4].

2.1 Physical basics of depth sensors

Most of the depth sensors available on the market are based on two types of technology: time-of-flight technology [45] and structured light technology [134]; the latter is based on the principles of stereo vision [47]. Note that although both types represent active sensing technology (in contrast to other depth recovery approaches, such as passive stereo vision [47] or depth reconstruction [105]), the principles behind them are fundamentally different.

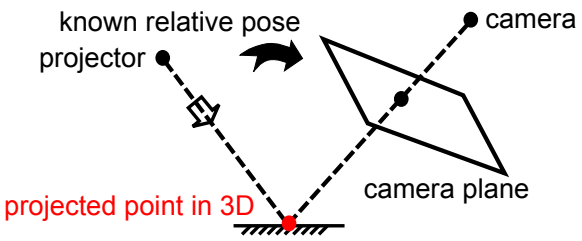


FIGURE 2.1: Principle of functioning of a structured light system: a pattern is projected by a projector and recorded by a camera; relative pose of the projector and the camera is known; the pattern seen on the image is matched to the known pattern; this information is used to determine the depth at the projected points.

Structured light technology The main idea behind that technology is essentially the same as in stereo vision [47, 92]. A structured light system typically consists of a projector and a camera [89]. The projector projects a known pattern on a scene, and the camera captures the pattern, that is then matched and compared with the known projected pattern, which allows to determine distances to the points on the image, where the infrared pattern was projected to (see Figure 2.1). Both colored light or infrared light can be used, but since the infrared pattern is not visible by a human eye and thus less disturbing, it is normally used for commercial products.

The main difficulty in structured light technology, as in stereo vision, is to recognize the pattern seen in the image by matching it to the known pattern. For that purpose, different pattern codes are used: temporal codes, spatial codes, etc [134, 135]. Microsoft Kinect [3] is reported to be based on spatial codes technology, although the exact details of implementation are unknown.

Since the internal parameters of a depth sensor are effectively the parameters of the imaging sensor, which is in case of Kinect an infrared camera, a standard pinhole camera model is adopted to model physical properties of the depth sensor.

Time-of-flight technology The basic idea behind time-of-flight technologies is to measure the time required for the emitted light to travel from the sensor to the object and back; then, the distance is calculated as:

$$d = 0.5c\tau, \tag{2.1}$$

where $c \approx 3e8$ m/s and τ is the measured time for the light to come back. The main difficulty is to measure precisely the value of τ.

To measure the value of τ, either **pulse** modulation or **continuous wave** modulation is applied.

During pulse modulation, a pulse of light with known duration is emitted by the projector, and then consumed by the receiver, which allows to estimate the traveling time. The advantage of this approach is that potentially high amounts of energy can be transferred as a pulse, allowing to detect objects that are far away. As a downside, this is not a safe method for the human eye [75]. Additionally, the equipment needed to produce very short strong pulses of light, as well as a fast shutter, is expensive, which makes the sensors based on continuous wave modulation more attractive for many applications.

Continuous wave modulation, on the contrary, does not require strong fast pulses of light, since the light traveling time is detected in the frequency domain instead of the amplitude domain by determining the phase shift of the reflected light.

Phase shift is detected as a peak of the auto-correlation function; in the simplest case, the algorithm is the following: firstly, the sinusoidal signal of the known modulation frequency is emitted: $s(t) = \cos(ft)$; the received signal is then described by $r(t) = B + A\cos(ft + \phi)$, where ϕ is the desired phase shift, while f is the modulation frequency. The cross correlation is computed as

$$c\left(\Delta t\right) = \int_0^{2\pi/f} s(t)r(t + \Delta t)dt = \frac{A}{2}\cos(f\tau + \phi) + B \tag{2.2}$$

The reflected signal is measured several times, i.e. with different phase offsets Δt, such that $C_i = c\left(i\frac{\pi}{2}\right), \quad i = 1, \ldots, 3$ and by solving the system of four equations, it can be found that

$$\phi = \arctan 2(A_3 - A_1, A_0 - A_2) \tag{2.3}$$

$$A = \frac{1}{2}\sqrt{(A_3 - A_1)^2 + (A_0 - A_2)^2} \tag{2.4}$$

Then, from known ϕ, the time-of-flight is derived as following:

$$\phi = 2\pi f\tau \tag{2.5}$$

Since ϕ can be measured unambiguously in the interval $[0, 2\pi]$ only, this constraints the distance range that can be measured by the sensor: $d_{max} = \frac{c}{2f}$ [45].

In practice, both modulation approaches can be combined to increase the robustness of the system and provide better characteristics, such as safety for human eye and less strict requirements on the light receivers.

Additionally, the system provides not only depth estimates, but also light amplitude measurements, which correspond to the objects' reflectivity. These measurements can be used as a confidence of the depth estimates (amplitude equal to zero assumes that the phase shift, and consequently, depth, are undefined).

In our experiments, we use the Intel Creative Depth Camera [2] and the Microsoft Kinect2 [4] camera, that both use time-of-flight technology. The implementation details for either camera are not publicly available.

Sources of errors in time-of-flight camera measurements Time-of-flight cameras are subject to both systematic and non-systematic errors. Systematic errors are related to various simplifications in the design of the time-of-flight cameras: error in depth measurements due to ambiguity in the signal amplitude, simplifications of demodulation, temperature of the sensor, difference in integration time (in case of pulse modulation), etc. [45].

To the non-systematic errors one relates multi-path problem (reflections of the infra-red light return to the sensor together with the original signal), illumination conditions, imprecise object boundary measurements. These errors are in general environment-dependent and cannot be corrected if the environment is unknown [45].

2.2 Projective geometry and homogeneous coordinates

In this section, we review the basics of projective geometry, which we extensively use as the tools for developing a calibration algorithm. We present the notion of homogeneous coordinates and homography [47], which we extensively use in subsequent chapters as well.

Projective geometry provides a mathematical formalism for description of cameras and associated transformations, and generalizes special cases, arising in Eucledian geometry, allowing their uniform handling within the projective geometry framework.

Projective geometry is defined by a set of axioms, which in fact define a projective space [47].

Definition 2.1. A point in a real projective space $\mathcal{P}^n$ is represented by a vector of real coordinates $X = (x_0, \ldots, x_n)^T$, at least one of which is non-zero; $\{x_i\}$ are called projective or homogeneous coordinates, and two vectors X and Y represent the same point of the projective space $\mathcal{P}^n$, iff.:

$$\exists k \in \mathbb{R} \backslash \{0\} : \quad x_i = k y_i, \quad \forall i \tag{2.6}$$

From the definition of the projective space it can be seen, that there is a one-to-one correspondence between a point in $X \in \mathcal{P}^n$, such that $x_n \neq 0$, and a point in vector space $\boldsymbol{v} \in \mathbb{R}^n$:

$$\boldsymbol{v} \leftrightarrow X = \left(\frac{x_0}{x_n}, \frac{x_1}{x_n}, \ldots, \frac{x_{n-1}}{x_n} \right) \tag{2.7}$$

We therefore can represent points in $\mathcal{R}^n$ in homogeneous coordinates as $\boldsymbol{v} = (v_1, \ldots, v_n, 1)^T$.

Additionally, all points in $\mathcal{P}^n$, such that $x_n = 0$, belong to a hyperplane at infinity and constitute $\mathcal{P}^n \backslash \mathbb{R}^n$; each point $X = (x_0, \ldots, x_{n-1}, 0)$ is called a point at infinity

In the projective space, a projective transformation is defined:

Definition 2.2. A matrix H of dimensions $(n+1) \times (n+1)$, such that $\det H \neq 0$, defines a liner transformation from $\mathcal{P}^n$ to itself, that is called a homography, or a projective transformation.

We further introduce the notion of cross-ratio, which is central in projective geometry. Cross-ratio is defined in the following way: given four points A,B,C,D on the projective line, the cross-ratio is defined as:

$$(A, B; C, D) = \frac{|AC||BD|}{|AD||BC|} \tag{2.8}$$

Any homography transforms points and lines in a way, that cross-ratio is preserved.

2.3 Pinhole camera model

Camera calibration, i.e. finding extrinsic and intrinsic parameters of a camera, is an important problem in vision and photometry. Intrinsic parameters of the camera are determined by lenses and physical principles of functioning of a camera. Extrinsic parameters of the camera determine its position and orientation in space in relation to other objects.

A simplified camera model, called pinhole camera model, was proposed to model the physical properties of the camera. Based on this model, one of the first calibration algorithms was proposed in [136].

The basic idea behind the model is that a scene is projected to a camera plane through an infinitely small hole (see Figure 2.2) instead of aperture. Therefore, no lenses are used to focus the light. Each camera also has its coordinate system, where Z axis is aligned with the optical axis of the camera and the coordinates origin is defined as an intersection of the camera optical axis with the camera plane.

Furthermore, the distance from the camera plane to the projective plane is defined as focal length, which we denote by f. Therefore, a point in 3D space with the coordinates $\boldsymbol{P_c} = \begin{pmatrix} X_c & Y_c & Z_c \end{pmatrix}^T$ in camera coordinate system is projected as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \frac{1}{Z_c} \boldsymbol{P_c} \tag{2.9}$$

In reality, lens systems introduces different kinds of non-linear distortions to the projected image, such as radial and tangential distortion, or their combination. A common distortion model assumes that radial and tangential distortion and can be modeled as following:

$$r^2 = x_n^2 + y_n^2 \tag{2.10}$$

$$x_n^d = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) x_n + 2k_3 x_n y_n + k_4 (r^2 + 2x_n^2) \tag{2.11}$$

$$y_n^d = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) y_n + k_3 (r^2 + 2y_n^2) + 2k_4 x_n y_n, \tag{2.12}$$

where $\boldsymbol{k} = \begin{pmatrix} k_1, & k_2, & \ldots, & k_5 \end{pmatrix}^T$ are the distortion coefficients [50] and $\begin{pmatrix} x_n^d & y_n^d \end{pmatrix}^T$ denotes the distorted coordinates.

If the pixels of the camera matrix are not squared, it is reflected by introducing f_x and f_y, $f_x = f s_x, \quad f_y = f s_y$, where (s_x, s_y) denote pixel size in vertical and horizontal directions.

To convert $2D$ coordinates on the projection plane to image coordinates, two more parameters, c_x and c_y, are introduced, that represent the coordinates of the intersection of the optical axis of the camera with the projection plane in image coordinates.

Finally, a point's position is usually known in world coordinates $\boldsymbol{P} = \begin{pmatrix} X & Y & Z & 1 \end{pmatrix}^T$ rather then in camera coordinates, so it should be firstly transformed to the camera

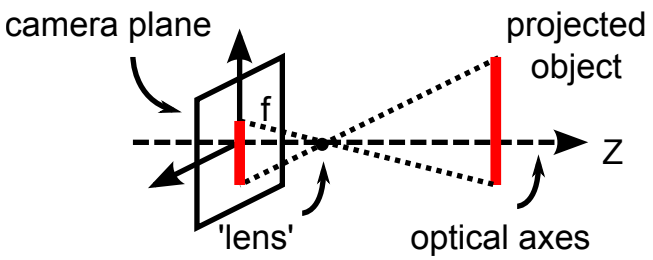


FIGURE 2.2: Pinhole camera model.

coordinates by

$$\boldsymbol{P_c} = \begin{pmatrix} R & | & \boldsymbol{t} \end{pmatrix} \boldsymbol{P} \tag{2.13}$$

where $R, \boldsymbol{t}$ are called extrinsic parameters. The final model therefore is expressed by:

$$s \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \begin{pmatrix} R & | & \boldsymbol{t} \end{pmatrix} \boldsymbol{P} \tag{2.14}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = K \begin{pmatrix} x_n^d \\ y_n^d \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{pmatrix} \begin{pmatrix} x_n^d \\ y_n^d \\ 1 \end{pmatrix}, \tag{2.15}$$

The parameters f_x, f_y, c_x, c_y, together with distortion coefficients $\boldsymbol{k}$, are called camera intrinsic parameters.

The pinhole camera model in general does not make any assumption about the light processing after it passed through the lenses, and therefore it can be adopted for depth cameras calibration.

2.4 Standard color camera calibration algorithm using checkerboard pattern

The most well known camera calibration algorithm was initially developed in [136]. The algorithms estimates extrinsic and intrinsic parameters of the camera (or multiple cameras) as specified by the pinhole camera model.

To calibrate a camera, several images of a checkerboard with the known size of checkers are firstly collected (see Figure 2.3). The checkerboard corners are then detected at each image, using, for example, the Harris corner detector [46]. Based

FIGURE 2.3: Examples of the checkerboard images; the detected corners are shown with red crosses.

on the detected corners, the initial estimation of the extrinsic and intrinsic parameters of the camera is done using homographies. Finally, bundle adjustment is used to refine the estimation.

2.4.1 Initial parameter estimation using homographies

A homography defines a transformation from one projective plane to another. Homographies are used in calibration to obtain initial guess for the extrinsic and intrinsic parameters.

During homography estimation, lens distortion is not taken into account, i.e., in the (2.14)-(2.15) $\begin{pmatrix} x_n^d & y_n^d & 1 \end{pmatrix}^T = \begin{pmatrix} x_n & y_n & 1 \end{pmatrix}^T$; let $\begin{pmatrix} x & y & 1 \end{pmatrix}^T$ denote the homogeneous image coordinates and $\begin{pmatrix} X & Y & Z & 1 \end{pmatrix}^T$ denote homogeneous model plane coordinates; without loss of generality it can be further assumed $Z = 0$, since it is always possible to find a coordinate system, in which the checkerboard model plane lies in the XY plane. Then, (2.14)-(2.15) without lens distortion can be written as:

$$s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \underbrace{K \begin{pmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{pmatrix}}_{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.16}$$

$$H = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \tag{2.17}$$

where H is the homography matrix and $R = \begin{pmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{pmatrix}$. From the equations it is easy to see that the homography is defined up to a scaling factor.

Given a set of points on the plane in the world coordinates $\left\{\begin{pmatrix} X_l & Y_l & 1 \end{pmatrix}^T\right\}_{l=1}^{L}$ and the corresponding points on the image $\left\{\begin{pmatrix} x_l & y_l & 1 \end{pmatrix}^T\right\}_{l=1}^{L}$, the homography estimation problem can be transformed into a system of homogeneous linear equations in the form of:

$$A\boldsymbol{h} = 0, \tag{2.18}$$

where $\boldsymbol{h} = \begin{pmatrix} H_{11} & H_{12} & H_{13} & H_{21} & H_{22} & H_{23} & H_{31} & H_{32} & H_{33} \end{pmatrix}^T$ and the matrix A has the following form:

$$A = \begin{pmatrix} -X_1 & -Y_1 & -1 & 0 & 0 & 0 & x_1X_1 & x_1Y_1 & x_1 \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\ -X_L & -Y_L & -1 & 0 & 0 & 0 & x_LX_L & x_LY_L & x_L \\ 0 & 0 & 0 & -X_1 & -Y_1 & -1 & y_1X_1 & y_1Y_1 & y_1 \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\ 0 & 0 & 0 & -X_L & -Y_L & -1 & y_LX_L & y_LY_L & y_L \end{pmatrix} \tag{2.19}$$

The non-trivial solution of the homogeneous system of linear equations $\boldsymbol{h}^*$ is found by using the SVD decomposition of the matrix A^T [47]:

$$A^T = U\Sigma V^T, V = \begin{pmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_9 \end{pmatrix} \in \mathbb{R}^{9\times 9}, UU^T = I, \tag{2.20}$$

$$\Rightarrow \boldsymbol{h}^* = \boldsymbol{u}_9, \tag{2.21}$$

i.e. the non-trivial solution is given by the eigenvector of the matrix A^TA, corresponding to its smallest eigenvalue.

As can be seen from Equation (2.16), homographies combine information from both extrinsic and intrinsic parameters. Given $\{H_i\}_{i=1}^{I}$ homography matrices estimated from the set of images $i = 1, \cdots, I$, the goal now is to separate extrinsics from intrinsics by estimating the common matrix K, along with the extrinsic parameters for each image $\{R_i, \boldsymbol{t}_i\}$.

Using that by definition $H_i = s\begin{pmatrix} \boldsymbol{h}_1^i & \boldsymbol{h}_2^i & \boldsymbol{h}_3^i \end{pmatrix} = K\begin{pmatrix} \boldsymbol{r}_1^i & \boldsymbol{r}_2^i & \boldsymbol{t}_i \end{pmatrix}$, the system of linear equations can be formed, taking into account that $\boldsymbol{r}_1^i \perp \boldsymbol{r}_2^i \Rightarrow (\boldsymbol{r}_1^i)^T\boldsymbol{r}_2^i = 0$ and $\|r_j^i\| = 1 \Rightarrow (\boldsymbol{r}_j^i)^T\boldsymbol{r}_j^i = 1, \quad j = 1, 2$:

$$(\boldsymbol{h}_1^i)^T(K^{-1})^TK^{-1}\boldsymbol{h}_2^i = 0, \tag{2.22}$$

$$(\boldsymbol{h}_1^i)^T(K^{-1})^TK^{-1}\boldsymbol{h}_1^i = (\boldsymbol{h}_2^i)^T(K^{-1})^TK^{-1}\boldsymbol{h}_2^i, \tag{2.23}$$

$$i = 1, \cdots, I \tag{2.24}$$

The solution K of the system of equations (2.22) − (2.24) is found in the closed form, as described in [136]. Afterwards, the extrinsic parameters are estimated as:

$$\boldsymbol{r}_1^i = sK^{-1}\boldsymbol{h}_1^i, \quad \boldsymbol{r}_2^i = sK^{-1}\boldsymbol{h}_2^i, \quad \boldsymbol{r}_3^i = \boldsymbol{r}_1^i \times \boldsymbol{r}_2^i, \quad \boldsymbol{t}_i = sK^{-1}\boldsymbol{h}_3^i, \tag{2.25}$$

$$s = \frac{1}{\|K^{-1}\boldsymbol{h}_1^i\|} = \frac{1}{\|K^{-1}\boldsymbol{h}_1^i\|} \tag{2.26}$$

2.4.2 Parameter refinement

After the initial parameter estimates are obtained using homographies, it is used as a starting point to solve a non-convex optimization problem, that refines the estimation and includes lens distortion into the computations.

We now denote the corner l extracted from an image i as $\boldsymbol{p}^{il} = \begin{pmatrix} x^{il} & y^{il} \end{pmatrix}^T$. Then, the estimate of the corresponding corner $\hat{\boldsymbol{p}}^{il}$ based on calibration parameters is obtained using (2.14) - (2.15) and (2.10) -(2.12).

The maximum likelihood estimate of the parameters $K, \boldsymbol{k}, \{R_i, \boldsymbol{t}_i\}_{i=1}^I$ is found by solving the following non-linear minimization problem:

$$\min_{K,\boldsymbol{k},\{R_i,\boldsymbol{t}_i\}_{i=1}^I} \sum_{i=1}^{I}\sum_{l=1}^{L} \|\boldsymbol{p}^{il} - \hat{\boldsymbol{p}}^{il}(K, \boldsymbol{k}, R_i, \boldsymbol{t}_i)\|^2 \tag{2.27}$$

using the Levenberg-Marquardt algorithm [80]. This procedure is called bundle adjustment in the literature.

2.4.3 Problems of standard camera calibration approaches

The standard procedure for time-of-flight (ToF) camera calibration is to use the amplitude image of the IR sensor and apply a color camera calibration algorithm described in previous the sections (see also [136]), since certain color variations are visible on the amplitude image. However, time-of-flight (ToF) cameras often have low resolution, which influences the precision of any corner detector, and consequently the accuracy of the calibration. Additionally, the calibration procedure itself does not take into account depth measurements of the camera, which could potentially improve the calibration results.

An example of this is presented in Figure 2.4. Here, the calibration algorithm from [136] is used to determine the camera extrinsics and intrinsics, and then the determined extrinsics are employed to predict the 3D position of the checkerboard

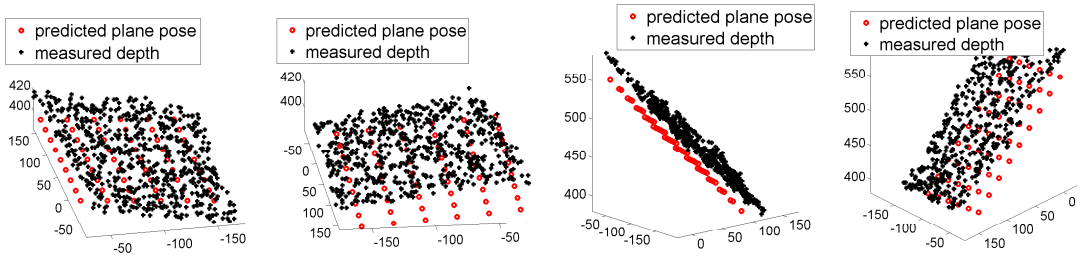


FIGURE 2.4: Mismatch between the predicted positions of the checkers from the calibration and the depth, measures at these points by a time-of-flight sensor.

plane. However, when depth measurements of the same plane are transformed in the three-dimensional space, there is a mismatch between the depth, predicted from the camera calibration, and the depth measurements. Such effects are undesirable in various applications, for example, when collecting ground truth data - therefore, integrating available depth information into the calibration procedure is desirable. One additional problem arises from the physical properties of the time-of-flight sensors: as reported in [62, 74, 82, 83], the depth measurements themselves are often inaccurate due to both systematic and non-systematic errors (see Section 2.1 for details). Therefore, simply including depth measurements into the calibration procedure might have negative effect on the calibration quality.

We therefore propose two main enhancements into the calibration algorithm:

- Correct for the inaccuracies in depth measurements.
- Include the corrected depth measurements into the calibration procedure.

2.5 Related work

As mentioned previously, initially ToF cameras were calibrated using amplitude images, acquired by a ToF camera, using color camera calibration procedure [136]. However, the resolution of ToF cameras is usually small (320×240 for the Intel Creative Depth sensor [2]), therefore the obtained calibration is not very precise, and additionally, systematic biases of the depth estimation are not taken into account.

In [62, 82, 83] it is shown that the bias is non-constant and shows high dependency on the distance to the camera. In the first work it is proposed to obtain ground

truth distance measurements to estimate the distance error and subsequently fitting a B-spline to correct it during test phase [82]; in the second work, instead of fitting a B-spline, a look-up-table based on depth is created [83]. Unfortunately, both methods rely on the ground truth data, which is hard to obtain without a special setup. In [62], the bias is computed by fitting a 6-degree polynomial to the difference between the depth values, obtained through checkerboard-based calibration, and the measured depth values. However, the fact that the amplitude-based calibration is not itself accurate due to low resolution of the amplitude images, is not accounted for.

Joint calibration of a system of several ToF and RGB cameras was proposed in [44]. There, projective alignment is used to find the mapping between different sensors. However, this method does not account for systematic depth errors and can only be used if at least one color camera is present.

A joint depth and color calibration method was proposed in [50] for structured light sensors. In this method, 3D planes are used instead of checkerboard corners to find the parameters of the depth sensor, since structured light sensors do not provide amplitude information. The depth measurement bias, called depth distortion, is corrected on the basis of several images of a planar surface. However, in our experiments with the Intel Creative Gesture Camera [2], this method was inapplicable due to the imprecision and greater amount of noise in depth data, provided by time-of-flight cameras.

Finally, in some applications depth inaccuracies are compensated after the calibration during the actual processing of 3D data, as in [22]. Firstly, a standard camera calibration algorithm is used to estimate camera calibration parameters, then the 3D point cloud is reconstructed from the depth images using these parameters, and the bias is taken into account during 3D object reconstruction. However, this approach adds complexity in the data processing method and cannot be directly transferred to the problems other then 3D reconstruction.

We address the disadvantages of the approaches mentioned above by combining plane-based calibration [50] with the standard calibration procedure for the amplitude image, therefore compensating for the low resolution of the ToF images and depth inaccuracies. We also model depth systematic error, using a non-parametric kernel regression approach [91]. The error is obtained by comparing the depth values measured by the ToF camera and the values obtained by the calibration procedure.

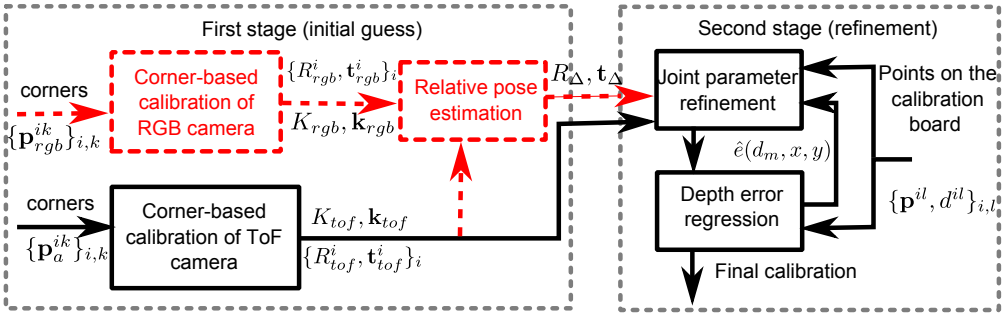


FIGURE 2.5: Overview of the calibration procedure.

2.6 ToF camera calibration algorithm

The basis of our algorithm is the standard color camera calibration algorithm, described in Section 2.4.

Initially 20 to 30 images of a calibration board should be recorded. The calibration board differs from the one used in [136]: it should consist of a black-and-white pattern and sufficient space on the same board without any pattern, that would be visible on the amplitude images (see Figure 2.6(a) for examples and Section 2.6.3 for further explanation). Similar to the standard camera calibration algorithm, there are two stages of the calibration process (see Figure 2.5). In the first stage, we obtain the first estimate of the parameters of the ToF camera using corner-based calibration on the amplitude image (Section 2.4,2.6.1). In case there are color cameras present in the system, we also obtain the parameters for each of them independently using the algorithm described in Section 2.4. Finally, the relative pose of all cameras are obtained using a closed-form solution for the pose estimation problem (Section 2.6.2). In this work we constrain the number of color cameras to be no more then one, but the algorithm is easy to extend to the case of multiple cameras.

In the second stage, we iteratively refine the initial guess by re-estimating the cameras' instrinsic parameters, their relative poses (if applicable) and the depth bias:

1. Re-estimate the parameters of the cameras using joint optimization (Section 2.6.4).

2. Depth bias is estimated as described in Section 2.6.3.

2.6.1 Corner-based calibration

As a pre-processing step, checkerboard corners $\boldsymbol{p}^{ik} = \begin{pmatrix} x^{ik} & y^{ik} \end{pmatrix}^T, k = 1 \dots K$ are extracted from each image with index $i = 1 \dots I$. The initial guess for the parameters is obtained using homographies as described in Section 2.4.1; the initial guess is then refined using an optimization procedure as explained in Section 2.4.2.

After this step, the independent estimations for the ToF camera parameters K_{tof}, $\boldsymbol{k}_{tof}$, $\{R^i_{tof}, \boldsymbol{t}^i_{tof}\}_i$ and color camera parameters K_{rgb}, $\boldsymbol{k}_{rgb}$, $\{R^i_{rgb}, \boldsymbol{t}^i_{rgb}\}_i$ are obtained. Here, $\{R^i_{tof}, \boldsymbol{t}^i_{tof}\}_i$ and $\{R^i_{rgb}, \boldsymbol{t}^i_{rgb}\}_i$ correspond to the transformation from the checkerboard local coordinate system in each frame i to the camera coordinate system, as described by Equation (2.14).

2.6.2 ToF and RGB relative pose estimation

In the applications where the color information is used on a par with depth information knowing the relative transformation between color and ToF camera coordinate systems is required. We search for the transformation in the form of

$$\boldsymbol{P}_{rgb} = R_\Delta \boldsymbol{P}_{tof} + \boldsymbol{t}_\Delta \tag{2.28}$$

where $\boldsymbol{P}_{rgb}$ — 3-dimensional point coordinates in the coordinate system, associated with the color camera, and $\boldsymbol{P}_{tof}$ — point coordinates in the system associated with the ToF camera.

The relative pose is estimated using the relative transformations between the camera coordinate system and the local calibration board coordinate system, $\{R^i_{tof}, \boldsymbol{t}^i_{tof}\}_i$ and $\{R^i_{rgb}, \boldsymbol{t}^i_{rgb}\}_i$, computed in the previous step.

Since the relative transformation for a frame i is equivalent to the position of the calibration plane relative to the camera, equivalently plane parametrization can be converted to the normal $\boldsymbol{n}^i$ and offset θ^i representation instead:

$$\boldsymbol{n}^i = \boldsymbol{r}^i_3, \quad \theta^i = \langle \boldsymbol{r}^i_3, \boldsymbol{t}_i \rangle \tag{2.29}$$

The parameters of the planes in the ToF coordinate system are denoted $\{\boldsymbol{n}^i_{tof}, \theta^i_{tof}\}_i$ and the parameters in the color camera coordinate system — by $\{\boldsymbol{n}^i_{rgb}, \theta^i_{rgb}\}_i$. The

normal vectors and the offsets are stacked together to form the matrices:

$$N_{tof} = \begin{pmatrix} \boldsymbol{n}_{tof}^1 & \boldsymbol{n}_{tof}^2 & \dots & \boldsymbol{n}_{tof}^I \end{pmatrix} \tag{2.30}$$

$$N_{rgb} = \begin{pmatrix} \boldsymbol{n}_{rgb}^1 & \boldsymbol{n}_{rgb}^2 & \dots & \boldsymbol{n}_{rgb}^I \end{pmatrix} \tag{2.31}$$

$$\boldsymbol{D}_{tof} = \begin{pmatrix} \theta_{tof}^1 & \theta_{tof}^2 & \dots & \theta_{tof}^I \end{pmatrix}^T \tag{2.32}$$

$$\boldsymbol{D}_{rgb} = \begin{pmatrix} \theta_{rgb}^1 & \theta_{rgb}^2 & \dots & \theta_{rgb}^I \end{pmatrix}^T \tag{2.33}$$

The problem of finding the relative rotation R_Δ is then re-formulated as:

$$\min_{R_\Delta} \|R_\Delta N_{tof} - N_{rgb}\|_F, \quad \text{subject to} \quad R_\Delta^T R_\Delta = I \tag{2.34}$$

The above minimization problem is known as Procrustes problem [41] and has the closed form solution, that can be obtained using SVD decomposition of $N_\Delta = N_{tof} N_{rgb}^T$:

$$N_\Delta = U \Sigma V^T \tag{2.35}$$

$$R_\Delta = V U^T \tag{2.36}$$

The translation is determined from the planes equation: given that $\langle \boldsymbol{n}_{rgb}^i, \boldsymbol{P}_{rgb} \rangle = \theta_{rgb}^i$, and (2.28), by substituting $\boldsymbol{P}_{rgb}$ with $R_\Delta \boldsymbol{P}_{tof} + \boldsymbol{t}_\Delta$, it can be shown that:

$$\langle \boldsymbol{n}_{rgb}^i, R_\Delta \boldsymbol{P}_{tof} + \boldsymbol{t}_\Delta \rangle = \theta_{rgb}^i \tag{2.37}$$

$$\langle \underbrace{R_\Delta^T \boldsymbol{n}_{rgb}^i}_{\boldsymbol{n}_{tof}^i}, \boldsymbol{P}_{tof} \rangle + \langle \boldsymbol{n}_{rgb}^i, \boldsymbol{t}_\Delta \rangle = \theta_{rgb}^i \tag{2.38}$$

$$\langle \boldsymbol{n}_{rgb}^i, \boldsymbol{t}_\Delta \rangle = \theta_{rgb}^i - \theta_{tof}^i \tag{2.39}$$

Consequently, finding $\boldsymbol{t}_\Delta$ for all images is formulated as following minimization problem:

$$\min_{\boldsymbol{t}_\Delta} \|(\boldsymbol{D}_{rgb} - \boldsymbol{D}_{tof}) - N_{rgb}^T \boldsymbol{t}_\Delta\|_F \tag{2.40}$$

having the following closed-form solution:

$$\boldsymbol{t}_\Delta = (N_{rgb} N_{rgb}^T)^{-1} N_{rgb} (\boldsymbol{D}_{rgb} - \boldsymbol{D}_{tof}) \tag{2.41}$$

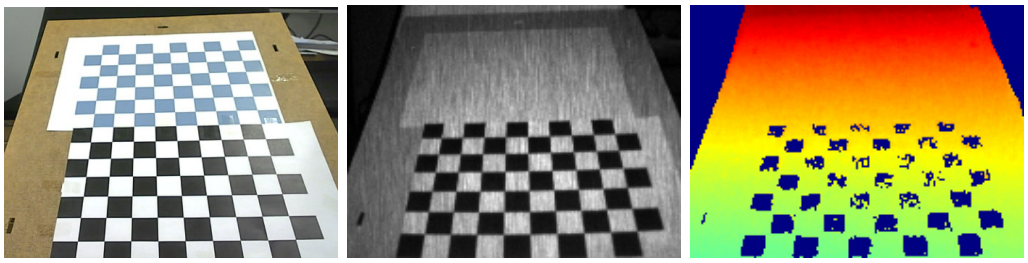
Consequently, the relative transformation between the extrinsic parameters of the color and Tof cameras is given as:

$$R_{rgb}^i = R_\Delta R_{tof}^i \tag{2.42}$$

$$\boldsymbol{t}_{rgb}^i = R_\Delta \boldsymbol{t}_{tof}^i + \boldsymbol{t}_\Delta \tag{2.43}$$

The initial guess $R_\Delta, \boldsymbol{t}_\Delta$ is later refined during joint optimization, as described in Section 2.6.4.

2.6.3 Depth correction



(a) (b) (c)

FIGURE 2.6: 2.6(a) Three-color calibration pattern; 2.6(b) amplitude image: the black pattern is visible, while the color pattern is not; 2.6(c) depth image: the areas with black squares have invalid depth values.

As discussed in Section 2.4.3, ToF cameras tend to have a systematic bias in depth estimates, which leads to the mismatch between expected and estimated depth measurements (see Figure 2.4). We propose to model the systematic error as an additive component that is dependent on the multiple parameters, such as real distance to the camera or position of the projected point on the camera matrix:

$$d_r = d_m + e + \xi, \tag{2.44}$$

Here, d_r corresponds to the real distance to the object, d_m is the depth measured by the camera, e is the systematic component of the error, and $\xi \sim \mathcal{N}(0, \sigma)$ represents Gaussian noise.

In our experiments, we observed that e depends on three parameters: the real distance to the object d_r at a pixel (x, y), as well as pixel coordinates on the camera matrix, i.e. $e = e(d_r, x, y)$.

Given that ground truth measurements are available, the dependence can be modeled by fitting a regression function to $e(d_r, x, y) = d_r(x, y) - d_m(x, y)$. We use a non-parametric kernel regression with Gaussian kernel [91], as it is a fairly simple and fast approach.

In this case, $\hat{e}(d_r, x, y)$ is computed as follows given the points with known error measurements $\{\boldsymbol{q}_j, e_j\}_j$:

$$\hat{e}(d_r, x, y) = \frac{\sum_j K(\boldsymbol{q}, \boldsymbol{q}_j) e_j}{\sum_j K(\boldsymbol{q}, \boldsymbol{q}_j)}, \tag{2.45}$$
$$\boldsymbol{q} = \begin{pmatrix} d_r & x & y \end{pmatrix}^T, K(\boldsymbol{q}, \boldsymbol{q}_j) = e^{-\frac{1}{2h^2}\|\boldsymbol{q}-\boldsymbol{q}_j\|^2}$$

Here h is the bandwidth parameter that is optimized using grid search.

However, obtaining ground truth measurements for each pixel is not a trivial task, which cannot be performed outside of a lab setup. We propose to avoid it by firstly calibrating extrinsic and intrinsic parameters of the camera without taking into account the depth bias and then using depth, predicted from the calibration, to determine $\hat{e}(d_r, x, y)$. Since d_r is not available at run-time, we estimate the bias as $\hat{e}(d_m, x, y)$, using $e_j = \hat{d}^{il} - d^{il}$, i.e. the difference between depth measurement d^{il} at pixel $\boldsymbol{p}^{il} = (x^{il}, y^{il})^T$ and the estimated depth $\hat{d}^{il}$ and $\boldsymbol{q}_j = (\hat{d}^{il}, x^{il}, y^{il})^T$. An example of the fitted systematic error surface is given in Figure 2.7(a). At the test time, the correction is applied to depth as follows:

$$d_r(x, y) = d_m(x, y) + \hat{e}(d_m(x, y), x, y) \tag{2.46}$$

To predict the expected depth value at a given pixel, we use the current estimates of the ToF camera parameters and the plane pose, and render the plane i onto each image. From the plane pose relative to the camera $R^i_{tof}, \boldsymbol{t}^i_{tof}$, the plane parametrization with normal $\boldsymbol{n}^i_{tof}$ and offset θ^i_{tof} can be computed using Equation 2.29. Then, the predicted depth value at pixel l, $\boldsymbol{p}^{il}$, is computed as:

$$\hat{d}^{il} = \frac{\theta^i_{tof}}{n^i_{tof,1} x^{il}_n + n^i_{tof,1} y^{il}_n + n^i_{tof,3}}, \tag{2.47}$$

where x^{il}_n, y^{il}_n are computed from $\boldsymbol{p}^{il}$ by first inverting (2.15) and undistorting the normalized coordinates, which in general case can be solved using a gradient descend algorithm.

Unfortunately, the calibration pattern itself cannot be used to produce reliable error estimation: one would be only able to detect checkers in the image, if the corresponding amplitude value is low (i.e., the checkers have black color), which would indicate unreliable depth estimates. Therefore, the checkerboard for the calibration contains a part without a pattern on it, since the black-and-white pattern would influence reliability of depth estimation (see Figure 2.6(b) and Figure 2.6(c)). In our case, this part contains a pattern of blue-and-white checkers,

used to determine re-projection error from ToF image to color camera, for example, for calibration evaluation.

2.6.4 Joint optimization

After the initial guess for the parameters is obtained, an iterative process is performed:

1. The parameters of the cameras $K_{tof}, \boldsymbol{k}_{tof}, \{R^i_{tof}, \boldsymbol{t}^i_{tof}\}_i$ ($K_{rgb}, \boldsymbol{k}_{rgb}$, $\{R^i_{rgb}, \boldsymbol{t}^i_{rgb}\}_i$, $R_\Delta, \boldsymbol{t}_\Delta$ in case of presence of a color camera) are jointly optimized.

2. Systematic error is estimated, as described in Section 2.6.3.

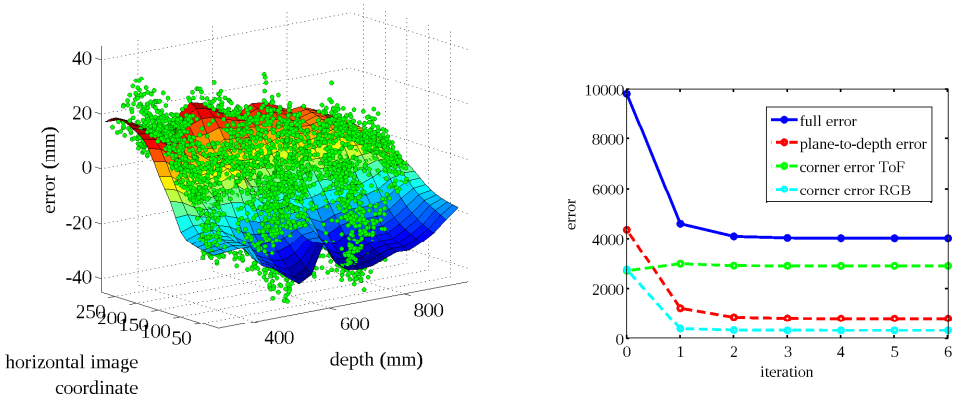


FIGURE 2.7: 2.7(a)Depth error e measured relative to d_m and the x coordinate of the image (grey dots); fitted regression surface $e(d_m, x)$. 2.7(b) Decrease of E_{full} depending on the iteration, as well as decrease in each of the sums of the error of E_{full}: depth-to-plane error (red), corner error for the amplitude images (green) and corner error on the RGB images (azure); as can be seen, 5 iterations are enough so that the algorithm converges.

The idea behind joint optimization of the parameters is the following: as mentioned above, corner-based calibration suffers from the low resolution of the ToF images and plane-based calibration from [50] gets confused by the systematic error and noise in depth measurements. We therefore fuse both approaches to compensate for their drawbacks.

Joint optimization of the parameters is performed by minimizing a corner-based error term and a plane-based error term together: joint parameter refinement

TABLE 2.1: Estimated calibration parameters for Intel Creative Gesture Camera.

	MFR	**CAL**	**CAL-D**	**CAL-DC**
$K_{rgb} : f_x, f_y$	$583.08, 596.20$	$595.71, 595.06$	$595.71, 595.06$	$595.71, 595.06$
c_x, c_y	$320, 240$	$310.80, 221.63$	$310.80, 221.63$	$310.80, 221.63$
$K_{tof} : f_x, f_y$	$224.5, 230$	$230.37, 230.46$	$234.38, 234.80$	$230.78, 230.72$
c_x, c_y	$160, 120$	$159.50, 118, 67$	$164.52, 113.36$	$166.0.3, 119.29$

	CAL	**CAL-D**	**CAL-DC**
$\boldsymbol{k}_{rgb} : k_1, k_2$	$0.01, -0.06$	$-0, 01, -0.06$	$-0, 01, -0.06$
k_3, k_4, k_5	$-0.0, 0.0, 0$	$-0, 0, 0.0, 0.0$	$-0, 0, 0.0, 0.0$
$\boldsymbol{k}_{tof} : k_1, k_2$	$-0.18, 0.13$	$-0.15, 0.14$	$-0.18, -0.14$
k_3, k_4, k_5	$-0.00, 0.00, -0.03$	$-0.01, 0.00, -0.08$	$0.00, 0.01, -0.07$
$\boldsymbol{t}_\Delta$	$24.93, 0.09, 0.85$	$25.26, -1.57, -9.98$	$25.16, 0.09, 0$
R_Δ	$\begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1 & 0.00 \\ 0.00 & 0.00 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.99 & 0.00 & 0.02 \\ 0.00 & 0.99 & -0.02 \\ -0.02 & 0.02 & 0.99 \end{pmatrix}$	$\begin{pmatrix} 0,99 & 0.00 & 0.03 \\ 0.00 & 1 & 0.00 \\ -0.03 & 0.0 & 0.99 \end{pmatrix}$

is essentially done by minimizing the following functional E_a, using Levenberg-Marquardt method:

$$e_{tof} = \frac{1}{IK} \sum_i \sum_k \|\boldsymbol{p}_a^{ik} - \hat{\boldsymbol{p}}_a^{ik}\|^2 \tag{2.48}$$

$$e_d = \frac{1}{IL} \sum_i \sum_l (d^{il} - \hat{d}^{il})^2 \tag{2.49}$$

$$E_a = \frac{e_{tof}}{\sigma_a^2} + \frac{e_d}{\sigma_d^2} = \sum_i \left(\underbrace{\sum_k \frac{\|\boldsymbol{p}_a^{ik} - \hat{\boldsymbol{p}}_a^{ik}\|^2}{\sigma_a^2}}_{\text{corner error ToF}} + \underbrace{\sum_l \frac{(d_r^{il} - \hat{d}^{il})^2}{\sigma_d^2}}_{\text{plane-to-depth error}} \right) \tag{2.50}$$

Here, $\boldsymbol{p}_a^{ik}$ is the k-th checkerboard corner detected on image i, and $\hat{\boldsymbol{p}}_a^{ik}$ is the projection of the model's k-th corner using ToF camera distortion parameters and its projection matrix, $\hat{d}^{il}$ is depth prediction computed using (2.47) and d_r^{il} is the depth measurement, corrected with the already estimated systematic error as specified in the Equation (2.46). Each error term is normalized by the variance in the corresponding error: σ_a^2 and σ_d^2, computed directly after the initialization step. Note that the variance is kept fixed during the whole optimization procedure.

Plane-to-depth error estimation follows the idea described in [50]. Unlike [50] though, we use the same distortion model for the ToF camera as for the color camera, and not the reverse one. In case of presence of a color camera, a new term

is added to E_a:

$$e_{rgb} = \frac{1}{IK} \sum_i \sum_k \|\boldsymbol{p}_{rgb}^{ik} - \hat{\boldsymbol{p}}_{rgb}^{ik}\|^2 \tag{2.51}$$

$$E_{full} = \frac{e_{tof}}{\sigma_a^2} + \frac{e_{rgb}}{\sigma_{rgb}^2} + \frac{e_d}{\sigma_d^2} = \tag{2.52}$$

$$= \sum_i \left(\sum_k \frac{\|\boldsymbol{p}_a^{ik} - \hat{\boldsymbol{p}}_a^{ik}\|^2}{\sigma_a^2} + \underbrace{\sum_k \frac{\|\boldsymbol{p}_{rgb}^{ik} - \hat{\boldsymbol{p}}_{rgb}^{ik}\|^2}{\sigma_{rgb}^2}}_{\text{corner error RGB}} + \sum_l \frac{(d^{il} - \hat{d}^{il})^2}{\sigma_d^2} \right), \tag{2.53}$$

where $\boldsymbol{p}_{rgb}^{ik}$ is the k-th checkerboard corner detected in the corresponding color image i and $\hat{\boldsymbol{p}}_{rgb}^{ik}$ is the projection of the k-th corner into the color image using the relative checkerboard position from equations (2.42)-(2.43).

As shown in Figure 2.7(b), the error decreases when iterating between joint optimization and error regression, and in our experiments became stable after ca. 5 iterations.

2.7 Evaluation

We perform several experiments to evaluate different aspects of the proposed calibration method. Firstly, following the literature, we report the value of the minimized function (2.53) as a measure of the model fit to the calibration data. Furthermore, we evaluate the reprojection error from the depth image to the color image. This experiment shows the accuracy of the acquired relative pose estimate between ToF camera and color camera, as well as allows to evaluate the influence of the depth correction. Following [50], we evaluate the depth calibration by estimating the angle between two perpendicular planes. Finally, to evaluate the influence of the depth correction directly, we evaluate the distance between two rigidly connected parallel planes, and compare it with the ground truth. We show, that this error depends on the distance to the depth sensor.

We compare three obtained calibrations: firstly, we provide the evaluation of the manufacturer calibration (**MFR**); secondly, we show the calibration result for the baseline [136](**CAL**), for our method without depth correction (**CAL-D**) and, finally, the method with depth correction (**CAL-DC**).

To show, that our method is independent of the concrete depth sensor, we perform the evaluation with two cameras produced by different manufacturers: Intel Creative Gesture Camera [2] and Microsoft Kinect 2 [4].

2.7.1 Intel Creative Gesture Camera

Intel Creative Gesture Camera [2] is a low-cost time-of-flight sensor. It has a separate color camera of resolution 640×480 and a time-of-flight sensor with resolution of 320×240. The acceptable distance range is from 10cm to 1m. The manufacturer provides the intrinsic parameters for both sensors, as well as mapping from the depth image to the color image in the form of pixel correspondences.

Firstly, we compute the calibration results using 38 images. The estimated parameters are given in Table 2.1.

TABLE 2.2: Model fit measurements for Intel Creative Gesture Camera; e_{rgb} stands for the corner error on color images, e_{tof} stands for the corner error on the confidence images, and e_d stands for the depth errors.

	e_{rgb}	e_{tof}	e_d
CAL	0.234	2.463	3.744
CAL-D	0.578	2.830	1.123
CAL-DC	0.578	2.830	0.653

Model fit: In Table 2.2 the results for model fitting are given in terms of the final values of the cost function after joint optimization. There e_{rgb} is defined as in (2.51), e_{tof} as in (2.48) and e_d as in (2.49).

Reprojection error from the depth to the color image: In this experiment, we evaluate the relative pose estimation between depth and RGB camera. In Table 2.3, the quantitative evaluation results are presented — we collect a

TABLE 2.3: Comparison between depth-color image alignment provided by manufacturer, our calibration without depth correction and with depth correction; the results are reported in pixels.

	m_d	σ_d	m_{conf}	σ_{conf}
MFR	2.70	1.49	–	–
CAL	1.78	0.77	1.97	0.73
CAL-D	1.56	0.96	1.88	0.78
CAL-DC	1.45	0.93	1.88	0.78

disjoint set of checkerboard images, estimate the plane pose on the depth images and reproject the corners from each depth image to the corresponding color image, using depth measurements in the respective points of the depth image, and

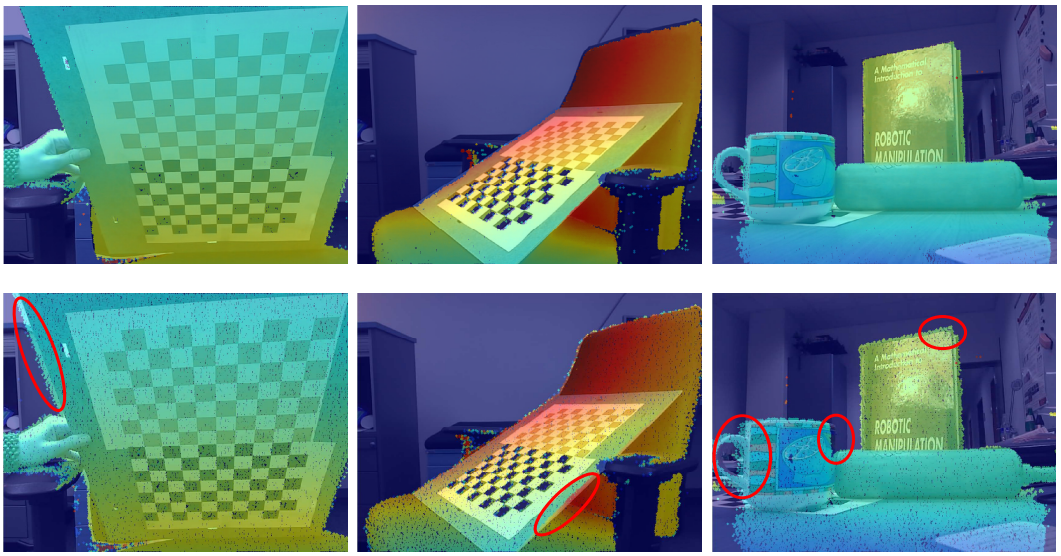


FIGURE 2.8: Examples of the alignment between depth and RGB data for Intel Creative Gesture Camera, obtained using our calibration (top row) and the manufacturer calibration (bottom row); manufacturer calibration often does not give a perfect alignment between depth and RGB image.

estimate mean error and standard deviation — m_d, σ_d. Since the plane pose is evaluated, we provide the results for the case when the pose is used instead of depth measurements to estimate the reprojection error in m_{conf}, σ_{conf}. It can be seen that the reprojection error is reduced by using depth data in the calibration procedure, and further reduced by introducing depth correction.

The better reprojection quality of the **CAL-DC** is also visually recognizable. In Figure 2.8 the alignment achieved by our calibration is compared with the manufacturer-provided mapping. It is easy to notice that in case of manually performed calibration the borders are preserved much better than in the case of manufacturer mapping.

Angle evaluation based on depth measurements: In this experiment, we collect a set of images of two perpendicular planes, in different positions relative to the camera. We manually select plane regions on each image. After selecting the regions, a plane is fitted to each region, by firstly computing the 3D point cloud, and then using SVD decomposition of the stacked 3D points to compute the plane normal as the vector corresponding to the smallest singular value. Then, the angle between two estimated normals is computed. The results are presented in Table 2.4.

TABLE 2.4: Angle $(m_{angle}, \sigma_{angle})$ and relative distance $(m_{rdist}, \sigma_{rdist})$ estimated error (mean and standard deviation of angle error given in degrees, for distance error — in mm).

	m_{angle}	σ_{angle}	m_{rdist}	σ_{rdist}
MFR	5.32	3.45	17.49	10.89
CAL	4.88	3.22	15.72	10.74
CAL-D	4.10	2.89	14.50	10.89
CAL-DC	3.35	2.50	13.69	10.42

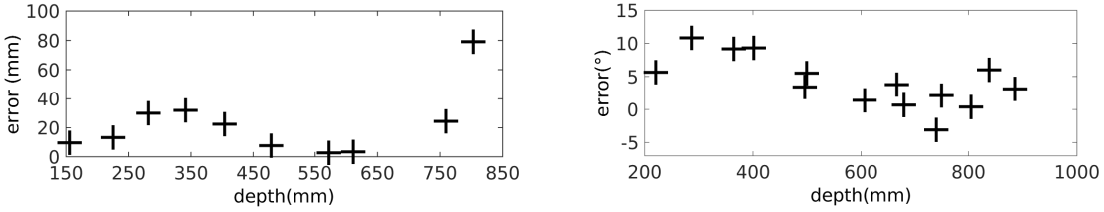


FIGURE 2.9: Error in relative distance measurements and angle measurements depending on the distance to the parallel planes.

Relative depth evaluation: It is impossible to obtain ground truth distance between the sensor and an object (since the exact position of the sensor is unknown). Therefore, we estimate the relative distance between the parallel planes in 3D and compare it to the ground truth distance. The results are reported in Table 2.4.

The structure of the error before the calibration procedure is clear to see in Figure 2.9. Unfortunately, as can be seen from the results, it was impossible to completely remove the depth bias, however, it was reduced by 21% in comparison with manufacturer calibration for the distance measurements and by 37% for the angle measurements.

2.7.2 Microsoft Kinect 2

The Microsoft Kinect2 [4] sensor appeared recently and employs time-of-flight technology as well. It has the resolution of the time-of-flight sensor of 512×424 and of the color sensor of 1920×1080; the admissible depth range is between 500mm and 4500mm. Firstly, we will investigate the properties of the depth sensor and then we compare factory calibration, standard camera calibration and the proposed enhancements to the calibration algorithm, that includes depth information into the calibration procedure.

TABLE 2.5: Estimated calibration parameters for Kinect2 Camera.

	MFR	**CAL**	**CAL-D**	**CAL-DC**
$K_{rgb} : f_x, f_y$	–	$1053.7, 1059.26$	$1053.7, 1059.26$	$1053.7, 1059.26$
c_x, c_y	–	$991.57, 526.45$	$991.57, 526.45$	$991.57, 526.45$
$K_{tof} : f_x, f_y$	$366.81, 366.81$	$368.93, 369.51$	$366.12, 366.66$	$369.11, 369.74$
c_x, c_y	$259.19, 207.73$	$266.58, 200.38$	$267.89, 203.77$	$264.88, 200.11$
$\boldsymbol{k}_{tof} : k_1, k_2$	$0.084, -0.268$	$0.121, -0402$	$0.106, -0.374$	$0.121, -0.395$
k_3, k_4, k_5	$0, 0, 0.103$	$0, 0, 0.245$	$0, 0, 0.23$	$0, 0, 0.22$

	CAL	**CAL-D**	**CAL-DC**
$\boldsymbol{k}_{rgb} : k_1, k_2$	$0.067, -0.063$	$0, 067, -0.064$	$0, 067, -0.064$
k_3, k_4, k_5	$-0.005, -0.004, 0$	$-0, 005, 0.004, 0.00$	$-0, 005, 0.004, 0.00$
$\boldsymbol{t}_\Delta$	$51.14, -0.36, -4.53$	$51.06, -0.25, 4.53$	$51.11, -0.30, -4.52$
R_Δ	$\begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1 & 0.00 \\ 0.00 & 0.00 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.99 & 0.00 & 0.01 \\ 0.00 & 0.99 & 0.01 \\ -0.01 & -0.01 & 0.99 \end{pmatrix}$	$\begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1 & 0.00 \\ 0.00 & 0.00 & 1 \end{pmatrix}$

The estimated parameters of the Microsoft Kinect 2 are presented in the Table 2.5; Kinect2 SDK allows to access the ToF sensor intrinsic parameters only.

Angle evaluation based on depth measurements: Depth sensor parameters, provided by the manufacturer for the Kinect2, give quite good results on the angle evaluation test, while standard calibration is not able to achieve the same results; however, adding depth into the calibration improves the angle estimation (see Table 2.6). As can be seen, in this experiment the manufacturer's calibration delivers the best results, although including depth information into calibration improves the estimate. We assume that better results of the manufacturer calibration, can be the result of better knowledge of the sensor properties. It is a question for further studies whether using more data for the calibration allows to improve the results of our method.

TABLE 2.6: Angle ($m_{angle}, \sigma_{angle}$) estimated error (mean and standard deviation, for angle the results are given in degrees, for distance error — in mm).

	m_{angle}	σ_{angle}
MFR	0.42	0.30
CAL	0.63	0.69
CAL-D	0.56	0.73
CAL-DC	0.96	0.97

TABLE 2.7: Comparison between depth-color image alignment provided by manufacturer, our calibration without depth correction and with depth correction; the results are reported in pixels.

	m_d	σ_d	m_{conf}	σ_{conf}
MFR	3.80	1.85	–	–
CAL	3.28	1.25	3.04	1.18
CAL-D	3.35	1.23	3.18	1.18
CAL-DC	3.23	1.22	3.04	1.21

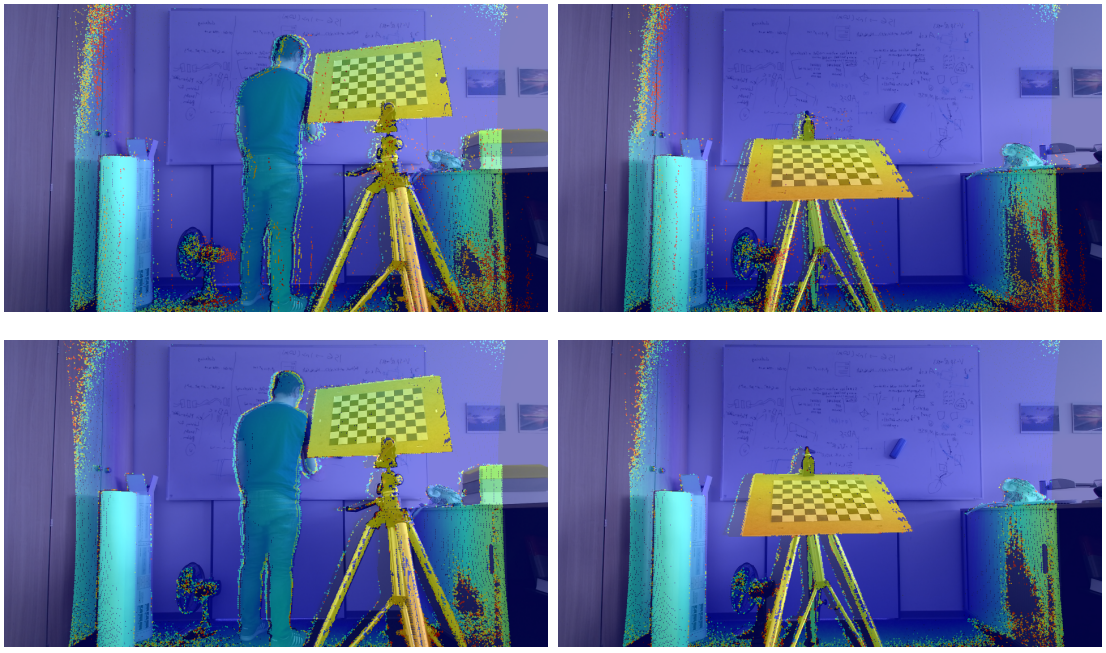


FIGURE 2.10: Examples of the alignment between depth and RGB data for Kinect2, obtained using our calibration (top row) and the manufacturer calibration (bottom row); here is it hard to see the difference between two calibrations; note, that the amount of flying pixels is greater for our calibration, since we do not filter them out explicitly, which is partially done during mapping; however, this effect is not related to the calibration quality itself.

Reprojection error from the depth to the color image: Kinect2 SDK provides a look up table-based mapping between a depth and a color image. In this experiment, the **CAL-DC** method outperforms the manufacturer-provided calibration, as seen in the Table 2.7. Some examples of image alignment achieved by using the manufacturer-provided calibration and the calibration algorithm are given in Figure 2.10. It can be seen, that both manufacturer calibration and our calibration algorithm allows to provide almost perfect alignment between the depth and the RGB data.

To conclude, for Kinect2 camera our calibration produces the results, roughly equivalent to manufacturer calibration, however, we provide the full set of parameters of both sensors.

2.8 Conclusions and discussion

Due to recent developments in time-of-flight imaging, many affordable time-of-flight sensors appeared in the market recently. Depth estimates provide great hints for various applications in computer vision, and therefore knowledge about the sensor properties is often required.

In this work, we in fact rely more on the depth data then on the color data for tackling the challenging problem of hand pose estimation from a single frame. To obtain the 3D information from the depth images, as well as to obtain the mapping between a color image and a depth image, knowledge of the sensor's calibration parameters is required.

Therefore, as a first step, we presented a method for time-of-flight sensor calibration, that accounts for the systematic bias in sensor's depth estimates and allows to partially correct them. We evaluated the proposed method using two popular low cost time-of-flight sensors, namely, the Intel Creative Gesture Camera [2] and the Microsoft Kinect2 [4], and showed, that for the first sensor we are able to obtain better calibration quality than the manufacturer and the full set of calibration parameters, and for the second sensor the quality of the calibration is comparable to manufacturer calibration, while we find the corresponding transformations instead of using a look-up table, provided by the Kinect2 SDK.

Chapter 3

Model-based hand pose estimation

3.1 Introduction

Often rigid and articulated object pose estimation and tracking problems are tackled using model-based (generative) approaches [18, 37, 94, 118]. These approaches aim at modeling (generating) the observation, usually by using a prior knowledge, such as the geometry of the object in question, and its parametrization.

The best parameter values are then found in terms of the fit between the generated prediction and the observation, usually by searching the parameter space using an optimization method. The quality of the fit between the generated prediction and the observation is measured in terms of some energy function. Depending on the model representation and the observation structure, either local optimization methods (such as gradient descend, as well as more advanced methods [80, 86]) or sampling-based methods [58, 63] are used to optimize the energy, depending on the parameters.

In general, model-based approaches are shown to obtain highly precise results, given that there is enough information about the object available and the energy models fit well enough. One of the properties of the pose estimation is that the energy function is usually non-convex, which makes is harder to optimize. The problem of rigid object pose estimation is considered to be easier, since the pose of an object is represented by 6 degrees of freedom (DoF) only. In case of articulated object pose estimation, the parameter space dimensionality can be

arbitrarily high, depending on the model used, and the energy function is much more complex with many local minima. Consequently, the optimization result in case of a local optimization method in general depends on the starting point of the optimization, while sampling-based methods require a lot of computational power as the dimensionality of the search space grows.

In general, for a generative approach, it is required, firstly, to define a suitable model with its parametrization; secondly, depending on the model and the energy function structure, a suitable optimization method is selected. In case of human or hand pose estimation, the model often consists of a set of connected locally rigid parts or a surface mesh, attached to a deformable skeleton. Depending on the concrete applications, rigid parts can be, for example, defined as cylinders or other geometric shapes [118, 127]. In case of a human hand, it is more convenient and intuitive to model it using a surface mesh, connected with a skeleton, that defines the pose. To represent the pose, different parametrizations, such as Euler angles, quaternions or exponential mapping can be used. In the next sections we will describe the model as well as the parametrization properties in detail.

3.2 Related work

Pose estimation of a human hand, although it has less DoFs then the full human body, represents a big challenge for pose estimation algorithms, especially in the case of a single camera setting. This happens due to several factors: while the human body is usually assumed to be approximately vertical, the hand can be seen in almost any position and orientation in an image, making even global pose estimation a tough problem. Additionally, the rigid part of the hand (palm) is relatively small in comparison with other parts (fingers) and often gets fully occluded, while the torso of a human body is large in comparison to other body parts and is normally at least partially visible in the image. The latter also causes difficulties for hand detection, since the unconstrained space of pose parameters significantly increases the appearance variation. Furthermore, fingers of a hand have almost identical appearance, while human upper and lower limbs are distinguishable.

There exists extensive literature on hand pose estimation; in general, all approaches can be divided into discriminative and generative approaches; the latter group can be further divided into the subgroup that uses sample-based optimizations methods and variations of the non-rigid iterative closest point (ICP) algorithm [14, 136]. Discriminative approaches often transform an image into a more

convenient representation and learn a classifier to differentiate between a discrete set of poses. In general, discriminative approaches have a set of limitations: firstly, they can only be used to distinguish between a limited number of poses; moreover, discrimination power of these methods decreases with an increasing number of poses, since the poses become more similar to one another; secondly, they require creation of a training set (or a template database) that strongly depends on the experimental setup.

Yet another type of approaches is based on template-matching, where an observation is matched against a predefined set of templates. The approaches, based on template matching, work as follows: firstly, a database of possible hand poses is created and each input image is matched against each image in the database using some image similarity measure [117]. This method was recently extended to incorporate depth data in [29].

Recently, more works appeared concentrating on a generative approach for hand pose estimation and tracking. In case of a single image, one often constrains the number of parameters to a subset of the 26 full DoF [118], or strong assumptions about the image are made [16, 25]. Several types of cues are used to evaluate the model, such as edges, optical flow [10], silhouettes, shading [26], color gloves [128], etc.

After the appearance of consumer depth cameras, such as the Kinect [3], the estimation of the full DoF hand pose became much more feasible, especially in the tracking scenario; in [94], particle swarm optimization was used to track a hand by generating a set of pose proposals, rendering the hand model in the corresponding pose and comparing it with the observation. Furthermore, since depth images can be transformed into 3D point clouds, they provide much more information than, for example, edges, and ICP-based approaches produce much more reliable results.

Unlike the discriminative approaches, generative approaches allow estimation of an infinite set of poses. However, model-based methods are usually computationally intensive and usually require an initial assumption about the hand pose for initialization. We address the latter problem by using several cues for pose initialization, such as fingertips and palm detections; we then use a top-down approach for pose refinement, firstly fitting global position and rotation, and then fitting the correct finger positions.

In Section 3.3 we describe the full DoF hand model, consisting of the mesh, attached to the skeleton, which is parametrized using exponential mapping, and

connected to the mesh using linear blend skinning. In subsequent sections we give a detailed description of our modification of a non-rigid ICP algorithm for hand pose estimation. In the last Section of this chapter, we evaluate the proposed method quantitatively on synthetic data and on the real-world Dexter dataset [116], collected using the Intel Creative depth sensor [2].

3.3 Hand model

The hand model consists of the polygonal mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$, where $\mathcal{V} = \{v_i\}_1^V$ is the set of all vertices of the mesh, $\mathcal{F}$ is the set of all faces and the skeleton $\mathcal{S} = \{B_i\}_{i=1}^N$, consisting of the bones B_i, connected by a parent-child relationship into kinematic chains. The pose of the skeleton is determined by a set of joint angles $\boldsymbol{\theta}$, that influence the bones positions, while the global hand pose is determined by 6-dimensional parameter vector $\boldsymbol{\xi}_0$, the meaning of which will be explained later.

3.3.1 Exponential mapping

Firstly, we formally define the notion of a rigid body: a rigid body is an object such that the distance between any two points of the object remains fixed, regardless of any motions of the object or forces exerted on the object. Rigid motion of the object is a continuous movement such that the distance between any two points of the object remains fixed and the cross product is preserved [90]. Rigid motion can be represented as rotation and translation, applied on all points of the rigid body simultaneously.

Rigid rotation: One can think about rotation as a change of orientation between the fixed (world) coordinate system and the coordinate system, attached to the rigid body. Every rotation in $\mathbb{R}^3$ can be represented using a rotation matrix $R \in SO(3)$ where $SO(3)$ is the space defined as: $SO(3) = \{R \in \mathbb{R}^{3\times3} : RR^T = I, \quad \det R = 1\}$. Every rotation further can be represented as a turn around an axis $\boldsymbol{\omega} \in \mathbb{R}^3$ by an angle θ, which is called axis-angle representation. It can be shown, that these two representations are equivalent and can be converted to one another.

We further introduce additional notation to neatly describe exponential mapping in the subsequent sections. The cross product between two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$

can be as well expressed through matrix multiplication by replacing $\boldsymbol{a}$ by the corresponding matrix $\widehat{a}$ as follows:

$$\boldsymbol{a} \times \boldsymbol{b} = \widehat{a}\boldsymbol{b}, \quad \widehat{a} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \tag{3.1}$$

A matrix in the form of $\widehat{a}$ is called skew-symmetric; the space of all such matrices is denoted as $so(3) = \{S \in \mathbb{R}^{3\times3} : S = -S^T\}$. In case $\boldsymbol{a}$ is represented in homogeneous coordinates, Equation 3.1 still holds and the forth coordinate is not used. Then the rotation matrix $R(\boldsymbol{\omega}, \theta) \in SO(3)$, representing rotation around axis $\boldsymbol{\omega}$ by an angle θ can be expressed as matrix exponential:

$$R(\boldsymbol{\omega}, \theta) = e^{\widehat{\omega}\theta} = I + \widehat{\omega}\theta + \frac{(\widehat{\omega}\theta)^2}{2!} + \frac{(\widehat{\omega}\theta)^3}{3!} + \dots \tag{3.2}$$

Furthermore, closed-form formula exists for $e^{\widehat{\omega}\theta} \in \mathbb{R}^{3\times3}$, called **Rodrigues' formula**:

$$e^{\widehat{\omega}\theta} = I + \widehat{\omega}\sin\theta + \widehat{\omega}^2(1 - \cos\theta) \tag{3.3}$$

Equation (3.2) provides mapping $so(3) \to SO(3)$. The inverse transformation exists as well, however, it is not single-valued.

General rigid motion: A general rigid motion can be decomposed into rotation and translation. It can be also thought of as relative position and orientation change between the world coordinate frame and the body coordinate frame and can be represented by a transformation matrix $T \in \mathbb{R}^{4\times4}$, that is applied to a point by matrix multiplication of the transformation matrix to the point in homogeneous coordinates. The transformation matrix has the following form:

$$T = \begin{pmatrix} R & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} \tag{3.4}$$

where $R \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$; we denote the space of all rigid transformations in the form (3.4) by $SE(3)$. Next, we introduce the notion of twist $\widehat{\xi} \in \mathbb{R}^{4\times4}$:

$$\widehat{\xi} = \begin{pmatrix} \widehat{\omega} & \boldsymbol{v} \\ 0 & 0 \end{pmatrix} \tag{3.5}$$

A twist as well allows to parametrize general rigid body motion and analogous to $so(3)$ the set of all matrices in the form (3.5) is denoted by $se(3)$. The first type of motion a twist describes is the rotation of a point $\boldsymbol{p}$ around the axes $\boldsymbol{\omega}$, where $\boldsymbol{\omega}$ might not pass through the origin of the coordinate system and its position

is characterized by a point $\boldsymbol{q}$ on the axes. Note, that now we use homogeneous coordinates, i.e. $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^4$. In this case, $\boldsymbol{v}$ in Equation (3.5) is defined as $\boldsymbol{v} = -\boldsymbol{\omega} \times \boldsymbol{q}$. The second type of motion is pure translation along $\boldsymbol{v}$. Since no rotation takes place, $\boldsymbol{\omega} = 0$.

Two operators are defined for twists: the operator $^\vee$ converts the twist representation to the twist coordinates, while the operator $^\wedge$ denotes the inverse conversion:

$$\widehat{\xi}^\vee = \begin{pmatrix} \widehat{\omega} & \boldsymbol{v} \\ 0 & 0 \end{pmatrix}^\vee = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{pmatrix} \in \mathbb{R}^6, \quad \boldsymbol{\xi}^\wedge = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{pmatrix}^\wedge = \begin{pmatrix} \widehat{\omega} & \boldsymbol{v} \\ 0 & 0 \end{pmatrix} \tag{3.6}$$

The exponential mapping of a twist represents a transformation matrix $T \in SE(3)$, in the same way as rotation matrix is represented by exponential mapping of the axis-angle representation:

$$T = e^{\widehat{\xi}\theta} = I + \widehat{\xi}\theta + \frac{(\widehat{\xi}\theta)^2}{2!} + \frac{(\widehat{\xi}\theta)^3}{3!} + \cdots \tag{3.7}$$

Here θ denotes the rotation angle (as in case of pure rotation) and the amount of translation. The closed form expression for the matrix exponential $e^{\widehat{\xi}\theta}$ can be derived. In case $\boldsymbol{\omega} \neq 0$

$$e^{\widehat{\xi}\theta} = \begin{pmatrix} e^{\widehat{\omega}\theta} & (I - e^{\widehat{\omega}\theta})(\boldsymbol{\omega} \times \boldsymbol{v}) + \boldsymbol{\omega}\boldsymbol{\omega}^T\boldsymbol{v}\theta \\ 0 & 1 \end{pmatrix} \tag{3.8}$$

Otherwise, if $\boldsymbol{\omega} = 0$, which corresponds to the case of pure translation:

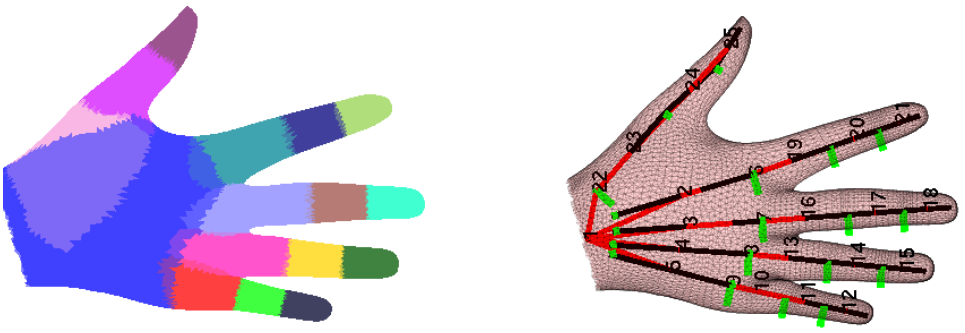
$$e^{\widehat{\xi}\theta} = \begin{pmatrix} I & \boldsymbol{v}\theta \\ 0 & 1 \end{pmatrix} \tag{3.9}$$

The Equations (3.8),(3.9) define a map $se(3) \to SE(3)$; as in case of $so(3) \to SO(3)$, the inverse mapping exists as well, but is not single-valued.

Consequently, given a point $\boldsymbol{p}(0) \in \mathbb{R}^4$ on a rigid body, that is rotated around an axes $\boldsymbol{\omega}$, which passes through the point $\boldsymbol{q}$, the new position of the point $\boldsymbol{p}(\theta)$ is then defined as:

$$\boldsymbol{p}(\theta) = e^{\widehat{\xi}\theta}\boldsymbol{p}(0), \quad \boldsymbol{\xi} = \begin{pmatrix} -\boldsymbol{\omega} \times \boldsymbol{q} \\ \boldsymbol{\omega} \end{pmatrix} \tag{3.10}$$

Kinematic chains In the previous paragraphs we defined the parametrization of a general rigid motion of a single rigid body. We now extend it to the case, when several rigid bodies are attached to each other with revolute joints. Consequently, each rigid body has its own coordinate frame. Given, that all the rigid bodies in the chain are connected, the movement of the body that is closer to the root affects



(a) (b)

FIGURE 3.1: (a) Hand mesh model; color shows different hand parts; the parts correspond to the rigid structures of the hand and are attached to the corresponding bones (except palm region, that is completely rigid). (b) The rigged skeleton with the joints' rotation axis.

all the subsequent rigid bodies. Knowing the motion of each part of the chain, we are interested in the positions of each joint in the world coordinate system, or equivalently in the final transformation between the world coordinate system and the coordinate system, attached to the i-th rigid body in the chain.

Let the kinematic chain consist of n rigid bodies; the joints are parametrized by the rotation angles $\theta_i, i = 1 \dots n$. Then it can be shown, that individual joint motions can be combined into the forward kinematic map, that depends on the parameters of the kinematic chain $\theta_1, \dots, \theta_n$; the forward kinematic map transforms a point $\boldsymbol{p}(\mathbf{0})$ to point $\boldsymbol{p}(\boldsymbol{\theta})$, $\boldsymbol{\theta} = \begin{pmatrix} \theta_1 & \theta_2 & \dots & \theta_n \end{pmatrix}^T$ as follows:

$$\boldsymbol{p}(\boldsymbol{\theta}) = e^{\widehat{\xi}_1 \theta_1} \dots e^{\widehat{\xi}_n \theta_n} \boldsymbol{p}(\mathbf{0}) \tag{3.11}$$

3.3.2 Hand deformation parametrization

Our hand model consists of a hand mesh $\mathcal{M}$ and a 26 DoF skeleton $\mathcal{S}$ rigged into the mesh (see Figure 3.1), using [11].

Skeleton The skeleton consists of N bones B_i connected by a parent-child relation with joints. Each bone, except the root bone, has a single parent bone. Since some of the hand joints can have multiple degrees of freedom (Figure 3.1), to achieve uniform representation we augment joints having two or more degrees of freedom by the corresponding number of bones, each having exactly one DoF.

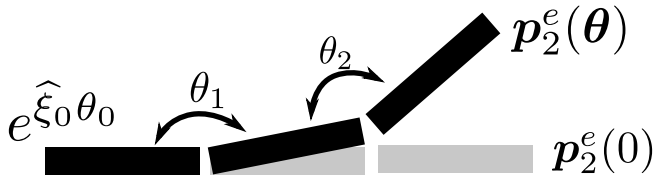


FIGURE 3.2: Kinematic chain.

That is, the hand skeleton is constructed in a way that each joint except the root joint has exactly one degree of freedom. The root joint has 6 DoF (3 DoFs representing translation and 3 of them representing rotation), while the pose of the hand is parametrized by 20 joint angles.

Let $\boldsymbol{p}_i^s$ and $\boldsymbol{p}_i^e$ be the beginning and the end of the bone B_i respectively. We denote the rotation axes, corresponding to the bone B_i and located at the start of the bone $\boldsymbol{p}_i^s$, by $\boldsymbol{\omega}_i$. This way, each bone has its own coordinate system, formed by $(\boldsymbol{p}_i^e - \boldsymbol{p}_i^s, \boldsymbol{\omega}_i, \boldsymbol{\omega}_i \times (\boldsymbol{p}_i^e - \boldsymbol{p}_i^s))$.

To represent the transformation of the skeleton, we use the exponential mapping parametrization of kinematic chains (see Section 3.3.1 for details). Exponential mapping parametrization allows to neatly express the Jacobian of the mesh vertices with respect to the skeleton parametrization, which is a very useful property for the estimation of pose parameters $\boldsymbol{\theta}$ using gradient-based local optimization methods, such as Levenberg-Marquardt algorithm [80].

Firstly, the transformation of the point $\boldsymbol{p}_i^e$ by angle θ_i around the rotation axes $\boldsymbol{\omega}_i$, placed at $\boldsymbol{p}_i^s$, is described by $T_i = e^{\widehat{\xi}_i\theta_i}$, where $\boldsymbol{\xi}_i = \begin{pmatrix} -\boldsymbol{\omega}_i \times \boldsymbol{p}_i^s \\ \boldsymbol{\omega}_i \end{pmatrix}$.

Consequently, given a kinematic chain (Figure 3.2) , consisting of $B_0, \ldots, B_n$, the transformation of the bones in this kinematic chain with angles $\boldsymbol{\theta}_i = \Big(\theta_0, \ldots, \theta_n\Big)^T$ is described as following:

$$\boldsymbol{p}_i^s(\boldsymbol{\theta}) = e^{\widehat{\xi}_0\theta_0} \ldots e^{\widehat{\xi}_{i-1}\theta_{i-1}} \boldsymbol{p}_i^s(0) = \left(\prod_{l=0}^{i-1} T_l\right) \boldsymbol{p}_i^s(0) = T_{i-1}^j p_i^s(0) \tag{3.12}$$

$$\boldsymbol{p}_i^e(\boldsymbol{\theta}) = e^{\widehat{\xi}_0\theta_0} \ldots e^{\widehat{\xi}_{i-1}\theta_{i-1}} e^{\widehat{\xi}_i\theta_i} \boldsymbol{p}_i^e(0) = \left(\prod_{l=0}^{i} T_l\right) \boldsymbol{p}_i^e(0) = T_i^j p_i^e(0) \tag{3.13}$$

The transformation of the rotation axis, associated with each bone, is described by:

$$\begin{pmatrix} \boldsymbol{\omega}_i(\boldsymbol{\theta}) \\ 0 \end{pmatrix} = e^{\widehat{\xi}_0\theta_0} \ldots e^{\widehat{\xi}_{i-1}\theta_{i-1}} \begin{pmatrix} \boldsymbol{\omega}_i(0) \\ 0 \end{pmatrix} = T_{i-1}^j \begin{pmatrix} \boldsymbol{\omega}_i(0) \\ 0 \end{pmatrix} \tag{3.14}$$

We further denote by Θ the set of all physically acceptable joint angles $\boldsymbol{\theta}$; in practice, Θ set is defined as a 20-dimensional hyperrectangle $\Theta = \{\boldsymbol{\theta} : l^i_{\boldsymbol{\theta}} \geq \theta^i \leq u^i_{\boldsymbol{\theta}}\}$.

Mesh and linear blend skinning: The hand model is represented using 3D triangulated mesh of a hand, consisting of $\mathcal{V} = \{\boldsymbol{v}\}_{j=1}^{V}$ vertices. The mesh vertices are posed by transferring the pose from the skeleton using linear blend skinning [81]. That is, for each mesh vertex $\boldsymbol{v}_j \in \mathbb{R}^4$, represented in homogeneous coordinates, a set of weights w_{ji} is computed, that links its transformation to the transformation of the skeleton bones B_i:

$$\tilde{\boldsymbol{v}}_j = \sum_i w_{ji} T_i^j \boldsymbol{v}_j, \quad \sum_j w_{ji} = 1 \tag{3.15}$$

where T_i^j is the bone transformation relative to the rigging hand configuration. The weights matrix $W = (w_{ji}) \in \mathbb{R}^{V \times N}$ can be computed, using for example [11].

Provided that a skeleton is transformed using global pose $\boldsymbol{\xi}_0$ and joint angles $\boldsymbol{\theta}$, we further denote the position of a vertex i of the mesh influenced by the skeleton, by $\boldsymbol{v}_i(\boldsymbol{\xi}_0, \boldsymbol{\theta})$; we omit the parameters to denote the vertex $\boldsymbol{v}_i$ of the mesh in its zero-pose $\boldsymbol{\theta}^{ini}$.

Vertex Jacobian: It can be shown, that given the transformation T_n in the twist coordinates, that affects the bone B_n, and depends on the bones that constitute a kinematic chain with indices $0, \ldots, n$, its Jacobian matrix $J(T_n) \in \mathbb{R}^{6 \times n+1}$ can be expressed as following:

$$J(T_n^j) = \begin{pmatrix} \boldsymbol{\xi}_0 & \boldsymbol{\xi}'_1 & \ldots & \boldsymbol{\xi}'_n \end{pmatrix} \tag{3.16}$$

Here $\boldsymbol{xi}'_i$ is ith joint twist $\boldsymbol{\xi}_i$, transformed by the adjoint transformation:

$$\boldsymbol{\xi}'_i = \mathrm{Ad}_{e^{\widehat{\xi}_0} \ldots e^{\widehat{\xi}_{i-1}\theta_{i-1}}} \boldsymbol{\xi}_i \tag{3.17}$$

Adjoint transformation $\mathrm{Ad}_T \in \mathbb{R}^{6 \times 6}$ is the transformation to map a twist from one coordinate system to another and is defined as:

$$\mathrm{Ad}_T = \begin{pmatrix} R & \widehat{\boldsymbol{t}}R \\ 0 & R \end{pmatrix}, \quad T = \begin{pmatrix} R & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} \tag{3.18}$$

Given linear skinning Equation 3.15, the Jacobian matrix $J(\tilde{v}_j)$ can be expressed by:

$$J(\tilde{\boldsymbol{v}}_j) = \frac{\partial \sum_i w_{ji} T_i^j \boldsymbol{v}_j}{\partial \boldsymbol{\theta}} = \sum_i w_{ji} \frac{\partial T_i^j}{\partial \boldsymbol{\theta}} \boldsymbol{v}_j = \sum_i w_{ji} J(T_i^j)^\wedge \boldsymbol{v}_j \tag{3.19}$$

where $J(T_i^j)^\wedge \boldsymbol{v}_j$ is interpreted as:

$$J(T_i^j)^\wedge \boldsymbol{v}_j = \begin{pmatrix} \boldsymbol{\xi}_0^\wedge \boldsymbol{v}_j & \boldsymbol{\xi}_1'^\wedge \boldsymbol{v}_j & \dots & \boldsymbol{\xi}_i'^\wedge \boldsymbol{v}_j \end{pmatrix} \tag{3.20}$$

Note, that the total displacement of the point $\tilde{\boldsymbol{v}}_j$ is then given by the expression:

$$\Delta \tilde{\boldsymbol{v}}_j = J(\tilde{\boldsymbol{v}}_j) \Delta \boldsymbol{\theta} \tag{3.21}$$

The root joint of the skeleton is in fact not parametrized by joint angles and its parameters can change arbitrary; therefore, the increment in point position depending on $\Delta\xi$ by definition is:

$$\Delta \boldsymbol{p} = \Delta \boldsymbol{\xi}_0^\wedge \boldsymbol{p} = \Delta \boldsymbol{v}_0 + \Delta \boldsymbol{\omega}_0 \times \boldsymbol{p} = \Delta \boldsymbol{v}_0 - \widehat{p} \Delta \boldsymbol{\omega} = \begin{pmatrix} I_{3\times 3} & -\widehat{\boldsymbol{p}} \end{pmatrix} \Delta \boldsymbol{\xi}_0 \tag{3.22}$$

Consequently, the expression $J(T_i^j)^\wedge \boldsymbol{v}_j$ can be re-written as

$$J(T_i^j)^\wedge \boldsymbol{v}_j = \begin{pmatrix} I_{3\times 3} & -\widehat{v}_j & \xi'^\wedge_1 \boldsymbol{v}_j & \dots & \xi'^\wedge_n \boldsymbol{v}_j \end{pmatrix} \tag{3.23}$$

Note, that the result of multiplication $\xi'^\wedge_i \boldsymbol{v}_j$ is 4×1 vector, but the 4th coordinate always equals zero and we omit it for convenience; $J(T_i)^\wedge \boldsymbol{v}_j$ is then a $3 \times (i+6)$ matrix. The final increment $\Delta \tilde{\boldsymbol{v}}_j$ is then expressed as

$$\Delta \tilde{\boldsymbol{v}}_j = J(\tilde{\boldsymbol{v}}_j) \begin{pmatrix} \Delta \boldsymbol{\xi}_0 \\ \Delta \boldsymbol{\theta} \end{pmatrix} \tag{3.24}$$

Equation (3.24) gives the expression for the increment of each point $\boldsymbol{v}_j$ of the mesh with respect to the skeleton parametrization, while Equation (3.23) defines the Jacobian matrix.

The Jacobian matrix, defined in Equation (3.23), is then used in a local optimization method to find the pose parameters. As will be shown in the subsequent Sections, the energy defining the quality of fit between the model and the observation can be expressed as the sum of the least squares; to find the pose parameters, Levenberg-Marquardt [80] optimization method is used as the method of choice for the non-linear unconstrained least square problems.

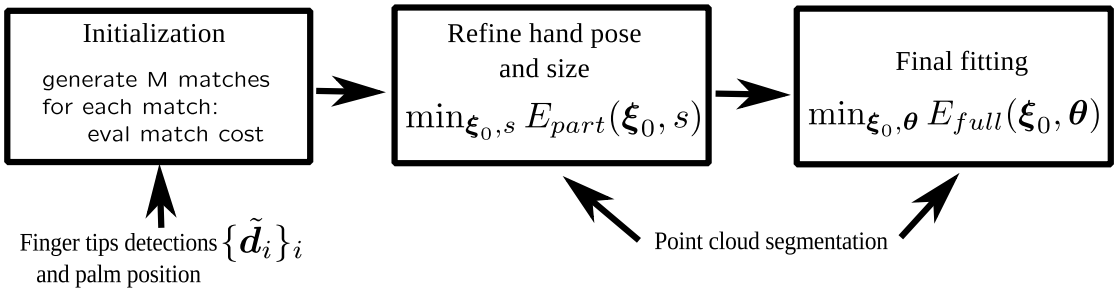


FIGURE 3.3: The work-flow of the algorithm.

3.4 Hand pose estimation algorithm

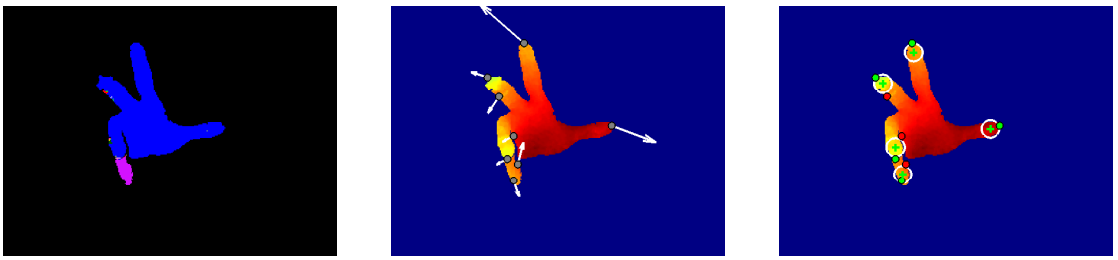
We formulate the problem of hand pose estimation from a single frame as a problem of finding the correct global pose, hand size and joint angles of the hand model, described in the previous section.

We constraint ourselves in this work to hand size estimation, but personalized hand model could potentially improve the pose estimation results. For obtaining a personalized hand pose, approaches presented in [9, 42] could be potentially used.

In general case, a cost function is defined describing how well the given parameters describe an observation. The cost function is optimized (minimized) to find the optimal parameters.

The cost function describing how well the model fits the observation given the parameters is usually non-convex. Although employing global optimization methods is possible, it is usually not feasible, given the dimensionality of the parameter space (as we explained earlier, a hand has 26 DoF plus the parameter describing scale). Therefore, local gradient-based optimization methods, such as Levenberg-Marquardt algorithm [80], are often used to minimize the cost function. However, found solution is only locally optimal and heavilty depends on the initialization.

Therefore, to minimize the susceptibility of the algorithm to local minimas, we separate hand size and global pose estimation from the full-DoF pose estimation, since a very good initialization for global pose and hand size is essential for obtaining correct estimation of the full-DoF pose. The algorithm consists of a number of steps (see Figure 3.3). At the first step, the initial position of a hand, as well as the approximate orientation is obtained using a number of heuristics, as described in Section 3.4.1; afterwards, the obtained pose and the hand size is refined by fitting a



(a) (b) (c)

FIGURE 3.4: Fingertips detection pipeline: (a) the point cloud is over-segmented using region growing algorithm; (b) for each sufficiently big region, fingertips candidates and their orientations are extracted, using geodesic distance extremas, combined with inner edges extraction; (c) geodesic extremas are filtered, using the fingertip shape prior and ensuring, that the fitted shape coincides with the extremas orientations, determined on the previous step.

hand palm region to the point cloud and the silhouette, using a local optimization algorithm (Section 3.4.2). Finally, the full-DoF hand pose is determined during the final optimization step using non-rigid modification of ICP algorithm (Section 3.4.3).

For hand pose estimation, we propose to use both the depth image $\mathcal{I}_d$ and the color image $\mathcal{I}_{rgb}$. From the color image, the hand silhouette $\mathcal{I}_s$ (such as $\mathcal{I}_s(x,y) = 1$ if a pixel (x,y) corresponds to a silhouette and $\mathcal{I}_s(x,y) = 0$ otherwise) is extracted, using, for example, a skin color segmentation algorithm [126]; in case color image is not provided, the silhouettes can be extracted from depth images as well); we denote by $\Omega = \{(x,y) \in R^2 : \mathcal{I}_s(x,y) = 1\}$ the set of all pixels, constituting the silhouette. The corresponding hand region of the depth image is transformed into a point cloud $\mathcal{P} = \{\boldsymbol{p}_i \in \mathcal{R}^3\}_i$ using camera's intrinsic parameters by inverting Equations (2.14)-(2.15) (Chapter 2).

3.4.1 Initialization step

For initialization, we first detect a number of hand features, specifically fingertips and the palm center. We firstly give an overview of the full algorithm for obtaining initial pose guess, and then discuss its parts in more details in Section 3.4.1.2 and 3.4.1.3.

Algorithm 1 Fingertips detection

Input: $\mathcal{I}_d$
Output: $\{\tilde{\boldsymbol{d}}_k\}_k$

while $\exists$ foreground pixels **do**
  $\Omega^{new} \leftarrow$ RegionGrowing($\mathcal{I}_d$) ▷ See Section 3.4.1.2
  $\{\boldsymbol{o}_i, \boldsymbol{d}_i\}_i \leftarrow$ GeodesicExtremas(Ω^{new}) ▷ see [96] for details
  $\{\tilde{\boldsymbol{d}}_k\}_k \leftarrow \{\tilde{\boldsymbol{d}}_k\}_k \cup$ FilterCandidates($\{\boldsymbol{o}_i, \boldsymbol{d}_i\}_i, \mathcal{I}_d, \Omega^{new}$)
end while

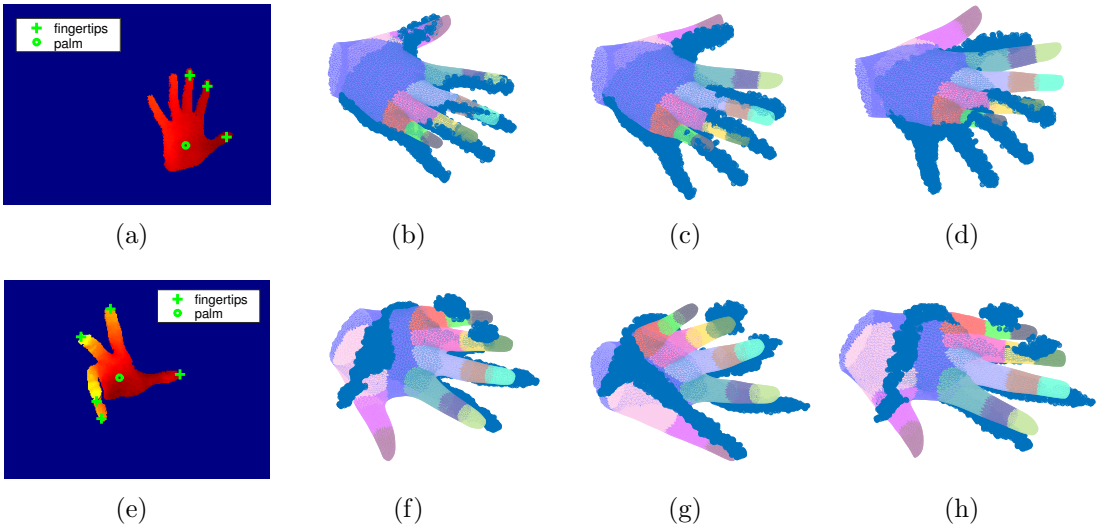
function GeodesicExtremas(Ω^{new}, N)
  for $n \leftarrow 1 \dots N$ **do** ▷ N is the number of extremas we expect
    $D \leftarrow$ Dijkstra(Ω^{new})
    $\boldsymbol{d}_n \leftarrow \text{argmax}_{(i,j)}\, D$
    $\boldsymbol{o}_n \leftarrow$ Backtrack($D, \boldsymbol{d}_n$)
  end for
  return $\{(\boldsymbol{d}_n, \boldsymbol{o}_n)\}$
end function

function FilterCandidates($\{\boldsymbol{o}_i, \boldsymbol{d}_i\}_i, \mathcal{I}_d, \Omega^{new}$)
  $C \leftarrow$ Contour($\Omega^{new}, \mathcal{I}_d$) ▷ get edge points $C = \{\boldsymbol{c}_j\}$ in Ω^{new}
  for $(\boldsymbol{o}_i, \boldsymbol{d}_i) \in \{\boldsymbol{o}_i, \boldsymbol{d}_i\}_i$ **do**
    $(r, \tilde{\boldsymbol{d}}) \leftarrow$ FitCircle($\boldsymbol{d}_i, C$) ▷ fit a circle, as described in Section 3.4.1.3
    $l \leftarrow \frac{1}{N(C)} \sum_j |r^2 - \|\tilde{\boldsymbol{d}} - \boldsymbol{c}_j\|^2|$ ▷ compute goodness-of-fit
    $a \leftarrow \frac{\langle \boldsymbol{o}_i, \boldsymbol{d}_i - \tilde{\boldsymbol{d}} \rangle}{\|\boldsymbol{o}_i\| \|\boldsymbol{d}_i - \tilde{\boldsymbol{d}}\|}$ ▷ angle between two vectors
    if $l < \tau_{loss}$ & $a > 0.5$ & $|r - R| < \varepsilon$ **then**
      $\{\tilde{\boldsymbol{d}}_k\}_k = \{\tilde{\boldsymbol{d}}_k\}_k \cup \tilde{\boldsymbol{d}}$ ▷ add a new detection to the detections set
    end if
  end for
  return $\{\tilde{\boldsymbol{d}}_k\}_k$
end function

3.4.1.1 Hand global pose initialization: overview

To detect fingertips, we first segment a point cloud using a modification of region growing algorithm to obtain initial hand regions (Section 3.4.1.2); we then find finger tips candidates and their orientations $\{(\boldsymbol{d}_i, \boldsymbol{o}_i) \in \mathbb{R}^2\}_i$ by finding geodesic extremas [96] for each region. Finally, we filter them and refine the positions of the detections (Section 3.4.1.3), therefore obtaining the final set of detections $\{\boldsymbol{d}_k\}_k$ (see the outcome of each step in Figure 3.4). The step-by-step description of the algorithm is provided in Algorithm 1.

We find the palm center using distance transform on the hand silhouette, as proposed in [64].



(a) (b) (c) (d)

(e) (f) (g) (h)

FIGURE 3.5: Initializations from matching detection to the finger tips; the leftmost image is the input image, while the other images are the proposed initializations with increasing cost.

As a next step, we use the fingertips and palm detections to determine approximate position and rotation of the hand. Since we do not know correspondences between the detected finger tips and those of the model fingers, we sub-sample possible matches. Each match j is a permutation of a subset of all detections, described as $i \rightarrow \pi^j(i)$, where i is the index of a detection and $\pi^j(i) \in \{1 \dots 5\}$ denotes a fingertip of the model. After fixing a match j we first find the global rotation and translation of the model. To evaluate each match, we check if the full DoF pose implied by the match is physically possible (i.e., does not violate the physical constraints on the hand's joint angles) by computing the cost of the pose:

$$E^j_{match}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) = \min_{\boldsymbol{\xi}_0, \boldsymbol{\theta} \in \Theta} \sum_i \|\tilde{\boldsymbol{d}}_i - \boldsymbol{p}^e_{\pi^j(i)}(\boldsymbol{\xi}_0, \boldsymbol{\theta})\|^2, \tag{3.25}$$

where $\boldsymbol{\xi}_0$ denotes the global pose of the hand and $\boldsymbol{\theta}$ are the joint angles, defining the pose. We then select the match j that delivers the smallest cost E^j_{match} and use the corresponding global pose parameters $\boldsymbol{\xi}^{ini}_0 = \text{argmin}_{\boldsymbol{\xi}_0, \boldsymbol{\theta}} E^j_{match}(\boldsymbol{\xi}_0, \boldsymbol{\theta})$ as an initialization for the subsequent optimization.

In practice, the number of matches can be reduced by making use of the fact that the fingertips can be ordered around the palm, assuming that the palm is facing the camera.

The resulting initializations obtained from different matches are shown in Figure 3.5. The proposed algorithm allows to determine hand orientation and position,

however, recently more advanced methods to obtain initial hand orientation were proposed, for example, based on random forests [133]. Another enhancement that can be potentially beneficial is to use a better cost function (3.25) to evaluate initial hand pose proposals.

3.4.1.2 Point cloud segmentation with region growing

Prior to obtaining the fingertips candidates, we segment the point cloud to obtain a set of point cloud regions, that correspond to continuous surfaces, since to correctly extract the geodesic extremas, a continuous surface is required.

We use a region growing-based segmentation algorithm on the depth images: a pixel is included into a current region if the two distance measures, (3.26) and (3.27), are smaller than the corresponding thresholds. We define empirical thresholds of 10mm for the Euclidean distance (3.26) and 30° for the angle (3.27) between normals.

$$d(\boldsymbol{p}_i, \boldsymbol{p}_j) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|, \quad \boldsymbol{p}_i, \boldsymbol{p}_j \in \mathbb{R}^3 \ , \tag{3.26}$$

$$d_n(\boldsymbol{p}_i, \boldsymbol{p}_j) = \angle(\boldsymbol{n}(\boldsymbol{p}_i), \boldsymbol{n}(\boldsymbol{p}_j)) = \arccos\langle \boldsymbol{n}(\boldsymbol{p}_i), \boldsymbol{n}(\boldsymbol{p}_j)\rangle \tag{3.27}$$

Here $\boldsymbol{n}(\boldsymbol{p}_i)$ and $\boldsymbol{n}(\boldsymbol{p}_j)$ are the normals to the point cloud at the corresponding points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. To compute the normals we use principal component analysis (PCA) on the neighborhood of each point as in [103].

After initial segmentation, we divide large regions (which usually correspond to the palm with multiple fingers) into smaller ones to separate the palm and the fingers. To do so, we employ morphological opening with a circle of a radius roughly corresponding to the radius of the fingers. The resulting segmentations are presented in Figure 3.6.

After the segmentation, we determine the palm as the largest region in the vicinity of the palm detection, and the rest of the points belong to the fingers. The point indices of the point cloud, corresponding to the palm, are denoted by I_{palm}, whereas the indices, corresponding to the i-th region, are denoted by I^i_{finger}; the union of all non-palm points is denoted by $I_{fingers} = \bigcup_i I^i_{finger}$.

Consequently, the silhouette can also be segmented into palm and fingers region. Let Ω_{palm} denote the set of all pixels, corresponding to the palm region, and $\Omega_{fingers}$ — the set of all pixels, corresponding to the finger region; consequently, $\Omega = \Omega_{palm} \bigcup \Omega_{fingers}$.

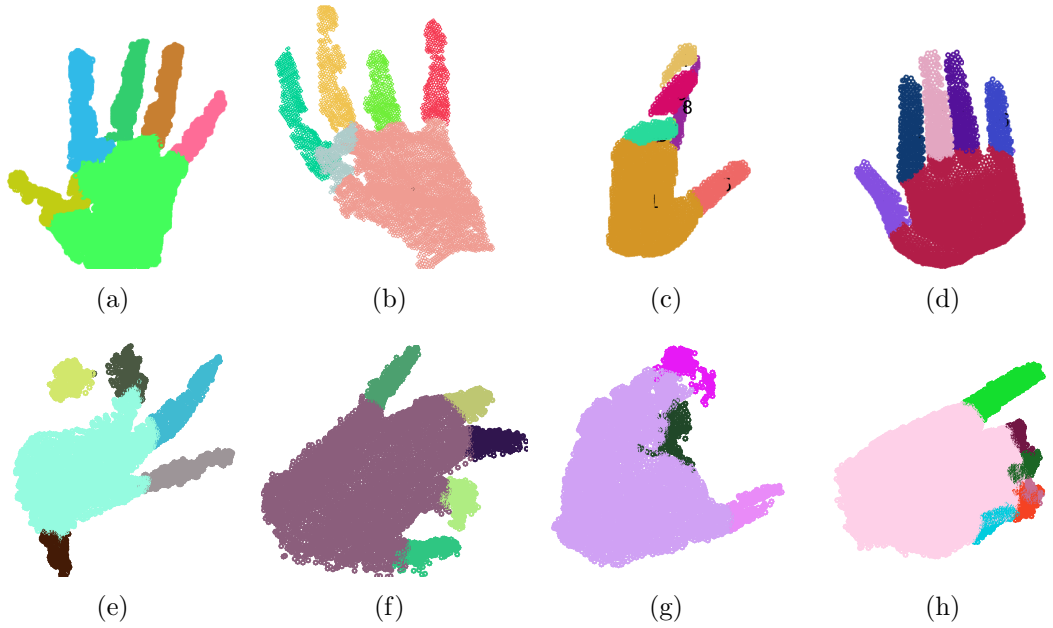


FIGURE 3.6: Top row: segmentation results for the real data produced by Kinect ((a) and (b)) and the artificial dataset ((c) and (d)); Bottom row: segmentation results for the real data produced by Intel Creative camera from MSR dataset and for Dexter dataset; (g) and (h) show failure cases.

Additionally, we define the corresponding segmentation for the vertex indices of the mesh $I^m = \{1, \ldots, V\}$: $I^m_{fingers} = \bigcup_i I^{m,i}_{finger}, I^m_{palm} \bigcup I^m_{fingers} = I^m$.

3.4.1.3 Fingertips filtering

Initially we obtain many fingertips detections, as shown on Figure 3.4(b). However, some of them do not correspond to the real fingertips, and therefore we filter them during an additional step, described in this subsection. We observed that the most of these false detection can be removed by estimating local hand contour curvature near the detection candidate and removing the candidates with small curvature. For each candidate fingertip detection $\boldsymbol{d}_i \in \mathbb{R}^2$ we extract the nearby contour $C_i = \{\boldsymbol{c}_j \in \mathbb{R}^2 : \|\boldsymbol{c}_j - \boldsymbol{d}_i\| < \tau_r\}$ of the finger and determine its local curvature by least square fitting of a circle with parameters $(r_i, \tilde{\boldsymbol{d}}_i)$:

$$(r_i, \tilde{\boldsymbol{d}}_i) = \underset{r, \tilde{\boldsymbol{d}}}{\operatorname{argmin}} \sum_{\boldsymbol{c}_j \in C_i} \left(\|\boldsymbol{c}_j - \tilde{\boldsymbol{d}}\|^2 - r^2 \right)^2 \tag{3.28}$$

If the radius r_i of the fitted circle corresponds to the expected finger radius, as well as if the fitting cost is sufficiently low, the fingertip candidate is classified as detection and we use $\tilde{\boldsymbol{d}}_i$ as a corrected fingertip location. Additionally we compute

the angle between vectors $\boldsymbol{d}_i - \tilde{\boldsymbol{d}}_i$ and $\boldsymbol{o}_i$ and filter out the detection, for which the angle is greater then 45°.

In general, we are only required to obtain some fingertip detections, since together with palm detection it allows to determine initial hand pose, as discussed in Section 3.4.1.

3.4.2 Hand pose and size refinement

As mentioned in Section 3.4.1, the initial global hand pose estimation, described in Section 3.4.1, is not accurate. Therefore, we firstly refine the global pose parameters $\boldsymbol{\xi}_0$ using a technique, similar in spirit to ICP optimization.

Let $\boldsymbol{p}_i \in \mathcal{P}$ denote a point from the hand point cloud, $\boldsymbol{v}_i \in \mathcal{V}$ denote a vertex of the model in the zero-pose $\boldsymbol{\theta}^{ini}$, $\Pr(\boldsymbol{v}_i) \in \mathbb{R}^2$ denote the projection of a model vertex onto the silhouette image $\mathcal{I}_s$. Furthermore, we denote by $\mu(\cdot) : i \mapsto \mu(i, I) \in I$ the match between the point with index i and the point with index $\mu(i, I)$ from the set of points I. In our case, the mapping is defined by finding the point in I, closest to the query point i with respect to Euclidean distance. The parameter vector has seven dimensions: $(\boldsymbol{\xi}_0, s)$, where $\boldsymbol{\xi}_0$ define hand position and rotation, and s defines scaling.

To do so, we minimize the following cost function, starting from initial global pose parameters $\boldsymbol{\xi}_0^{ini}$ (obtained in the previous step) and $s = 1$ as the initial value:

$$E_{part}(\boldsymbol{\xi}_0, s) = E_{palm}(\boldsymbol{\xi}_0, s) + \alpha E_{fingers}(\boldsymbol{\xi}_0, s) + \tag{3.29}$$

$$+ \beta E_{bg}(\boldsymbol{\xi}_0, s) + \lambda \mathcal{R}_{part}(s) \tag{3.30}$$

Here, the $E_{palm}(\boldsymbol{\xi}_0, s)$ and $E_{fingers}(\boldsymbol{\xi}_0, s)$ terms pull the model towards the point cloud, while $E_{bg}(\boldsymbol{\xi}_0, s)$ term restricts the changes by requiring the position and the size of the hand to fit inside the silhouette.

In Equation (3.30), E_{palm} is the error term responsible for the distance between the point cloud region, identified as palm, and the palm of the model:

$$E_{palm}(\boldsymbol{\xi}_0, s) = \frac{1}{|I_{palm}|} \sum_{i \in I_{palm}} \|\boldsymbol{p}_i - \boldsymbol{v}_{\mu(i, \tilde{I}^m_{palm})}(\boldsymbol{\xi}_0, s)\|^2, \tag{3.31}$$

For each point $\boldsymbol{p}_i$ of the point cloud we search for the nearest point $\boldsymbol{v}_{\mu(i, \tilde{I}^m_{palm})}(\boldsymbol{\xi}_0, s)$ on the visible part of the palm of the hand model (the indices of the vertices of the palm, visible from the camera, are denoted by $\tilde{I}^m_{palm}$). We use a kd-tree [13] on the points from $\mathcal{P}$ to accelerate point search.

The $E_{fingers}$ term is responsible for the distance between finger points in the point cloud and finger points of the model.

$$E_{fingers}(\boldsymbol{\xi}_0, s) = \frac{1}{|I_{fingers}|} \sum_{i \in I_{fingers}} \|\boldsymbol{p}_i - \boldsymbol{v}_{\mu(i, \tilde{I}^m_{fingers})}(\boldsymbol{\xi}_0, s)\|^2 \tag{3.32}$$

Note, that here $\mu(i, \tilde{I}^m_{fingers})$ denotes the closest visible vertex of the model on the fingers. Finally, the E_{bg} term defines the distance between the 2D silhouette of the hand, and the projection of the model:

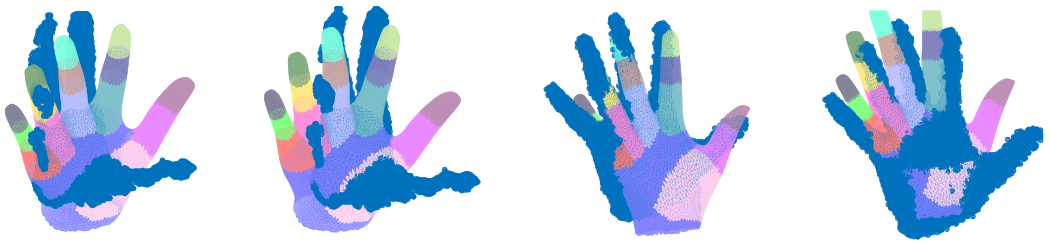
$$E_{bg}(\boldsymbol{\xi}_0, s) = \frac{1}{|I^m_{palm}|} \sum_{i \in I^m_{palm}} \min_{\boldsymbol{p} \in \Omega_{palm}} \|\mathrm{Pr}(\boldsymbol{v}_i(\boldsymbol{\xi}_0, s)) - \boldsymbol{p}\|^2, \tag{3.33}$$

where $\mathrm{Pr}(\boldsymbol{v}_i)$ is the projection of a model vertex $\boldsymbol{v}_i$ to the silhouette image.

Finally $\mathcal{R}_{part}(s) = (s-1)^4$ defines regularization on the scale parameter.

The cost function in Equation (3.30) is the sum of squared terms. The optimization procedure consists of three steps: firstly, the correspondences between model points and the point cloud are updated; secondly, a new value of the cost function $E_{part}(\boldsymbol{\xi}_0, s)$ and its Jacobian is computed using the obtained correspondences (see Section 3.3 and Equation (3.24)); thirdly, a step of Levenbeg-Marquardt is performed towards the minimum of the cost function.

After the optimization stage, we obtain global hand pose parameters $\boldsymbol{\xi}_0$ and an estimate of hand scale s (see Figure 3.7), that are used as initial global pose parameters for the optimization at the final step of the pose estimation algorithm — full DoF pose estimation.



(a) (b) (c) (d)

FIGURE 3.7: Hand fitting initialization ((a) and (c)) and refined hand pose and size ((b) and (d)); notice, that fingers are now better aligned with the point cloud.

3.4.3 Final hand pose estimation

On the previous step, we estimated hand pose and size, but the full pose parameters $\boldsymbol{\theta}$ are still unknown. In this stage, we find the full pose parameters $\boldsymbol{\theta}$, as well as further refine the global pose of the hand $\boldsymbol{\xi}_0$, so that the data is explained in the best way.

3.4.3.1 Pre-matching the fingers

The crucial part of the optimization procedure of full DoF pose estimation, described in this section, is finding the correct matches between the hand model and the point cloud. In a frequent case of an open hand, when more fingers are straitened or half-straitened, the false matches between the point cloud and the model can lead to a completely incorrect result even for this simple case. Therefore, prior to optimization, we pre-match extracted point cloud regions to the parts of the model and retain this match through optimization. For matching each region of the point cloud to a model part, we define a cost for each match as Euclidean distance between the centers of two segments. Each model segment can only be matched to zero or one point cloud segment. The problem can be mathematically written as follows:

$$\boldsymbol{\rho}^* = \underset{\boldsymbol{\rho}}{\operatorname{argmin}} \sum_i \|\boldsymbol{c}_i^{pc} - \boldsymbol{c}_{\rho_i}^{m}\|^2, \quad \boldsymbol{\rho} \quad \text{denotes a permutation of } \{1 \dots 5\} \tag{3.34}$$

where $\boldsymbol{c}_i^{pc}$ is the center of the i-th region of the point cloud and $\boldsymbol{c}_{\rho_i}^{m}$ is the center of the ρ_i-th region of the model. $\boldsymbol{\rho}^*$ is the resulting vector, which defines which region from a point cloud is matched to which finger; consequently, all coordinates of the vector are distinct. Note, that palm region is already known, so it is excluded from the matching procedure. The solution $\boldsymbol{\rho}^*$ is found using brute force search, since the number of possible combinations is low.

However, in case of more difficult poses, when not all fingers are visible or segmentable using a simple algorithm described in Section 3.4.1.2, this heuristic is not robust enough, and therefore we disregard segmentation during final optimization. The condition to discard the matching step is that the number of segmented regions is not equal to 6 (i.e., the number of fingers plus the palm region); in this case, no pre-matching is used and we rely on distance-based matches between the point cloud and the model.

3.4.3.2 Final optimization

The full pose of the hand is parametrized by 26 parameters: $(\boldsymbol{\xi}_0, \boldsymbol{\theta})$, where the initial value of $\boldsymbol{\xi}_0$ is found during the previous step, and $\boldsymbol{\theta}$ is a 20-dimensional vector of joint angles defining fingers' position.

To find the pose parameters, we optimize the following cost function:

$$E_{full}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) = E_{pc}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) + \zeta E_{bg}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) + \lambda \mathcal{R}_{full}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) \tag{3.35}$$

$$E_{pc}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) = \begin{cases} E_{palm}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) + \left(\sum_i E^i_{fingers}(\boldsymbol{\xi}_0, \boldsymbol{\theta})\right), \text{hand segmented} \\ \frac{1}{|I|} \sum_{j \in I} \|\boldsymbol{p}_j - \boldsymbol{v}_{\mu(j, \tilde{I}^{m,i})}(\boldsymbol{\xi}_0, \boldsymbol{\theta})\|^2, \text{otherwise} \end{cases} \tag{3.36}$$

The initial parameters $\boldsymbol{\xi}_0$ are obtained in the previous (refinement) step, and the optimization starts with the zero-pose $\boldsymbol{\theta}^{ini}$. In the equation above, $E_{pc}(\boldsymbol{\xi}_0, \boldsymbol{\theta})$ denotes the cost of the fit between the model and the point cloud. In case point cloud segmentation was successful, it is then used to reduce the number of false matches between the point cloud and the data; otherwise, the match is done based on the Euclidean distance only; $\tilde{I}^{m,i}$ here denotes the set of all visible vertices of the model. The $E_{palm}(\boldsymbol{\xi}_0, \boldsymbol{\theta})$ is defined in the same way as in Equation (3.31), and the $E^i_{fingers}(\boldsymbol{\xi}_0, \boldsymbol{\theta})$ term is defined as follows:

$$E^i_{fingers} = \frac{1}{|I^i_{finger}|} \sum_{j \in I^i_{finger}} \|\boldsymbol{p}_j - \boldsymbol{v}_{\mu(j, \tilde{I}^{m,\rho^*_i}_{finger})}(\boldsymbol{\xi}_0, \boldsymbol{\theta})\|^2 \tag{3.37}$$

Here $\mu(j, \tilde{I}^{m,i}_{finger})$ is the closest model point to the point j from $\tilde{I}^{m,\rho^*_i}_{finger}$ (visible vertices on the corresponding finger) and $|I^i_{finger}|$ is the number of indices in the set I^i_{finger}. In this way, smaller finger regions gain more weight and have more influence in the optimization.

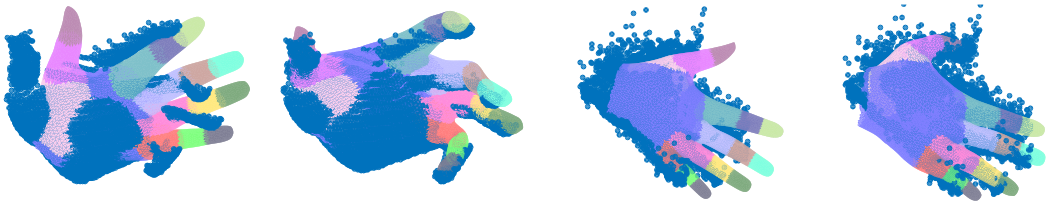
The last term is the background term, having the same meaning as the term defined in Equation (3.33), except that we observe the full silhouette instead of taking into account the palm region only:

$$E_{bg}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) = \frac{1}{|I^m|} \sum_{i \in I^m} \min_{\boldsymbol{p} \in \Omega} \|\Pr(\boldsymbol{v}_i(\boldsymbol{\xi}_0, \boldsymbol{\theta})) - \boldsymbol{p}\|^2, \tag{3.38}$$

Finally, we introduce regularization term on the pose parameters $\boldsymbol{\theta}$, that penalizes physically impossible configurations:

$$\mathcal{R}_{full}(\boldsymbol{\xi}_0, \boldsymbol{\theta}) = \|(\boldsymbol{\theta} - l_{\boldsymbol{\theta}})^+\|^4 + \|(u_{\boldsymbol{\theta}} - \boldsymbol{\theta})^+\|^4 \tag{3.39}$$

where $l_{\boldsymbol{\theta}}$ and $u_{\boldsymbol{\theta}}$ correspond to the lower and upper bounds on the physically possible joint angles $\boldsymbol{\theta}$ and $\boldsymbol{x}^+ = (\max(x^1, 0), \ldots, \max(x^N, 0)), \quad \boldsymbol{x} \in \mathbb{R}^N$.



(a) (b) (c) (d)

FIGURE 3.8: The global hand pose after refinement ((a),(c)) and the final hand pose after the final optimization step ((b),(d)); the first two images correspond to the synthetic data, while the second two images — to the real data, provided by Intel Creative Gesture Camera.

The cost function is again represented as a sum of squares of non-linear scalar functions, and therefore we use the same algorithm for constrained non-linear optimization, as in the previous section. The solution of the optimization problem delivers the full-DoF pose estimation of a hand.

Examples of the hand fits after the final optimization step are provided in Figure 3.8 and 3.15.

3.5 Evaluation

3.5.1 Fingertips detection evaluation

Firstly, we evaluate the proposed fingertip detector. We report the results on synthetic data, as well as on real data from the Dexter [116] dataset, using two metrics: precision and recall.

The results for fingertip detections are given in Table 3.1. As can be seen, the precision is very high, which means that the detector rarely makes type one errors. On the other hand, recall is significantly lower. Note, that recall is a pessimistic estimate in this case, since the fingertips are often marked, but are in fact not detectable, since they are occluded.

TABLE 3.1: Fingertips detector performance; the fingertip is considered detected, if it is within 7 pixels from the ground truth detection, which roughly correspond to the fingertip radius in the images.

seq.	precision	recall
synth. [94]	93.28	65.56
adbaddb [116]	68.23	42.45
fingercount [116]	62.48	36.53
fingerwave [116]	66.89	45.47
flexex1 [116]	71.81	48.66
pinch [116]	69.20	42.29
random [116]	61.28	42.56
tigergrasp [116]	63.43	42.95

3.5.2 Full DoF pose estimation evaluation

We provide the results of each step of the algorithm, and evaluate the performance gain due to splitting the optimization procedure, as well as using point cloud pre-segmentation, described earlier.

The baselines are the following:

- **INI** — initial pose guess, based solely on the detections;
- **REF** — pose guess after the refinement step, described in Section 3.4.2
- **FULL** — full pose estimation algorithm, described in Sections 3.4.1-3.4.3.
- **FULL-NSEG** — full pose estimation algorithm, but without fingers pre-matching step, described in Section 3.4.3.1.
- **FULL-NSEG-NREF** — full pose estimation algorithm, without pose refinement (Section 3.4.2) and fingers pre-matching steps (Section 3.4.3.1).

We provide the results for the last two baselines, namely **FULL-NSEG** and **FULL-NSEG-NREF**, to show the improvement due to separation of the global pose refinement and final pose estimation, as well as the gain due to point cloud pre-segmentation.

We evaluate the fit, using the following metric: average joint error $\bar{e}^f$ for each frame $f = 1 \dots F$, computed as Euclidean distance Δ_j^f between the ground truth

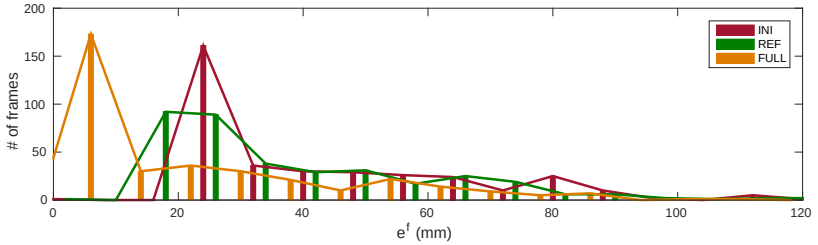


FIGURE 3.9: The error e^f through the entire sequence; note, that we do not use any temporal information in this case; we show the improvement with respect to ground truth from the initialization (**INI**) to the refinement (**REF**) and to the final pose estimation **FULL**.

joints positions $\boldsymbol{p}_j^f$ and the predicted positions $\hat{\boldsymbol{p}}_j^f$ for all available joints $j = 1 \dots J$:

$$\Delta_j^f = \|\boldsymbol{p}_j^f - \hat{\boldsymbol{p}}_j^f\| \tag{3.40}$$

$$e^f = \frac{1}{J}\sum_{j=1}^{J} \Delta_j^f \tag{3.41}$$

The distance is provided in millimeters.

To be compatible with [94], we employ m^f per-frame metric, which corresponds to the median of the error of all joints:

$$m^f = \text{med}\{\Delta_j^f, j = 1 \dots J\} \tag{3.42}$$

Furthermore, we provide the proportion $\rho(\tau)$ of the frames, where for all joints the error was less then a certain threshold τ (mm):

$$\rho(\tau) = \frac{\#\{f : m^f < \tau\}}{F} \tag{3.43}$$

3.5.3 Synthetic sequence

For the evaluation, we use synthetic sequence from [94]. The sequence contains 360 frames. The positions of all hand joints are provided for evaluation.

In the Figure 3.9, the average error, computed for **INI**,**REF** and **FULL** methods through the sequence is shown. It can be seen, that the error decreases from the **INI** method, to the **REF** and then decreases further to **FULL**, as expected. Further, we compare **FULL** with **FULL-NSEG** and **FULL-NSEG-NREF** to show the advantages of the proposed enhancement. The results are presented in Figure 3.10. Note, that for all baselines we use the same detections and the same

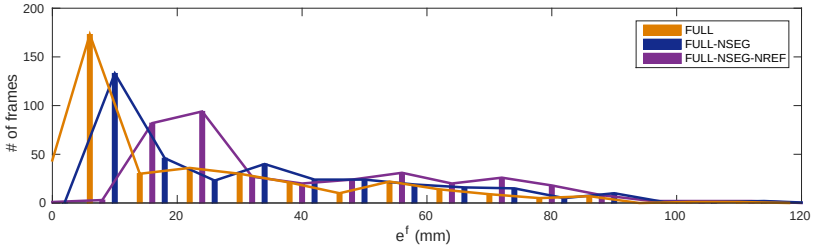


FIGURE 3.10: The error e^f through the entire sequence; we do not use any temporal information; we compare the pose estimation results of **FULL** with **FULL-NSEG** and further with **FULL-NSEG-NREF**.

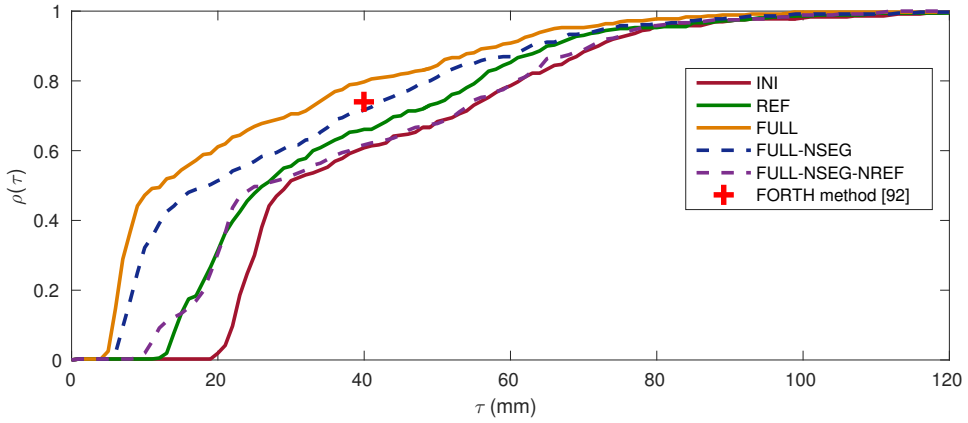


FIGURE 3.11: The error $\rho(\tau)$ through the entire sequence; we compare all baselines to show the effect of each of the steps.

initialization. Finally, all the baselines are compared using the second metric, $\rho(\tau)$, in Figure 3.11. As can be seen, we outperform the published baseline for this sequence [94], where 74% of the frames have the estimated pose deviation 40mm or less from the ground truth, even though the model used in the experiments is significantly different from the model used to produce synthetic data.

3.5.4 Evaluation on the Dexter dataset

We furthermore evaluate the algorithm on the Dexter dataset [116]. It consists of seven sequences, recorded using the Intel Creative Depth Sensor, as well as five RGB cameras and a structured light sensor. Each sequence is around $400 - 500$ frames long. In the experiments, we use only the data from the Intel Creative Depth Sensor. Further, the dataset contains annotations for the fingertip positions,

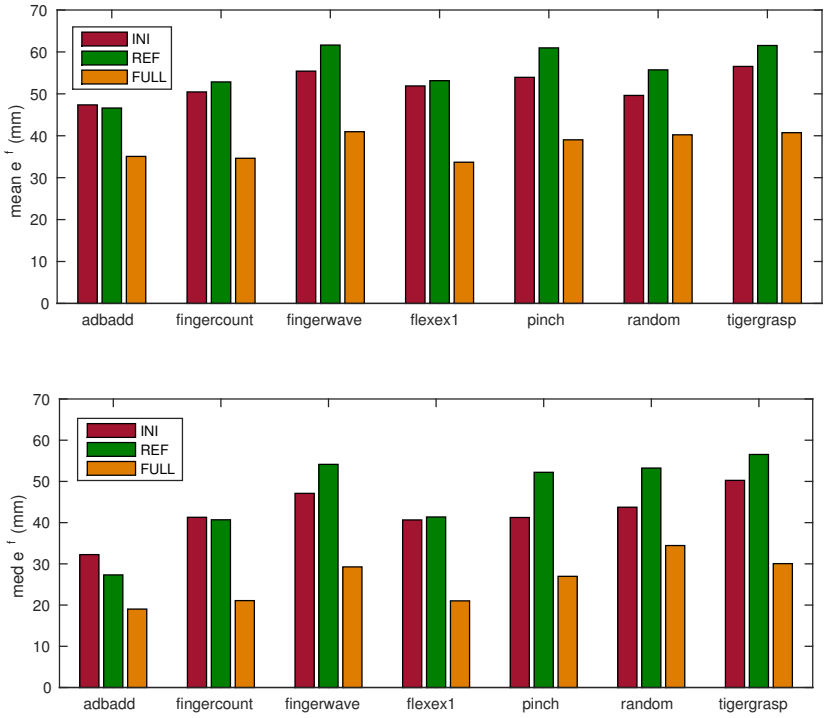


FIGURE 3.12: Comparison of the **INI**,**REF** and **FULL** baselines on the sequences of Dexter dataset; top graph shows average of e^f for the **INI**,**REF** and **FULL** parts of the method, while bottom graph reports the median of the e^f across frames.

as well as the palm center positions. The results for each sequence are shown in Figure 3.12. It can be seen, that the average error is high. However, it is mainly due to several outlier frames, where initialization is initially computed completely wrong, which results in high error values — therefore, we as well provide the median of the e^f to show that except these frames, the pose estimation quality is much better in most of the frames.

Note, that here we as well do not use any temporal consistency assumptions. In general, in both experiments we observed that the error jumps as soon as the initialization provided is completely wrong.

As can be seen from the Figure 3.13, the **FULL** method outperforms both other methods, namely **FULL-NSEG-NREF** and **FULL-NSEG** on the sequences of [116], or at least performs on par with **FULL-NSEG** baseline.

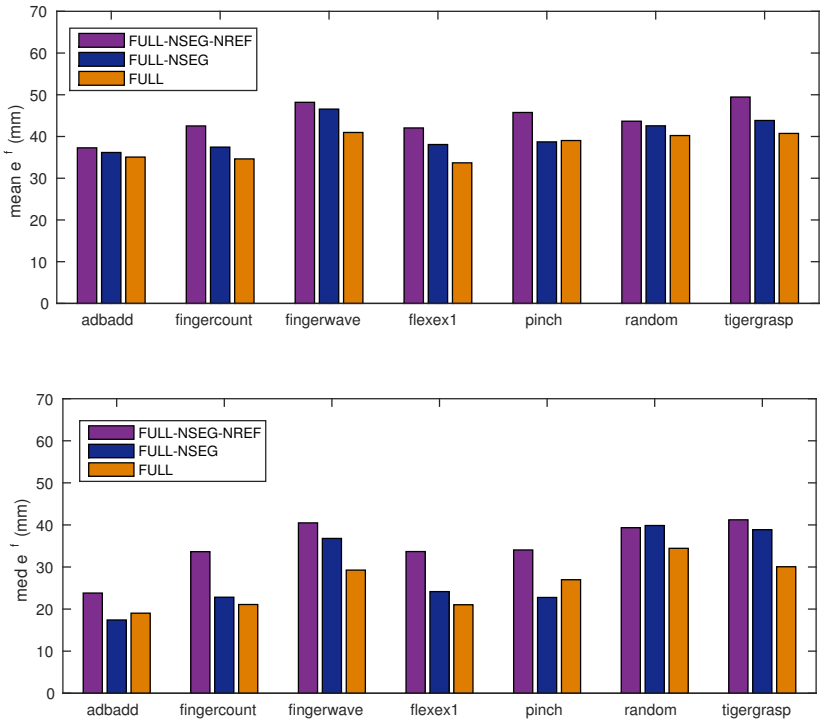


FIGURE 3.13: Comparison of the **FULL-NSEG-NREF**,**FULL-NSEG** and **FULL** baselines on Dexter dataset; top graph shows average of e^f across all frames, while bottom graph reports the median of the e^f across frames.

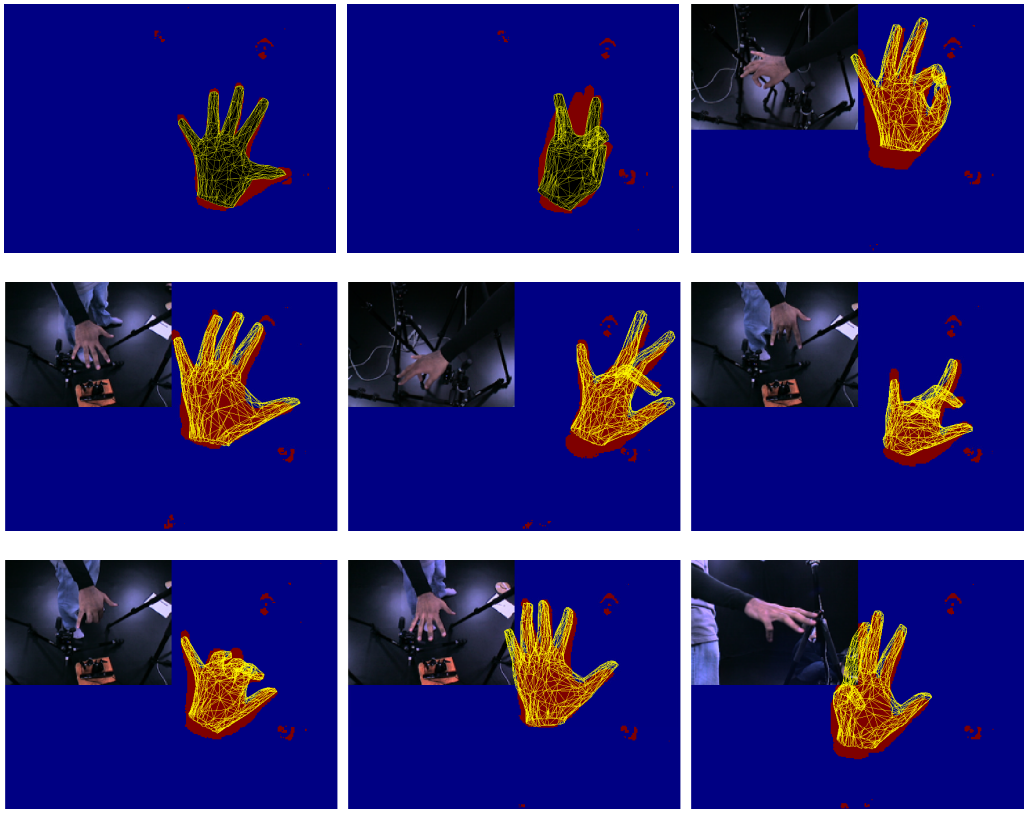
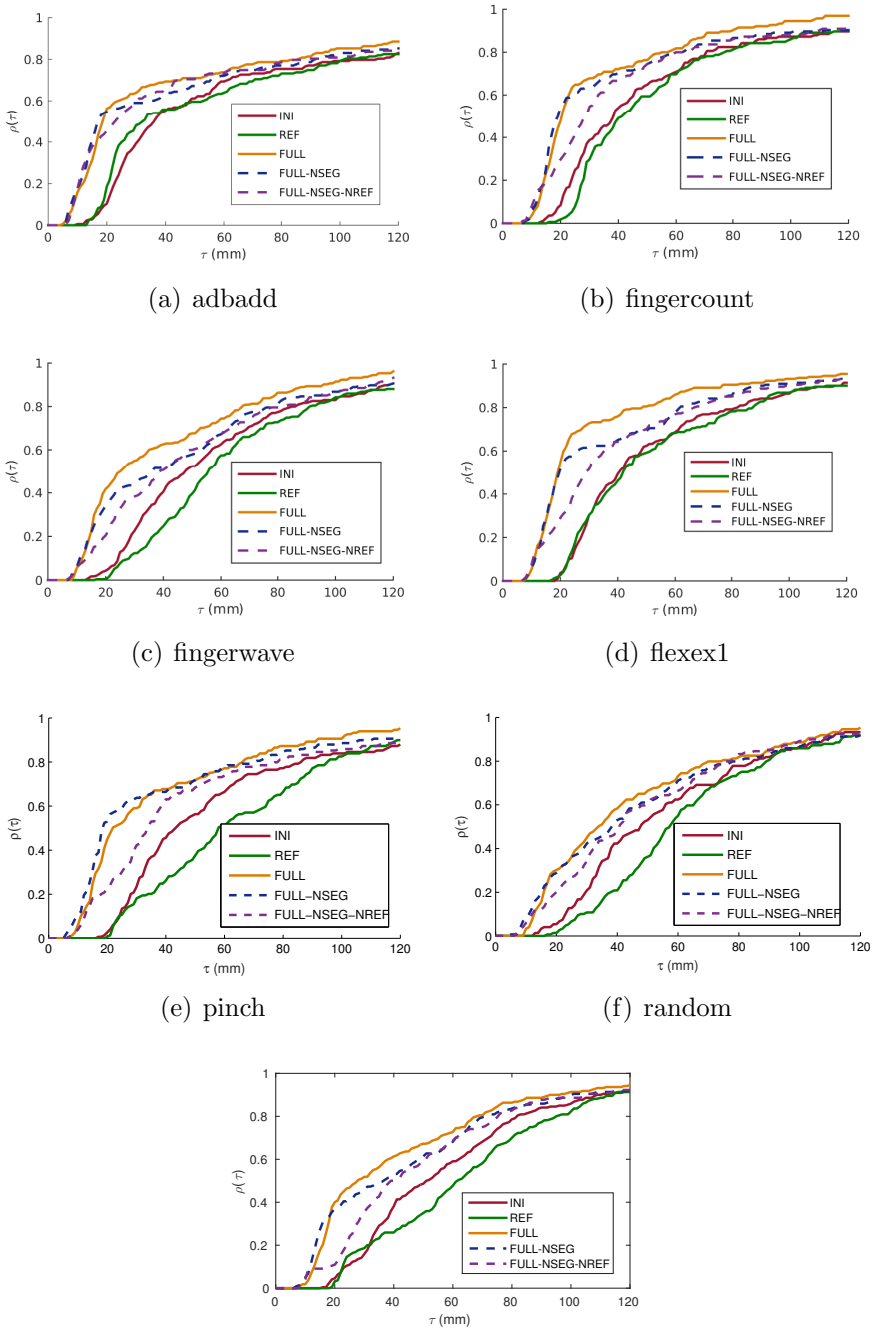


FIGURE 3.15: Examples of the fitted hand model for the frames from DEXTER dataset; RGB image is overlayed with depth image to show the actual hand pose.



(a) adbadd

(b) fingercount

(c) fingerwave

(d) flexex1

(e) pinch

(f) random

(g) tigergrasp

FIGURE 3.14: Comparison of all methods on the sequences of Dexter dataset in terms of $\rho(\tau)$

Finally, as in the previous section we compare all methods using $\rho(\tau)$ metric in Figure 3.14. As can be seen, **FULL** method outperforms all other methods for most of the sequences, except *pinch* sequence, for which **FULL-NSEG** performs slightly better for the low values of τ; it can be explained by the fact that for pinch sequence it is hard to obtain the reliable hand fingers segmentation, since the thumb and the index finger are connected together.

The examples of the estimated pose are presented in Figure 3.15.

Finally, we compare our results on the Dexter dataset with the more recent state-of-the-art hand pose tracking methods [109, 113, 115] as well as one method for a single frame hand pose estimation [119].

The approach from [115] is the modification of [116] using a single depth camera. It is as well model-based tracking approach, however, a user-specific hand model is used for tracking while we use a general hand model; [113] is an extension of [115], where a hand parts detector is used to robustify model fitting procedure. The approach from [109] is as well model-based, however, the hand model is represented in terms of subdivision surfaces and additional re-initialization step is used to account for track-lost situation. Finally, [119] is as well a single frame pose estimation approach; we use the evaluation results reported in [115].

Note, that most of the approaches mentioned make use of time consistency and therefore it is natural that the result obtained by our method from a single frame are worse then the results obtained employing time consistency hints. We therefore provide the results of our method in tracking setting (denoted by **FULL** + t) for comparison. As an error measurement, we compute the percentage of total frames in a sequence that have error e^f for fingertips locations less then a threshold, as well as average error e^f over the entire sequence.

The results for each sequence are reported in Table 3.2. Firstly, in case of single frame pose estimation our approach performs comparable to [119], although we do not require any training data; secondly, while **FULL** delivers overall much worse results than the tracking-based methods for the first five sequences, in the tracking scenario **FULL**+t we actually outperform the existing approaches on some sequences and performs on par on other sequences. However, on the last two sequences, *random* and *tigergrasp*, the tracking approach performes worse then the single frame pose estimation approach. These two sequences contain fast movements and the tracker breaks in the beginning of the sequence, which leads to wrong pose estimates on all subsequent frames.

The other effect to observe is that employing time consistency allows to increase the performance by nearly twice, which means that there is a lot of room for improving the initialization.

3.6 Discussion: strengths and weaknesses of the proposed generative approach

We presented an approach for model-based hand tracking, based on the classical non-rigid ICP algorithm for combined RGB-D data. We evaluated our approach both on the synthetic and the real-world data and showed, that it is capable of producing plausible hand pose estimations. Our approach can be used both for one-shot hand pose estimation and extended for tracking and it does not require use of special hardware, such as GPUs. Furthermore, our unoptimized MATLAB implementation requires around two minute per frame in a single hand pose estimation scenario and around 30 seconds in the tracking scenario; therefore, there is a potential to make the approach real-time. Additionally, our approach allows to partially overcome the problem of missing depth data by additionally relying on the silhouettes.

In general, we observed in the experiments, that when the initialization is completely wrong, the algorithm is not able to recover from the local minimum, while as soon as the initialization is sufficiently close to the correct pose, the optimization converges to the right solution. The most critical error that can happen is the wrong global pose of the hand. Furthermore, in case the finger poses are too far from the initial pose, the algorithm does not recovered as well.

Therefore, in the subsequent chapters, we focus on providing a more stable initialization for the optimization algorithm, using a machine learning method.

TABLE 3.2: Comparison of the proposed approach to the current state-of-the-art approaches; **FULL** denotes a single frame hand pose estimation, while **FULL** + t denotes the hand pose tracking scenario.

seq.	15mm	20mm	25mm	30mm	av.(mm)
adbadd					
[115]	79.4	97.7	98.6	99.1	14.0
[113]	56.6	70.6	76.2	84.9	19.0
[109]	–	–	–	–	15.0
FULL [71]	31.7	53.0	60.1	65.6	37.5
FULL + t	64.4	73.4	89.8	95.6	14.5
fingercount					
[115]	66.0	92.5	95.3	95.7	16.0
[113]	50.0	66.5	77.7	85.8	19.5
[109]	–	–	–	–	12.3
FULL [71]	15.7	38.8	55.1	60.6	37.2
FULL + t	41.5	55.1	73.9	79.0	21.6
fingerwave					
[115]	56.6	92.0	99.0	100.0	15.0
[113]	56.2	71.2	78.3	85.0	21.0
[109]	–	–	–	–	18.2
[119]	–	–	–	–	49.1
FULL [71]	19.0	34.4	40.0	48.5	44.2
FULL + t	29.4	39.8	44.3	49.2	36.2
flexex1					
[115]	88.5	100.0	100.0	100.0	13.0
[113]	53.7	68.1	76.7	85.5	21.0
[109]	–	–	–	–	10.7
FULL [71]	17.6	38.9	55.6	62.9	36.2
FULL + t	55.5	61.9	67.5	71.2	26.3
pinch					
[115]	49, 5	89.0	99.5	100.0	16.0
[113]	56.7	83.9	93.1	97.4	21.0
[109]	–	–	–	–	12.9
FULL [71]	16.1	33.6	46.3	50.3	42.4
FULL + t [71]	38.5	49.7	61.5	72.1	24.9
random					
[115]	29.1	48.8	58.9	70.8	27.1
[113]	19.1	40.7	59.0	70.6	28.5
[109]	–	–	–	–	23.3
[119]	–	–	–	–	46.1
FULL [71]	15.4	28.9	32.9	38.2	43.6
FULL + t	4.4	6.8	13.1	16.5	57.7
tigergrasp					
[115]	36.7	87.2	95.0	98.1	17.0
[113]	62.9	80.6	87.3	91.8	16.5
[109]	–	–	–	–	12.9
[119]	–	–	–	–	33.1
FULL [71]	19.1	40.4	44.5	47.7	44.1
FULL + t	12.9	17.9	19.5	24.0	53.7

Chapter 4

Sign language letters recognition

As mentioned in the previous chapter, model-based generative approach allows to achieve good results in the task of full DoF pose estimation, provided, however, sufficiently good initialization. The initialization can potentially be obtained using a temporal consistency assumption by transferring the pose estimation result from the previous frame. However, this approach has several disadvantages: firstly, since temporal consistency is assumed, the method is non-robust towards frame loss and fast motions; secondly, in case of track-lost situation, there is no way to re-initialize the tracker. Additionally, a discriminative approach-based initialization can be easily combined with temporal priors for the final pose prediction.

We pose the problem of finding a good initialization for the local optimization as a classification problem, i.e. instead of obtaining the initialization in continuous joint angles space, we propose to map points in joint angles space to a discrete number of class labels. The reasoning behind that is the following: from one side, knowing a discrete class label(and the corresponding initial point) is in many cases sufficient to converge to the correct solution in the vicinity of this point; from another side, the classification problem, provided a reasonable number of visually distinct classes is used (i.e., such that intra-class variance is much less then inter-class variance), is a much easier problem then the full DoF hand pose estimation.

Additionally, the general gesture recognition problem, which includes classification of static hand shapes, for example, American Sign Language signs recognition, is a separate research problem by itself. Moreover, classifying between a finit set of gestures seems to be an easier problem then full DoF hand pose estimation

FIGURE 4.1: The signs of the American Sign Language [1]. Notice the similarity between groups of signs, for example, between letters 'm', 'n' and 'e', or by 'p' and 'q'.

and therefore it seems unreasonable to require full DoF hand pose estimation be performed prior to recognition.

Therefore, in this chapter we concentrate on the problem of static sign letters recognition using a depth camera. To this end, we propose to firstly represent point clouds, corresponding to hand poses, using a global point cloud descriptors, invariant to the scale, translation and rotation, and then distinguish between them using a well-known classification method — random forests.

We evaluate the proposed approach for static poses classification on the public dataset, containing signs of American Sign Language Alphabet [100]. Note, that the substantial difference between the problem of static signs recognition and the problem of finding good initializations for the model based full DoF hand pose estimation is that we do not have control of the signs selected in the first case, while we can select the sufficiently different initializations in the second case, which makes the latter task easier.

In Section 4.1 we give an overview of the existing approaches for hand gesture recognition, in Section 4.2 we give a general introduction to the ensamble methods and in Sections 4.3-4.5 we describe the proposed approach. Finally, in Section 4.6 we provide an extensive evaluation.

4.1 Related work

The problem of sign language recognition was addressed in literature many times.

Early works used different kind of image convolutions to form feature vectors based on a single RGB image of a hand. In [54], the authors use wavelet families, computed on edge images, as features to train a neural network for 24 sign classification. Haarlet-like features, computed on a gray-scale images and on silhouettes were used in [27] for classification of 10 hand shapes. Principal Component Analysis (PCA) was applied directly on images to derive a subspace of hand poses, which is then used to classify the hand poses (see [23]). In [84], a modification of HOG descriptors are employed to recognize static signs of the British Sign Language. A SIFT-feature based description was used to recognize signs of ASL in [61]. All these methods depend heavily on the lighting conditions, subject's appearance and background. Additionally, pose normalization is required before feature computation, since the features mentioned above are not rotation-, position- and scale-invariant.

Due to appearance of range sensors there have been several breakthroughs in the area of recognition using a single camera. The most well-known advances are in the area of human pose recognition using the Microsoft Kinect sensor [111]. The authors used a special kind of features, computed directly on depth images, to classify human body parts. The same features were lately applied for the problem of hand pose estimation [60] and shape classification [59].

Hand detection and segmentation is a much easier task in case of availability of the depth data, therefore a number of methods appeared relying solely on segmentation derived from depth data, such as contour-based method, described in [101], and rule-based method, presented in [15].

Additionally, since depth data is robust to illumination and subject appearance changes, a number of methods appeared on calculating image features directly on depth images [100, 125]. Obviously, these methods still do not account for different viewpoints.

In our work, we address the problems mentioned above by using a rotation-, position- and scaling- invariant global descriptor, Ensemble of shape functions (ESF) [132]. Additionally, in the fashion of [59], we apply a multi-layered random

forest (MLRF) for classification. This allows us to significantly reduce the training time for MLRF and the memory consumption to store the trained forest in memory. Additionally, it allows to use different features in different layers.

In the consecutive sections, we firstly give an introduction to the theory behind random forests, as well as provide the description of the ESF descriptor. We then discuss the algorithm for building the multi-layered random forest, and finally evaluate the proposed approach on the publicly available dataset [100], as well as our own dataset.

4.2 Introduction to ensemble methods

The first ensemble-like methods were firstly introduced in 70s [24, 123], and became popular as the foundation of AdaBoost was laid [106]. The existing methods for building ensembles can be roughly divided into two groups [102]: in the first group of methods new classifiers are built based on the output of the previously created classifiers (dependent framework), while in the second group of methods all the classifiers are constructed independently (independent framework).

4.2.1 Dependent framework properties

In [106] two concepts are used: *weak learnability* requires the existence of a polynomial - time classification algorithm (weak classifier) that performs slightly better than random with high confidence (i.e., independent of the distribution of the training set); *strong learnability* requires the algorithm (strong classifier) to perform arbitrarily good with high confidence. It is then shown, that weak learnability and strong learnability are equivalent. Furthermore, a recursive algorithm for building a strong classifier from a set of weak classifiers is explicitly described. This approach defines dependent framework for building ensembles.

4.2.2 Independent framework properties

In the same time, an independent framework suggest training a set of classifiers independently from one another. If the classifiers are statistically uncorrelated (or the correlation is sufficiently low), the performance of the ensemble is much higher then the performance of each separate classifier and increases with the number of

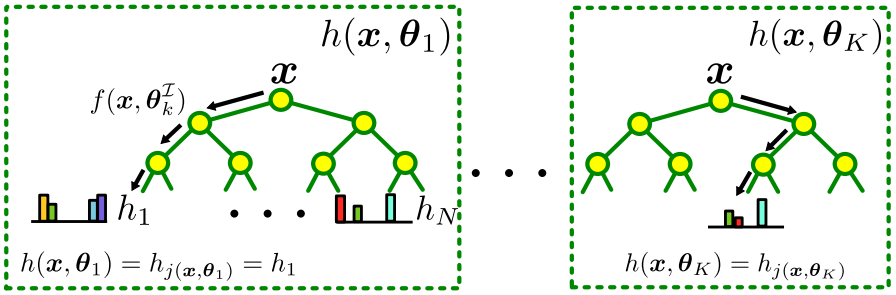


FIGURE 4.2: Random forest framework: sample $\boldsymbol{x}$ is propagated through multiple trees $h(\boldsymbol{x}, \boldsymbol{\theta}_k)$; at each node, a weak classifier $f(\boldsymbol{x}, \boldsymbol{\theta}_k^{\mathcal{I}})$ is applied and depending on its outcome, the sample is passed either to the left or the right subtree; when the sample reaches the root node, the corresponding predictor $h_j(\boldsymbol{x}, \boldsymbol{\theta}_k)$, stored in this node, forms the resulting prediction of the tree k for the sample $\boldsymbol{x}$

classifiers. The intuition behind that assumption is that uncorrelated classifiers make errors in different, uncorrelated, cases, and therefore the errors are mutually compensated in case the output of the classifiers is combined.

A well-known example of the independent ensemble is decision forests. For a subclass of decision forests, the above assumption was proven in [17]. We review the proof in the following Section.

4.2.2.1 Decision Forests and Random Forests

Firstly, a decision forest classifier consists of a collection of tree-structured classifiers $\{h(\boldsymbol{x}, \boldsymbol{\theta}_k), k = 1, \dots K\}$, where $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^D$ is the input vector, $y \in \mathcal{Y}$ is the class label and $\boldsymbol{\theta}_k$ are the parameters of the classifier, and $h(\boldsymbol{x}, \boldsymbol{\theta}_k) \in \mathcal{Y}$, where $\mathcal{Y}$ is a discrete set of class labels. We denote the training data $(\boldsymbol{x}, y) \in T \subset \mathcal{X} \times \mathcal{Y}$ Here we consider the binary tree structure of the classifiers. Each internal node $\mathcal{R}$ of a tree contains a (very) weak binary classifier $f(\boldsymbol{x}, \boldsymbol{\theta}_k^{\mathcal{I}}) \in \{-1, 1\}$ (for example, a simple threshold rule), parametrized by subset of values $\boldsymbol{\theta}_k^{\mathcal{I}}$ from $\boldsymbol{\theta}_k$. A sample $\boldsymbol{x}$ is propagated through the tree from the root to one of the leafs, where at each level the corresponding $f(\boldsymbol{x}, \boldsymbol{\theta}_k^{\mathcal{I}})$ classifier is applied and the sample is propagated down to either right or left sub-tree, depending on the classifier's output (see Figure 4.2). In the leaf nodes, the predictors are stored.

Next, we will review the theory behind decision forest and motivation for the randomization, as well as some properties of the decision forests and their subclass, random forests.

Decision forests performance Following [17], we firstly define a classification margin of a set of classifiers with fixed $\Theta = \{\boldsymbol{\theta}_k\}_{k=1}^K$ as:

$$mg(\boldsymbol{x}, y, \Theta) = \frac{1}{K}\sum_k \mathbb{I}(h(\boldsymbol{x}, \boldsymbol{\theta}_k) = y) - \max_{j \neq y} \frac{1}{K}\sum_k \mathbb{I}(h(\boldsymbol{x}, \boldsymbol{\theta}_k) = j) \tag{4.1}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function (note, that in [79] classification margin is defined in a slightly different way, but in fact has the same meaning). For a sample to be correctly classified, $mg(\boldsymbol{x}, y, \Theta) \geq 0$.

The generalization error for a fixed set of parameters Θ is then defined as:

$$e^* = P_T(mg(\boldsymbol{x}, y, \Theta) < 0) \tag{4.2}$$

where P_T denotes the probability over a distribution in T space of the incorrect classification result ($mg(\boldsymbol{x}, y) < 0$). The following theorem can be proven for the case when $K \to \infty$:

Theorem 4.1. *For almost surely all sequences of $\{\boldsymbol{\theta}_k\}_{k=1}^K$, generalization error converges to the following expression:*

$$\lim_{K \to +\infty} e^* = P_T\left(\left\{P_\Theta(h(\boldsymbol{x}, \boldsymbol{\theta}) = y) - \max_{j \neq y} P_\Theta(h(\boldsymbol{x}, \boldsymbol{\theta}) = j)\right\} < 0\right) \tag{4.3}$$

We denote the limit of the classification margin for a random forest with infinite number of trees as margin function:

$$mr(\boldsymbol{x}, y) = \lim_{K \to +\infty} mg(\boldsymbol{x}, y, \Theta) = P_\Theta(h(\boldsymbol{x}, \boldsymbol{\theta}) = y) - \max_{j \neq y} P_\Theta(h(\boldsymbol{x}, \boldsymbol{\theta}) = j) \tag{4.4}$$

The strength of a classifier $\{h(\boldsymbol{x}, \Theta)\}_\Theta$ is then defined as:

$$s = E_T mr(\boldsymbol{x}, y) \tag{4.5}$$

Then, taking into account Theorem 4.1 and Chebychev inequality, and denoting variance of a random variable by $\sigma^2(\cdot)$:

$$e^* \leq \frac{\sigma^2(mr(\boldsymbol{x}, y))}{s^2} \tag{4.6}$$

We further introduce the notation of raw margin function as:

$$rmg(\Theta, \boldsymbol{x}, y) = \mathbb{I}(h(\boldsymbol{x}, \Theta) = y) - \mathbb{I}(h(\boldsymbol{x}, \Theta) = \hat{j}(\boldsymbol{x}, y)), \tag{4.7}$$

where $\hat{j}(\boldsymbol{x}, y) = \text{argmax}_{j \neq y} P_\Theta(h(\boldsymbol{x}, \Theta) = j)$. It is easy to see that $mr(\boldsymbol{x}, y) = E_\Theta rmg(\Theta, \boldsymbol{x}, y)$.

Theorem 4.2. *The upper bound on the generalization error e^* is given by:*

$$e^* \leq \bar{\rho}\frac{(1 - s^2)}{s^2}, \tag{4.8}$$

where

$$\bar{\rho} = \frac{E_{\Theta,\Theta'}\left\{\rho(rmg(\Theta, \boldsymbol{x}, y), rmg(\Theta', \boldsymbol{x}, y))\sigma(rmg(\Theta, \boldsymbol{x}, y))\sigma(rmg(\Theta', \boldsymbol{x}, y))\right\}}{E_{\Theta,\Theta'}\left\{\sigma(rmg(\Theta, \boldsymbol{x}, y))\sigma(rmg(\Theta', \boldsymbol{x}, y)\right\}} \tag{4.9}$$

$\rho(\cdot, \cdot)$ *and* $\sigma(\cdot)$ *denotes the correlation and the standard deviation on* T *accordingly.*

To get an intuition about the result above, let us consider a binary classification problem, i.e., $\mathcal{Y} = \{-1, +1\}$. In this case, raw margin is defined as $rmg(\Theta, \boldsymbol{x}, y) = 2I(h(\boldsymbol{x}, \Theta) = y) - 1$.

Consequently, the expression in the Equation (4.9) can be simplified to:

$$\bar{\rho} = E_{\Theta,\Theta'}\rho(h(\cdot, \Theta), h(\cdot, \Theta')) \tag{4.10}$$

It is easy to see, that $\bar{\rho}$ defines here the correlation between two different members of the forest.

From the Theorem 4.2 and from the Equation (4.10) it follows, that the generalization error decreases with the correlation between different trees.

There are several ways introduced to decrease the correlation of the separate trees:

- In [7], random parameter sampling during training was proposed.
- In [17], bagging was proposed: each tree is trained on a new training set, formed from the initial training set by drawing random instances from it with replacement. Thus, each tree is trained on a separate training set.

Random forests are in fact decision forests, that make use of the both proposed methods [7, 17] to decrease the correlation between individual trees.

There are many adaptations of random forests exist for different tasks [21], such as classification, regression, semi-supervised learning and clustering, as well as for object detection and localization [36].

Maximum margin properties of random forests The important result concerning random forests is its maximum margin properties [79] in classification task.

Firstly, let us consider training procedure for the random forest. During training, the parameters of the nodes of each tree, as well as the predictors, are trained. To

do so, the training set (or its part, in case bagging is used) is propagated down the tree and recursively split at each internal node, i.e. let the set $\mathcal{S}_i$ reach the node $\mathcal{R}_i$ — then it gets split into two parts: $\mathcal{S}_i^L$ and $\mathcal{S}_i^R$, each propagated down to the corresponding sub-tree. At each internal node, the parameters of a binary classifier are determined to optimize some goodness of split criterion. As proposed in [7], parameter vectors are randomly sampled and evaluated against the selected criterion, which allows to generate uncorrelated trees, provided that the parameter space is sufficiently large.

Goodness of split criterion is usually defined as a difference between a local score $\mathcal{L}(\cdot)$ computed on the training set in the node $\mathcal{R}$ versus the same score computed on the split set. The score typically measure the purity of a node with respect to class labels. For the task of classifying the samples with a discrete finite set of labels, i.e random variable $Y \in \mathcal{Y} = 1 \dots C$, entropy is often used:

$$H(Y) = -\sum_{i=1}^{C} P(Y = i) \log P(Y = i) \tag{4.11}$$

Another common purity index is Gini index:

$$G(Y) = \sum_{i=1}^{C} P(Y = i)(1 - P(Y = i)) \tag{4.12}$$

Consequently, the goodness of split is defined as

$$I(\mathcal{S}, \mathcal{S}^L, \mathcal{S}^R) = \mathcal{L}(\mathcal{S}) - \sum_{i=\{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} \mathcal{L}(\mathcal{S}^i) \tag{4.13}$$

where $\mathcal{L}(\cdot)$ is either $H(\cdot)$ or $G(\cdot)$.

It is easy to see, that recursively splitting the training set is in fact a greedy optimization of some (unknown) loss $\ell(\cdot)$ on the whole training set. The following theorem provides the connection between the recursively optimized cost and the general loss being optimized [79]:

Theorem 4.3. *Let us define a multi-valued function* $\boldsymbol{g}(\boldsymbol{x}) = \Big(g_1(\boldsymbol{x}), \quad \dots, \quad g_C(\boldsymbol{x})\Big)^T$; *the function is called a margin vector if:*

$$\forall \boldsymbol{x} : \sum_{i=1}^{C} g_i(\boldsymbol{x}) = 0$$

Then, given a margin maximizing loss function $\ell(g_y(\boldsymbol{x}))$, *the local score for a decision node* $\mathcal{R}$ *is defined as* $\mathcal{L}(\mathcal{R}) = \sum_{i=1}^{C} p_i \ell(p_i - \frac{1}{C})$

The direct consequence of this theorem is that entropy score minimizes negative log-likelihood of the data, while Gini index is related to the hinge loss function.

Prediction of a random forest and different split functions The prediction of random forest is formed by the leafs of individual trees. Whenever a sample reaches a leaf j, the corresponding predictor $h_{j(\boldsymbol{x},\boldsymbol{\theta}_k)}$, stored at this leaf (note, that notation $h_{j(\boldsymbol{x},\boldsymbol{\theta}_k)}$ is used because in practice the predictor depends solely on the training set, while the leaf index does depend on $\boldsymbol{x}$). The prediction of a tree itself then coincides with the leaf prediction: $h(\boldsymbol{x},\boldsymbol{\theta}_k) = h_{j(\boldsymbol{x},\boldsymbol{\theta}_k)}$.

For the classification problem, where $\mathcal{Y} = \{1,\dots,C\}$, the prediction is usually constructed as a histogram of the training samples' classes, that are gathered in each leaf after propagating the training samples through the tree, i.e.

$$h_{j(\boldsymbol{x},\boldsymbol{\theta}_k)} = \begin{pmatrix} p_{j(\boldsymbol{x},\boldsymbol{\theta}_k)}(1) & \dots & p_{j(\boldsymbol{x},\boldsymbol{\theta}_k)}(C) \end{pmatrix}^T \tag{4.14}$$

$$h(\boldsymbol{x},\boldsymbol{\theta}_k) = h_{j(\boldsymbol{x},\boldsymbol{\theta}_k)} = \begin{pmatrix} p_k(1|\boldsymbol{x}) & \dots & p_k(C|\boldsymbol{x}) \end{pmatrix}^T \tag{4.15}$$

The prediction of the forest is formed from the predictions of each tree as:

$$h(\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^{K} h(\boldsymbol{x},\boldsymbol{\theta}_k) = \begin{pmatrix} p(1|\boldsymbol{x}) & \dots & p(C|\boldsymbol{x}) \end{pmatrix}^T \tag{4.16}$$

Finally, the predicted class is selected as $y^* = \operatorname{argmax}_c p(c|\boldsymbol{x})$.

The typical node-level binary classifier is a threshold function, for example, $f(\boldsymbol{x}, i, \tau) = x^i - \tau$ (therefore, $\boldsymbol{\theta}_k^{\mathcal{I}} = (i, \tau)$, and the binary decision is defined by the predicate $f(\boldsymbol{x}, i, \tau) > 0$).

In our work, we use a slightly more advanced function $f(\boldsymbol{x}, i_1, i_2, w_1, w_2) = x^{i_1}w_1 + x^{i_2}w_2 - 1$ ($\boldsymbol{\theta}_k^{\mathcal{I}} = (i_1, i_2, w_1, w_2)$).

4.2.2.2 Clustering with random forest

Clustering is directed towards finding structures in data, which is equivalent to distinguishing between the given data and the data having same marginal distributions but no correlations between different dimensions [110]. We therefore create an artificial training dataset, containing the training data, which is ment for clustering, in one class, and the generated data in another class. The latter is created from the training data as following: given N samples in the training dataset T, for each of D dimension of the feature space we randomly sample with

replacement N samples $j_d \in 1 \dots N, \quad d \in 1 \dots D$; consequently, the vectors of the second class are then formed as $\boldsymbol{x} = \begin{pmatrix} x_{j_1}^1 & x_{j_2}^2 & \dots & x_{j_D}^D \end{pmatrix}^T$.

Afterwards, a random forest with many decision trees (we used 1000 trees in our experiments) is trained for two-class classification problem.

One of the random forest features is that in some sense similar samples tend to fall within the same leafs during training. This can be characterized by so-called proximity matrix P of size $N \times N$:

$$P_{ij} = P_{ji} = \frac{|\{m|\boldsymbol{x}_i, \boldsymbol{x}_j \in L_k^m\}|}{K} \tag{4.17}$$

Here L_k^m is the set of all samples $\boldsymbol{x}$, that end up in the leaf k for the decision tree m.

The proximity matrix P can be regarded as a similarity measurement and therefore can be used as an input for a clustering algorithm. We transform the proximity matrix to a distance measure $m_P(i,j) = \sqrt{1 - P_{ij}}$.

The distance measure proposed above is in fact not a metric, and therefore a suitable clustering algorithm is required.

We chose to use hierarchical clustering [48] to cluster the data based on the produced similarity measure. For hierarchical clustering, we use complete linkage to aggregate data into a cluster tree. We find the level at which the cluster tree should be cut, by setting the required number of clusters K.

4.3 Proposed approach

We propose to improve sign language classification using two main ideas.

Firstly, since a hand sign can have very different appearance when seen from a different view (see Fig. 4.3), we propose to use rotation-, translation- and scaling-invariant image features to reduce intra-class variation. Such features exist for 3D point clouds, which can be derived from the depth data, delivered by a depth sensor. The overview of the state-of-the-art features for object recognition is presented, for example, in [6]. We chose to use the ensemble of shape function (ESF) descriptor [132], since it is translation-, rotation- and scale- invariant, and can be computed in real time.

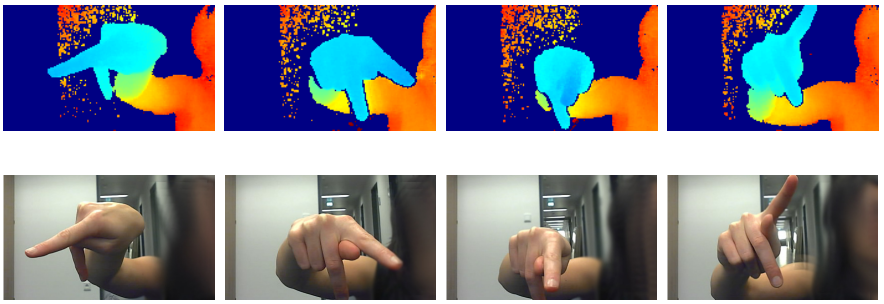


FIGURE 4.3: Difference in appearance of the sign 'P' depending on the view (the new dataset).

Secondly, we propose to use a multi-layered random forest (MLRF) for classification (Fig. 4.5). The intuition behind this approach is to firstly discover the groups of similar feature descriptors using clustering and then for each group train a separate classifier that can better distinguish the classes withing this group. Ideally, each group contains less classes then the original set.

There are several advantages of using MLRF:

- **Potentially higher accuracy:** Due to the usage of two levels of forest, averaging happens between the trees on the first level, and therefore the error already made in some trees is not propagated to the second level. The second-level random forests are trained on clusters containing much less variation then the initial training set, and therefore a smaller forest is required to make classification robust.

- **Smaller training time and memory consumption:** Since random forest consists of binary trees, for each node the training parameters should be stored, the memory size to store the forest increases exponentially with the depth of the forest. Since MLRF contains much less nodes, the memory consumption is reduced drastically. For example, to store a forest of depth $D = 20$ with $T = 10$ full binary trees, where each node has 5 parameters, the minimum amount of memory required is 560 Mb, and a forest of depth $D = 25$ requires over 18 Gb memory. A MLRF having 10 forests on the second level, and the depth of the forests on the both level equal to 10, requires around 6 Mb of memory, and to store a MLRF with depths $D_1 = 13$ and $D_2 = 12$ only 26.5 Mb is needed.

4.4 Global point cloud descriptor

ESF descriptor consists of ten histograms, concatenated together. To compute the histograms, three functions are used. The first function D2 is the distances between two random points in the point cloud. The second function A3 describes the angles enclosed by two lines created by randomly sampling three points from a point cloud. The third function D3 is defined as a triangle's area, where each triangle is defined by three randomly sampled points.

Additionally, each function's result is classified into three classes, depending on the selected random points: if a line (angle, triangle), formed by two (three) points, is inside the point cloud (a line is consider to be inside the point cloud for 2.5D data if it is not visible from the camera), the result is classified as the ON class, if outside — as OFF, and finally if the line (angle, triangle) is both inside and outside of the model, the result is classified as MIXED. Consequently, depending on the function type and the class of the function (ON, OFF, MIXED), the results are aggregated into 9 histograms.

Additionally, for the MIXED lines the ratios between parts of the lines lying inside the point cloud are computed and used to form the 10 histogram.

To determine if a line lies inside or outside a point cloud, the space is voxelized into 64×64 voxel grid and Bresenham algorithm is applied to trace the line and determine, if it intersects the point cloud (or lies inside or outside of it).

Each histogram has 64 bins, and therefore the whole descriptor, consisting of the 10 histograms, has the length of 640 bins.

This descriptor has the following features:

- given that the whole object is visible, it is rotation- and translation- invariant; additionally, the descriptor is also scale invariant, since the range of possible values of D2, D3 and A3 is normalized before computing the histograms;
- invariant under mirror transformation;
- it does not require unstable and time consuming normal computation;
- it is reported to be fairly discriminative [6, 132].

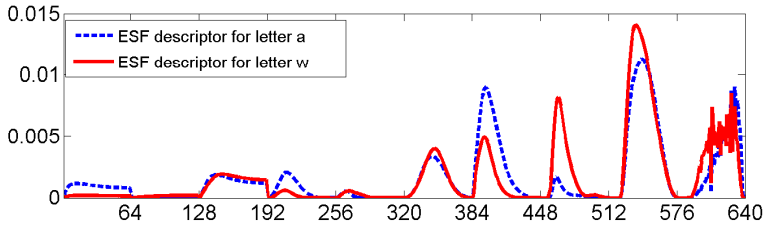


FIGURE 4.4: ESF descriptors for the letters 'a' and 'w'.

The hand has fairly different appearances depending on the gesture, so we treat the sign classification problem as object classification problem. We compute the ESF descriptor for the segmented hand point cloud. Examples of ESF descriptors for different classes are given in Fig. 4.4.

4.5 Multi-layered random forest

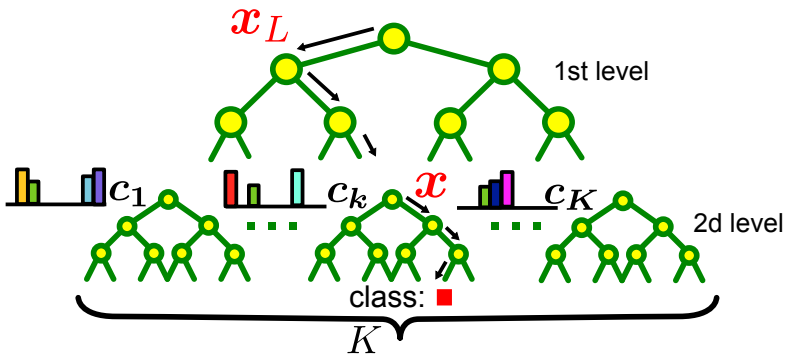
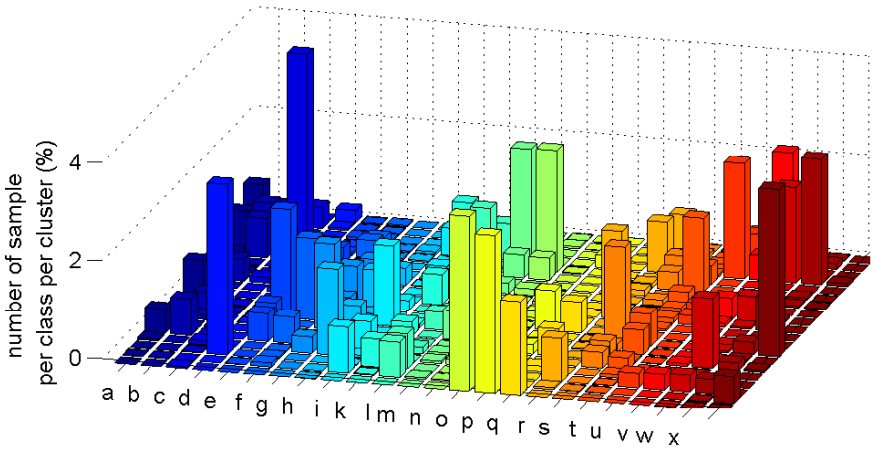
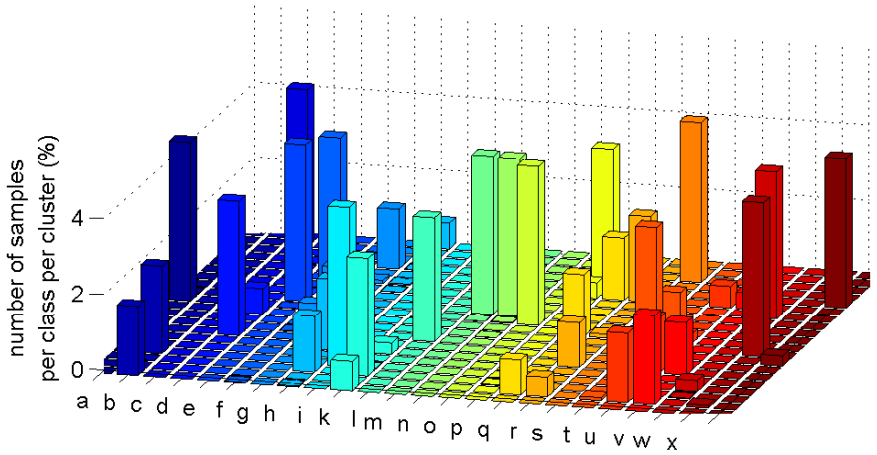


FIGURE 4.5: Multi-layered forest structure; on the level one, the forest determines the cluster of the feature vector $\boldsymbol{x}$; on the level two, the final class label is assigned by the forest, corresponding to the cluster label assigned on the first level.

Clustering the data Recently several works appeared on creating multi-layered random forest. For example, in [59] the authors use spectral clustering to pre-cluster data and then use multi-layered random forest to perform hand parts regression. Intuitively it is easier to distinguish between smaller number of classes, so we pre-clustered the data to improve classification accuracy. Our goal in this case is to produce clusters that contain much less classes then the whole training set. We opted to use RF-based clustering technique, described in Section 4.2.2.2. In general, we experimented with other types of clustering algorithms, such as



(a)



(b)

FIGURE 4.6: (a) Data clustering results for the [100] dataset using hierarchical clustering with $K = 20$ clusters. (b) Data clustering results for the synthetic data using hierarchical clustering with $K = 20$ clusters. The horizontal axis show the classes (letters) and the clusters, the vertical axes shows the number of samples in each cluster, corresponding to the current class.

spectral clustering, applied on the ESF descriptors directly, as well as on the RF-based proximity matrix P (see Equation (4.17)), and found that the produced clusters to much less extent fulfil the goal of separating groups of classes.

We use less bins in the ESF descriptor for clustering, because it reduces the noise in the computed descriptors. According to our experiments, more coarse histograms already contain enough information to find similar signs, in the same time containing much less noise. We define L as the histogram aggregation level parameter, where $L = 0$ defines the full 64-bins histograms, $L = 1$ reduces the size of histogram to 32 bins, level $L = 2$ produces 16 bins, etc.

(a) (b) (c) (d) (e)

FIGURE 4.7: The depth images of the signs, forming two different clusters in the clustering, depicted in Fig. 4.6; images 4.7(a),4.7(b),4.7(c) come from one cluster, 4.7(d),4.7(e) — from another cluster.

Examples of clustering results are shown in Fig. 4.6(a) and Fig. 4.6(b) for the real data from the [100] dataset and for synthetic data, containing a hand model rendered in the poses, corresponding to the ASL signs, respectively. It can be seen, that the clustering produces the desired result, i.e. similar signs are grouped together in clusters and only a subset of all 24 classes constitutes each cluster. For example, for real data the signs 'o','p','q' are grouped together in one cluster, and the signs 'm' and 'n' are grouped into another cluster; as shown in Fig. 4.7, the depth images of these signs are quite similar and so are the descriptors.

Finally, while synthetic data clustering separates the classes almost perfectly, producing clusters, containing only one class, real data contains much more noise and therefore the clusters contain small portions of different classes (Fig. 4.6).

Training The multi-layered random forest (MLRF) is a random forest, consisting of two layers.

To train the MLRF, we first perform clustering of the data. Initially we set the number of clusters to $K = 20$. As we showed earlier, the clusters build from the real data are noisy and contain small portions of data from many classes. Our goal is to create clusters, containing small number of classes, therefore we prune clusters by:

- merging small clusters to bigger clusters; we set the minimum cluster size to 50% of the size of a class in the training data; if a cluster is smaller than this size, for each element of this cluster we find another cluster, containing the most elements of the same class, and merge this element to the new cluster; due to this threshold the number of clusters is in fact determined

automatically for each dataset. We could observe, that in case of ASL signs classification problem, $12 - 15$ big clusters are formed.

- pruning by removing samples from a cluster, if the proportion of samples of this class in the cluster is less then 10% of the cluster size.

After the data is clustered, we train the first level random forest on the aggregated feature vectors. This forest assigns a cluster label to each incoming vector. Afterwards, for each of the derived clusters, we train a separate random forest on the full feature vectors (as oppose to aggregated feature vectors) to distinguish between similar signs.

We define the **cumulative depth** of the MLRF as the sum of depths of the RF on the first and the second layers. We also set the number of trees to be the same for each forest in the first and the second layers.

It is important to ensure that the feature vectors and the random forest parameters have the same scale. To do so, we re-normalize feature vectors in each cluster before training by substructing its mean and dividing by the standard deviation. Such re-normalization is useful, because similar vectors are clustered together, and therefore the variance withing each cluster is much lower, then the variance of the whole sample. The variance is derived by the classes that are contained in this cluster, and skipping re-normalization can decrease classification performance of the trained forest. We also exclude the dimensions, in which variance inside the cluster is insignificant.

Testing During the testing phase, each sample is passed through the first-level forest to determine, from which cluster the sample comes from. The first-level forest determines the cluster label of a sample. Afterward, the sample is passed to the corresponding forest on the second level to determine its class label.

4.6 Evaluation

We evaluate our algorithm in different settings on synthetic data, publicly available database, containing ASL signs [100] and the depth data collected from the Intel Creative Depth Camera. We compare the results of our method to the state-of-the-art methods [100] and [59] in terms of classification error (the number of

incorrectly classified samples divided by the size of the test set), training time and memory consumption.

We use an unparallelized MATLAB random forest implementation [57]. The computational times are provided for one core CPU employed in computations.

4.6.1 Synthetic data

In a natural environment, a subject's hand position performing different signs relative to a camera can change a lot. Therefore, using the features robust to such changes increases the performance.

To demonstrate robustness of the ESF features, we generated synthetic data using the hand model from [10] to create depth images of 24 static letters of ASL. For the training set, we generated 500 images per letter, in all cases the hand was parallel to the camera. For the test set, we applied rotations around three main axis to evaluate the classification error versus the angle of rotation of the hand model.

We separate rotation around the axis X and Y, forming the camera plain, and around the camera Z axis. As shown in Fig. 4.8, the latter has no influence on the entire performance, since the point cloud shape itself stays exactly the same. Stronger rotation around X and Y axis causes stronger performance decrease, since it changes point cloud appearance (see Fig. 4.3). This result shows, that to be able to robustly recognize gestures from a different viewpoint in general the corresponding training data is still needed. However, the rotation of a hand within the limits of 10 degrees does not degrade classification performance significantly.

To justify the usage of the multi-layered RF, we evaluated the training time of a conventional RF vs. the MLRF with the same cumulative depth (see Section 4.5). As can be seen, the multi-layered RF training is around 5 times faster then a simple RF (Figure 4.9) in case $K = 20$ clusters are used.

Note, that the classification error for both forests is the same (Fig. 4.10) starting from the depth $D = 20$. The classification error for the MLRF is a little higher for the smaller depth. Our experiments showed that having depth less then 10 makes the first-level forest too weak to classify the data into the correct cluster.

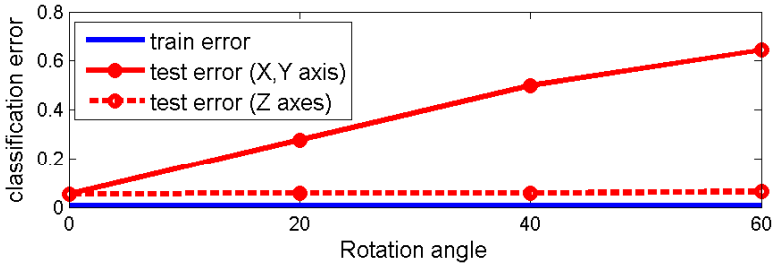


FIGURE 4.8: Train and test error depending on rotation angle limits of the test sample; the forest used has the number of trees $T = 10$ with depth $D = 20$.

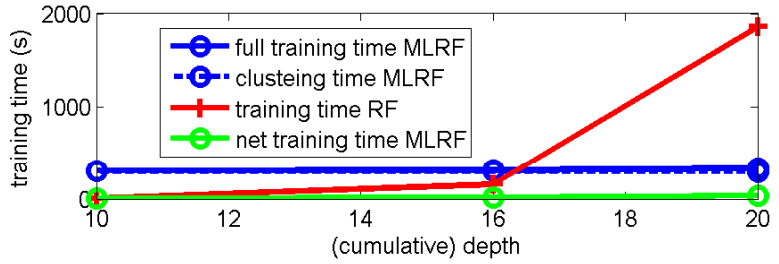


FIGURE 4.9: Training time (s) needed for a simple RF and for the multi-layered RF with $T = 10$ depending on the forest depth (for the multi-layered forest, depth of the first layer is $D/2$, as well as the depth of the second layer).

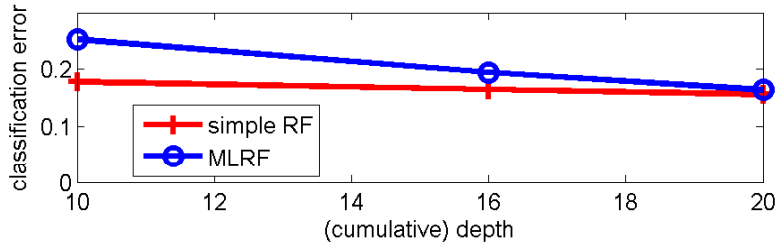


FIGURE 4.10: Classification error depending on the depth of the forests.

We also evaluated the MLRF performance depending on the separation between two layers. The optimal separation between two layers of forest in depth is approximately $D/2$ (Fig. 4.11).

4.6.2 Real data

The public dataset [100] The dataset contains 24 static signs from American sign language (ASL), performed by 5 subjects. There is a high variability in how the people perform signs and in the relative position to the camera, although the variability in pose is relatively low for one subject.

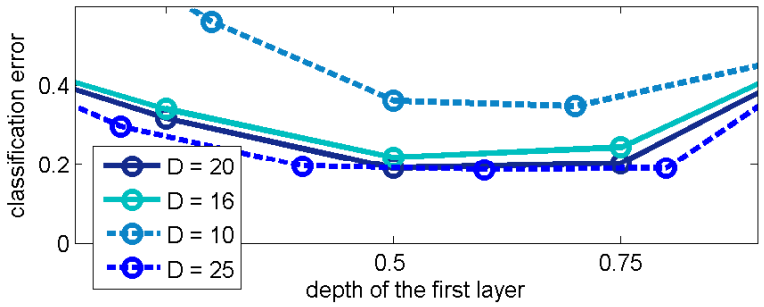


FIGURE 4.11: Classification error depending on the separation in depth between the first and the second layers of the forest ($T = 10$).

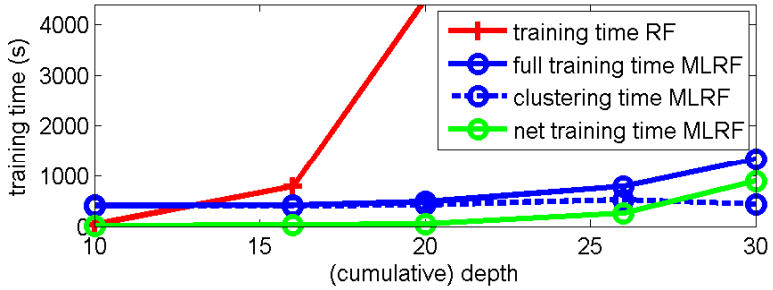


FIGURE 4.12: Training time depending on the depth of the forests.

To evaluate how good our method generalizes, we trained the forests on 4 subjects and then evaluated the performance on the subject, left out of the training. In Tab. 4.1 this is denoted as l-o-o experiment. The reported baseline error [100] for the database is 51% for the depth data. As can be seen, the training error in these experiments is decreased by almost 8%. Additionally, in case of using half of the data for testing and half for training (h-h experiment), we were able to decrease the error to 13%.

To justify the usage of the MLRF, we compare its performance to the performance of a standard random forest. We vary the depth of the random forest and compare the classification error of the simple RF with the MLRF and measure the training time (Fig. 4.12 and Fig. 4.13). Since the database overall size is around 65000 of images, the MLRF achieves even more win in training time — it is around 10 times faster then the simple RF for the depth $D = 20$.

In [59], the performance of the shape classification for the [100] dataset is presented. In the l-o-o experiment, the achieved accuracy is 85% vs. 57% of our method and in the h-h experiment, the reported accuracy is 97.8%. However, the reported training time is around 4000 s per tree on a quad core PC. For our

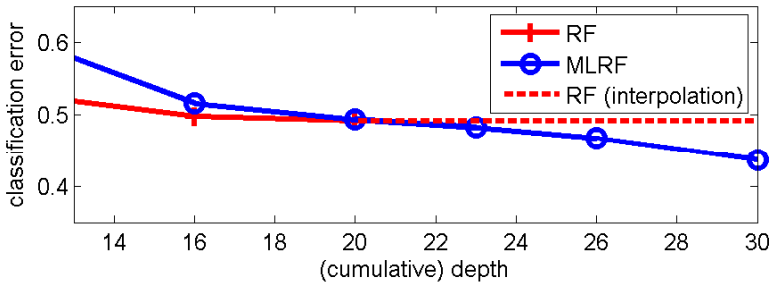


FIGURE 4.13: Classification error depending on the depth of the forests.

TABLE 4.1: Performance comparison on the [100] database (in all forests, $T = 10$) in terms of the error for h-h and l-o-o scenarios; memory consumption is given in Mb.

method	h-h	l-o-o	time(s)	mem
GF + RF [100]	31%	51%	—	—-
RF ($D = 20$)	15%	49.1%	4485	560
MLRF ($D = 20$)	23.4%	50.1%	522	6.05
MLRF ($D = 30$)	**13%**	**43%**	1132.3	192.5

TABLE 4.2: Performance evaluation (MLRF) for one subject on the data from [100] and on the data, collected with Intel Gesture Camera.

	dataset [100]	Intel Camera data
error $D = 20$	6%	22%
time (s)$D = 20$	212	68.77
error $D = 30$	2.45%	15.3%
time (s)$D = 30$	821	476

method, the training time for an unoptimized MATLAB version of the MLRF is less then one thousand of seconds on one core. In one-subject experiment (i.e., when the classifier is trained on a training set, containing a single subject performing gestures, and the tested in recognition of the gestures, performed by the same subject), our method shows the accuracy of 97.4% (see Tab. 4.2). Therefore, in one-subject experiment our method has the same performance as the method in [59], while the training times allows to re-calibrate the system for a new subject very fast.

Evaluation on the new dataset To illustrate applicability of the recognition method to other sensors, we collected a dataset using Intel Creative Gesture Camera [2]. The device represents a low-cost time-of-flight camera with the range

from 10cm to 1m, maximal frame rate of 50fps and the resolution of 240×320 pixels (for depth data). High speed and near range make the camera potentially useful in the field of gesture recognition.

The dataset contains 3 subjects performing 24 signs from the ASL sign language. For each letter, we collected around 250 data frames. The purpose of this dataset is to account for natural variation in a hand pose relative to the camera, and therefore the participants were asked to rotate the hand relative to the camera by 30 degrees in all directions to create more variability (see Fig 4.3 for an example).

Since the data has lower resolution (240×320 vs. 480×640 pixels for the Kinect), we expect some recognition performance decrease. Indeed, the results show (see Tab. 4.2) some performance decrease, which we explain by low camera resolution and high variance in hand pose.

4.7 Discussion: strengths and weaknesses of the discriminative approach

As evaluation shows, the proposed approach has fairly good discriminative properties. However, some signs, such as 'm' and 'n' are not easy to distinguish. On the other hand, these signs are fairly similar in joint angles space as well, and therefore inability to distinguish between them should not present a problem, if the proposed approach is used as an initialization for the subsequent pose refinement using a model-based approach. The main question to address here is how to build a representative set of the initial poses (classes).

Therefore, in the next chapter we provide the method and the experimental results on combining both the discriminative and generative approaches.

Chapter 5

Combining discriminative and generative approaches

5.1 Introduction

In the previous sections, we discussed two main approaches to hand pose estimation: generative and discriminative. While the discriminative approach performs well in a controlled setting with a small number of pose classes, its performance degrades significantly, as soon as the number of classes increases or as soon as the pose classes become more visually similar. Consequently, it cannot be used to discriminate between a significant number of poses with the goal of full DoF hand pose estimation, since there is always a tradeoff between the precision of the estimated pose and the performance of the classifier. On the other hand, generative approach performs for the task of continuous full DoF hand pose estimation, provided, however, a good initialization. We therefore propose to combine both approaches.

To do so, we cluster the available hand point clouds in appearance space, and then refine the clusters in joint angles space to obtain easily distinguishable clusters, each cluster containing hands in roughly the same poses. Each cluster has a corresponding pose in terms of joint angles, associated with it. We then train a classifier to distinguish between the clusters. Finally, we integrate the classification into the hand pose estimation pipeline at several stages, by firstly classifying an input point cloud to one of the earlier found clusters and using the joint angle parameters, associated with the cluster, as an initialization for the local optimization algorithm.

5.2 Obtaining the set of initial poses

To obtain a set of initial poses, we use an unsupervised clustering approach. Let $\mathcal{X}$ denote the set of all global point cloud descriptors (ESF [132] descriptors, in our case), Θ is the set of all poses, parametrized by the joint angles $\boldsymbol{\theta}$. Then, the training set for obtaining initial poses consists of pairs $(\boldsymbol{x}_i \in \mathcal{X}, \boldsymbol{\theta}_i \in \Theta)_{i=1}^{N}$. We

Algorithm 2 Extracting pose clusters from the data.

Input: $\{\boldsymbol{x}_i \in X, l_i \in \{1 \dots L\}, \boldsymbol{\theta}_i \in \Theta\}_{i=1}^{N}$
Output: $\{\{\boldsymbol{x}_i\}, l, \boldsymbol{m}_l\}_{l=1}^{\hat{L}}$

▷ Find cluster centroids in pose space

for $l \leftarrow 1 \dots L$ **do**
  $\boldsymbol{m}_l \leftarrow \text{med}\{\boldsymbol{\theta}_i : l_i = l\}$ ▷ select initial centroid
  $I_l = \{i : l_i \leftarrow l \ \ \& \ \ \max_k |\theta_i^k - m_l^k| < \tau\}$ ▷ we set $\tau = 45°$
  $\boldsymbol{m}_l \leftarrow \text{med}\{\boldsymbol{\theta}_i : i \in I_l\}$ ▷ refine centroid
end for

▷ Unite similar clusters

$D \leftarrow$ PoseDistance$(\{\boldsymbol{m}_l\}_{l=1}^{L})$ ▷ $D \in \mathbb{R}^{L \times L}$
for $(i, j) : D_{ij} < \tau$ **do**
  $I_i = I_i \cup I_j$
end for

▷ The resulting set contains $\hat{L}$ clusters

function PoseDistance$(\{\boldsymbol{m}_l\}_{l=1}^{L})$
  for $i \in 1 \dots L$ **do**
    for $j \in 1 \dots L$ **do**
      $D_{ij} \leftarrow \max_k |m_i^k - m_j^k|$
    end for
  end for
end function

employ the random forest-based clustering method, described in Section 4.2.2.2, to obtain the initial cluster label $l_i \in \{1 \dots L\}$ for each sample $\boldsymbol{x}_i$

We chose to compute the clustering in the appearance space instead of the joint angles space, as in [59], since by computing clusters in joint angle space we are risking to create clusters, that are hard to differentiate between, which, in turns, will lead to the poor initialization during the application phase.

Ideally, we would like to obtain such clusters, that:

- contain samples that are similar in appearance within one cluster and different in appearance within different clusters (i.e., intracluster variation in appearance is less then intercluster variation);

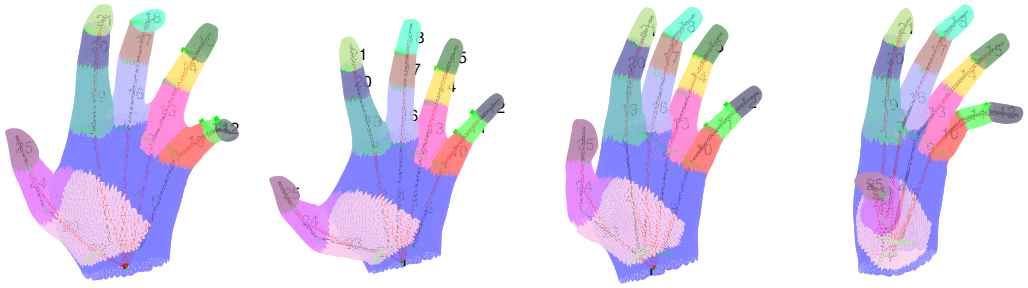


FIGURE 5.1: Hand poses, obtained from clustering the training data from the synthetic sequence [94].

- the joint angles within one cluster do not contain much variation, i.e. samples from one cluster correspond to roughly the same pose in joint angles space.

Additionally, clustering rarely provides clean clusters, therefore some post-processing is needed. The algorithm of obtaining the final clusters is described in Algorithm 2. Since the clusters are merged at the last step of post processing, the initial number of clusters L set for the clustering algorithm is not important; we set $L = 10$ in all experiments, and after the merging step it is reduced to $\hat{L} = 3$ or 4 clusters.

Note, that for pose initialization we use global rotation of the hand around X and Y axis of the camera (i.e., not the camera axes), as well as joint angles, specifying the pose. The translation is not used. We do not consider the rotation around the camera axes, since the ESF descriptor is invariant to this type of rotation (see Section 4.6.1 and Figure 4.8 in particular) and the initialization procedure from Section 3.4.1 is likely to determine the in-plane rotation correctly.

After obtaining clean clusters from the training data, we use them to train a discriminate classifier (random forest in our case), to provide a clusters prediction at run-time. We use the pose centroids, associated with each cluster, to provide an initialization for the subsequent local optimization.

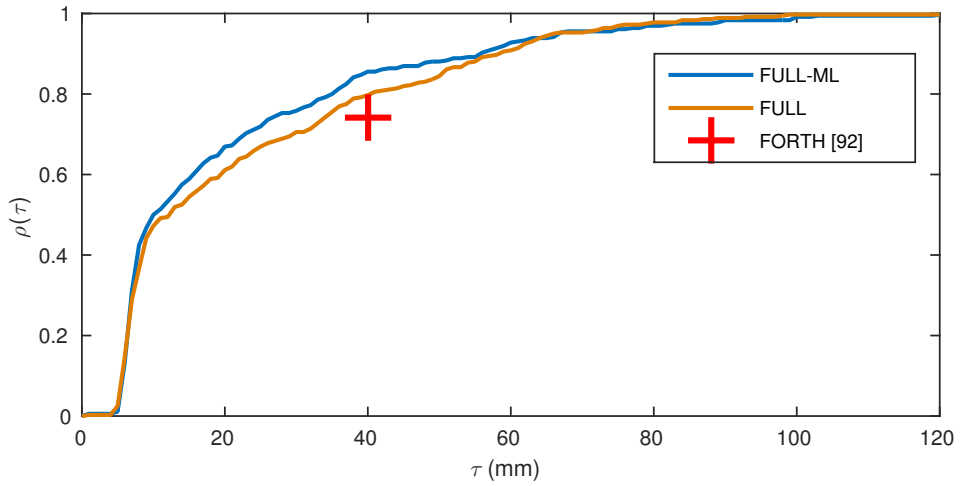


FIGURE 5.2: Comparison between **FULL** and **FULL-ML** baselines on the synthetic sequence [94] in terms of $\rho(\tau)$

5.3 Combining pose prediction with model-based pose estimation

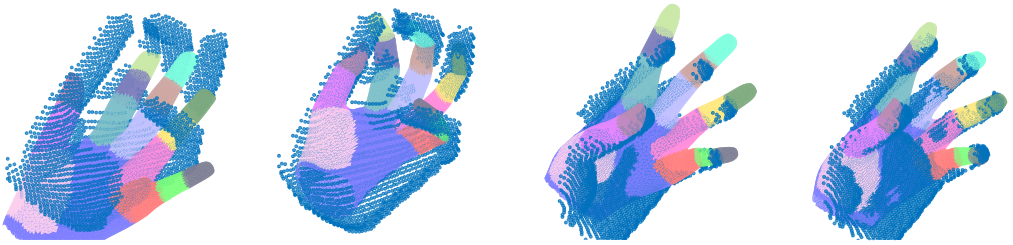
The initial pose prediction is integrated into the hand pose estimation pipeline, described in Chapter 3 at several stages. Firstly, the ESF descriptor $\boldsymbol{x}$ is computed for an input point cloud, representing a hand. The cluster label l and the corresponding pose initialization $\boldsymbol{m}_l$ is obtained using the random forest classifier, trained as described in the previous section. At the initialization stage the posed hand is used (instead of the default initialization pose) to find the global hand pose and as an initial pose when evaluating the finger matching cost (3.25) from Section 3.4.1.

Afterwards, during the refinement step (Section 3.4.2), the posed hand mesh is used in optimization as opposed to the hand mesh in zero-pose; therefore, the cost function from Section 3.4.2 is modified such as:

$$E^{\boldsymbol{m}_l}_{palm}(\boldsymbol{\xi}_0, s) = \frac{1}{|I_{palm}|} \sum_{i \in I_{palm}} \|\boldsymbol{p}_i - \boldsymbol{v}_{\mu(i, \tilde{I}^m_{palm})}(\boldsymbol{\xi}_0, s, \boldsymbol{m}_l)\|^2, \tag{5.1}$$

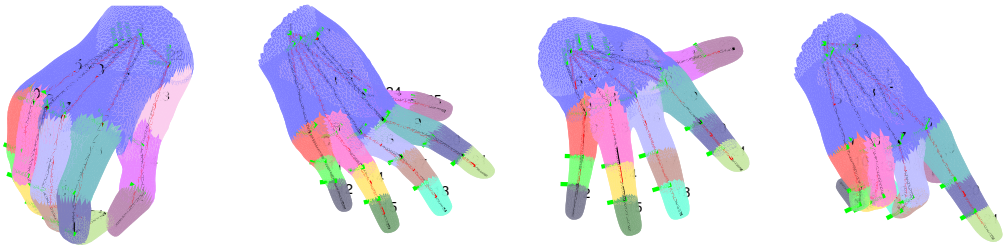
$$E^{\boldsymbol{m}_l}_{fingers}(\boldsymbol{\xi}_0, s) = \frac{1}{|I_{fingers}|} \sum_{i \in I_{fingers}} \|\boldsymbol{p}_i - \boldsymbol{v}_{\mu(i, \tilde{I}^m_{fingers})}(\boldsymbol{\xi}_0, s, \boldsymbol{m}_l)\|^2 \tag{5.2}$$

$$E^{\boldsymbol{m}_l}_{bg}(\boldsymbol{\xi}_0, s) = \frac{1}{|I^m_{palm}|} \sum_{i \in I^m_{palm}} \min_{\boldsymbol{p} \in \Omega_{palm}} \|\mathrm{Pr}(\boldsymbol{v}_i(\boldsymbol{\xi}_0, s, \boldsymbol{m}_l)) - \boldsymbol{p}\|^2 \tag{5.3}$$



(a) (b) (c) (d)

FIGURE 5.3: Typical examples from the synthetic sequence, where **FULL** method converges to the wrong result ((a),(c)), while **FULL-ML** converges to the correct result ((b),(d)).



(a) (b) (c) (d)

FIGURE 5.4: Pose centroids of the four clusters extracted from the sequences of the ICL dataset.

Finally, during the final optimization step (Section 3.4.3), the pose initialization $\boldsymbol{m}_l$ is used instead of $\boldsymbol{\theta}^{ini}$ as an initialization for the local optimization of the cost function in Equation 3.35 (Section 3.4.3).

5.4 Evaluation

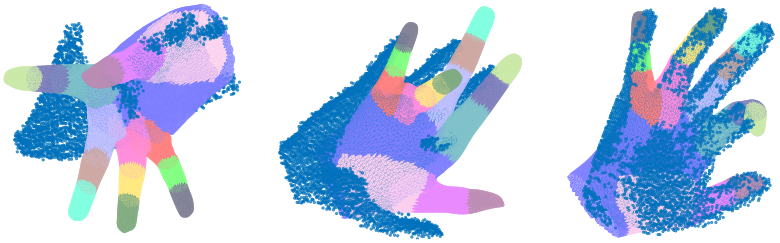
We evaluate the proposed approach on the same synthetic sequence [94], as in Chapter 3, as well as on the public dataset [116]. The [116] dataset contains annotations for hand fingertips and palm only, and the annotations are quite noisy. Therefore, we cannot use them to obtain ground truth joint angles describing the pose of the hand using these annotation. For the purpose of creating pose clusters, we employ another dataset, namely ICL [119], that provides the annotations for all hand joints. The usage of a different dataset for training and for testing presents an additional challenge, since the recording of the datasets were done in different

settings with different subjects, which introduces a so-called domain shift (see Appendix for a more detailed explanation).

To obtain the joint angles representation for ground truth annotations, that would be valid for our hand model, we directly fit the hand skeleton to the ground truth joints annotations. For the evaluation, we use the same error measurements, as described in Section 3.5.2. We further denote the combined method by **FULL-ML**.

5.4.1 Synthetic sequence

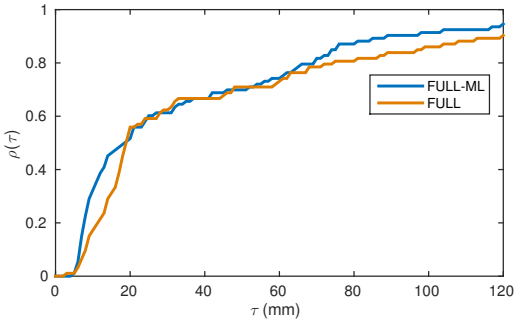
On the synthetic dataset, we compare the initial model-based method **FULL** (Section 3.5.2) with its enhancement **FULL-ML** on the previously introduced synthetic sequence. For the synthetic sequence we set the number of classes to 4 based on the poses present in the sequence. The obtained initialization poses are shown in Figure 5.1.



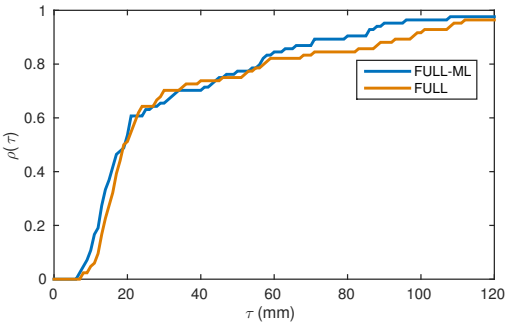
(a) (b) (c)

FIGURE 5.5: Illustration of the different m^f(mm) error values: (a) illustraits the complete failure ($m^f = 138.5$); (b) illustrates the case, when the global hand pose is determined correctly, while some fingers are mismatched ($m^f = 63.5$); (c) illustrates the correctly estimated pose ($m^f = 6.73$)

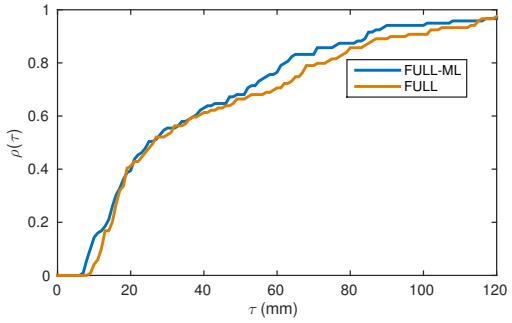
In Figure 5.2, the results for the $\rho(\tau)$ are provided. As can be seen **FULL-ML**, outperforms **FULL** significantly. Furthermore, from Figure 5.3, it is clear, that the initialization allows to estimate the pose correctly for the cases, when the **FULL** approach does not provide a good estimate of the hand pose. This is because the hand pose on these frames is too far from the initialization pose that we use, and therefore the optimization converges to the wrong result, while in case of the correct initialization provided, we are able to recover the true pose.



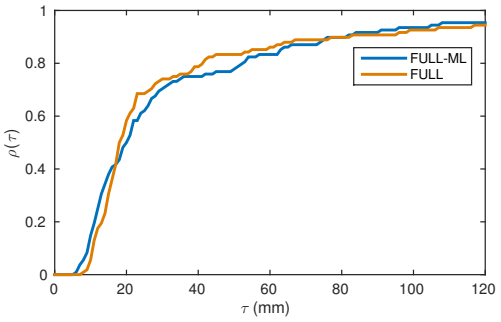
(a) adbadd



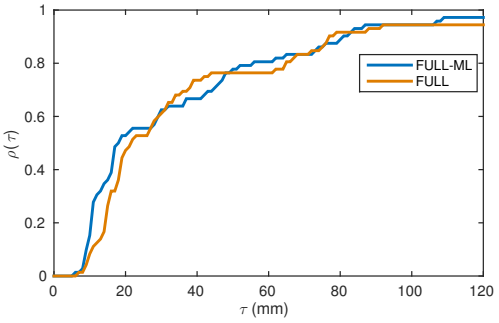
(b) fingercount



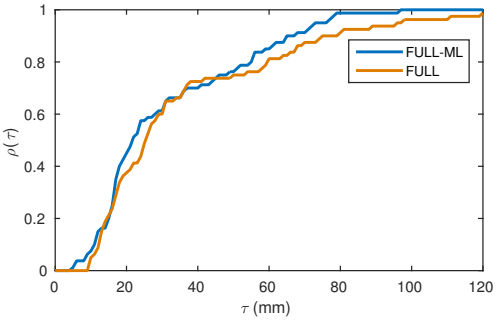
(c) fingerwave



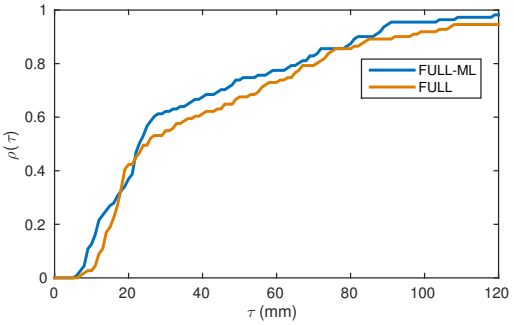
(d) flexex1



(e) pinch



(f) random



(g) tigergrasp

FIGURE 5.6: Comparison of all methods on the sequences of Dexter dataset in terms of $\rho(\tau)$; notice, that for some sequences the RF-based initialization brings significant gain (tigergrasp, random, adbadd, fingercount), while for other sequence it is not very clear; it can be explained by the fact that these sequences contain similar poses to the clusters that we extracted.

5.4.2 Evaluation on the public dataset

We further evaluate the proposed method on the Dexter dataset [116], that we already used in Chapter 3.

Dexter dataset provides annotations for the finger tips and the palm center. Unfortunately, the annotations are fairly noisy and in general are not enough to obtain good pose representation in terms of joint angles. We therefore employ another dataset for this purpose, namely ICL datatset [119].

We firstly obtain pose representation in terms of joint angles for three sequences of the ICL dataset by fitting the skeleton model to the hand joints annotations provided in the dataset. We then obtain the clusters as described in Section 5.2. Initially we obtain $L = 10$ clusters, but after pruning only $\hat{L} = 4$ clusters are left. We observed that the resulting pose centroids $\{\boldsymbol{m}_l\}_{l=1}^{\hat{L}}$ are stable with the fixed parameters of clustering and do not change after re-running the clustering algorithm. The obtained poses are shown in Figure 5.4. Note, that using a different dataset for training of the classifier presents an additional challenge, since the datasets contain different subjects and different types of motions, although both datasets are recorded with the same ToF sensor — Intel Creative Gesture Camera [2].

We then train a pose classifier and apply it for hand pose estimation on the Dexter dataset, as described in Section 5.3.

The results of the evaluation in terms of $\rho(\tau)$ are presented in Figure 5.6. It can be seen, that for some sequences the initialization brings significant gains, while for others the gain is smaller. There are two reasons for that: firstly, the clusters are obtained from different sequences, and consequently obtained poses might not even occur in the sequences of Dexter dataset. For example, *tigergrasp* sequence contains many poses similar to the cluster centroid in Figure 5.4(a), while *flexex1* sequence does not contain similar poses, except the one depictured in Figure 5.4(c). Secondly, for some sequences the other reason for incorrect pose estimation might prevail, such as completely incorrect initial guess about the global pose of the hand.

Another important thing to notice is that the improvement is visible for the median error in the interval of $[0, 20]$mm for all the sequences; that can be explained by the fact that the error less then 20mm usually means that single fingers are not fitted precisely, while the global pose is correctly estimated (see Figure 5.5 for examples).

This is precisely the case when using the correct initialization for the finger poses during the final optimization step for full DoF pose estimation (Section 3.4.3) brings the most significant gain.

Consequently, since the initialization provided by the cluster centroids does not directly influence the global hand pose, if the global pose was incorrectly estimated by the model-based algorithm from Chapter 3, the chances that the incorrect global pose will be corrected through random forest prediction are low. That said, through indirect influence of the predicted hand pose on the obtained initialization, still some improvement might be observed (see Section 5.3 for details).

5.5 Conclusions and discussion

In this chapter we combined the generative and the discriminative approach: the discriminative approach distinguishes between a finit set of poses, therefore providing a better initialization for the generative approach, that is based on a local optimization algorithm. The combined approach allows to partially compensate the weakness of the model-based pose estimation by using a better initialization of the local optimization, and in the same time allows to determine full DoF hand pose in the continuous (as opposed to discrete) pose space. We further showed in multiple experiments, that the proposed approach allows to increase hand pose estimation performance for a single frame hand pose estimation.

Chapter 6

Conclusions and future work

6.1 Summary

This work addresses the problem of full DoF hand pose estimation using a single depth sensor. Obtaining the full DoF hand pose is becoming increasingly important for such applications as human computer interactions and virtual reality. Furthermore, due to specifics of the problem, extensive use of time consistency for hand pose prediction is not feasible in many applications.

Therefore, in this work we concentrate on hand pose estimation from a single image, while many state-of-the-art trackers focus on tracking a hand over time. A traditional approach to a non-rigid body pose estimation is derived from a non-rigid ICP algorithm. However, the cost function employed in this approach is highly non-convex and contains many local minima. We therefore firstly use a simple fingertips-based heuristic to obtain an initial estimate of the global hand pose. However, the problem of initial global pose estimate is dependent on the correct hand pose, which is unknown. Additionally, even provided a correct global pose, it is not sufficient to recover the full pose using a local optimization approach.

We therefore investigate a discriminative approach for hand poses classification, which relies on the global point cloud descriptor (ESF) as a hand pose representation, classified using a multi-layered random forest. The modification allows to increase the depth of the forest while reducing the number of nodes in each tree and consequently reducing the training time and the required memory. The discriminative approach, however, has the natural drawback that it can only differentiate

between a finite number of poses, and its performance in general decreases as the number of classes to differentiate between grows.

Therefore, we propose to combine both approaches, obtaining a suitable initialization for the model-based hand pose estimation by categorizing each hand shape at run-time to one of the automatically found pose clusters. To obtain pose clusters, we cluster the appearance features of a training dataset containing hands in different poses; we argue, that clustering samples in appearance space instead of joint angles space allows us to avoid deriving clusters, that are far from each other in the joint angles space, but close to each other in the appearance space, which could lead to difficulties in their classification at test time. We evaluate the combined approach on the public dataset and show the improvement due to integration of the hand pose prediction in the model-based pose estimation pipeline.

The contributions of the thesis were published in international conferences and workshops, such as ISVC, ICCV and ECCV Workshops. The side project, described in the Appendix, is published in CVPR.

6.2 Possible directions of future work

The proposed approach has several drawbacks and although we believe that the pipeline is reasonable, different parts of the pipeline can be improved.

Firstly, the problem of extracting relevant initial pose clusters is still largely open. Ideally the clustering should not only take into account the properties of the data itself, but also the properties of the following model-based local optimization algorithm. Even doing a joint clustering in pose and appearance space might potentially further improve the results of both the discriminative classifier and the final pose estimation.

Secondly, an interesting direction of research is to further specialize the discriminative part to target a specific application, since for both virtual reality and especially human computer interaction applications certain hand poses are much more important, while recognizing other poses is mainly required for correct visualization.

Thirdly, although the tracking scenario is in general not desirable, including some temporal hints, as well as contextual hints (such as surroundings or type of the

activity performed) is definitely a promising direction of research. Additionally, testing multiple initial pose hypothesis should definitely improve the final result.

Finally, the question of the effect of personalized hand model used for both discriminative and generative parts is still not investigated, although it is natural to expect that a precise hand model will improve the performance of the generative hand fitting substantially.

Additionally, the problem of reliable pose estimation of two interacting hands (or two hands interacting with an object) remains largely unaddressed, although very common in many scenarios.

Chapter 7

Appendix

In this thesis we mainly concentrated on the problem of hand pose estimation and did not consider the applications, for example, understanding egocentric videos. In this chapter, we propose the first step in this direction by developing an object detector, which could be one of the building blocks of a video understanding system. Furthermore, we address an important problem in object detection — the problem of domain shift.

Essentially domain shift denotes the change in the distribution of features depending on the source of data or external conditions. To be more specific, in case of computer vision, domain shift usually happens when either the source of images changes (for example, from a high-resolution professional camera to a low-resolution mobile camera) or external conditions, such as lighting or weather, change. In this case, the performance of machine learning models, trained in one domain and applied in another, degrades significantly.

When considering object detection in egocentric videos, the common problem is to find well-annotated data to train a reliable object detector. Furthermore, since egocentric videos can be shot both in indoor and outdoor settings, and different indoor environments are characterized by different lighting conditions, style, etc, each egocentric video in fact represents a new domain, that might be slightly different from the previously seen ones. In this setting, it is desirable to be able to adapt the machine learning models, trained on one domain, to another domain, in unsupervised manner.

In the next chapter, we proposed a method, that allows not only to adapt a model from a source domain to a target domain, but rather to remember all

previously seen domains and improving its performance gradually as the number of the domains increases. We denote this procedure by *domain expansion*.

7.1 Improving an object detector via *Domain Expansion*

While most domain adaptation techniques focus on applications where both training and test instances are images [65, 104], taken with conventional cameras, a few address the problem in the context of image-to-video object detector adaptation [30, 35, 107, 108, 120]. The image-to-video scenario is both compelling and challenging. It is highly desirable to utilize image datasets for training detectors to be used in videos, because images are easier to label and plenty of richly labeled datasets already exist. Obtaining a video equivalent of the ImageNet [28], in terms of scope, would be an insurmountable task. However, there are often significant appearance differences between images (e.g., obtained on the web) and videos (e.g., obtained on Youtube or using egocentric cameras). Web images tend to be of high resolution and are object-centric [28]. Videos, on the other hand, often come at lower resolution, are not object centric and, at least in egocentric setting, have a widely different appearance due to the quality of the sensor and motion artifacts. Hence domain shift between images and videos is often severe (e.g., see results in [95]).

Nearly all domain adaptation techniques assume that data is separated into well defined discrete domains, most often a source (training) domain the the target (test) domain, and the task is to effectively transfer learned information (or labeled samples) from source to the target domain. This notion of discrete domains and focus on performance in only the target domain is somewhat of an oversimplification. In practice, as noted in [52], the target domain is often continuously evolving. Further, one can argue that as object instance, appearance, lighting and view are changing, the resulting evolution is actually an *expansion* of the original domain of this object, not formation of a new or evolution of the old domain. The difference is subtle. In continuous domain adaptation [52] (and incremental learning [107, 108]) the goal is to continuously adapt (or learn) a fixed complexity model to perform as accurately as possible on the arriving target batch of data. We argue for continuously adapting the complexity of the model itself. This should allow the adapted model to not only improve with respect to the arriving data,

but also to at least retain its performance on the prior and future domains. We also do not assume that data arrives in a continuously evolving stream [52].

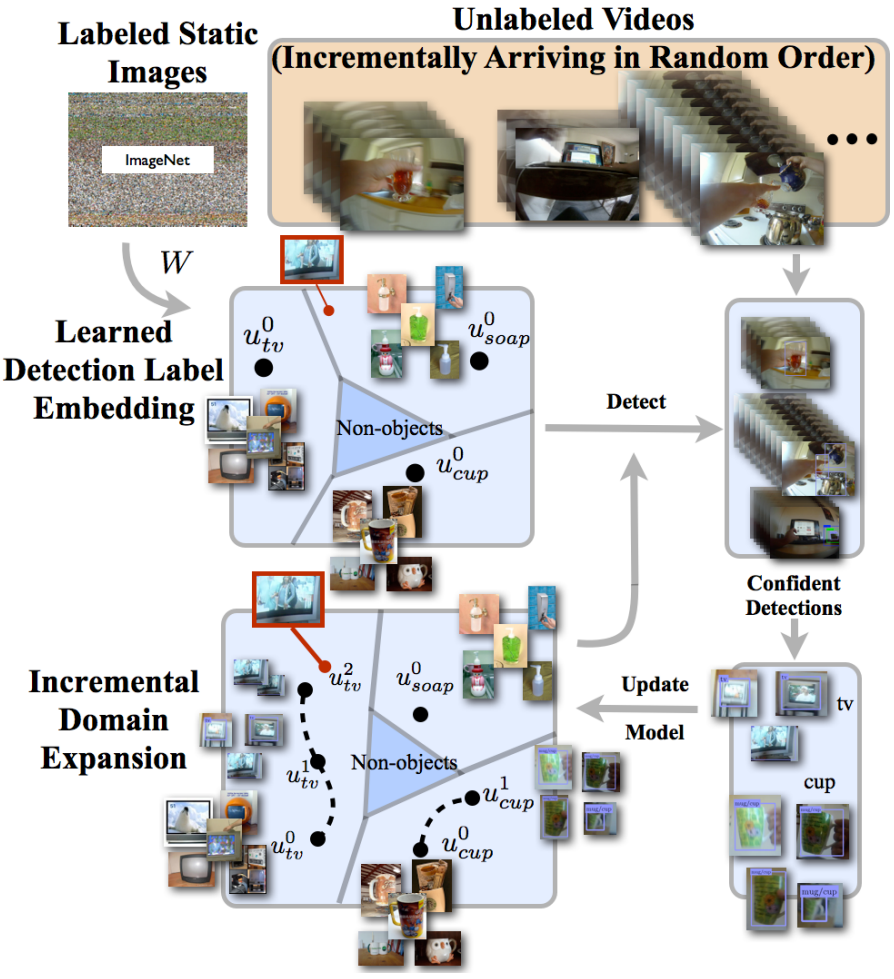


FIGURE 7.1: **Incremental Domain Expansion:** Illustration of the the overall proposed learning framework. First a large margin embedding (LME) detector is built based on labeled static images from ImageNet. As unlabeled videos arrive, detected objects are ranked based on detection confidence. Top ranked detections are expanded into tracks and used for new class prototype learning. Note that while a *TV* test sample (in red) may be too far in appearance from the original ImageNet trained model and hence misclassified, new prototypes, added based on tracks from videos, help to bridge the gap leading to correct classification.

To this end we propose an incremental, self-paced inspired, approach to expanding the domain from images to unlabeled videos. We start from a large-margin embedding (LME) model [130], which we adapt to a detection task. Using this detection model, objects in the arriving unlabeled videos are found, and tracks associated with most confident instances are extracted (Figure 7.1 (right)). If instances from these tracks form a cluster, they are further used to adjust the complexity of the

model by adding new class prototypes (Figure 7.1 (bottom left)). This process of extracting confident instances and learning expanded domain model, continues as additional videos arrive.

Our method is inspired by the overarching goals of lifelong learning [20, 112]. We note that our approach is related to sub-categorization, but unlike sub-categorization, which assumes fully labeled [51] or weakly-labeled instances [20], we work in an entirely unsupervised scenario. Further, while sub-categories, in general, do not form any sort of coherent structure in appearance space, our model ensures that object class prototypes form a coherent manifold, through regularization, limiting drift in learning.

Contributions: Our main contribution is the framework for *incremental domain expansion*, where complexity of an image-based object detection model is continuously adjusted to newly arriving unlabeled videos in a way that, over time, improves the performance on the evolving video domain but at the same time maintains (or improves) accuracy on the original image domain. Effectively, domain expansion, is about building a better overall detector using unsupervised video data. As part of this larger goal we formulate a new object detection model, inspired by the large-margin embedding (LME). We show how to extend the LME from multi-class object categorization to multi-class object detection problem, by introducing novel detection constraints to deal with the negative instances. We also propose a probabilistic formulation for LME, which allows the model to perform intuitive confidence evaluation for test instances and a novel multi-prototype LME formulation, that supports incremental learning. We show incremental domain expansion is effective in applying object detectors, trained with only ImageNet, to videos, improving performance by 48% (13% through expansion) with respect to original LME on the ADL dataset[95] and by 15% on the YTO dataset [99].

7.2 Related Work

Domain adaptation from image to image domain: Our domain expansion method is closely related to domain adaptation (DA), which is a statistical method that focuses on the adaptation of an existing model in one domain (source) to a new data domain (target). The domain adaptation can be categorized into supervised methods [8, 65, 104], where labels are available for the samples in the target domain, and unsupervised methods, where no label is provided [33, 38, 39, 53, 120].

Our method relates to the latter case, as we aim to expand a model learned on labeled images to encompass unlabeled video data. The main difference between our method from the existing method is that, while the existing methods assume that there exist multiple discrete domains, we view all domains as related, as in [52], which models the source and target data on a single continuous manifold without clear distinction between the two. Based on this assumption, our model aims to improve on both source and target domains, while most methods care only about the performance on a given target domain.

Adapting object detectors trained on images to videos: Among many DA tasks, the task of adapting detectors trained on images to unlabeled video data is a topic of particular interest, largely due to the difficulty of video data annotation. Many models resort to a strategy that selects negative and positive samples from the test data based on their confidence with respect to the existing detector [108, 120, 129]. In [129], the baseline detector with low threshold generates positive/negative samples for the new vocabulary tree-based classifier that then decides on the label. Our model also leverages an existing model to select test samples, but in our case, the model is not fixed, but is allowed to expanded in complexity. We also aim to not only improve on the video domain, but also maintain, or improve, performance on the image domain. When deciding which detections to add to the training pool, many works further exploit the temporal continuity of frames in the video data [30, 108, 120], such as [108] which utilizes tracks, and leverage the matches between tracks and confident detections from a baseline detector as additional positive samples.

Perhaps the closest works to ours are [120] and [30]. Tang et.al.. [120] proposed a self-paced method that incrementally adds positive samples, in the order to increase classification performance. Our self-paced learning algorithm is similar, but we expand the model instead of retraining it. Donahue et.al.. [30] proposed a method that incorporates an instance similarity graph to regularize the model, and applied it to the case of video data, where the distance of the instances within a track were utilized as the auxiliary instance similarity. Our method also leverages such similarities between entities, but it models group(video)-to-category similarity rather than instance-to-instance similarity, and the similarity graph is not given but is implicitly built from the order the videos arrive. Some works attempt the opposite of using weakly supervised YouTube videos to train image object detectors [99].

Self-paced learning: Our ranking of the unlabeled video samples based on their classification confidence, is related to self-paced learning [66], where the data points are presented in a meaningful order, which is often determined by the difficulty of classifying a given sample. In the original work of [66], self-paced learning was used to learn latent variables, and in [78], it was used to discover object categories from clustered image patches. Self-paced learning was also used in Tang et.al.. [120] to incrementally add in unlabeled samples into the labeled pool. Our work leverages a similar selection method, but our model considers the multi-class case while [120] considers the single-class model.

Lifelong learning: The idea of lifelong learning, which is a continuos learning that transfers the knowledge learned at earlier learning stages to later stages, was first conceived in [121], and has become an active topic of research following the success of Never Ending Language Learner (NELL) [19]. NELL is an incremental model that learns about new concepts and rules by continuously observing textual input. Similar work has been proposed for the case of image data in [20]. Our work can be also viewed as an instance of lifelong learning, since the model is incrementally improved leveraging continuous stream of inputs, and the new subcategory prototypes are learned in the context of existing category prototypes.

Large-margin manifold embedding models for recognition: Our model builds on the large-margin (class) embedding (LME) [12, 130, 131], which aims to learn a low-dimensional space that is optimized for class discrimination. LME recently gained popularity, largely due to it's ability to scale to many class labels, which is becoming increasingly important with image classification becoming more focused on large-scale datasets. While there are many variants of LME, the one that is particularly relevant is [87], which presents a probabilistic multi-centroid model, that bares similarity to ours. However, the k-centroids for each class in [87] are obtained from k-means clustering on the original labeled samples, while in our model the multiple centroids are incrementally added as the model expands with new videos.

7.3 Incremental Learning Framework

We consider the general problem of applying an object detector, trained on images, for detecting objects in videos in a completely unsupervised manner.

To formally state the problem, given a training image set $\mathcal{D}_{\mathcal{I}} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N_{\mathcal{I}}}$, such as ImageNet [28], that has $N_{\mathcal{I}}$ labeled instances, where $\boldsymbol{x}_i \in \mathbb{R}^D$ is a D-dimensional feature descriptor of an image patch containing an object and $y_i \in \{1, \ldots, C\}$ is the object label, we propose to first learn a large-margin embedding (LME)-based object detection model. The choice of proposing an embedding-based detection paradigm over the more traditional SVMs or latent SVM, stems from flexibility and scalability of such models, their ability to generalize with little to no data [87], as well as their state-of-the-art performance on large-scale categorization tasks [34].

Once the initial LME detection model is trained (Sections 7.3.1 and 7.4.1), we want to utilize unlabeled data from a sequence of arriving videos to incrementally improve the learned model. We propose an incremental learning framework that iteratively refines and adds complexity to the model as it is needed and consists of the following steps:

1. From each video we extract object proposals $\{\boldsymbol{b}_i\}_{i=1}^{N_v}$, using [124], and corresponding feature vectors $\{\boldsymbol{x}_i\}_{i=1}^{N_v}$.
2. We evaluate each $\boldsymbol{x}_i$ using the proposed probabilistic multi-center LME model to obtain a set of detections, labels and corresponding confidences $\mathcal{D}_v = \{\boldsymbol{x}_i, c_i, p(y = c_i, d = 1|\boldsymbol{x}_i)\}_{i=1}^{N_v}$ (see Section 7.4.5)
3. We extend the set of detections by exploiting temporal consistency (see Section 7.4.5).
4. Finally, we update the model using selected samples, as described in Section 7.4.4.

This process continues while videos arrive. The framework is illustrated in Figure 7.1.

7.3.1 Background: Large Margin Embedding

Large-margin embedding [130] is a method for classification that projects samples into a low-dimensional space in a way that achieves separation among instances belonging to different classes, with respect to Euclidean metric.

As above, we denote the labeled training data[1] as $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$. The goal of LME is to learn a linear low-dimensional embedding defined by a projection matrix

[1] We drop $\mathcal{I}$ subscript to avoid clutter.

$W \in \mathbb{R}^{d \times D}$ ($d \ll D$), together with class prototypes $\boldsymbol{u}_c \in \mathbb{R}^d, c = \{1 \dots C\}$, in the embedding space, such that a sample projected into this low dimensional space is closer to the correct class prototype than to all other prototypes.

Let us denote $d(\boldsymbol{z}_i, \boldsymbol{u}_c)$ as a similarity measure between a projected feature vector $\boldsymbol{z}_i = W\boldsymbol{x}_i$ and a prototype $\boldsymbol{u}_c$. The LME objective described above can be encoded by a positive margin between similarity of $\boldsymbol{z}_i$ and its true prototype and all the other prototypes:

$$\begin{aligned} &d(\boldsymbol{z}_i, \boldsymbol{u}_{y_i}) + \xi_{ic} \geq d(\boldsymbol{z}_i, \boldsymbol{u}_c) + 1, \\ &i = \{1 \dots N\}, c = \{1 \dots C\}, c \neq y_i, \end{aligned} \tag{7.1}$$

where ξ_{ic} play the role of slack variables that we want to minimize. The learning of the optimal W and $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_c\}$ can be formulated as minimization of:

$$\sum_{i,c:c \neq y_i} \xi_{ic}^+ + \lambda \|W\|_F^2 + \gamma \|U\|_F^2, \tag{7.2}$$

where U us the columnwise concatenation of prototypes $\boldsymbol{u}_c$, ξ^+ is defined as $\max(\xi, 0)$ and λ and γ are weights of the regularizers. Here $\|\cdot\|_F$ denotes Frobenius norm. The label of a new sample $\boldsymbol{x}^*$ at the test time can then be determined by comparing the similarity of this new sample to prototypes in the embedding space:

$$y^* = \operatorname*{argmax}_c d(\boldsymbol{z}^*, \boldsymbol{u_c}) = \operatorname*{argmax}_c d(W\boldsymbol{x}^*, \boldsymbol{u_c}). \tag{7.3}$$

In the initial formulation [130], L2-based similarity measure was used, however, we employ the scalar product to measure similarity in the embedding space

$$d(\boldsymbol{z}_i, \boldsymbol{u}_c) = d_W(\boldsymbol{x}_i, \boldsymbol{u}_c) = \langle W\boldsymbol{x}_i, \boldsymbol{u}_c \rangle. \tag{7.4}$$

7.4 Multi-prototype LME for object detection

The initial LME model is designed for the *object classification* task. We extend the LME formulation to be applicable for *object detection* and provide the corresponding probabilistic interpretation. We also derive a multi-prototype formulation and present an algorithm for incremental learning.

7.4.1 LME model for object detection

The trivial way to extend the LME model for object detection is to assume existence of a *non-object* class. However, this would lead to modeling of this *non-object* class in LME using a *non-object* prototype. Since the variability in the appearance within the *non-object* class is much higher then within any other class, this may not be ideal.

Hence, instead, we define a patch as not containing an object of interest if it is sufficiently *dissimilar* to all known object class prototypes. This can be expressed as a set of additional large-margin constraints in the optimization:

$$d_W(\boldsymbol{x}_j^0, \boldsymbol{u}_c) \leq 1 + \xi_{j0}, \quad c = \{1, \ldots, C\}, \xi_{j0} \geq 0, , \tag{7.5}$$

that require the similarity to be low (distance to object prototypes high) for the non-object samples. Here $\boldsymbol{x}_j^0, \quad j = \{1, \ldots, N_0\}$ are patches, that do not contain any object of the target classes, and ξ_{j0} are positive slack variables.

We note that for our specific similarity measure this actually pushes negative samples towards the center of the embedding space and effectively amounts to feature selection (or suppression) between all positive and a negative class; for other metrics, e.g., a Euclidian metric, the geometric interpretation would be different.

The training objective is changed respectively to:

$$\sum_{i,c:c\neq y_i} \xi_{ic}^+ + \sum_j \xi_{j0}^+ + \lambda \|W\|_{FRO}^2 + \gamma \|U\|_{FRO}^2. \tag{7.6}$$

The prediction for a new feature vector $\boldsymbol{x}^*$ is formulated as follows:

$$y^* = \begin{cases} \text{argmax}_c \, d_W(\boldsymbol{x}^*, \boldsymbol{u_c}), & d_W(\boldsymbol{x}^*, \boldsymbol{u_c}) \geq \tau, \\ c_0, \quad \forall c = 1, \ldots, C: & d_W(\boldsymbol{x}^*, \boldsymbol{u_c}) < \tau, \end{cases} \tag{7.7}$$

where τ is chosen based on the precision-recall trade-off, and c_0 denotes a non-object class.

Numerical optimization: Optimization in Equation (7.6), with the corresponding constraints, is bi-convex in W and U. We optimize Equation (7.6) using alternating optimization, where we alternate between solving for U and W while keeping the other variable fixed, using stochastic gradient descent. The alternation process is repeated until the convergence criterion is met[2].

[2] $\|U - U_{prev}\|_2 + \|W - W_{prev}\|_2 \leq \epsilon$

7.4.2 Probabilistic LME interpretation

Estimation of confidence of the detector will be critical in ordering and selecting samples for domain expansion. In [120] authors use the value of the loss (or margin) as confidence. We, however, are dealing with a multi-class problem, so instead we derived the following probabilistic interpretation of the LME. We define the posterior probability of a sample that is considered to be a detection to belong to class c, by mapping the similarity between the projected instance and a class embedding to the range between 0 and 1 as follows:

$$p(y = c|d = 1, \boldsymbol{x}) = \frac{e^{d_W(\boldsymbol{x},\boldsymbol{u}_c)/2\sigma^2}}{\sum_{i=1}^{C} e^{d_W(\boldsymbol{x},\boldsymbol{u}_i)/2\sigma^2}}. \tag{7.8}$$

where $d = 1$ indicates that a sample is considered to be a detection.

In this setting, the probability of $\boldsymbol{x}$ being a detection can be formulated as follows:

$$p(d|\boldsymbol{x}) = \frac{1}{1 + e^{ad_W^m(\boldsymbol{x})+b}}, \tag{7.9}$$

where $d_W^m(\boldsymbol{x}) = \max_c d_W(\boldsymbol{x}, \boldsymbol{u}_c)$, and a, b are parameters, serving the same purpose as τ in (7.7). Therefore, given a sample $\boldsymbol{x}^*$, the probability of detection of an instance of a class c is defined as:

$$p(y^* = c, d|\boldsymbol{x}^*) = p(y^* = c|d, \boldsymbol{x}^*)p(d|\boldsymbol{x}^*), \tag{7.10}$$

that is interpreted as a detection confidence for a class c.

7.4.3 Multi-prototype LME

Domain shift is accompanied by change in feature distribution in the original space and consequently, in the low-dimensional embedding space. This shift causes the performance decrease of a detector and to cope with such domain shift, we need a more flexible class representation in the embedding space. Following the work of [87], we learn several, K_c, prototypes for each class c to represent multimodal feature distribution across domains: $U_c = [\boldsymbol{u}_c^1, \ldots, \boldsymbol{u}_c^{K_c}]$. Then, the similarity score between an instance and a class can be computed using the similarities to the different prototypes of the same class:

$$\tilde{d}_W(\boldsymbol{x}_i, U_c) = f(d_W(\boldsymbol{x}_i, \boldsymbol{u}_c^1), \ldots, d_W(\boldsymbol{x}_i, \boldsymbol{u}_c^{K_c})). \tag{7.11}$$

Different choices exist for the function $f(\cdot)$. However, in spirit of LME, $f(\cdot) = \max_k d_W(\boldsymbol{x}_i, \boldsymbol{u}_c^k)$, seems like an appropriate choice.

We further replace max(·) function by its smooth approximation $S_W^\alpha(\boldsymbol{x}_i, U_c)$ to simplify the numerical optimization of Equation (7.5)-(7.6):

$$S_W^\alpha(\boldsymbol{x}_i, U_c) = \frac{\sum_{k=1}^{K_c} d_W(\boldsymbol{x}_i, \boldsymbol{u}_c^k) e^{\alpha d_W(\boldsymbol{x}_i, \boldsymbol{u}_c^k)}}{\sum_{j=1}^{K_c} e^{\alpha d_W(\boldsymbol{x}_i, \boldsymbol{u}_c^j)}}, \tag{7.12}$$

where the greater the parameter α, the better the function approximates max(·). The optimization problem for multi-prototype model can be formulated in the same manner as the LME model with detection constraints in Equation(7.5)-(7.6) by replacing $d_W(\boldsymbol{x}, \boldsymbol{u}_c)$ with $S_\alpha(\boldsymbol{x}_i, U_c)$.

7.4.4 Incremental multi-prototype LME model expansion

The multi-prototype LME model is naturally suitable for domain expansion. As the model encounters new data, which is not well approximated by the current prototypes, we can add new prototypes incrementally to more precisely model the feature distribution in the embedding space. The problem of learning a new prototype then can be formulated within the LME framework as the following incremental learning procedure.

Suppose we want to expand the prototype-based representation for the class c_n. When adding a new prototype $\boldsymbol{u}_{c_n}$ to the model it should satisfy two properties: (i) the new prototype should be representative and discriminative for its class; (ii) it should not cause misclassification of samples from other classes, i.e., it should be sufficiently far from existing category prototypes for other classes. More formally, the optimization problem can be formulated as follows:

minimize:

$$\sum_{\substack{i,c: y_i = c_n, \\ c \neq c_n}} \xi_{ic}^+ + \sum_{i: y_i \neq c_n} \zeta_i^+ + \sum_j \xi_{j0}^+ + \nu \|\boldsymbol{u}_{c_n} - \boldsymbol{u}_0\|^2 + \\ + \eta \|W - W_0\|^2, \tag{7.13}$$

subject to:

$$S_W^\alpha(\boldsymbol{x}_i, \tilde{U}_{c_n}) + \xi_{ic} \geq S_W^\alpha(\boldsymbol{x}_i, U_c) + 1, y_i = c_n \tag{7.14}$$

$$S_W^\alpha(\boldsymbol{x}_i, U_{y_i}) + \zeta_i \geq S_W^\alpha(\boldsymbol{x}_i, \tilde{U}_{c_n}) + 1, y_i \neq c_n \tag{7.15}$$

$$S_W^\alpha(\boldsymbol{x}_j^0, \tilde{U}_{c_n}) \leq 1 + \xi_{j0}, \tag{7.16}$$

where W is a newly learned data embedding, W_0 is the existing data embedding, $\boldsymbol{u}_0$ is the original prototype for the given category, and $\tilde{U}_{c_n} = [U_{c_n}, \boldsymbol{u}_{c_n}]$. Equation (7.14) is a softmax LME constraint between the new category and the

existing categories, Equation (7.15) is the same constraint between each of the existing categories to the new category embedding, and Equation (7.16) is the detection constraints. The parameters ν and η are the regularization weights[3] which determine how similar the newly learned embeddings should be to the original category and data embeddings. The optimization problem in Eq (7.13)-(7.16) is non-convex, but provided with a good initialization stochastic gradient descent allows to obtain reasonable local minima.

The incremental update is especially beneficial, when not all data is available and the newly arriving data has a different, or evolving, feature distribution. However, to apply the derived model, the remaining core question is how to select the samples from unlabeled videos for the incremental model update; we address this in the next section.

7.4.5 Discovering objects from unlabeled video

Initial detection set extraction: Given an unlabeled video, we first extract the initial set of detections by computing object proposals $\{\boldsymbol{b}_i\}_{i=1}^{N_v}$ and their features, using off-the-shelf proposal method [124]. Then we extract visual feature $\boldsymbol{x}_i$ for each object proposal i and evaluate them using the multi-prototype model to obtain probability score for each proposal. Then a set of detected objects $\mathcal{D}_v = \{\boldsymbol{x}_i, c_i, p(y = c_i, d = 1|\boldsymbol{x}_i)\}_{i=1}^{D_v}$ could be formed by selecting the object proposal i, s.t. $p(y = c_i, d = 1|\boldsymbol{x}_i) > \nu$, where ν is some threshold; $\nu = 0.6$ allowed us to obtain fairly good results in our experiments. The obtained set of detection $\mathcal{D}_v$ can be then used as new positive training samples to train the new category prototype.

Tracks formation: To obtain more samples, we further exploit the temporal consistency, i.e., if object is detected in one frame, it is likely to persist for a number of frames at relatively similar position and scale.

Specifically, we employ idea proposed in [120] and extract tracks from a video using the KLT tracker [85, 122]. After computing a set of confident object proposals $\mathcal{D}_v$ with the corresponding bounding boxes $\{b_i\}_{i=1}^{|D_v|}$, for each object proposal bounding box b_i, we select the longest track t_i that intersects it. We then compute the relative positions of the object proposals that intersect this track t_i across frames, and at each frame select the proposal that has the highest PASCAL overlap[4] with

[3] In practice, we set η to a high number to prevent model drift.

[4] $overlap(b_1, b_2) = \frac{area(b_1 \cap b_2)}{area(b_1 \cup b_2)}$

TABLE 7.1: Detection performance for each class and all categories averaged by mAP on the ADL dataset, for the baselines and our method's variants. We also report the detection results on the ImageNet subset (ImNet) containing the 8 classes from the ADL dataset.

	bottle	fridge	mwave	mug/cup	oven/stove
DPM[95]	**9.8**	0.4	20.2	14.8	0.1
GK [38]	2.11	1.77	41.19	14.70	19.57
LME	0.00	0.28	3.07	0.00	0.52
LME-A	1.93	**3.42**	40.30	18.34	27.84
LME-D	1.69	1.63	39.87	13.06	19.33
LME-DT	1.85	1.76	52.37	15.91	24.54
IDE-LME	2.04	2.73	**56.69**	**21.86**	**29.94**
	soap	tap	tv	av.	ImNet
DPM[95]	**2.5**	0.1	26.9	9.35	–
GK [38]	0.20	1.62	**60.67**	17.73	–
LME	0.03	0.55	3.73	1.02	–
LME-A	0.37	1.46	53.26	18.36	76.96
LME-D	0.35	1.67	40.64	14.78	78.91
LME-DT	0.42	**2.41**	56.16	19.43	–
IDE-LME	0.25	2.26	59.53	**21.91**	**79.23**

b_i swept across the track. In this way we obtain a set of object proposals for each b_i, which constitute a track. To obtain track score, we evaluate them and accept track if more then half of the detections on the track have $p(y^* = c, d = 1|\boldsymbol{x}) > \nu$. If a track is accepted, we add all the samples from the track to $\mathcal{D}_v$.

7.5 Experiments

We validate our method on real-world image and video datasets. For image dataset, to train the base detector, we use the subsets of the ImageNet [28] dataset (ca. 600 images per class) with the corresponding classes. We use a disjoint subset of the ImageNet [28] (ca. 400 images per class) to report detector performance on images before and after incremental domain expansion where appropriate.

We test our method on the Activities of Daily Living (ADL) [95] and the YouTube Objects (YTO) [99] datasets[5]:

ADL Dataset: The ADL [95] dataset contains 20 first-person videos, recorded by different subjects. This is a challenging dataset and straightforward application

[5]Note that our domain expansion method is entirely unsupervised and the annotations on the target datasets are only used for evaluation.

FIGURE 7.2: Illustration of data and detection results. Top row: example of images for ImageNet, used for training of the initial model. Bottom row: instances of detections that were correctly detected using IDE-LME. Notice the significant differences in how objects appear in ImageNet and ADL dataset.

of the detector learned on static images does not work well [95], since the objects suffer from large viewpoint/scale variations and occlusions due to interactions. Each video in the ADL dataset has bounding box annotations for objects from 48 classes. We select a subset of the 8 most frequently encountered classes, namely *bottle*, *fridge*, *microwave*, *mug*, *oven*, *soap liquid*, *tap*, and *tv*, to test our model.

YTO Dataset: The YTO [99] dataset consists of the collection of internet videos, each video containing a single object out of 10 classes: *aeroplane*, *bird*, *boat*, *car*, *cat*, *cow*, *dog*, *horse*, *motorbike*, *train*. The dataset is divided into train and test parts, where the test portion contains a single frame with bounding box annotation of the target object per video. For the evaluation we used the test part of the dataset only; it contains $15 - 60$ videos for each class.

We use the following methods as baselines:

DPM: Performance reported in the papers [56, 95] obtained using Deformable Part Model [32].

GK: An approach of [38] for unsupervised domain adaptation. We first select samples from $\mathcal{D}_v$ and use them for learning feature transformation between the source and the target domains. We then use the learned mapping to reproject all

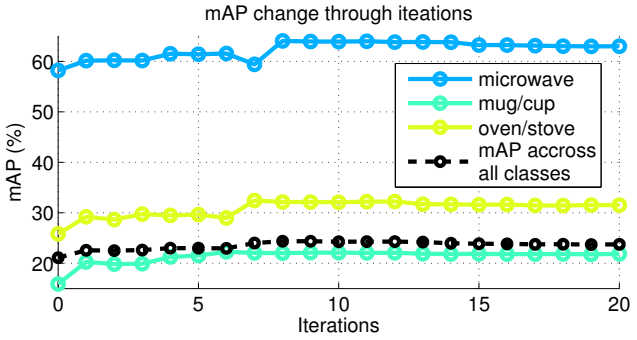


FIGURE 7.3: mAP as a function of videos seen (x-axis) for subset of classes in ADL dataset [95]; the mAP is average across videos used for expansion and the rest of the videos in the ADL dataset; the increase of mAP illustrates, that as the model gains complexity, the performance improves also on unseen videos.

features from the target (video) domain to the source (image) domain and perform detection in the source domain.

LME: A baseline LME model that formulates a prediction using Equation (7.7).

LME-A: A baseline doman adaptation (DA) approach that adapts to video domain in a batch (without increasing model complexity): we select confident samples, as described in Section 7.4.5, and re-train our model using baseline LME detector as initialization.

To show the performance gain obtainable by each step of our algorithm, we implement the following variants:

LME-D: LME with the detection constraints in Equation (7.5).

LME-DT: LME with the detection constraints and exploiting temporal consistency by using tracks.

IDE-LME: Our full probabilistic multi-centroid LME model with detection constraints, that is incrementally expanded with the unlabeled data.

We use Caffe features [55], which are deep image representations obtained at layer *fc7* of a convolutional neural network, for all LME baselines and our variants.

TABLE 7.2: Detection performance (mAP) for each class and mean mAP across all classes on the YouTube Objects (YTO) dataset and the performance of the final model on the ImageNet (ImNet).

	aero	bird	boat	car	cat	cow
DPM[56]	30.79	10.46	0.97	48.62	18.30	33.69
GK[38]	**40.05**	23.16	24.44	32.62	**24.26**	38.26
LME	35.00	24.13	16.08	27.41	4.30	31.18
LME-A	39.78	**35.18**	35.20	**48.67**	15.02	37.90
LME-D	29.61	22.91	32.39	25.53	18.63	38.94
LME-DT	31.67	21.83	40.13	25.94	17.59	41.44
IDE-LME	33.07	21.40	**42.26**	34.49	18.33	**46.92**
	dog	horse	mbike	train	av.	ImNet
DPM[56]	13.67	**26.78**	35.85	**23.98**	24.31	–
GK[38]	24.23	17.75	**36.27**	10.69	27.17	–
LME	2.12	0.23	6.83	10.30	15.75	–
LME-A	**30.70**	25.86	28.93	10.82	**30.80**	79.91
LME-D	15.55	9.22	31.47	12.09	23.63	83.16
LME-DT	15.47	11.74	30.56	13.67	25.00	–
IDE-LME	17.24	11.83	34.73	12.50	**27.28**	**83.20**

7.5.1 Quantitative Evaluation

We evaluate object detection performance of the baselines and our models using mean average precision (mAP) [31] on ADL (Table 7.1) and YTO (Table 7.2) datasets. For both datasets we observe that while the baseline LME model trained on the images without detection constraints performs very poorly, adding detection constraints results in performance on par with DPM baselines. Incorporating temporal consistency using tracks (LME-DT) improves the performance by over 31% with respect to LME-D for ADL dataset (5% for YTO dataset). Incrementally updating the model (IDE-LME) using our approach brings further significant performance improvement of 13% (9% for YTO dataset) in comparison with LME-DT, leading to overall 48% and 15% improvement over LME-D on ADL and YTO dataset respectively. The smaller performance gain on YTO dataset can be attributed to the fact that for YTO dataset, feature distribution is similar to that of images as each video contains one or few objects in typical viewpoints; another reason is sparse annotations of the YTO dataset, that limit the ability to estimate the performance improvement.

Note that the other baseline, GK, outperforms IDE-LME on the classes with high initial precision (e.g. *tv* and *microwave* for ADL dataset), while performs significantly worse on the other classes. We believe that such classes effectively determine GK transformation, while the change in the distribution of other classes

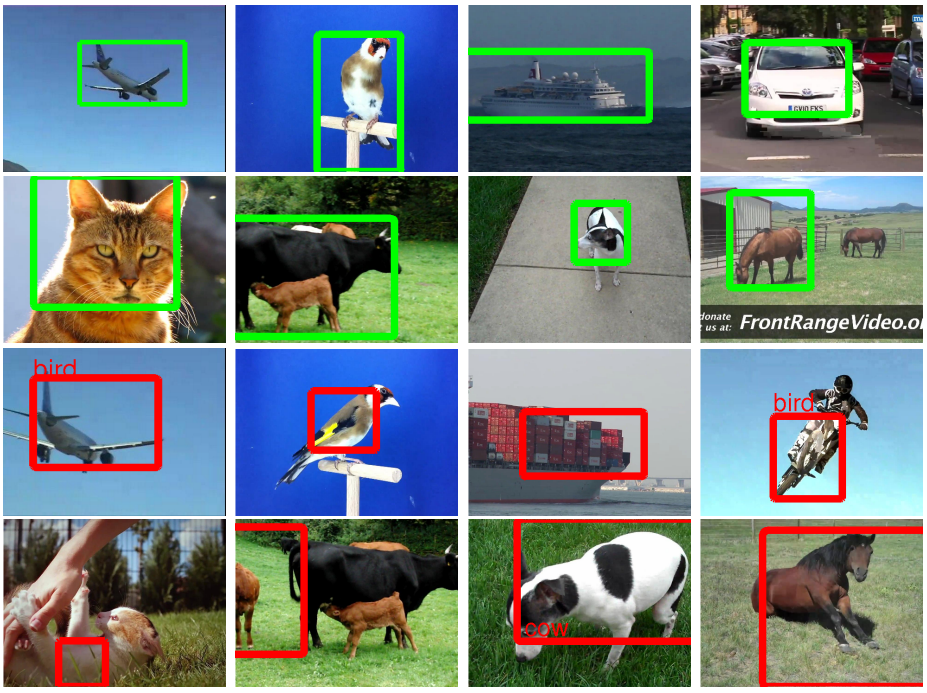


FIGURE 7.4: Top row: examples of the correctly detected objects. Bottom row: examples of the incorrect detections (from left to right), due to incorrect classification, inaccurate bounding boxes, or incorrect labels.

is not taken into account. Another trend seen on both datasets is that the initial model should have enough precision to be able to select samples from the videos for the update to work effectively, otherwise a slight performance drop can occur (*soap liquid* class in ADL dataset or *cat* class in YTO dataset).

Above experiments suggest that increased complexity of the model captures previously unseen variations in the object class appearance. To support this claim and to show that our model also improve on the original image domain, we report classification results on the test split of ImageNet dataset. In Table 7.1 and 7.2 we observe small but positive gains on the ImageNet, over LME-D. This suggests that newly added samples do not only improve the detection performance for the test video data, but also improve the classification performance on the source image data. Note that our domain adaptation (DA) baseline LME-A improves on videos but degrades on source image domain (a typical behavior for DA), on both datasets.

The performance of the model generally increases with more observed videos, but asymptotes after first 10 iterations for ADL dataset (see Figure 7.3). This early performance saturation might be due to high appearance similarity among objects in the target domain (egocentric videos). YTO dataset shows a similar

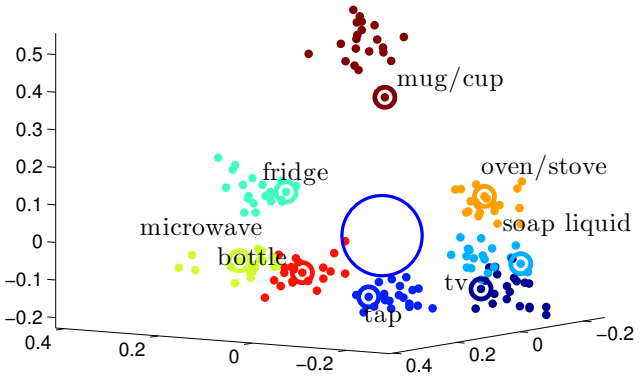


FIGURE 7.5: The visualization of the learned (expanded) multi-center LME, on the ADL dataset [95], projected into the 3D space; a group of prototypes of the same color represents a class; the initial prototypes are marked with additional circle around them.

trend. However, if target domain constantly changes or evolves over time, the performance might continue to increase.

7.5.2 Qualitative Analysis

Figure 7.2 show examples detections on the ADL dataset. Notice the significant difference between the source domain and the target domain. Figure 7.5 is the 3D visualization of the learned 8-dimensional embedding, where each category is represented as a set (manifold) of category prototypes which were expanded over the learning process. We observe that for some object classes, such as *mug*, the later added prototypes are placed far from the original center, that represents feature distribution change between the *mug* class in the ImageNet dataset and in the ADL dataset.

Figure 7.4 shows the detection examples on the YTO dataset, obtained using IDE-LME. Note that object is often identified correctly, but the bounding box is either too small or too large. We attribute this to the fact that background comprises large portion of ImageNet images, which might rank loose detections higher than tight ones.

7.6 Conclusion

We have tackled the problem of domain expansion, where the scope of the object detector learned on the initial image labeled training set is incrementally expanded to cover incoming unlabeled videos. To this end, we have developed a novel online probabilistic multi-center large margin embedding model with detection constraints, where each object category is represented with multiple prototypes, which incrementally increase in number as self-paced learning algorithm selects confident samples from the incoming unlabeled data to add. Experimental validations on the ADL and YTO public datasets shows that the proposed model significantly improves the detection performance not only on the target unlabeled videos, but also on the source image domain. Our incremental domain expansion model could serve as a lifelong learning system for object detection—as the model expands to encompass continuous stream of unlabeled new video data. One potential problem that might arise is *model drift*. We have not seen this in our experiments and our regularization is designed to prevent this, but it is possible that such drift may arise with much larger scale datasets. As future work, we plan to explore a human-in-the loop system with active learning to prevent such drift, such that model can essentially self-train itself, with infrequent human intervention only triggered by the model's request.

Bibliography

[1] Fingerspelled alphabet. http://lifeprint.com/.

[2] Intel creative gesture camera. https://software.intel.com/en-us/perceptual-computing-sdk.

[3] Kinect camera. http://www.xbox.com/de-DE/Xbox360/Accessories/kinect.

[4] Kinect2 camera. http://www.microsoft.com/en-us/kinectforwindows/.

[5] Vicon products. www.vicon.com/.

[6] Alexandre, L. A. 3D descriptors for object and category recognition: a comparative evaluation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshops (IROSW)* (Vilamoura, Portugal, 2012).

[7] Amit, Y., and Y, D. G. Shape quantization and recognition with randomized trees. *Neural Computation* (1997), 1545–1588.

[8] Aytar, Y., and Zisserman, A. Tabula rasa: Model transfer for object category detection. In *IEEE International Conference on Computer Vision (ICCV)* (2011).

[9] Balan, A., Sigal, L., Black, M. J., Davis, J., and Haussecker, H. Detailed human shape and pose from images. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR* (Minneapolis, June 2007), pp. 1–8.

[10] Ballan, L., Taneja, A., Gall, J., Gool, L. V., and Pollefeys, M. Motion capture of hands in action using discriminative salient points. In *IEEE European Conference on Computer Vision (ECCV)* (Firenze, October 2012).

[11] Baran, I., and Popović, J. Automatic rigging and animation of 3d characters. In *ACM special interest group on Computer GRAPHics and Interactive Techniques (SIGGRAPH)* (New York, NY, USA, 2007), ACM.

[12] Bengio, S., Weston, J., and Grangier, D. Label embedding trees for large multi-class task. In *Advances in Neural Information Processing Systems (NIPS)* (2010).

[13] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975).

[14] Besl, P. J., and McKay, N. D. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 14*, 2 (1992).

[15] Billiet, L., Oramas Mogrovejo, J. A., Hoffmann, M., Meert, W., and Antanas, L. Rule-based hand posture recognition using qualitative finger configurations acquired with the kinect. In *International Conference on Pattern Recognition Applications and Methods* (2013).

[16] Bray, M., Koller-Meier, E., Mueller, P., Gool, L. V., and Schraudolph, N. N. 3d hand tracking by rapid stochastic gradient descent using a skinning model. In *European Conference on Visual Media Production (CVMP)* (2004).

[17] Breiman, L. Random forests. *Machine Learning Journal 45*, 1 (2001).

[18] Brox, T., Rosenhahn, B., Gall, J., and Cremers, D. Combined region- and motion-based 3d tracking of rigid and articulated objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 32*, 3 (2010).

[19] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. Toward an architecture for never-ending language learning.

[20] Chen, X., Shrivastava, A., and Gupta, A. Neil: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision (ICCV)* (2013).

[21] Criminisi, A., Shotton, J., and Konukoglu, E. Decision forests: A unified framework for classification, regression, density estimation, manifold

learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision* (2012).

[22] Cui, Y., Schuon, S., Chan, D., Thrun, S., and Theobalt, C. 3d shape scanning with a time-of-flight camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010).

[23] Dardas, N., and Petriu, E. Hand gesture detection and recognition using principal component analysis. In *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA)* (2011).

[24] Dasarathy, B. V., and Sheela, B. V. Composite classifier system design: concepts and methodology. In *Proceedings of the IEEE* (1979), vol. 67.

[25] de La Gorce, M., Fleet, D. J., and Paragios, N. Model-based 3d hand pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 33*, 9 (2011).

[26] de La Gorce, M., Paragios, N., and Fleet, D. J. Model-based hand tracking with texture, shading and self-occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).

[27] den Bergh, M. V., Bosche, F., Koller-Meier, E., , and Gool, L. V. Haarlet-based hand gesture recognition for 3d interaction. In *IEEE Workshop on Applications of Computer Vision (WACV)* (2009).

[28] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and ei, L. F.-F. Imagenet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[29] Doliotis, P., Athitsos, V., Kosmopoulos, D. I., and Perantonis, S. J. Hand shape and 3d pose estimation using depth data from a single cluttered frame. In *International Symposium on Visual Computing (ISVC)* (2012).

[30] Donahue, J., Hoffman, J., Rodner, E., Saenko, K., and Darrell, T. Semi-supervised domain adaptation with instance constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[31] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal on Computer Vision (IJCV)* (2010).

[32] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2010).

[33] Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision (ICCV)* (2013).

[34] Fu, Y., Hospedales, T., Xiang, T., Fu, Z., and Gong, S. Transductive multi-view embedding for zero-shot recognition and annotation. In *IEEE European Conference on Computer Vision (ECCV)* (2014).

[35] Gaidon, A., Zen, G., and Rodriguez-Serrano, J. A. Self-learning camera: Autonomous adaptation of object detectors to unlabeled video streams. In *ArXiv* (2014).

[36] Gall, J., Yao, A., Razavi, N., Van Gool, L., and Lempitsky, V. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 33*, 11 (2011).

[37] Gavrila, D. M., and Davis, L. S. 3-d model-based tracking of humans in action: a multi-view approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (1996).

[38] Gong, B., Shi, Y., Sha, F., and Grauman, K. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[39] Gopalan, R., Li, R., and Chellappa, R. Domain adaptation for object recognition: An unsupervised approach. In *IEEE International Conference on Computer Vision (ICCV)* (2011).

[40] Gordon, N., Salmond, D., and Smith, A. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings, Radar and Signal Processing 140*, 2 (1993).

[41] Gower, J. C., and Dijksterhuis, G. B. *Procrustes problems*, vol. 30 of *Oxford Statistical Science Series*. Oxford University Press, Oxford, UK, 2004.

[42] Guan, P., Weiss, A., Balan, A., and Black, M. J. Estimating human shape and pose from a single image. In *Int. Conf. on Computer Vision, ICCV* (2009), pp. 1381–1388.

[43] Hähnel, D., Thrun, S., and Burgard, W. An extension of the ICP algorithm for modeling nonrigid objects with mobile robots. In *International Joint Conference on Artificial Intelligence* (Acapulco, Mexico, 2003).

[44] Hansard, M. E., Evangelidis, G., and Horaud, R. Cross-calibration of time-of-flight and colour cameras. *ArXiv* (2014).

[45] Hansard, M. E., Lee, S., Choi, O., and Horaud, R. *Time-of-Flight Cameras - Principles, Methods and Applications.* Springer Briefs in Computer Science. Springer, 2013.

[46] Harris, C., and Stephens, M. A combined corner and edge detector. In *Fourth Alvey Vision Conference* (1988).

[47] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004.

[48] Hastie, T., Tibshirani, R., and Friedman, J. H. *The Elements of Statistical Learning*. Springer, 2003.

[49] Henseler, H., Kuznetsova, A., Vogt, P., and Rosenhahn, B. Validation of the kinect device as a new portable imaging system for three-dimensional breast assessment. *Journal of Plastic, Reconstructive & Aesthetic Surgery* (2014).

[50] Herrera, D., Kannala, J., and Heikkila, J. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 34*, 10 (2012).

[51] Hoai, M., and Zisserman, A. Discriminative sub-categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[52] Hoffman, J., Darrell, T., and Saenko, K. Continuous manifold based adaptation for evolving visual domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[53] Hoffman, J., Kulis, B., Darrell, T., and Saenko, K. Discovering latent domains for multisource domain adaptation. In *IEEE European Conference on Computer Vision (ECCV)* (2012).

[54] Isaacs, J., and Foo, S. Hand pose estimation for american sign language recognition. In *Southeastern Symposium on System Theory (SSST)* (2004).

[55] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. *ArXiv* (2014).

[56] Kalogeiton, V., Ferrari, V., and Schmid, C. Analysing domain shift factors between videos and images for object detection. *ArXiv* (2015).

[57] Karpathy, A. Machine learning matlab toolbox. `https://github.com/karpathy/Random-Forest-Matlab`.

[58] Kennedy, J., and Eberhart, R. C. Particle swarm optimization. In *IEEE International Conference on Neural Networks* (1995), pp. 1942–1948.

[59] Keskin, C., Kıraç, F., Kara, Y. E., and Akarun, L. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *IEEE European Conference on Computer Vision (ECCV)* (2012).

[60] Keskin, C., Kıraç, F., Kara, Y. E., and Akarun, L. Real time hand pose estimation using depth sensors. In *EEE International Conference on Computer Vision Workshops (ICCVW)* (2011).

[61] Kim, T., Livescu, K., and Shakhnarovich, G. American sign language fingerspelling recognition with phonological feature-based tandem models. In *Spoken Language Technology Workshop (STL)* (2012).

[62] Kim, Y. M., Chan, D., Theobalt, C., and Thrun, S. Design and calibration of a multi-view tof sensor fusion system. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2008).

[63] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *SCIENCE 220*, 4598 (1983), 671–680.

[64] Krejov, P., and Bowden, R. Multi-touchless: Real-time fingertip detection and tracking using geodesic maxima. In *IEEE International Conference on Automatic Face and Gesture Recognition (FG)* (2013).

[65] Kulis, B., Saenko, K., and Darrell, T. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).

[66] Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NIPS)* (2010).

[67] Kuznetsova, A., Hwang, S. J., Rosenhahn, B., and Sigal, L. Expanding object detector's horizon: Incremental learning framework for object detection in videos. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[68] Kuznetsova, A., Hwang, S. J., Rosenhahn, B., and Sigal, L. Exploiting view-specific appearance similarities across classes for zero-shot pose prediction: A metric learning approach.

[69] Kuznetsova, A., Leal-Taixe, L., and Rosenhahn, B. Real-time sign language recognition using a consumer depth camera. *EEE International Conference on Computer Vision Workshops (ICCVW)* (2013).

[70] Kuznetsova, A., Pons-Moll, G., and Rosenhahn, B. Pca-enhanced stochastic optimization methods. *Annual Symposium of the German Association for Pattern Recognition (DAGM, now GCPR)* (2012).

[71] Kuznetsova, A., and Rosenhahn, B. Hand pose estimation from a single rgb-d image. *International Symposium on Visual Computing (ISVC)* (2013).

[72] Kuznetsova, A., and Rosenhahn, B. On calibration of a low-cost time-of-flight camera. *IEEE European Conference on Computer Vision Workshops (ECCVW)* (2014).

[73] Kuznetsova, A., Troje, N. F., and Rosenhahn, B. A statistical model for coupled human shape and motion synthesis. *International Conference on Computer Graphics Theory and Applications* (2013).

[74] Lachat, E., Macher, H., Mittet, M.-A., Landes, T., and Grussenmeyer, P. First experiences with kinect v2 sensor for close range 3d modelling. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)* (2015).

[75] Lange, R., Seitz, P., Biber, A., and Schwarte, R. Time-of-flight range imaging with a custom solid state image sensor. vol. 3823.

[76] Larsson, A., Kuznetsova, A., Caster, O., and Ekenberg, L. Implementing second-order decision analysis: Concepts, algorithms, and tool.

[77] Leal-Taixe, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., and Savarese, S. Learning an image-based motion context for multiple people tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[78] Lee, Y. J., and Grauman, K. Learning the easy things first: Self-paced visual category discovery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).

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

[80] Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics 2* (1944).

[81] Lewis, J. P., Cordner, M., and Fong, N. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), ACM special interest group on Computer GRAPHics and Interactive Techniques (SIGGRAPH).

[82] Lindner, M., and Kolb, A. Lateral and depth calibration of pmd-distance sensors. In *International Symposium on Visual Computing (ISVC)* (2006).

[83] Lindner, M., Schiller, I., Kolb, A., and Koch, R. Time-of-flight sensor calibration for accurate range sensing. *Computer Vision and Image Understanding (CVIU) 114*, 12 (2010).

[84] Liwicki, S., and Everingham, M. Automatic recognition of fingerspelled words in british sign language. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2009).

[85] Lucas, B. D., and Kanade, T. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence* (1981).

[86] Matthies, H., and Strang, G. The solution of nonlinear finite element equations. *International Journal for Numerical Methods in Engineering 14*, 11 (1979).

[87] Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2013).

[88] Mittal, A., Zisserman, A., and Torr, P. H. S. Hand detection using multiple proposals. In *British Machine Vision Conference (BMVC)* (2011).

[89] Morano, R., Ozturk, C., Conn, R., Dubin, S., Zietz, S., and Nissano, J. Structured light using pseudorandom codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 20*, 3 (1998).

[90] Murray, R. M., Sastry, S. S., and Zexiang, L. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1994.

[91] Nadaraya, E. A. On estimating regression. *Theory of Probability and its Applications 9* (1964).

[92] Nalpantidis, L., Sirakoulis, G. C., and Gasteratos, A. Review of Stereo Vision Algorithms: From Software to Hardware. *International Journal of Optomechatronics 2*, 4 (2008).

[93] Ohser, J., Ferrero, C., Wirjadi, O., Kuznetsovac, A., Düll, J., and Rack, A. Estimation of the probability of finite percolation in porous microstructures from tomographic images.

[94] Oikonomidis, I., Kyriazis, N., and Argyros, A. Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference (BMVC)* (2011).

[95] Pirsiavash, H., and Ramanan, D. Detecting activities of daily living in first-person camera views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[96] Plagemann, C., Ganapathi, V., Koller, D., and Thrun, S. Real-time identification and localization of body parts from depth images. In *IEEE International Conference on Robotics and Automation (ICRA)* (2010).

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

[98] Pons-Moll, G., Leal-Taixe, L., Truong, T., and Rosenhahn, B. Efficient and robust shape matching for model based human motion capture. In *Annual Symposium of the German Association for Pattern Recognition (DAGM, now GCPR)* (2011).

[99] Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. Learning object class detectors from weakly annotated video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2012).

[100] Pugeault, N., and Bowden, R. Spelling it out: Real-time asl fingerspelling recognition. In *EEE International Conference on Computer Vision Workshops (ICCVW)* (2011).

[101] Ren, Z., Yuan, J., and Zhang, Z. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *ACM Multimedia* (2011).

[102] Rokach, L. Ensemble-based classifiers. *Artificial Intelligence Review 33*, 1-2 (2010).

[103] Rusu, R. B. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.* PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[104] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. Adapting visual category models to new domains. In *IEEE European Conference on Computer Vision (ECCV)* (2010).

[105] Saxena, A., Chung, S. H., and Ng, A. Y. 3dd depth reconstruction from a single still image. *International Journal on Computer Vision (IJCV) 76*, 1 (2008).

[106] Schapire, R. E. The strength of weak learnability. *Machine Learning Journal 5*, 2 (1990).

[107] Sharma, P., Huang, C., and Nevatia, R. Unsupervised incremental learning for improved object detection in a video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[108] Sharma, P., and Nevatia, R. Efficient detector adaptation for object detection in video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[109] 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. Accurate, robust, and flexible real-time hand tracking. In *ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2015), ACM.

[110] Shi, T., and Horvath, S. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics 15*, 1 (2006), 118–138.

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

[112] Silver, D. L., Yang, Q., and Li, L. Lifelong machine learning systems: Beyond learning algorithms.

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

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

[115] Sridhar, S., Oulasvirta, A., and Theobalt, C. Fast tracking of hand and finger articulations using a single depth camera. Research Report MPI-I-2014-4-002, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, October 2014.

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

[117] Stenger, B. Template-based hand pose recognition using multiple cues. In *Asian Conference on Computer Vision (ACCV)* (2006).

[118] Stenger, B., Mendonça, P. R. S., and Cipolla, R. Model-based 3d tracking of an articulated hand. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2001).

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

[120] Tang, K., Ramanathan, V., Fei-Fei, L., and Koller, D. Shifting weights: Adapting object detectors from image to video. In *Advances in Neural Information Processing Systems (NIPS)* (2012).

[121] Thrun, S. A lifelong learning perspective for mobile robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1995.

[122] Tomasi, C., and Kanade, T. Detection and tracking of point features. Tech. rep., 1991.

[123] Tukey, J. W. *Exploratory Data Analysis*. Behavioral Science: Quantitative Methods. Addison-Wesley, Reading, Mass., 1977.

[124] Uijlings, J., van de Sande, K., Gevers, T., and Smeulders, A. Selective search for object recognition. *International Journal on Computer Vision (IJCV)* (2013).

[125] Van den Bergh, M., and Van Gool, L. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *IEEE Workshop on Applications of Computer Vision (WACV)* (2011).

[126] Vezhnevets, V., Sazonov, V., and Andreeva, A. A survey on pixel-based skin color detection techniques. In *GRAPHICON* (2003).

[127] Vondrak, M., Sigal, L., and Jenkins, O. Dynamical simulation priors for human motion tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 35*, 1 (2013).

[128] Wang, R. Y., and Popović, J. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics (TG) 28*, 3 (2009).

[129] Wang, X., Hua, G., and Han, T. Detection by detections: Non-parametric detector adaptation for a video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[130] Weinberger, K. Q., and Chapelle, O. Large margin taxonomy embedding for document categorization. In *Advances in Neural Information Processing Systems (NIPS)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. 2009.

[131] Weston, J., Bengio, S., and Usunier, N. Wsabie: Scaling up to large vocabulary image annotation. In *International Joint Conference on Artificial Intelligence* (2011).

[132] Wohlkinger, W., and Vincze, M. Ensemble of shape functions for 3d object classification. In *ROBIO* (2011).

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

[134] Zalevsky, Z., Shpunt, A., Maizels, A., and Garcia, J. Method and system for object reconstruction, 2007. WO Patent App. PCT/IL2006/000,335.

[135] Zhang, L., Curless, B., and Seitz, S. M. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2003).

[136] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 22*, 11 (2000).

Alina Kuznetsova

Contact information
E-mail: alinakuznetsova@gmail.com
Mobile phone: +4917681679277
Web: `www.tnt.uni-hannover.de/staff/kuznetso/`

Address
Calenberger Esplanade 7
Hannover, 30169, Germany

EDUCATION

Leibniz University Hannover, Hannover, Germany
Institute for Information Processing, *11/2011 - 11/2015 [expected]*
Ph.D. in Computer Science & Engineering,
Adviser: Prof. Dr. Bodo Rosenhahn

Lomonosov Moscow State University, Moscow, Russia
Faculty of Applied Mathematics and Cybernetics, *07/2005-07/2010*
Degree in mathematics and system programming (with honors),
Note: 4.8/5.0

KTH Royal Institute of Technology, Stockholm, Sweden
Exchange student *08/2009-01/2010*

EXPERIENCE

Research Assistant, Leibniz University Hannover, Hannover, Germany
Teaching (*Pattern Recognition* course , *Image processing* lab) *11/2011-current*
Supervision and co-supervision of several master and bachelor students.
Projects:
- Full DoF hand pose estimation and recognition using consumer depth cameras [7], [6].
- Modeling dependencies between human appearance and motion, using gait motion [8].
- Depth/Time-of-flight camera calibration [4].

Research Intern, Microsoft Research Cambridge, Cambridge, UK
Improving hand tracking using consumer depth cameras. *06/2015 - 09/2015*

Research Intern, Disney Research Pittsburgh, Pittsburgh, USA
Development of an adaptive method for object detection in videos. [2] *08/2014 - 12/2014*
Adviser: Dr. Leonid Sigal

Student developer, Google summer of code, *05/2012-09/2012*
OpenCV project: `https://github.com/kapibara/GSOC2012`
Mentor: Dr. Caroline Pantofaru

Research Assistant, Fraunhofer ITWM, Kaiserslautern, Germany
Image processing algorithms development *10/2010 — 10/2011*
for MAVI software (C/C++, CMake)

Research Contractor, Uppsala Monitoring Center, Uppsala, Sweden
Projects: *02/2010 — 05/2010*
- A belief distribution simulation algorithm for DecideIT software (Java) [2]
- Analysis and clustering of multidimensional medical data using EM algorithm.

Advisers: Dr. Ola Caster, Prof. Dr. Nicolas Noren

System Analyst, Deloitte Russia, Moscow, Russia
System integration for a retail company (SQL) *04/2010 – 08/2010*

Java Developer, Hewlett-Packard Russia, Moscow, Russia
Development of distributed automated documentation generating system *07/2008 – 04/2010*
for HP Internet Usage Manager (Java, XML/XSLT)

PUBLICATIONS

[1] A. Kuznetsova, S. Hwang, B. Rosenhahn, L. Sigal, *Exploiting View-Specific Appearance Similarities Across Classes for Zero-shot Pose Prediction: A Metric Learning Approach.* (AAAI, 2016)

[2] A. Kuznetsova, S. Hwang, B. Rosenhahn, L. Sigal, *Expanding Object Detector's Horizon: Incremental Learning Framework for Object Detection in Videos.* (CVPR, 2015)

[3] A. Larsson, A. Kuznetsova, O. Caster, L. Ekenberg, *Implementing Second-Order Decision Analysis: Concepts, Algorithms, and Tool.* (Advances in Decision Sciences, 2014)

[4] A. Kuznetsova, B. Rosenhahn, *On calibration of a low-cost time-of-flight camera.* (ECCV Workshops, 2014)

[5] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, S. Savarese, B. Rosenhahn, *Learning an image-based motion context for multiple people tracking*, (CVPR 2014)

[6] A. Kuznetsova, L. Leal-Taixé, B. Rosenhahn, *Real-time sign language recognition using a consumer depth camera.* (ICCV Workshops, 2013)

[7] A. Kuznetsova, B. Rosenhahn, *Hand pose estimation from a single RGB-D image.* (ISVC, 2013)

[8] A. Kuznetsova, N. Troje, B. Rosenhahn, *A statistical model for coupled human shape and motion synthesis.* (VISIGRAPP, 2013)

[9] A. Kuznetsova, G. Pons-Moll, B. Rosenhahn, *PCA-enhanced stochastic optimization methods.* (DAGM, 2012)

[10] J. Ohser, C. Ferrero, O. Wirjadi, A. Kuznetsova, J. Duell, A. Rack, *Estimation of the probability of finite percolation in porous microstructures from tomographic images.* (IJMR, 2012)

HONORS AND AWARDS

Mathematics as a Key Technology grant, TU Kaiserslautern *2010*

Third-level diploma winner in mathematics and physics, *Lomonosov* contest *2005*