**SAARLAND UNIVERSITY**
**Faculty of Natural Sciences and Technology I**
**Department of Computer Science**

Master's Thesis

# Real-Time Hand-Object Tracking Using a Single Depth Camera

submitted by

Franziska Müller

submitted on

April 1, 2016

Reviewers:

Prof. Dr. Christian Theobalt

Prof. Dr. Jürgen Steimle

## Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

## Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

## Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

## Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, ______________________ ______________________

(Datum/Date) (Unterschrift/Signature)

# Abstract

Reconstructing the pose of the human hand has been an actively researched topic in computer vision over the past decades. New devices that require more flexible and natural input modalities than the ones used in the typical desktop setting have been developed in the last years. Since humans usually use their hands for everyday interactions, tracking of hand gestures seems to be a suitable candidate. But most of the time also objects in the environment are manipulated during interaction. Especially when considering virtual or augmented reality scenarios, joint reconstruction of the hand and the object is crucial.

Whereas there exist approaches to robustly track the motion of a single hand using a single depth camera in real time, no such method is known for joint hand and object tracking. Severe occlusions caused by the object and the higher dimensionality of the problem are only few of the new challenges. In this thesis, an approach for reconstructing both hand and object motion with an RGB-D camera in real time is developed. It combines a generative pose optimization framework with additional information obtained from a discriminative machine learning algorithm in an uncertainty-aware way. Prior knowledge that is specific to the hand and object setting, like contact point constraints and heuristics for regions occluded by the object, is used to deal with inherent ambiguities of this ill-posed problem. Physical plausibility of the tracked poses is further ensured by avoiding hand-object interpenetrations. The ability to track hand and object motion simultaneously utilizing a lightweight single camera setup opens up many possible applications, for example in augmented and virtual reality.

# Acknowledgments

I want to thank Professor Christian Theobalt for offering this thesis and Professor Jürgen Steimle for being my second reviewer.

It was a pleasure to work with all the people involved in the project: Michi, Dan, Srinath, Christian and Antti. Thanks for all your hard work for the paper corresponding to this thesis. I'm looking forward to work with you again in the future.

Also the whole Graphics, Vision and Video group did a great job in providing an inspiring working atmosphere and integrating me like a proper group member. I'm happy and proud to start as a PhD student in this wonderful group.

I would further like to thank my proof readers Fabian, Wiam, Chris and Srinath for their useful feedback and comments.

Furthermore, I thank my family and friends for their support and their ability to bear a stressed Franzi.

# Contents

# List of Figures

# List of Symbols

| | |
|---|---|
| $\mathcal{F}_t = (\mathcal{C}_t, \mathcal{D}_t)$ | input at time $t$, color and depth respectively image |
| $\Theta/\Theta_h/\Theta_o$ | (Combined/hand only/object only) pose parameters |
| $\mathcal{I}/\mathcal{I}_h/\mathcal{I}_o$ | (Combined/hand only/object only) input Gaussian Mixture |
| $N_{\mathcal{I}}$ | Number of input Gaussians |
| $\mathcal{M}/\mathcal{M}_h/\mathcal{M}_o$ | (Combined/hand only/object only) model Gaussian Mixture |
| $N_{\mathcal{M}}$ | Number of model Gaussians |
| $E_{depth}$ | Energy without part label information |
| $E_{label}$ | Energy including part label information |
| $E_{comp}$ | Energy for comparing the results from depth and label proposal |
| $E_a$ | Spatial alignment energy term |
| $E_l$ | Label alignment energy term |
| $E_p$ | Anatomical plausibility term |
| $E_t$ | Temporal smoothness term |
| $E_c$ | Contact points term |
| $E_o$ | Object occlusion term |
| $E_i$ | Interpenetration avoidance term |

# List of Abbreviations

| | |
|---|---|
| AR | augmented reality |
| DOF | degree of freedom |
| GMM | Gaussian mixture model |
| ICP | iterative closest point |
| IK | inverse kinematics |
| PSO | particle swarm optimization |
| PCA | principal component analysis |
| RDF | random decision forest |
| VR | virtual reality |

# Chapter 1

# Introduction

Reconstructing the complex articulated motion of the hand has been an active topic in research over the past decades. Whereas people used to interact with a computer via a keyboard and a mouse for a long time, these input modalities are not suitable for more mobile scenarios such as smart phones or smart watches. Especially new upcoming devices, like virtual reality (VR) or augmented reality (AR) glasses where the user should fully immerse in a virtual world, need natural and flexible input possibilities. Because humans usually use their hands to interact with the environment, employing hand tracking to make interactions with a virtual environment possible is a logical choice.

But even the task of tracking single hand motions is challenging. The expressiveness of the human hand leads to a high-dimensional optimization problem that is in general non-convex. Consequently, iterative solvers that are not guaranteed to always find the global optimum need to be used. Performing complex poses results in self-occlusions and ambiguities that are a lot more severe than in full body pose estimation. Additional knowledge that is useful in full-body tracking can often not be transferred to the hand. For example the color information that could be exploited based on clothing is less helpful since the hand color is uniform. Furthermore, assumptions about the orientation, like "a person is usually standing upright", are invalid for the hand. Researchers have developed a lot of different methods to tackle hand tracking (see Section 2.1). Some used a multi-camera setup to deal with occlusions and ambiguities while losing flexibility and processing speed which are both essential for interactive techniques [4]. Most of the recent approaches tend to use a single RGB-D camera to enable adaptability for mobile setups and real-time frame rates ($\geq$ 30 FPS) while exploiting the extra depth information [5, 6]. The existing approaches can be split into data-driven discriminative learning [7, 8], generative optimization of an objective measuring similarity [9, 10] and combinations of these two types [1, 11].

However, people are not only naturally interacting using hand gestures but also by manipulating objects in their environment. Especially when considering AR or VR applications, reconstructing both, the hand motion and the manipulations of an object, is necessary. Figure 1.1 shows two applications where joint tracking of hand and object is needed to enable natural interaction of a user with the augmented or virtual environment.

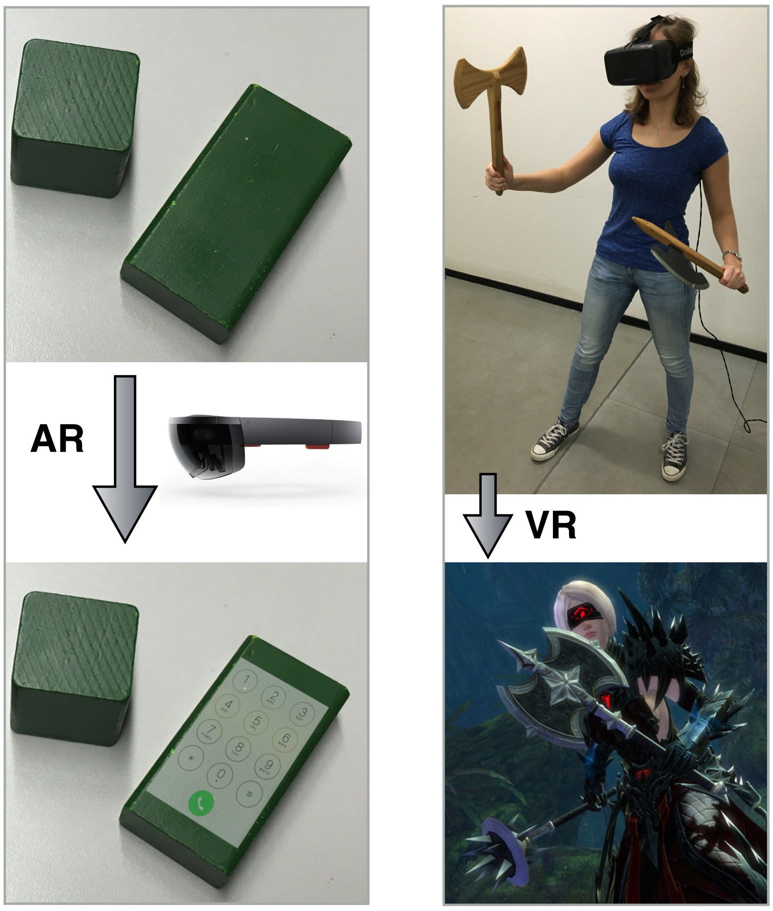


FIGURE 1.1: **Left:** When wearing AR glasses, various interfaces can be virtually added to simple objects. This example shows a cuboid that is overlayed with a keypad. Simultaneous hand and object tracking would enable to recognize which key was touched and makes the interface look plausible in terms of position and rotation.[1] **Right:** The possibility to track hands and objects in interaction enables control over a virtual character, for example in a video game. Hence, the VR experience becomes even more immersive.

[1]Image "HoloLens" (https://www.flickr.com/photos/microsoftsweden/16153490837), published by Microsoft Sweden on Flickr, licensed under CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/)

The scenario of joint hand-object reconstruction leads to new specific challenges that come in addition to the ones already present in single hand tracking. Firstly, the algorithm needs to distinguish between the hand and the object in the input. This is a demanding task in general scenes without any assumptions like known and unique object color and shape. Apart from the various self-occlusions in single hand pose reconstruction, the close interaction with an object introduces heavy occlusions. Especially when using only one or few cameras, this leads to large areas in the 3D scene for which no input data is available. Furthermore, the estimation of the pose of the object adds additional dimensions to the already high-dimensional pose space. While six parameters are needed to describe the rigid transform of the object, significantly more parameters are necessary to capture non-rigid deformations. In generative frameworks, the increase in dimensionality yields a harder optimization problem whereas in discriminative techniques it complicates training data acquisition and learning.

In comparison to the field of single hand tracking, there has been less research about hand and object tracking (see Section 2.2). Some approaches are only able to reconstruct the hand motion but under strong occlusions by the object [12, 13]. Others employed multi-camera systems to overcome occlusions [14, 15]. But like in single hand tracking, recent work is trending towards the use of a single RGB-D camera [2, 16, 17]. This yields a more flexible and mobile setup. Nevertheless, none of these methods is able to reconstruct the hand pose and the rigid transform of an object that is interacted with in real time.

This work makes the following contributions to the research on joint hand and object tracking:

- A generative pose optimization framework is combined in an uncertainty-aware way with discriminative classification information.
- The tracking is physically plausible and remains stable under strong occlusions by exploiting contact point constraints, heuristics for occluded areas and interpenetration restrictions.
- Due to a smooth mathematical representation, the algorithm runs in real time while mostly relying on simple CPU parallelization.

# Chapter 2

# Related Work

The field of recognition and tracking of objects and hands has been widely researched. There has been work on detecting and tracking single or multiple objects, including their rigid transform and sometimes also their deformations [18–22]. Furthermore, pose estimation of a single hand as well as tracking of more or less complicated interactions between two hands or hands and objects have emerged as interesting topics.

## 2.1 Tracking of a Single Hand

Many researchers tackled the problem of tracking the articulations of a single hand. Most of the approaches can be divided into discriminative and generative methods. Whereas discriminative algorithms are data-driven and based on machine learning techniques, generative tracking algorithms use a 3D model and phrase pose estimation as an optimization problem over the pose parameters. Both types of approaches have individual advantages and disadvantages. Purely discriminative methods are fast in actual tracking but are comparably time-consuming in the training phase and the data generation in the first place. Generalization to unseen data is a highly active research topic but yet unsolved. Discriminative methods often do per-frame estimations, so tracking failures do not propagate but this also leads to temporal jitter in the poses. Since hand pose estimation is a complex and high-dimensional task, the corresponding problem that generative methods try to solve is usually non-convex and hard to optimize. But careful engineering of the optimization problem ensures — in theory — that all poses can be handled.

In practice, a good starting point for the optimizer is needed in each frame to achieve fast convergence to the global optimum. For this purpose, tracked poses from previous frames are commonly used. That also induces temporal smoothness but makes tracking failures more severe. In recent years, combining both aforementioned types of methods has become more popular as these combinations might be able to overcome inherent weaknesses of the types when used in isolation.

**Discriminative.** Some discriminative approaches use a database of hand poses which reduces the pose estimation task to an indexing problem [7, 23]. Both, Wang *et al.* and Athitsos *et al.*, rely on RGB data but the first approach runs in real time whereas the second needs 15 seconds per frame to process. The first method requires the user to wear a glove with a colored pattern and defines a distance metric on images of the glove to run a nearest-neighbor search on the database. The second method does not require to wear anything like markers or gloves and uses chamfer distance on edge images and line matchings for indexing.

Without any markers, using RGB data for recognizing the pose of the hand is not beneficial due to the uniform color of the hand. Because depth sensors are becoming cheaper and more ubiquitous, people started to use this input cue for tracking hand motions. Keskin *et al.* [8] trained random decision forests on depth input to obtain likelihoods for joint locations. Afterwards, the maximum mode is estimated using mean shift while also handling occluded joints. Tompson *et al.* [5] applied convolutional networks to depth data to obtain heatmaps for joint locations. These are fed into an inverse kinematics (IK) approach to get the pose of the skeleton. Latent regression forests were used by Tang *et al.* [24] whereas Sun *et al.* [25] employed a cascaded ensemble of regressors to obtain the hand pose parameters directly.

**Generative.** The first model-based generative tracker to run at 10 Hz was proposed in 1996 by Heap *et al.* [26]. They used a deformable mesh model of the hand and a single video camera to track simple motions but do not yet handle occlusions properly. Also in the field of generative hand tracking, research projects started to exploit the depth channel when the sensors became more common *e. g.* through the Kinect[1]. Bray *et al.* [27] combines particle filtering with an inbetween gradient descent step what

[1]Microsoft Kinect, https://dev.windows.com/en-us/kinect

takes 22 seconds to process a frame. Using a GPU implementation of particle swarm optimization (PSO), Oikonomidis *et al.* [9] achieve 15 Hz. They follow an analysis-by-synthesis approach, generating a depth map for a pose hypothesis and comparing it to the given input depth. There also has been work that does hand tracking in real time. Melax *et al.* [28] employ an approach that is based on physical simulation and includes collision handling, contact constraints and joint mechanics. A low-dimensional subspace of plausible poses, based on PCA, was used as prior and soft constraint in the work of Tagliasacchi and colleagues [10] where they fitted a 3D hand model using registration. Zollhöfer *et al.* [18] demonstrated reconstruction of hand motion as one application of their non-rigid reconstruction algorithm for objects of any shape.

**Combined.** Sridhar *et al.* [4] use a calibrated camera setup consisting of multiple RGB cameras and one depth sensor. They apply discriminative fingertip detection on the depth map to obtain a complete or partial hand pose that is later on fused with the result of a generative optimizer using a 2D Gaussian mixture representation. Because multi-camera setups are expensive, immobile and often slow due to the pure amount of data to be processed, methods using a more lightweight and mobile setup, like a single depth sensor, were researched.

Some approaches use a discriminative algorithm to get an initial estimate of the hand pose to start the generative optimizer from [11, 29]. Qian *et al.* [29] run a combination of gradient-based iterative closest point (ICP) and stochastic PSO from a pose obtained from part detections. A hierarchical distribution of hand poses is discriminatively generated by Sharp *et al.* [11] to launch a PSO-like optimization scheme. They optimize an analysis-by-synthesis error measure that compares the actual depth map to the depth map generated by the pose hypothesis. A similar objective is used by Tang *et al.* [6] in their so called hierarchical sampling optimization. They optimize the pose parameters following the structure of the kinematic skeleton. For each parameter they first sample from a discriminatively trained regression forest and then the best parameter is propagated along the kinematic chain so that in the end a complete pose is acquired. Sridhar *et al.* [1] work with hand part labels that come from a random decision forest trained on depth data. Instead of using the pose induced by the positions of the parts, they fuse the part label information directly into the objective of the generative optimization.

## 2.2 Tracking of Interacting Hands

In comparison to the tracking of a single hand, it is an even more ambitious task to track two hands or hands and objects in interaction. In addition to the higher dimensionality of the problem, more occlusions and ambiguities are especially challenging. Not explicitely tackling these challenges often results in tracking failures. Therefore, employing a naïve combination of single-hand and object-only trackers does not work (see also Section 6.3 for experiments).

**No Explicit Object Motion.** Approaches have been proposed that track a hand motion in such a scenario, but not explicitly the movement of the object. Hamer *et al.* [12] use a single RGB-D camera and employ a local tracker for every segment of the articulated skeleton what helps to handle strong occlusions. They optimize by performing belief propagation on a markov random field constructed based on the kinematic structure. In contrast to [12] — where 6.2 seconds are needed to process one frame — the method of Romero *et al.* [13] runs in real time. They work with a database of synthetic images of hand poses, occluded and unoccluded, and apply nearest neighbor search. This search also takes the temporal pose history into account helping to avoid temporal inconsistency that is common for pure discriminative approaches.

**Hands and Objects.** In contrast to single hand tracking, where the majority of recent works use a single camera to make the methods mobile, interactive and flexible, setups consisting of multiple RGB cameras are frequently employed for the hands and objects case [14, 15, 30]. Oikonomidis *et al.* [14] jointly track the poses of the hand and an object while modeling occlusions and physical interpenetration constraints. They optimize edge map and skin color map similarity of the input and the rendering of the pose hypothesis by PSO. A discriminative approach for detecting salient points, here finger nails, is combined with a generative analysis-by-synthesis method by Ballan *et al.* [15]. They take optical flow and edge maps into account and simultaneously solve for an association of the detected salient points because it is in general ambiguous which finger nail belongs to which finger. Wang *et al.* [30] also apply analysis-by-synthesis, based on edges, silhouettes and color. They furthermore propose contact-based sampling for the optimization: New poses are sampled considering also the contact points detected in the previous pose. Using this technique, it is less likely to sample implausible poses, like floating objects.

With the use of depth sensors becoming more common, there has been recent work on tracking interacting hands that utilizes a single RGB-D camera [16, 17]. Tzionas *et al.* [16] track the motion of two hands interacting as shown in Figure 2.1. The approach that is based on the work by Ballan *et al.* [15] integrates discriminatively learned salient points with a generative model that considers occlusions and collisions. In follow-up work, Tzionas *et al.* [2] extend their framework with physics simulation to jointly track the motion of hands and an object in a physically plausible way. This computationally expensive algorithm runs at offline frame rates. The method of Kyriazis *et al.* [17] is able to track complex scenes involving two hands and multiple objects (see Figure 2.1) but far from real time. They use an ensemble of collaborative trackers (ECT) in contrast to a single joint tracker (JT) or a set of independent trackers (SIT). The JT tries to optimize all parameters at the same time. It can handle interactions but is slow due to the high dimensionality of the problem. Using an SIT means that an independent tracker is used for each object to be tracked. There is no shared information between the trackers what makes interactions and especially collisions hard to handle. In the proposed ECT, information is shared regularly between the trackers. Apart from this information the trackers are independent, so that the overall pose estimation for all objects is comparably fast.

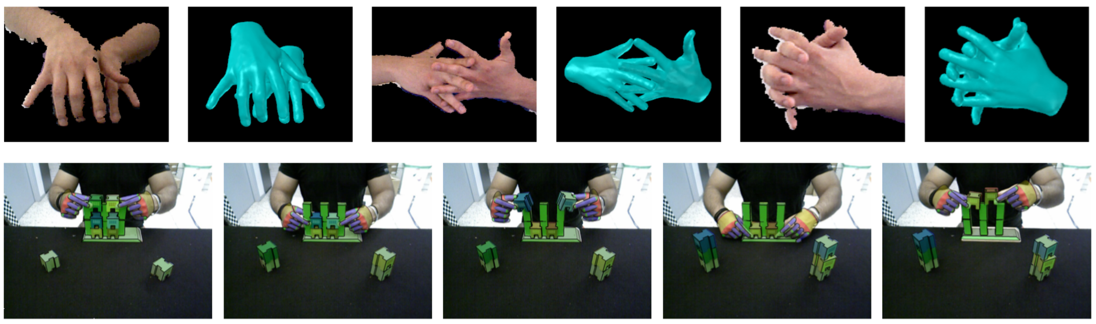


FIGURE 2.1: **Top:** An example of the poses that are estimated in [16]. **Bottom:** In this scene, two hands and several objects (together 159 DOFs) are tracked — at a speed of 0.48 FPS [17].

**In-hand Scanning.** A newer topic in the hand and object tracking research field is in-hand scanning. The goal is to reconstruct the geometry of an object that is rigidly moved by a hand in front of a camera, usually an RGB-D sensor [3, 31]. Paneteleris *et al.* [31] start the scanning process with a known hand model and build an object model progressively. They use the estimated fingertips that were tracked with PSO to segment the manipulated object. This segmented region is then passed on to the object tracker that applies ICP with the current object model. After tracking, the newly discovered object parts are fused into the object model represented as truncated signed distance function (TSDF). The color channel is only used for obtaining the texture of the object. This allows for reconstruction of skin-colored objects. The whole method runs at 10 FPS when only a single hand interacts with the object. In case of two hands, the frame rate drops to 4 FPS. Whereas [31] only use the hand pose estimation to segment the object, Tzionas and Gall [3] integrate this information also in the object tracking process. They employ tracked contact correspondences for aligning the new object point cloud to the currently built model. Having these additional features for alignment enables the method to also handle symmetric and featureless objects.

This work aims at tracking the motion of the hand and a rigid object in real time, under strong occlusions and in complex poses. As presented above, none of the related research projects achieves all of these goals. Also commercial tracking systems, like the LeapMotion [32] and NimbleVR [33], easily fail to reconstruct the motion of a hand interacting with another hand or an object.

In this thesis, prior knowledge specific to the hand and object scenario is exploited to guide the pose estimation. This leads to physically plausible poses and tracking that is stable under occlusions. Discriminative object and hand part information is fused with a generative model-based multi-proposal optimization framework while taking uncertainty in the prediction into account. Thereby, the analytic nature of the objective is preserved enabling the use of fast gradient-based optimizers. This makes tracking at real-time frame rates possible.

# Chapter 3

# Overview

The goal of this work is to enable simultaneous tracking of a hand and an object in interaction using a single close-range RGB-D camera. Thereby, the fully articulated pose of the hand as well as the rigid transform of the object is reconstructed. In contrast to related work — that has achieved this goal only running far from real time (see Chapter 2) — this approach aims at a real-time performance. Figure 3.1 shows one of the possible setups. Note that the camera location can be chosen arbitrarily since there are no assumptions made.

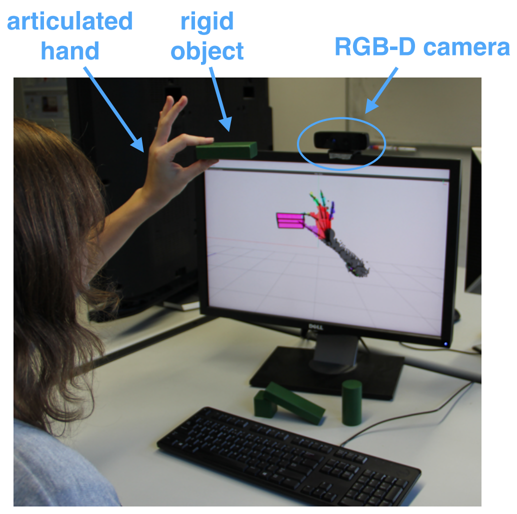


FIGURE 3.1: One possible camera setup for the approach presented in this thesis. The 3D reconstruction is shown live on the screen.

The algorithm estimates the pose of the hand and the object on a per-frame basis while using a prior to softly enforce temporal consistency. A graphical overview of the steps performed for each input frame is provided in Figure 3.2.

In the first step of the method, the input is classified on pixel-level into individual hand parts and object pixels. This yields correspondences between the shape prior used for tracking and the observation that are used later on. The second step is clustering of the input. Here the input is transformed into a mathematical density representation, a 3-dimensional Gaussian mixture model (GMM) (see Chapter 4).

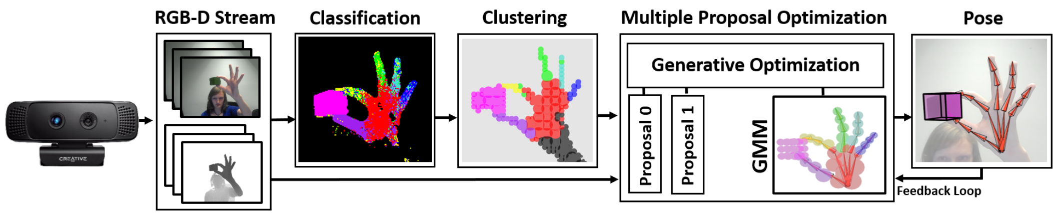


FIGURE 3.2: Firstly, the input from an RGB-D camera is preprocessed, *i. e.* classified in hand parts and the object and clustered. Afterwards, a model-based generative optimization using two hypotheses is performed to find the pose of the hand and the object in the current frame.

The 3D GMM representation is also used for the shape prior in the model-based pose optimization step. Similar GMM formulations were successfully employed for tracking tasks before, both in full body tracking [34] as well as hand-only tracking [4].

The generative pose optimization is performed by minimization of a new energy that also incorporates terms specifically tailored to the hand-object scenario such as contact points and occlusion contraints. Additional robustness and recovery from failures is achieved by integrating the part label information obtained before while also accounting for the uncertainty therein. Due to the analytical character of the GMM representation, the energy is analytically differentiable. Hence a fast gradient-based multi-proposal optimizer can be applied, making tracking in real time possible (see Chapter 5).

The algorithm is evaluated quantitatively and qualitatively on a new real hand-object dataset and two publicly available datasets. Furthermore, experiments showing the significance of single parts of the formulation are performed (Chapter 6). The thesis ends with a conclusion and discussion of possible future work (Chapter 7).

# Chapter 4

# Preprocessing and Input Transformation

The input to the algorithm is provided by a single RGB-D camera. The color and depth image of a time step $t$ are further denoted as $\mathcal{C}_t$ and $\mathcal{D}_t$, respectively. To make the input more easily manageable during pose optimization and more compact, a mathematical representation is usually used. Based on related work from Sridhar *et al.* [1], this work uses a density representation with Gaussian functions that also includes correspondence information for the object and particular parts of the hand. This part information is obtained from a machine learning classification algorithm. The remainder of this chapter explains how the classification is performed and how the input frame $\mathcal{F}_t = (\mathcal{C}_t, \mathcal{D}_t)$ is transformed to a Gaussian mixture model (GMM).

## 4.1 Hand and Object Classification

The goal of the classification in this work is to obtain a per pixel label that indicates that this pixel belongs either to the object or to one of six hand parts. The possible classes with corresponding colors for the images shown here are:

- palm (red),
- thumb (purple),
- index finger (yellow),
- middle finger (green),
- ring finger (cyan),
- little finger (dark blue)
- and object (pink).

The whole classification algorithm consists of two preprocessing steps, namely viewpoint selection and object segmentation, and a two-layered random decision forest (RDF) as main classifier (see Figure 4.1).

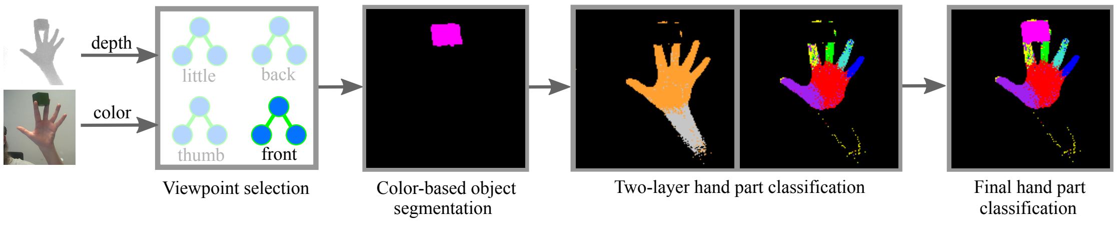

depth
color
little
back
thumb
front
Viewpoint selection
Color-based object segmentation
Two-layer hand part classification
Final hand part classification

FIGURE 4.1: First, a viewpoint is selected based on the pose tracked in the previous frame. Then, the color frame $\mathcal{C}_t$ is used to segment out pixels belonging to the object. After removing the object pixels from the depth image $\mathcal{D}_t$, it is passed to an RDF that classifies in the first step hand vs. arm pixels and then different hand parts. The final result combines information about hand parts and the object.

### 4.1.1 Viewpoint Selection

The viewpoint selection determines from which of four sides the hand is seen in the current frame $\mathcal{F}_t$. The four possible viewpoints are *front*, *back*, *thumb side* and *little finger side* and are distributed equidistant around the hand.

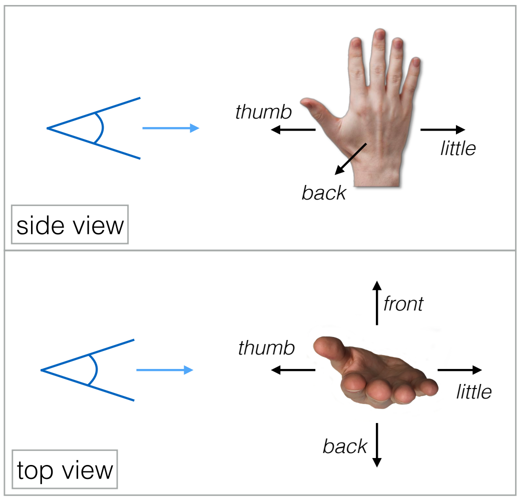


FIGURE 4.2: An example for the viewpoint selection based on the pose tracked in the previous frames and the viewpoint vectors (black). As the *thumb* vector is pointing in the opposite direction of the camera viewing direction (blue), it minimizes the cosine objective.

Note that additional viewpoints, *e. g. top* or *bottom*, could be integrated but are omitted here because they occur rarely in the tested camera setups. When the camera is placed for example in a smart watch, a *bottom* viewpoint might become necessary. For the viewpoint selection step, the tracked hand pose from time step $t-1$ is used instead of the input from the camera. Each possible viewpoint is represented by a vector in the local coordinate system of the hand as shown in Figure 4.2. Together with the global pose of the hand from time $t-1$, these vectors can be transformed to the global coordinate system.

Since also the normalized camera viewing direction $\hat{v}_c$ is known in this coordinate system, the best matching viewpoint can be determined as

$$\underset{i \in \{front,\, back,\, thumb,\, little\}}{\arg\min} \quad \cos(\angle(\hat{v}_c, \hat{v}_i)) \quad ,$$

where $\hat{v}_i$ is the normalized vector in global coordinates that represents viewpoint $i$. This viewpoint selection scheme is based on the assumptions that at least the global position and rotation of the hand have been tracked correctly in the previous frame and that the motion between two frames is small.

### 4.1.2 Object Segmentation

The goal of this step is to decide for each pixel if it is part of the object or not. Because the object color is assumed to be known and sufficiently distinct in the image, the color input $\mathcal{C}_t$ is used. For each pixel $p$, it is checked if the color value lies in the object color range and the label is assigned accordingly:

$$\text{if } \mathcal{C}_t(p) \in \delta_{HSV} \text{ then } \textit{object} \text{ else } \textit{non-object} \ ,$$

where $\delta_{HSV}$ is the predefined range of the object color in HSV color space. The HSV color space is preferable because hue and brightness are better seperable than *e. g.* in the RGB space. This makes the thresholding in situations with varying lighting easier.

### 4.1.3 Two-Layered Hand Parts Classification

All pixels that are in the foreground and were not classified as object in the object segmentation step before are further processed. They are passed on to one of four random decision forests, based on the viewpoint that was selected for the current time step $t$.

One such forest operates only on the depth image $\mathcal{D}_t$ and consists of two layers as illustrated in Figure 4.3. The first one performs per pixel classification into hand and arm. The decision is obtained by majority vote over the trees. All hand pixels are then classified as individual hand parts, *i. e.* palm, thumb, index finger, middle finger, ring finger or little finger, by the second layer of the forest.

Each forest is trained on 38 000 examples. The accuracies for the random decision forests used in this thesis on a disjoint test set are: 60% for *front*, 65% for *back*, 61% for *little* and 54% for *thumb*.

Note that the classification result also includes uncertainty. Each leaf node of a tree does not only represent a single class but rather a class distribution where the class with highest confidence determines the final label. These distributions are (amongst others) learned when training the RDF. Summing up the histogram of the reached leaf node over all trees and computing the average yields the final histogram. An example of such a class histogram is shown in Figure 4.3. The histograms are normalized so that they sum up to 1.

The result of the whole per-pixel classification algorithm is an image $\mathcal{H}_t$ with the same size as $\mathcal{D}_t$ that stores a class histogram for each pixel. Pixels that were assigned the object label have a histogram with confidence 1 in the object class. The bottom left part of Figure 4.4 shows an image where the best class per pixel from $\mathcal{H}_t$ is color-coded.

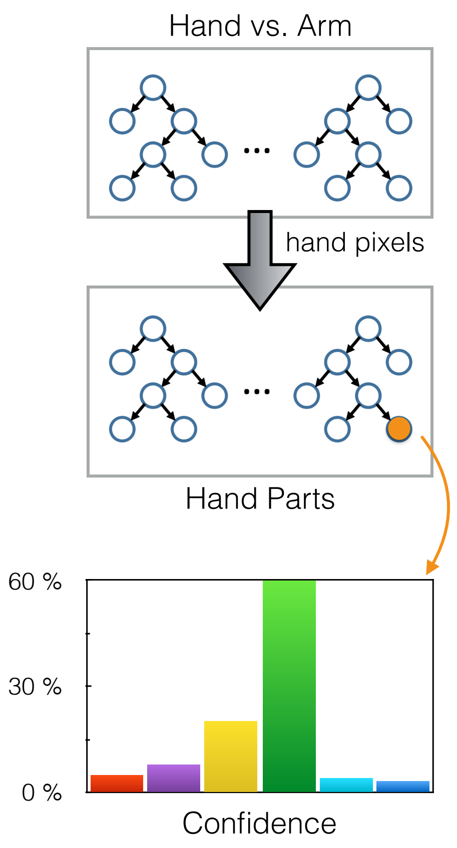


FIGURE 4.3: From top to bottom: the depth pixels from $\mathcal{D}_t$ are processed by the hand-arm forest. All pixels that are classified as belonging to the hand are passed on further to the hand part forest. Each leaf node in a tree stores a class histogram that represents the confidence for each class. In this example, the confidence is high that a pixel that reaches this leaf corresponds to the middle finger.

## 4.2 Building the Input GMM

As mentioned before, a Gaussian mixture model is used as input representation for the pose estimation algorithm. A general formulation for such a GMM is given by a sum of Gaussian functions

$$\sum_{i=1}^{N} G_i(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i) \quad ,$$

where in this work the isotropic Gaussians are specified by mean $\boldsymbol{\mu}$ and standard deviation $\sigma$.

As a first step when building the input GMM, the depth input $\mathcal{D}_t$ is clustered using bottom-up quadtree clustering.

Starting from single pixels, four neighboring quad regions of the same size are clustered to a bigger quad region. The clustering for a quad region is stopped either if it reaches the maximum size of 8 by 8 pixels or if the depth difference in the quad would become larger than the depth difference threshold $\epsilon_d$ in the next step. Choosing $\epsilon_d = 30\,\text{mm}$ yields a good trade-off between processing speed and geometric detail. Each quad $q_{\mathcal{D}}$ in the final quadtree that belongs to the foreground is then transformed into a 3D Gaussian.

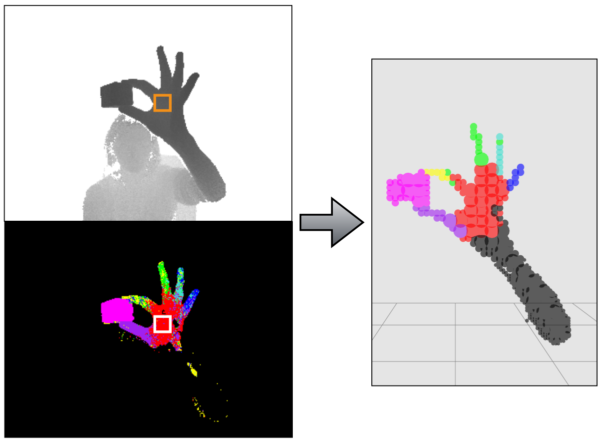


FIGURE 4.4: **Top left:** A quad (orange) resulting from depth-based quadtree clustering. All pixels contained in the quad and therefore also the corresponding 3D points have a similar depth from the camera (up to $\epsilon_d$). **Bottom left:** A visualization of the highest confidence classes for the above input. The matching quad (white) can be found at the same image location. **Right:** Using the camera intrinsics, the centers of the quads can be converted to 3D points in the coordinate system of the camera. Around these points, a 3-dimensional GMM is built.

The standard deviation $\sigma$ is set to half the backprojected side length of the quad. The mean $\boldsymbol{\mu} \in \mathbb{R}^3$ is obtained by backprojecting the center of the quad — from which the 2D image coordinates and the corresponding depth is known — using the camera intrinsics. Because only the front-facing surface is observed in the depth image $\mathcal{D}_t$, the mean after backprojection is positioned on the surface in 3-dimensional space. But since in the GMM the surface should coincide with the Gaussian's isosurface at one standard deviation from the mean, the mean $\boldsymbol{\mu}$ is further shifted by $\sigma$ away from the camera.

Furthermore, the Gaussian includes semantic part information taken from the histogram image $\mathcal{H}_t$. As shown in Figure 4.4 (left), a quad $q_{\mathcal{H}}$ in $\mathcal{H}_t$ (here marked white) that corresponds to the quad $q_{\mathcal{D}}$ in $\mathcal{D}_t$ (marked orange) can be found easily considering that both images share the same intrinsics and extrinsics. For fusing the part information over the whole quad, all the per-pixel histograms are summed up by summing the confidence values for each class separately. To obtain a normalized histogram — where the class confidences sum up to 1 again — the summed histogram is divided by $size(q_{\mathcal{H}})$.

The right half of Figure 4.4 illustrates the final result of the transformation to the input GMM. So in contrast to previous work by Sridhar *et al.* [1] that used so called 2.5D Gaussian mixtures, the input representation here is a proper 3D Gaussian mixture model

$$\mathcal{I}(\mathbf{x}) = \sum_{i=1}^{N_{\mathcal{I}}} G_i(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i) \quad , \tag{4.1}$$

where $N_{\mathcal{I}}$ is the number of input Gaussians and each $G_i$ is a 3-dimensional Gaussian with mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$ and standard deviation $\sigma_i \in \mathbb{R}$. The Gaussian mixture for the whole input can be divided into a hand and an object part based on the best class for each Gaussian

$$\mathcal{I}(\mathbf{x}) = \mathcal{I}_h(\mathbf{x}) + \mathcal{I}_o(\mathbf{x}) = \sum_{i=1}^{N_{\mathcal{I}_h}} G_i^h(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i) + \sum_{j=1}^{N_{\mathcal{I}_o}} G_j^o(\mathbf{x}; \boldsymbol{\mu}_j, \sigma_j) \quad . \tag{4.2}$$

Here $\{G_i^h(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i)\}_i$ is the set of hand input Gaussians with cardinality $N_{\mathcal{I}_h}$ and $\{G_j^o(\mathbf{x}; \boldsymbol{\mu}_j, \sigma_j)\}_j$ is the set of object input Gaussians with cardinality $N_{\mathcal{I}_o}$. Note that $N_{\mathcal{I}_h} + N_{\mathcal{I}_o} = N_{\mathcal{I}}$ holds.

# Chapter 5

# Tracking

The input in the 3D Gaussian representation is used in a generative proposal-based optimization framework to obtain the pose parameters for the hand as well as the rigid transform of the object. A model of the hand together with the object, also in 3D GMM representation, is known to the tracker. The optimization is performed by minimizing an energy which is a function in the pose parameters $\Theta$. Thereby, the GMM formulation allows for analytical derivatives which makes fast gradient-based solvers applicable.

## 5.1 Model Representation

The model used for the hand is a kinematic skeleton with 26 degrees of freedom (DOFs), where 6 parameters determine the global position and rotation and 20 parameters correspond to joint angles — 4 per finger. The distribution of the joints is visualized in Figure 5.1.

Only rigid objects are used in this work hence the pose of the object is fully determined by the 6 DOFs for global position and rotation. Consequently, the tracker is optimizing a total of 32 parameters jointly for obtaining the hand and object pose. The method can be easily generalized to more than one object as shown in Section 6.8.

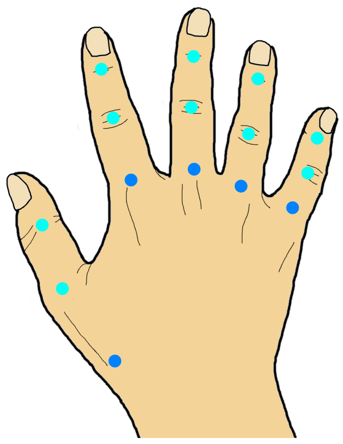


FIGURE 5.1: The locations of the joints in the hand is shown here as blue dots. Light blue dots indicate joints with one DOF whereas dark blue dots belong to joints with two DOFs.

But the computation time per frame increases with increasing number of DOFs. Besides primitive objects like cuboids or cylinders, also more complex objects can be tracked. Such results are presented in Chapter 6. For allowing a better comparison to the input that is encoded in a 3D Gaussian mixture model after preprocessing, also a 3D GMM for the model is built.

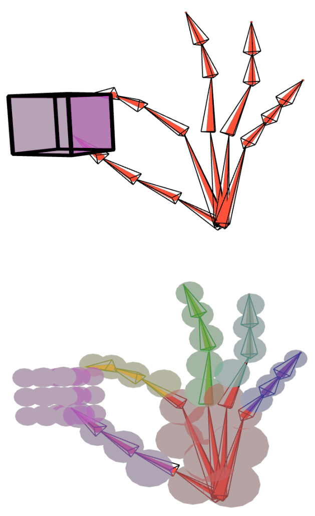


FIGURE 5.2: **Top:** The kinematic skeleton of the hand and a cube represented using its faces. **Bottom:** The 3D GMM of hand and object. The colors indicate different parts.

For the hand, 3-dimensional Gaussians are rigidly attached to the kinematic skeleton as already done in [4]. They are positioned such that the surface coincides with the Gaussians at one standard deviation around the mean. Based on this assumption, the surface of the object is represented with 3-dimensional Gaussians. So the model can be mathematically described as a 3D Gaussian mixture model

$$\mathcal{M}(\Theta, \mathbf{x}) = \sum_{i=1}^{N_{\mathcal{M}}} \upsilon_i G_i(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i) \quad ,$$

where $N_{\mathcal{M}}$ is the number of model Gaussians and $\upsilon_i \in [\upsilon_{low}, 1]$ is the visibility weight for model Gaussian $i$. This weight is needed because only camera facing surfaces have corresponding Gaussians in the input GMM. The influence of occluded parts of the model is lowered by $\upsilon_i$ since there is no correspondence in the input. The visibility weights are computed before the pose optimization of each frame based on the tracked pose from the previous frame. $\upsilon_i$ is set to the maximum of the percental occlusion of model Gaussian $i$ and the lower limit $\upsilon_{low}$. Choosing this lower limit $> 0$ ensures that also parts that are predicted to be fully occluded can still move. This can help in situations where the motion between consecutive frames is large and the visibility prediction is therefore not correct.

In analogy to the input Gaussian mixture (cf. Equation 4.2), a GMM for the part of the model representing the hand, $\mathcal{M}_h$, and a GMM for the object part of the model, $\mathcal{M}_o$, can be defined. Here, $N_{\mathcal{M}_h}$ and $N_{\mathcal{M}_o}$ denote the numbers of Gaussians used to model the hand and the object, respectively. The transition from the hand skeleton and the spatial extent of the object to the 3D GMM is depicted in Figure 5.2. The same parts that are utilized in the classification of the input (cf. Section 4.1) can be annotated for the model Gaussians. These model part labels are used for a semantic comparison to the input during pose optimization.

## 5.2 Energy Formulation

The best pose parameters for the hand and the object are found by minimizing an energy that measures discrepancy between the model in the current pose and the input. Because this is a highly ill-posed problem, especially when only a single camera is used, the energy also includes some regularizers that correspond to reasonable prior assumptions about the tracked motion. Some of these assumptions are specific to the hand and object scenario. The full energy is composed from the following energy terms. The effect of different terms is evaluated in Section 6.7.2.

### 5.2.1 Spatial Alignment

The spatial alignment term $E_a$ as shown in Equation 5.1 is the main data term. It measures the squared difference between the input 3D GMM and the model 3D GMM for the current pose hypothesis $\Theta$, for both hand and object. This difference is integrated at every point over the whole 3D domain.

$$E_a(\Theta) = \int\limits_{\Omega} (\mathcal{I}_h(\mathbf{x}) - \mathcal{M}_h(\Theta, \mathbf{x}))^2 + (\mathcal{I}_o(\mathbf{x}) - \mathcal{M}_o(\Theta, \mathbf{x}))^2 \, d\mathbf{x} \tag{5.1}$$

The integral in $E_a$ has a closed form solution. For the computation please see Section B.1 in the appendix. Furthermore, the spatial alignment term can be put into relation with parts of the objective introduced by Sridhar *et al.* [1]. But in contrast to the objective proposed here, they only work with 2D Gaussians and an additional depth value, a so-called 2.5D representation, leading to a non-continuous formulation. Additional computations and comparisons are demonstrated in Appendix A and in Section 6.7.1.

Figure 5.3 presents the intuition about spatial alignment between two 1D Gaussian functions. The concept can be easily lifted to higher dimensions.

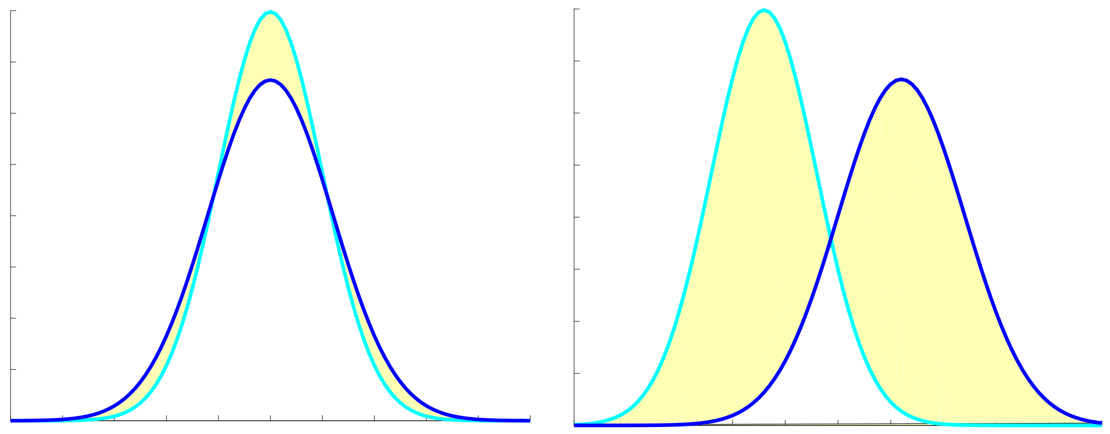


FIGURE 5.3: Graphical example for the spatial alignment term using 1D Gaussians. The difference between the two Gaussians is illustrated as yellow area. Since this difference is squared and integrated over the whole domain, the energy value on the left is lower than the value on the right. That matches the visual intuition about spatial alignment.

### 5.2.2 Label Alignment

Since finding correspondences between the input and the model is in general hard, the previous chapter (more specifically Section 4.1) introduced a classification method for hand parts and the object. Output of this method are per-pixel histograms representing the class confidence distribution that are clustered together with the input depth $\mathcal{D}_t$. Instead of using the so found correspondences in an inverse kinematics (IK) approach as hard constraints, the label alignment term $E_l$, stated in Equation 5.2, softly incorporates the part histograms.

$$E_l(\Theta) = \sum_{i=1}^{N_{\mathcal{I}}} \sum_{j=1}^{N_{\mathcal{M}}} \alpha_{i,j} \cdot ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2^2 \tag{5.2}$$

For evaluating the label alignment, every pair of an input and a model Gaussian is considered. The distance between them should be minimized if the highest confidence class $l_i$ of the input is equal to the class label $l_j$ of the model Gaussian. Thereby also the confidence $p_i$ for $l_i$ has to be taken into account: a higher confidence should yield stronger attraction forces. Furthermore, a pair of Gaussians is discarded if they are farther apart than the maximal interaction range $r_{max}$. Assuming the tracking of the previous frame to be correct, this helps to eliminate classification outliers. All this is realized in the following weight

$$\alpha_{i,j} = \begin{cases} 0 & \text{if } (l_i \neq l_j) \text{ or } (d_{i,j} > r_{max}) \\ (1 - \frac{d_{i,j}}{r_{max}}) \cdot p_i & \text{else} \end{cases} ,$$

where $d_{i,j}$ is the distance between input Gaussian $i$ and model Gaussian $j$.

### 5.2.3 Anatomical Plausibility

The first general prior term is concerned with anatomical plausibility. For the human hand it applies that fingers cannot be stretched or bent over a certain limit. These joint limits have been empirically measured in user studies [35] and can be described by a vector of lower limits $\Theta^l$ and a vector of upper limits $\Theta^u$. Since there is no hard constraint for these limits in the kinematic skeleton, they are softly enforced by the following term

$$E_p(\Theta) = \sum_{\theta_j \in \Theta} \begin{cases} 0 & \text{if } \theta_j^l \leq \theta_j \leq \theta_j^u \\ (\theta_j - \theta_j^u)^2 & \text{if } \theta_j > \theta_j^u \\ (\theta_j - \theta_j^l)^2 & \text{if } \theta_j < \theta_j^l \end{cases} . \tag{5.3}$$

As soon as either the lower or upper joint limit for a DOF is exceeded, a penalty in the form of the squared deviation is added to the energy. Using a soft constraint makes tracking of extreme poses possible, *e. g.* when someone has significantly more flexible fingers than the average, if there is strong evidence in the input.

### 5.2.4 Temporal Smoothness

The second general prior penalizes temporally non-smooth pose changes. This energy term is based on the assumption that the parameter variation of consecutive frames is similar. As shown in Equation 5.4, the difference between the current pose gradient and the pose gradient of the previous frame is penalized.

$$E_t(\Theta^{(t)}) = ||\nabla\Theta^{(t)} - \nabla\Theta^{(t-1)}||_2^2 \tag{5.4}$$

Note that the superscripts $t$, $t-1$, $t-2$ refer to the current and the previous time steps, respectively, and are used here for disambiguation. The pose parameter variation $\nabla\Theta$

is approximated using first-order finite backward differences. One can reformulate $E_t$ employing this approximation which yields

$$\begin{aligned} E_t(\Theta^{(t)}) &= ||(\Theta^{(t)} - \Theta^{(t-1)}) - (\Theta^{(t-1)} - \Theta^{(t-2)})||_2^2 \\ &= ||\Theta^{(t)} - 2 \cdot \Theta^{(t-1)} + \Theta^{(t-2)}||_2^2 \qquad . \end{aligned} \tag{5.5}$$

The underlying assumption of similar pose changes for successive points in time seems to be reasonable for tracking natural hand motion as presented *e. g.* in [13]. The temporal smoothness term avoids high-frequency jitter in the tracked result.

### 5.2.5 Contact Points

Although joint tracking of the hand interacting with an object introduces new challenges and intensifies challenges known from single hand tracking, specific knowledge about the scenario can be exploited for achieving this goal. During natural interaction, the human hand usually grabs or holds the object. This leads to contact between — at least — the fingertips and the surface of the object. The contact points term $E_c$ tries to preserve these touch constraints once detected until there is enough evidence that the touch was released. For this purpose, a set of contacts $\mathcal{T}$ is maintained. The elements of $\mathcal{T}$ are triples $(f, j, d_\mathcal{T})$ where $f$ is the fingertip identifier, $j$ is the unique identifier of the object model Gaussian that is touched and $d_\mathcal{T}$ is the so-called *touch distance*.

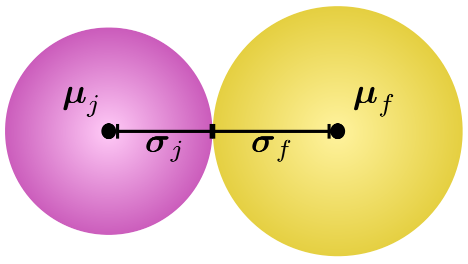


FIGURE 5.4: A fingertip Gaussian (right, yellow) is touching an object Gaussian (left, pink). One can see that the length of the vector connecting the means is equal to the sum of the standard deviations.

Note that the Gaussian mixture models for the hand and the object were built satisfying the invariant that the surface coincides with the isosurface of the Gaussians at one standard deviation from the mean. Therefore, as depicted in Figure 5.4, a fingertip touches a point on the object if and only if the distance between the means of the corresponding Gaussians is equal to the sum of the standard deviations.

Hence, a pair of fingertip Gaussian $f$ and object Gaussian $j$ with touch distance $d_\mathcal{T} = \sigma_f + \sigma_j$ is added to $\mathcal{T}$ if their distance $||\boldsymbol{\mu}_f - \boldsymbol{\mu}_j||_2$ lies inside the respective touch distance interval $[d_\mathcal{T} - \epsilon_\mathcal{T}, d_\mathcal{T} + \epsilon_\mathcal{T}]$ $(\epsilon_\mathcal{T} > 0)$. For a more detailed description of variants to update

$\mathcal{T}$ see Section 5.2.8. For a given set of contact points, $E_c$ penalizes deviations from the touch distance for each element of $\mathcal{T}$ as follows

$$E_c(\Theta) = \sum_{(f,j,d_{\mathcal{T}}) \in \mathcal{T}} \left( ||\boldsymbol{\mu}_f - \boldsymbol{\mu}_j||_2^2 - d_{\mathcal{T}}^2 \right)^2 \quad . \tag{5.6}$$

Preserving the detected contact points may seem rigorous. But experimental results (see Chapter 6) show that careful balancing between the energy terms lets the data term take over when the evidence that the contact was released is strong enough.

### 5.2.6 Object Occlusion

Similar to the contact points term, the object occlusion term $E_o$ tries to exploit information about the hand-object-scenario to guide the pose estimation to more plausible poses.

A hard challenge that is prominent to the single camera setup are occlusions. Whereas in single hand tracking mostly self-occlusions have to be handled, joint tracking of hand and object motion needs to be robust to frequently occuring occlusions of the hand caused by the object. In the following, *object occlusion* refers to this specific kind of occlusion. Because there is no data in occluded regions, some prior assumption for such regions is necessary to stabilize the tracking. In this work, it is assumed that occluded hand parts move rigidly with the closest non-occluded ancestor in the kinematic chain. Only global translation and rotation of the hand is always possible. A value $\rho_i \in [0, 1]$ can be computed for every hand model Gaussian $i$ that measures percental occlusion by the object as observed from the camera. Therefore, the Gaussian representation for the model as shown in the bottom part of Figure 5.5 and the camera intrinsics are used. $\rho_i = 0$ corresponds to no occlusion and $\rho_i = 1$ to a fully occluded Gaussian. This continuous occlusion factor controls how strong the rigid movement with the nearest non-occluded ancestor is enforced. Note that $\rho_i$ is computed on the volumetric representation of hand and object, the GMM. To connect the factors to the rigidity of hand parts, a mapping to the DOFs of the kinematic skeleton is needed. Thus, hand model Gaussian $i$ is put in relation with the DOFs that directly affect this Gaussian, the so-called *parent DOFs*. These DOFs are described by the set $\mathcal{H}_i$. The top part of Figure 5.5 visualizes the parent DOFs of Gaussians with high $\rho$-values in red.

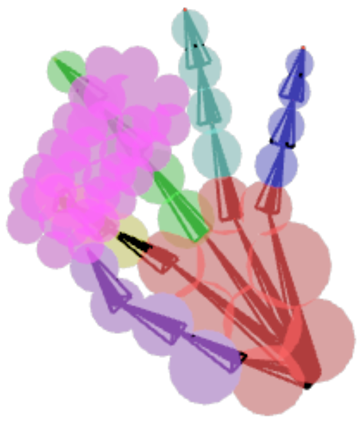
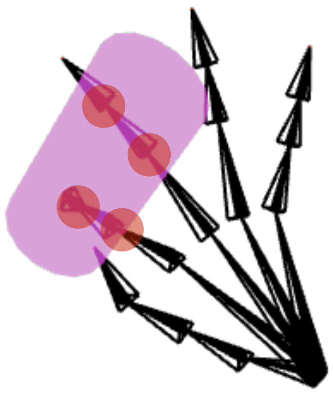



FIGURE 5.5: **Left:** An example pose of the GMM model as seen from the camera. The pink cylinder object Gaussians occlude huge parts of the index and the middle finger. **Right:** The skeleton and the cylinder object in the same pose as above. The DOFs of the hand that directly influence the severely occluded finger Gaussians are marked in red.

The aforementioned concepts are integrated in the energy as

$$E_o(\Theta) = \sum_{i=1}^{N_{\mathcal{M}_h}} \sum_{j \in \mathcal{H}_i} \rho_i \cdot (\theta_j - \theta_j^{saved})^2 \qquad . \tag{5.7}$$

The soft constraint that hand parts should adapt to the rigid motion of non-occluded ancestors is realized by penalizing deviation from a saved value for DOF $\theta_j$. So based on the value $\rho_i$, the parent DOFs are more or less significantly disabled. For saving the reference values $\Theta^{saved}$ different schemes can be employed. Details are described in Section 5.2.8.

### 5.2.7 Interpenetration Avoidance

Although the object specific terms $E_c$ for contact points and $E_o$ for object occlusion relate the hand to the object, the main data term — the spatial alignment term $E_a$ — does not. This can be seen from Equation 5.1 by splitting the sum under the integral. Therefore, a regularizer is necessary to prevent the hand and the object part of the model from interpenetration. When such an interpenetration happens, the affected object model Gaussians from $\mathcal{M}_o$ and the affected hand model Gaussians from $\mathcal{M}_h$ are very close, *i. e.* their spatial overlap is high. The overlap of two Gaussians can be measured by integrating the product over all points in the 3D domain. Generalizing this to Gaussian mixture models leads to the following formulation for the interpenetration avoidance term

$$E_i(\Theta) = \int_{\Omega} \mathcal{M}_h(\Theta, \mathbf{x}) \cdot \mathcal{M}_o(\Theta, \mathbf{x})\, d\mathbf{x} \quad . \tag{5.8}$$

Because interpenetration should also be avoided in occluded regions to obtain a physically plausible pose, the visibility weights $\upsilon_i$ are ignored for $E_i$. This is a legitimate step since $E_i$ does not compare against the input GMM that represents only parts that are visible from the camera.

### 5.2.8 Per-Frame Constraints vs. Constraint Propagation

The contact points $\mathcal{T}$ for $E_c$ as well as the reference DOF values $\Theta^{saved}$ for $E_o$ need to be available before the pose optimization for the current time step $t$ is started.

**Per-Frame Constraints.** The local approach determines these constraints on a per-frame basis using the tracked pose from the previous frame $t-1$, here denoted as $\Theta^{(t-1)}$. Before starting the actual pose estimation, $\Theta^{saved} = \Theta^{(t-1)}$ is set. Thus, if hand Gaussians are occluded by the object, their parent DOFs $\theta_j$ are penalized for deviations from the previous DOF value $\theta_j^{(t-1)}$. Such a constraint is reasonable in cases where $\theta_j^{(t-1)}$ is plausible for the occluded hand part. But when parts are occluded for a longer period of time, it is possible that these parts drift away from the reference they had when the occlusion began. This happens because the deviation sums up over the frames without increasing the penalty. Similar issues can be encountered when the contact points in $\mathcal{T}$ are recomputed at each frame. Whereas the touch of a fingertip might have started at a specific object Gaussian, the contact correspondence might be shifted to one of the neighbors due to small jitter in the tracking. In other situations, *e. g.* when the object is touched and the finger is then slided over the surface, the per-frame recomputation is desirable because it makes tracking of such motions possible.

**Constraint Propagation.** A more global approach is to update the constraints when the contact or the occlusion starts happening and to propagate this information across consecutive frames until the contact is released or the occlusion is gone. So for a DOF $\theta_j$, the reference value is set when the occlusion value $\rho_i$ of the directly attached Gaussians becomes larger than an occlusion threshold $\hat{\rho}$. This saved DOF value is then fixed and propagated until the occlusion drops below $\hat{\rho}$. For DOFs corresponding to only lightly occluded Gaussians ($\rho_i < \hat{\rho}$), the references are again saved per-frame. This is based on the assumption that $\theta_j^{(t-1)}$ is a plausible reference because there was enough evidence in the input. Propagating the reference for highly occluded regions over frames reduces drifting. For the contact constraints $\mathcal{T}$, an element is added when the touch is first detected. This contact correspondence between fingertip $f$ and object Gaussian $j$ is kept and propagated until the distance $||\boldsymbol{\mu}_f - \boldsymbol{\mu}_j||_2$ exceeds a threshold $d_{\mathcal{T}}$. By choosing $d_{\mathcal{T}}$ larger than the distance to the neighboring Gaussians, the touch correspondence to $j$ is not lost when the fingertip shifts slightly due to jitter. Only one correspondence per fingertip at the same time is possible in this version. Otherwise multiple contact constraints to not neighboring Gaussians could be accumulated over frames yielding the mean of all constraints as the optimum. As already pointed out in the previous paragraph, tracking of touch and slide motions is hard if the contact points are not recomputed per frame. When the constraint propagation over frames is used, such motions can only be tracked if the evidence in the input is strong enough to pull the fingertip away from the saved contact point.

Since there are advantages and disadvantages for both variants, they are further discussed and evaluated for the contact points term $E_c$ and the object occlusion term $E_o$ in Chapter 6.

## 5.3 Gradient-Based Multi-Proposal Optimization

The generative optimizer is proposal-based, *i. e.* multiple pose hypotheses are maintained during optimization of a single frame. In this work, two proposals are used that also have a different objective to minimize, respectively. One proposal corresponds to the energy $E_{depth}$ (see Equation 5.9) that does not include any information from the discriminative part classification. The second proposal optimizes $E_{label}$ (see Equation 5.10) in which this additional knowledge is included. It has been shown in previous work, *e. g.* by Sridhar *et al.* [1], that a split like this can be beneficial. Especially in situations where the classification result is bad, considering a hypothesis that is independent from this information avoids tracking failures.

$$E_{depth}(\Theta) = E_a + w_p E_p + w_t E_t + w_c E_c + w_o E_o + w_i E_i \tag{5.9}$$

$$E_{label}(\Theta) = E_a + w_l E_l + w_p E_p + w_i E_i \tag{5.10}$$

Whereas the depth proposal uses all regularizers and hand-object specific energy terms that have been introduced in this section, some of the terms are excluded for $E_{label}$. Because the label proposal should help to recover from tracking failures, the temporal smoothness assumption is dropped. Furthermore, the object occlusion and the contact points term are ignored out of similar reasons: DOFs disabled due to occlusion and contact constraints for fingertips should not prevent motion according to correspondences found via the classification algorithm. Additional experiments on the employed multi-hypotheses strategy are provided in Chapter 6.

The optimization is performed by step-size-adaptive gradient descent that is initialized with the winner pose from the previous time step $t-1$. Since both $E_{depth}$ and $E_{label}$ have analytical gradients with respect to the pose parameters $\Theta$, the algorithm runs in real time without using GPU acceleration. A comprehensive computation of the gradients is included in Appendix B. The gradient descent is stopped if either a fixed number of iterations is reached or the gradient vanishes. Let the final poses after the optimization for the proposals be denoted as $\Theta^*_{depth}$ and $\Theta^*_{label}$. To determine which of these two poses better explains the input, they are compared using a comparison energy $E_{comp}$ that is defined as

$$E_{comp}(\Theta) = E_a + w_p E_p + w_i E_i \qquad . \tag{5.11}$$

To make the comparison as unbiased as possible, all regularizers except the anatomical plausibility and the interpenetration avoidance are ignored. So the pure spatial alignment of the found poses to the input is crucial for the decision. To enable fast recovery also from smaller errors, the label pose $\Theta^*_{label}$ is slightly preferred, based on the factor $\lambda > 1$. Thus, the winner pose is given as

$$\Theta^* = \begin{cases} \Theta^*_{label} & \text{if } E_{comp}(\Theta^*_{label}) < \lambda E_{comp}(\Theta^*_{depth}) \\ \Theta^*_{depth} & \text{otherwise} \end{cases} \qquad . \tag{5.12}$$

# Chapter 6

# Evaluation and Discussion

This section describes various experiments and tests carried out to evaluate the performance of the presented hand and object tracking approach. These include, for example, evaluation on a newly captured and annotated dataset (Section 6.4), comparisons on two publicly available datasets (Sections 6.5 and 6.6), qualitative results with different users, objects and camera locations (Section 6.8) and results showing the significance of several algorithm parts (Section 6.7).

## 6.1 Captured Sequences

To evaluate the performance of the approach proposed in this work, several challenging sequences showing a hand interacting with an object have been recorded.

The objects vary in size and shape as one can see in Figure 6.1. For easier color segmentation of the object (see Section 4.1.2), all objects used in the captured sequences have the same dark green color. Note that also non-primitive shapes and objects with different and multiple colors can be tracked as long as the colors are unique in the foreground of the scene.

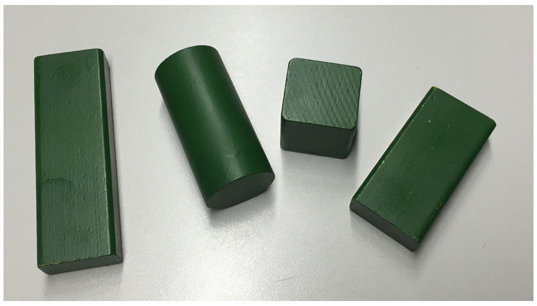


FIGURE 6.1: The primitive objects used for the evaluation (from left to right): *LongCuboid*, *Cylinder*, *Cube* and *ShortCuboid*. Note that also different shapes are possible and the color does not need to be green as shown amongst others in Sections 6.5 and 6.6.

This capability is presented on different datasets (see Sections 6.5 and 6.6) and in live captures (see Section 6.8). All following sequences have been captured with an Creative Senz3D at a frame rate of 30 Hz for both depth and color stream.

### 6.1.1 Annotated Dataset *Dexter++*

Six of the captured sequences have been manually annotated and form the new benchmark dataset *Dexter++*. The sequences show interactions of the left hand with the *ShortCuboid* and the *LongCuboid* objects performed by two different subjects (one female, one male). In contrast to other datasets that only provide ground-truth annotations for the hand, *e. g.* Tzionas *et al.* [2], *Dexter++* has fingertip as well as cuboid annotations for more than 3000 frames. The sequences consist of various complex articulated motions and severe occlusion situations:

- *Occlusion* (Subject 1, *ShortCuboid*, 353 frames):
  The hand is static in the same pose for the whole sequence while the cuboid is moved in front of the hand yielding heavy occlusions.
- *Rigid* (Subject 1, *ShortCuboid*, 530 frames):
  The second longest edge of the cuboid is held between thumb and index finger and the hand moves rigidly. The pose of hand and cuboid is affected by large positional and rotational changes.
- *Pinch* (Subject 2, *ShortCuboid*, 457 frames):
  The hand is turned sideways and holds the long edge of the cuboid between thumb and index finger. Middle finger and ring finger pinch on the cuboid repeatedly.
- *Rotate* (Subject 1, *ShortCuboid*, 797 frames):
  The short edge of the cuboid is held between thumb and index finger. The cuboid is rotated sideways and moved up and down. Towards the end, the cuboid is grasped with the whole hand.
- *Grasp1* (Subject 1, *ShortCuboid*, 355 frames):
  The hand is seen from the back grasping the cuboid with thumb and index finger. It is rotated and put down again.

- *Grasp2* (Subject 1, *LongCuboid*, 747 frames):

  The hand is seen from the back, grasping the cuboid multiple times. The cuboid is rotated by 90° in each grasp. The sequence ends with a full grasp (using all fingers).

Figure 6.2 displays example depth and color images for each sequence including annotated points. For the cuboid, three corners on the largest face are annotated in a fixed order: start in one corner, proceed along the longest edge, then proceed along the second longest edge. Note that these three points together with the position and viewing direction of the camera fully determine the rigid transform of the cuboid. If none of the two largest faces is visible, no ground truth for the cuboid can be provided for this frame. Similarly, there is no ground truth for occluded fingertips. Qualitative and quantitative results on *Dexter++* are presented in Section 6.4.

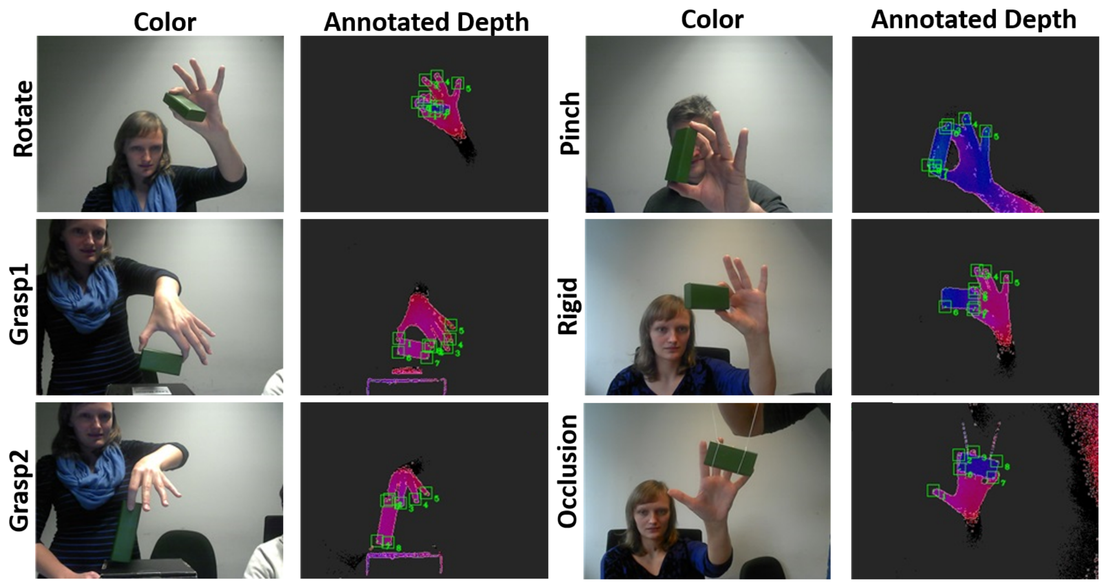


FIGURE 6.2: Characteristic examples for depth and color input for each of the *Dexter++* sequences. The depth image is shown color-coded where blue indicates close pixels and magenta indicates pixels that are farther away. In addition, the depth map includes the annotations for the fingertips and three cuboid corners in green.

### 6.1.2 Additional Sequences

More sequences have been captured but not annotated. These sequences show interactions with other objects or interactions that are hard to annotate manually due to their high complexity.

- *ThrowCube*: The *Cube* is swinging on a thread and is caught with the hand.
- *Cylinder*: The *Cylinder* is grasped and lifted from a table producing severe occlusions of the fingers.
- *Complex*: Some highly complex motions, similar to pen spinning, are performed with the *ShortCuboid*.

The qualitative performance of the algorithm on these sequences is demonstrated in Sections 6.7.2 and 6.8.

## 6.2 Runtime and Parameters

All experiments were performed using an Intel Xeon E5-1620 CPU with 16 GB of RAM and an Nvidia GeForce GTX Titan X GPU. The main part of the algorithm is parallelized on the CPU using OpenMP. Only the classification using random decision forests runs in parallel on the GPU. Overall, the implementation achieves real-time performance of **25–35 Hz**. The stages of the proposed hand and object tracking method take: 3 ms for preprocessing, 3.5 ms for RDF classification, 2 ms for clustering and building the input GMM, 20–30 ms for proposal-based optimization. The variation in the latter is mostly due to the varying number of Gaussians in the models for different objects. The use of propagated constraints vs. per-frame constraints (see Section 5.2.8) does not noticeably change the processing speed.

When running the algorithm in the configuration for tracking a single hand — *i. e.* the contact point constraints, the object occlusion handling and the interpenetration avoidance are disabled — only 12 ms are needed for the multi-proposal pose optimization on average. This yields a frame rate of 50 Hz for single hand tracking.

All weights and parameters for all energy components were determined empirically on the *Dexter++* dataset. The weights are: $w_l = 3 \cdot 10^{-7}$, $w_p = 0.1$, $w_t = 0.1$, $w_c = 5 \cdot 10^{-7}$, $w_o = 1.0$, $w_i = 3 \cdot 10^{-7}$. The additional parameters were chosen as:

- minimum visibility for model Gaussians $\upsilon_{low} = 0.15$
- label alignment: maximal interaction range $r_{max} = 250\,\text{mm}$

- contact points: touch threshold $\epsilon_{\mathcal{T}} = 5\,\text{mm}$, touch release distance $d_{\overline{\mathcal{T}}} = 20\,\text{mm}$ (propagated version)
- object occlusion: occlusion threshold $\hat{\rho} = 0.75$ (propagated version)

## 6.3 Are Separate Hand and Object Trackers Sufficient?

A first interesting question is whether joint hand and object tracking is necessary given that there are plenty of fast and robust hand-only and object-only tracking approaches as presented in Chapter 2. Already Kyriazis *et al.* [17], for example, found out that the performance is worse when using separate independent trackers since there are no mutual constraints incorporated in the tracking. Especially in situations with heavy occlusions, knowledge about both, the hand and the object, improves the tracking results.

To support this claim, the state-of-the-art hand tracker of Sridhar *et al.* [1] is evaluated on the *Occlusion* sequence from the new *Dexter++* dataset. Using separate trackers for the hand and the object relies on the assumption that accurate hand-object segmentation is available. To ensure this, all object pixels are removed from the depth map by setting their value to infinity before running the hand tracker.

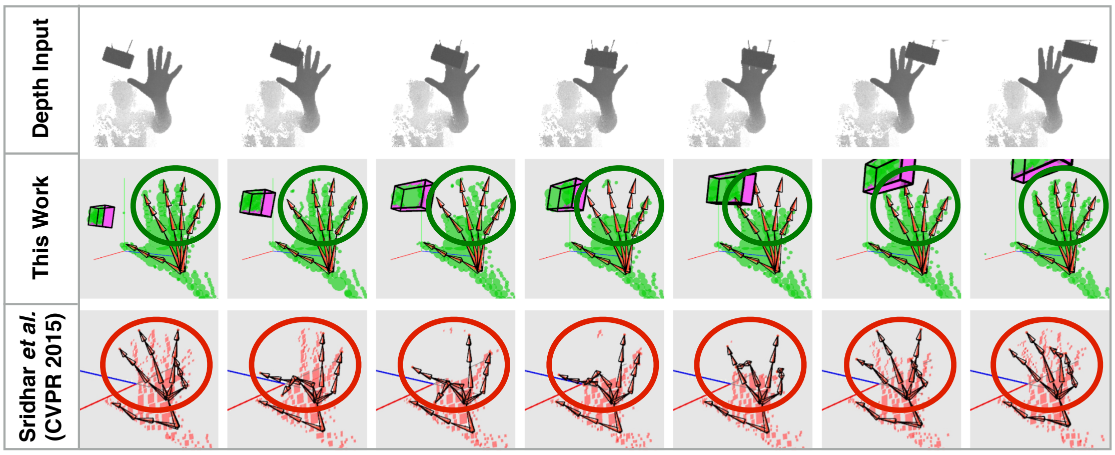


FIGURE 6.3: **Top:** The original input depth images. Note that the object pixels are removed for the hand tracker of Sridhar *et al.* [1]. **Middle:** Results achieved by the joint hand and object tracking approach proposed in this thesis. The occluded fingers remain stable because the pose of the object from the previous frame is known and the new pose of hand and object is optimized jointly. **Bottom:** Qualitative results of the hand tracker of Sridhar *et al.* The tracking of the fingers is not stable under occlusion. Note how the fingers snap towards the remaining depth pixels of the palm because there is no information about the object.

The results are presented in Figure 6.3, together with the results achieved by this work for comparison. Whereas the joint hand-object approach tracks correctly, the method of Sridhar *et al.* fails.

## 6.4 Evaluation on *Dexter++*

The proposed method for simultaneous tracking of hand and object pose is evaluated on the new dataset *Dexter++*.

**Error Measure.** The frames from the dataset provide ground-truth annotations for all visible fingertips and for three corners of the cuboid. The average 3D position error in millimeters is computed over all backprojected annotated points and the corresponding tracked positions.

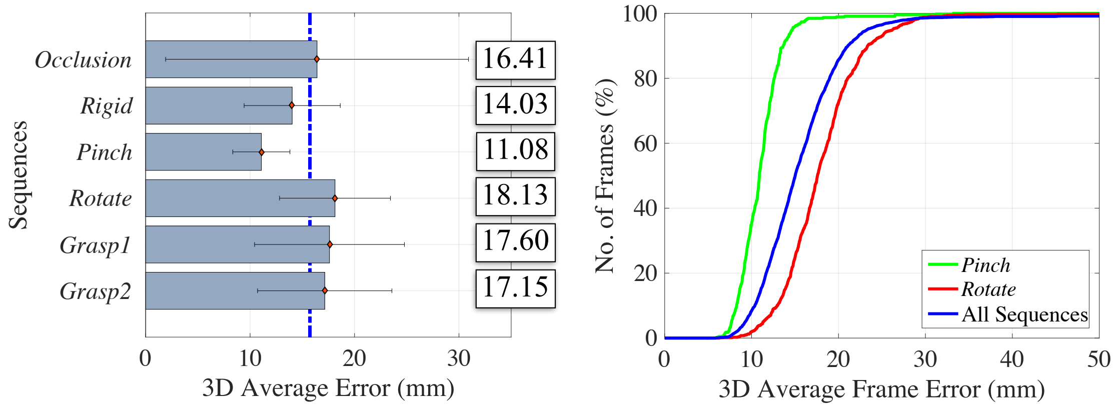
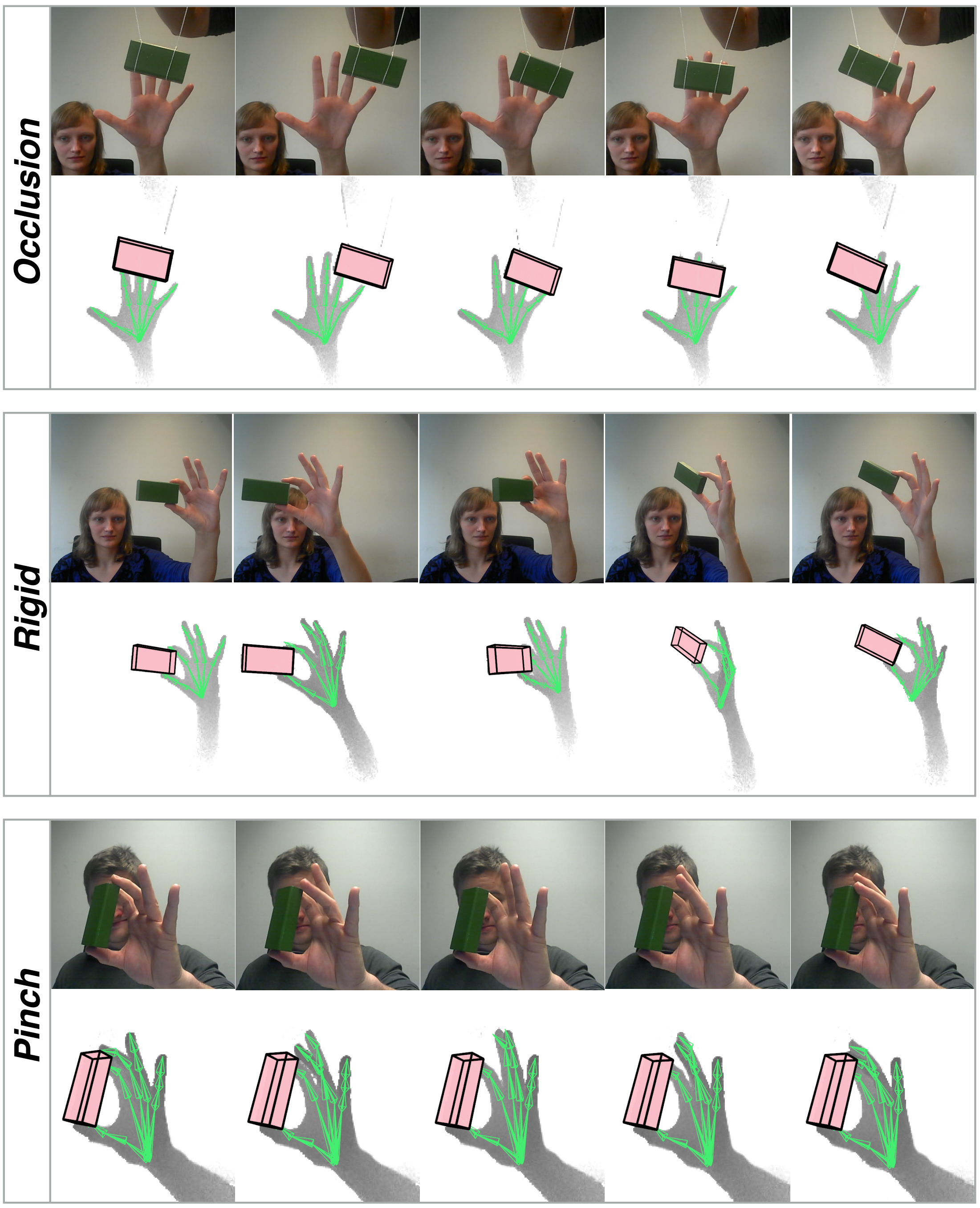


FIGURE 6.4: **Left:** Average error and standard deviation per sequence of the *Dexter++* dataset. The exact number is displayed on the right. The average error over all sequences is indicated by the dark blue line. The error for *Pinch* is lowest whereas the error for *Rotate* is highest but still close to the average. **Right:** Consistency curves for the best (*Pinch*) and the worst (*Rotate*) sequence together with the average consistency curve over all sequences. On average, an error $< 15$ mm is achieved on ca. 40% of the frames. For the best sequence, such a low error is obtained on almost all frames ($> 95\%$), but on the worst sequence this holds only for 20% of the frames.

Figure 6.4 (left) displays the average frame error per sequence. Over all sequences, the total average error is **15.73 mm**. The average fingertip error (**15.63 mm**) is slightly lower than the error of the object (**16.20 mm**). Since the average error is not stable to outliers, the right part of Figure 6.4 visualizes the consistency of the tracking accuracy. This plot shows the percentage of frames (y-axis) that achieved an error below a certain threshold (x-axis). Qualitative results on all six sequences are shown in Figure 6.5.


Occlusion
Rigid
Pinch

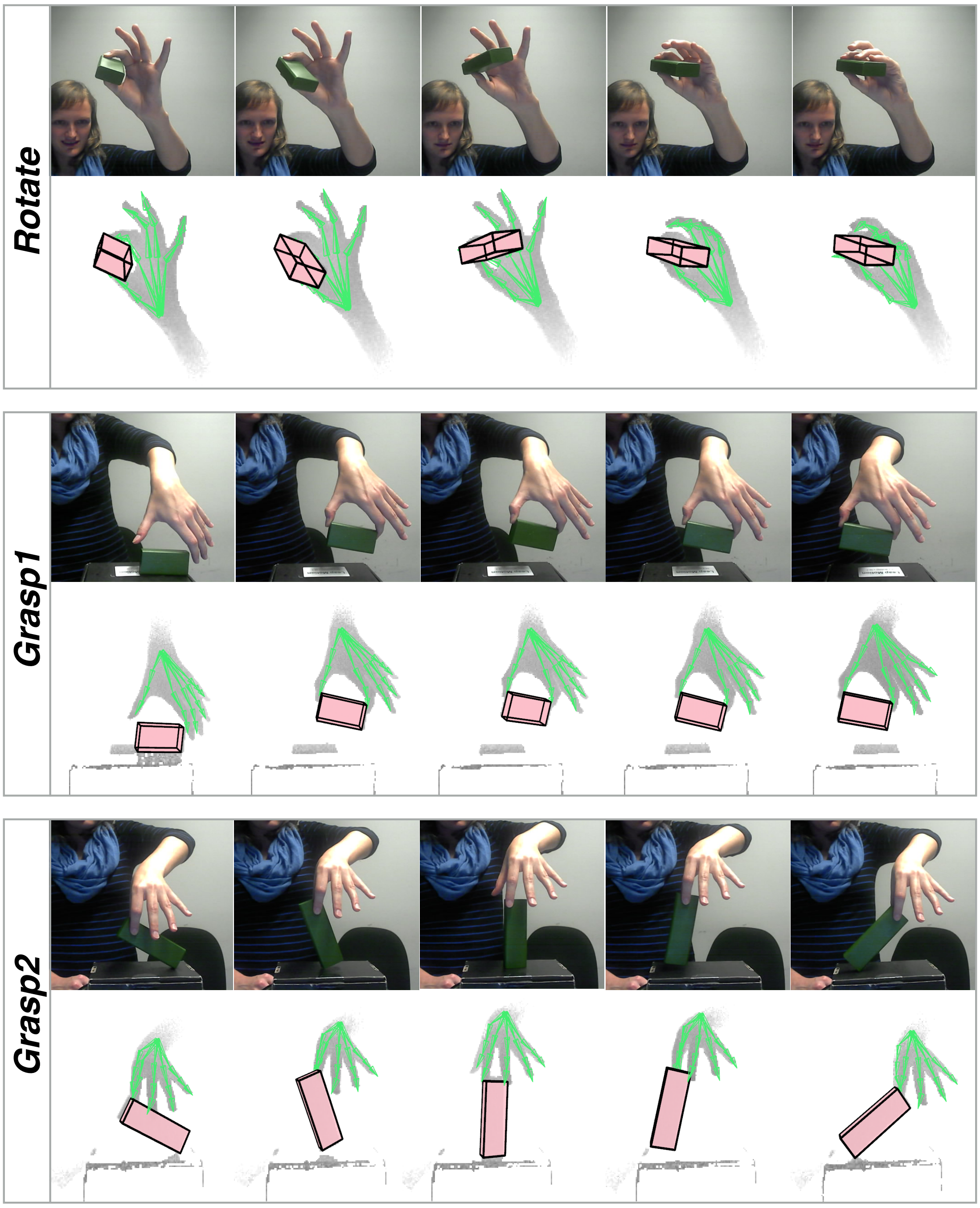


FIGURE 6.5: Qualitative results for all *Dexter++* sequences. The top row shows color input frames and the bottom row depth input together with overlaid tracking results (green for hand and pink for object), respectively. Note how the tracking remains stable under occlusions and how natural grasps are reproduced using contact point and interpenetration avoidance constraints.

## 6.5 Evaluation on *IJCV hand-object interactions*

Tzionas *et al.* recently proposed a tracker for hands in interaction [2]. Their dataset includes sequences with a single hand, two interacting hands and one or two hands manipulating an object rigidly or nonrigidly. They also provide their tracked hand motions for comparison. Because the approach in this thesis is not able to cope with nonrigid motion of the object, the performance is only evaluated on the sequences including a rigid object. There are six such sequences, from which five have ground-truth annotations for the hand:

- *Ball1*: Number 15 in dataset. A blue ball is moved with one hand from a raised platform to a lower platform.
- *Ball2*: Number 16 in dataset. A blue ball is moved with two hands from the lower to the raised platform.
- *Ball1 Occ*: Number 19 in dataset. A blue ball is moved with one hand from the lower to the raised platform. The thumb is occluded by the ball.
- *Cube*: Number 20 in dataset. A Rubik's Cube is moved with one hand from the lower to the raised platform.
- *Cube Occ*: Number 21 in dataset. A Rubik's Cube is moved with one hand from the lower to the raised platform. The thumb is occluded by the cube.
- *X1*: Number $X1$ in dataset, without annotations. A blue ball is moved with one hand from the raised platform to the palm of the second hand and from there to the lower platform.

All sequences were captured at 30 Hz using a Primesense Carmine 1.09 short-range structured-light RGB-D camera.

**Error Measures.** Ground-truth annotations are only provided for the hand. Therefore, the quantitative evaluation is limited to the accuracy of the hand pose estimation. The joints that are used for the evaluation are depicted in Figure 6.6. Two error measures are computed for all annotated frames:

- **2D pixel error (in px):**
  The average distance between the annotated pixel and the projection of the tracked joint position into the depth image over all fingers of the right hand.

- **3D position error (in mm):**
  The average distance between the backprojected annotated position and the tracked joint position over all fingers of the right hand.

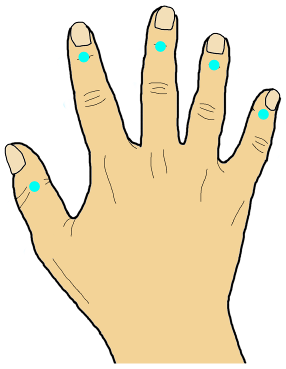


FIGURE 6.6: The joints (in cyan) used for the quantitative comparison to Tzionas *et al.*

Figure 6.7 shows quantitative results for the five annotated sequences. Overall, the error of the method presented in this thesis is **7.9 px** (2D) and **8.4 mm** (3D) whereas the approach of Tzionas *et al.* achieves **6.0 px** and **8.0 mm**. These differences lie within the variance of manual annotation. In comparison to Tzionas *et al.* who use computationally expensive physics simulation, this work runs more than 60 times faster while obtaining similar results. Qualitative results for visual comparison are provided in Figure 6.8. Furthermore, Figure 6.9 shows qualitative results from the *X1* sequence.

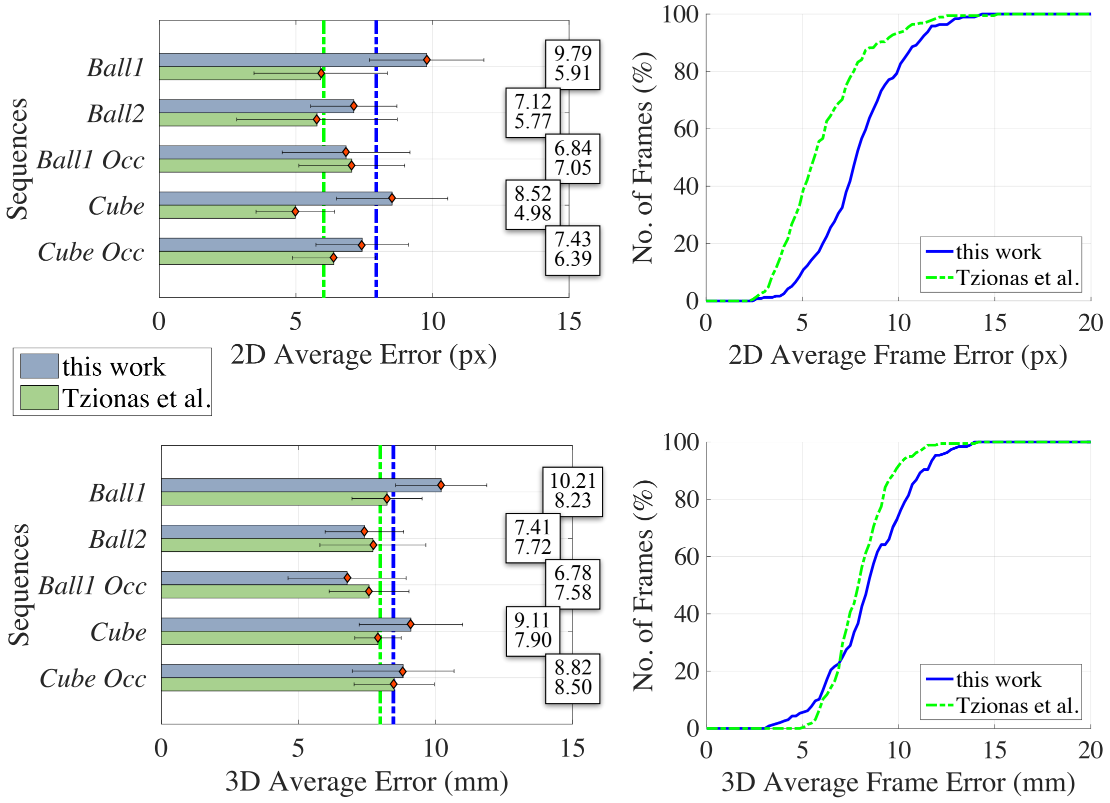
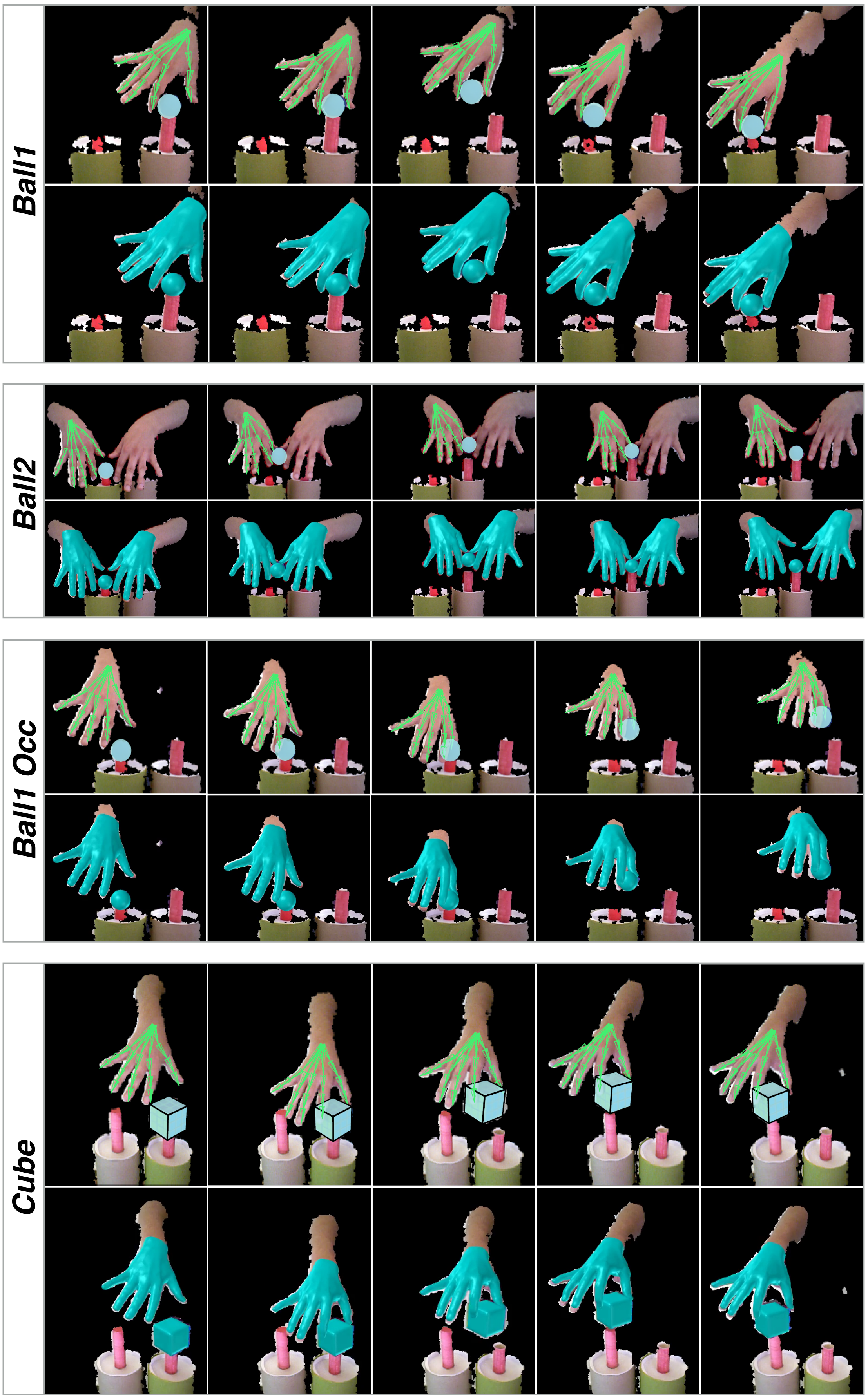


FIGURE 6.7: **Left:** The average errors in 2D and 3D for all sequences. The exact numbers are displayed in the boxes on the right. All 2D errors (top) lie below 10 px and almost all 3D errors (bottom) below 10 mm. The difference in performance between this work and Tzionas *et al.* [2] is slightly larger for the 2D error metric. On *Ball1* and *Cube*, Tzionas *et al.* perform significantly better in 2D but only slightly better in 3D. Overall, the algorithm from this thesis achieves lower average error on one sequence using the 2D metric and on two sequences using the 3D metric while running more than 60 times faster. **Right:** Consistency plots as introduced in Section 6.4 for the 2D and 3D error metric over all sequences. Similarly to the average errors (left), one can see that the discrepancy between the trackers is larger in the 2D metric (top). Here the consistency curve of Tzionas *et al.* [2] is always above indicating more frames with lower error. But for the 3D metric, the curve of this work is above until a threshold of ca. 7 mm, showing that there are more frames that are tracked very accurately. Overall, both methods track all frames with an error lower than 15 px (2D) and 15 mm (3D).


Ball1
Ball2
Ball1 Occ
Cube

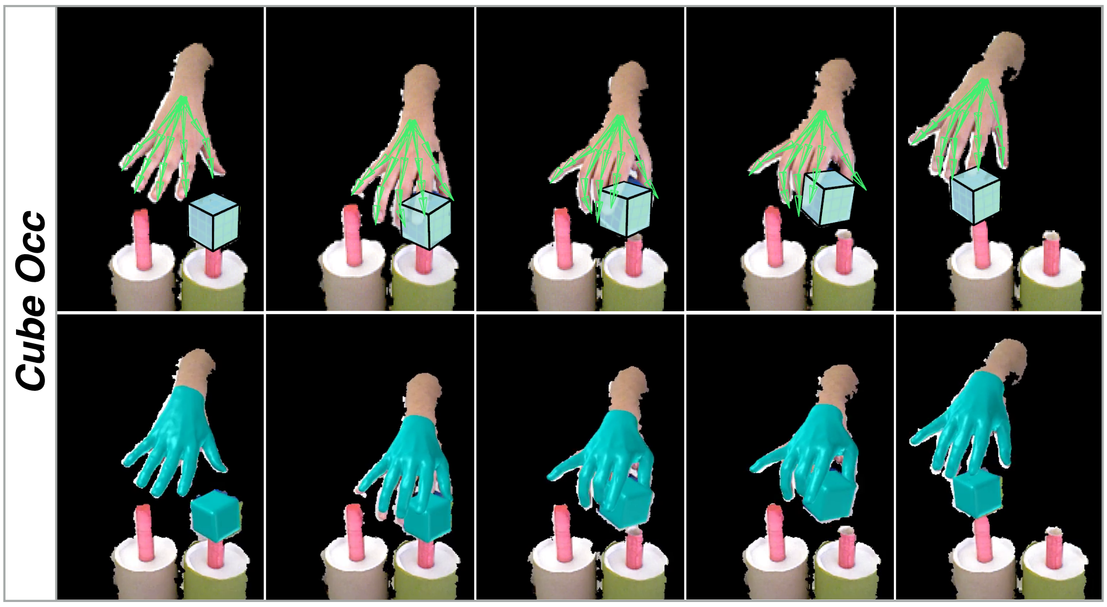


FIGURE 6.8: Qualitative results on the annotated sequences presented as overlay with the colored input point cloud. The approach presented in this thesis (top row in green and light blue, respectively) and the method of Tzionas *et al.* [2] (bottom row in turquoise, respectively) achieve similar visual quality although the latter runs offline and the first in real time. Note that this work can only track one hand for the *Ball2* sequence but the tracking remains stable even though a second hand is present in the input. The capability of tracking objects with multiple colors is also demonstrated by tracking the Rubik's Cube.

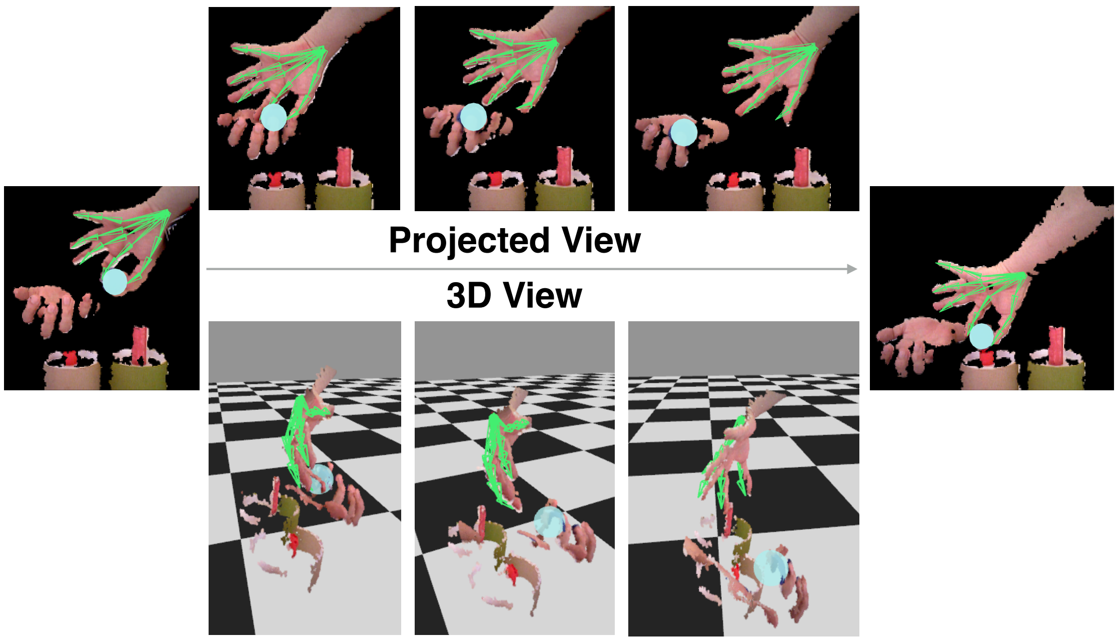


FIGURE 6.9: Qualitative results (in green and light blue) on the *X1* sequence from Tzionas *et al.* [2]. In the projected view, the colored input depth image is overlaid with the projected tracking result. The bottom row shows the corresponding 3D view of the colored point cloud and the tracking result for the hand and the object. In this view, one can see that the ball is tracked accurately on the palm of the second hand although it is partially occluded by the fingers.

## 6.6 Evaluation on *ICCV in-hand scanning*

Even though the work on in-hand scanning by Tzionas and Gall [3] solves a different problem, the performance of the hand and object tracking approach from this thesis can be evaluated on their sequences.

The data was captured using a Primesense Carmine 1.09 short-range structured-light RGB-D camera.

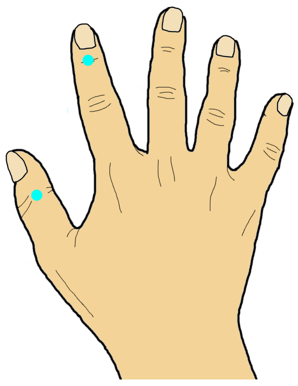


FIGURE 6.10: The outermost joint of the thumb and the index finger (in cyan) are annotated for the in-hand scanning sequences of Tzionas and Gall.

**Error Measures.** On this dataset, two error metrics similar to the ones used for *IJCV hand-object interactions* (see Section 6.5) are employed. Ground-truth annotations are provided for the outermost joint of the thumb and the index finger as depicted in Figure 6.10. Therefore, the corresponding 2D and 3D error measures consider only these two fingers instead of all fingers as introduced in Section 6.5.

The *ICCV in-hand scanning* dataset consists of four sequences where a large bottle, a bowling pin, a small bottle and a ball are turned by a hand in front of a camera. In addition, the hand motion tracked by the hand tracker used by Tzionas and Gall for in-hand scanning is available. A quantitative comparison with this hand tracker is presented in Figure 6.11. Note that they use the output from the hand-only tracker as additional constraints to a KinectFusion-style approach [36, 37] to reconstruct the shape of the object. Therefore, they do not have a joint reconstruction of hand and object motion to show in a qualitative comparison. Similar to the results on the *IJCV hand-object interactions* dataset, the quantitative discrepancy between the trackers is larger in the 2D metric. The method from this thesis — that runs at real-time frame rates — achieves an average error of **8.67 px** while the hand tracker of Tzionas and Gall improves on this result with **5.42 px** while running offline. Considering the 3D error measure, the results are almost interchangeably close given the noise in the sensor and the annotations: **10.92 mm** for this work and **10.20 mm** for Tzionas and Gall.

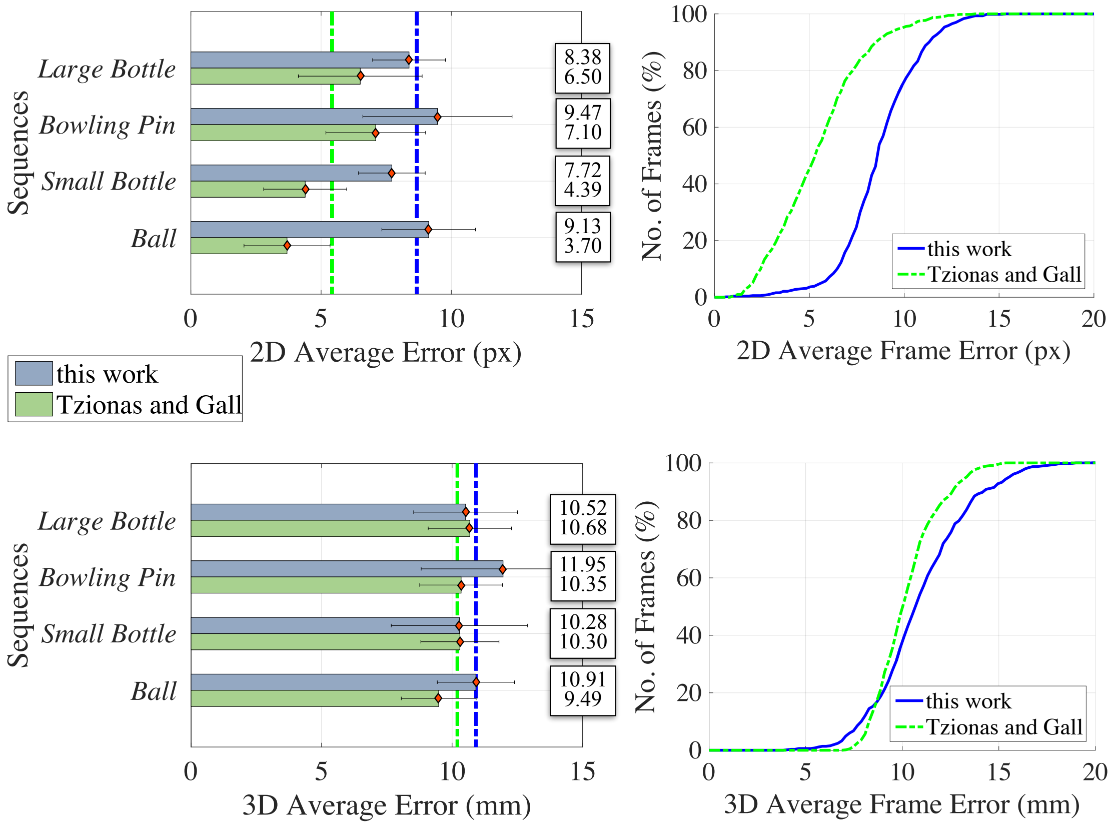
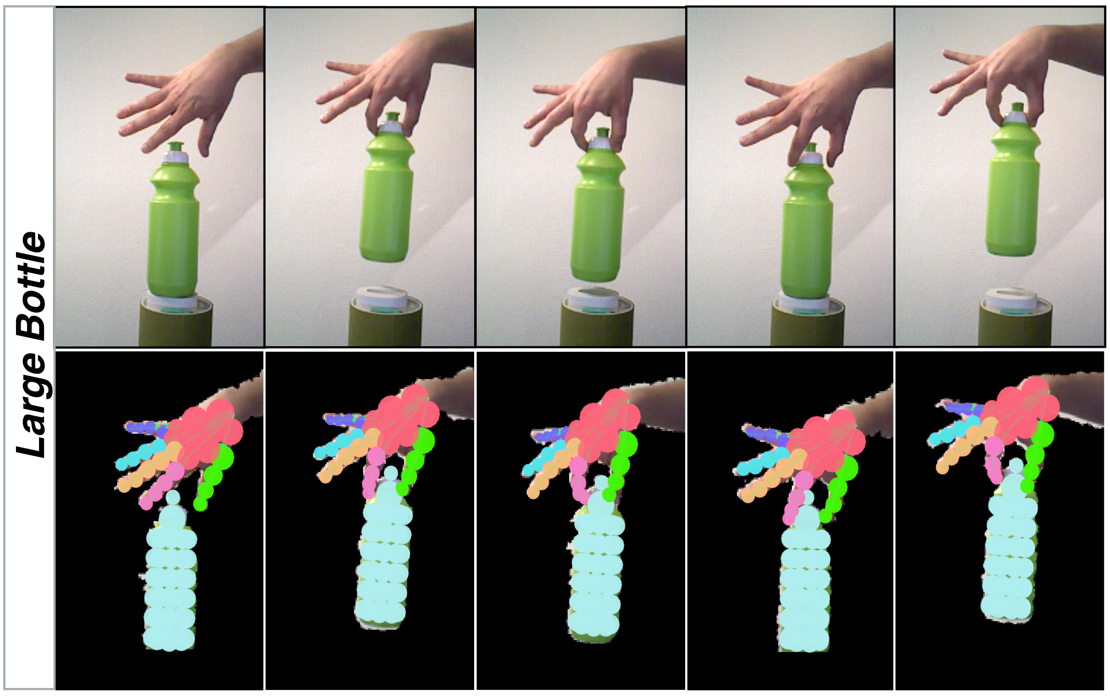
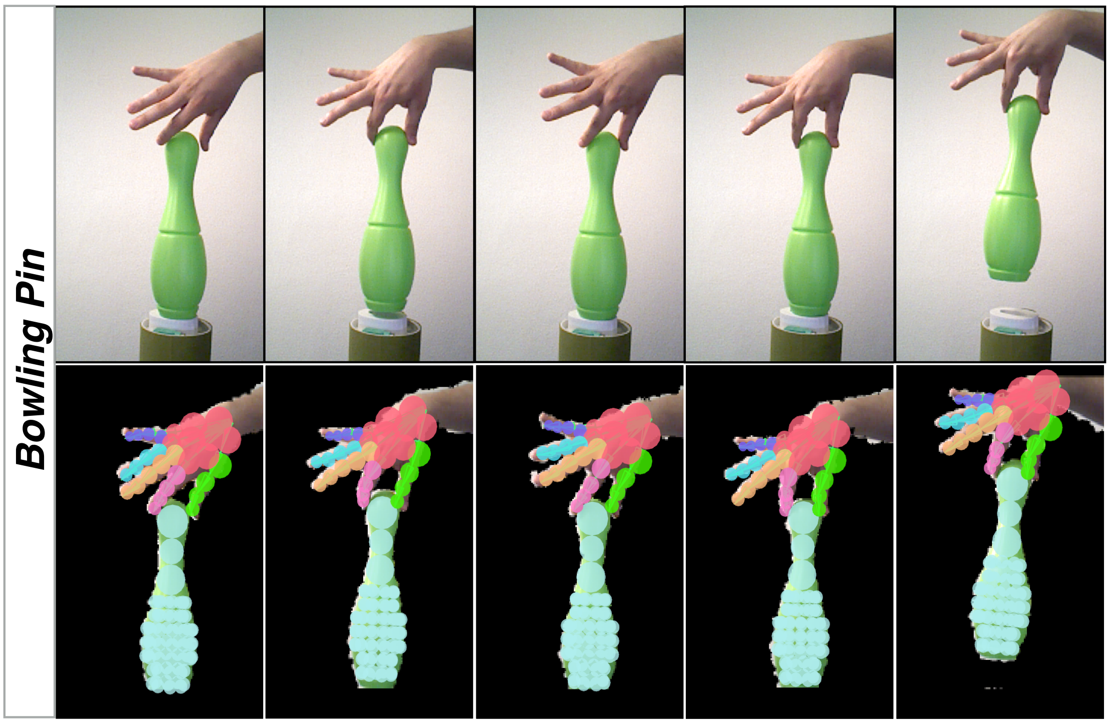


FIGURE 6.11: **Left:** Average error per sequence for both methods using the 2D (top) and the 3D (bottom) error metric. The exact errors in numbers are provided on the right. In the 2D metric, the offline approach of Tzionas and Gall achieves a lower error on all four sequences. But similar and even equal results are obtained when using the 3D measure. When comparing the performance of [3] relative to this work in both metrics, one can see that it is significantly worse in 3D, especially on the *Small Bottle* and the *Ball* sequence. For *Small Bottle*, this work performs slightly better in 3D and for *Ball*, the errors are at least a lot closer. **Right:** Consistency plots for both methods over all four sequences. In the 2D metric (top), the curve of Tzionas and Gall is always above. They achieve an error $< 5$ px on a larger percental amount of frames. Both approaches track all frames with less than 15 px error. When the 3D error measure is used (bottom), the curve of this work is above until ca. 8 mm what indicates that more frames are tracked very accurately. For both methods, the error is $< 20$ mm for all frames.

Figure 6.12 shows qualitative results on the four sequences. These results demonstrate that also objects with more complex shapes can be modeled in the Gaussian representation and can be tracked by the method presented in this thesis.


Large Bottle


Bowling Pin

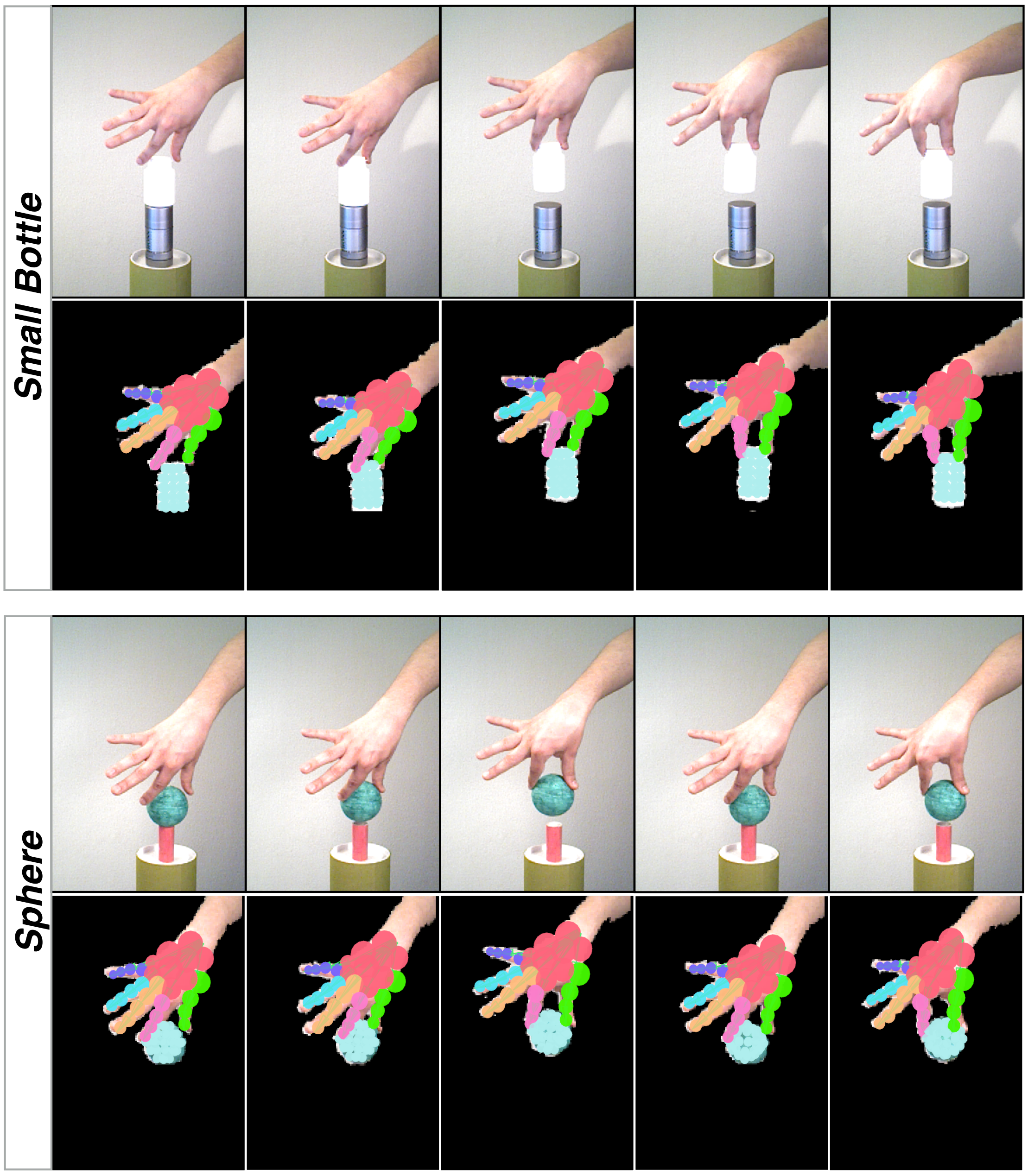


FIGURE 6.12: Qualitative results on the in-hand scanning sequences from Tzionas and Gall [3]. The top row shows color input frames and the bottom row tracking results in the Gaussian representation overlaid with the colored projection of the point cloud, respectively. Note that different grasps with thumb and index finger are accurately reconstructed by the tracker.

## 6.7 Evaluation of Algorithm Design

In this section, the importance and significance of different parts of the algorithm design and the energy are discussed. In a first step, the improvement provided by using 3D Gaussian mixture models is demonstrated. Afterwards, a detailed analysis of the energy is performed by incrementally enabling particular parts or using different variants of the terms.

### 6.7.1 Significance of 3D Formulation

In contrast to related hand-only tracking work by Sridhar *et al.* [1] that uses a 2.5D Gaussian mixture representation for the input and the evaluation of the energy, this work employs 3D Gaussians. Whereas the 3D Gaussians are smooth and continuous over the whole 3D space, the 2.5D representation is not: the 2D Gaussians are only continuous and smooth in a two-dimensional subspace at a certain depth but do not have any spatial extent in the third dimension.

As one can see from Appendix A, the spatial alignment term $E_a$ (Equation 5.1) is equivalent to the overlap and the collision term used by Sridhar *et al.* [1] up to a linear transform. To determine the improvement provided by the 3D representation, the performance of the 3D approach is evaluated on the hand-only dataset *Dexter* [4]. Therefore, all object-specific terms ($E_c$, $E_o$ and $E_i$) are disabled so that the remaining energy terms are a subset of the terms used by Sridhar *et al.* [1].

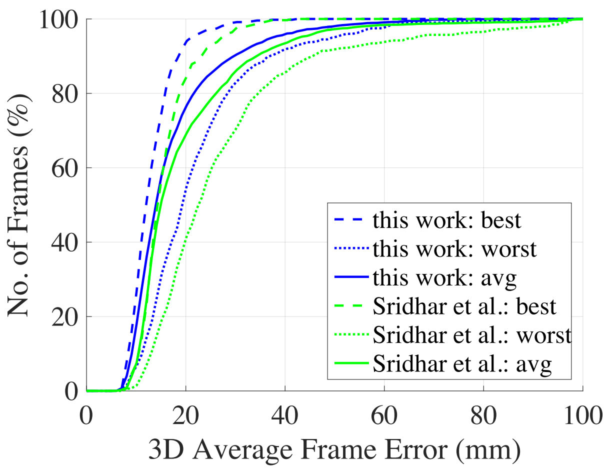


Figure 6.13: The consistency curves for the 3D formulation (blue) lie above the ones for Sridhar *et al.* [1] (green) for the best and the worst sequence and on average. This means that for more frames a lower error has been achieved. Until a threshold of ca. 15 mm, the average curve for this work lies above the best curve of Sridhar *et al.* This indicates that even on average over all sequences, the 3D formulation obtained more frames with an error $< 15$ mm than the 2.5D formulation on its best sequence.

The quantitative results demonstrate an improvement from **19.6 mm** to **17.1 mm** average fingertip error when using the approach from this thesis. The 3D formulation yields a lower error on 4 out of 7 sequences and has a higher percentage of frames with low error. The runtime for the hand-only version (see Section 6.2) is equal to the runtime of Sridhar *et al.* More detailed results and consistency plots are shown in Figures 6.13 and 6.14.

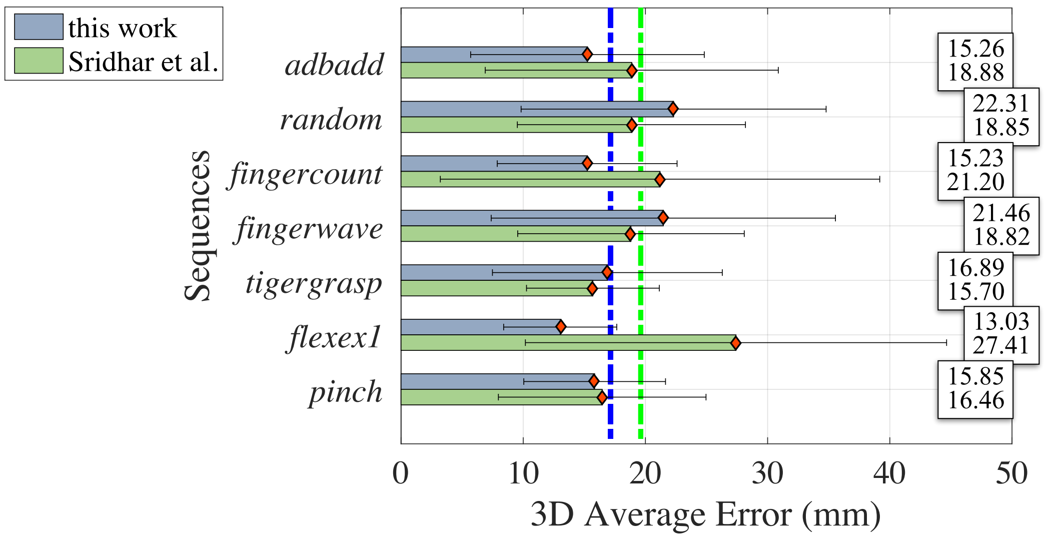


FIGURE 6.14: The average fingertip error for each sequence together with the standard deviation. The exact errors per sequence are displayed in the boxes on the right. The dark blue and green lines indicate the average error over all sequences for the 3D formulation and the approach from Sridhar *et al.* [1], respectively. The first outperforms the latter on 4 out of 7 sequences. The improvement is largest on *flexex1*, where the error is roughly cut in half.

### 6.7.2 Effect of Different Energy Parts

Apart from the 3D Gaussian mixture model formulation, this work introduces a new multi-layered classification pipeline for the object and hand parts and various new energy components specific to the hand and object tracking scenario. To determine the influence of each part, the performance of the tracker is evaluated in six configurations where steps of the algorithm are added incrementally:

1. *Depth Only*: single proposal that optimizes spatial alignment, anatomical plausibility and temporal smoothness.

2. *without Viewpoints*: second proposal in addition to *Depth Only* that uses spatial alignment, label alignment and anatomical plausibility. The discriminative part information comes from a single RDF that is trained on data from all viewpoints.

3. *with Viewpoints*: same energy as *without Viewpoints* but the discriminative part information comes from the full classification pipeline as discussed in Section 4.1.

4. *with Object Terms*: two proposals as in *with Viewpoints*. In addition, the depth proposal uses the contact points and the object occlusion term.

5. *with Propagated Object Terms*: same energy as in *with Object Terms* but the propagated versions of the contact points and the object occlusion term are optimized.

6. *with Interpenetration Avoidance*: two proposals as in *with Propagated Object Terms*. Additionally, both proposals use the interpenetration avoidance term.

All versions are quantitatively evaluated on the *Dexter++* dataset. The results in Figure 6.15 show that there is a significant improvement from variant 1 to 3 and from 2 to 3. Afterwards, the changes are negligible due to sensor noise and uncertainty of manual annotation. Interestingly, using no discriminative part information at all performs better on average than using a single RDF without viewpoint selection.

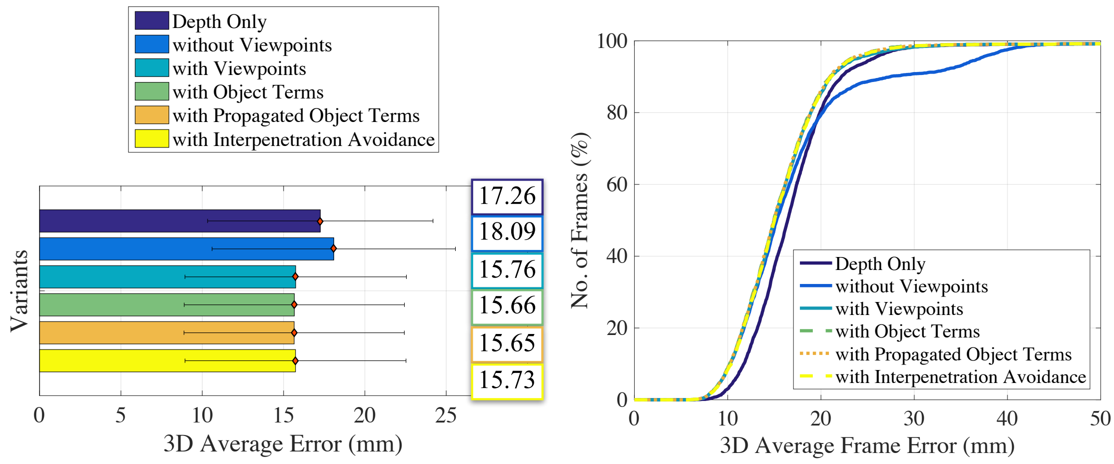


FIGURE 6.15: **Left:** The average error over all sequences from *Dexter++* for the six discussed variants. The exact numbers are displayed on the right. Whereas the error is higher for *Depth Only* and *without Viewpoints*, there is no significant quantitative change when adding contact points, object occlusion and interpenetration avoidance. **Right:** Consistency plots for all six presented versions. Similar to the average errors (left), there is no significant difference between the variants 3-6. The curves for *Depth Only* and *without Viewpoints* are worse. The first achieves a lower percentage of frames for all thresholds. The latter has a similar amount of low error frames (< 15 mm), but medium error frames (20–30 mm) are lacking.

Although the difference in numbers is small for some configurations, the visual quality and plausibility differs noticeably. The remainder of this section shows qualitative comparisons of different variants of the approach that illustrate the advantages of additional terms.

**Significance of Label Alignment Term $E_l$ and Viewpoint Selection.** As already shown in related work (see Chapter 2), combining discriminative pose estimation and generative tracking approaches can improve robustness, accuracy and the ability to recover from tracking failures. Figure 6.16 shows results on the *Pinch* sequence from the *Dexter++* dataset tracked with variants *Depth Only* (1) and *with Viewpoints* (3). For the latter, the pose is tracked correctly whereas tracking partially fails for the first variant and does not recover afterwards.

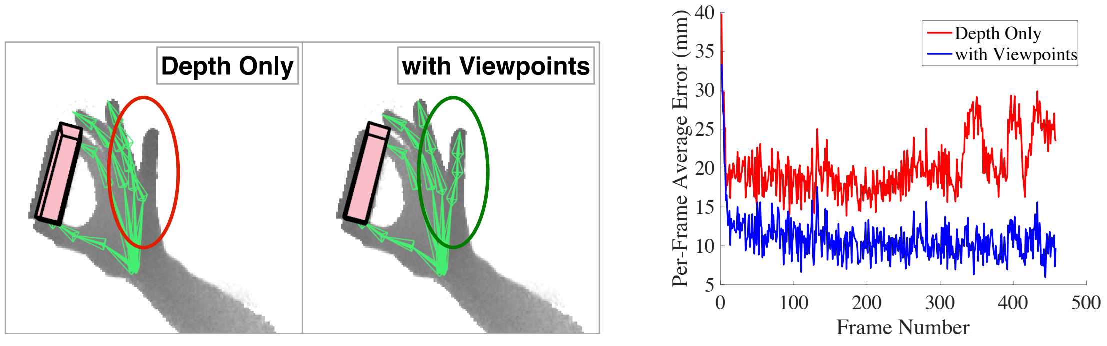


FIGURE 6.16: **Left:** Visual comparison between *Depth Only* and *with Viewpoints*. For the first, the tracking of the little finger is lost while the latter variant tracks correctly. **Right:** The per-frame average error (y-axis) is plotted for the frames of the *Pinch* sequence (x-axis). Note that the error for *Depth Only* (red) remains high over the whole sequence in comparison to *with Viewpoints* (blue) which indicates that the tracker does not recover from the failure.

In contrast to recent work, *e. g.* [1], which uses a single random decision forest trained on data from multiple viewpoints, the method in this thesis selects one out of four specifically trained random decision forests based on the best-matching viewpoint. This procedure reduces the classification complexity in each forest and allows to train on more data overall. Figures 6.17 and 6.18 show the quantitative and qualitative improvements of using multiple forests and viewpoint selection compared to a single forest combining all viewpoints.

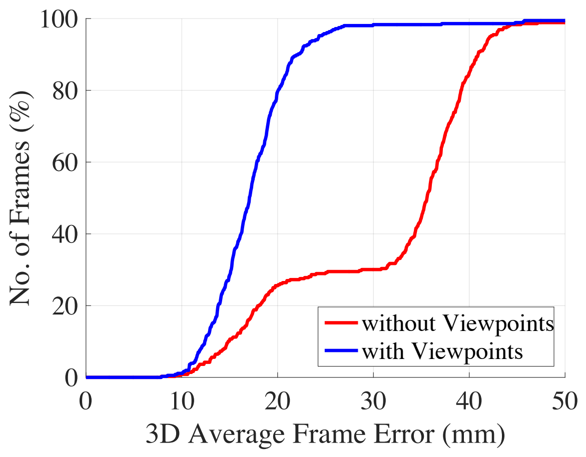


FIGURE 6.17: Consistency plots for the variants *with Viewpoints* and *without Viewpoints* on the *Grasp1* sequence. While the first achieves an error lower than 20 mm on 80% of the frames, the latter obtains such a low error only for roughly 27% of the frames. Apart from these frames, there are a lot of frames with error $>$ 30 mm as one can see from the steep increase (from ca. 30% to more than 80%) between 30 mm and 40 mm.

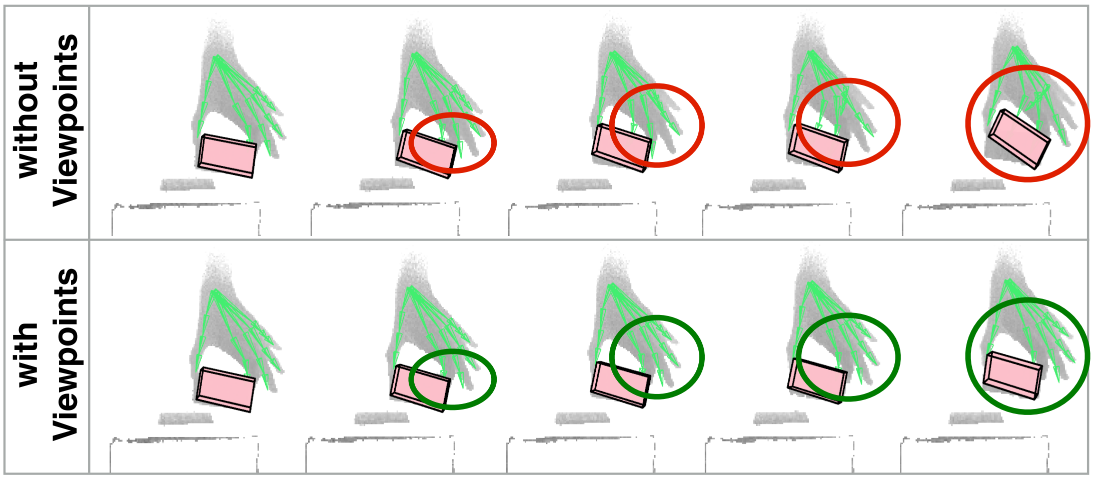


FIGURE 6.18: Comparison of the variant *without Viewpoints* (top) and *with Viewpoints* (bottom) on the *Grasp1* sequence from the *Dexter++* dataset. When employing only a single RDF that is trained on multiple views, the tracking fails for all fingers (except the thumb) and in the end also for the object. The quality of the tracking is better when using viewpoint selection and multiple RDFs trained on distinct viewpoints.

**Significance of Object-Specific Terms $E_c$ and $E_o$.** When adding the object specific terms for contact points and object occlusion handling, the quantitative error does not improve significantly.

Nevertheless, these two terms ensure qualitatively better and more plausible results in situations that occur frequently. For a more detailed evaluation, the version *with Object Terms* is further split here: Figure 6.19 shows results when only the contact points term $E_c$ is added and the improvement provided by only adding the object occlusion term $E_o$ is presented in Figure 6.20. The contact points term increases the stability of the object while it is held in the hand. The object occlusion term keeps occluded fingers stable whereas they move nondeterministically in case of missing data if this term is not used.

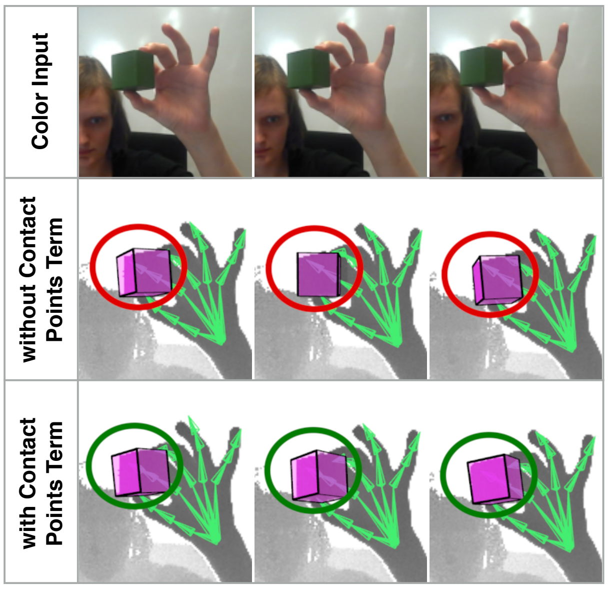


FIGURE 6.19: From the color input (top), one can see that the *Cube* is held statically in the hand. Tracking results for the version *with Viewpoints* (middle row) and *with Object Terms* where only $E_c$ is added (bottom row) are shown overlaid on the depth input. When using the contact points term $E_c$, the orientation of the object is more stable.

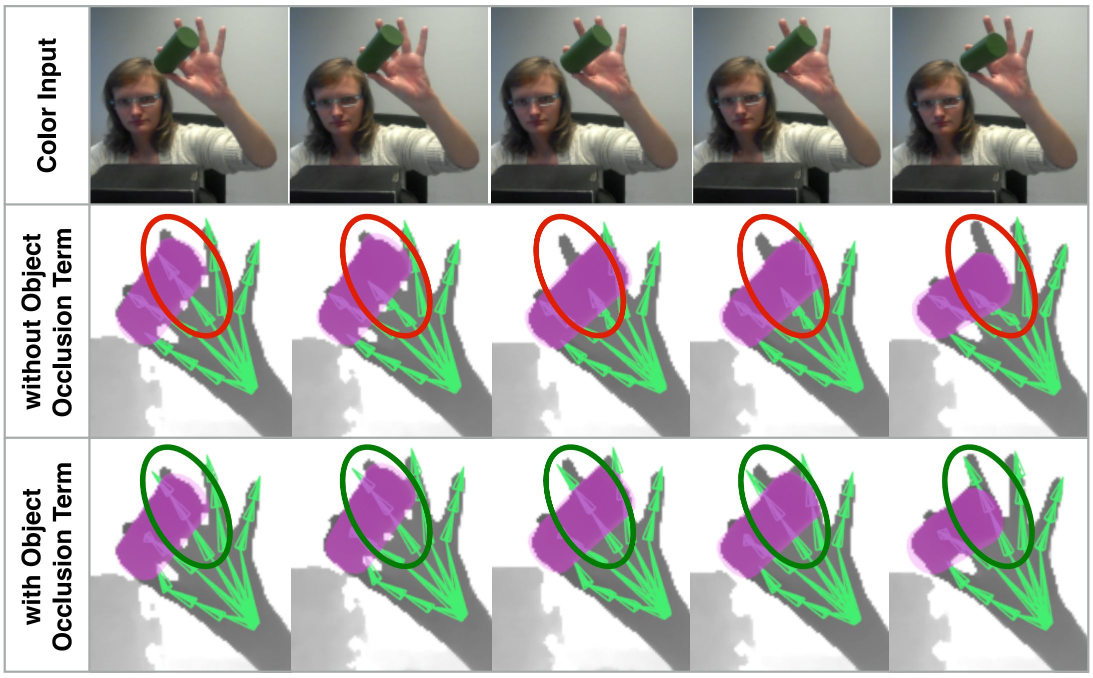


FIGURE 6.20: Comparison of the versions *with Viewpoints* (middle row) and *with Object Terms* with only $E_o$ added (bottom row) on the *Cylinder* sequence. The color input (top row) is provided for reference and the tracking results are overlaid on the depth input. While the middle finger remains stable under occlusion when the object occlusion term is enabled, it moves implausibly without this term.

In comparison to these results, the difference between the propagated version and the non-propagated version of the contact points term and the object occlusion term is more subtle. Because the same reference value — either the contact point or the saved value for the occluded DOF — is usually kept for a longer period of time, temporal jitter is further removed. This adds additional stability to the fingers and the object. Figure 6.21 shows a qualitative comparison that illustrates this difference between the variants *with Viewpoints*, *with Object Terms* and *with Propagated Object Terms*.

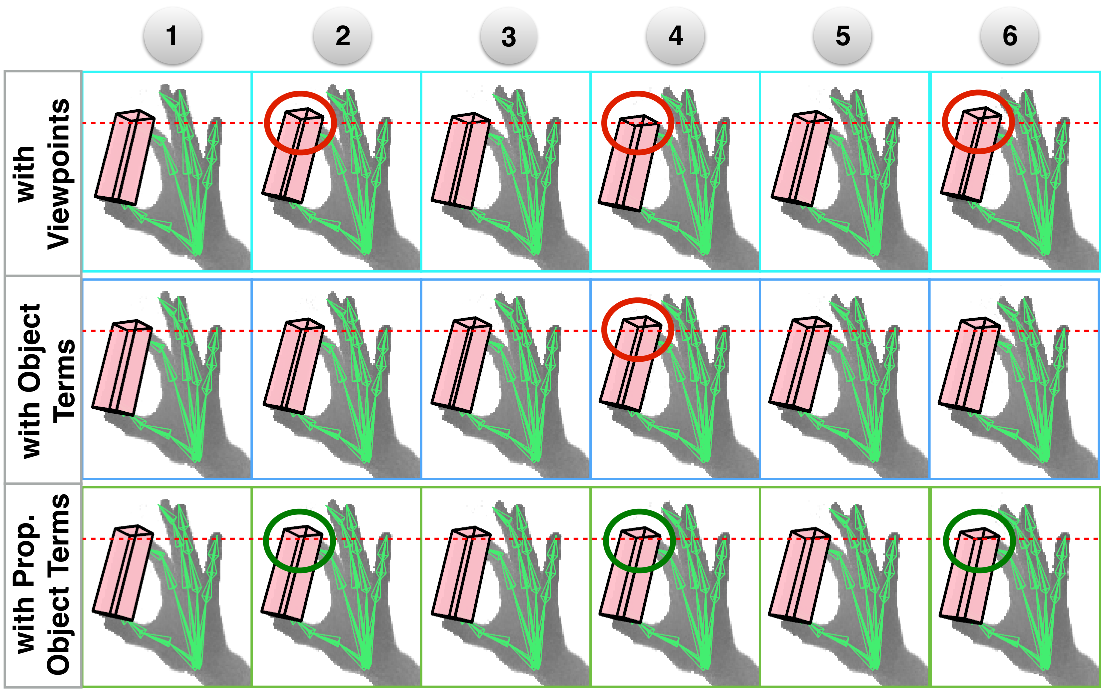


FIGURE 6.21: Analysis of tracking stability for six consecutive frames of the *Pinch* sequence where the cuboid is not moved. The tracking results are overlaid on the depth image. The red dashed line indicates the constant position of the cuboid for reference. When using no object terms at all (top row, cyan boxes) the cuboid moves noticeably above (frames 2 and 6) and below the line (frame 4). With the non-propagated object terms (middle row, blue boxes), such an error happens only once (frame 4). The propagated versions of the contact points and the object occlusion term resolve this jitter entirely.

**Significance of Interpenetration Avoidance Term $E_i$.** Although the use of the interpenetration avoidance term $E_i$ does not significantly improve the average error, it helps to produce more plausible tracking results. As demonstrated on the *Grasp2* sequence in Figure 6.22, it can happen that fingers interpenetrate the object when $E_i$ is missing. This is physically impossible and running the tracker with all steps enabled, the version *with Interpenetration Avoidance*, avoids such results.

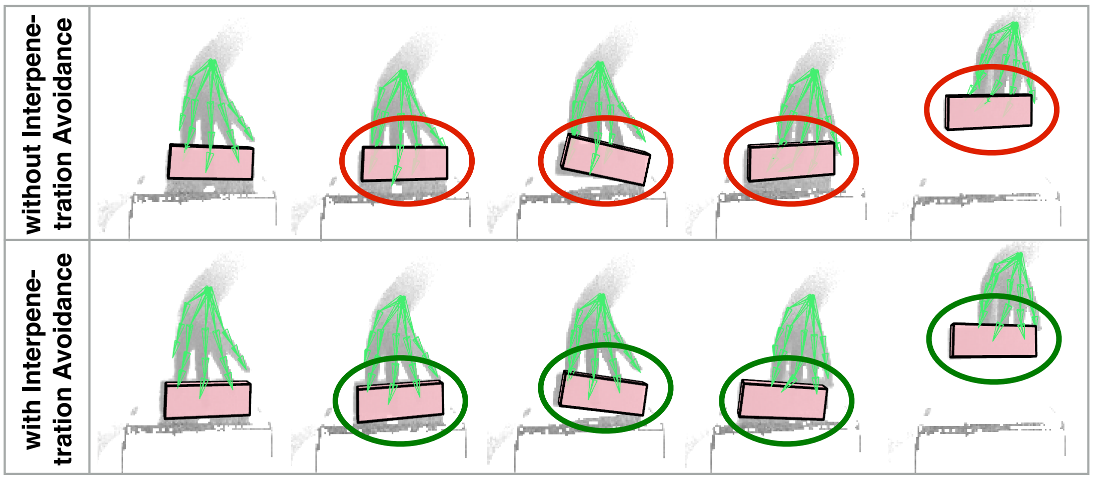

without Interpene-tration Avoidance
with Interpene-tration Avoidance

FIGURE 6.22: Qualitative comparison of the variants *with Propagated Object Terms* (top) and *with Interpenetration Avoidance* (bottom) on *Grasp2*. When the cuboid is grasped in the first variant, the fingers get stuck inside the object. Using interpenetration avoidance between the hand and the object leads to a plausible reconstruction where the fingers are lying on the surface of the cuboid.

## 6.8 Further Qualitative Evaluation

This section demonstrates more qualitative results from simultaneous tracking of a hand and an object using the approach presented in this thesis.

Figures 6.23 and 6.24 show tracking results on the additional captured sequences *Complex* and *ThrowCube*. These illustrate that the method is capable of reconstructing fast and complex interactions. The real-time runtime allows to show live captures like in Figure 6.26 where the tracking result is rendered on the screen while the motion is performed.

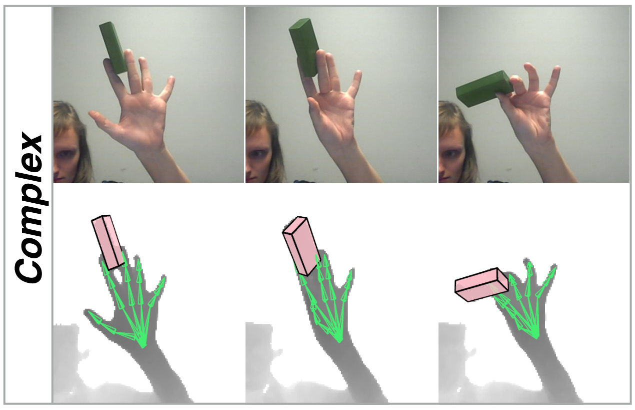


FIGURE 6.23: Qualitative results on the *Complex* sequence. Even unusual motions can be tracked since the framework is generative.

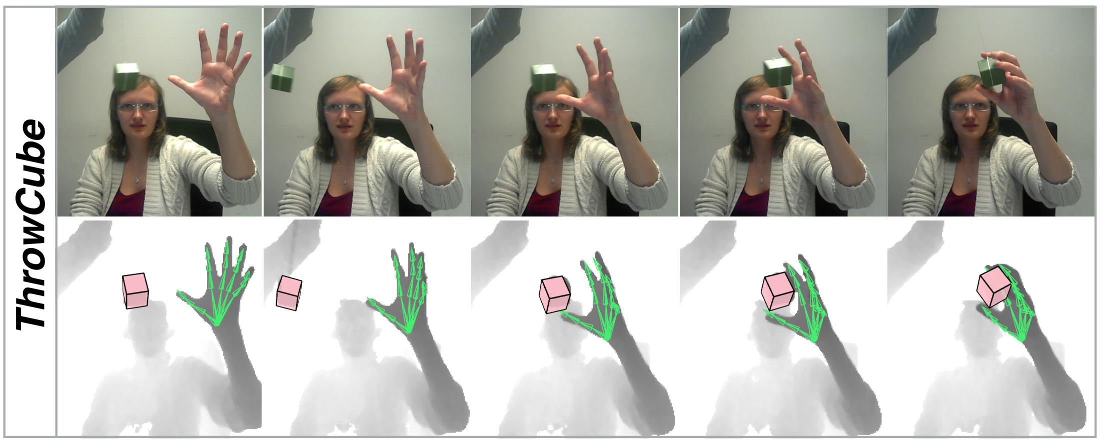


FIGURE 6.24: Qualitative results on the *ThrowCube* sequence. Note that the cube is moving so fast that motion blur is present in the input. Nevertheless, the rigid transform of the cube is reconstructed. Also the catching hand is tracked without interpenetrating the object. A natural grasp with physically plausible hand-object contacts is reproduced.

Tracking results from an egocentric viewpoint are shown in Figure 6.25. This setup is especially relevant for augmented or virtual reality applications. The results in Figure 6.26 exhibit several achievements of the approach like tracking different users and objects with different colors, tracking in the presence of multiple objects in the foreground, as well as generalization to track more than one object.

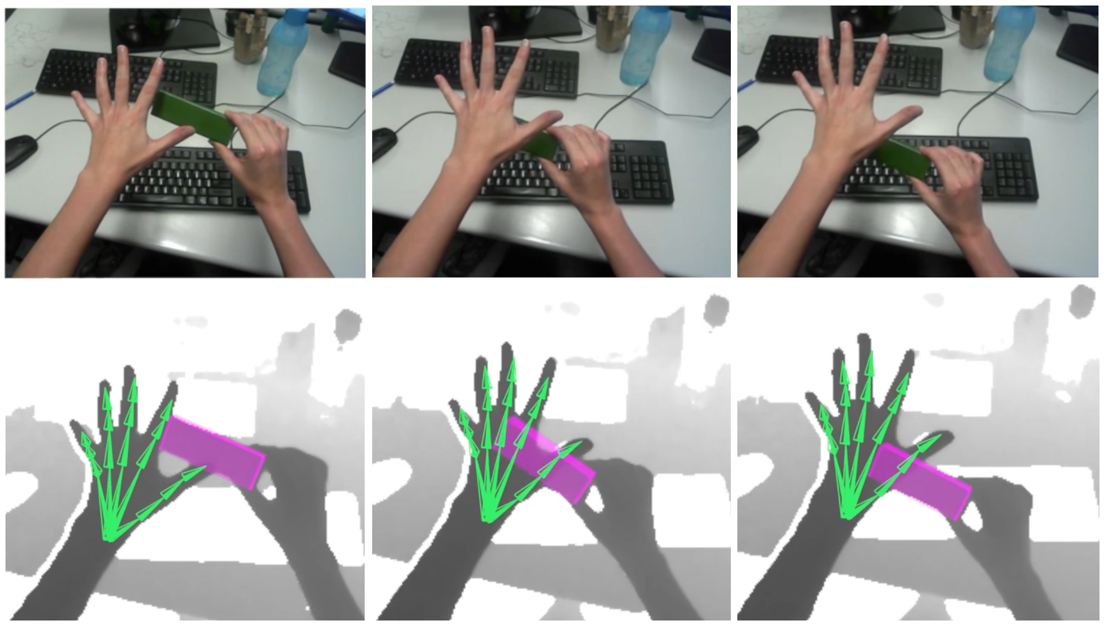


FIGURE 6.25: Tracking results from an egocentric viewpoint. The tracking is stable despite the cluttered background.

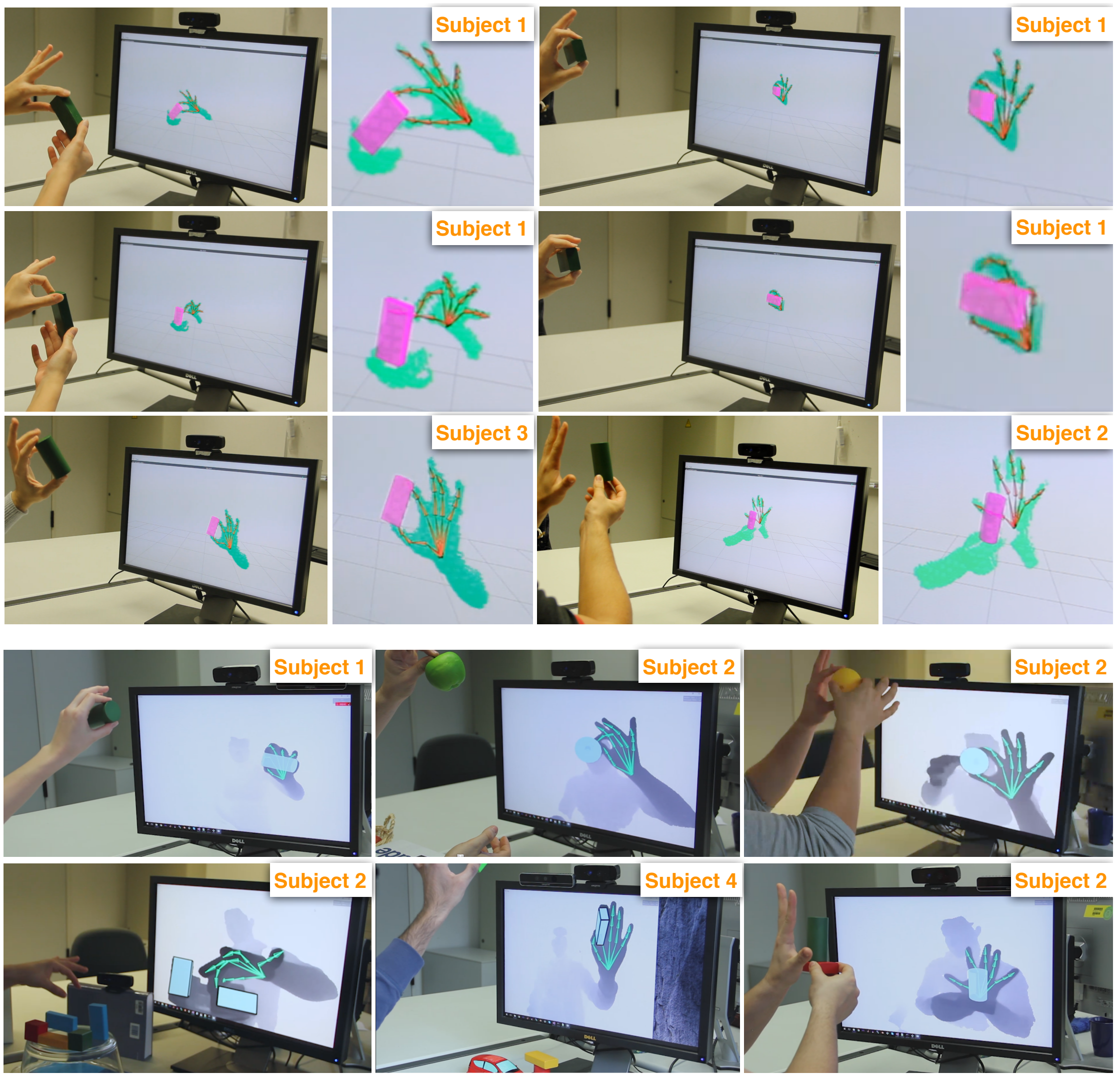


FIGURE 6.26: Live capture results from four different subjects (1 female, 3 male). The results show stability under severe occlusions, robustness to the presence of multiple hands or objects in the foreground, the ability to track differently colored and shaped objects and generalization to track multiple objects (bottom left). **Top:** The real-time reconstruction of the motion is displayed on the screen and also in a zoom on the right in pink (object) and orange (hand). The 3D Gaussian representation of the input is shown in green. **Bottom:** The tracking result in green (hand) and light blue (object) is presented overlaid on the depth input on the screen.

## 6.9 Limitations

The presented approach achieves robust and accurate simultaneous tracking of a hand and an object in interaction in many scenarios, although it also has limitations:

- **Motions under occlusion**:
  While prior terms like the object occlusion term $E_o$ enable the method to deal with globally rigid motion of the occluded area, tracking can fail if the assumption of rigidity is violated. The discriminative part information often ensures recovery as soon as the previously occluded part is visible again. Better priors for motion under occlusion, for example priors that consider the relation between different fingers, could lead to improvement.

- **Very fast motion**:
  Since several parts of the algorithm use the previously tracked pose, *e. g.* for initialization of the pose optimization or for the computation of visibility weights, temporal coherence between frames is important. If motions are performed too fast, tracking might fail, but is usually able to recover when the movement slows down. Faster cameras together with a faster implementation of the main algorithm (*e. g.* using a GPU) increase temporal coherence in the input.

- **Failure of color segmentation**:
  The simple HSV color segmentation can at least partially fail when used for example with specular objects. Since the main data term $E_a$ (see Equation 5.1) relies on a good hand-object segmentation in the input, tracking quality decreases. A more sophisticated segmentation approach could help to overcome this limitation.

- **Very complex objects**:
  The results in this chapter demonstrate the ability to track objects like a bowling pin, and in theory every object can be modeled given an unlimited number of Gaussians. Nevertheless, the number of model Gaussians influences the processing time per frame. All objects tracked in the evaluation consist of at most 72 Gaussians. Tracking of an object that is modeled using several hundreds of Gaussians is currently not possible in real time. Parallelization on the GPU can help to reduce the processing time.

Figure 6.27 demonstrates situations where the tracking partially fails due to different reasons.

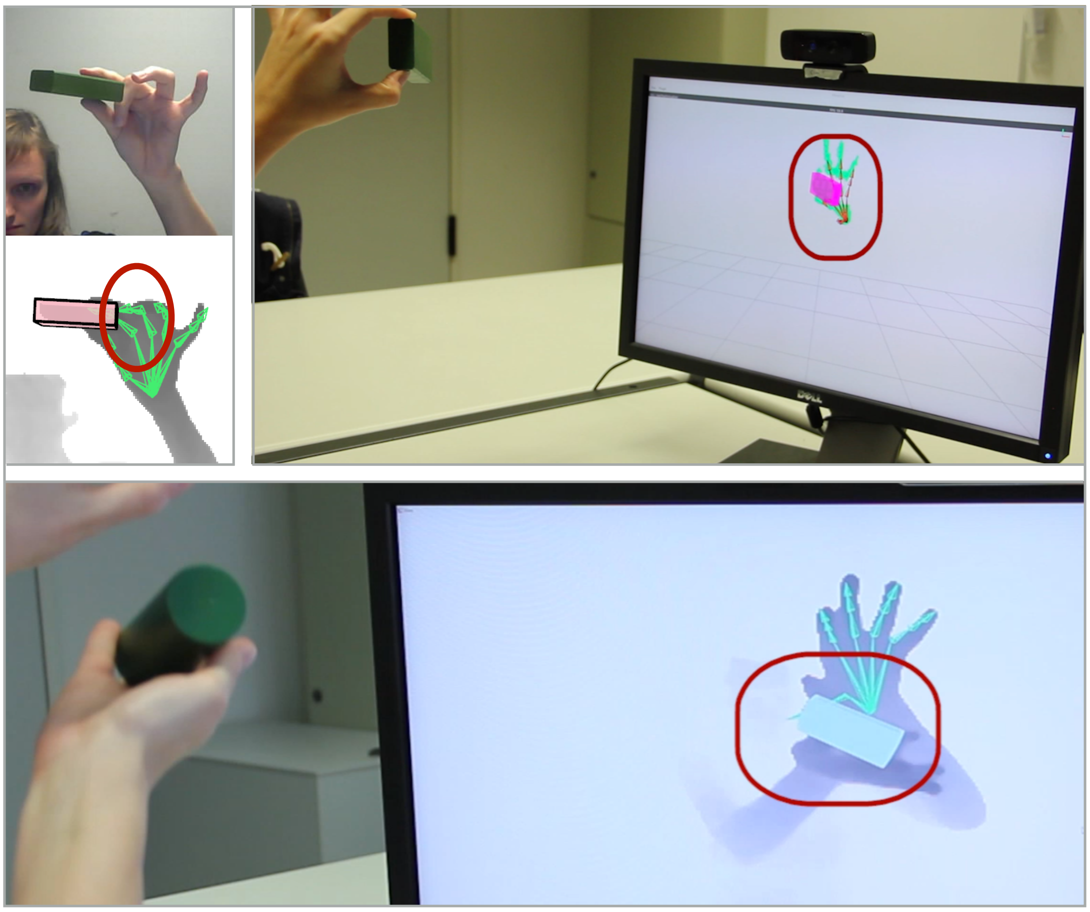


Figure 6.27: **Top left:** The middle finger has performed a complex motion while being partially occluded by the cuboid. The tracking was lost when the finger became occluded. **Top right:** While the tracking is stable in many occlusion situations, it can fail when important parts like the palm become fully occluded. **Bottom:** Whereas a second hand in the input is in general not a problem for the approach, single fingers might be wrongly attracted if they come too close to the other hand for a longer period of time.

# Chapter 7

# Conclusion and Future Work

This thesis presented an approach for simultaneous reconstruction of the motion of a hand and an object in interaction. The method uses input from a single commodity RGB-D sensor, works for objects of different shape, size and color and is the first to run at real-time frame rates. It generalizes to different users and camera locations while even allowing tracking of more than one object. Extensive evaluations showed comparable performance to state-of-the-art offline approaches, robust tracking on a new annotated hand and object dataset and live capture results. The experiments concerning algorithm design verified the effect of different components and illustrated the improvement by exploiting hand-object specific prior knowledge.

There are many possibilities for future work based on this thesis. One idea is to experiment with other machine learning techniques, like convolutional neural networks or boosting, to improve the accuracy of the classification. Additionally, training a classifier for the hand-object segmentation task might be useful as color segmentation is inherently dependent on lighting conditions and is error-prone when using non-diffuse objects. The optimization framework in this thesis already works with two proposals from which one is independent of the classification result to avoid failures due to bad classification. Furthermore, a quality measure for the classification result could be constructed to lower the influence of the corresponding energy term if the quality is detected to be bad. Such quality measures could involve entropy or noise in the part label image or the number of clusters per class. Also learning such a measure from examples of good and bad classification should be possible in general.

Another related problem is the construction of a user-specific surface model of the hand. Given enough scans from human hands, a principal component analysis could yield a low-dimensional subspace that captures most of the hand variations. For each user, the hand shape can then be optimized in this subspace to obtain a model that is more accurate than the 3D Gaussian approximation. Having such a user-adapted surface hand model could improve hand and object tracking because some energy terms — like contact points and interpenetration avoidance — are dependent on the location of the surface of the hand. While the current Gaussian model offers simple scaling, it is far from the accuracy of a proper surface model.

A natural extension of this work is the tracking of hands manipulating an articulated or a deformable object. This task is more challenging since there is a higher number of degrees of freedom for the object. Moreover, the Gaussian representation in the current form is too coarse to accurately track deformations. A surface or a volumetric representation like a mesh or a signed distance function have already been successfully used for deformable objects (*e. g.* by Zollhöfer *et al.* [18]).

# Appendix A

# Data Term Equivalence

In contrast to Sridhar *et al.* [1] where only 2D Gaussian mixture models with extra depth information (so called 2.5D GMMs) are used, this work uses a 3D Gaussian mixture representation for the input and hence also in the energy. Despite this difference, it can be shown that the main data term — the spatial alignment term $E_a$ — in general is equivalent to parts of the objective of [1] up to a constant and an additional factor. In the following, only the hand part of $E_a$ is considered since Sridhar *et al.* only track a single hand. Simple computations lead to a sum consisting of three parts:

$$\begin{aligned}
&\int_\Omega (\mathcal{I}_h(\mathbf{x}) - \mathcal{M}_h(\Theta, \mathbf{x}))^2 \, d\mathbf{x} \\
&= \int_\Omega \mathcal{I}_h^2(\mathbf{x}) \, d\mathbf{x} - 2 \cdot \int_\Omega \mathcal{I}_h(\mathbf{x}) \cdot \mathcal{M}_h(\Theta, \mathbf{x}) \, d\mathbf{x} + \int_\Omega \mathcal{M}_h^2(\Theta, \mathbf{x}) \, d\mathbf{x} \\
&= \int_\Omega \mathcal{I}_h^2(\mathbf{x}) \, d\mathbf{x} - 2 \cdot \underbrace{\sum_{i=1}^{N_\mathcal{I}} \sum_{j=1}^{N_\mathcal{M}} \int_\Omega G_i(\mathbf{x}) \upsilon_j G_j(\mathbf{x}) \, d\mathbf{x}}_{\text{model-image-part}} + \underbrace{\sum_{i=1}^{N_\mathcal{M}} \sum_{j=1}^{N_\mathcal{M}} \int_\Omega \upsilon_i G_i(\mathbf{x}) \upsilon_j G_j(\mathbf{x}) \, d\mathbf{x}}_{\text{model-model-part}} \quad ,
\end{aligned}$$

where the parameterization of the Gaussians is dropped for better readability. The first term is a constant w.r.t. $\Theta$, *i. e.* not relevant in the pose optimization. The second term, the model-image-part, is up to the normalization constant $\frac{1}{E(\mathcal{C}_I, \mathcal{C}_I)}$, the visibility weight $\upsilon_i$ and the distance cutoff factor $\Delta(p, q)$ equivalent to the depth similarity term from

Sridhar *et al.*

$$E_{sim}(\mathcal{C}_p, \mathcal{C}_I) = \frac{1}{E(\mathcal{C}_I, \mathcal{C}_I)} \sum_{p \in \mathcal{C}_p} \sum_{q \in \mathcal{C}_I} \Delta(p, q) \int_{\Omega} G_p(\mathbf{x}) G_q(\mathbf{x}) \, d\mathbf{x} \quad ,$$

where $\mathcal{C}_I$ and $\mathcal{C}_p$ are the 2.5D GMM for the input and the projected model, respectively.

The third term is again up to a normalization constant $\frac{1}{E(\mathcal{C}_h, \mathcal{C}_h)}$ and the visibility weights $v_i$ equivalent to a term from Sridhar *et al.* , the collision penalty

$$E_{col}(\Theta) = \frac{1}{E(\mathcal{C}_h, \mathcal{C}_h)} \sum_{p \in \mathcal{C}_h} \sum_{\substack{q \in \mathcal{C}_h, \\ q > p}} \int_{\Omega} G_p(\mathbf{x}) G_q(\mathbf{x}) \, d\mathbf{x} \quad ,$$

where $\mathcal{C}_h$ is the 3D GMM for the model. Note that $E_{col}$ cosiders every pair of two model Gaussians only once and does not consider the pair $(p, p)$ since this is a constant and not depending on $\Theta$ anyway. This introduces additionally an implicit factor 2 and a constant between $E_{col}$ and the model-model-part of $E_a$.

In total, the there is a strong relation between the spatial alignment term $E_a$ and the similarity and collision term from Sridhar *et al.* [1]. Up to several constants and constant factors as well as the use of the 2.5D representation in $E_{sim}$, they can be seen as equivalent.

# Appendix B

# Gradients

An advantage of the GMM representation is that the gradients of the energy terms w.r.t. the pose parameters $\Theta$ can be computed analytically. This makes the gradient-based optimization fast and enables the tracking algorithm to run in real time. In this chapter, an analytical expression for the gradient of each of the energy terms as introduced in Section 5.2 is derived.

## B.1 Spatial Alignment Term $E_a$

The differentiation for the two summands of $E_a$ (see Equation 5.1) works completely analogous. The derivation is shown here for the hand part

$$\int\limits_{\Omega} (\mathcal{I}_h(\mathbf{x}) - \mathcal{M}_h(\Theta, \mathbf{x}))^2 \, d\mathbf{x} \quad .$$

As derived in Appendix A, this is equal to

$$\int\limits_{\Omega} \mathcal{I}_h^2(\mathbf{x}) \, d\mathbf{x} - 2 \cdot \underbrace{\int\limits_{\Omega} \mathcal{I}_h(\mathbf{x}) \cdot \mathcal{M}_h(\Theta, \mathbf{x}) \, d\mathbf{x}}_{\text{model-image-part}} + \underbrace{\int\limits_{\Omega} \mathcal{M}_h^2(\Theta, \mathbf{x}) \, d\mathbf{x}}_{\text{model-model-part}} \quad .$$

The first summand here is independent of the pose parameters $\Theta$, so it will vanish in the derivative. Both, the model-image-part and the model-model-part are integrals over a product of weighted Gaussian mixtures. In the next step, the derivative of such a term in general,

$$\int_\Omega \mathcal{G}_A(\mathbf{x}) \cdot \mathcal{G}_B(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} w_i^A w_j^B \underbrace{\int_\Omega G_i^A(\mathbf{x}; \boldsymbol{\mu}_i, \sigma_i) \cdot G_j^B(\mathbf{x}; \boldsymbol{\mu}_j, \sigma_j) d\mathbf{x}}_{=: \mathcal{S}_{i,j}} \quad ,$$

is computed. This boils down to finding an analytical expression for the derivative of $\mathcal{S}_{i,j}$ since the weights are assumed to be constant. Note that this holds for the visibility weights $v_i$ in the concrete case because they are set to a fixed value before the optimization for each frame starts. As explained in [38], an integral over a product of two un-normalized anisotropic Gaussians is given as:

$$\frac{\sqrt{(2\pi)^d |\Sigma_i \Sigma_j|}}{\sqrt{|\Sigma_i + \Sigma_j|}} \exp\left( -\frac{1}{2} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T (\Sigma_i + \Sigma_j)^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \right) \quad .$$

Here, the dimension $d = 3$ and the Gaussians are isotropic what leads to:

$$\mathcal{S}_{i,j} = \frac{(2\pi)^{\frac{3}{2}} (\sigma_i^2 \sigma_j^2)^{\frac{3}{2}}}{(\sigma_i^2 + \sigma_j^2)^{\frac{3}{2}}} \exp\left( -\frac{||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2^2}{2(\sigma_i^2 + \sigma_j^2)} \right)$$

Therefore, the $k$-th component of the gradient is given as:

$$\begin{aligned}
\frac{\partial \mathcal{S}_{i,j}}{\partial \theta_k} &= \frac{\partial \mathcal{S}_{i,j}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \theta_k} \qquad \text{where } \mathbf{z} := \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \\
&= \mathcal{S}_{i,j} \cdot \left( -\frac{\mathbf{z}}{\sigma_i^2 + \sigma_j^2} \right) \cdot \frac{\partial \mathbf{z}}{\partial \theta_k} \\
&= -\frac{(2\pi)^{\frac{3}{2}} (\sigma_i^2 \sigma_j^2)^{\frac{3}{2}}}{(\sigma_i^2 + \sigma_j^2)^{\frac{5}{2}}} \exp\left( -\frac{||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2^2}{2(\sigma_i^2 + \sigma_j^2)} \right) \cdot (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \cdot \underbrace{\frac{\partial (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}{\partial \theta_k}}_{\boldsymbol{\mu}\text{-difference derivative}}
\end{aligned}$$

Whereas the whole derivation for the model-image-part and the model-model-part of the energy term was the same until this point, the $\boldsymbol{\mu}$-difference derivative differs between them.

- model-image-part: only $\boldsymbol{\mu}_j$, that belongs to a Gaussian in the GMM $\mathcal{M}$ for the model, is influenced by the pose parameters $\Theta$. Therefore

$$\frac{\partial (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}{\partial \theta_k} = -\frac{\partial \boldsymbol{\mu}_j}{\partial \theta_k} \quad . \tag{B.1}$$

- model-model-part: here both, $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$, are influenced by the pose parameters $\Theta$. Hence

$$\frac{\partial(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}{\partial\,\theta_k} = \frac{\partial\,\boldsymbol{\mu}_i}{\partial\,\theta_k} - \frac{\partial\,\boldsymbol{\mu}_j}{\partial\,\theta_k} \quad . \tag{B.2}$$

Note that these simple $\boldsymbol{\mu}$-derivatives can be computed using the kinematic structure of the model. This yields the full gradient for the spatial alignment $E_a$.

## B.2 Label Alignment Term $E_l$

As a first step of finding the gradient for the label alignment term $E_l$ (see Equation 5.2), the derivative operator can be dragged inside the summation. For the computation of the gradient, $\alpha_{i,j}$ is treated as a constant w.r.t. $\Theta$. Afterwards, the chain rule is applied.

$$\begin{aligned}\frac{\partial\,E_l(\Theta)}{\partial\,\theta_k} &= \sum_{i=1}^{N_{\mathcal{I}}}\sum_{j=1}^{N_{\mathcal{M}}} \alpha_{i,j} \cdot \frac{\partial\,||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2^2}{\partial\,\theta_k} \\ &= 2\sum_{i=1}^{N_{\mathcal{I}}}\sum_{j=1}^{N_{\mathcal{M}}} \alpha_{i,j} \cdot (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \cdot \underbrace{\frac{\partial(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}{\partial\,\theta_k}}_{\text{see Equation B.1}}\end{aligned}$$

The only part that still needs to be computed is again a $\boldsymbol{\mu}$-difference derivative. In this case, it is equal to the model-image version as provided in Equation B.1 because $E_l$ only considers pairs of one image and one model Gaussian.

## B.3 Anatomical Plausibility Regularizer $E_p$

Since the anatomical plausibility regularizer $E_p$ as given in Equation 5.3 is a sum over all DOFs in $\Theta$, only a single term remains in each component of the gradient.

$$\frac{\partial\,E_p(\Theta)}{\partial\,\theta_k} = \begin{cases} 0 & \text{if } \theta_k^l \leq \theta_k \leq \theta_k^u \\ 2(\theta_k - \theta_k^u) & \text{if } \theta_k > \theta_k^u \\ 2(\theta_k - \theta_k^l) & \text{if } \theta_k < \theta_k^l \end{cases}$$

## B.4 Temporal Smoothness Regularizer $E_t$

The gradient of the temporal smoothness regularizer $E_t$ is computed w.r.t. the approximation introduced in Equation 5.5.

$$\begin{aligned}\nabla E_t(\Theta^{(t)}) &= \nabla ||\Theta^{(t)} - 2 \cdot \Theta^{(t-1)} + \Theta^{(t-2)}||_2^2 \\ &= 2 \cdot (\Theta^{(t)} - 2 \cdot \Theta^{(t-1)} + \Theta^{(t-2)})\end{aligned}$$

## B.5 Contact Points Term $E_c$

For computing the gradient of the contact points term $E_c$ (see Equation 5.6), the derivative operator is firstly dragged inside the sum. Then, the substitution $\mathbf{z} := \boldsymbol{\mu}_f - \boldsymbol{\mu}_j$ is applied.

$$\begin{aligned}\frac{\partial E_c(\Theta)}{\partial \theta_k} &= \sum_{(f,j,d_\mathcal{T})} \frac{\partial (||\boldsymbol{\mu}_f - \boldsymbol{\mu}_j||_2^2 - d_\mathcal{T}^2)^2}{\partial \theta_k} \\ &= \sum_{(f,j,d_\mathcal{T})} \frac{\partial (||\mathbf{z}||_2^2 - d_\mathcal{T}^2)^2}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \theta_k} \qquad \text{where } \mathbf{z} := \boldsymbol{\mu}_f - \boldsymbol{\mu}_j \\ &= \sum_{(f,j,d_\mathcal{T})} 2(||\mathbf{z}||_2^2 - d_\mathcal{T}^2) \cdot (2\,\mathbf{z}) \cdot \frac{\partial \mathbf{z}}{\partial \theta_k} \\ &= 4 \sum_{(f,j,d_\mathcal{T})} (||\boldsymbol{\mu}_f - \boldsymbol{\mu}_j||_2^2 - d_\mathcal{T}^2) \cdot (\boldsymbol{\mu}_f - \boldsymbol{\mu}_j) \cdot \underbrace{\frac{\partial (\boldsymbol{\mu}_f - \boldsymbol{\mu}_j)}{\partial \theta_k}}_{\text{see Equation B.2}}\end{aligned}$$

After re-substituting, one can spot that the only remaining derivative is again a $\boldsymbol{\mu}$-difference derivative. Since $E_c$ is considering pairs of two model Gaussians — namely one hand and one object Gaussian — the model-model version from Equation B.2 can be used here.

## B.6 Object Occlusion Term $E_o$

Also for the gradient of the object occlusion term $E_o$ (see Equation 5.7), the derivative operator can be dragged inside the summation. For the $k$-th component of the gradient, all terms in the sum that are associated with Gaussians that are not directly influenced by $\theta_k$ vanish. Therefore only the terms corresponding to Gaussians with parent DOF $k$ remain.

$$\begin{aligned}\frac{\partial\, E_o(\Theta)}{\partial\, \theta_k} &= \sum_{i\,\in\, ch(k)} \rho_i \cdot \frac{\partial\, (\theta_k - \theta_k^{saved})^2}{\partial\, \theta_k} \\ &= 2 \sum_{i\,\in\, ch(k)} \rho_i \cdot (\theta_k - \theta_k^{saved})\end{aligned}$$

where $ch(k) = \{i \mid 1 \leq i \leq N_{\mathcal{M}_h} \wedge k \in \mathcal{H}_i\}$ is the set of all Gaussians directly affected by $\theta_k$.

## B.7 Interpenetration Avoidance Term $E_i$

The interpenetration avoidance term is similar to the model-model-part of the spatial alignment term $E_a$ as derived in Section B.1. The only differences are that the visibility weights are not included in $E_i$ and that the product is computed between two different parts of the model, namely hand and object, instead of the same part. Because the derivative of an integral over a product of weighted Gaussian mixtures has been determined in general in Section B.1, the same computation applies here. The $\boldsymbol{\mu}$-difference derivative in the case of $E_i$ always occurs in the model-model-version as introduced in Equation B.2.

# Bibliography

[1] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proc. IEEE CVPR*, June 2015. URL `http://handtracker.mpi-inf.mpg.de/projects/FastHandTracker/`.

[2] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision (IJCV)*, 2016. URL `http://files.is.tue.mpg.de/dtzionas/Hand-Object-Capture`.

[3] Dimitrios Tzionas and Juergen Gall. 3D Object Reconstruction from Hand-Object Interactions. In *Proc. IEEE ICCV*, 2015.

[4] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Proc. IEEE ICCV*, 2013.

[5] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM TOG*, 33 (5):169:1–169:10, 2014. ISSN 0730-0301. doi: 10.1145/2629500.

[6] Danhang Tang, Jonathan Taylor, and Tae-kyun Kim. Opening the Black Box : Hierarchical Sampling Optimization for Estimating Human Hand Pose. In *Proc. IEEE ICCV*, 2015.

[7] Robert Y Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics (TOG)*, 28(3):63, 2009.

[8] Cem Keskin, Furkan Kiraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *ICCV Workshops*, pages 1228–1234. IEEE, 2011. ISBN 978-1-4673-0062-9. URL `http://dblp.uni-trier.de/db/conf/iccvw/iccvw2011.html#KeskinKKA11`.

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

[10] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust Articulated-ICP for Real-Time Hand Tracking. *Computer Graphics Forum (Proc. of SGP)*, 34(5), 2015.

[11] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proc. ACM CHI*, 2015.

[12] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *Proc. IEEE ICCV*, pages 1475–1482, 2009.

[13] J. Romero, H. Kjellstrom, and D. Kragic. Hands in action: real-time 3d reconstruction of hands in interaction with objects. In *Proc. ICRA*, pages 458–463, 2010. doi: 10.1109/ROBOT.2010.5509753.

[14] I. Oikonomidis, N. Kyriazis, and A.A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Proc. IEEE ICCV*, pages 2088–2095, 2011. doi: 10.1109/ICCV.2011.6126483.

[15] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc J. Van Gool, and Marc Pollefeys. Motion Capture of Hands in Action Using Discriminative Salient Points. In *Proc. ECCV*, pages 640–653. 2012. ISBN 978-3-642-33782-6.

[16] Dimitrios Tzionas, Abhilash Srikantha, Pablo Aponte, and Juergen Gall. Capturing Hand Motion with an RGB-D Sensor, Fusing a Generative Model with Salient Points. In *Proc. GCPR*, 2014.

[17] N. Kyriazis and A. Argyros. Scalable 3d Tracking of Multiple Interacting Objects. In *Proc. IEEE CVPR*, pages 3430–3437, June 2014. doi: 10.1109/CVPR.2014.438.

[18] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. Real-time non-rigid reconstruction using an rgb-d camera. *ACM TOG*, 33(4), 2014.

[19] Carl Yuheng Ren, Victor Prisacariu, Olaf Kaehler, Ian Reid, and David Murray. 3d Tracking of Multiple Objects with Identical Appearance using RGB-D Input. In *Proc. 3DV*, 2014.

[20] Jan Knopp, Mukta Prasad, and Luc Van Gool. Orientation Invariant 3d Object Classification Using Hough Transform Based Methods. In *ACM Workshp. Obj. Retriev.*, 3DOR '10, pages 15–20, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0160-2. doi: 10.1145/1877808.1877813. URL `http://doi.acm.org/10.1145/1877808.1877813`.

[21] J. Knopp, M. Prasad, and L.V. Gool. Scene Cut: Class-Specific Object Detection and Segmentation in 3d Scenes. In *Proc. 3DIMPVT*, pages 180–187, May 2011. doi: 10.1109/3DIMPVT.2011.30.

[22] Vasilis Papadourakis and Antonis Argyros. Multiple objects tracking in the presence of long-term occlusions. *CVIU*, 114(7):835–846, 2010.

[23] Vassilis Athitsos and Stan Sclaroff. Estimating 3d hand pose from a cluttered image. In *Proc. IEEE CVPR*, pages 432–442, 2003.

[24] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proc. IEEE CVPR*, pages 3786–3793, 2014. doi: 10.1109/CVPR.2014.490. URL `http://dx.doi.org/10.1109/CVPR.2014.490`.

[25] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proc. IEEE CVPR*, 2015.

[26] T. Heap and David Hogg. Towards 3d hand tracking using a deformable model. In *Proc. Intl. Conf. Automatic Face and Gesture Recognition*, pages 140–145, Oct 1996. doi: 10.1109/AFGR.1996.557255.

[27] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3d hand tracking. In *Proc. Intl. Conf. Automatic Face and Gesture Recognition*, pages 675–680, 2004. doi: 10.1109/AFGR.2004.1301612.

[28] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Proc. GI*, pages 63–70, 2013.

[29] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proc. IEEE CVPR*, 2014.

[30] Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. Video-based hand manipulation capture through composite motion control. *ACM TOG*, 32(4):43:1–43:14, 2013.

[31] Paschalis Panteleris, Nikolaos Kyriazis, and Antonis A. Argyros. 3d tracking of human hands in interaction with unknown objects. In *Proc. BMVC*, 2015. ISBN 1-901725-53-7. doi: 10.5244/C.29.123. URL `https://dx.doi.org/10.5244/C.29.123`.

[32] Leap Motion. `https://www.leapmotion.com/`.

[33] NimbleVR. `http://nimblevr.com/`.

[34] Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *Proc. IEEE ICCV*, pages 951–958, 2011.

[35] Edgar Simo Serra. *Kinematic Model of the Hand using Computer Vision.* PhD thesis, Institut de Robòtica i Informàtica Industrial, 2011. URL `http://www.iri.upc.edu/pfc/show/74`.

[36] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047270. URL `http://doi.acm.org/10.1145/2047196.2047270`.

[37] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[38] Srinath Sridhar, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, and Christian Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 319–326. IEEE, 2014.