Articulation Estimation and Real-Time Tracking of Human Hand Motions

Dissertation

zur Erlangung des akademischen Grades des
Doktors der Naturwissenschaften (Dr. rer. nat.)
an der Technischen Fakultät der Universität Bielefeld

vorgelegt von
Matthias Schröder

Bielefeld 2015

Versicherung

Hiermit versichere ich,

- dass mir die geltende Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte von Dritten oder eigener Prüfungsarbeiten ohne Kennzeichnung übernommen und alle benutzten Hilfsmittel und Quellen in meiner Arbeit angegeben habe,
- dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Vermittlungstätigkeiten oder für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe und
- dass ich keine gleiche, oder in wesentlichen Teilen ähnliche oder eine andere Abhandlung bei einer anderen Hochschule als Dissertation eingereicht habe.

Bielefeld, 2015

Matthias Schröder

Abstract

This thesis deals with the problem of estimating and tracking the full articulation of human hands. Algorithmically recovering hand articulations is a challenging problem due to the hand's high number of degrees of freedom and the complexity of its motions. Besides the accuracy and efficiency of the hand posture estimation, hand tracking methods are faced with issues such as invasiveness, ease of deployment and sensor artifacts. In this thesis several different hand tracking approaches are examined, including marker-based optical motion capture, data-driven discriminative visual tracking and generative tracking based on articulated registration, and various contributions to these areas are presented. The problem of optimally placing reduced marker sets on a performer's hand for optical hand motion capture is explored. A method is proposed that automatically generates functional reduced marker layouts by optimizing for their numerical stability and geometric feasibility. A data-driven discriminative tracking approach based on matching the hand's appearance in the sensor data with an image database is investigated. In addition to an efficient nearest neighbor search for images, a combination of discriminative initialization and generative refinement is employed. The method's applicability is demonstrated in interactive robot teleoperation. Various real human hand motions are captured and statistically analyzed to derive low-dimensional representations of hand articulations. An adaptive hand posture subspace concept is developed and integrated into a generative real-time hand tracking approach that aligns a virtual hand model with sensor point clouds based on constrained inverse kinematics. Generative hand tracking is formulated as a regularized articulated registration process, in which geometrical model fitting is combined with statistical, kinematic and temporal regularization priors. A registration concept that combines 2D and 3D alignment and explicitly accounts for occlusions and visibility constraints is devised. High-quality, non-invasive, real-time hand tracking is achieved based on this regularized articulated registration formulation.

Contents

Chapter 1

Introduction

Ever since technology in general and computers in particular have started to permeate throughout the workplaces, homes and personal spaces of people, human-computer interfaces have trended towards increasingly natural and intuitive interaction concepts. After the keyboard and mouse, multi-touch surfaces have emerged as one of the most popular paradigms for interaction technology, being used in the vast majority of mobile devices, such as smartphones and tablets, and also advancing to larger-scale touch monitor interfaces. A commonality of these input methods is that they are based on the hands and fingers being the central interface to the user. The human hand is extremely well-suited for general tool use and gestural expression, which is why technology interfaces are increasingly headed towards gesture-based interaction (*Microsoft Kinect*, *Leap Motion*, *Nimble VR*). However, despite the progress being made in the field of gestural interfaces, and the general interest in the topic in the fields of character animation and robotics, the problem of detecting and tracking hands in their full expressiveness remains a difficult problem. This thesis deals with various aspects of the hand tracking problem, with emphasis put on facilitating real-time estimation of the hand's full articulation for interactive applications.

Hand tracking is a fundamentally challenging problem due to the high dimensionality and complexity of the space of hand configurations. Hand motions are often fast and intricate, exhibiting complex articulations and contact patterns. Methods aiming at recovering hand articulations from sensor input deal with these challenges in a variety of ways. Tracking solutions like optical motion capture systems (e.g. *Vicon*) or instrumented data gloves (e.g. *CyberGlove*) place markers or sensors directly at the joints of the user's hand, which allows them to recover the hand posture almost directly from the observed data. However, besides requiring complex setup and calibration procedures, a main drawback of these methods is that they are *invasive* to the user being tracked. Wearing a dense set of markers or a bulky data glove restricts the user's hand movements and thereby limits the naturalness of the motions. In contrast, *non-invasive* methods aim to estimate the hand articulations without such restrictions, usually from camera images.

These *visual* hand tracking methods are faced with different problems. Monocular camera setups suffer from occlusions, ambiguities and visual clutter, all of which impede on the tracking quality. Some of these difficulties can be overcome by using multi-camera setups, which help in resolving occlusions or ambiguities. However, these setups usually require lengthy calibration and computations to map the 2D information of the cameras to 3D. As an alternative, *RGBD* sensors (e.g. *Microsoft Kinect*) directly capture color images and depth maps simultaneously, which significantly alleviates a lot of the problems of vision-based tracking. Using RGBD sensors for visual hand tracking is therefore a highly viable option, especially since they are low-cost devices, and targeted towards deployment in desktop or living room environments. Although the data obtained from such sensors often exhibits artifacts like noise, motion blur or gaps, their prevalence and ease of deployment make them one of the most viable alternatives for non-invasive hand tracking. In addition to low-quality data, various other challenges also remain for visual hand tracking using these devices, such as the issue of self-occlusions, which frequently occur during complex hand movements.

In order to cope with these challenges, visual tracking algorithms employ various different techniques. The two main classes of algorithms for visual hand tracking are *appearance-based*, or *discriminative* methods, and *model-based*, or *generative* methods. Discriminative methods estimate the hand posture from a single frame based on a database of known configurations. While these approaches suffer when the input data strays from the database, generative methods are more flexible as they directly optimize for the kinematic parameters of a virtual hand model in order to align it with the observed data. However, purely generative approaches can be prone to sub-optimal local minima and generally suffer from the high dimensionality of the hand posture space. A general trade-off between accuracy and efficiency can be identified in hand posture estimation algorithms. Offline estimation methods describe detailed models of the hand's kinematics and geometry, and can produce highly accurate results using sophisticated optimization frameworks, but they are not applicable to real-time contexts. Real-time approaches introduce simplifications that allow for fast computations, but reduce reconstruction accuracy.

This thesis aims to resolve some of the above issues by examining a variety of different tracking approaches, analyzing hand articulations to identify concepts that can be used to reduce their complexity, and developing approaches that succeed in accomplishing non-invasive, real-time hand tracking that is accurate and robust.

Contributions

This thesis examines the problem of hand tracking and posture estimation from various angles and analyzes aspects of several different hand tracking approaches, including marker-based optical motion capture, discriminative vision-based tracking, and generative tracking based on articulated registration. In the following, the particular contributions of this thesis in these areas are clarified.

We developed a method for hand motion reconstruction from positional marker data obtained from an optical motion capture system. In particular, we addressed the problem of determining the optimal placement of a reduced number of markers, in order to facilitate accurate posture reconstruction from sparse marker data. The main contributions in this area include:

- A method for automatically generating functional reduced marker layouts based on establishing a relationship between marker positions and subspace representations of hand articulations.
- Specific quality criteria for marker layouts that combine numerical stability with geometric feasibility, and a specialized marker layout optimization algorithm that generates such layouts.

We extended previous work on image based discriminative hand tracking for the purpose of interactive teleoperation of an anthropomorphic robot hand. The main contributions in this area include:

- An efficient nearest neighbor search algorithm for color-labeled images based on cascaded image matching using chamfering to efficiently approximate color class distance transforms.
- An additional generative pose estimation step using a color-sensitive iterative closest point (ICP) optimization.

We utilized reduced subspace representations of hand articulations that we derived from principal component analysis (PCA) of motion captured hand posture data, and integrated these as subspace constraints in a generative hand tracking approach using articulated registration. The main contributions in this area include:

- A robust, subspace-constrained inverse kinematics hand posture estimation method, integrated into a registration-based real-time hand tracking approach.
- A fast, online adaptive PCA model that is continuously updated according to new input data based on efficient incremental covariance computations.
- A publicly available dataset of motion captured hand posture data containing a high variety of natural hand articulations.

We formulated generative, or model-based, hand tracking as a regularized articulated registration process that aligns a template model with point cloud data in real-time. The main contributions in this area include:

- A robust method for real-time hand tracking that combines geometric model fitting with statistical, kinematic and temporal priors in a unified articulated registration framework.
- A combined 2D/3D registration and correspondence computation approach that explicitly takes visibility and occlusion constraints into account.
- The complete source code of our implementation.

The contributions of this thesis are presented in the following publications:

- M. Schröder, C. Elbrechter, J. Maycock, R. Haschke, M. Botsch & H. Ritter (2012), Real-Time Hand Tracking with a Color Glove for the Actuation of Anthropomorphic Robot Hands, *Proc. IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pp. 262–269.
- M. Schröder, J. Maycock, H. Ritter & M. Botsch (2013), Analysis of Hand Synergies for Inverse Kinematics Hand Tracking, *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Workshop: Hand synergies – how to tame the complexity of grasping.
- M. Schröder, J. Maycock, H. Ritter & M. Botsch (2014), Real-Time Hand Tracking using Synergistic Inverse Kinematics, *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5447–5454.
- M. Schröder & M. Botsch (2014), Online Adaptive PCA for Inverse Kinematics Hand Tracking, *Proc. Vision, Modeling and Visualization (VMV)*, pp. 111–118.
- A. Tagliasacchi*, M. Schröder*, A. Tkach, S. Bouaziz, M. Botsch & M. Pauly (2015), Robust Articulated-ICP for Real-Time Hand Tracking, *Computer Graphics Forum 34, Proc. Symposium on Geometry Processing (SGP)*, to appear. (* equal contributors)
- M. Schröder, J. Maycock & M. Botsch (2015), Reduced Marker Layouts for Optical Motion Capture of Hands, submitted to *Motion in Games (MIG)*.

Outline

The remainder of this thesis is structured as follows. Chapter 2 provides an overview of the related work on hand modeling and tracking, and introduces some key concepts that are relevant to this thesis. Chapter 3 explores the problem of reduced marker layout optimization for hand tracking based on marker-based optical motion capture. In Chapter 4 a data-driven, discriminative visual hand tracking approach based on nearest neighbor matching in an image database is discussed. Details on subspace representations of hand postures, as well as a generative hand tracking approach based on subspace-constrained inverse kinematics are given in Chapter 5. In Chapter 6 a generative hand tracking method is presented that formulates the posture estimation problem as a real-time regularized articulated registration process. Finally, Chapter 7 concludes the thesis with a summary and discussion.

Chapter 2

Hand modeling and tracking

This chapter provides a general overview of the landscape of works related to hand modeling and tracking, and introduces some key concepts that are relevant to the contributions presented in this thesis. Later on, the individual chapters give specific and self-contained discussions of their respective related works. In the following, the initial topic discussed is the modeling of hands. In particular, the anatomy of the human hand is addressed, and what considerations go into the kinematic and geometric modeling and design of virtual hand representations. After that, the topic of hand tracking and posture estimation is approached, and a summary of the most relevant tracking methods and how they relate to the work in this thesis is given.

2.1 Hand modeling

The human hand consists of an intricate composition of bones, cartilage, muscles, tendons and skin, and constitutes one of the most complex musclotendon systems in the human body (Wheatland et al. 2015). The anatomical structure of the hand is made up of 27 bones, which are held together by ligaments and muscles that allow the individual joints to flexibly bend and rotate based on contraction and relaxation forces (Agur and Lee 1999, Napier 1980). This structural setup is what gives the hand its high capability for articulation and the high density of nerve endings in the fingers is what makes it a useful tool for tactile actions. The 27 bones in the hand include eight that are tightly grouped in the wrist or carpus, five bones in the palm called metacarpals connecting the wrist to the fingers, and the five digits consist of bones called phalanges, of which the thumb has two and the remaining four fingers have three (Agur and Lee 1999). These phalanges are categorized into the proximal, middle and distal phalanx, and while the four fingers all have three phalanges each, the thumb only has a proximal and a distal phalanx. The phalanges within each digit are connected with joints, which facilitate the hand's articulation, and are categorized into metacarpophalangeal joints (MCP) between the metacarpal and proximal phalanx, the proximal interphalangeal joints (PIP) between the proximal and middle phalanx, and the distal interphalangeal joints (DIP) between the middle and distal phalanx. Each of the four fingers articulate at all three joints, but as the thumb lacks a middle phalanx it only articulates its MCP and DIP joints. Regarding the rotational degrees of freedom (DoFs) of the hand's joints, the interphalangeal joints (PIP, DIP) predominantly act as hinge joints, which can perform forward flexion and extension. The metacarpophalangeal joints (MCP) are more flexible and allow for sideways abduction and adduction in addition to flexion and extension. Beyond that, the MCP joints are also slightly articulated with respect to the carpal bones in the wrist (Wheatland et al. 2015). As the structure of the hand is not locked in place but rather held together by flexible tendons and ligaments, there are additional minor DoFs in the joints, such as for translation, hyperextension and twist of the joints, although with limited ranges of motion.

Mapping the complex anatomical structure of the hand to a virtual hand representation is challenging, because there is a fundamental trade-off between the accuracy of the simulation and the efficiency of the model for practical use. For this reason most virtual representations of the hand make abstractions and simplifications upon the hand's true anatomical structure. While there are projects that specifically aim at reconstructing the hand with high anatomical accuracy, including the modeling of bones, muscles and skin (Albrecht et al. 2003, Tsang et al. 2005, Sueda et al. 2008), they are still based on a set of simplified assumptions about the hand structure. The joints are represented as articulated, rigid links as opposed to a flexible musclotendon system, and the number of bones is commonly reduced from 27 to 16 (Tsang et al. 2005). However, this reduction is justified because the movement of the additional joints is negligible in most situations and the simplified structure still allows for accurate anatomical modeling (Albrecht et al. 2003). The articulation of the piecewise rigid kinematic chains representing the digits is usually driven by a limited set of flexion/extension and abduction/adduction joint angles (Oikonomidis et al. 2011a, Schröder et al. 2014). Using four joint angles for each digit (one abduction, three flexion) results in a kinematic model of 20 parameters representing the finger articulation, or hand posture. The wrist can be modeled using six DoFs representing the position and orientation, or the global pose of the hand, resulting in a total number of 26 DoFs representing the kinematic state of the virtual hand model.

While the 20-dimensional representation of hand postures allows for adequate articulation of all individual joints in the kinematic hand model, a closer examination of hand movements reveals that the motions of joints during hand articulations are highly redundant and correlated, indicating that the kinematic DoFs of the hand can be represented using a lower number of control parameters. A commonly observed correlation between finger joints is that between the interphalangeal joints (Rijpkema and Girard 1991, Lin et al. 2000), which can be approximated as $\theta_{\text{DIP}} = 2/3 \cdot \theta_{\text{PIP}}$, where θ_{DIP} and θ_{PIP} represent the DIP and PIP joint angles, respectively. In a more principled approach, Bernstein (1967) generalized this concept to the idea of hand synergies, which represent high-level control schemes for kinematic parameters. Santello et al. (1998) expanded on the hand synergies concept and processed grasp motion data using principal component analysis (PCA) to reveal that 90% of the variance in grasping data could be described by only 3 principal components. The concept of identifying high-level control schemes for hand articulations based on hand posture subspaces has since been further examined and applied in the field of robotics (Bicchi et al. 2011, Gabiccini et al. 2011). Low-dimensional representations for hand kinematics have also been used for the animation of virtual hands (Mulatto et al. 2013, Hoyet et al. 2012) and hand tracking (Wu et al. 2001, Kato et al. 2006, Schröder et al. 2014). In this thesis, extensive use is made of the concept of low-dimensional hand posture representations to increase the robustness of hand posture estimations.

In addition to the structural and kinematic layout of the hand, the hand's geometry also has to be accounted for in virtual hand representations. Early works on the popular linear blend skinning (LBS) animation technique were concerned with the applications of hand animation and object grasping (Magnenat-Thalmann et al. 1988, Gourret et al. 1989). In LBS the deformation of the surface of an articulated geometric model is approximated based on a weighted combination of the joint transformations of the skeleton embedded in the model. While being an efficient and intuitive method for generating smooth skin deformations for animated objects, its realism is limited. More advanced skinning techniques have since been developed, which account for effects such as skin folding and bulging (Vaillant et al. 2013, 2014). Methods targeting anatomically accurate modeling of hands compute the geometric shape and deformation of the hand model based on physical simulation of muscle activations and contractions (Albrecht et al. 2003, Tsang et al. 2005, Sueda et al. 2008).

Accurate hand models can also be generated based on sensor data in offline optimization processes. In the works of Albrecht et al. (2003) and Rhee et al. (2006), hand models that fit to the dimensions of specific users are generated by warping an initial template model according to features extracted from 2D photographs of hands. For a more detailed geometric reconstruction based on sensor data, depth sensing devices can be used, which provide 3D point cloud information. Reconstruction of 3D geometry from point clouds has been addressed in registration techniques (Li et al. 2008, 2009), in which a template model is deformed to fit to the input data by employing non-rigid deformation models in regularized energy optimization frameworks. However, these particular methods do not specifically account for the articulated, kinematic structures of hands. In contrast, Taylor et al. (2014) presented a method for generating user-specific hand models by simultaneously adapting a hand triangle mesh and its embedded skeleton to a sequence of depth images. Similarly, Zhu et al. (2015) created user-specific anatomically based models of upper and lower limbs by adapting the bone, skin and kinematic structure of an initial template model to depth data.

While anatomically based hand models are well-suited for realistic simulations of hands, they are not suitable for use in interactive real-time applications, which is why most hand models used for the purpose of tracking are kept as simple as possible. This thesis demonstrates that highly robust and accurate real-time hand tracking is achievable using a simplified geometric hand representation based on piecewise rigid cylinder segments, when combined with carefully designed kinematic constraints and priors.

2.2 Hand tracking

One of the most reliable and widely deployed solutions for motion tracking in general is marker-based optical motion capture (Guerra-filho 2005, Kitagawa and Windsor 2008), where the 3D positions of markers attached to a performer are tracked by a calibrated setup of high-speed cameras with high accuracy. A typical motion capture (mocap) setup is equipped with 4 to 32 cameras and numerous commercial solutions (e.g. *OptiTrack*, *PhaseSpace*, *Qualisys*, *Vicon*) are deployed in research and industrial contexts. For the specific application of hand tracking, marker-based mocap systems are also commonly employed. Maycock et al. (2010) purposefully built an optical aquisition setup with 14 MX3+ cameras (*Vicon*) in a small capture volume for tracking hand movements during object interactions. Lee and Tsai (2009) used an optical mocap setup for tracking hands for the purpose of sign language recognition. Zhao et al. (2012) combined marker-based mocap with a depth sensor for improved accuracy. In an optical system based on LED-markers, Aristidou and Lasenby (2010) tracked fingertip positions and reconstructed joint angles in a highly constrained inverse kinematics approach. Data-driven hand motion reconstruction from sparse marker sets was addressed in (Kang et al. 2012, Wheatland et al. 2013) and Hoyet et al. (2012) studied the perception of hand movements captured in an optical mocap system using reduced marker sets. In (Schröder et al. 2015) we approached the problem of automatically generating reduced marker layouts for optical mocap of hands based on low-dimensional hand posture representations.

A downside to optical tracking systems is their sensitivity to marker occlusions, which can occur frequently during complex hand movements. Another reliable and widespread alternative for hand tracking that does not have this problem is to use instrumented data gloves (e.g. *CyberGlove*), which directly measure the hand's joint angles with integrated sensors. A common application case for data gloves is hand tracking for the purpose of robot control (Fischer et al. 1998, Griffin et al. 2000, Turner 2001, Steffen et al. 2010). However, while these devices can be used to infer hand articulations directly from the measured data, this requires non-trivial calibration procedures to map from the raw sensor values to joint angles (Steffen et al. 2011).

In addition to being complex and expensive, a major drawback to both marker-based optical systems and data gloves is that they are *invasive* tracking mechanisms that require the user to wear markers or a bulky glove, both of which impede natural movements. In contrast to this, a highly viable non-invasive, cheap and simple alternative to these systems has emerged from the proliferation of consumer-level RGBD sensors (e.g. *Microsoft Kinect*, *ASUS Xtion PRO*, *Creative Senz3D*), which capture depth maps in real-time in addition to color images. This depth information significantly reduces some of the problems often associated with markerless visual tracking, such as occlusions, depth ambiguities and background clutter. The pioneering piece of hardware in this development of commodity depth sensors was the *Microsoft Kinect*, which was built for the purpose of real-time full body tracking for interactive games (Shotton et al. 2011). These sensors have since been used in various other research contexts, such as real-time surface reconstruction (Izadi et al. 2011, Newcombe et al. 2011) or face tracking (Weise et al. 2011, Bouaziz, Wang and Pauly 2013, Li et al. 2013). Early attempts at hand tracking using RGBD sensors detected fingertips in the depth map in order to facilitate interactive, gesture-based user interfaces (Lozano-Perez et al. 2010). Hardware solutions based on depth sensing that were specifically developed for gestural interaction in desktop and virtual reality environments have come up as well (*Leap Motion*, *Nimble VR*). However, despite the multitude of sensors and tracking approaches, motion tracking in general, and hand tracking in particular based on RGBD sensors remains a challenging problem, because the data obtained from these low-cost sensors often has low quality, the range at which they can operate is limited and markerless articulated tracking is inherently difficult. A detailed overview of general human motion analysis from depth data is given in the survey of Ye et al. (2013).

Approaches for visual hand tracking can generally be subdivided into two main categories, namely *appearance-based* methods and *model-based* methods (Erol et al. 2007). Appearance-based methods are also referred to as *discriminative* methods, as they perform the hand posture estimation based on discriminating and classifying the visual input based on a set of known configurations. Such methods usually involve training a classifier or a regressor with a large database of example hand postures in order to map image features to hand articulations. The main advantage of appearance-based methods is that they allow for hand posture estimation from a single frame, which makes them highly robust against loss of tracking. Conversely, these methods suffer when the observed input hand motions stray from the database upon which the discriminative model was built. Therefore, appearance-based methods work optimally when only an approximate posture estimate is desired or when highly discriminative features can be extracted from the visual input. Model-based methods are also referred to as *generative* methods and they produce hand posture estimations by directly optimizing the kinematic parameters of a hand model, thereby generating hand articulations that explain the observed visual input. In these methods the hand posture estimation problem is regarded as an alignment problem, in which an objective function that measures the discrepancy between a virtual hand model and the observed input data is minimized. As model-based methods directly optimize for the kinematic parameters of the hand model and can therefore cover the entire configuration space of hand articulations, they are more flexible and can achieve higher accuracy than appearance-based methods. However, the high dimensionality and complexity of hand articulations can make the optimization of the kinematic parameters difficult, and these optimizations can be prone to local minima and loss of tracking, which is usually dealt with using regularization or reinitialization strategies.

Appearance-based methods using nearest neighbor search match the observed input image, or features extracted from it, to a database of images or features annotated with hand postures. Wang and Popović (2009) detected a color glove worn by the user in the input image and found the closest match in a synthetically created database of tiny images (Torralba et al. 2007). The glove's unique color pattern provided a highly discriminative feature set, which made it possible to distinguish between a large variety of hand postures and orientations based on monocular color input. In (Schröder et al. 2012) we extended this discriminative color glove method with a generative estimation step that aligned the textured hand model with the color-classified point cloud of an RGBD sensor in a color-sensitive iterative closest point (ICP) process. In an effort to remove the dependency on the color glove, Wang et al. (2011) later adapted their approach to silhouette images of hands instead of color-classified glove images. Due to the reduced distinctiveness of the hand's appearance in silhouette images as opposed to color images, the number of postures estimated by this system was limited. In (Romero et al. 2010, 2013) color images of bare hands were used in a similar nearest neighbor approach for tracking hand-object interactions. Other appearance-based methods are built upon decision trees and forests, which allow for per-pixel classification of joint labels in the input depth image. This method was notably used for full body tracking using RGBD sensors by Shotton et al. (2011) and variants of the approach have been similarly employed for hand tracking (Keskin et al. 2012, Tang et al. 2013, 2014, Krupka et al. 2014). Decision forests are constituted by an ensemble of decision tree classifiers, which have to be trained using a large database of annotated example depth maps of the tracked subjects. In a different approach, Tompson et al. (2014) used pre-trained convolutional networks to identify joint locations in depth images of hands. These methods have demonstrated that appearance-based approaches can be successfully used for real-time hand tracking and posture estimation from a single frame, but their reliance of large training data sets and the limited generalization to previously unknown hand articulations are drawbacks.

Model-based methods estimate hand postures by directly optimizing the kinematic parameters of a hand model in order to align it with the observed input data. In (Stenger et al. 2001, Sudderth et al. 2004) edge and contour features were used in model-based approaches using probabilistic estimation algorithms. While these methods used simplified hand representations, de La Gorce et al. (2008) encoded not only the hand's kinematics, but also the details of its geometry, texture and shading in their model. They used an analysis-by-synthesis approach, in which explicit mathematical models for these quantities were used to generate synthetic color images that matched input camera images by performing error minimization in the image domain. Hamer et al. (2009) addressed the challenges of occlusions and hand-object interaction by connecting multiple individual trackers for the joint segments in a pairwise Markov random field. Reconstruction of complex hand-hand and hand-object interactions has been approached using model-based tracking in multi-camera setups (Oikonomidis et al. 2011b, Ballan et al. 2012, Wang et al. 2013). These methods simultaneously model the articulation of hands, the movement of objects and their interactions in optimization frameworks. While these approaches showcase the high potential for accuracy and robustness of model-based tracking, they are offline methods and not suitable for interactive real-time applications. In (Sridhar et al. 2013, 2014) real-time tracking was achieved using Gaussian-based tracking models in multi-camera setups. Sridhar et al. (2015) combined the model-based approach with an appearance-based decision forest classifier for hand tracking using an RGBD sensor.

Other approaches for model-based real-time hand tracking using RGBD sensors are based on the work of Oikonomidis et al. (2011a), who used particle swarm optimization (PSO, Kennedy and Eberhart 1995, 2001) to minimize the model alignment objective function. PSO is a sampling-based optimization heuristic that iteratively evaluates and updates a set of candidate solutions. Its main advantage is that it allows for the optimization of arbitrary objective functions without prior knowledge or gradient information, however it comes at the cost of requiring high computational resources. Zhao et al. (2012) combined the PSO hand tracking technique for RGBD sensors with marker-based optical mocap for improved accuracy. Qian et al. (2014) extended the PSO approach with fingertip detection for reinitialization and registration-based refinement. Sharp et al. (2015) employed a PSO-based model fitting in combination with a per-frame reinitializer for recovery from loss of tracking. While modifying and extending the PSO technique has been shown to produce good results, its inherent probabilistic and heuristic nature make it less preferential than gradient-based approaches, which are more direct and converge faster and more accurately when close to the solution (Qian et al. 2014). Melax et al. (2013) used a gradient-based convex rigid body simulation in combination with reinitialization heuristics for real-time hand tracking. In (Schröder et al. 2013, 2014, Schröder and Botsch 2014) we performed hand tracking using a constrained inverse kinematics approach, where the gradient-based optimization was performed in hand posture subspaces. In (Tagliasacchi & Schröder et al. 2015) we formulated the hand tracking problem as a robust, real-time geometric registration process, which performs gradient-based optimization of an objective function that combines geometric fitting with kinematic regularizers in a unified approach.

Chapter 3

Marker-based motion capture

The approach for hand tracking discussed in this chapter is using marker-based optical motion capture, or mocap, which is widely regarded as the standard method for acquiring motions of human performers in both research and industrial or entertainment contexts. Numerous commercial solutions (*Vicon*, *OptiTrack*, *PhaseSpace*, *Qualisys*) and considerable scientific literature exist on the topic. While there is a multitude of alternative solutions for motion tracking, such as markerless methods (*Organic Motion*, *Microsoft Kinect*) or systems using inertial sensors (*Xsens*, *Biosyn*), they are not as widely deployed due to the reliability of marker-based systems. Marker-based optical mocap systems track the 3D positions of markers attached to a performer, which can then be used to infer the articulation of a skeletal model of the tracked subject. Such systems typically consist of 4 to 32 cameras that capture at 30 to 2000 Hz and acquire the marker locations with very high accuracy (Kitagawa and Windsor 2008).

However, despite the quality of marker-based mocap there are drawbacks and limitations to these systems. The captured data usually needs to be post-processed extensively, occlusions can cause gaps or mislabelings in the captured data, and any rotational information needs to be computed retrospectively. Some of these issues are amplified as the number of markers used for tracking increases. A common guideline for capturing articulated objects is to cover all major joints with markers (Guerra-filho 2005, Kitagawa and Windsor 2008). In addition to making the marker attachment process tedious and error-prone, a high number of markers causes problems when capturing multiple subjects or tracking body movements and hand articulations simultaneously. Capturing hand articulations in detail typically requires a dense marker set consisting of 18–23 markers in a small capture volume. In a large capture volume that also allows for full body mocap the resolution of the optical tracking system and the required size of the markers prohibit the usage of a full marker set. Instead, reduced marker sets have been employed in large capture volumes – however, this strongly limits the expressiveness of the captured hand motions. Therefore, body and hand movements are sometimes captured in isolated sessions and combined in post-processing (Wheatland et al. 2015).

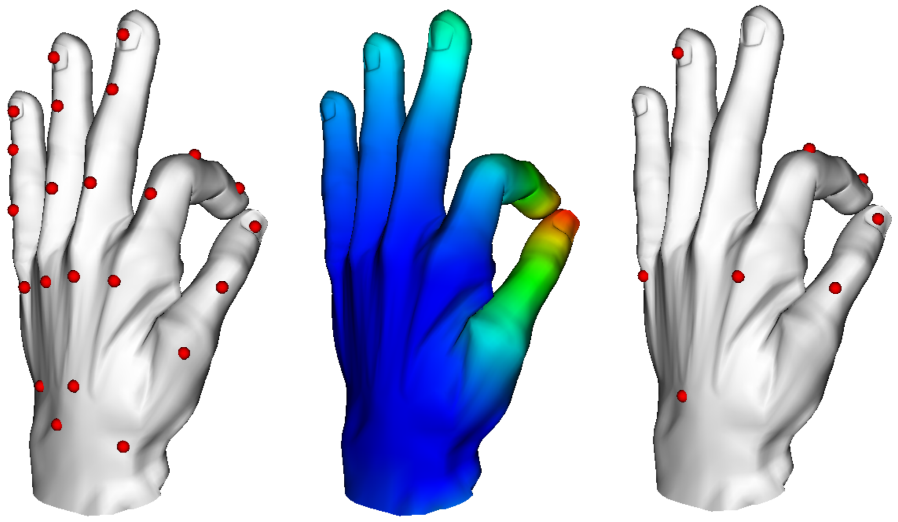


Figure 3.1: Our method generates reduced marker layouts for optical motion capture of hands based on analyzing hand movements. Left: full marker set covering all joints of the hand. Center: qualitative illustration of regions that are static (blue) and in motion (red) during the analyzed precision grasp movement. Right: reduced marker set that is sufficient to reconstruct the observed motions using our method.

In this chapter, this particular problem of optical mocap is explored more closely and a method for automatically determining reduced marker layouts for inverse kinematics (IK) based hand motion reconstruction is presented (Schröder et al. 2015). The employed motion reconstruction method is based on performing the IK optimization in a subspace learned from prior hand movements, which allows for realistic recovery of hand articulations even from sparse input data. Our method for reduced marker set optimization is sensitive to this reconstruction method, particularly the employed subspace constraints, and thus produces layouts that are optimal for solving the subspace-constrained IK problem. We developed an approach that minimizes an objective function which jointly optimizes numerical stability of the marker-IK problem and the geometric feasibility of the resulting layout. The optimization is done using a specialized surface-constrained particle swarm optimization (PSO, Kennedy and Eberhart 1995, 2001), which generates marker layouts bound to the surface of an animated 3D hand model (see Figure 3.1).

We show that, rather than specifying one marker per joint of the articulated object, it is sufficient to specify one marker per degree of freedom (DoF) of the parameter space that represents particular hand articulations. Reduced marker layouts can therefore be determined by reducing the parameter space of hand postures based on prior knowledge. Furthermore, we show the principles by which a reduced marker layout that best corresponds to the subspace DoFs can be determined. We demonstrate marker layout results for various hand motions, in particular manual interaction movements based on the grasp taxonomy of Cutkosky (1989), which distinguishes between different types of power grasps and precision grasps (see Figure 3.2).

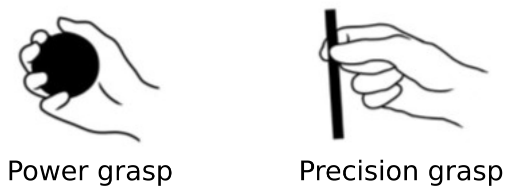


Figure 3.2: Examples of different grasping types in the grasp taxonomy of Cutkosky (1989), which we use to evaluate our results. Power grasps usually involve the whole hand for interaction with large objects, whereas precision grasps usually involve only some fingers for handling smaller objects. Illustrations from (Zheng et al. 2011).

3.1 Related work

There is a substantial amount of literature on optical motion capture, therefore the focus is placed on the related work that is most relevant to the topics discussed in this chapter, which include motion reconstruction based on motion subspace priors, as well as optimized or reduced marker configurations.

Employing subspace representations of human motions has been shown to be effective for motion reconstruction from sparse input. In (Chai and Hodgins 2005, Liu et al. 2006) local linear models were used to represent full-body motions and recover skeletal articulations from sparse marker sets. While these methods are completely data-driven and can therefore limit the space of recovered articulations, our approach uses data-driven subspaces as a prior but also allows for articulation refinements that lie outside of the ground truth database using a layered inverse kinematics approach. Liu et al. (2006) also target the problem of determining reduced marker configurations by finding a subset of an initial input marker set that can produce accurate predictions of the remaining markers. In contrast, we developed a bottom-up approach for generating optimal reduced marker layouts for hands based on the kinematic DoFs of an articulated hand model. While previous methods usually determine reduced marker sets by subsampling a specific initial marker set, our method more generally prescribes properties that candidate marker regions on the surface of a hand model should exhibit, and automatically computes the optimal marker placement within these regions.

Other works deal with the optimal placement of markers, although not necessarily reduced marker layouts. Recently, Loper et al. (2014) demonstrated an approach that is able to capture fine details of soft tissue deformations in addition to full-body skeletal motions without having to rely on very dense marker sets. To improve the accuracy of their motion and shape capture, they extend their initial sparse marker set in a greedy approach that iteratively adds the next best mesh vertex that minimizes an error metric. We show that, for the problem of finding good reduced hand marker layouts, such greedy approaches are outperformed by our PSO-based global search, as it is less prone to suboptimal local minima. Le et al. (2013) explore the problem of determining optimal marker layouts for facial performance capture using an approach that minimizes the reconstruction error for ground truth sequences of high-resolution facial meshes. While their approach is based on surface deformations of facial meshes, our method finds reduced marker layouts by purposefully exploiting the kinematic structure and correlations within an articulated hand model.

While a common guideline for marker placement on hands is to use one marker per joint (Guerra-filho 2005, Kitagawa and Windsor 2008), reduced marker layouts for hands have been frequently discussed. In (Kitagawa and Windsor 2008) an example for a reduced "mitten" layout was given, where only one marker was placed at the tip of a single finger. Given an estimation for the global location and orientation of the hand, the relative movement of this marker can be interpreted as the simultaneous bending of all fingers. Our work examines this concept more closely by considering how correlations and redundancies in hand articulations affect marker placement. Regarding the degree of realism of finger motions with reduced marker sets Hoyet et al. (2012) found that humans are not particularly sensitive to the subtle details of finger animations and the perceived quality of motions is not significantly affected by reduced marker sets. While they manually selected reduced marker configurations, we developed an automatic approach based on subspace-constrained inverse kinematics. In contrast, Chang et al. (2007) determine the most important markers in a reduced marker set for the purpose of grasp motion recognition by using supervised feature selection based on the prediction accuracy of grasp classifiers. In (Kang et al. 2012, Wheatland et al. 2013) a data-driven approach for hand motion reconstruction from sparse marker sets was used, where motions are synthesized by finding database postures that most resemble the low-dimensional input. Wheatland et al. (2013) computed a subset of an initial full marker set by performing principal component analysis (PCA) on the marker trajectories and selecting the most influential ones. Our method differs from theirs in two significant aspects: first, our IK-based approach allows for the recovery of hand articulations that are not present in the prior database, and second, we determine reduced marker layouts in a bottom-up way based on the PCA of joint angles, which explicitly captures the correlations and redundancies present within hand kinematics, unlike positional marker trajectories.

Using PCA or other dimension reduction techniques for hand kinematics has found widespread success in hand tracking, animation and automation (Bernstein 1967, Wu et al. 2001, Kato et al. 2006, Mulatto et al. 2013, Schröder et al. 2014). To reconstruct the kinematic parameters of an articulated hand model from positional marker data, we apply the subspace-constrained inverse kinematics approach that we developed in (Schröder et al. 2014) and is presented in more detail in Chapter 5. A key insight of said approach is that while using subspace constraints, the hand posture estimations remain realistic even when input data is missing. In this chapter, we reverse the problem and seek to find the minimal amount of marker input data necessary to reconstruct postures accurately using subspace priors. As in previous works on reduced marker sets for hand mocap (Chang et al. 2007, Kang et al. 2012,

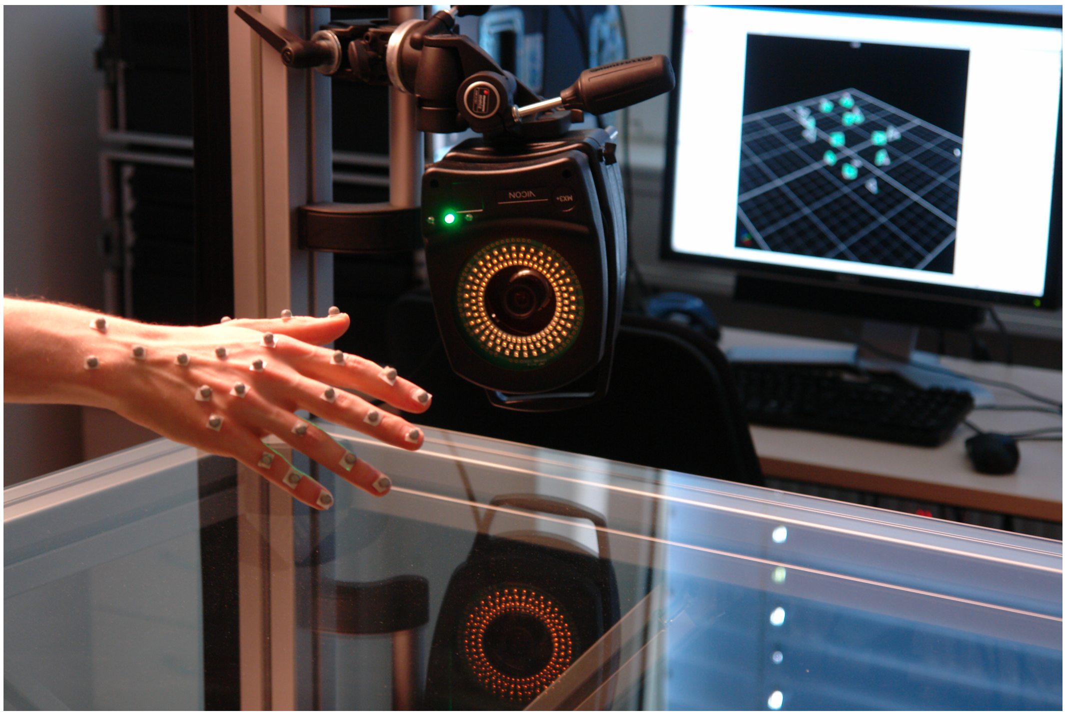


Figure 3.3: Human hand with reflective markers attached for tracking using a Vicon motion capture system (*Vicon*, Maycock et al. 2010).

Wheatland et al. 2013, Hoyet et al. 2012), our marker layouts describe only the articulation of the hand, whereas the global position is given by markers placed on the forearm near the wrist.

3.2 Motion reconstruction

In our setup, human hand motions are recorded using a *Vicon* motion capture system with 14 MX3+ cameras capturing at 200 fps. This acquisition setup was purposefully built to allow for the tracking of human hand motions (Maycock et al. 2010). The dimensions of the system's working volume are 2.1m by 1.3m by 2.1m. Given a set of target marker positions from the optical mocap system, our motion reconstruction method estimates the hand posture, from which the observed positions originate, by fitting an articulated hand model to the data using inverse kinematics. In the following, the hand model and the inverse kinematics fitting process are described in detail.

3.2.1 Kinematic hand model

We use a kinematic hand model consisting of 16 joints: three for the proximal, intermediate and distal phalanges of each finger and one wrist joint. The articulation

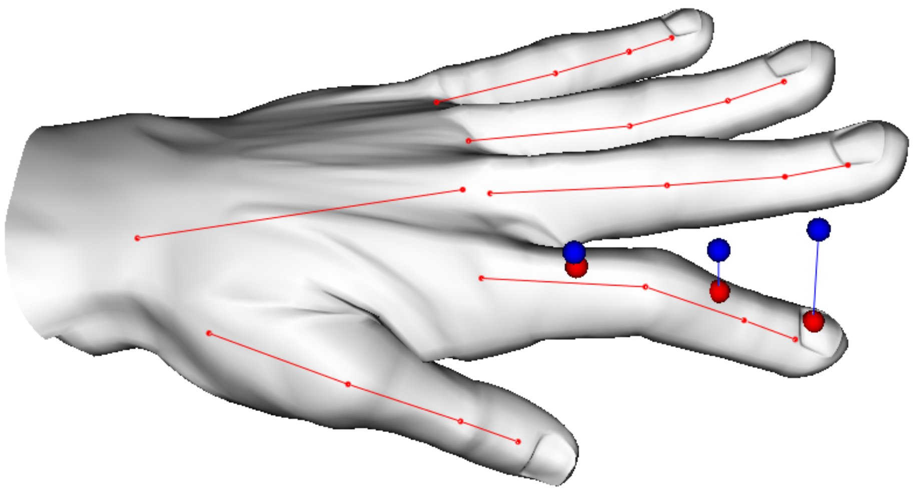


Figure 3.4: Hand model and its underlying skeleton. Also shown are three exemplary markers on the hand model (red) that are constrained to move towards their target positions (blue) during inverse kinematics.

of the hand is represented by 20 degrees of freedom in our model: each finger joint has a flexion-extension axis and the fingers' base joints each have an additional abduction-adduction axis. In addition to the 20 joint angles controlling the hand's posture, the pose of the hand is represented by 6 degrees of freedom for the global translation and rotation. In total we use 26 parameters to control the pose and posture of the hand in our system.

These parameters and the kinematic chains of the joint hierarchy define the forward kinematics of the hand, which can be expressed in terms of a product of affine transformations. A joint's *local* transformation consists of the rotation defined by its joint angle parameters and the translation relative to its parent joint, if there is one. The *global* transformation matrix $\mathbf{T}_j$ of joint j is given by the product of the local transformations along its kinematic chain:

$$\mathbf{T}_j = \prod_{i=1}^{n} T_i(\theta_i), \tag{3.1}$$

where $T_i(\theta_i)$ is the local transformation matrix associated with the element θ_i of the kinematic parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{26})^T$.

The geometric representation of the hand model is based on a triangle mesh, which is deformed according to the articulation of the joints defined in the kinematic hand model. The joint hierarchy serves as the *skeleton* of the virtual hand model, as depicted in Figure 3.4. A point $\mathbf{x}_i$ on the model surface can be transformed relative to a joint j based on the forward kinematics of the skeleton:

$$\mathbf{x}'_i = \mathbf{T}_j \hat{\mathbf{T}}_j^{-1} \mathbf{x}_i, \tag{3.2}$$

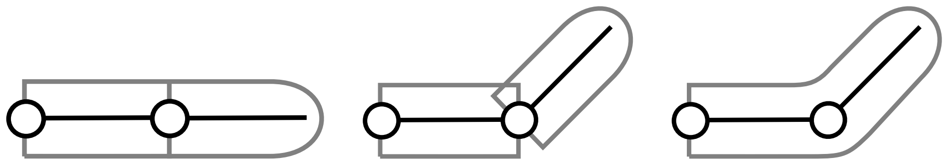


Figure 3.5: Illustration of rest pose (left), piecewise rigid forward kinematics (center) and smooth deformation using linear blend skinning (right).

where $\hat{\mathbf{T}}_j$ is the rest pose transformation of joint j and its inverse is used to transform $\mathbf{x}_i$ to the joint's local coordinate frame. Since the transformation matrices $\mathbf{T}_j$ depend on the parameter vector $\boldsymbol{\theta}$, the transformation of a point $\mathbf{x}_i$ based on the skeleton can be expressed as a function of the parameters: $\mathbf{x}_i = \mathbf{x}_i(\boldsymbol{\theta})$. We use this expression to calculate the forward kinematics during the motion reconstruction process.

However, computing the deformation of the mesh-based hand model in this manner results in a blocky, unnatural animation. Instead, we obtain a smooth deformation of the mesh model using linear blend skinning (LBS, Jacka et al. 2007), which computes deformed vertex positions by linearly blending the transformation matrices of the joints influencing a vertex, which is specified by a set of convex weights $(\omega_1, \ldots, \omega_{16})$ for each vertex. The position $\mathbf{x}_i'$ of vertex $\mathbf{x}_i$ according to LBS is given by

$$\mathbf{x}_i' = \sum_{j=1}^{16} \omega_j \mathbf{T}_j \hat{\mathbf{T}}_j^{-1} \mathbf{x}_i. \tag{3.3}$$

This results in a smooth deformation of the virtual hand model in accordance to the control parameters of the kinematic model. Figure 3.5 schematically illustrates the forward kinematics and skinning.

On the surface of the hand model, *effector* positions are defined, which correspond to the marker *target* positions in the input data, as illustrated in Figure 3.4. The associations between the target and effector positions can be obtained by either manually labeling the observed data or computing the labels automatically (Meyer et al. 2014). The problem of finding the hand model parameters that move the effector positions to their corresponding targets is solved using inverse kinematics. The details of this process are described in the following.

3.2.2 Inverse kinematics

We denote the effector positions by a stacked vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)^T$ and the target positions by $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_k)^T$. As stated in Section 3.2.1, the effector positions can be expressed as functions of the parameters $\mathbf{x}_i = \mathbf{x}_i(\boldsymbol{\theta})$ or $\mathbf{x} = \mathbf{x}(\boldsymbol{\theta})$. The IK problem,

$\mathbf{t} = \mathbf{x}(\boldsymbol{\theta})$, can then be solved by iteratively finding a parameter update $\Delta\boldsymbol{\theta}$ from the previous frame $\boldsymbol{\theta}$ that minimizes the objective function:

$$E_{\text{IK}}\left(\Delta\boldsymbol{\theta}\right) = \frac{1}{2}\|\mathbf{x}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) - \mathbf{t}\|^2 + \frac{1}{2}\|\mathbf{D}\Delta\boldsymbol{\theta}\|^2 \tag{3.4}$$

The first term models the least squares error between the effectors $\mathbf{x}_i$ and their target positions $\mathbf{t}_i$. The second term regularizes the problem by damping the parameter update $\Delta\boldsymbol{\theta}$ with a diagonal matrix $\mathbf{D}$ of damping weights $\lambda_1, \ldots, \lambda_{26}$. In our system we use no damping ($\lambda_i = 0$) for the six pose parameters and only a small amount of damping ($\lambda_i = 1.0$) for the 20 posture parameters.

We minimize (3.4) using a Gauss-Newton approach, each iteration of which involves solving the linear system

$$\left(\mathbf{J}^T\mathbf{J} + \mathbf{D}\right)\,\delta\boldsymbol{\theta} = \mathbf{J}^T\mathbf{e}, \tag{3.5}$$

where $\mathbf{e} = \mathbf{t} - \mathbf{x}$ is the current target-effector error and $\mathbf{J}$ is the $(3k \times 26)$ Jacobian matrix of the effector positions

$$\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} = \left(\frac{\partial \mathbf{x}_i}{\partial \theta_j}\right)_{i,j}. \tag{3.6}$$

The entries of the Jacobian $\mathbf{J}$ are computed as described in (Buss 2004). For a rotational joint with position $\mathbf{k}_j$, rotation axis $\mathbf{a}_j$ and joint angle θ_j, the Jacobian entry for an effector position $\mathbf{x}_i$ that is affected by the joint is given by

$$\frac{\partial \mathbf{x}_i}{\partial \theta_j} = \mathbf{a}_j \times \left(\mathbf{x}_i - \mathbf{k}_j\right). \tag{3.7}$$

For a translational joint with translation axis $\mathbf{a}_j$ and translation distance θ_j, the Jacobian entry for an effector position $\mathbf{x}_i$ that is affected by the joint is given by

$$\frac{\partial \mathbf{x}_i}{\partial \theta_j} = \mathbf{a}_j. \tag{3.8}$$

For effectors i that are not influenced by joint j the corresponding Jacobian entry is zero.

Solving the system (3.5) yields the update direction $\delta\boldsymbol{\theta}$, whose step size typically is determined by a line search. We start with $\delta\boldsymbol{\theta}$ and successively halve it ($\delta\boldsymbol{\theta} \leftarrow \frac{1}{2}\delta\boldsymbol{\theta}$) until the error eventually decreases, i.e., $E_{\text{IK}}\left(\Delta\boldsymbol{\theta} + \delta\boldsymbol{\theta}\right) < E_{\text{IK}}\left(\Delta\boldsymbol{\theta}\right)$. Only then the update $\Delta\boldsymbol{\theta} \leftarrow \Delta\boldsymbol{\theta} + \delta\boldsymbol{\theta}$ is accepted. This process is iterated until the Gauss-Newton minimization converges, which typically requires 5–10 iterations. As a starting value for $\Delta\boldsymbol{\theta}$ we simply use the update from the previous frame, i.e., we use a linear prediction as initial guess.

While our technique is very similar to the popular damped least squares method described in (Buss 2004), it differs in two important aspects: The selective damping (cf. Buss and Kim 2004) by $\mathbf{D}$ (instead of $\lambda\mathbf{I}$) increases responsiveness and performance of the system, and the step size control for $\delta\boldsymbol{\theta}$ improves robustness by preventing oscillations. Without this step size control, a much higher damping is required to keep the tracking stable, which however leads to a significantly higher "laggyness".

In order to prevent the IK optimization from generating physically impossible hand postures, we constrain the posture parameters to plausible value ranges by adapting the joint limit avoidance of Chan and Dubey (Chan and Dubey 1995) to our setup. A posture parameter θ_i is slowed down by increasing its damping parameter λ_i in the matrix $\mathbf{D}$ as soon as it reaches its lower or upper joint limit $\theta_{i,\min}$ or $\theta_{i,\max}$, respectively. To this end a joint limit function H_i is defined as

$$H_i(\theta_i) = \frac{(\theta_{i,\max} - \theta_{i,\min})^2}{(\theta_{i,\max} - \theta_i)(\theta_i - \theta_{i,\min})},$$

and the damping parameter λ_i is scaled by $(1 + |\partial H_i / \partial \theta_i|)$ iff θ_i is moving *towards* its limits. The latter criteria is characterized by an increase of $|\partial H_i / \partial \theta_i|$ from the previous frame to the current, i.e., if $\Delta |\partial H_i / \partial \theta_i| \geq 0$. If the parameter is moving away from its limit ($\Delta |\partial H_i / \partial \theta_i| < 0$), its movement is unrestricted. This simple method causes parameters that move close towards their limits to slow down and to virtually stop when the joint limit is nearly reached. We obtain the joint limits $\theta_{i,\min}$ and $\theta_{i,\max}$ by extracting the minimum and maximum values from a database containing real human hand postures (Schröder et al. 2014).

The result of the inverse kinematics process is an update to the kinematic parameter vector $\boldsymbol{\theta}$ that moves the effector positions on the model to the marker target positions in the input data. Given a full marker set that specifies the articulation of every joint this produces accurate reconstructions of the input motion. However, when using reduced marker sets the input data is sparse and the motions of joints that are not constrained by marker positions cannot be recovered. For this reason, a subspace prior that captures the correlations of joint movements is employed in the inverse kinematics scheme.

3.2.3 Subspace prior

In order to obtain a suitable training data set for a subspace representation of hand articulations, we captured various human hand motions using a full set of markers (see Figure 3.3) and computed the joint angles of our hand model from the marker positions with inverse kinematics. The global pose information was removed from the data, since they have no influence on the correlations in hand articulation we want to exploit. The recorded hand motions include various manual interaction tasks, such as different grasping and twisting motions, sign language, and general hand movements exploring the hand's natural degrees of freedom. This gave us a varied set of hand postures that covered many aspects of natural hand articulation.

The final data matrix $\mathbf{\Theta}$ of m entries of the 20-dimensional posture data was pre-processed to have zero mean before PCA was performed on it. This resulted in a 20×20 matrix $\mathbf{V}$ of eigenvectors and the set of 20 eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{20})$. Taking the eigenvectors in $\mathbf{V}$ corresponding to the l largest eigenvalues yields a $20 \times l$ matrix of principal components, $\mathbf{P}$. Since the data used for PCA only contains the 20 joint angles and not the additional 6 pose DoFs, we construct the conversion matrix $\mathbf{M}$ that maps from the reduced $(6+l)$-dimensional principal component-space (PC-space) to the $(6+20)$-dimensional parameter space:

$$\mathbf{M} = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{P} \end{pmatrix}, \tag{3.9}$$

where $\mathbf{I}$ is a 6×6 identity matrix, requiring the global pose parameters to be the first 6 in the parameter vector. The full parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{6+20}$ can be converted to the reduced parameter vector in PC-space, $\boldsymbol{\alpha} \in \mathbb{R}^{6+l}$, by the mapping (and vice versa by its inverse)

$$\boldsymbol{\alpha} = \mathbf{M}^T(\boldsymbol{\theta} - \boldsymbol{\mu}), \quad \text{and} \quad \boldsymbol{\theta} = \mathbf{M}\boldsymbol{\alpha} + \boldsymbol{\mu}. \tag{3.10}$$

Here, $\boldsymbol{\mu} \in \mathbb{R}^{26}$ is the mean of the data matrix $\mathbf{\Theta}$ with six leading zero-entries for the pose DoFs. This allows the PC-space parameters to be expressed as a function of the kinematic parameters, $\boldsymbol{\alpha} = \boldsymbol{\alpha}(\boldsymbol{\theta})$, and vice versa, $\boldsymbol{\theta} = \boldsymbol{\theta}(\boldsymbol{\alpha})$. In order to perform the inverse kinematics hand posture estimation using the reduced PC-space parameters, the parameter update rule must be adapted.

Given the above mapping, the forward kinematics of an effector point $\mathbf{x}_i$ can be written as a function of the PC-space parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)^T$: $\mathbf{x}_i = \mathbf{x}_i(\boldsymbol{\alpha}) = \mathbf{x}_i(\boldsymbol{\theta}(\boldsymbol{\alpha}))$. According to the chain rule, the $3k \times l$ Jacobian matrix $\mathbf{J}_{\text{PC}}$ of the effector positions w.r.t. the PC-parameters $\boldsymbol{\alpha}$ is:

$$\mathbf{J}_{\text{PC}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\alpha}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\alpha}} = \mathbf{J} \cdot \mathbf{M}, \tag{3.11}$$

where $\mathbf{J}$ is the Jacobian matrix defined in (3.6). The joint limit avoidance based on selective damping can be applied to the PC-parameters in exactly the same way if upper and lower limits $\alpha_{i,\min}$ and $\alpha_{i,\max}$ for the PC-space parameters α_i are available. We determine these parameter limits by iterating through all postures in the training data set, projecting them into PC-space, and storing the minimum and maximum values along all PC-axes. The PC-space parameter update $\Delta\boldsymbol{\alpha}$ is obtained by replacing $\mathbf{J}$ by $\mathbf{J}_{\text{PC}}$ and $\mathbf{D}$ by an analogous $(6+l) \times (6+l)$ damping matrix in (3.5). The number of subspace dimensions l determines the amount of variance in the input data covered by the subspace and can be seen as a control variable for the eventual number of markers k employed in a reduced marker layout. In (Schröder et al. 2013, 2014) and in Chapter 5 we show that in order to represent 90% of given hand movements, 3–6 subspace dimensions are sufficient (see Section 5.5). The subspace solution naturally constrains the reconstructed hand postures to linear combinations of the principal components of the posture database and allows joints to move in correlation to others even when they are not constrained by markers.

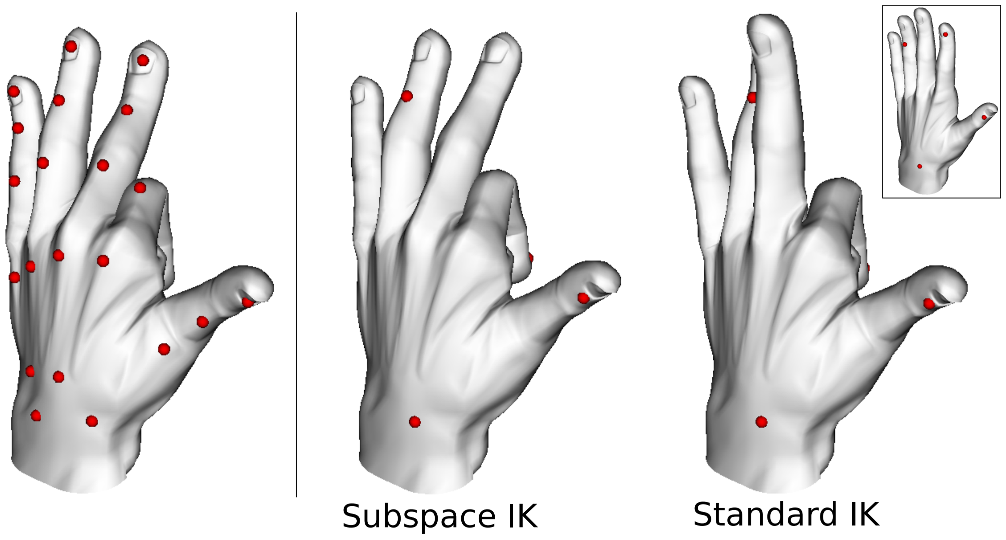


Figure 3.6: Full and reduced marker sets and reconstructed hand postures with standard inverse kinematics optimizing for all joint angles and subspace inverse kinematics optimizing for reduced subspace parameters. While the standard approach cannot articulate the markerless fingers, the subspace approach captures the correlations between fingers and articulates them using the reduced marker set.

However, as there can be variations between the movements contained in the database and the ones observed in the mocap data, we only use this subspace estimate as an initialization for a subsequent refinement of the full posture parameters. By removing the subspace constraints after the initialization of the subspace parameters $\boldsymbol{\alpha}$ and refining the estimate by solving the IK problem again for the full parameter vector $\boldsymbol{\theta}$, the joints with markers are allowed to move more closely to the observed marker positions. This layered IK scheme makes it possible to obtain hand motion reconstructions that are both realistic, due to the subspace prior, and accurate, due to the full kinematic refinement. Figure 3.6 shows a comparison of standard IK with the subspace approach we employ.

3.3 Reduced marker layouts

Subspace-constrained inverse kinematics makes it possible to fully articulate a hand model based on a sparse set of marker points. However, the choice of marker placement is not arbitrary, and to find the optimal marker layout necessitates a method that can assess the quality of a given layout in relation to others. In the following, we discuss the general considerations taken into account and the specific quality metrics employed in our marker layout optimization.

Given a ground truth trajectory of hand motions from a database, the most straightforward way to evaluate the quality of a given marker set is to compare the ground

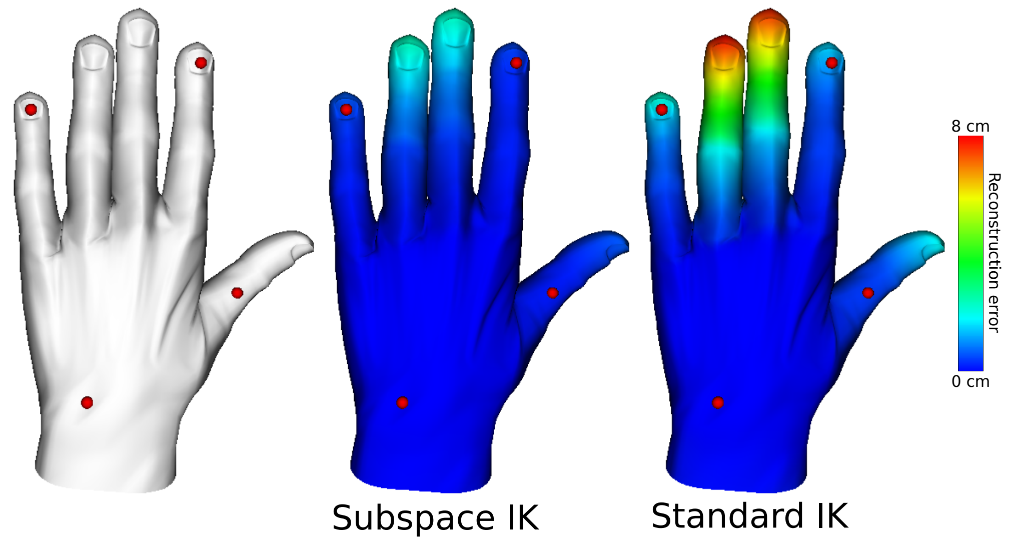


Figure 3.7: Visualization of the reconstruction error of a reduced marker layout for grasping motions. In the analyzed motion data, all fingers bend forward simultaneously and the reduced layout only specifies markers on three out of the five fingers. Using standard IK this causes a large error in the markerless fingers, but using subspace IK the movement of these fingers is correlated to the outer markers, which lowers the error.

truth trajectory with one reconstructed using a reduced marker set. The specific metric we consider here is the positional reconstruction error, which measures the deviation of the reconstructed trajectories of the model vertices $\mathcal{V}$ from the ground truth trajectories. While this is an intuitive measurement for the deviations in the results of the motion reconstruction (see e.g. Figure 3.7), it is not convenient as a metric for choosing an optimal marker layout. Its computation is prohibitively inefficient and it does not generalize beyond the specific input trajectory. Instead, we use metrics that effectively incorporate the IK problem setup, the subspace DoFs and generic geometric considerations.

A reduced marker set must be configured in such a way that the subspace IK can produce the most accurate results. Additionally, the layout must be designed such that is well suited for practical use, which means that it should be unobtrusive, easy to apply and should obviate occlusions and self-contact. In the following, we break these requirements down into two categories, numerical stability and geometric feasibility.

3.3.1 Numerical stability

Our IK hand motion reconstruction is based on solving the linear system (3.5). The numerical stability of the IK problem is measured by the invertibility of the left hand side matrix $\mathbf{J}^T\mathbf{J} + \mathbf{D}$, the key component of which is the Jacobian $\mathbf{J}$ (or $\mathbf{J}_{\text{PC}}$), which is the derivative of the marker positions with respect to the kinematic (or subspace) parameters. Different marker layouts define different Jacobians, each marker defines three rows in the Jacobian matrix. Therefore we denote the Jacobian matrix produced by a specific marker layout $\mathcal{M}$ as $\mathbf{J}_{\mathcal{M}}$. Each kinematic (or subspace) DoF corresponds to a column in the Jacobian. As we are only interested in the minimal layout necessary for accurate posture estimation (joint angles), we omit the three columns in the Jacobian that correspond to translational DoFs, which means that ${\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}$ is a 23×23 matrix for the full parameter space and a $(3+l) \times (3+l)$ matrix for the reduced parameter space.

A criterion for the invertibility of a matrix is its condition number, which is low when the problem is well-conditioned and high when it is ill-conditioned. As we are interested in the most numerically stable marker layout, we omit the damping matrix $\mathbf{D}$, which is not impacted by the markers, and only regard the condition number of the matrix ${\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}$. We compute the condition number of the matrix ${\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}$ using its singular values as

$$\kappa\left({\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}\right) = \left|\frac{\sigma_{\max}\left({\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}\right)}{\sigma_{\min}\left({\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}\right)}\right|, \tag{3.12}$$

where $\sigma_{\max}(\mathbf{A})$ and $\sigma_{\min}(\mathbf{A})$ denote the maximum and minumum singular values of matrix $\mathbf{A}$, respectively.

Optimizing the marker layout $\mathcal{M}$ for the condition number $\kappa\left({\mathbf{J}_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}\right)$ produces marker layouts whose IK solutions are numerically stable by covering the kinematic DoFs of the hand. Taking into account the subspace prior in the IK system by using the subspace Jacobian (3.11), the marker positions tend to positions that optimally cover the subspace DoFs. Figure 3.8 illustrates this concept. Note that the number of markers needed to specify the IK problem is determined by the number of DoFs representing the posture. The full posture space therefore cannot be used to produce sparse marker sets (less than 8 markers), since the IK problem would be underspecified. Employing a subspace representation facilitates reduced marker sets.

3.3.2 Geometric feasibility

Optimizing only for the condition number of the system matrix produces numerically stable and kinematically meaningful marker layouts, however they can be unsuitable for practical use by placing markers at positions that are obstructive for the mocap performer or are sensitive to occlusions and self-contact. Therefore, we consider geometric feasibility in addition to numerical stability in order to produce well-conditioned marker layouts that are also good in practice. We do this in part by limiting the areas where markers can be placed. While this could be done by manually predefining allowed regions, this would cause the need for user intervention. Instead, we define some generic properties that the model vertices should exhibit to select

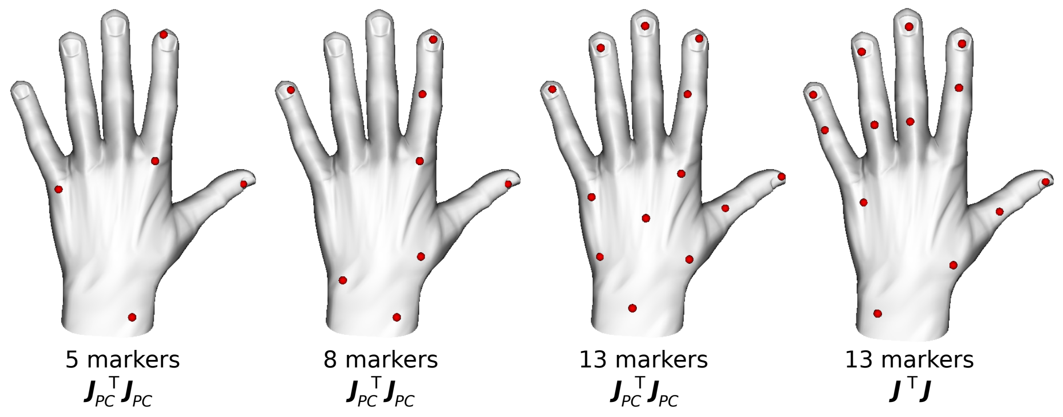


Figure 3.8: Marker layouts of different sizes for a precision grasp movement involving the index finger and thumb. The rightmost layout with 13 markers was computed using the full Jacobian $\mathbf{J}$ for the condition number metric, whereas the others were computed using the reduced Jacobian $\mathbf{J}_{\mathrm{PC}}$.

feasible ones automatically. Additionally, we need to model geometric properties that cannot be accounted for by preselecting vertices, as they change during hand motions (e.g. self-contact).

The first set of geometric feasibility properties is the potential areas for positioning the markers on the surface of the hand model. As the hand naturally bends inwards and can come in contact with objects in the front, markers should generally not be placed on the front side, but rather on the back. Similarly, the markers should be prevented from touching the other fingers during motion and therefore markers should not be placed towards the sides of the fingers. We therefore define feasible regions on the surface of the hand model based on the vertex normals. Only vertex positions $\mathbf{p}_i \in \mathcal{V}$ whose normals $\mathbf{n}_i$ satisfy the the condition $\mathbf{n}_i \cdot \mathbf{h} > 0.9$, where $\mathbf{h}$ is the hand model's back-facing vector, are eligible as marker positions.

The second set of geometric feasibility properties taken into account is marker movement. In practice, markers placed near the joint pivot can move non-rigidly along with the joint rotation due to stretching and sliding of the skin. To prevent this, we identify regions on the skinned mesh that move rigidly relative to joints by considering the hand model vertices' convex skinning weights and only using vertices with weight 1 for one joint. Another movement-related issue is when markers can come in contact with each other during motions, which is especially important even with reduced marker layouts when using large markers. To prevent marker contact from occurring, we maximize the minimum distance between markers across multiple keyframes in the input trajectory. For a single frame, the minimum distance between two markers in a marker set $\mathcal{M}$ is

$$\delta(\mathcal{M}) = \min_{\mathbf{a}\in\mathcal{M}} \left\{ \min_{\mathbf{b}\in\mathcal{M}\setminus\{\mathbf{a}\}} \left\{ \|\mathbf{a}-\mathbf{b}\|^2 \right\} \right\}. \tag{3.13}$$

Maximizing this metric over all frames causes markers to spatially disperse as far from each other as possible, particularly when finger movements cause otherwise spatially distant markers to approach each other more closely.

The combination of these criteria serve as a geometric regularization to the kinematic constraints imposed on the marker set. As a result, the markers are placed in geometrically feasible hand regions during the optimization. The layouts shown in Figure 3.8 combine the numerical and geometric criteria. In the following, the combination of the discussed metrics and their respective influences are discussed.

3.4 Layout optimization

We now combine the quality measures for reduced marker layouts in an energy minimization scheme, in which the marker set $\mathcal{M}$ that minimizes an objective function $E(\mathcal{M})$ is found using stochastic optimization. To this end, we employ a specialized surface-constrained particle swarm optimization (PSO) scheme, which confines the solution domain to the vertices $\mathcal{V}$ of an animated hand model. In addition to the vertices, the input to this optimization includes the vertex normals and skinning weights, as well as a training set of example hand motions. The marker set quality properties are evaluated on the model's vertex positions. A distinction can be made between static properties, which are invariant to hand motion and relative marker placements, and dynamic properties, which vary with different motions and marker layouts.

Static aspects of marker layout quality are those that prevent negative effects of skin sliding, by penalizing the vertices' skinning weights, and obstructiveness, by penalizing the vertices' normal angles. These properties can be incorporated by preselecting only vertices that satisfy them. This yields a set of preselected vertices $\mathcal{V}' \subset \mathcal{V}$ on the hand model surface that are eligible as potential marker positions. Ultimately, the optimized marker layout will be a subset $\mathcal{M} \subset \mathcal{V}'$ of this preselection.

In contrast, dynamic aspects of marker layout quality cannot be evaluated as isolated vertex properties, as they vary with changes in hand articulation and placement of the remaining markers within the layout. These include the numerical stability measured by the condition number of the IK system matrix $\mathbf{J_{\mathcal{M}}}^T\mathbf{J}_{\mathcal{M}}$ and the minimum marker distance. To account for these changes with respect to different hand articulations, we evaluate and accumulate these metrics over a set of F representative keyframes of a given input hand motion trajectory, which can be automatically computed using farthest point optimization (Schlömer et al. 2011) in the hand posture domain. These dynamic properties of marker set $\mathcal{M}$ are modeled in the objective function $E(\mathcal{M})$, whose definition and optimization are discussed in the following.

3.4.1 Objective function

The objective function that is minimized in the marker layout optimization is a weighted combination of energy terms with respect to marker set $\mathcal{M}$

$$E(\mathcal{M}) = w_1 \cdot E_{\text{cond}}(\mathcal{M}) + w_2 \cdot E_{\text{dist}}(\mathcal{M}) \,, \tag{3.14}$$

where $E_{\text{cond}}(\mathcal{M})$ penalizes the condition number of the IK system matrix induced by the marker layout Jacobian, and $E_{\text{dist}}(\mathcal{M})$ penalizes the minimum distance between any two marker positions in the layout. Both of these terms are evaluated over a set of F frames in a hand motion trajectory that are representative of the movements that should be captured in the reduced marker set. We denote the marker configuration of layout $\mathcal{M}$ in frame f as $\mathcal{M}^{(f)}$.

Based on (3.12), the energy term penalizing the condition numbers of the induced system matrices is defined as

$$E_{\text{cond}}(\mathcal{M}) = \frac{1}{F} \sum_{f=1}^{F} \kappa\Big(\mathbf{J}_{(f)}{}^{T} \mathbf{J}_{(f)}\Big), \tag{3.15}$$

where $\mathbf{J}_{(f)}$ denotes the Jacobian of marker configuration $\mathcal{M}^{(f)}$. This term minimizes the average condition number across all F frames. Since the considered marker layout is a subset of the preselected vertices $\mathcal{M} \subset \mathcal{V}'$, we can precompute the vertex Jacobian $\mathbf{J}_{\mathcal{V}'}$ for all F frames and construct the respective marker Jacobians by selecting the corresponding rows in this matrix.

Based on (3.13), the energy term penalizing the minimum distance between two marker positions across all keyframes is defined as

$$E_{\text{dist}}(\mathcal{M}) = -\frac{1}{L} \min_{f \in [F]} \Big\{ \delta\Big(\mathcal{M}^{(f)}\Big) \Big\} \,, \tag{3.16}$$

where $[F] = \{1, \ldots, F\}$ and L is the length of the hand model, making the term scale invariant. As we want to maximize the minimum distance between two markers, this term aims to minimize the negative of the overall minimum distance over all F frames.

Combining these two energy terms integrates the desired numerical stability and geometric feasibility properties of the marker layout in a single objective function. The results of minimizing the two energy terms and their weighted sum is illustrated in Figure 3.9. In this particular example, the condition energy places two markers close to each other, because the linear system for the subspace parameters is overspecified by the number of markers, which means that close-by markers do not corrupt the matrix conditioning. Combining the two energies improves the resulting layout. We use weights $w_1 = 0.1$ and $w_2 = 100$ in our experiments. In the following, the optimization of the objective function (3.14) is detailed.

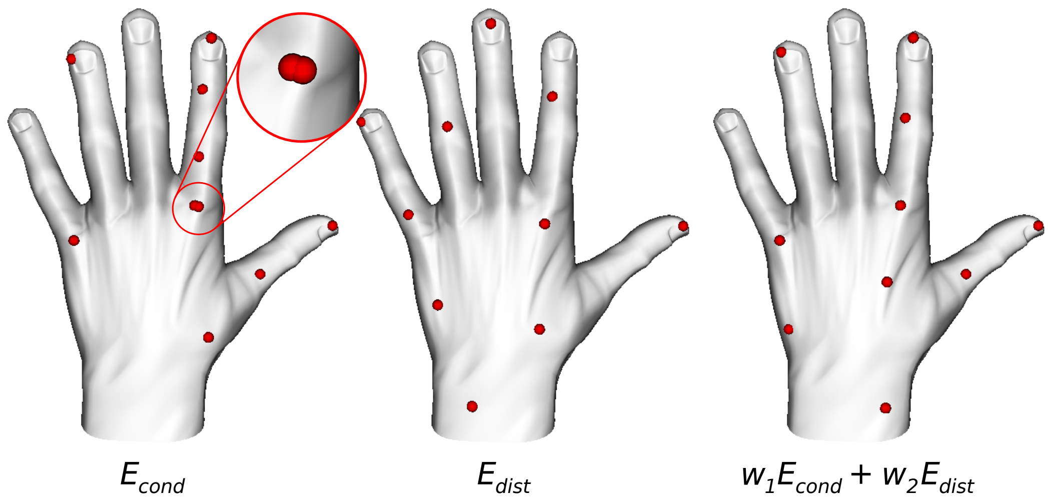


Figure 3.9: Example layouts with 10 markers for the objective function terms. The input data is a precision grasp, where mostly the index finger and thumb are in motion. Left: when optimizing only for the numerical stability term E_{cond} markers can be placed in close proximity, which is geometrically impractical. Center: optimizing for the geometric distance term E_{dist} results in spatially distant markers, but the layout does not capture the analyzed hand articulations. Right: a weighted combination of the two terms results in a layout that is both numerically stable and geometrically feasible.

3.4.2 Marker PSO

We find reduced marker layouts by minimizing the objective function (3.14) using particle swarm optimization (PSO, Kennedy and Eberhart 1995, 2001). PSO is a stochastic metaheuristic for finding global optima of arbitrary objective functions without the need for prior knowledge or assumptions about the optimized problem. The method has recently found widespread application and success in the context of visual hand tracking (Oikonomidis et al. 2011a, Qian et al. 2014, Sharp et al. 2015). Our use of PSO for marker placement aims to overcome the issues of suboptimal local minima often associated with non-global or greedy approaches.

In the PSO method, an optimal solution to a given problem is found by iteratively updating and evaluating candidate solutions, or solution hypotheses. A large set of such hypotheses is managed as a swarm or population of particles, each of which has an associated position $\mathbf{x}_t$ and velocity $\mathbf{v}_t$ in the solution domain of the objective function at iteration t. Each particle keeps track of its local previous best position $\bar{\mathbf{x}}_{\text{par}}$ in the solution domain and the population keeps track of the global optimum $\bar{\mathbf{x}}_{\text{pop}}$ across all particles. In each iteration of the PSO process, the velocity of every particle is updated such that the particle is attracted to the local and global optima in addition to moving along its own inertia. The local and global optima are updated after each particle movement by evaluating the objective function at the

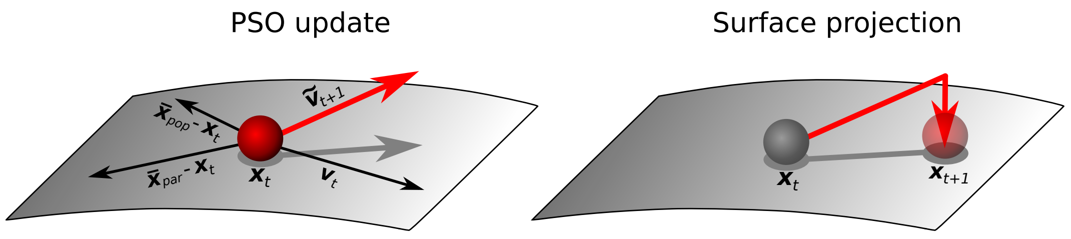

PSO update
Surface projection
$\bar{\mathbf{x}}_{pop} - \mathbf{x}_t$
$\tilde{\mathbf{v}}_{t+1}$
$\bar{\mathbf{x}}_{par} - \mathbf{x}_t$
$\mathbf{x}_t$
$\mathbf{v}_t$
$\mathbf{x}_t$
$\mathbf{x}_{t+1}$

Figure 3.10: Illustration of a PSO update for one marker position. First, the new velocity $\tilde{\mathbf{v}}_{t+1}$ of the marker is computed as a weighted linear combination of the vectors towards the particle's local optimum $\bar{\mathbf{x}}_{\text{par}} - \mathbf{x}_t$, the population's global optimum $\bar{\mathbf{x}}_{\text{pop}} - \mathbf{x}_t$ and the particle's current velocity vector $\mathbf{v}_t$. This update can send the marker off the surface of the hand model due to the curvature of the model surface. Therefore, in a second step, the new position $\mathbf{x}_{t+1}$ is computed by projecting back onto the surface. The velocity $\mathbf{v}_{t+1}$ is then recomputed accordingly.

new particle position. Finally, the solution of the PSO process is the global optimum achieved after a given number of iterations or after convergence of the optimum value.

In our application, the solution domain of the objective function is the domain of marker layouts $\mathcal{M}$. To map this to the PSO scheme, we define a particle at iteration t as the stacked vector of k marker positions $\mathbf{x}_t \in \mathbb{R}^{3k}$ of the candidate solution. We further modify the generic PSO scheme such that the 3D positions within each particle are constrained to the surface of the hand model. Specifically, after every particle update we project each marker position in $\mathbf{x}_t$ onto its spatially closest vertex in the set $\mathcal{V}'$ of preselected feasible positions on the hand model. The new position $\mathbf{x}_{t+1}$ of a particle is determined by computing its new velocity $\mathbf{v}_{t+1}$ and translating along this vector. To this end, we first compute the standard PSO velocity update as

$$\tilde{\mathbf{v}}_{t+1} = w \cdot (\mathbf{v}_t + c_1 \cdot r_1 \cdot (\bar{\mathbf{x}}_{\text{par}} - \mathbf{x}_t) + c_2 \cdot r_2 \cdot (\bar{\mathbf{x}}_{\text{pop}} - \mathbf{x}_t)), \tag{3.17}$$

where w is a weight determining the overall step length of the update, c_1 and c_2 are importance weights for the local and global attractors respectively, and r_1 and r_2 are uniformly distributed random numbers in $[0, 1]$. Due to the curvature of the hand model surface, applying this linear update to the current particle position can cause the markers to stray from the surface. To counteract this, we project the updated marker positions back onto the permissible regions defined by vertices $\mathcal{V}'$, which we denote by a projection operator $\Pi_{\mathcal{V}'}$. The final particle position update is therefore

$$\mathbf{x}_{t+1} = \Pi_{\mathcal{V}'}(\mathbf{x}_t + \tilde{\mathbf{v}}_{t+1}) \,. \tag{3.18}$$

After this, the new particle velocity is computed as $\mathbf{v}_{t+1} = \mathbf{x}_{t+1} - \mathbf{x}_t$. Figure 3.10 illustrates the surface-constrained PSO update.

Similar to Oikonomidis et al. (2011a), we perturb one randomly chosen marker position in 50% of the particles once in every third iteration, and use the weights $c_1 = 2.8$, $c_2 = 1.3$ and $w = 2/|2 - \psi - \sqrt{\psi^2 - 4\psi}|$ with $\psi = c_1 + c_2$. We use a total

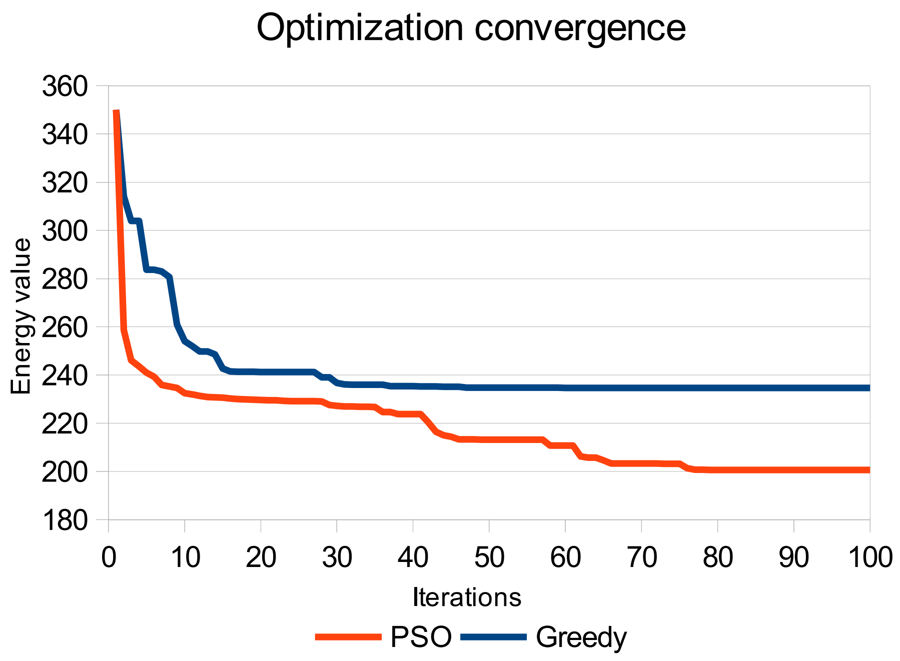


Figure 3.11: Example for the energy minimization convergence of our PSO method compared to a greedy approach with identical initialization. While the greedy approach converges to a suboptimal local minimum after about 50 iterations, our stochastic optimization minimizes the energy faster and achieves a better result.

of 1000 particles, perform 100 PSO iterations and use between 3 and 10 keyframes depending on the input hand motion trajectory. Using this method, we can find reduced marker layouts that optimize the objective function (3.14) and as a result are numerically stable and geometrically feasible.

3.5 Results

We evaluated the convergence properties of our marker PSO scheme and the motion reconstruction accuracy of the marker layouts generated using our method in a varied set of evaluation trials. The hand movements involved in these trials included a variety of grasping and other manual interaction movements, as well as generic finger movements and gestures. In the performed trials, we measured runtime statistics and average per-vertex errors of the reconstructed hand motions compared to the ground truth input. For proper evaluation of the accuracy of our approach, the input motions being reconstructed were not contained in the database used to generate the subspace model. As our reduced marker sets are optimized to represent only rotational DoFs of the hand articulation, an initial estimate for the global position of the hand is given by a fixed anchor marker on the forearm.

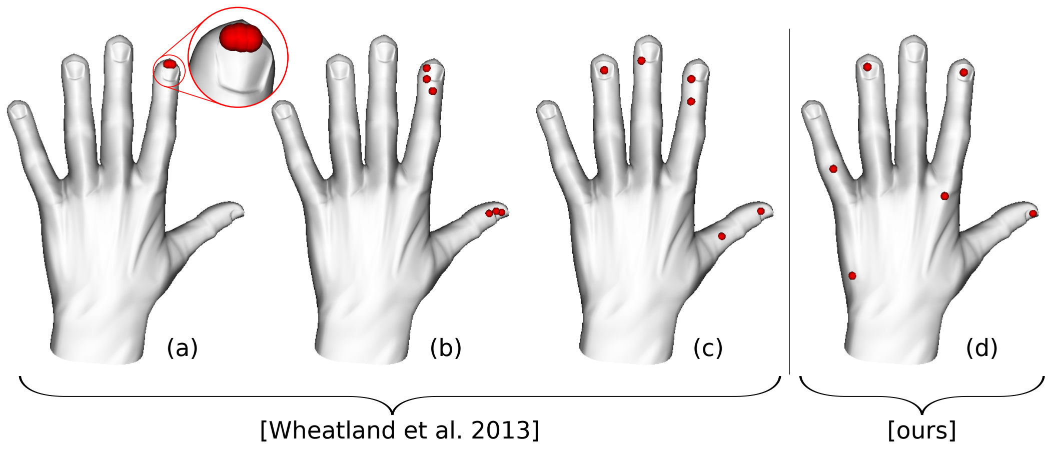


Figure 3.12: Six marker layout generation for precision grasp movements using the method of Wheatland et al. (2013) and our approach. In (a) the complete set of preselected vertices $\mathcal{V}'$ is used as the base marker set, which causes the selected markers to cluster at the index fingertip, as it exhibits the most movement. In (b) a random subset with 5% of $\mathcal{V}'$ is used as the base marker set, which leaves 3 candidate positions per joint. In this case the markers cluster around the index and thumb tips. In (c) 1% of $\mathcal{V}'$ is used, which leaves one candidate position per joint. The resulting marker set is distributed among the most active joints in the input motion. In (d) our approach generates a marker layout from the complete set $\mathcal{V}'$ based on the DoFs of our subspace model.

The convergence properties of our PSO-based marker layout optimization were evaluated by comparing it to a more straightforward greedy approach. For this, we adapted the farthest point optimization scheme of Schlömer et al. (2011) to find the marker subset of the initial vertex set $\mathcal{V}'$ that minimizes the objective function (3.14). Briefly stated, this method first iteratively selects the next best vertex as a marker position that reduces the objective value until the desired number of markers has been placed. Then, this greedy process is repeated such that each selected marker position is replaced by the next better remaining vertex position, until no more substitutions can be done to improve the objective value. This is already a more sophisticated approach than the greedy methods for constraint selection used in (Loper et al. 2014, Thiery et al. 2012) and can therefore serve as an upper bound for the effectiveness of such methods. Figure 3.11 compares this greedy approach with our PSO-based one with identical initialization and shows that our method converges faster and achieves better objective values. The runtime for our PSO method varies between 5 and 10 seconds for 100 iterations, depending on the number of keyframes (up to 10). For the same problem setup, the greedy approach takes between 45 seconds and 3 minutes to converge.

A comparison of our marker layout optimization method with the marker subset selection approach of Wheatland et al. (2013) is shown in Figure 3.12. A crucial aspect to note regarding this comparison is that the two methods are based on different marker layout generation paradigms. While Wheatland et al. (2013) select

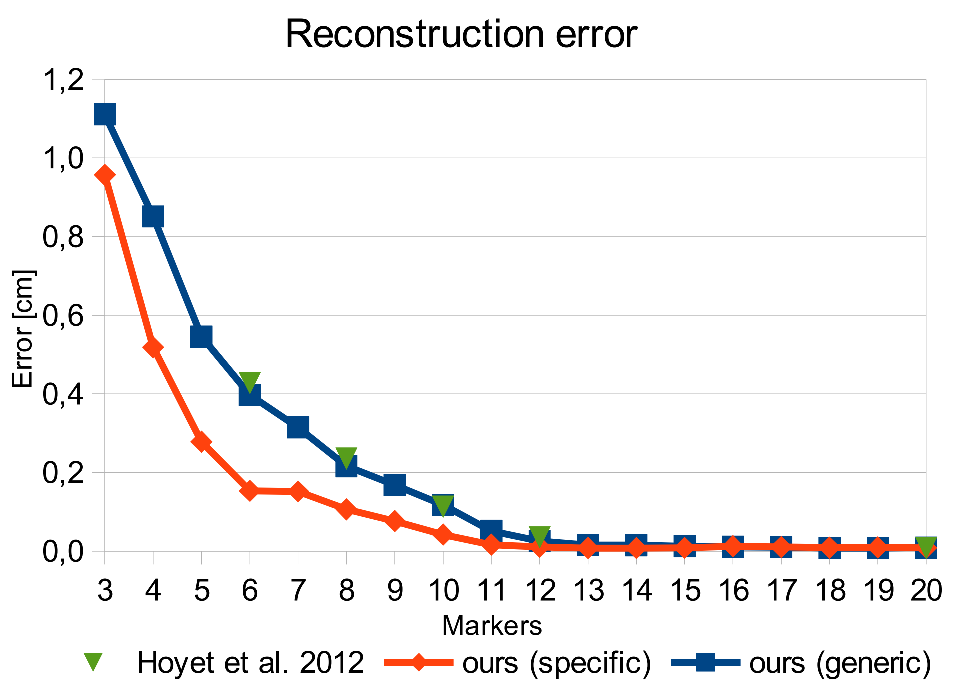


Figure 3.13: Average reconstruction error using marker sets of varying sizes for grasping motions. Using marker layouts specifically generated based on grasping input motions the reconstruction error is lower than when using a layout based on generic motions. The manually selected marker layouts of Hoyet et al. (2012) produce similar results as our automatically generated generic layouts.

the most influential markers in an initial base marker set, our method generates marker layouts more freely within the dense set of preselected vertices $\mathcal{V}'$ (see Section 3.4). Figure 3.12 shows that the results of the subset selection method are strongly influenced by the choice of the base marker layout. As the method of Wheatland et al. (2013) is based on computing an importance ranking for the base markers based on their positional trajectories, the selected marker layouts are clustered around the areas of the hand that move the most in the considered hand motion. In contrast, our method is sensitive to the hand kinematics and the subspace model employed in our approach, which produces layouts that are well-suited for subspace-constrained IK, as opposed to the data-driven regression approach of Wheatland et al. (2013).

To assess the suitability of the marker layouts generated by our method for motion reconstruction, we compare the reconstruction error of differently obtained marker layouts in Figure 3.13. The testbed of this evaluation is a set of grasping motions based on the grasp taxonomy of Cutkosky (1989). We generated two different types of marker sets with varying sizes – a specific type based on grasping input motions, and a generic type based on general gestures and hand articulations. Additionally, we compare with the manually selected marker layouts of (Hoyet et al. 2012, Figures 4 and 8), who also performed motion reconstruction based on constrained IK. These

Method	Average error	Maximum error
Subspace IK	0.89 cm	2.1 cm
Standard IK	1.79 cm	7.9 cm

Table 3.1: Average and maximum reconstruction error using subspace-constrained IK and standard IK with a 4 marker layout generated for a variety of manual interaction motions. While the standard method can deviate by almost 8 cm, the subspace method achieves adequate results consistently.

manually selected layouts produce similar results as our automatically generated generic layouts. The reconstruction error is lower when using the grasp-specific marker layouts than generic ones. In particular, to achieve a reconstruction error below 2 mm, a specific layout generated by our method only requires 6 markers, whereas generic layouts require 9 markers or more. Examples for our generated layouts are given in Figure 3.14.

We verified the accuracy and generalization capability of the subspace-constrained IK motion reconstruction based on marker layouts produced by our method by comparing its average reconstruction error to the error when using standard IK. Table 3.1 shows the average and maximum errors for a variety of manual interaction motions using standard IK and subspace IK and the 4 marker layout shown in Figure 3.7. The improvement of the subspace method over the standard method ranges between 9 mm to almost 6 cm. The overall error produced by our layered IK scheme using the subspace prior for initialization produces more accurate results for sparse marker layouts than the standard IK approach.

Figure 3.14 shows some examples for reduced marker layouts computed by our approach for various different movements. The results show that markers are preferentially placed in areas that have the most involvement in the considered hand motion. If the motions contain more varied articulations for specific fingers over others, these fingers will receive more markers, as the low-frequency details of the remaining markers are not influenced by as many subspace DoFs. In the third row of Figure 3.14, the input motion involves all fingers and the reduced marker layout accordingly distributes markers across all of them.

3.6 Discussion

We have presented a method that automatically computes reduced marker layouts for optical motion capture using subspace-constrained inverse kinematics motion reconstruction. Our marker layout optimization method minimizes an objective function that jointly measures the numerical stability and geometric feasibility of the reduced marker configuration. The objective function is minimized using a specialized surface-constrained particle swarm optimization scheme, which stochastically explores the solution space of feasible marker configurations on the surface of an

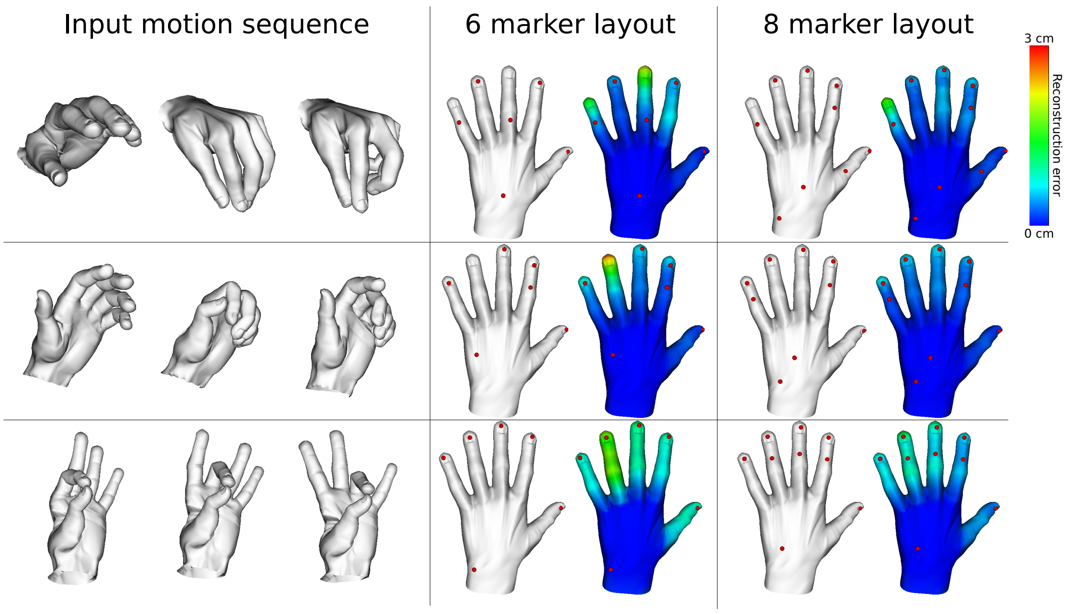


Figure 3.14: Reduced marker layouts for some example motions. First row: precision grasp motion involving multiple fingers. Second row: power grasp motion of a small object. Third row: sequence in which the thumb touches all the other fingers. The marker layouts are optimized to allow for accurate reconstruction of the input motion. Markerless fingers tend to have slightly larger reconstruction errors, however they still move in correlation to the marker-constrained fingers due to the subspace approach.

animated hand model. We showed that the resulting marker layouts are suitable for solving the subspace-constrained inverse kinematics problem for motion reconstruction from sparse input. Additionally, the optimized marker layouts are specific to the type of hand motions that should be expressed with and recovered from the sparse marker data. The marker sets are well-suited for practical use as they are intuitive and scale well with reduced resolution of the mocap system.

Our method makes it possible to generate marker layouts that are fine-tuned to the parameters of a given mocap setup. If there is a limitation to the number of markers that can be used in the mocap setup, our method computes the optimal placements for the given number of markers that allows for realistic motion reconstruction that is also rich in expressiveness. An insight provided by our work is that it is sufficient for high quality motion reconstruction to place individual markers on the hand that correspond to low-dimensional control parameters of hand articulations. For instance, to track grasping motions with high quality using our method, it is sufficient to only place one marker on the thumb, index finger, pinky finger and wrist. The subspace based reconstruction will plausibly interpolate the movements of joints that are not immediately constrained by markers.

Limitations of our approach include the stochastic nature of the particle swarm optimization and the need for parameter tweaking. Another drawback of our subspace-oriented method is that while it produces good results for specific hand movements, it does not necessarily provide a general-purpose marker layout result that can be used for all types of motions and produce high-quality results. The marker placement as well as the motion reconstruction are limited by the subspace priors employed. However, given prior knowledge of the motions intended to be tracked, our method produces accurate and robust results. It is conceivable that our marker placement optimization approach can be transferred to other mocap applications, such as facial or full body performance capture. Beyond marker placement, our approach could be used generally to identify salient regions in articulated bodies, which could be of interest for different avenues of motion detection and reconstruction.

Chapter 4

Data-driven visual tracking

Despite the effectiveness of commercial tracking solutions like optical motion capture systems, their drawbacks can outweigh their advantages in many cases. They usually require an elaborate setup in especially constructed motion capture locations, their usage is complex and their cost prohibits widespread consumer-level deployment. In contrast, the recent proliferation of consumer-grade commodity depth sensors, pioneered by the *Microsoft Kinect*, has opened an alternative avenue for affordable tracking solutions. These sensors non-invasively capture both color images and depth maps in real-time and consequently alleviate common problems of visual motion tracking, like occlusions and depth ambiguities. However, because the data obtained from these low-cost sensors can exhibit various artifacts, visual tracking techniques must implement mechanisms to cope with these effects. One such mechanism is to use data-driven approaches, which facilitate robust visual tracking by utilizing prior knowledge from a ground truth database.

In this chapter, a data-driven, appearance-based method for visual hand tracking using a color glove is explored. This approach is based on detecting a color glove worn by the user in the sensor input data and using a nearest neighbor search in an image database to retrieve the closest matching image and corresponding hand posture and coarse rotation. Our approach, following that of Wang and Popović (2009), is a low cost real-time system that can provide accurate hand posture and position estimation. A simplification of their algorithm along with the addition of a *Microsoft Kinect* camera distinguishes both approaches. The database used to perform the data-driven hand posture estimation was built by capturing a variety of hand movements using an optical motion capture system (see Chapter 3).

The particular goal in our work (Schröder et al. 2012) was to create a low-cost real-time hand tracking and pose estimation system to control anthropomorphic robotic hands (*Shadow Robot Company*). The system we developed in this work allows for the fast generation of real hand posture data that can be used in, for example, teaching by demonstration scenarios or in the direct control of anthropomorphic robotic hands. Whereas the approach of Wang and Popović (2009) was concerned with producing accurate hand postures, our primary focus is on accurately estimating the pose

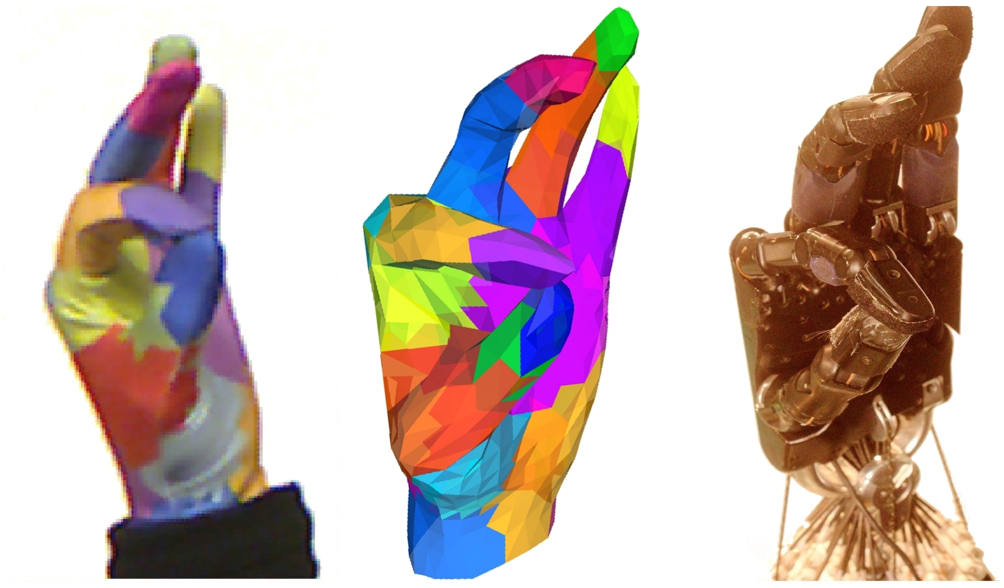


Figure 4.1: The color glove is used to facilitate hand posture estimation. The resulting posture and pose are used to control an anthropomorphic robot hand in real-time.

(position and orientation) of the hand. In order to perform robot actuation in tasks such as grasping, controlling the pose of the hand takes on greater importance than having an accurate posture, especially when using, as we are, compliant robot hands. The Kinect camera provides a 3D point cloud along with color information, which we use to initially place the hand and then optimize this position using a color-sensitive iterative closest point-to-triangle (ICP-T) algorithm.

4.1 Related work

There are many applications for hand tracking in the field of robotics and manual interaction research, and the most commonly utilized methods for tracking are based on commercial solutions like marker-based optical mocap systems or data gloves (Maycock et al. 2011). Optical tracking systems that employ markers have been used to capture highly accurate kinematic movement data of the hand (Maycock et al. 2011, Lee and Tsai 2009), but these approaches can suffer from occlusions of one or more markers, which can occur especially when grasping and manipulating objects, and have the disadvantage that usually an expensive and large optical system is required. An alternative is to use data gloves that measure joint angles directly and this method was extensively used to control robot hands in the past (Fischer et al. 1998, Griffin et al. 2000, Turner 2001). However, data gloves with embedded sensors can be quite bulky to wear and thus impede natural movements. Furthermore, they generally do not provide position and orientation information, and the mapping of raw sensor values to joint angles requires non-trivial calibration procedures (Steffen et al. 2011).

The difficult problem of visual bare-hand tracking has been extensively examined but often deemed too computationally complex for real-time applications (Sudderth et al. 2004, de La Gorce et al. 2008, Hamer et al. 2009). It was not until significant advances in hardware have made the problem possible to handle (Oikonomidis et al. 2011a). As a compromise between approaches based on markers or data gloves and bare-hand tracking approaches, Wang and Popović (2009) devised a method for real-time hand tracking and posture estimation using a color glove. A camera captures images of the glove and after segmentation and normalization the closest matching image in a large database, along with its unique posture, is found. A database of synthetically generated images is used for directly matching the detected features instead of performing a complex search in the hand configuration space. This data-driven approach simplifies the hand tracking problem considerably and is able to provide good posture and orientation information. Use of a color glove improves detection, feature extraction and speed over approaches that use bare hands. In (Wang et al. 2011) the authors adapted their method to facilitate markerless 6D hand pose estimation using a dual camera setup in a CAD application scenario. To cope with the drawbacks of using only images of bare hand silhouettes instead of color images, several restrictions to the user's freedom of movement had to be introduced in this work. While these restrictions were acceptable in a CAD context, they are not in our case. We use the color glove-based work by Wang and Popović (2009) as the foundation of our hand tracking approach, however we have made a number of simplifications to their algorithm and have introduced a Kinect camera in order to be able to much more precisely control the absolute position and orientation of the hand in 3D space. This is crucial if robust control of anthropomorphic robot hands is to be achieved.

The task of hand tracking for robot control was previously dealt with in (Do et al. 2009), where the authors employed a data-driven approach for grasp recognition and hand pose estimation and mapped the estimation results to a humanoid robot. In their method (Romero et al. 2010) features extracted from bare hand images are matched in a synthetically generated database to obtain the posture and orientation of the hand. This represents a discriminative hand pose estimation approach, whereas our approach combines discriminative and generative estimation by fitting a virtual hand model to the Kinect point cloud after obtaining the initial posture estimate from a database, making use of the glove's unique color pattern in both estimation steps.

4.2 Color glove hand tracking

The color glove was designed by Wang and Popović (2009) with a pattern of 20 patches in 10 distinguishable colors (see Figures 4.1 and 4.2). We make use of its distinctive pattern to efficiently find the current image's closest match in the database and thus retrieve the hand posture and a coarse estimation of the rotation of the hand. Aligning the color image with the depth values provided by the Kinect camera

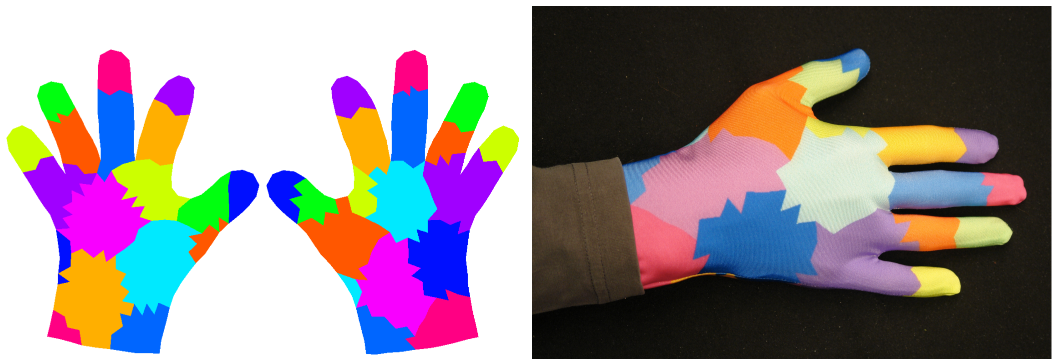


Figure 4.2: The glove pattern on the left, the actual glove on the right (glove pattern courtesy of Wang and Popović (2009)). Images from (Schröder 2011).

allows us to accurately estimate the global position and orientation of the hand. Combining the posture (joint angles) and the pose (position and orientation) allows us to control an anthropomorphic robot hand.

Our approach relies on a database populated with images of a virtual color glove along with their associated posture and rotation information. The depth information from the Kinect camera is currently only used to optimize the pose of the hand and therefore is not contained in the database. The first step was to capture natural realistic postures with a mocap system and then to map these onto a kinematic hand model. As in Chapter 3, the model used has 20 joint angles, specifically 15 flexion angles and 5 abduction angles for each of the finger bases and the wrist joint (Maycock et al. 2011). This kinematic hand model matches the kinematics of the shadow robot hand well and makes the task of mapping joint angles to the robot easier. Ground truth data was captured using a Vicon motion tracking system (Maycock et al. 2010) to track reflective markers placed on a human hand as it performed various movements. After converting the raw positional Vicon data into joint angles, we subsampled the postures using a low-dispersion sampling method (Wang and Popović 2009) and constructed several task-specific posture databases (see Sec. 4.3), varying the number and type of postures contained within them. The low-dispersion sampling of the database postures produces a subset of the original database that contains only hand postures that are maximally different from each other. This sampling is implemented as an iterative greedy approach where in each iteration, the next farthest data point in the hand posture data is selected.

Each database contains a number of posture entries. The joint angles for each posture were used to animate a virtual hand model textured with the glove's color pattern. The hand model was then rendered from multiple camera views and each database posture entry was indexed by these different views. The camera positions were produced by uniformly sampling a virtual sphere surrounding the hand model, which was accomplished by performing a uniform subdivision of an icosahedron to approximate the surrounding sphere and placing the cameras at its vertices. At each position, the camera was rotated around the optical axis in several steps to encode the rotation of the color glove in the image plane. We used 42 camera positions with

18 rotations around the optical axis each, resulting in a total of 756 views per posture in all of our databases. Finally, for each posture and view entry, a number of tiny images of various sizes were created. Tiny images (Wang and Popović 2009) are simply normalized downscaled color-classified images of the color glove. We generated several tiny image sets using image dimensions ranging from 8×8 pixels up to 64×64 pixels. These are used to improve efficiency when searching for the best match in the database for the current camera image of the hand.

Figure 4.3 provides an overview of our system. The three main parts of the system that produce as output a final hand posture and pose are introduced in the following sections.

4.2.1 Color-labeled point cloud

The data obtained from the Kinect sensor is processed to create a 3D point cloud containing the positional and color information of the detected glove. The camera parameters of the Kinect's color and depth cameras were obtained in a stereo calibration procedure and are used to perform an RGBD-mapping, which maps color values to the pixel coordinates of the depth image D. After applying this mapping, the color glove is detected in the mapped image M and using a lookup table-based color segmentation procedure a color labeled image L is created. The color segmentation is performed in the YUV color space to reduce sensitivity to changes in lighting. Each of the glove's ten unique colors are labeled with numbers 1 to 10 and colors that do not belong to any known color class are labeled 0. Now we are in a position to compute a color-labeled 3D point cloud

$$C = \left\{ (\mathbf{p}_{ij}, L_{ij}) \mid L_{ij} \neq 0, \mathbf{p}_{ij_z} \in [z_{min}, z_{max}] \right\}, \tag{4.1}$$

where $\mathbf{p}_{ij}$ is the 3D position and L_{ij} the color class label for every glove pixel (i, j). To create C, we first ensure that only pixels in L that are not 0 are considered. The (x, y, z) position is computed for these pixels and in order to minimize sensitivity to color segmentation outliers only those whose z world coordinates reside in the working volume $[z_{min}, z_{max}]$ are kept.

4.2.2 Database lookup

The output from the previous step is a labeled point cloud C and from this we create a new labeled image by masking C with the labeled image L from the RGB camera. Using a bounding box, we extract the hand from C and from this we create a set of tiny images (with dimensions ranging from 8×8 pixels up to 64×64 pixels). We need to be able to quickly compute the similarity between a tiny image in the

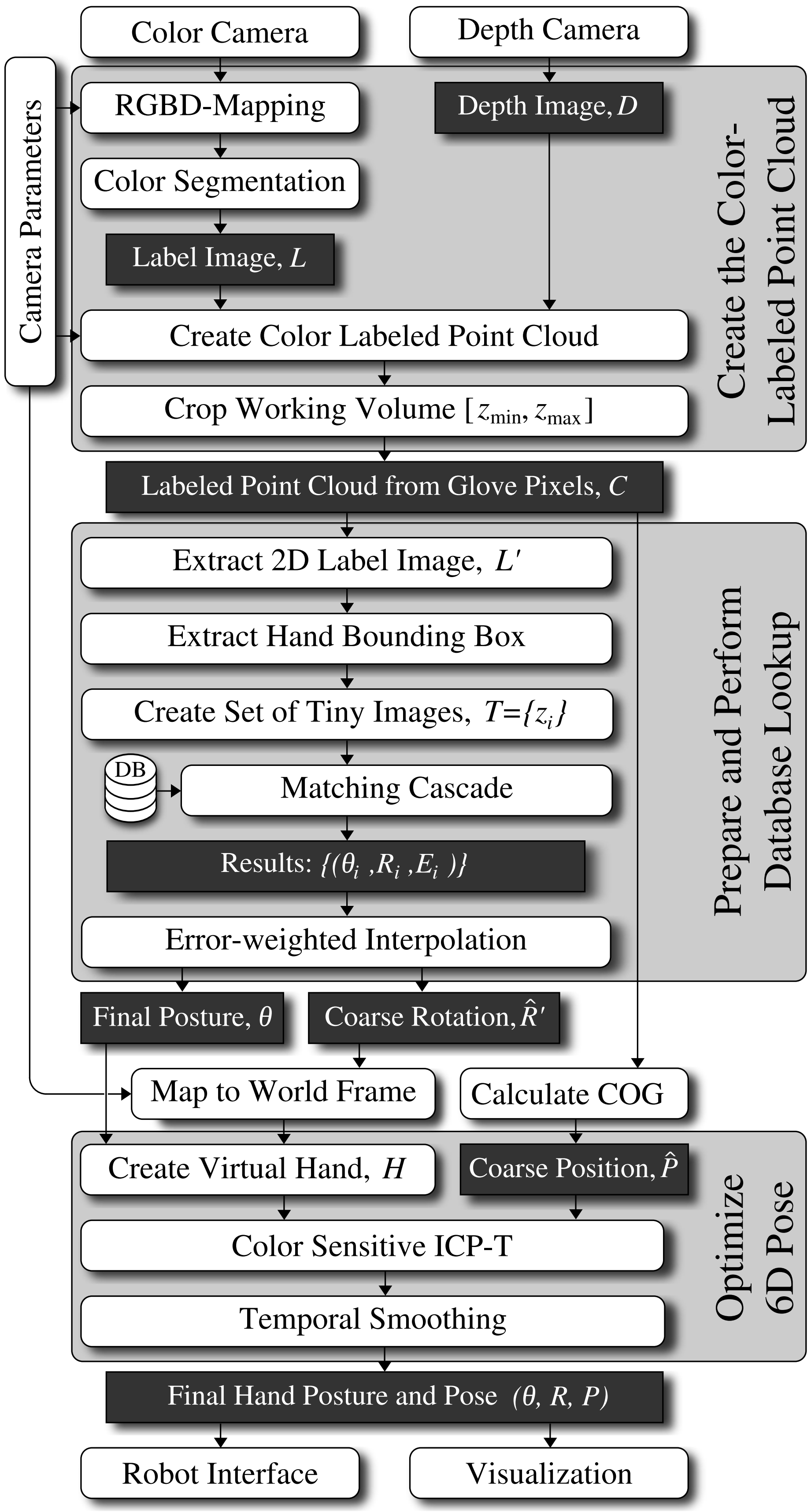
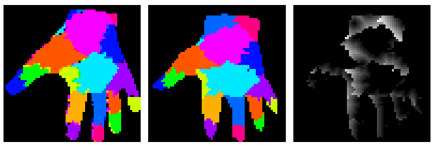
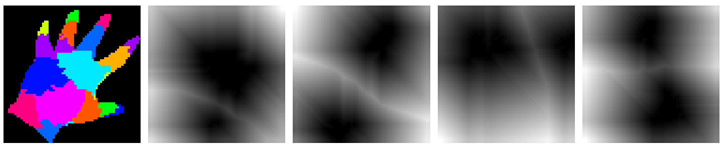

Color Camera
Depth Camera
Camera Parameters
RGBD-Mapping
Depth Image, D
Color Segmentation
Label Image, L
Create Color Labeled Point Cloud
Crop Working Volume $[z_{\min}, z_{\max}]$
Create the Color-Labeled Point Cloud
Labeled Point Cloud from Glove Pixels, C
Extract 2D Label Image, L'
Extract Hand Bounding Box
Create Set of Tiny Images, $T=\{z_i\}$
DB
Matching Cascade
Results: $\{(\theta_i, R_i, E_i)\}$
Error-weighted Interpolation
Prepare and Perform Database Lookup
Final Posture, θ
Coarse Rotation, $\hat{R}'$
Map to World Frame
Calculate COG
Create Virtual Hand, H
Coarse Position, $\hat{P}$
Color Sensitive ICP-T
Temporal Smoothing
Optimize 6D Pose
Final Hand Posture and Pose (θ, R, P)
Robot Interface
Visualization

Figure 4.3: Overview of the system.



(a) Tiny image distance



(b) Chamfer maps

Figure 4.4: Tiny image metric illustrated using 64×64 images. Figure 4.4(a) shows the distance between a query image and a database image. The distance is visualized by displaying the Hausdorff-like distance for each pixel. Figure 4.4(b) shows four of the ten chamfer maps generated for a tiny image. Images from (Schröder 2011).

database and one computed from the current camera image. To do this we use a Hausdorff-like distance (Huttenlocher et al. 1993) defined as

$$m(A, B) = \frac{1}{|G_A|} \sum_{a \in G_A} \min_{b \in C_a} \|a - b\|, \tag{4.2}$$

where $A = \{a_1, ..., a_n\}$ and $B = \{b_1, ..., b_n\}$ are the two images, G_A is the set of glove pixel coordinates in image A, C_a is the set of pixel coordinates in image B that have the same color as the pixel at $a \in G_A$ and $\| \cdot \|$ is the Euclidean norm. We used chamfering (Barrow et al. 1977), in contrast to Wang and Popović (2009) who employed similarity sensitive coding (Torralba et al. 2008), to approximate Euclidean distance transforms for every color class in the images, allowing for an efficient direct lookup-based calculation of the distance between two images. Figure 4.4 shows examples illustrating the the tiny image distance and the chamfer maps approximating the per-pixel Euclidean distance transforms of the glove's color classes.

To take advantage of the varying speed and accuracy properties of differently sized tiny images, we perform a multi-stage k nearest neighbor lookup for each sized tiny image computed from the current camera image. The set $T = \{t_i\}_i$ of tiny images is used as input to the matching cascade database lookup step (see Fig. 4.5) and is matched successively against all N database indices d_{ji}, $j \in \{1, \dots, N\}$. In the figure only 3 different tiny image sizes are used, i.e., $i \in \{1, 2, 3\}$. Initially, smaller tiny images, whose Hausdorff distance can be computed quickly, albeit yielding a lower accuracy, are used to efficiently preselect nearest neighbor candidates. At each iteration, larger tiny image matching, which is computationally more expensive but results in higher accuracy, is performed to find the k_i nearest neighbors in significantly smaller sets of candidate images (in the figure $k_1 = 5, k_2 = 3, k_3 = 2$). The output is created from the database indices that remain after the last cascade step (in the figure d_{23} and d_{N3}). This results in a set $\{(\boldsymbol{\theta}_j, \mathbf{R}_j, E_j)\}_j$, where $\boldsymbol{\theta}_j$ and $\mathbf{R}_j$

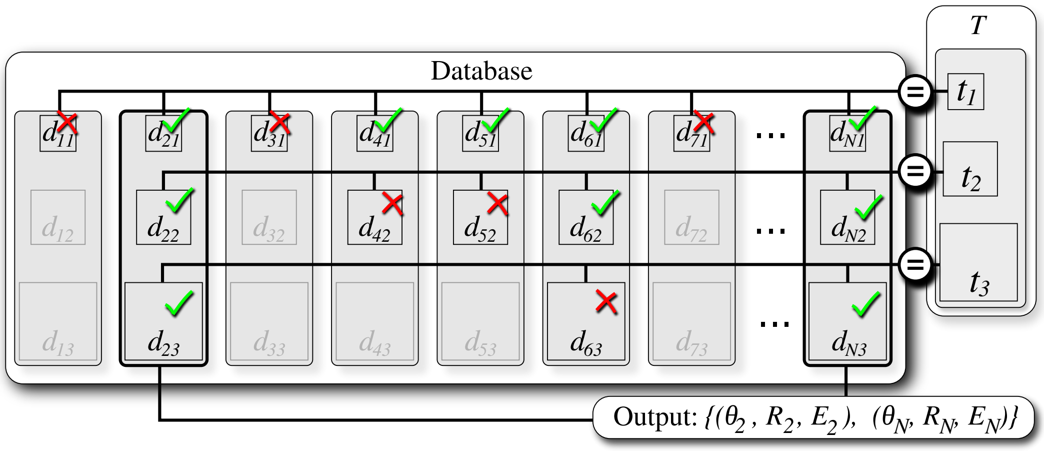

Database
T
t_1
t_2
t_3
d_{11} d_{21} d_{31} d_{41} d_{51} d_{61} d_{71} … d_{N1}
d_{12} d_{22} d_{32} d_{42} d_{52} d_{62} d_{72} … d_{N2}
d_{13} d_{23} d_{33} d_{43} d_{53} d_{63} d_{73} … d_{N3}
Output: $\{(\theta_2, R_2, E_2), (\theta_N, R_N, E_N)\}$

Figure 4.5: The set $T = \{t_i\}_i$ of tiny images is matched successively against all N database indices d_{ji}, $j \in \{1 \,..\, N\}$. The output is a 3-tuple set $\{(\boldsymbol{\theta}_j, \mathbf{R}_j, E_j)\}_j$, where $\boldsymbol{\theta}_j$, $\mathbf{R}_j$ and E_j are the estimated posture, rotation and matching error, respectively.

are the posture and rotation associated with the database index d_j, and E_j is the corresponding matching error in the last cascade step. This set $\{(\boldsymbol{\theta}_j, \mathbf{R}_j, E_j)\}_j$ is then interpolated, weighting the contributions of $(\boldsymbol{\theta}_j, \mathbf{R}_j)$ pairs with their matching error E_j to the tiny image computed from the current camera image.

4.2.3 6D pose estimation

The result of the database lookup is the 20-dimensional hand posture estimate, $\boldsymbol{\theta}$, along with a coarse estimate of the orientation of the hand towards the camera, $\hat{\mathbf{R}}$. We compute the final estimate of the hand's 6D pose (rotation $\mathbf{R}$ and position $\mathbf{p}$) by aligning a virtual 3D model of the hand to the observed point cloud. For this we perform a color-sensitive Iterative Closest Point-to-Triangle (ICP-T) computation, where the relative rigid transformation between the point cloud C and the surface of the virtual hand model H is computed and iteratively refined. By performing a color-sensitive closest point search, the observed points are only matched to model surface patches with corresponding colors. As a result, the color patterns in the observation point cloud are matched to the same color patterns on the surface of the virtual hand model during the ICP-T process. This ensures a plausible alignment of the model to the sensor data even if the posture estimation from the previous step does not fit perfectly, which cannot be accomplished when using only positional information.

As a preparation for the alignment process, the coarse rotation estimate $\hat{\mathbf{R}}'$ is mapped to the world coordinate frame and the center of gravity (COG) of C is computed. This yields a coarse pose estimate $(\hat{\mathbf{R}}, \hat{\mathbf{p}})$. We represent this pose estimate as a 4×4 transformation matrix $\hat{\mathbf{T}}$. The virtual hand model H is initialized by animating it according to the joint angles $\boldsymbol{\theta}$ using linear blend skinning (Jacka et al. 2007) and transforming it using $\hat{\mathbf{T}}$. In each iteration of the color-sensitive ICP-T, we iterate over the points in C and for each point find the closest point on the surface of H of corresponding color. H is textured with the color glove pattern, so the color class label of each triangle in the model is known. To find the surface point closest to $(\mathbf{p}_{ij}, L_{ij}) \in C$, we iterate through all triangles in H with the same color class L_{ij} and store the surface point $\mathbf{p}_{ij}^H$ with the minimal point-to-triangle distance (Schneider and Eberly 2002). We perform back-face culling beforehand to exclude triangles directed away from the sensor camera. Based on the set of corresponding points $\{(\mathbf{p}_{ij}^H, \mathbf{p}_{ij})\}_{i,j}$ that results from the closest point search, we calculate the 4×4 relative transformation matrix $\mathbf{T}_k$ from H to C in ICP-T iteration k using a common approach based on eigenvector analysis and quaternions (Horn 1987). The transformation $\mathbf{T}_k$ is then applied to the virtual hand model H, moving it towards the target point cloud C. After this, the alignment error is given by the mean of the new point-to-triangle distances. The ICP-T process is repeated until the alignment error converges. After convergence, the final hand pose estimate $\mathbf{T}_H$ is given by the product of the initial coarse transformation matrix and all K ICP-T iteration matrices:

$$\mathbf{T}_H = \left(\prod_{k=K}^{1} \mathbf{T}_k \right) \hat{\mathbf{T}} = \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{pmatrix}. \tag{4.3}$$

The output of our hand tracking system is the final hand posture and pose estimate $(\boldsymbol{\theta}, \mathbf{R}, \mathbf{p})$. To reduce jitter, we smooth the posture and pose estimation by interpolating with estimations from the three previous frames, which still allows for responsive tracking of dynamic movements.

4.3 Results

In this section we evaluate the performance of our system in three main test scenarios. First we consider the accuracy of the posture and pose, then the effect of various different matching cascades on the database retrieval accuracy and the runtime efficiency of the database lookup, and finally the system runtime speed versus position error with various different subsampling factors for the ICP-T algorithm.

We use four different databases in the following tests: two 2-posture databases, $\text{DB}_{\text{open+pinch}}$ and $\text{DB}_{\text{open+power}}$ containing an open hand and a pinch and power grasp, respectively, a 3-posture database, $\text{DB}_{\text{open+pinch+power}}$, containing all three postures, and a database with 20 postures, $\text{DB}_{\text{20 postures}}$, containing the previous 3 postures plus 17 random movement postures. Ground truth data was created by capturing five different movement trajectories: two containing a pinch grasp, an open hand and some movements (Tr_1 and Tr_2) and three containing a power grasp, an open hand and some movements (Tr_3, Tr_4 and Tr_5). These movements were tracked by our system and their computed postures and poses became the ground truth for the subsequent tests. Re-playing these movement trajectories allowed us to create

Posture Error	Tr_1	Tr_2	Tr_3	Tr_4	Tr_5
$\text{DB}_{\text{open+pinch}}$	1.34	0.95	–	–	–
$\text{DB}_{\text{open+power}}$	–	–	3.42	2.60^{*}	1.60
$\text{DB}_{\text{open+pinch+power}}$	4.89	4.04	3.23	2.57	2.17
$\text{DB}_{\text{20 postures}}$	18.0^{**}	10.30	7.40	8.91	8.99

Rotation Error	Tr_1	Tr_2	Tr_3	Tr_4	Tr_5
$\text{DB}_{\text{open+pinch}}$	4.98	4.06	–	–	–
$\text{DB}_{\text{open+power}}$	–	–	3.11	2.69^{*}	3.10
$\text{DB}_{\text{open+pinch+power}}$	5.59	4.47	3.03	3.46	4.59
$\text{DB}_{\text{20 postures}}$	11.67^{**}	6.28	3.52	4.39	4.79

Position Error	Tr_1	Tr_2	Tr_3	Tr_4	Tr_5
$\text{DB}_{\text{open+pinch}}$	3.90	5.41	–	–	–
$\text{DB}_{\text{open+power}}$	–	–	4.72	3.88^{*}	4.41
$\text{DB}_{\text{open+pinch+power}}$	4.64	6.01	4.65	4.04	4.82
$\text{DB}_{\text{20 postures}}$	7.33^{**}	7.46	4.81	4.96	6.11

Figure 4.6: Posture and pose estimation errors for various ground truth experiments. Posture and rotation errors are given in deg, position errors are given in mm. For all ground truth trajectories, only databases containing the respective grasping postures were used. The trajectory of the errors labeled with superscript * is shown in detail in Figure 4.7, the trajectory of the errors labeled with superscript ** is shown in detail in Figure 4.8.

synthetic ground truth Kinect data (synthetic color and depth images), which was then used as input for our hand tracking system. Gaussian noise, whose variance was scaled with the gradient intensity image, was added to each synthetic depth frame in order to simulate real data. We were then able to directly calculate the error between the ground truth postures and poses and their estimation.

4.3.1 Posture and pose estimation

To evaluate the estimation quality of our system, we computed posture, rotation and position errors for all ground truth trajectories using all databases. Figure 4.6 shows the average errors for all experiments. Databases were only used if they contained the main grasp (pinch or power) performed in a particular movement trajectory. In most cases, the errors are lowest when using the respective small grasp-specific databases ($\text{DB}_{\text{open+pinch}}$ or $\text{DB}_{\text{open+power}}$) and highest with a large database containing several hand postures in addition to both grasping postures ($\text{DB}_{\text{20 postures}}$).

Figure 4.7 gives a detailed overview of the posture and pose errors for trajectory Tr_4 using database $\text{DB}_{\text{open+power}}$. This combination resulted in a low average error in all estimation parameters. The maxima that can be observed in the posture error can be explained by a quick hand posture change in the ground truth trajectory

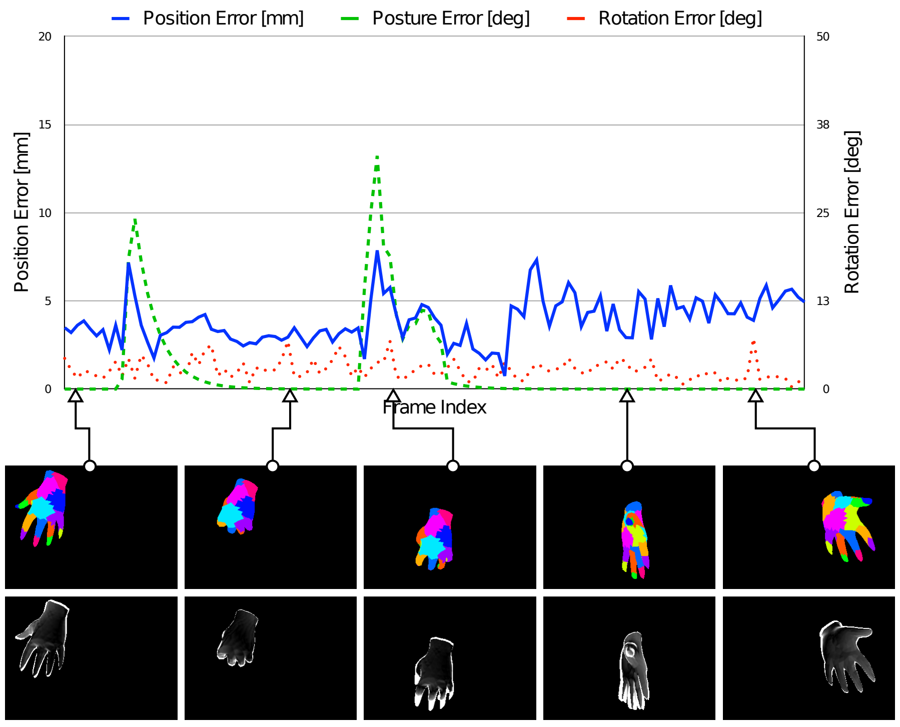


Figure 4.7: Detailed overview of trajectory Tr_4 using database $DB_{open+power}$. Several keyframes of the trajectory are visualized in the bottom of the figure. The upper row of images shows the virtual hand model according to the ground truth, the lower row of images shows the difference of the ground truth and estimation depth images.

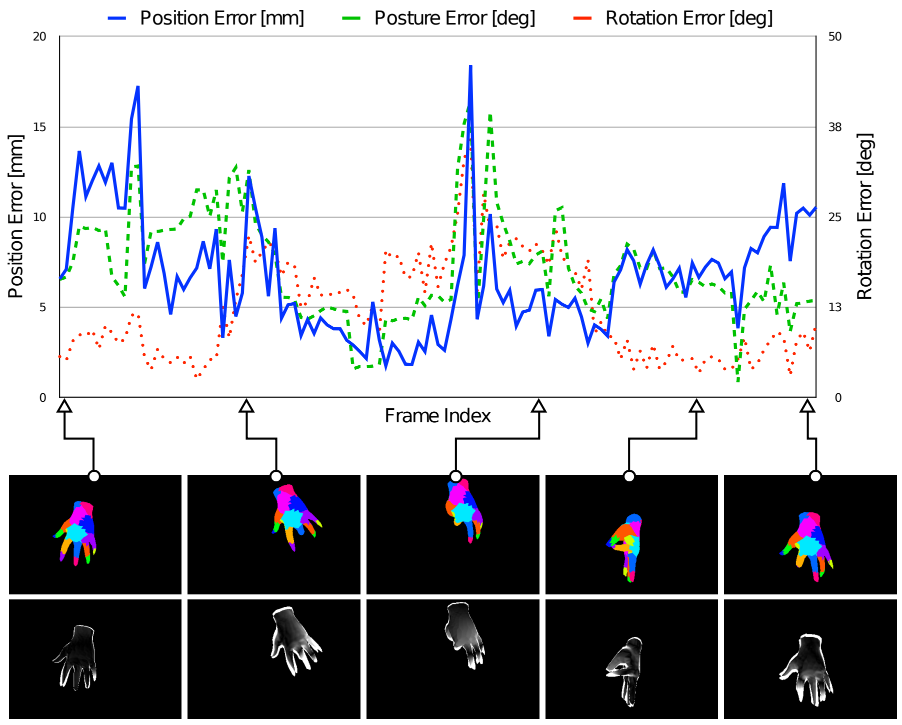


Figure 4.8: Detailed overview of trajectory Tr_1 using database $DB_{20\ postures}$. Analogous to Figure 4.7.

while opening and closing the hand. The posture estimation was lagging behind the ground truth for several frames before assuming the correct posture. Figure 4.8 shows the posture and pose errors for trajectory Tr_1 using database $\mathrm{DB}_{20\ \mathrm{postures}}$. In this experiment, larger errors occurred when database entries not related to the pinch grasp motion of the ground truth trajectory influenced the estimation result.

The average positional error of our hand pose estimation lies below 1 cm in all of our experiments. This shows a significant improvement over the results of Wang and Popović (2009), who reported translational errors of 5–15 cm along the optical axis of the tracking camera.

4.3.2 Database matching cascades

We evaluated the accuracy and efficiency of our system's posture estimation with regard to different matching cascades during the database lookup. Figure 4.9 illustrates the accuracy/efficiency trade-off associated with different tiny image sizes. While very small images have a low distance computation cost, the information loss due to the image downscaling significantly limits their accuracy. In the database matching cascades, the smallest images can be used for a coarse preselection of nearest neighbor candidates in the early stages that is efficiently re-ranked using the larger images in the later stages. We performed four different matching cascades to evaluate the behavior of the overall efficiency and accuracy of our system using two example databases. Figure 4.10 shows the performance of the different matching cascades. The slowest database lookup uses only 64×64 images and interpolates between the 3 nearest neighbors, the fastest and least accurate one uses only 8×8 images and interpolates between the 15 nearest neighbors. Performing multiple lookup stages improves the runtime efficiency while largely maintaining accuracy. Based on these findings, we used a two-stage matching cascade in our experiments, preselecting 500 nearest neighbor candidates based on 16×16 images in the first stage and interpolating the 3 nearest neighbors based on 64×64 images in the second stage.

4.3.3 Point cloud subsampling

To improve the runtime performance we uniformly subsample the sensor point cloud that is used during the ICP-T pose optimization. This introduces a trade-off between pose estimation accuracy and overall efficiency. A subsampled point cloud is less computationally expensive to match to the virtual hand surface, but it contains less positional information. This trade-off is visualized in Figure 4.11, which shows that the overall runtime of our system quickly converges to the database lookup time while the position estimation error slowly rises as less points are included in the sensor point cloud. Based on this, it is possible to select a subsampling factor that

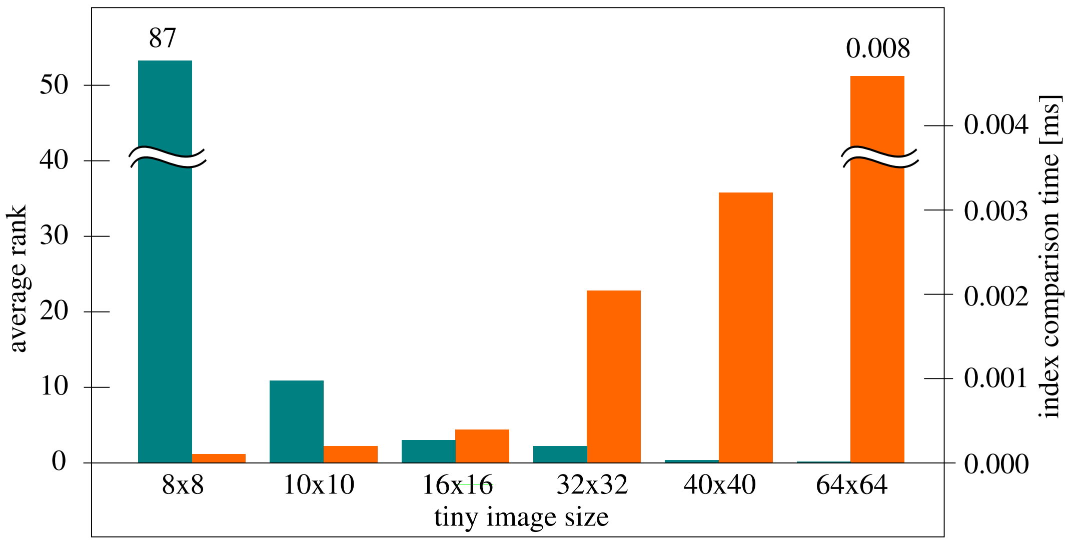


Figure 4.9: Runtime efficiency and database retrieval accuracy for different tiny image sizes. Efficiency is measured by the average computation time of the distance between two images, accuracy is measured by the average rank of the true nearest neighbor in the lookup results for a database of about 8000 images over 500 trial queries. A low average rank represents high accuracy. A trade-off between efficiency and accuracy with regard to image size can be observed.

DB_{20} postures

Matching Cascade	E_{posture} $[deg]$	Time $[ms]$
$16_{1000} \rightarrow 32_{300} \rightarrow 64_3$	16.61	121.6
$16_{500} \rightarrow 64_3$	16.49	91.2
64_3	16.69	178.3
8_{15}	29.68	65.1

$DB_{\text{open-pinch-power}}$

Matching Cascade	E_{posture} $[deg]$	Time $[ms]$
$16_{1000} \rightarrow 32_{300} \rightarrow 64_3$	4.88	86.2
$16_{500} \rightarrow 64_3$	4.88	73.5
64_3	4.88	78.0
8_{15}	24.92	58.8

Figure 4.10: Posture error and execution time for different matching cascades. The notation D_k indicates a k nearest neighbor lookup using $D \times D$ images. Arrows between two such lookups indicate the candidate preselection.

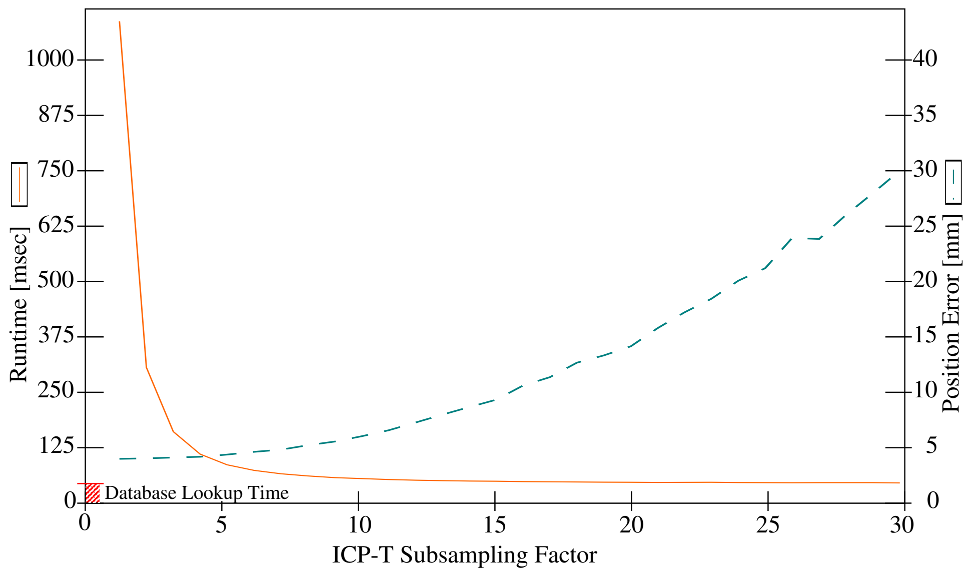


Figure 4.11: Overall runtime and position estimation error in regard to an increasing point cloud subsampling factor.

provides a good compromise to achieve interactive framerates while maintaining high accuracy.

4.4 Application

The main goal of our hand tracking system was to provide a low cost system that facilitates interactive robot control. To test this, we conducted several experiments in which we used the estimated posture and pose to actuate a compliant Shadow robot hand. This application was carried out in the Bielefeld "Curious Robot" setup (Lütkebohle et al. 2009), which has two redundant 7-DoF Mitsubishi PA-10 robot arms each equipped with a 20 DoF Shadow Dexterous Hand (*Shadow Robot Company*), resulting in a total of 54 DoFs. The Shadow Dexterous Hands are distinguished by their human-like design: in size, number and flexibility of joints, the hands resemble their human counterparts in a very realistic manner. The entire system (see Fig. 4.12) is controlled by numerous processes distributed over three PCs. The tests were carried out on a single 27-DoF PA-10/Shadow hand combination. Since the kinematic hand model used in our hand tracking system closely matches that of the Shadow robot hands (see Sec. 4.2), the estimated joint angles can be directly transferred to the robot by sending a command to the hand-server component. The global rotation and translation of the hand is mapped from the tracker's coordinate frame to the robot's coordinate frame and transferred to the robot by issuing a command to the arm-server component. The Kinect camera was positioned to have a top-down view of the user's hand, which provided a large working volume and minimal occlusion in our experiments.

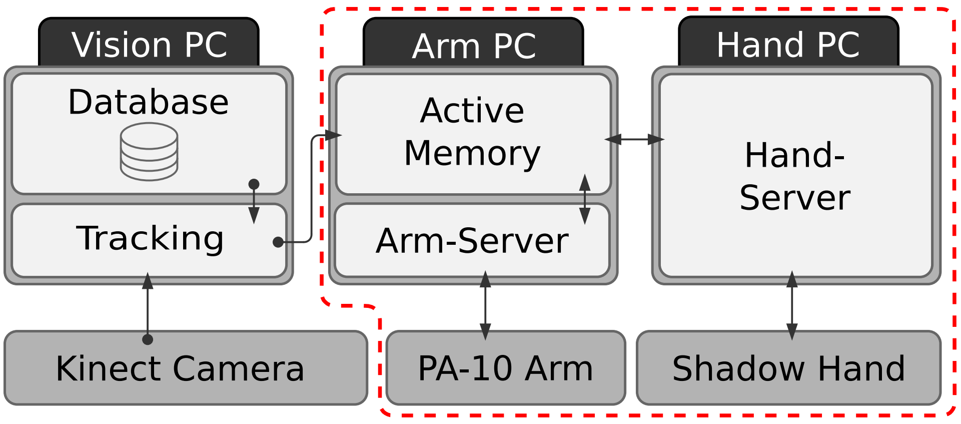


Figure 4.12: System component collaboration diagram. The whole system is distributed over 3 PCs for vision, hand control and arm control. IPC is implemented using a global Active-Memory node. The dashed line highlights the robot application setup.

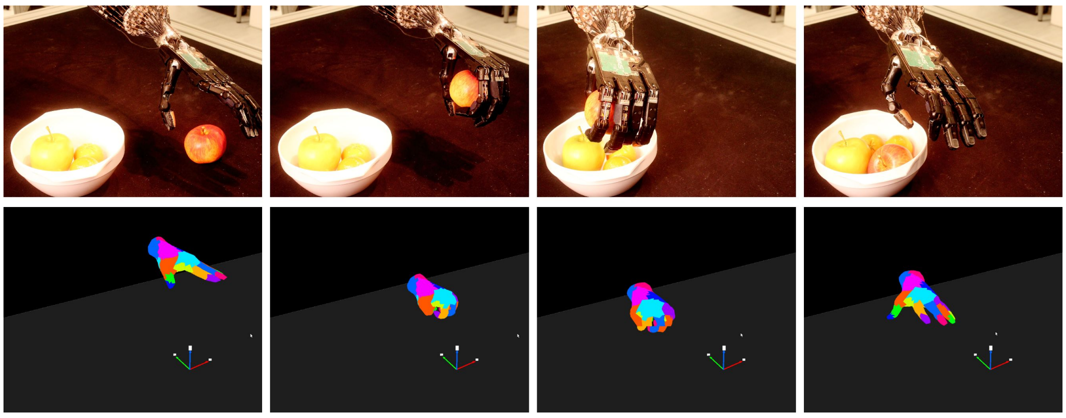


Figure 4.13: Pick-and-place experiment, in which the user controls the robot hand to grasp an apple and place it in a bowl. The image sequences show the performed power grasp motion for the actuated robot hand and our hand posture and pose estimation.

The hand movements we investigated during the experiments ranged from general hand movements to full interaction with objects. To illustrate the effectiveness of our system we constructed a database with three different postures: *open hand*, *power grasp* and *pinch grasp*. Using these three postures, the user controlled the robot in pick-and-place tasks (see Figure 4.13). By observing the movements of the robot hand while performing movements as input commands, the user can utilize this direct visual feedback to naturally adjust the pose and perform the grasping motion with relative ease. We found that for well-defined tasks, such as power or pinch grasping an object, the estimation results of our system are very good with a small database containing as little as two or three postures. The speed of our system (approximately 15Hz on a PC with four Intel Xeon E5530 2.4 GHz CPU cores and 4 GBs of RAM) is completely sufficient for interactive robot actuation and the accurate 3D position estimation provided by using the Kinect camera facilitates an intuitive transfer of the user's hand motions to the robot.

4.5 Discussion

Our approach to real-time hand tracking for the control of anthropomorphic robot hands is built on that of Wang and Popović (2009), but differs in that we place more emphasis on retrieving an accurate hand pose than on hand posture. To produce an accurate hand posture Wang and Popović (2009) implemented fast similarity sensitive coding (Torralba et al. 2008), allowing more postures to be present in the database while maintaining near real-time performance. They also added a 2D inverse kinematics step which adapts the estimated hand posture to that of the current camera image. These features mean that in terms of the estimated posture, our system is not as robust as that of Wang and Popović (2009). However, for the task of controlling a compliant robot hand in tasks such as grasping, an accurate pose is much more important than an exact approximation of the user's hand posture. In our approach we achieve a higher accuracy in pose estimation by using a Kinect camera and color-sensitive ICP-T. We argue that database approaches such as ours can be used to provide a good initial estimate of the posture and pose of the hand and then this can be used as a pre-initialization step for model-based tracking approaches that suffer from situations in which the hand is temporarily lost.

We have observed that under certain conditions, such when there are occlusions or the database is sparsely populated, it is possible that the output is an interpolation between two quite distinct postures in the database. While this is a feature of our system, allowing us to compute smooth transitions between discrete entries in the database, it can result in large discrepancies between the real posture and that given by the system. This is an artifact of the employed appearance-based method. An inverse kinematics step could improve on this, however, as the focus of this particular work was on the control of an anthropomorphic robot, accurate estimation of the pose of the hand took precedence over the posture. Using a combination of visual feedback, allowing us to adapt our hand during robot control according to what the situation required, and taking advantage of the compliance in the robot hand, we were able to smoothly actuate the robot hand to perform various grasping and interaction tasks.

Chapter 5

Constrained inverse kinematics

Artifacts and inaccuracies in the data obtained from commodity depth sensors, like the *Microsoft Kinect*, can be partly overcome by employing data-driven, appearance-based tracking methods, which incorporate knowledge about the tracked subject from ground truth databases. The downside to such appearance-based approaches is that they can only discriminate between known postures and their tracking capability is therefore limited by the employed database. In contrast, generative or model-based approaches optimize directly for the kinematic parameters of a model and thus have the capability to track its full articulation. However, particularly in hand tracking, optimizing the kinematic parameters is complex due to the high dimensionality of the space of hand articulations, which further complicates the task of hand tracking using low-quality commodity sensors. In this chapter, an approach is presented that combines the advantages of generative, model-based methods with those of data-driven approaches by employing hand posture subspace constraints in a generative, inverse kinematics-based real-time hand tracking system.

The method presented in this chapter is based on fitting an articulated hand model to the 3D point cloud obtained from a Kinect sensor's depth camera and using a subspace representation for hand articulations to simplify the problem (Schröder et al. 2013, 2014, Schröder and Botsch 2014). In this method, we estimate the hand articulation by finding the pose and posture parameters that minimize the error between the observed point cloud and the model surface using inverse kinematics (IK). In doing so, we find the articulation of the 3D hand model that best approximates the observed state of the user's hand. Regarding the issue of the high dimensionality of hand articulations, the analysis of hand synergies aims to identify high-level relationships in hand articulation in order to sensibly reduce the dimensionality of hand posture representations. We obtain such hand synergies through the principal component analysis (PCA) of motion capture data and use them directly in the tracking process to reduce the parameter space and to naturally constrain the hand posture estimation. The database of motion captured hand movements is publicly available[1].

[1] `http://graphics.uni-bielefeld.de/publications/icra14/`

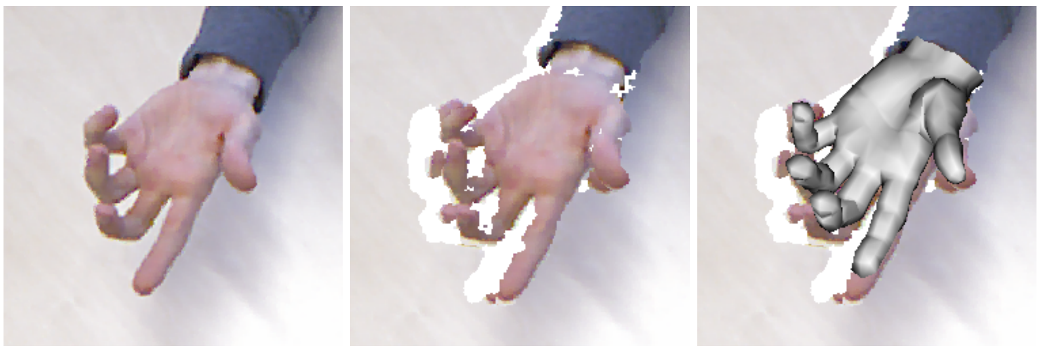


Figure 5.1: The user's hand posture is estimated in real-time by fitting a 3D hand model to the Kinect point cloud using inverse kinematics.

Solving the IK problem in a subspace reduces computational complexity and constrains the tracking to realistic hand postures even in cases of incomplete or ambiguous sensor data. However, this subspace tracking is a trade-off between robustness and flexibility of the hand posture estimation. Hand articulations that are not contained in the database cannot be reconstructed when tracking in a synergistic subspace. While a large amount of natural hand postures can be represented in such a subspace, some hand articulation details are not captured. To deal with this problem, we extend the subspace-constrained IK method in a way that adds flexibility to the posture estimation while maintaining the robustness of tracking in a synergistic subspace. To this end, we define an *adaptive* PCA model that is adjusted during real-time tracking to account for observed hand articulations that are not covered by the initial parameter subspace. This method extends the approach outlined in (Li et al. 2013), where such an adaptive model was used for real-time facial performance capture. Using an adaptive PCA model, we robustly combine the natural constraints provided by subspace optimization with the flexibility of optimizing in the full parameter space.

5.1 Related work

Model-based hand tracking approaches freely optimize for the kinematic parameters of a hand model and therefore provide a high degree of control over the reconstructed hand articulations. In addition to the hand itself, Hamer et al. (2009) modeled its interaction with objects in an offline tracking system. Similarly, Ballan et al. (2012) modeled objects and multiple hands in interaction. They used features such as edges, optical flow, and salient points extracted from videos in a multi-camera setup to estimate the articulated pose of hands interacting with objects within a single objective function. Wang et al. (2013) realized motion capture of hand grasping and manipulation data by simultaneously modeling hand articulation, object movement, and interactions between the two in an optimization framework. While producing highly accurate results, these model-based methods suffer from the high computational complexity associated with explicitly optimizing for the kinematic parameters of the hand and objects, and are therefore far from real-time.

On the other hand, advances in hardware have given rise to the proliferation of GPU-optimized model-based approaches that use stochastic optimization of the hand parameters for real-time hand tracking (Oikonomidis et al. 2011a, Qian et al. 2014, Sharp et al. 2015). These methods use variants of particle swarm optimization (Kennedy and Eberhart 1995, 2001), which optimizes arbitrary objective functions by iteratively evaluating and improving candidate solutions under a hypothesize-and-test paradigm. While achieving real-time results, these methods require a high amount of computational resources to function well, and they are inherently heuristical and probabilistic. In contrast, we present a gradient-based hand tracking method that optimizes the hand articulation directly in a constrained inverse kinematics point cloud fitting process.

To cope with the complexity of hand articulations, we reduce the high-dimensional hand posture space based on the concept of hand synergies. Bernstein (1967) was the first to come up with the idea of synergies and defined them to be high-level control schemes for kinematic parameters. He suggested that they could provide a mechanism by which the central nervous system controls the high DoF human hand in an efficient way. It was not, however, until Santello et al. (1998) published their paper on hand synergies that the rehabilitative prostheses and robots research communities realized their potential in terms of controlling articulated hands and arms. Their work revealed that 90% of the variance in the data of grasps directed towards household objects could be described by as little as 3 principal components (PCs). Many other studies have since supported this view (Daffertshofer et al. 2004, Tresch et al. 2006).

Using synergies or other methods to reduce the dimensionality of the problem of hand pose estimation for hand tracking applications has a precedence. In an early paper, Lee and Kunii (1995) placed a set of constraints on joint angle limits and movement types to reduce the DoFs or reject infeasible inverse kinematics solutions. Wu et al. (2001) used the fact that hand articulations could be represented in a lower dimensional space to perform a Monte Carlo tracking algorithm. Unlike Lee and Kunii, they were able to track the hand in real-time, but their method was view dependent and rotation and scaling were not considered. Another view dependent hand tracking particle filter approach (Kato et al. 2006) also reduced the dimensionality of the problem by using independent component analysis to compute basis components for all fingers. Bray et al. (2007) used smart particle filtering to efficiently explore the high-dimensional search space with fewer samples.

We use dimension reduction in order to simplify and constrain the problem of inverse kinematics. A similar concept was previously employed in the work of Grochow et al. (2004), who presented an inverse kinematics system based on a learned model of human poses. Safonova et al. (2004) also used dimension reduction to synthesize realistic human motion. Using a synergistic approach to reduce the DoFs of a virtual hand model, we also reduce the high computational complexity associated with model based approaches, while at the same time realizing realistic (view unrestricted) real-time bare hand tracking.

In addition to integrating a synergistic PCA subspace in the IK optimization, we define an *adaptive* PCA model that can adjust the synergistic subspace to previously unknown hand articulations in real-time. This concept was used in a similar fashion in (Li et al. 2013) for real-time facial performance capture in order to accurately adapt a blend-shape face model to user-specific facial expressions. Using an adaptive PCA model we optimize the hand articulation in a low-dimensional space that both constrains the estimation to realistic postures while still allowing for high flexibility and accuracy. We develop an incremental update of the adaptive PCA subspace based on direct rank-one updates, which allows for a highly efficient adaptation process.

Our adaptive PCA model extends our subspace-constrained method in a simple and efficient way by complementing the initial subspace model with a continuously updated local linear model. While there are various other methods for linear or non-linear local embeddings (Urtasun and Darrell 2008, Roweis et al. 2002), usually their performance depends on the quality of model parameters or they do not meet real-time requirements. Data-driven local linear models were previously used for full-body motion capture in (Chai and Hodgins 2005, Liu et al. 2006). Our method differs from these approaches in that we aim to specifically model articulations that are not present in the ground truth database to facilitate robust and flexible real-time hand tracking.

5.2 Inverse kinematics hand tracking

In our hand tracking approach, the pose and posture of the user's hand are estimated by fitting the virtual hand model to the point cloud obtained from a *Microsoft Kinect* sensor. By finding the articulation of the hand model that minimizes the distance to the point cloud, the state of the user's hand that causes the observation is approximated. The hand model consists of 16 joints, which are driven by 26 kinematic parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{26})^T$. Of those parameters, 6 describe the global pose of the hand and the remaining 20 describe the posture.

We experimented with two distinct geometric representations of the hand model, which are depicted in Figure 5.2. One representation is based on a skinned triangle mesh and the other is based on piecewise rigid capsule segments for each joint. The main advantage of the simplified capsule-based model is that correspondence computations between the sensor point cloud and the hand model are very efficient. Both models are animated according to the articulation of the joints defined in the kinematic hand model. While a piecewise rigid forward kinematics scheme is sufficient to animate the capsule-based model, the deformation of the mesh-based

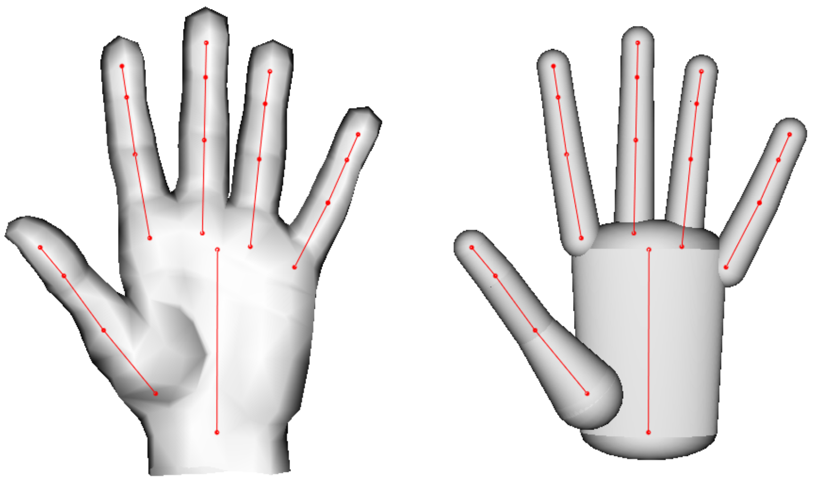


Figure 5.2: Hand models used for tracking. Left: skinned triangle mesh. Right: cylindroid capsule segment model. Both models use the same kinematic control skeleton and only differ in their hand geometry representations.

model is done using linear blend skinning (Jacka et al. 2007). In the latter case, we therefore use a different animation technique for *visualization* than for *tracking*.

The point cloud is calculated from the Kinect's color and depth images based on a precomputed RGBD-mapping, which maps color values to the pixel coordinates of the depth image and uses the camera parameters of the Kinect's color and depth cameras to calculate the global 3D positions of the sensor points. The hand is then segmented by detecting skin-colored pixels and omitting points whose coordinates are outside of a predefined working volume. The remaining points define the target constraints for the hand model fitting.

These target points are matched to their spatially closest points on the surface of the hand model. Based on these point-to-point correspondences we formulate the problem of estimating the posture of the hand as finding the posture parameters (joint angles and global pose) that transform the hand's skeleton such that the error between the deformed model and target positions is minimized. This is an inverse kinematics (IK) problem in which the points on the model surface $\mathbf{x}_i$ are regarded as effector positions relative to the skeleton, which are constrained to move towards their corresponding target positions $\mathbf{t}_i$ in the sensor point cloud. Figure 5.3 illustrates the principle with a simplified example.

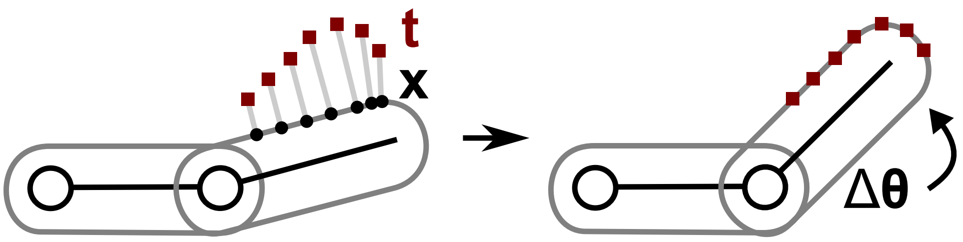


Figure 5.3: Inverse kinematics posture estimation. The red squares are target positions in the sensor point cloud, the black circles are the corresponding effector positions on the surface of the model. The joint angle is updated such that the target-effector error is minimized.

5.2.1 Inverse kinematics

The IK optimization method used in this work is based on the same principle as the one presented in Chapter 3. Here we briefly reiterate the most important components of the optimization.

The positions of the k effector points on the surface of the hand model are represented as a stacked vector $\mathbf{x} \in \mathbb{R}^{3k}$ and move relative to the model articulation, and can therefore be expressed as a function of the kinematic parameters, $\mathbf{x} = \mathbf{x}(\boldsymbol{\theta})$. These effector positions are subject to move to their corresponding target positions $\mathbf{t} \in \mathbb{R}^{3k}$ in the point cloud. The IK problem $\mathbf{t} = \mathbf{x}(\boldsymbol{\theta})$ is solved by finding an update to the kinematic parameter vector $\boldsymbol{\theta}$ that minimizes the objective function

$$E_{\text{IK}}(\Delta\boldsymbol{\theta}) = \frac{1}{2}\left\|\mathbf{x}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) - \mathbf{t}\right\|^2 + \frac{1}{2}\left\|\mathbf{D}\Delta\boldsymbol{\theta}\right\|^2. \tag{5.1}$$

In this objective function, the first term models the least squares error between the positions of the effector points $\mathbf{x}$ and the positions of their corresponding target points $\mathbf{t}$. The second term is a selective damping term for the parameter update $\Delta\boldsymbol{\theta}$ with a diagonal matrix $\mathbf{D}$. This damping stabilizes the solution and is used for joint limit avoidance by selectively increasing the damping of kinematic parameters that approach their upper or lower limits (Schröder et al. 2014).

To find the parameter update $\Delta\boldsymbol{\theta}$, the objective function (5.1) is minimized with a Gauss-Newton approach, in which a linear system is solved in multiple iterations. The objective function leads to the linear system

$$\left(\mathbf{J}^T\mathbf{J} + \mathbf{D}\right)\,\Delta\boldsymbol{\theta} = \mathbf{J}^T\left(\mathbf{t} - \mathbf{x}(\boldsymbol{\theta})\right), \tag{5.2}$$

where $\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}}$ is the $(3k \times 26)$ Jacobian matrix of the effector positions (Buss 2004). After solving the linear system, the resulting update $\Delta\boldsymbol{\theta}$ is scaled using a line search in order to guarantee convergence. The process of solving the linear system (5.2) and updating the effector positions is iterated 5–10 times.

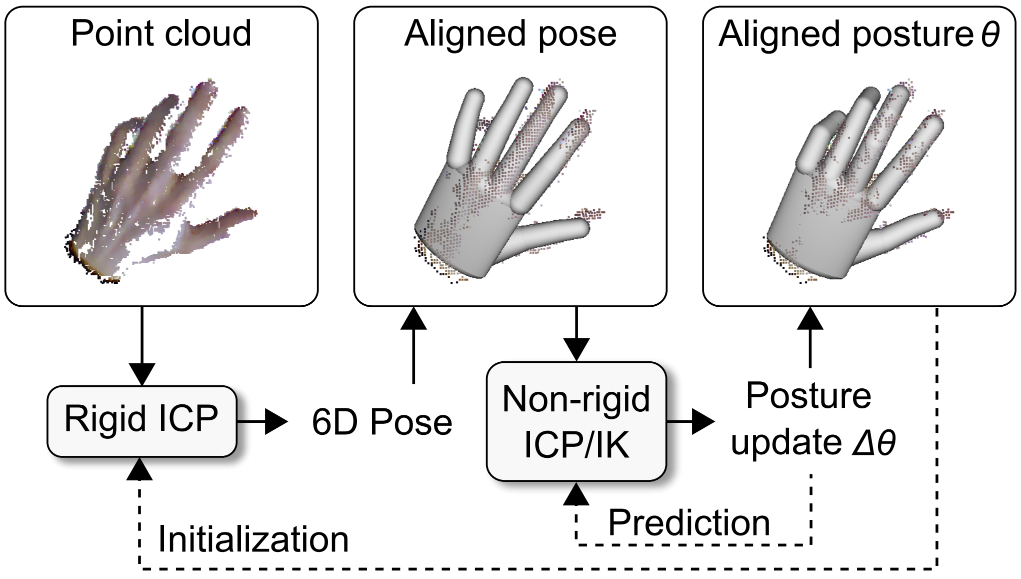


Figure 5.4: Schematic overview of the overall hand tracking process.

5.2.2 Hand tracking process

The overall hand tracking process is illustrated in Figure 5.4. After segmenting the hand in the Kinect point cloud and finding the target-effector correspondences, the pose estimation is initialized by finding the rigid transformation between the target and effector point clouds using the well known rigid iterative closest points (ICP) approach of Horn (1987), using the posture parameters from the previous frame. In the first frame, this ICP fitting provides a good initialization if the posture and orientation of the user's hand roughly matches the initial neutral state of the hand model.

Then the correspondences are updated and used as input for the iterative IK-based pose and posture estimation. During this process, the parameter update is computed according to (5.2) and the effector points are moved according to the updated skeleton forward kinematics. This process is iterated until the target-effector error converges. In practice, our IK optimization usually takes less than 10 Gauss-Newton iterations and for real-time tracking 5 iterations are sufficient.

The process of recomputing the correspondences and solving the IK optimization is iterated several times in a non-rigid ICP manner. As a result the virtual hand model is aligned with the observation point cloud, which yields the hand pose and posture estimation.

5.3 Principal component analysis

In the approach outlined in the previous section all 26 parameters of the kinematic model are freely optimized within their joint limits during the IK update. While this allows for high freedom of movement and mostly yields plausible hand articulations, it can also result in inaccurate or unnatural hand posture reconstructions in cases of incomplete or ambiguous sensor data.

This problem can be overcome by employing subspace representations of realistic hand articulations. As shown in Chapter 3, the space of possible hand postures can be reduced in a meaningful way using hand synergies derived from the principal component analysis (PCA) of a training data set containing real human hand motions. Performing dimension reduction based on the most significant principal components provides a way to decrease the number of parameters that need to be optimized and to implicitly constrain the estimated hand postures to plausible ones resulting from the training data, preventing unnatural hand articulations.

5.3.1 Principal component space

In the following, the most important components of the subspace IK approach outlined in Chapter 3 are briefly reiterated. Performing PCA on a database of 20-dimensional hand posture data yields a set of eigenvectors and eigenvalues, which can be used to construct a $26 \times (6 + l)$ matrix $\mathbf{M}$, which maps between the full 20-dimensional posture space and a reduced l-dimensional subspace and has the form

$$\mathbf{M} = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{P} \end{pmatrix}, \tag{5.3}$$

where $\mathbf{I}$ is a 6×6 identity matrix and $\mathbf{P}$ is the $20 \times l$ matrix of principal components. The first 6 dimensions encode the global pose of the hand, which is not captured in the PCA model.

Given the PCA matrix $\mathbf{M}$, the full parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{26}$ can then be computed from the reduced subspace parameters $\boldsymbol{\alpha} \in \mathbb{R}^{6+l}$ as $\boldsymbol{\theta} = \mathbf{M}\boldsymbol{\alpha} + \boldsymbol{\mu}$, where $\boldsymbol{\mu} \in \mathbb{R}^{26}$ is the mean of the database postures. This makes it possible to represent the forward kinematics of the effector points $\mathbf{x}$ subject to the subspace parameters: $\mathbf{x} = \mathbf{x}(\boldsymbol{\alpha}) = \mathbf{x}(\boldsymbol{\theta}(\boldsymbol{\alpha}))$. Based on this representation, the IK problem can be expressed in terms of the subspace parameters as well. Optimizing for the subspace parameters in (5.1) and (5.2) is possible using the subspace Jacobian

$$\mathbf{J}_{\text{PC}} := \frac{\partial \mathbf{x}}{\partial \boldsymbol{\alpha}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\alpha}} = \mathbf{J} \cdot \mathbf{M}. \tag{5.4}$$

Substituting $\mathbf{J}_{\text{PC}}$ for $\mathbf{J}$ in the linear system (5.2) and analogously changing the damping matrix $\mathbf{D}$ yields the IK solution for the subspace parameters.

This facilitates hand tracking as described in Section 5.2 in the reduced principal component-space (PC-space), which decreases the size of the matrices in the update calculation and reasonably constrains the possible hand postures estimated by the tracking system. The number l of PCs used for the dimension reduction depends on the distribution of the data variance among the principal directions (see Section 5.5).

5.3.2 Hierarchical optimization

While optimizing in a reduced PC-space yields plausible hand articulations, the estimated hand postures are limited to those represented by linear combinations of the PCs of the posture database. Although this improves robustness and performance, it inherently restricts tracking flexibility to a subset of the movements contained in the database. To combine the benefits of reduced-DoF estimation with the flexibility of full-DoF estimation, we propose a hierarchical optimization scheme, in which a low-dimensional posture estimate is successively refined by incrementally adding DoFs during the estimation process.

This means that the IK optimization in the hand tracking process is performed in multiple stages, increasing the number of PC-parameters used in each stage. The early low-DoF estimation stages yield coarse initializations for the subsequent higher-DoF stages. In the final stage of this hierarchical scheme, all 26 DoFs are optimized. This allows for a coarse-to-fine optimization process that robustly refines the posture estimation based on hand synergies and yields highly accurate posture reconstructions. The hierarchical approach is compared to the non-hierarchical IK in Section 5.5 and an example for the results is shown in Figure 5.10. While accurate, this method is not suitable for real-time tracking, firstly due to high computational cost and secondly because the method does not take advantage of temporal coherence, as the local refinements outside of the low-dimensional synergistic subspace are lost across frames, requiring a high number of ICP iterations. In the following, we describe an adaptive PCA model that allows for such refinements to be performed without negative impact on the real-time tracking performance.

5.4 Online adaptive PCA model

To overcome the limitations related to optimizing the hand posture in a reduced parameter space, we define an adaptive PCA model. This model can be automatically modified to account for newly observed postures which cannot be represented within the initial PCA subspace. To this end, the PC-space conversion matrix $\mathbf{M}$ defined in

(5.3) is extended by d columns corresponding to adaptive PC basis vectors, resulting in the $26 \times (6 + l + d)$ subspace matrix

$$\mathbf{M}_\mathrm{A} = \left(\mathbf{M} \;\middle|\; \begin{matrix} 0 \\ \mathbf{C} \end{matrix} \right) = \left(\begin{matrix} \mathbf{I} & 0 \\ 0 & \mathbf{P} \end{matrix} \;\middle|\; \begin{matrix} 0 \\ \mathbf{C} \end{matrix} \right), \tag{5.5}$$

where $\mathbf{I}$ is the 6×6 identity matrix, $\mathbf{P}$ is the $20 \times l$ matrix containing the original principal components and $\mathbf{C}$ is a $20 \times d$ matrix containing the “adaptive columns”, which by construction will lie in the null space of $\mathbf{P}$.

Following the terminology of (Li et al. 2013), we refer to $\mathbf{P}$ as the *anchor* matrix and $\mathbf{C}$ as the *corrective* matrix. The anchor matrix remains fixed and prevents gradual drift of the PCA model, whereas the corrective matrix is adaptive and represents the observed hand articulations that cannot be represented in the *anchor space* spanned by $\mathbf{M}$. The number of corrective dimensions d depends on the desired flexibility of the adaptive model (see Sections 5.4.2 and 5.4.3).

The inverse kinematics posture estimation can be performed in the extended adaptive PC-space by using the Jacobian matrix $\mathbf{J}_\mathrm{A} = \mathbf{J} \cdot \mathbf{M}_\mathrm{A}$ and an analogously extended damping matrix in the Gauss-Newton process (5.2). The online adaptation of the corrective matrix $\mathbf{C}$ takes place after the non-rigid ICP process, during which the hand posture is estimated using the current subspace matrix $\mathbf{M}_\mathrm{A}$ in the inverse kinematics optimization. Figure 5.5 gives a schematic overview of the adaptation process.

At the beginning of the adaptation procedure, the posture estimate from the current subspace $\mathbf{M}_\mathrm{A}$ is refined by an additional IK optimization in the *full* 26-dimensional parameter space. This aligns the model more closely with the observed point cloud and thereby captures details of the user’s hand articulation that cannot yet be represented in the adaptive PC-space. Since the full-DoF IK optimization starts from a good initial guess (the subspace IK result), it robustly improves the posture low-DoF estimate. The result of this refinement is an updated parameter vector $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{26}$.

Based on this refined posture we compute the anchor space residual $\hat{\mathbf{s}}$ as the orthogonal projection of the refined posture $\hat{\boldsymbol{\theta}}$ onto the complement of the anchor space:

$$\hat{\mathbf{s}} = \left(\mathbf{I} - \mathbf{M}\mathbf{M}^T\right) \left(\hat{\boldsymbol{\theta}} - \boldsymbol{\mu}\right). \tag{5.6}$$

As the leading six pose DoF entries of $\hat{\mathbf{s}} = (s_1, \ldots, s_{26})^T$ are zero by construction, we only consider the vector of joint angle residuals $\mathbf{s} = (s_7, \ldots, s_{26})^T$ in the following. Intuitively, the residual vector $\mathbf{s}$ represents those aspects of the refined posture that lie outside of the initial PC subspace $\mathbf{P}$. A new residual sample $\mathbf{s}$ is considered valid if $\|\mathbf{s}\|$ (the distance of the refined posture from the anchor space) is above a threshold s_min (significant improvement) and below a threshold s_max (no outlier). In this case, it is stored in a FIFO ring buffer matrix $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_N)$. We use validity bounds $s_{min} = 0.1$ and $s_{max} = 3$ and a buffer size of $N = 250$.

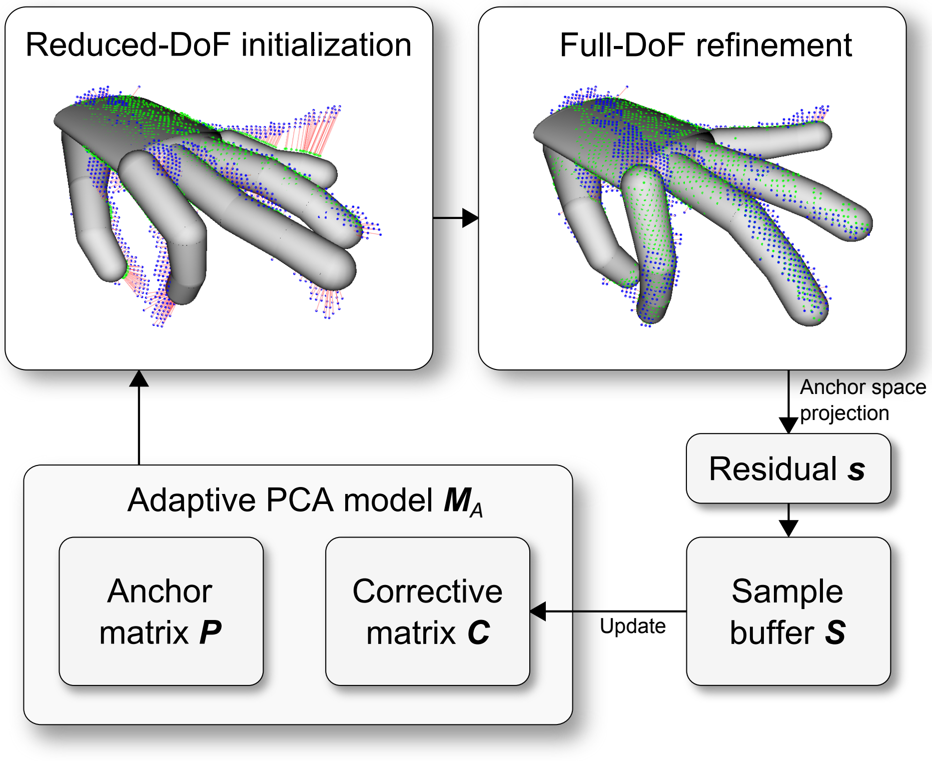


Figure 5.5: Online adaptation of the adaptive PCA model.

Once the buffer is full, i.e., N frames contributed significant residuals, the corrective matrix $\mathbf{C}$ can be computed by performing PCA on the sample matrix $\mathbf{S}$. Standard methods for PCA compute an eigenvector decomposition of the data covariance, e.g. by singular value decomposition (SVD) of the data matrix, or eigenvalue decomposition (EVD) of the covariance matrix itself. While the former is numerically more accurate, we found that the latter generally has better runtime performance for the rather tall $N \times 20$ matrices in our context. However, neither method scales well with the size of the data matrix (see Table 5.1).

In the following, we focus on PCA computed by EVD of the sample covariance. The covariance matrix $\mathbf{K}$ of the sample matrix $\mathbf{S}$ is defined as

$$\mathbf{K} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T, \tag{5.7}$$

where $\bar{\mathbf{s}} = \sum_{i=1}^{N} \mathbf{s}_i / N$ is the mean of the sample points. This sum of outer products involves many numerical calculations and can impact performance significantly for large N. Since we need to update the corrective matrix for every new incoming sample $\mathbf{s}_i$, this way of performing PCA can cause a performance bottleneck. In the following, we explore alternative methods for performing PCA in order to compute the corrective matrix $\mathbf{C}$ efficiently.

5.4.1 Corrective matrix computation

In (Li et al. 2013) the iterative expectation maximization (EM) algorithm for PCA presented in (Roweis 1998) was used to compute the corrective matrix more efficiently. This algorithm progressively updates an approximation of a dataset's PC subspace given only a limited number of sample points at a time. A single EM iteration for updating the corrective matrix involves the following calculations:

1. E-step: $\mathbf{Y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{S}$
2. M-step: $\mathbf{C} = \mathbf{S}\mathbf{Y}^T(\mathbf{Y}\mathbf{Y}^T)^{-1}$

These steps are iterated several times before orthonormalizing the result $\mathbf{C}$. While this method can outperform standard PCA methods in some cases, it still involves many numerical operations and calculations, including a matrix orthonormalization, and needs to be iterated 3–4 times to converge (Roweis 1998), which altogether diminishes the runtime benefits in our application context, making it perform slightly worse than EVD PCA for large N (see Table 5.1).

It can be observed that for larger N the cost of the EVD PCA method is dominated *not* by the 20×20 eigenvector decomposition of $\mathbf{K}$, but instead by the computation of the matrix $\mathbf{K}$ itself as in (5.7), which scales linearly with N.

In contrast, we exploit the incremental nature of the adaptation process by revising the computation of the covariance matrix $\mathbf{K}$ in such a way that allows us to efficiently update the covariance in an incremental way, given a single new corrective sample at a time. The method we propose below results in *constant* costs for computing $\mathbf{K}$ and its eigenvector decomposition—independent of the buffer size N.

We achieve this goal by rewriting the definition of the covariance matrix $\mathbf{K}$ in a way that allows for an incremental adaptation based on rank-one updates. Expanding the outer products in (5.7) yields

$$\begin{aligned}
\mathbf{K} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T \\
&= \frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{s}_i\mathbf{s}_i^T - \mathbf{s}_i\bar{\mathbf{s}}^T - \bar{\mathbf{s}}\mathbf{s}_i^T + \bar{\mathbf{s}}\,\bar{\mathbf{s}}^T\right] \\
&= \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - \left(\sum_{i=1}^{N}\mathbf{s}_i\right)\bar{\mathbf{s}}^T - \bar{\mathbf{s}}\sum_{i=1}^{N}\mathbf{s}_i^T + N\,\bar{\mathbf{s}}\,\bar{\mathbf{s}}^T\right] \\
&= \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - N\,\bar{\mathbf{s}}\,\bar{\mathbf{s}}^T - N\,\bar{\mathbf{s}}\,\bar{\mathbf{s}}^T + N\,\bar{\mathbf{s}}\,\bar{\mathbf{s}}^T\right] \\
&= \frac{1}{N}\left[\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T\right] - \bar{\mathbf{s}}\,\bar{\mathbf{s}}^T. \qquad (5.8)
\end{aligned}$$

PCA method	$N = 100$	$N = 500$	$N = 1000$
Jacobi SVD	$125\mu s$	$272\mu s$	$477\mu s$
EVD	$60\mu s$	$119\mu s$	$198\mu s$
Exp. Max.	$54\mu s$	$137\mu s$	$254\mu s$
Inc. cov. (ours)	$35\mu s$	$35\mu s$	$35\mu s$

Table 5.1: Runtimes for the corrective matrix update using different PCA methods for varying sample buffer sizes. PCA using SVD, EVD and EM scale poorly with increasing buffer sizes, whereas our method runs in constant time.

Based on this expression, the mean and covariance of the sample points are decoupled, allowing us to directly and separately update them given a single new sample point at a time. Once the sample buffer $\mathbf{S}$ is full, the mean $\bar{\mathbf{s}}$ and the covariance $\mathbf{K}$ are explicitly initialized by computing (5.7). After this, each subsequent incoming sample $\mathbf{s}_{in}$ replaces an old sample $\mathbf{s}_{out}$ in the FIFO ring buffer matrix $\mathbf{S}$. The two samples are then used to directly compute the updated mean $\bar{\mathbf{s}}'$ and covariance $\mathbf{K}'$ in an incremental way:

$$\bar{\mathbf{s}}' = \bar{\mathbf{s}} + \frac{\mathbf{s}_{in}}{N} - \frac{\mathbf{s}_{out}}{N} \tag{5.9}$$

$$\mathbf{K}' = \mathbf{K} + \frac{\mathbf{s}_{in}\,\mathbf{s}_{in}^T}{N} - \frac{\mathbf{s}_{out}\,\mathbf{s}_{out}^T}{N} + \bar{\mathbf{s}}\,\bar{\mathbf{s}}^T - \bar{\mathbf{s}}'\bar{\mathbf{s}}'^T. \tag{5.10}$$

Update (5.9) shifts the mean according to the incoming and outgoing samples and update (5.10) is a series of rank-one updates to the covariance matrix derived from (5.8), which can be computed efficiently in a single loop. Finally, the new corrective matrix $\mathbf{C}$ is obtained by performing eigenvalue decomposition of the updated covariance matrix $\mathbf{K}'$.

This computation of the corrective matrix is independent from the size N of the buffer matrix $\mathbf{S}$ and therefore allows for an efficient update of the adaptive model in *constant time* given a single new sample. The computational cost of the update is dominated by the eigenvalue decomposition. Table 5.1 lists average runtimes for PCA using SVD, EVD, expectation maximization, and our incremental covariance method and shows the latter to outperform the previous methods.

We note that it is also possible to directly update the EVD of $\mathbf{K}$ or the SVD of $\mathbf{S}$ after rank-one modifications (Bunch et al. 1978, Brand 2006), but our approach is more straightforward to implement and provides a significant improvement over non-incremental methods with only minor algorithmic modifications.

5.4.2 Tracking in adaptive space

The adaptive PCA model allows us to perform local posture refinements after fitting in a reduced PC-space without losing temporal coherence. Since increasing the buffer size does not negatively impact runtime performance, we can choose a large buffer size containing a long history of samples. However, while using a long history captures more details missing from the anchor space, it can cause the adaptive PCA model to drift from the initial synergistic model, which can compromise the plausibility of the reconstructed postures. As our tracking system runs at approximately 25 fps, a sample buffer size of $N = 250$ captures a history of new postures for approximately ten seconds, which can fully account for the local refinements and additionally captures hand articulation details that are not present in the initial synergistic subspace in a robust way.

Our analysis of hand postures in (Schröder et al. 2013, 2014) and Section 5.5 shows that 90% of the variance of a dataset of highly varying hand postures can be represented by six PCs and 90% of the variance of specific hand movements, such as grasping, can be covered by as little as three PCs. Based on these findings we use $l = 3$ dimensions for the initial PC subspace and $d = 3$ corrective dimensions for continuous tracking with the adaptive PCA model. Using more corrective dimensions increases flexibility but comes at the price of losing robustness (see Section 5.5). Less than three corrective dimensions cause the estimation to be driven mostly by the initial PCs.

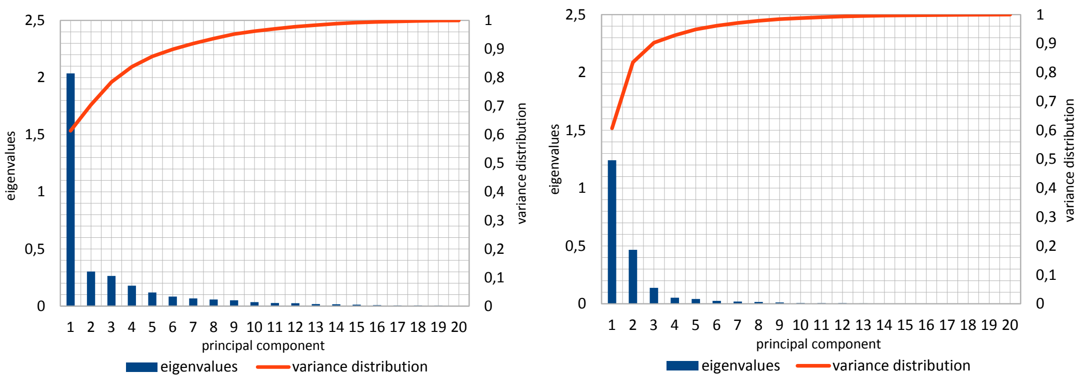
5.4.3 Learning a synergistic model

Beyond capturing local posture refinements, the adaptive PCA model can be used to generate a synergistic model from scratch, as an alternative to relying on a prerecorded database of human hand postures. To this end, the user demonstrates individual hand movements in a training phase, during which the adaptive model learns the corrective DoFs that represent these movements. Then, the anchor space is incrementally expanded to include the trained corrective dimensions.

In the beginning of the learning process, the anchor matrix $\mathbf{P}$ should be initialized with a single manually defined hand posture. After training the adaptive model by demonstrating a certain new hand movement, the anchor space can be expanded by joining the anchor matrix $\mathbf{P}$ with the corrective matrix $\mathbf{C}$ and re-initializing the adaptive model with this new anchor space. As only isolated movements are demonstrated during the training phase, it is sufficient to use $d = 1$ new corrective dimension at a time. Alternatively, more corrective dimensions can be used during training to capture more involved movements and learn multiple synergistic DoFs simultaneously.

5.5 Results

In this section we first illustrate the PCA of the captured hand posture data and give examples for the DoFs that can be represented in a reduced PC-space. Then, we present a detailed evaluation of the most important aspects of our system and



(a) Full posture data **(b)** Grasp posture data

Figure 5.6: Eigenvalues and variance distribution among the principal components of motion captured hand posture data sets.

compare the accuracy achieved by the adaptive PCA model to that of a non-adaptive method based on synthetic input data. Finally, we provide experimental results of our real-time tracking system using a Kinect camera.

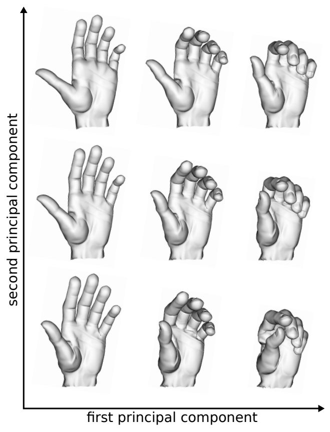
5.5.1 Hand posture PCA results

We discuss the results of the PCA for two ground-truth posture data sets: the complete data set of all captured movements (including various manual interaction motions, sign language and general hand movements) and a data set containing only grasping movements.

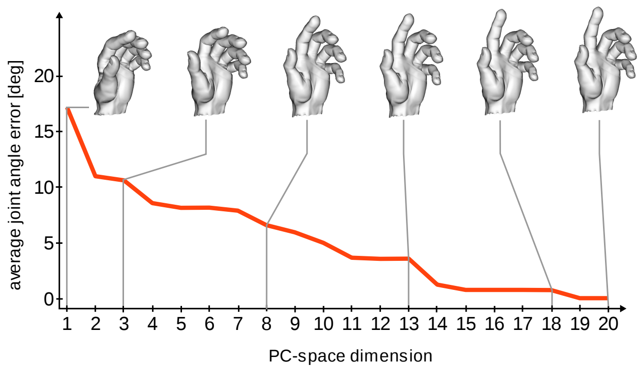
Figure 5.6(a) shows the distribution of the data variance among the principal components for the complete data set. Approximately 80% of the variance is covered by the principal components associated with the largest 3 eigenvalues, and approximately 90% of the variance is covered by 6 principal components. The 2 most significant principal components cover about 70% of the data variance and can already be used to represent meaningful hand synergies, which is illustrated in Figure 5.7(a). For the grasping data set the 3 most significant principal components already cover 90% of the variance, which can be seen in Figure 5.6(b). The less varied a data set is in terms of movements contained, the less principal components are needed to cover most of its variance. To clarify the information loss resulting from dimension reduction, Figure 5.7(b) shows the approximation error for a specific hand posture w.r.t. the number of dimensions used to represent it.

5.5.2 Evaluation with synthetic data

In order to evaluate the accuracy and overall performance of our hand tracking system we generated sequences of synthetic Kinect images using the virtual hand model.



(a) DoFs represented by two PCs

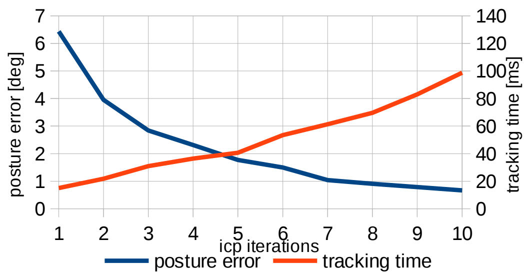


(b) PCA reconstruction error

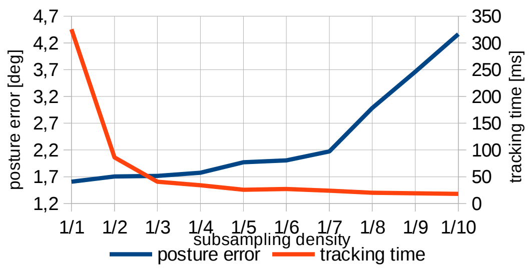
Figure 5.7: Illustration of the information represented by PCA subspaces. (a) Visualization of the DoFs represented by the first and second most significant PCs of the captured hand posture data set containing grasp movements. (b) Posture reconstruction and approximation error for one particular posture with increasing number of PCs. The hand postures on top show examples of the reconstruction of the original hand posture using different numbers of PCs.

We performed the experiments with both the mesh model and the simplified capsule model (see Figure 5.2). We explicitly state which model was used for the respective presented results when it is significant. The model was animated using pre-recorded hand trajectories and synthetic depth images were rendered, which could then be used as input for our tracking system. This made it possible to compare the postures generated by our system with the known ground truth data.

The synthetic input data was generated with the same model that was also used for tracking; no noise was added to the depth images. This results in an idealized experimental setup, in which the potential for highly accurate posture reconstructions was given. We analyze the issues of noisy input data and mismatch between the tracked real hand and the virtual hand model in Section 5.5.5. Among the experiments we conducted are tests for evaluating the trade-off between the accuracy of the hand posture reconstruction and the runtime efficiency of the tracking, analyzing the benefits of using PC-space tracking, and the performance of single-frame posture estimation using hierarchical optimization. All experiments were conducted on an Octa-Core Intel Xeon(R) E5-1620 CPU at 3.60GHz with 8 GBs of RAM. Our implementation is heavily parallelized and fully utilizes all eight cores during the correspondence search and the construction of the Jacobian matrix.



(a)



(b)

Figure 5.8: Posture error and tracking time using (a) subsampling density of 1/3 and varying numbers of non-rigid ICP iterations, (b) five non-rigid ICP iterations and varying subsampling densities.

Accuracy-efficiency trade-off

The reconstruction accuracy increases with (i) the number k of point cloud-to-model correspondences used as input for the estimation and (ii) the number of non-rigid ICP iterations. However, in turn the runtime efficiency deteriorates with increased correspondences and more non-rigid ICP iterations.

In order to be able to track at interactive rates, we reduce the number of correspondences by linearly subsampling the input point cloud. The point cloud sampling density and the number of correspondence iterations are therefore the parameters that need to be adjusted to optimize the accuracy-efficiency trade-off. We tracked a synthetic input sequence with varying values for these parameters in order to find the best values for them. Evaluating the results for all combinations revealed that for our purposes a subsampling density of $1/3$ and five non-rigid ICP iterations provided the best trade-off between accuracy and efficiency. Figure 5.8(a) shows the posture error and tracking time for a subsampling density of $1/3$ and varying numbers of ICP iterations. After five iterations the error is lower than 2° and less than one third of the error of a single ICP iteration. The tracking time increases linearly in the number of iterations. Figure 5.8(b) shows the posture error and tracking time for five ICP iterations and varying subsampling values. The tracking time is approximately 40 ms at a subsampling density of 1/3, which facilitates sufficiently accurate tracking at approximately 25 fps.

PCA-constrained tracking

The main benefits of PCA-constrained tracking are reduced computational complexity and improved posture reconstruction in cases of incomplete or ambiguous input data. To test this, we used a synthetic sequence of a grasping motion, in which the fingers are occluded by the back of the hand during the grasp and thus disappear from the input data. We tracked this sequence once with all DoFs and once with

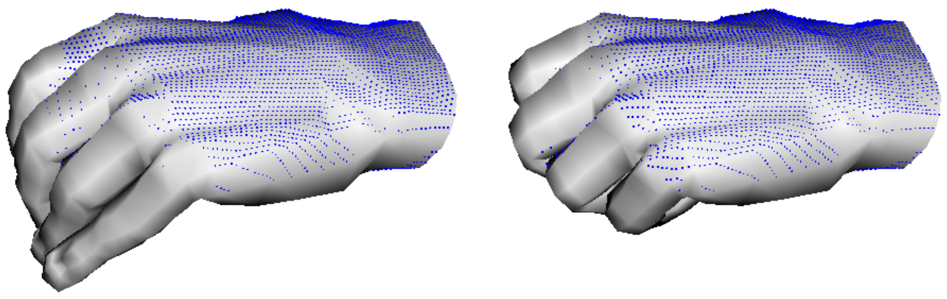
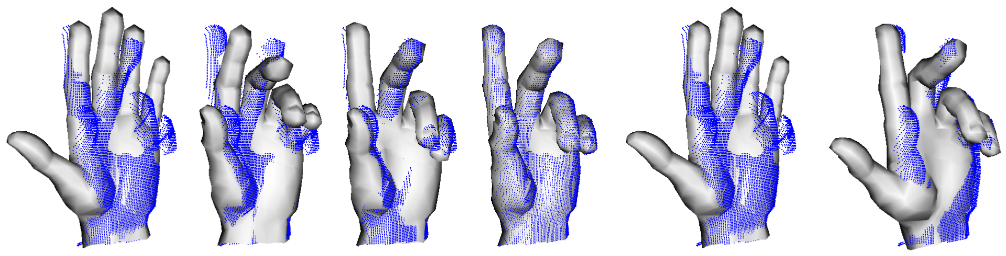


Figure 5.9: Example for posture estimation with incomplete data. The blue points constitute the synthetically created input point cloud. The full-DoF estimation (left) cannot correctly reconstruct the grasp posture because the fingers in the input point cloud are partially occluded. The reduced-DoF estimation (right) is able to reconstruct the posture despite the missing data.

reduced DoFs based on PCA of the grasp training data set. The number of principal components (PCs) was chosen such that 99% of the variance in the data set was covered, which resulted in five PCs. In this experiment, the full-DoF tracking produced an average posture error of about $12°$, whereas the reduced-DoF tracking produced an average posture error of only about $4°$, despite the severe self occlusions present. The full-DoF estimation is less accurate because due to the partly missing correspondence data for individual fingers, the model's fingers stop moving or collapse into the wrong part of the input point cloud. The reduced-DoF estimation reconstructs the input data more accurately because the employed synergies cause the hand's DoFs to move in correlation to each other. This means that only some parts of the hand model need explicit correspondence data in order to approach an accurate reconstruction of the postures underlying the input. An example of this using the mesh hand model can be seen in Figure 5.9. Thanks to the reduced number of DoFs, our PCA-based hand tracking improves runtime performance from 25 fps up to 30 fps in our experiments.

Hierarchical single-frame estimation

Since our hand tracking approach is dependent on temporal coherency in the input data, cases in which a good posture initialization is not known can be difficult to handle. The hierarchical optimization scheme described in Section 5.3.2 alleviates this dependency thanks to a coarse-to-fine estimation process. We confirmed this experimentally by performing single-frame posture estimation with and without hierarchical optimization. The input data for this experiment were several frames of distinct sign language gestures and the only initialization given was the global pose of the hand. The data set used for the PCA was the complete training set of recorded sign language hand trajectories. To facilitate maximally accurate reconstructions, we used the complete point cloud as input and performed 10 non-rigid ICP iterations.



(a) Hierarchical single-frame estimation

(b) Non-hierarchical single-frame estimation

Figure 5.10: Example illustrating hierarchical and non-hierarchical single-frame posture estimation. The blue points constitute the synthetically created input point cloud. The hierarchical estimation successively approximates the input posture with increasing DoFs (from left to right: initial state, 1 DoF, 4 DoF, full DoF). The non-hierarchical estimation gets stuck in a local minimum due to bad initial correspondences (left: initial state, right: full DoF).

In cases of highly self-occluded input data it was impossible for the non-hierarchical full-DoF estimation to produce accurate posture reconstructions from a single frame, resulting in an average posture error of approximately 22° over all input frames. On the other hand, the hierarchical estimation arrived at an accurate reconstruction for *every* posture, resulting in an average posture error of approximately 0.4°. The computation time for each frame was about 10s. Figure 5.10 shows a comparison between hierarchical and non-hierarchical single-frame estimation for a specific posture example using the mesh hand model.

5.5.3 Adaptive PCA model

In the following, we evaluate the performance of our adaptive PCA model compared with non-adaptive approaches using synthetic data. The synthetic images were particularly designed to include a high amount of self-occlusions during complex finger movements.

The generated image sequences were tracked in multiple runs, varying the optimization method (full-DoF, reduced PC-space, adaptive PC-space). For all methods, we measured the difference between the postures generated by our system and the known ground truth postures by computing the average joint angle error. Additionally, we report the average distance between the sensor point cloud and their corresponding points on the hand model surface.

Figure 5.11 shows exemplary results of this evaluation using the capsule hand model. For the depicted posture the standard full-DoF optimization produces an inaccurate posture reconstruction due to highly occluded data, which cause bad correspondences. The reduced-DoF IK approximates the ground truth posture more accurately,

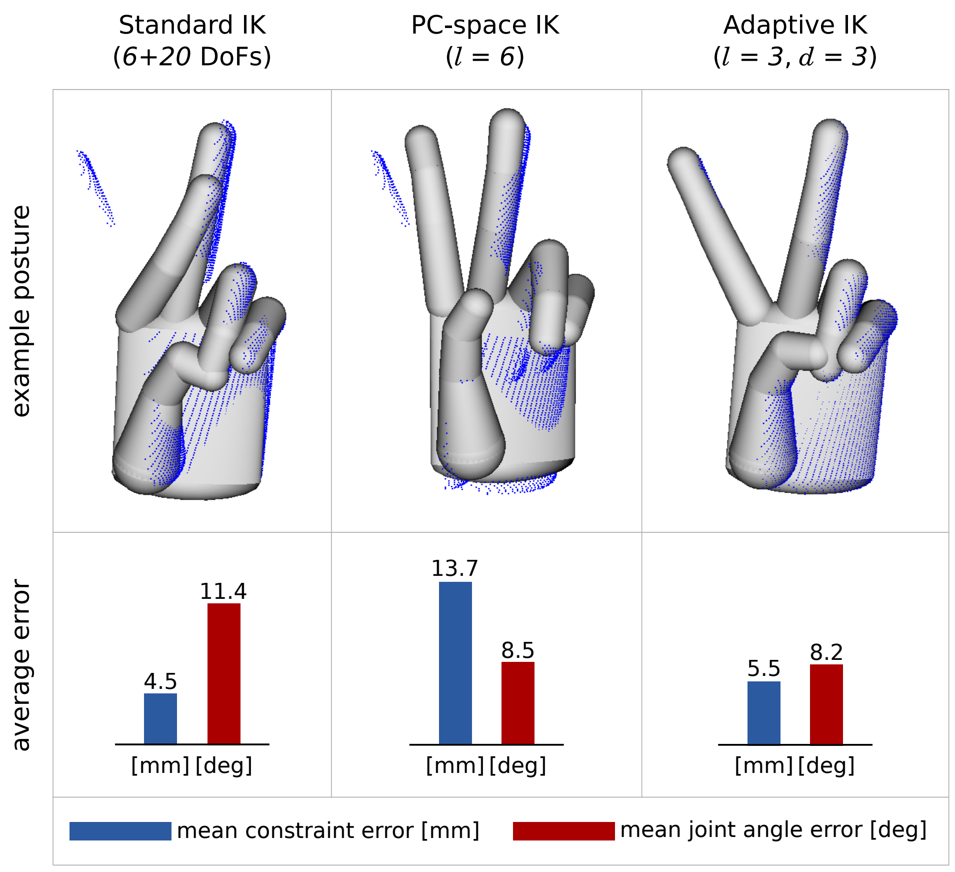


Figure 5.11: Posture reconstruction accuracy using the full 26-dimensional parameter space (left), reduced space with 6 PCs (center), and adaptive subspace with 3 anchor DoFs and 3 corrective DoFs (right). The blue points show the input point cloud for one specific frame. The error values are averages over the entire synthetic sequence.

but due to the inflexibility of the PC-space the hand model is not closely aligned with the point cloud. Our adaptive PCA model produces a result that is closer to the ground truth posture. The average error values over the whole synthetic sequence (Figure 5.11, bottom) reflect these properties of the different optimization methods. The full-DoF estimation is the least accurate in terms of posture recovery, although producing low constraint errors (partly due to wrong correspondences). The adaptive model combines flexible and accurate reconstruction of the hand animation with the robustness of PC-space optimization.

When using an adaptive model with $d = 6$ corrective DoFs instead of $d = 3$, the average constraint error slightly decreases from 5.5 mm to 5 mm, but the average posture error increases from $8.2°$ to $9.4°$. This indicates that the flexibility gained by additional corrective DoFs comes at the price of lower estimation quality when incomplete sensor data produces unreliable correspondences.

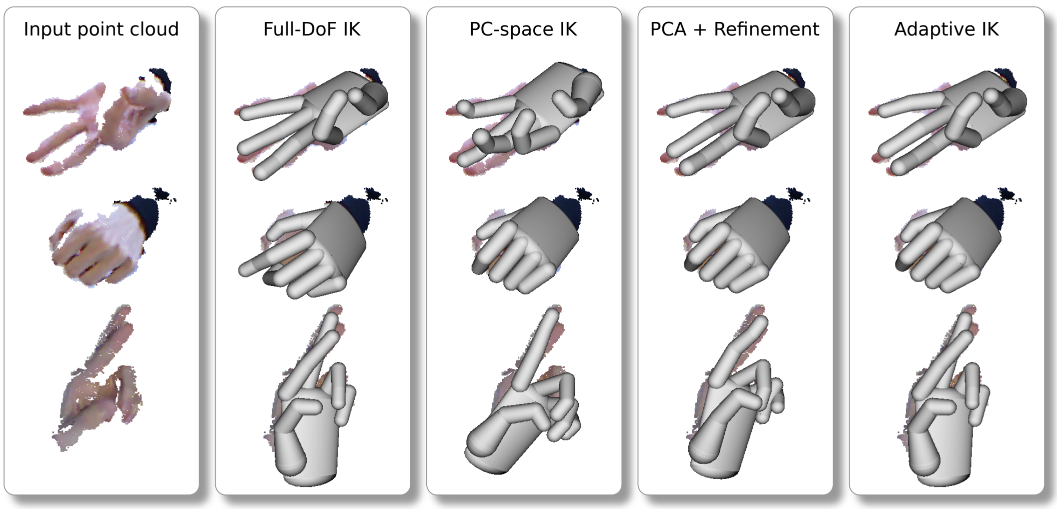


Figure 5.12: Tracking results with Kinect input. The full-DoF estimation cannot correctly recover the middle posture due to self-occlusions. Estimation in the reduced PC-space yields a plausible result in spite of these occlusions, but lacks the flexibility to accurately recover the upper and lower postures. Estimation in the fixed PC-space with subsequent refinement cannot recover the lower posture due to the inaccurate PC-space initialization. Our adaptive model successfully recovers all postures.

5.5.4 Non-adaptive PCA and refinement

Our adaptive method continuously updates the PCA subspace to account for deviations from the fixed PCA model, allowing previously unknown observed postures to be incorporated in a temporally coherent way. Performing local posture refinements without subsequently adapting the subspace causes the estimation to be mainly driven by the fixed PCA model, which can produce inaccurate results in cases of unknown hand articulations.

Figure 5.12, bottom row shows an example where the input hand posture cannot be accurately represented in the fixed PCA subspace (third column). Subsequent full-DoF refinement of the posture based on this initialization without an adaptive model reaches an incorrect local minimum (fourth column). In contrast, estimation with an adaptive model reproduces the input posture well (fifth column), because the adaptive subspace was robustly updated according to the refined observations during the previous frames.

5.5.5 Real-time hand tracking

In practice, most of the idealized conditions of our experimental setup do not necessarily hold true, because the data obtained from the Kinect sensor can be noisy and the virtual hand model used for tracking usually does not perfectly match the user's

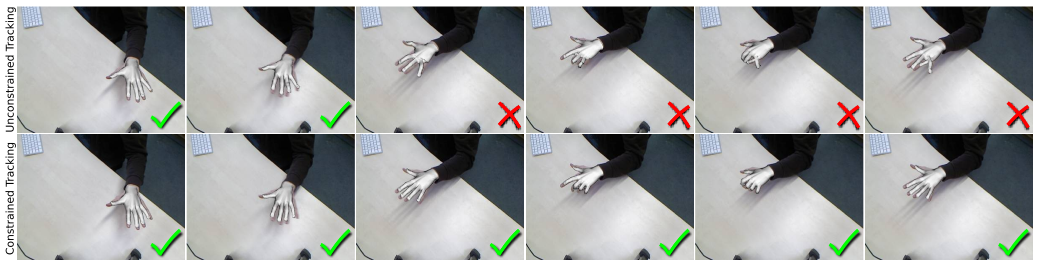


Figure 5.13: Example hand tracking sequence. Top row: full-DoF posture estimation. Bottom row: reduced-DoF posture estimation. The full-DoF posture estimation fails to correctly track the posture during a rapid hand movement. The ring and pinky fingers collapse into the same point cloud segment, resulting in an unnatural posture. The reduced-DoF posture estimation has less mobility but maintains plausible hand postures across the whole sequence.

real hand. We found that sensor noise did not severely impair the tracking quality within the distance of about 0.8 to 1.5 meters. To adjust to the hand sizes of different users we changed the overall scale of the hand model, which gave satisfactory results. This suggested that the geometric details of the virtual hand model are not that important during tracking, as long as the model's overall scale and proportions matched the user's hand.

We furthermore found that using the simplified cylinder-based capsule hand model during tracking did not negatively affect the tracking accuracy and sped up the correspondence search. To evaluate the effect of using a capsule hand model instead of a triangle mesh, we compared the posture reconstruction error using a capsule model and the mesh model in our evaluation framework. In both cases the synthetic input data was rendered using the mesh model. Using full-DoF tracking the average posture error increased from approximately $2°$ to $4°$ using the capsule model. Using reduced-DoF tracking there was virtually no difference in the posture error.

Figure 5.12 shows examples comparing the hand posture reconstruction using the full-DoF, reduced PC-space, reduced PC-space with subsequent refinement and adaptive PC-space optimization for point clouds from a Kinect camera. Our camera setup is arranged with a top-down view of the workspace, which contains minimal clutter to facilitate robust segmentation of the user's hand. Figure 5.13 shows an example of a tracking sequence in which the full-DoF posture estimation results in an unnatural state due to a rapid hand motion causing incomplete sensor data, whereas the reduced-DoF estimation results in a natural approximation of the user's hand. This illustrates the benefit of performing the optimization in a reduced parameter space based on natural hand synergies.

Our tracking system runs at approximately 25 fps with full-DoF tracking and about 30 fps with $(6+6)$-DoF tracking in PC-space. The PCA adaptation procedure involving the full-DoF posture refinement and the computation of the corrective matrix usually takes less than 5 ms to complete and therefore does not negatively impact runtime performance.

5.6 Discussion

In our hand tracking approach, the hand posture is estimated using positional information from a Kinect point cloud as geometric constraints to fit a virtual hand model by means of inverse kinematics. The extension of the method to allow for optimization in a dimensionally reduced PC-space is simple and serves to constrain the parameter space in a meaningful way. Using a varied set of motion capture data as training data set facilitated the derivation of natural hand synergies that covered a wide range of natural hand movements. Using an adaptive PCA model constrains the estimation to realistic hand postures while simultaneously allowing for continuous hand articulation refinements.

Optimizing in a reduced PC-space as opposed to the high-dimensional hand posture space improves runtime performance and prevents implausible hand postures by constraining the estimation to realistic postures. These constraints can reduce the mobility of the posture estimation to some extent, but the overall tracking benefits from the increase in stability and plausibility of the estimated hand postures, especially in cases where the input data is incomplete or ambiguous.

The number of principal components needed to cover most of the variance in the data depends on the number and type of movements contained in the data set. The less varied the captured movements are, the less parameters are needed to represent the most meaningful hand synergies involved. This simplifies the problem of tracking specific movements, such as various types of grasping, to an optimization of only 2 or 3 posture parameters, but impairs the approximation of more general hand movements.

By employing a hierarchical optimization scheme, we are able to overcome the fact that low-DoF estimations are limited by the postures contained in the ground truth data. Our synergistic approach improves estimation quality by explicitly including prior knowledge about the tracked movements. If the types of movements are unknown prior to tracking or a very general posture estimation is desired, our purely IK-based approach yields satisfactory results unless the input data is highly inconsistent.

Using an adaptive PCA model allows for highly efficient continuous refinements of the observed postures while keeping the estimation close to realistic hand articulations from a database. The direct modification of the corrective matrix based on incremental rank-one updates during the online adaptation of the PCA model is efficient and generally useful for applications related to dimension reduction. Usage of sensors with higher frame-rate and resolution will improve the quality of our refinement and adaptation process. While a higher frame-rate requires a larger buffer size N for our adaptive PCA model in order to cover the same time span, this will not negatively impact runtime performance, since our incremental adaptation method is independent from the buffer size N.

Chapter 6

Regularized articulated registration

The task of real-time hand tracking using commodity depth sensors poses a variety of challenges that need to be addressed. In addition to the data from such sensors being prone to noise and artifacts, hand movements themselves are often fast, complex and highly articulated, which leads to occlusions and ambiguities. Discriminative and generative real-time tracking approaches that use data-driven priors to facilitate and constrain the posture estimation can partly deal with these challenges but ultimately impinge on the freedom, expressiveness and accuracy of the reconstructed motion. In order to robustly and flexibly incorporate point cloud and hand model geometries, as well as kinematic and data-driven articulation priors in a unified approach, hand tracking can be expressed as a regularized geometric registration problem. In this chapter, an articulated registration approach is presented, which estimates the state of the user's hand in real-time by fitting an articulated template model to the sensor point cloud while explicitly taking into account visibility and occlusions, as well as employing a set of priors that regularize the estimation to ensure plausibility even with strongly degrading sensor data.

In the system we developed (Tagliasacchi & Schröder et al. 2015), the hand model is registered to the sensor input data using depth, silhouette, and temporal information. To effectively map low-quality depth maps to realistic hand postures, the registration is regularized with kinematic and temporal priors, as well as a data-driven prior built from a database of realistic hand postures. This ensures the inferred hand postures are plausible and helps the algorithm in recovering from loss of tracking. We present a principled way of integrating such priors into the registration optimization to enable robust tracking without severely restricting the freedom of motion. Our articulated registration algorithm efficiently integrates these data and regularization priors into a *unified* real-time solver. A core technical contribution is a new method for computing tracking correspondences that directly models occlusions typical of single-camera setups. To this end, our method combines 2D and 3D registration to align the hand model the the acquired depth map and extracted silhouette image, and data-to-model correspondences are computed in a way that accounts for visibility and occlusions, which significantly improves the robustness of the tracking.

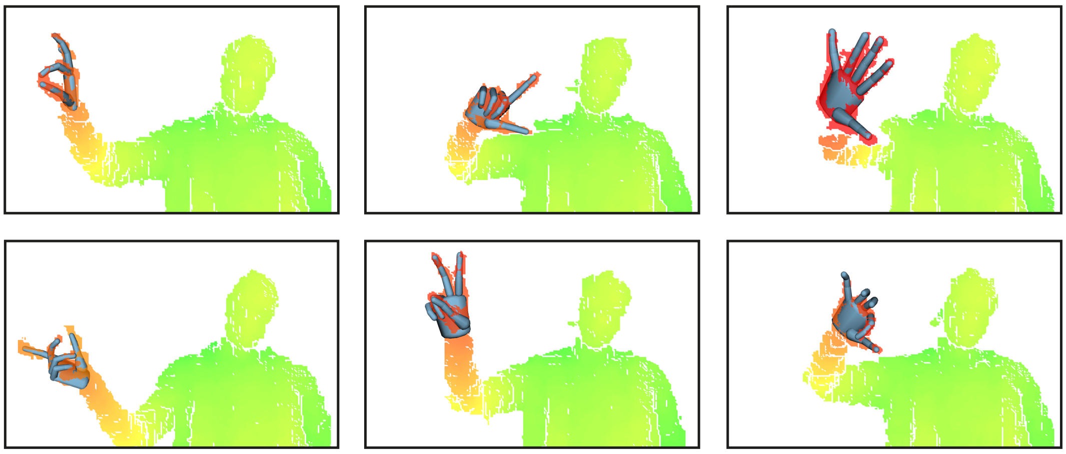


Figure 6.1: Our method captures hand motions in real-time using a single depth camera based on an online registration process that accurately and robustly reconstructs hand postures by fitting an articulated hand model to depth images.

Notably, there is a widespread belief (Wei et al. 2012, Zhang et al. 2014, Qian et al. 2014) that ICP-like techniques are too local and prone to local minima to successfully deal with fast articulated motion. One of our contributions is to show this commonly held belief should be reconsidered. We demonstrate that a regularized geometric registration approach in the spirit of ICP can achieve outstanding performance, and therefore represents a highly viable alternative to commonly employed techniques from the vision community like discriminative (Tompson et al. 2014) and PSO (Oikonomidis et al. 2011a, Qian et al. 2014, Sharp et al. 2015) methods.

Our regularized geometric registration achieves robust, highly articulated hand tracking at up to 120 frames per second (fps). We quantitatively and qualitatively compare the performance of our algorithm to recent appearance-based and model-based techniques (see Section 6.5). These comparisons show a significant improvement in accuracy and robustness compared to the current state-of-the-art. The source code of our implementation is publicly available[2].

6.1 Related work

The main focus of the work presented in this chapter is on improving the robustness and accuracy of model-based hand tracking approaches by combining effective 2D and 3D registration energies with carefully designed priors. While our work is targeted towards using a single commodity depth sensor, accurate model-based tracking has been achieved in multiple camera setups (Sridhar et al. 2013, 2014), where the multiple vantage points help resolving challenging occlusions. Multiple camera sys-

[2]`https://github.com/OpenGP/htrack`

tems have also been used successfully to model precise hand-hand and hand-object interactions (Oikonomidis et al. 2011b, Ballan et al. 2012, Wang et al. 2013). All of these methods require a complex acquisition setup and manual calibration, which makes them less suitable for the kind of consumer-level applications that we target with this work.

Particle swarm optimization (PSO) methods achieve interactive (15 fps) tracking with a single RGBD camera (Oikonomidis et al. 2011a). PSO techniques have also been applied successfully to model challenging interaction between two hands (Oikonomidis et al. 2012) at a reduced rate of 4 fps. PSO is an optimization heuristic that does not use the gradient information of the considered optimization problem, but instead uses a stochastic sampling strategy. For this reason the accuracy and efficiency of PSO approaches heavily rely on the number of samples used. Oikonomidis et al. (2014) introduced a more advanced sampling strategy that improves tracking efficiency without compromising quality. Sharp et al. (2015) combined the PSO model-fitting technique with a per-frame reinitializer for recovery from loss of tracking, which increased robustness. However, gradient-based optimization approaches converge faster and more accurately than PSO when close to the solution, and are therefore well suited for real-time applications (Qian et al. 2014).

Compelling 60 fps real-time performance was recently shown using gradient-based optimization by Melax et al. (2013), where the optimization is expressed as a convex rigid body simulation, and numerous heuristics for reinitialization were employed to avoid tracking failures. Rather than resorting to reinitialization for robustness, in (Schröder et al. 2014) we formulated the optimization in a subspace of likely hand postures. While the lower number of optimization variables leads to efficient computations, tracking accuracy can be limited by the reduced posture complexity induced by the subspace. We compare the method presented in this chapter to these algorithms and demonstrate substantial improvement in tracking robustness and accuracy.

Regarding recent appearance-based methods (Tompson et al. 2014, Tang et al. 2014), we demonstrate comparable or improved performance when tracking hands in isolation. We also highlight the potential of model-based tracking by observing how appearance-based methods suffer from severe loss of tracking when extraneous geometry is introduced in proximity of the hand. In contrast, our technique is able to remain stable during data aberrations typical in hand-hand, hand-object interactions.

In this work, we show that hand tracking can be formulated as a single gradient-based optimization to obtain an efficient and accurate real-time tracking system running at up to 120 fps. By using a combination of geometric and data-driven priors we achieve significant improvements in tracking quality and robustness.

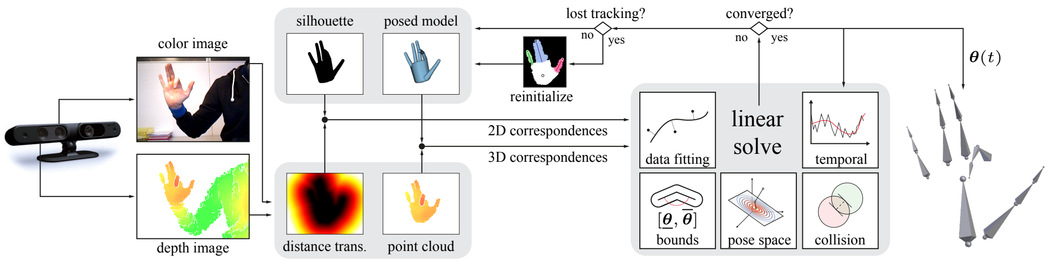


Figure 6.2: Overview of our algorithm. For each acquired frame we extract a 3D point cloud of the hand and the 2D distance transform of its silhouette. From these we compute point correspondences to align a cylinder model of the hand to best match the data. This registration is performed in an ICP-like optimization that incorporates a number of regularizing priors to ensure accurate and robust tracking.

6.2 Articulated ICP hand tracking

Robust hand tracking with a commodity depth sensor is highly challenging due to self-occlusion, low quality/density of sensor data and the high degree of articulation of the human hand. We address these issues by proposing a regularized articulated ICP-like optimization that carefully balances data fitting with suitable priors (Figure 6.2). Our data fitting performs a joint *2D-3D* optimization. The 3D alignment ensures that every point measured by the sensor is sufficiently close to the tracked model $\mathcal{H}$. Simultaneously, as we cannot create such constraints for occluded parts of the hand, we integrate a 2D registration that pushes the tracked model to lie within the sensor visual hull. A carefully chosen set of priors regularizes the solution to ensure the recovered posture is plausible.

Acquisition device

Our system processes raw data acquired at 60 fps from a single RGBD sensor. Figure 6.3 illustrates this data for the *PrimeSense Carmine 1.09* structured light sensor as well as the *Creative Gesture Camera* time-of-flight sensor. From the raw data our algorithm extracts a 2D *silhouette image* $\mathcal{S}_s$ and a 3D *point cloud* $\mathcal{X}_s$. The two sensors exhibit different types of imperfections. The precision of depth measurements in the PrimeSense camera is significantly higher. However, substantial holes often occur at grazing angles, e.g. note the gap in the data where we would expect to see the index finger. Conversely, the Creative Gesture Camera provides an accurate and gap-free silhouette image, but suffers from high noise in the depth measurements, therefore resulting in very noisy point clouds. Our algorithm is designed to handle both types of imperfections. This is achieved by formulating an optimization that *jointly* considers silhouette and point cloud, balancing their contribution in a way that conforms to the quality of sensor data.

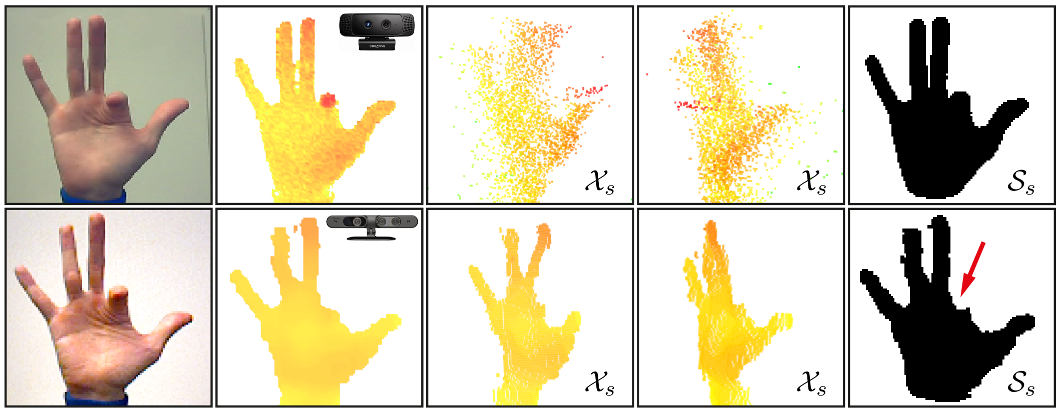


Figure 6.3: The two different sensors used in our experiments provide data with substantially different characteristics. Top: Intel's Creative Interactive Gesture camera (time of flight) provides a complete silhouette image, but low quality depth measurements, resulting in severe point cloud noise. Bottom: Point clouds acquired by the PrimeSense camera (structured light) are much smoother, but the silhouette image can contain significant gaps.

Tracking model

Our algorithm registers a template hand model to the sensor data. Similar to other techniques (Oikonomidis et al. 2011a, Schröder et al. 2014), we employ a simple (sphere capped) cylinder model as a geometric template; see Figure 6.4. We optimize for 26 degrees of freedom, 6 for global rotation and translation and 20 for articulation. Like in (Melax et al. 2013), the model can be quickly adjusted to the user by specifying global scale, palm size and finger lengths. In most scenarios, it is sufficient to perform a simple uniform scaling of the model. Such a coarse geometry is sufficient for hand tracking, as the signal-to-noise ratio for commercially available RGBD sensors is low for samples on the fingers when compared to the size of a finger. Furthermore, the computation of closest-point correspondences can be performed in closed form and in parallel, which is essential for real-time performance. While the geometry of the model used for tracking remains coarse, our algorithm computes joint angles (including rigid transformation) in the widespread *BVH* motion sequence format; these can be used to drive a high-resolution skinned hand rig as illustrated in Figure 6.4.

Preprocessing

The silhouette image $\mathcal{S}_s$ is not directly available from the sensor and needs to be computed. This labeling can be obtained by extracting the sensor color image and performing a skin color segmentation (Oikonomidis et al. 2012, Schröder et al. 2014), or can be obtained directly from depth images by performing a classification with randomized forests (Tompson et al. 2014). Another possibility is to exploit a

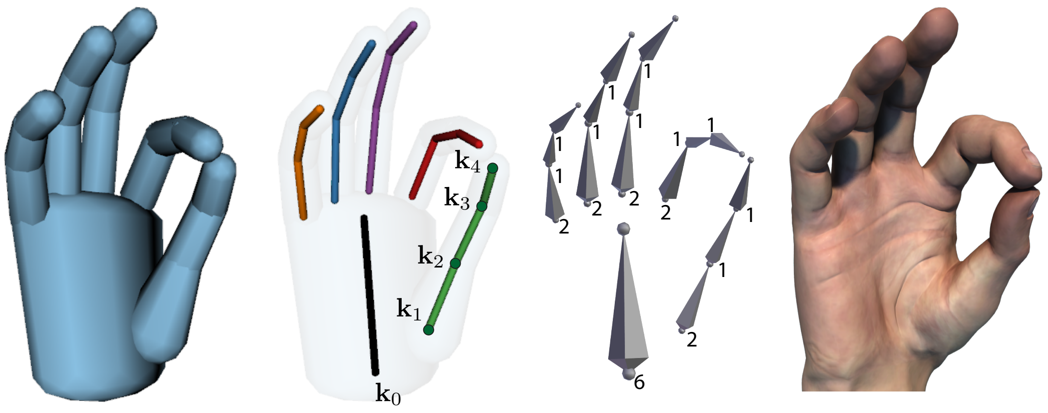


Figure 6.4: A visualization of the template hand model with the number and location of degrees of freedom of our optimization. From left to right: The cylinder model used for tracking, the skeleton, the BVH skeleton exported to Maya to drive the rendering, the rendered hand model.

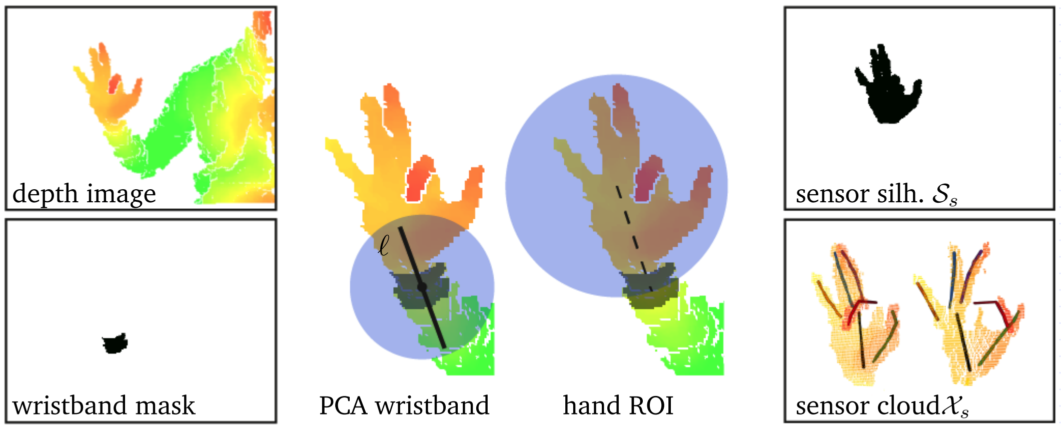


Figure 6.5: We first identify the wristband mask by color segmentation, then compute the 3D orientation of the forearm as the PCA axis of points in its proximity. Offsetting a 3D sphere from the wristband center allows isolating the region of interest. The obtained silhouette image and sensor point clouds are shown on the right.

full-body tracking algorithm (Shotton et al. 2011) and segment the hand according to the wrist position. For gestural tracking, where the hand is typically the closest object to the sensor (Qian et al. 2014), a black wristband can be used to simplify segmentation by creating a gap in the depth image. Similarly to this method, in our system the user wears a *colored wristband*. We first identify the position of the wristband in the scene by color segmentation, then retrieve the 3D points in the proximity of the wristband and compute the principal axis. This axis, in conjunction with the wristband centroid, is then used to segment the hand point cloud. Any depth pixel within the hand point cloud is labeled as belonging to the silhouette image $\mathcal{S}_s$ as shown in Figure 6.5.

6.3 Energy optimization

In this section we derive the objective functions of our model-based optimization method and provide the rationales for our design choices. Let $\mathcal{F}$ be the sensor input data consisting of a 3D point cloud $\mathcal{X}_s$ and 2D silhouette $\mathcal{S}_s$ (see Figure 6.3). Given a 3D hand model $\mathcal{H}$ with joint parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_{26})$, we aim at recovering the pose and posture $\boldsymbol{\theta}$ of the user's hand, matching the sensor input data $\mathcal{F}$. To achieve this goal, we solve the optimization problem

$$\min_{\boldsymbol{\theta}} \underbrace{E_{\text{3D}} + E_{\text{2D}} + E_{\text{wrist}}}_{\text{Fitting terms}} + \underbrace{E_{\text{posture}} + E_{\text{kinematic}} + E_{\text{temporal}}}_{\text{Prior terms}}, \tag{6.1}$$

combining fitting terms that measure how well the hand parameters $\boldsymbol{\theta}$ represent the data frame $\mathcal{F}$, with prior terms that regularize the solution to ensure realistic hand postures. For brevity of notation we omit the arguments $\boldsymbol{\theta}, \mathcal{X}_s, \mathcal{S}_s$ of the energy terms. We first introduce the fitting terms and present our new solution to compute tracking correspondences. Then we discuss the prior terms and highlight their benefits in terms of tracking accuracy and robustness. More details on the implementation of the optimization algorithm is given in Section 6.4.

6.3.1 Fitting Energies

For the geometric fitting energies, we distinguish between 3D and 2D correspondences. This makes it possible to deal with discrepancies between the observed hand and the hand model. Considering correspondences in only one direction (e.g. only sensor to model as in Schröder et al. 2014) results in a fitting bias, which deteriorates tracking quality. By considering 3D constraints for observed point cloud positions and 2D constraints in image space for the rendered hand model, we make use of all available data, which facilitates a much closer fit between the model and the data.

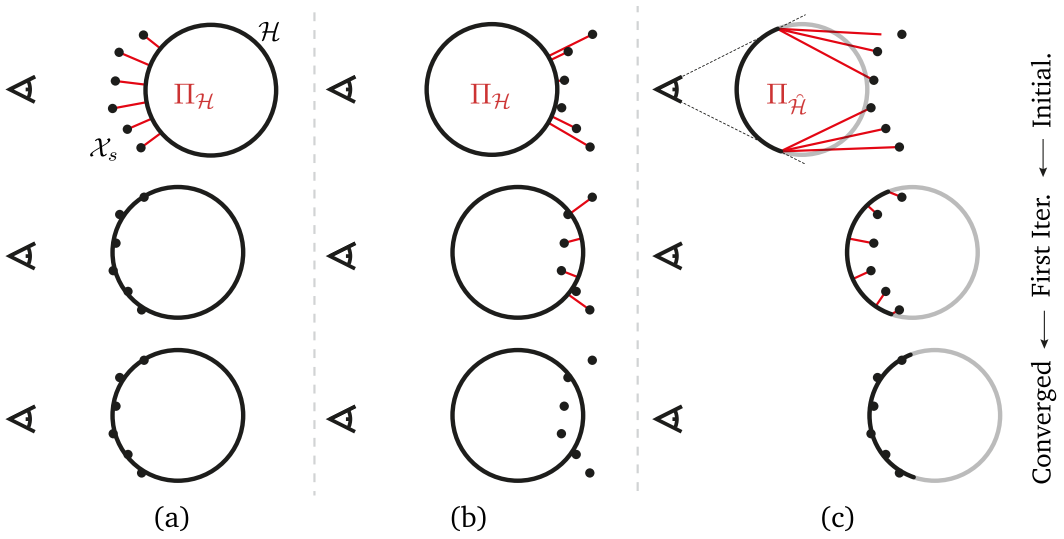


Figure 6.6: Illustration of correspondences computations. The circles represent cross-sections of the fingers, the small black dots are samples of the depth map. (a) A configuration that can be handled by standard closest point correspondences. (b) Closest point correspondences to the back of the cylinder model can cause the registration to fall into a local minimum. Note that simply pruning correspondences with back-pointing normals would not solve this issue, as no constraints would remain to pull the finger towards the data. (c) This problem is resolved by taking visibility into account, and computing closest points only to the portion of the model facing the camera.

Point cloud alignment

The term E_{3D} models a 3D geometric registration in the spirit of ICP as

$$E_{3D} = \omega_1 \sum_{\mathbf{t} \in \mathcal{X}_s} \|\mathbf{t} - \Pi_{\mathcal{H}}(\mathbf{t}, \boldsymbol{\theta})\|_2, \tag{6.2}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm, $\mathbf{t}$ represents a 3D point of $\mathcal{X}_s$, and $\Pi_{\mathcal{H}}(\mathbf{t}, \boldsymbol{\theta})$ is the projection of $\mathbf{t}$ onto the hand model $\mathcal{H}$ with hand posture $\boldsymbol{\theta}$. Note that we compute a sum of absolute values of the registration residuals, not their squares. This corresponds to a mixed ℓ_2/ℓ_1 norm of the stacked vector of the residuals. For 3D registration such a sparsity-inducing norm has been shown to be more resilient to noisy point clouds containing a certain amount of outliers such as the ones produced by the Creative sensor (Figure 6.3). We refer to (Bouaziz, Tagliasacchi and Pauly 2013) for more details.

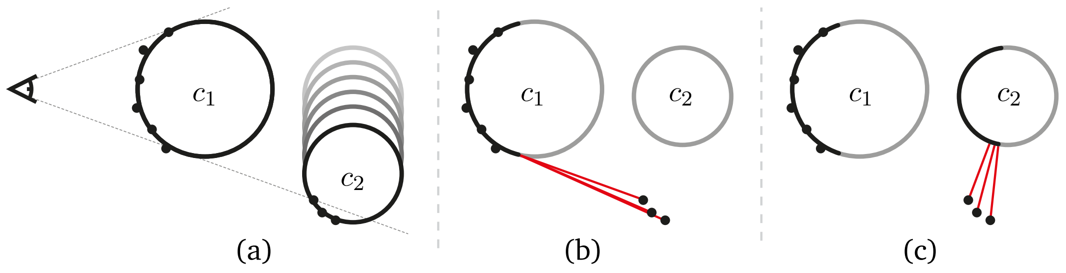


Figure 6.7: Illustration of the impact of self-occlusion in correspondences computations. (a) The finger c_2 initially occluded by finger c_1 becomes visible, which causes new samples to appear. (b) Closest correspondences to the portion of the model visible from the camera do not generate any constraints that pull c_2 toward its data samples. This is the approach in (Wei et al. 2012), where these erroneous matches are then simply pruned. (c) Our method also considers front-facing portions of the model that are occluded, allowing the geometry to correctly register.

3D correspondences

The 3D registration terms involves computing the corresponding point $\mathbf{y} = \Pi_{\mathcal{H}}(\mathbf{t}, \boldsymbol{\theta})$ on the cylinder model $\mathcal{H}$ for each sensor point $\mathbf{t} \in \mathcal{X}_s$. In contrast to standard closest point search, we define the correspondence $\mathbf{y}$ as the closest point on the *front-facing* part $\hat{\mathcal{H}}$ of $\mathcal{H}$. This includes parts of the model that are oriented towards the camera but occluded by other parts. In our experiments we learned that this seemingly simple extension proved absolutely essential to obtain high-quality tracking results. Only considering model points that are visible from the sensor viewpoint, i.e., matching to the rendered model, is not sufficient for handling occlusions or instances of disappearing and reappearing sensor data; see Figure 6.6 and Figure 6.7.

To calculate $\mathbf{y}$, we first compute the closest points $\mathbf{t}_{\mathcal{C}}$ of $\mathbf{t}$ to each cylinder $\mathcal{C} \in \mathcal{H}$. Recall that our hand model consists of sphere-capped cylinders so these closest points can be computed efficiently in closed form and in parallel for each $\mathbf{t} \in \mathcal{X}_s$. We then identify back-facing points using the dot product of the cylinder surface normal $\mathbf{n}$ at $\mathbf{t}_{\mathcal{C}}$ and the view ray vector $\mathbf{v}$. For efficiency reasons, we use a simplified orthographic camera model where the view rays are constant, i.e., $\mathbf{v} = [0\ 0\ 1]^T$. If a point on a cylinder is back-facing ($\mathbf{n}^T\mathbf{v} > 0$), we project $\mathbf{t}$ onto the cylinder's silhouette contour line from the camera perspective, whose normals are orthogonal to $\mathbf{v}$.

A different strategy to address visibility issues has been introduced in (Qian et al. 2014). These methods propose an energy that penalizes areas of the model falling in front of the data, which is then optimized using particle swarms. This energy can be integrated into our optimization following the formulation in (Wei et al. 2012, Eq. 15). However, such an energy is prone to create local minima in gradient-based optimization, as illustrated in Figure 6.8. Here the thumb has difficulty entering the palm region, as it must occlude palm samples before reaching its target configuration. Our correspondence search avoids such problems. Furthermore, Qian et al. (2014)

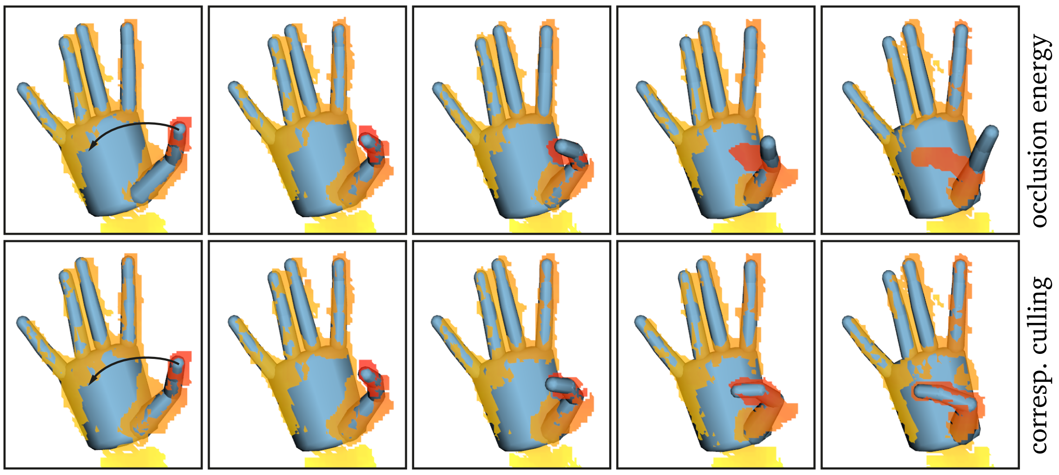


Figure 6.8: Correspondence computations. The top row shows the strategy used in (Qian et al. 2014) adapted to our gradient-based framework according to the formulation given in (Wei et al. 2012). The bottom row shows the improved accuracy of our new approach.

follow a *hypothesize-and-test* paradigm where visibility constraints in the form of *ray-casting* are easy to include. As discussed in (Ganapathi et al. 2012), such constraints are much more difficult to include in iterative optimization techniques like ours. However, our front-facing correspondences computation provides a simple and elegant way to deal with such shortcomings.

Silhouette alignment

The 3D alignment energy $E_{3\mathrm{D}}$ robustly measures the distance between every point in the 3D point cloud $\mathcal{X}_s$ to the tracked model $\mathcal{H}$. However, as hands are highly articulated, significant self-occlusions are common during tracking. Such self-occlusions are challenging, because occluded parts will not be constrained when only using a 3D alignment energy. For this reason, we use a 2D silhouette term $E_{2\mathrm{D}}$ that models the alignment of the 2D silhouette of our rendered hand model with the 2D silhouette extracted from the sensor data as

$$E_{2\mathrm{D}} = \omega_2 \sum_{\mathbf{p}\in\mathcal{S}_r} \|\mathbf{p} - \Pi_{\mathcal{S}_s}(\mathbf{p},\boldsymbol{\theta})\|_2^2, \tag{6.3}$$

where $\mathbf{p}$ is a 2D point of the *rendered* silhouette $\mathcal{S}_r$, and $\Pi_{\mathcal{S}_s}(\mathbf{p},\boldsymbol{\theta})$ denotes the projection of $\mathbf{p}$ onto the *sensor* silhouette $\mathcal{S}_s$. Figure 6.9 shows why the silhouette term is crucial to avoid erroneous postures when parts of the model are occluded. Without the silhouette energy, the occluded fingers can mistakenly move to wrong locations, since they are not constrained by any samples in the depth map.

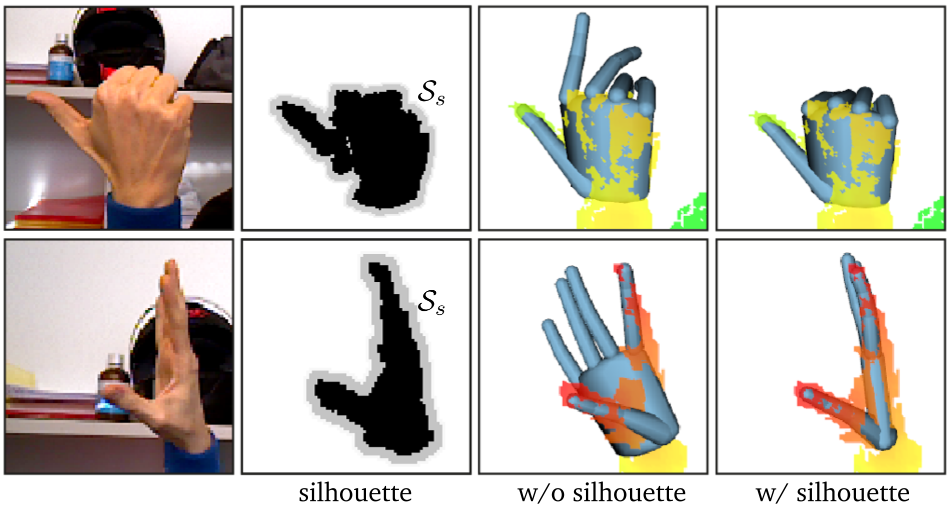


Figure 6.9: Our 2D silhouette registration energy is essential to avoid tracking errors for occluded parts of the hand. When no depth data is available for certain parts of the model, a plausible posture is inferred by ensuring that the model is contained within the sensor silhouette image $\mathcal{S}_s$.

2D correspondences

In Equation (6.3), we compute the silhouette image $\mathcal{S}_r$ by first rendering the hand model $\mathcal{H}$ from the viewpoint of the sensor, caching the bone identifier and the 3D location associated with each pixel in a texture. The projection function $\Pi_{\mathcal{S}_s}(\mathbf{p}, \boldsymbol{\theta})$ to compute the closest corresponding point to the sensor silhouette is evaluated efficiently using the 2D distance transform of $\mathcal{S}_s$. We use the linear time algorithm of (Felzenszwalb and Huttenlocher 2012) and store at every pixel the index to the closest correspondence.

Wrist alignment

The inclusion of the forearm for hand tracking has been shown beneficial in (Melax et al. 2013). Our wrist alignment energy encodes a much simplified notion of the forearm in the optimization that enforces the the wrist joint to be located along its axis.

$$E_{\text{wrist}} = \omega_3 \|\Pi_{2\text{D}}\left(\mathbf{k}_0(\boldsymbol{\theta})\right) - \Pi_{\ell}(\mathbf{k}_0(\boldsymbol{\theta}))\,\|_2^2. \tag{6.4}$$

Minimizing this energy helps preventing the hand from erroneously rotating/flipping during tracking. Here $\mathbf{k}_0$ is the 3D position of the wrist joint, and ℓ is the 2D line extracted by PCA of the 3D points associated with the wristband; see Figure 6.5. Note that $\Pi_{2\text{D}}$ causes residuals to be minimized in screen-space, therefore the optimization of this energy will be analogous to the one of Equation (6.3). We optimize in screen space because, due to occlusion, we are only able to observe half of the wrist and this causes its axis to be shifted toward the camera.

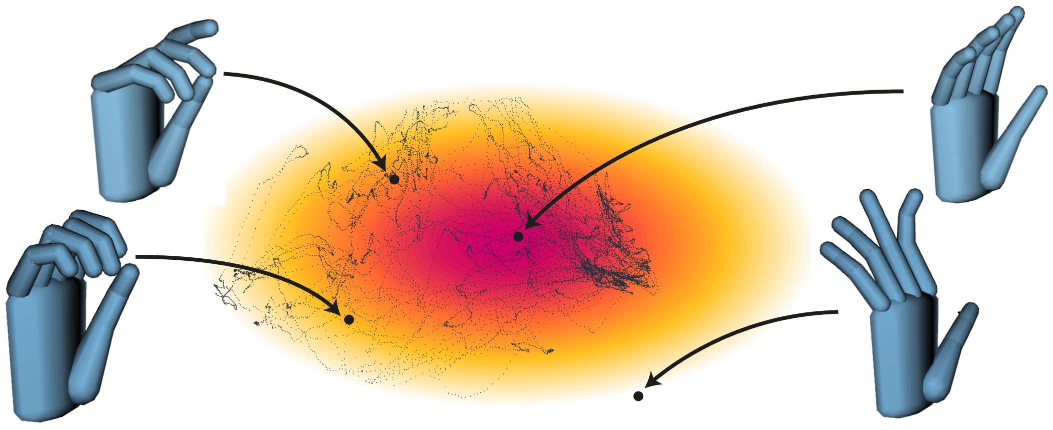


Figure 6.10: An illustration of the PCA posture-space used to regularize the optimization. Black dots denote the samples of the data base. High likelihood postures are located nearby the mean of the latent space (dark red). The eigenvalues of the PCA define the metric in the low-dimensional space, skewing it in certain directions. Postures that, according to this metric, are far from the mean are likely to be unnatural and will be penalized in the optimization.

6.3.2 Prior Energies

Minimizing the fitting energies alone easily leads to unrealistic or unlikely hand postures, due to the deficiencies in the input data caused by noise, occlusions, or motion blur. We therefore regularize the registration with data-driven, kinematic, and temporal priors to ensure that the recovered hand postures are plausible. Each of these terms plays a fundamental role in the stability of our tracking algorithm, as we illustrate below.

Data-driven prior

The complex and highly coupled articulation of human hands is difficult to model directly with geometric or physical constraints. Instead, we use our publicly available database of recorded hand postures (Schröder et al. 2014) to create a data-driven prior E_{posture} that encodes this coupling. We construct a low-dimensional subspace of plausible postures by performing dimensionality reduction using PCA (see Figure 6.10). We enforce the hand parameters $\boldsymbol{\theta}$ to lie close to this low-dimensional linear subspace using a data term $E_{\text{posture}} = E_{\text{projection}} + E_{\text{mean}}$. To define the data term, we introduce auxiliary variables $\tilde{\boldsymbol{\theta}}$, i.e, the PCA weights, representing the (not necessarily orthogonal) projection of the hand posture $\boldsymbol{\theta}$ onto the subspace; see Figure 6.11. The projection energy measures the distance between the hand parameters and the linear subspace as

$$E_{\text{projection}} = \omega_4 \|(\boldsymbol{\theta} - \boldsymbol{\mu}) - \Pi_{\mathcal{P}} \tilde{\boldsymbol{\theta}}\|_2^2, \tag{6.5}$$

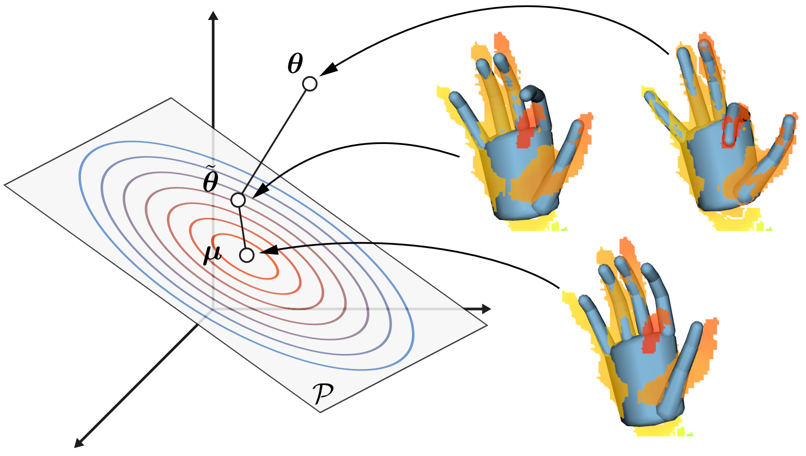


Figure 6.11: An illustration of the energies involved in our posture-space prior. For illustration purposes the full dimensional parameter vector $\boldsymbol{\theta} \in \mathbb{R}^3$, while latent space variable $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^2$. The PCA optimization in (Schröder et al. 2014) constrains the posture parameters $\boldsymbol{\theta}$ to lie on the subspace $\mathcal{P}$. Conversely, we penalize the distance of our posture from $\mathcal{P}$ (Equation (6.5)); simultaneously, we ensure our posture remains likely by preventing it from diverging from the mean of the distribution (Equation (6.6)).

where $\boldsymbol{\mu}$ is the PCA mean. The matrix $\Pi_{\mathcal{P}}$, i.e., the PCA basis, reconstructs the hand posture from the low-dimensional space. To avoid unlikely hand postures in the subspace, we regularize the PCA weights $\tilde{\boldsymbol{\theta}}$ using an energy

$$E_{\text{mean}} = \omega_5 \|\boldsymbol{\Sigma}\tilde{\boldsymbol{\theta}}\|_2^2. \tag{6.6}$$

$\boldsymbol{\Sigma}$ is a diagonal matrix containing the inverse of the standard deviation of the PCA basis. Our tracking optimization is modified to consider the posture space by introducing the auxiliary variable $\tilde{\boldsymbol{\theta}}$ and then jointly minimizing over $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}$. The difference between our approach and optimizing directly in the subspace is further discussed in Section 6.4.4.

The regularization energy in Equation (6.6) helps the tracking system recover from tracking failures. When no sensor constraints are imposed on the model, the optimization will attempt to push the posture toward the mean – a statistically likely posture from which tracking recovery is highly effective.

Figure 6.13 illustrates how the PCA data prior improves tracking by avoiding unlikely postures, in particular when the input data is incomplete. We found that even when data coverage is sufficient to recover the correct posture, the data term improves the convergence of the optimization as illustrated in Figure 6.12. Figure 6.14 shows how our regularized projective PCA formulation outperforms the direct subspace optimization proposed in previous work.

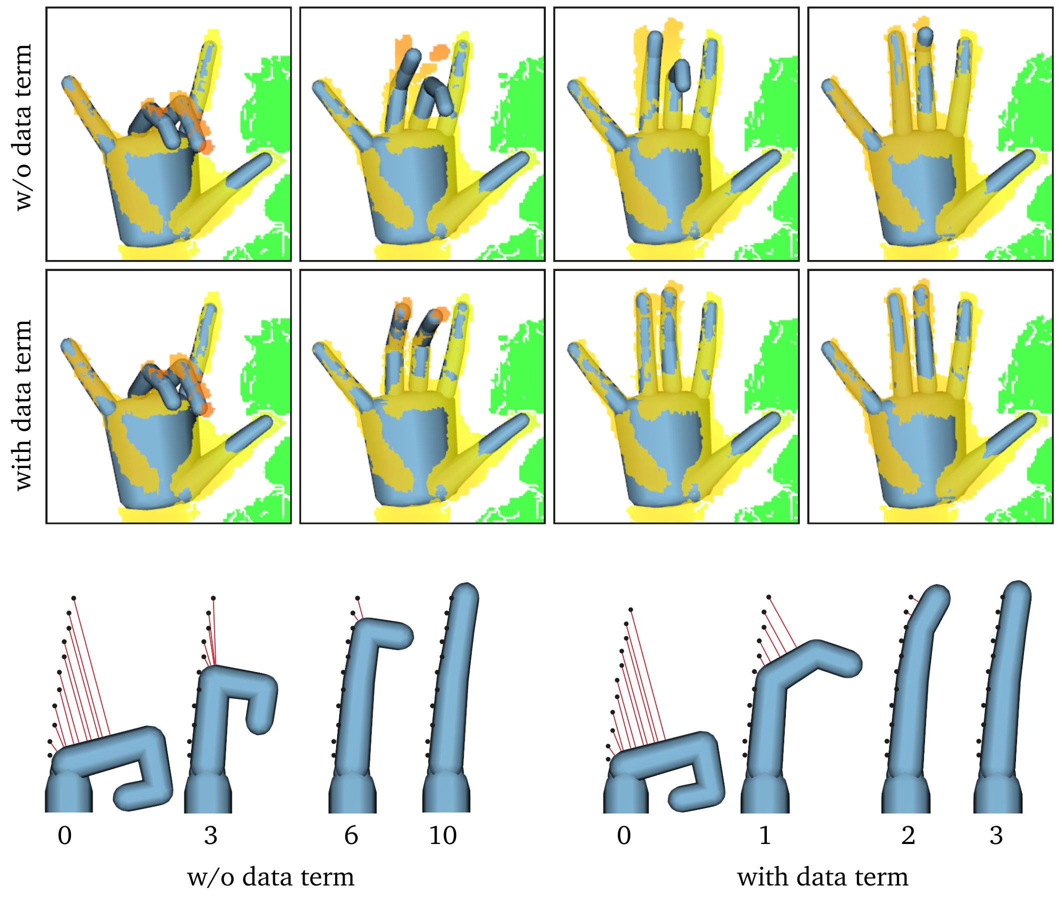


Figure 6.12: Beyond favoring natural postures, the data prior term also positively affects convergence speed. Top: With the same number of iterations, only with activated data term does the model fully register to the scan. The illustration below shows how the same final state requires significantly fewer iterations with the data term.

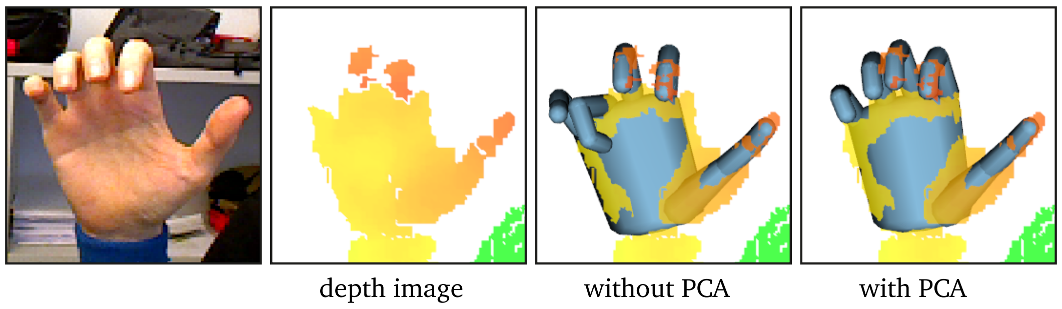


Figure 6.13: Our posture-space regularization using a PCA prior ensures that a meaningful posture is recovered even when significant holes occur in the input data.

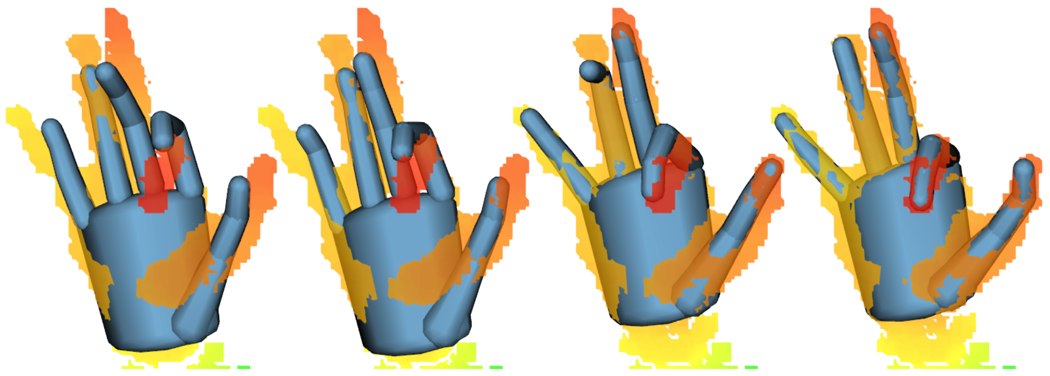


Figure 6.14: Optimizing directly in the PCA subspace (Schröder et al. 2014) can lead to inferior registration accuracy. We replicate this behavior by setting ω_4 in Equation (6.5) to a large value. Even when increasing the number of PCA bases to cover 99% of the variance in the database, the model remains too stiff to conform well to the input. Our approach is able to recover the correct hand posture by optimizing for projection distances even with a very limited number of bases (right).

Kinematic prior

The PCA data term is a computationally efficient way of approximating the space of plausible hand postures. However, the PCA model alone cannot guarantee that the recovered posture is realistic. In particular, since the PCA is symmetric around the mean, fingers bending backwards beyond the physically realistic joint angle limits are not penalized by the data prior. Similarly, the PCA model is not descriptive enough to avoid self-intersections of fingers. These two aspects are addressed by the kinematic prior $E_{\text{kinematic}} = E_{\text{collision}} + E_{\text{bounds}}$. Under the simplifying assumption of a cylinder model, we can define an energy $E_{\text{collision}}$ that accounts for the inter-penetration between each pair of (sphere-capped) cylinders:

$$E_{\text{collision}} = \omega_6 \sum_{\{i,j\}} \chi(i,j)(d(\mathbf{c}_i, \mathbf{c}_j) - r)^2, \tag{6.7}$$

where the function $d(\cdot,\cdot)$ measures the Euclidean distance between the cylinders axes $\mathbf{c}_i$ and $\mathbf{c}_j$, and r is the sum of the cylinder radii. $\chi(i,j)$ is an indicator function that evaluates to one if the cylinders i and j are colliding, and to zero otherwise. The top row of Figure 6.15 shows how this term avoids interpenetrations of the fingers.

To prevent the hand from reaching an impossible posture by overbending the joints, we limit the joint angles of the hand model:

$$E_{\text{bounds}} = \omega_7 \sum_{\theta_i \in \boldsymbol{\theta}} \underline{\chi}(i)(\theta_i - \underline{\theta}_i)^2 + \overline{\chi}(i)(\theta_i - \overline{\theta}_i)^2, \tag{6.8}$$

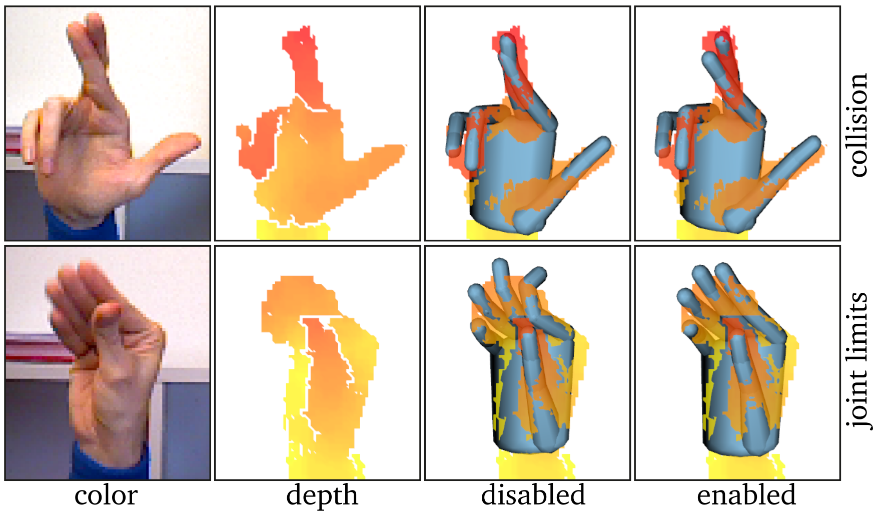


Figure 6.15: Kinematic priors augment the data prior to account for inconsistencies in the posture space. The collision term avoids self-collisions (top row), while the term for joint angle bounds avoids overbending of the finger joints.

where each hand joint is associated with conservative bounds $\left[\underline{\theta}_i, \overline{\theta}_i\right]$. For the bounds, we use the values experimentally determined by Chan and Dubey (1995). $\underline{\chi}(i)$ and $\overline{\chi}(i)$ are indicator functions. $\underline{\chi}(i)$ evaluates to one if $\theta_i < \underline{\theta}_i$, and to zero otherwise. $\overline{\chi}(i)$ is equal to one if $\theta_i > \overline{\theta}_i$, and zero otherwise. The bottom row of Figure 6.15 illustrates the effect of the joint angle bounds.

Temporal prior

A common problem in particular with appearance-based methods are small-scale temporal oscillations that cause the tracked hand to jitter. A standard way to enforce temporal smoothness is to penalize the change of model parameters $\boldsymbol{\theta}$ through time, for example, by penalizing a quadratic energy accounting for velocity $||\dot{\boldsymbol{\theta}}||^2$ and acceleration $||\ddot{\boldsymbol{\theta}}||^2$ (Wei et al. 2012). However, if we consider a perturbation of the same magnitude, it would have a much greater effect if applied at the root, e.g., global rotation, than if applied to an element further down the kinematic tree, e.g., the last phalanx of a finger. Therefore, we propose a solution that measures the velocity and acceleration of a set of points attached to the kinematic chain. We consider the motion of vertices $\mathbf{k}$ of the kinematic chain $\mathcal{K}$ (Figure 6.4) and build an energy penalizing the velocity and acceleration of these points:

$$E_{\text{temporal}} = \omega_8 \sum_{\mathbf{k}_i \in \mathcal{K}} ||\dot{\mathbf{k}}(\boldsymbol{\theta})||_2^2 + \omega_9 \sum_{\mathbf{k}_i \in \mathcal{K}} ||\ddot{\mathbf{k}}(\boldsymbol{\theta})||_2^2. \tag{6.9}$$

Figure 6.16 illustrates how the temporal prior reduces jitter and improves the overall robustness of the tracking.

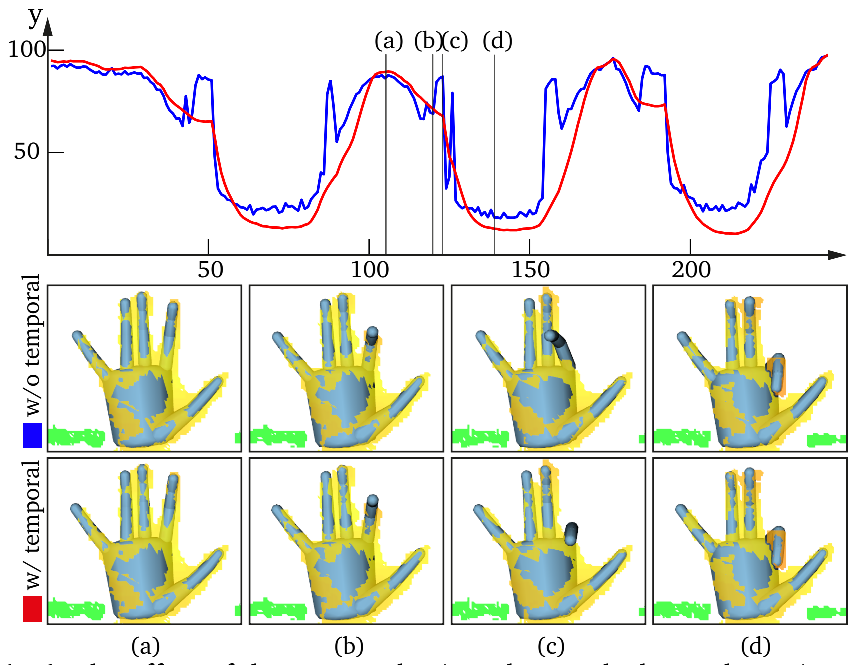


Figure 6.16: The effect of the temporal prior. The graph shows the trajectory of the y-coordinate of the fingertip over time as the index finger is bend up and down repeatedly. The temporal prior reduces jitter, but also helps avoiding tracking artifacts that arise when fragments of data pop in and out of view.

6.4 Optimization implementation

In this section we provide more details on the implementation of our optimization algorithm and describe the derivation of the necessary gradients and Jacobians.

Optimization

The optimization of the tracking energy of Equation (6.1) over the posture $\boldsymbol{\theta}$ is performed by solving the non-linear least squares problem with a Levenberg-Marquardt approach. The assumption is that a current estimate of $\boldsymbol{\theta}$ is known from which we then compute an update. More specifically, the high acquisition speed of the sensing device allows us to employ the optimized parameters from the previous time frame as the starting estimate. We then iteratively approximate the energy terms using Taylor expansion and solve a linear system to get the update $\delta\boldsymbol{\theta}$ at each iteration. As our algorithm achieves 60 fps tracking, the previously reconstructed posture is of sufficiently high quality allowing our solve to converge within seven iterations.

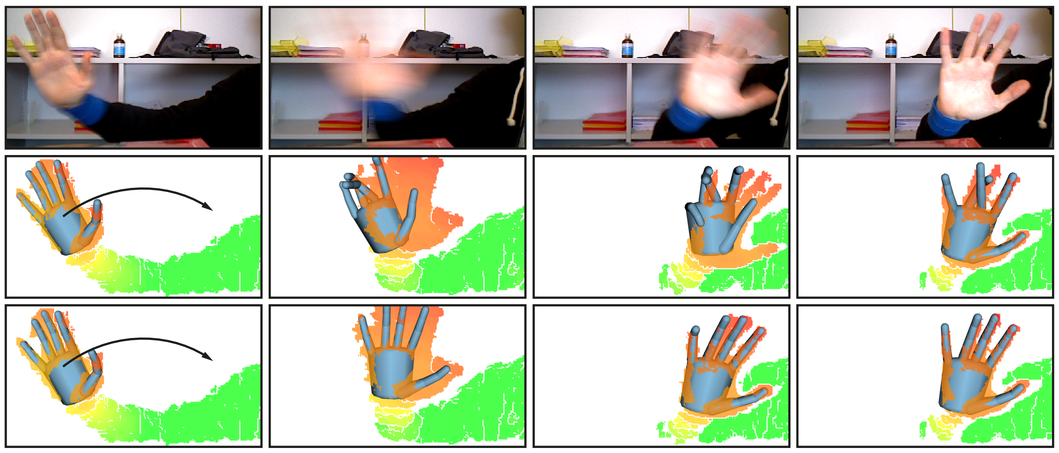


Figure 6.17: During fast motion, optimizing directly for a fully articulated hand can lead to incorrect correspondences and cause loss of tracking (middle row). By compensating for the rigid motion ahead of solving for joint angles, our system can better capture fast movements (bottom row).

Initialization

As a user enters the scene our method is initialized by the fingertip detection and fitting from (Qian et al. 2014). Other appearance-based methods could be used for initialization as well (Tompson et al. 2014). We also reinitialize the tracking in case a severe tracking failure is detected using the method of (Wei et al. 2012). Such reinitialization occurs rarely (e.g. less than 0.5% of the frames in the sequence of Figure 6.23).

Rigid bias

To improve the convergence of our solver in case of fast motion, we first perform the optimization in Equation (6.1) for the rigid motion only by optimizing for the root of the kinematic chain. As shown in Figure 6.17, optimizing first for the rigid motion prior to the full posture estimation leads to improved robustness of the tracking.

System parameters

For all our results we fix our parameters to $\omega_1 = \omega_2 = \omega_5 = 1$, $\omega_4 = 10^3$, $\omega_3 = \omega_6 = \omega_7 = 10^8$, $\omega_8 = \omega_9 = 3$. We determined these weights empirically by re-tracking multiple sequences with different sets of parameters. Our system was tested on an Intel Core i7 4GHz with NVIDIA GTX980 GPU running Ubuntu 12.02. To run on a 60Hz RGBD device such as the PrimeSense Carmine 1.09 or the Creative Gesture Camera, we perform 1 rigid iteration and 7 full iterations, at 1.5ms per iteration. We perform closed form closest point correspondences and Jacobian computation for the fitting energies on the GPU. The number of iterations can be easily adapted to run on the new Intel RealSense 3D Camera (F200) at 120Hz or at even higher frame rates on future devices.

6.4.1 Jacobian matrices

In the following, we briefly provide the Jacobians relevant for the optimization. In particular, the perspective projection Jacobian and the skeleton Jacobian.

Perspective projection Jacobian

The Jacobian of the perspective projection is a $[2 \times 3]$ matrix depending on the focal length of the camera $\mathbf{f} = [f_x, f_y]$ and the 3D position $\mathbf{x}$ at which it is evaluated (Bouaziz et al. 2014):

$$\mathbf{J}_{\text{persp}}(\mathbf{x}) = \left[\begin{pmatrix} f_x/\mathbf{x}_z & 0 & -\mathbf{x}_x f_x/\mathbf{x}_z^2 \\ 0 & f_y/\mathbf{x}_z & -\mathbf{x}_y f_y/\mathbf{x}_z^2 \end{pmatrix}\right]$$

Skeleton Jacobian

The skeleton Jacobian $\mathbf{J}_{\text{skel}}(\mathbf{x})$ is a $[3 \times 26]$ matrix. For each constraint, the bone identifier $b = id(\mathbf{x})$ associated to each 3D point $\mathbf{x}$ determines the affected portion of the kinematic chain. That is, it identifies the non-zero columns of $\mathbf{J}_{\text{skel}}(\mathbf{x})$. As discussed in (Buss 2004), the i-th column of $\mathbf{J}_{\text{skel}}(\mathbf{x})$ contains the linearization of i-th joint about $\mathbf{x}$ (see also Section 3.2.2).

6.4.2 Approximation using linearized functions

To approximate the following energies, we approximate $E = \|\mathbf{f}(\mathbf{x})\|_2^2$ by linearizing $\mathbf{f}(\mathbf{x})$ as

$$\mathbf{f}(\mathbf{x})|_{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x}.$$

The approximation is then expressed as

$$\bar{E} = \|\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x}\|_2^2. \tag{6.10}$$

Joint bounds

The joint bounds energy can be written as

$$\bar{E}_{\text{bound}} = \omega_7 \sum_{\theta_i \in \boldsymbol{\theta}} \overline{\chi}(i)(\delta\boldsymbol{\theta}_i + \boldsymbol{\theta}_i - \overline{\boldsymbol{\theta}}_i)^2 + \underline{\chi}(i)(\delta\boldsymbol{\theta}_i + \boldsymbol{\theta}_i - \underline{\boldsymbol{\theta}}_i)^2.$$

This energy penalizes kinematic parameters violating their upper or lower bounds by generating a high damping weight for them.

Temporal coherence

To compute the velocity $\dot{\mathbf{k}}(\boldsymbol{\theta})$ and the acceleration $\ddot{\mathbf{k}}(\boldsymbol{\theta})$ of a point $\mathbf{k}$ attached to the kinematic chain, we use finite differences. The linearization of the temporal energy becomes

$$\begin{aligned}\bar{E}_{\text{temporal}} = \omega_8 &\sum_{\mathbf{k} \in \mathcal{K}} \|\mathbf{J}_{\text{skel}}(\mathbf{k})\delta\boldsymbol{\theta} + (\mathbf{k} - \mathbf{k}_{t-1})\|_2^2 \\ + \omega_9 &\sum_{\mathbf{k} \in \mathcal{K}} \|\mathbf{J}_{\text{skel}}(\mathbf{k})\delta\boldsymbol{\theta} + (\mathbf{k} - 2\mathbf{k}_{t-1} + \mathbf{k}_{t-2})\|_2^2,\end{aligned}$$

where $\mathbf{k}_{t-1}$ and $\mathbf{k}_{t-2}$ are the position of such points from the two previously optimized frames.

Data-driven prior

The data-driven projection energy can be rewritten as

$$\bar{E}_{\text{posture}} = \omega_4 \left\| (\mathbf{I} - \Pi_{\mathcal{P}} \mathbf{M} \Pi_{\mathcal{P}}^T)(\delta\boldsymbol{\theta} + \boldsymbol{\theta} - \boldsymbol{\mu}) \right\|_2^2,$$

where $\mathbf{M} = \omega_4(\omega_5 \boldsymbol{\Sigma}^2 + \omega_4 \mathbf{I})^{-1}$. The distinction between this formulation and an orthogonal subspace projection is further discussed in 6.4.4.

6.4.3 Approximation using linearized ℓ_2 distance

To approximate the following energies, we first reformulate the quadratic form $E = \|\mathbf{f}(\mathbf{x})\|_2^2$ as $E = (\|\mathbf{f}(\mathbf{x})\|_2)^2$. We then linearize the ℓ_2 norm $\|\mathbf{f}(\mathbf{x})\|_2$ as

$$\|\mathbf{f}(\mathbf{x})\|_2|_{\mathbf{x}} \approx \|\mathbf{f}(\mathbf{x})\|_2 + \frac{\mathbf{f}(\mathbf{x})^T}{\|\mathbf{f}(\mathbf{x})\|_2} \mathbf{J}(\mathbf{x})\delta\mathbf{x}.$$

The approximation is then expressed as

$$\bar{E} = \left(\|\mathbf{f}(\mathbf{x})\|_2 + \frac{\mathbf{f}(\mathbf{x})^T}{\|\mathbf{f}(\mathbf{x})\|_2} \mathbf{J}(\mathbf{x}) \delta \mathbf{x} \right)^2 .$$

When the energy is of the form $E = \|\mathbf{x} - \Pi(\mathbf{x})\|_2^2$ where $\Pi(\mathbf{x})$ is a projection operator, Bouaziz et al. (2012) showed that $\mathbf{f}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T$. In this case, the approximate energy can be simplified as

$$\bar{E} = \left(\|\mathbf{f}(\mathbf{x})\|_2 + \frac{\mathbf{f}(\mathbf{x})^T}{\|\mathbf{f}(\mathbf{x})\|_2} \delta \mathbf{x} \right)^2 .$$

Contrary to the approximation in Equation (6.10), the Jacobian of the projection function does not need to be known. This formulation is useful as the approximation in the equation above only needs to evaluate the projection function and therefore allows to use arbitrarily complex projection functions.

Point cloud alignment

We linearize the point cloud alignment energy as

$$\bar{E}_{\text{3D}} = \omega_1 \sum_{\mathbf{t} \in \mathcal{X}_s} \omega_{\text{re}} (\mathbf{n}^T (\mathbf{J}_{\text{skel}}(\mathbf{y}) \delta \boldsymbol{\theta} + d))^2,$$

where $\mathbf{y} = \Pi_{\mathcal{H}}(\mathbf{t}, \boldsymbol{\theta})$ is the closest point from $\mathbf{t}$ on the hand model $\mathcal{H}$ with hand posture $\boldsymbol{\theta}$. $\mathbf{n}$ is the surface normal at $\mathbf{y}$, and $d = (\mathbf{y} - \mathbf{t})$. As we minimize the ℓ_2 norm we use a weight $\omega_{\text{re}} = 1/\|d\|_2$ in an iteratively re-weighted least squares fashion (Bouaziz, Tagliasacchi and Pauly 2013).

Silhouette alignment

The silhouette energy is expressed in screen space, and therefore employs the perspective projection Jacobian $\mathbf{J}_{\text{persp}}(\mathbf{x})$, where $\mathbf{x}$ is the 3D location of a rendered silhouette point $\mathbf{p}$. Similarly to the point cloud alignment the linearization can be expressed as

$$\bar{E}_{\text{silh.}} = \omega_2 \sum_{\mathbf{p} \in \mathcal{S}_r} (\mathbf{n}^T (\mathbf{J}_{\text{persp}}(\mathbf{x}) \mathbf{J}_{\text{skel}}(\mathbf{x}) \delta \boldsymbol{\theta} + d))^2,$$

where $d = (\mathbf{p} - \mathbf{q})$ with $\mathbf{q} = \Pi_{\mathcal{S}_s}(\mathbf{p}, \boldsymbol{\theta})$, and $\mathbf{n}$ is the 2D normal at the sensor silhouette location $\mathbf{q}$.

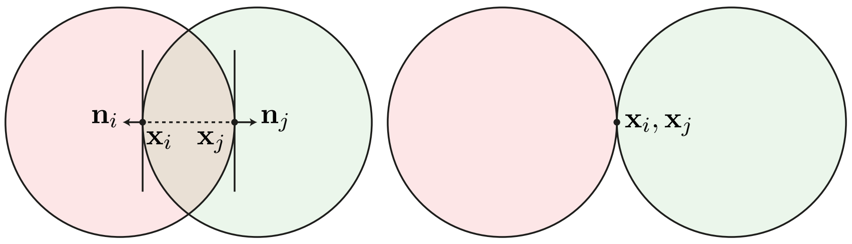


Figure 6.18: (left) Collision constraints definition, deepest penetration points marked as $\mathbf{x}_i, \mathbf{x}_j$. (right) When the collision energy is minimized in isolation the penetration points are co-located.

Collision

Figure 6.18 illustrates the necessary notation with a 2D example, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the end-points of the shortest segment between the two cylinders axes. The linearized energy is defined as

$$\bar{E}_{\text{coll.}} = \omega_6 \sum_{\{i,j\}} \chi(i,j) \left(\mathbf{n}_i^T \left((\mathbf{J}_{\text{skel}}(\mathbf{x}_i) - \mathbf{J}_{\text{skel}}(\mathbf{x}_j))\delta\boldsymbol{\theta} + d \right) \right)^2$$

where $\mathbf{n}_i$ is the surface normal at $\mathbf{x}_i$ (as shown in Figure 6.18), and $d = (\mathbf{x}_i - \mathbf{x}_j)$.

6.4.4 Projective vs. subspace PCA

In Equation (6.6), minimizing E_{posture} over $\tilde{\boldsymbol{\theta}}$ has a closed form solution:

$$\tilde{\boldsymbol{\theta}} = (\omega_5 \boldsymbol{\Sigma}^2 + \omega_4 \mathbf{I})^{-1} (\omega_4 \Pi_{\mathcal{P}}^T (\boldsymbol{\theta} - \boldsymbol{\mu})).$$

We can therefore rewrite our data-driven energy only as a function of $\boldsymbol{\theta}$ as

$$E_{\text{posture}} = \omega_4 \| (\boldsymbol{\theta} - \boldsymbol{\mu}) - \Pi_{\mathcal{P}} \mathbf{M} \Pi_{\mathcal{P}}^T (\boldsymbol{\theta} - \boldsymbol{\mu}) \|_2^2,$$

where $\mathbf{M} = \omega_4 (\omega_5 \boldsymbol{\Sigma}^2 + \omega_4 \mathbf{I})^{-1}$. Our formulation does not only allow the solution to stay close to the posture space, but also penalizes unlikely postures replacing the conventional orthogonal projection matrix $\Pi_{\mathcal{P}} \Pi_{\mathcal{P}}^T$ by a matrix $\Pi_{\mathcal{P}} \mathbf{M} \Pi_{\mathcal{P}}^T$ taking into account the PCA standard deviation. Note that when $\omega_5 = 0$ we retrieve the orthogonal projection matrix $\Pi_{\mathcal{P}} \Pi_{\mathcal{P}}^T$.

6.4.5 Non-linear least squares optimization

To solve our optimization problem we use a Levenberg-Marquardt approach. We iteratively solve Equation (6.1) using the approximate energies described in Section 6.4.1 through Section 6.4.3 leading to a damped least squares minimization

$$\min_{\delta\boldsymbol{\theta}} \bar{E}_{\text{3D}} + \bar{E}_{\text{silh.}} + \bar{E}_{\text{posture}} + \bar{E}_{\text{kin.}} + \bar{E}_{\text{temporal}} + \bar{E}_{\text{damping}},$$

and update our hand posture using the update $\boldsymbol{\theta} = \boldsymbol{\theta} + \delta\boldsymbol{\theta}$. Note that since our energies are written in the form:

$$\Sigma_i \bar{E}_i = \Sigma_i \|\mathbf{J}_i \delta\boldsymbol{\theta} - \mathbf{e}_i\|_2^2,$$

our solve can be re-written as

$$\delta\boldsymbol{\theta} = \left(\Sigma_i \mathbf{J}_i^T \mathbf{J}_i\right)^{-1} \left(\Sigma_i \mathbf{J}_i^T \mathbf{e}_i\right) = 0.$$

To stabilize the optimization, we introduce a damping energy $\bar{E}_{\text{damping}} = \lambda \|\delta\boldsymbol{\theta}\|_2^2$, where $\lambda = 100$.

6.4.6 CPU/GPU optimization

Our technique elegantly de-couples the components of our optimization on CPU and GPU. With regards to Figure 6.2 only large-scale and trivially parallelizable tasks, like the computation of constraints associated with 2D/3D ICP correspondences are performed on GPU, while all others run efficiently on a *single* CPU thread. In particular, the inversion in Equation (6.4.5) is performed on CPU by Cholesky factorization (Eigen3). As the final solve is performed on CPU, we designed our optimization to minimize memory transfers between CPU/GPU. First of all, note that although at each iteration we need to render an image of the cylinder model, the texture is already located on the GPU buffers. Furthermore, although the large ($\approx 20k \times 26$) Jacobian matrices associated with E_{3D} and E_{2D} are assembled on the GPU, a CuBLAS kernel is used to compute the much smaller (26×26, 26×1) matrices $\mathbf{J}_i^T \mathbf{J}_i$ and $\mathbf{J}_i^T \mathbf{e}_i$. Only these need to be transferred back to CPU for each solver iteration.

6.5 Results

In the following, we analyze the performance of our real-time tracking system by providing a comparison to several state-of-the art solutions.

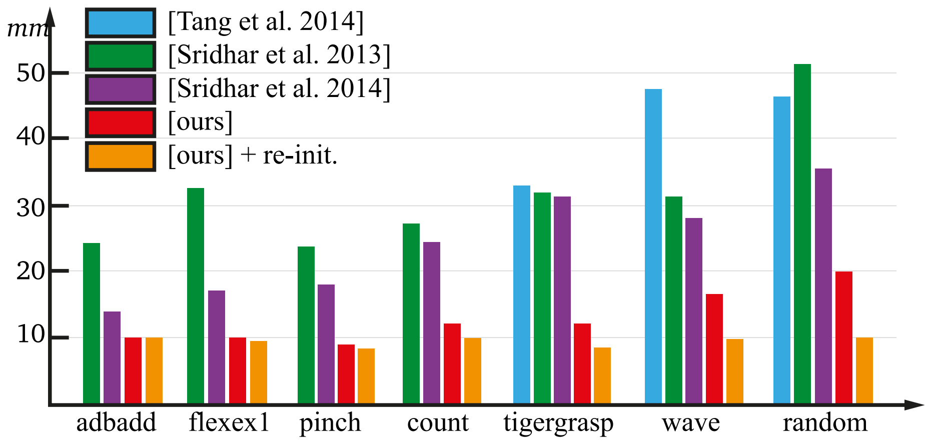


Figure 6.19: We quantitatively evaluate our algorithm on the **Dexter-1** dataset from (Sridhar et al. 2013). The measurements report the root mean square errors of fingertip placements. The acquisition setup consists of several calibrated video cameras and a single depth camera. For our results and the method of Tang et al. (2014), only the depth image is used for tracking, while the algorithms of Sridhar and colleagues also use the video streams. The blue, green, and purple bars are reproduced from (Sridhar et al. 2014). For our algorithm we report results without (red) and with (orange) reinitialization.

Dexter-1 dataset (Sridhar et al. 2013)

Figure 6.19 shows a quantitative comparison with several existing methods on a publicly available data set acquired at 25 Hz. As the graph illustrates, our solution clearly outperforms the method of (Tang et al. 2014) that uses regression forest classifiers in an appearance-based approach to estimate hand postures. We also significantly improve upon the gradient-based optimization methods of (Sridhar et al. 2013, 2014) that, in addition to the depth information, use RGB data from five additional video cameras. As the dataset is acquired at 25 Hz, the performance of our algorithm (red) is suboptimal. In particular, in a single frame fingers are occasionally displaced by 2 to 3 times their radii, thus corrupting ICP correspondences. By reinitializing with finger detection as in (Qian et al. 2014) our performance considerably improves, as shown in the figure.

Subspace ICP (Schröder et al. 2014)

Figure 6.20 shows a comparison to our model-based approach from Chapter 5. As the figure illustrates, our method outperforms this previous work. A key difference is that the previous method optimized directly in a PCA subspace, which tends to over-constrain the solution, while the method we present here introduces a PCA data term as a regularizer, which preserves the full expressiveness of the tracking model. In addition, we introduce collision handling, apply robust norms for auto-

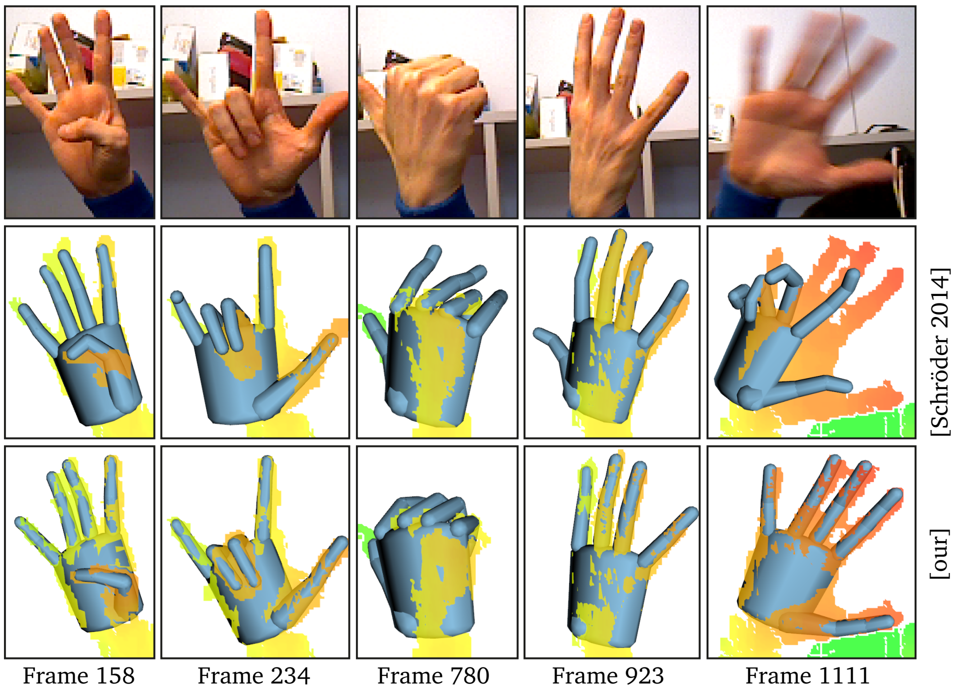


Figure 6.20: A few comparison frames illustrating the difference in performance of our method compared to (Schröder et al. 2014). From left to right we can observe problems related to: correspondences to the back of the model, lack of silhouette energy (3 times) and loss of tracking due to fast motion.

matic outlier detection, and employ a more advanced correspondence search that handles self-occlusions. In combination, these factors lead to substantial improvements in tracking accuracy and robustness without compromising computational efficiency.

Convex body solver (Melax et al. 2013)

We compare to this algorithm by employing the precompiled binaries from the Intel Perceptual Computing SDK. We modifed the demo application to save the recorded depth/color frames to disk while tracking. We then re-tracked this data from scratch using our technique. As illustrated in Figure 6.21, our method offers a substantial increase in tracking robustness compared to (Melax et al. 2013). This can be attributed to any of the improvements we presented, but it is difficult to quantitatively identify the causes, because no control on tracking parameters nor source code is given. Their approach computes closest correspondences to the entire model, therefore not explicitly handling occlusion. The authors also proposed a technique to ensure that the model is fully contained in the 3D convex hull of the data. Note that in camera space, this amounts to constraints similar to the ones enforced by our 2D registration (Equation (6.3)), except that the distance transform would be computed from the 2D convex hull of the silhouette image. Figure 6.21 (Frame 448) illustrates

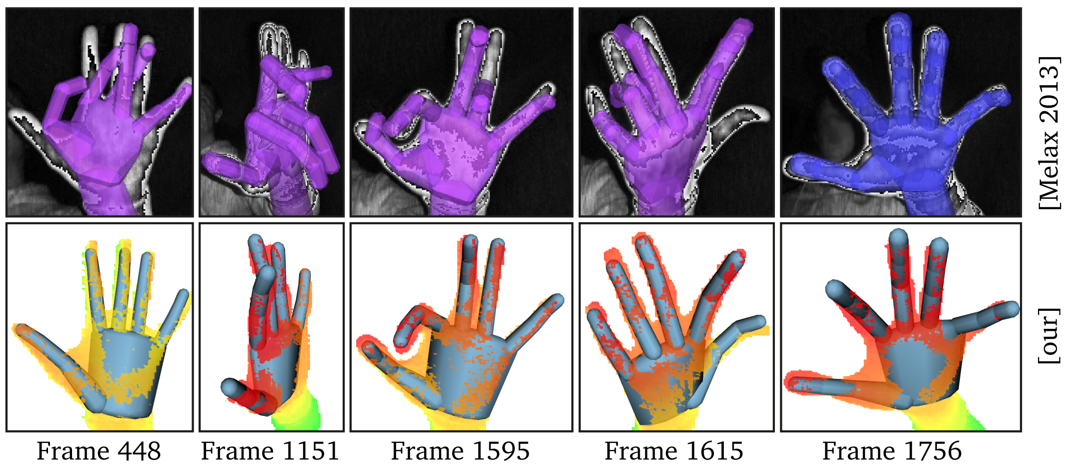


Figure 6.21: Comparison to the method of Melax et al. (2013). The full sequence can be seen in the accompanying video. We highlight a few frames that are not resolved correctly by this method, but that can be handled successfully with our solution. The last frame shows the better geometric approximation quality of the convex body model used in (Melax et al. 2013) compared to our simpler cylinder model.

how our 2D registration better constrains feasible solutions. While in (Melax et al. 2013) correlation between fingers is manually introduced as a *grasping bias*, our optimization is data driven and encodes correlation in a more principled way. As illustrated in Figure 6.21, this approach often loses tracking during complex motion. However, it is sometimes capable of recovering by sampling and then evaluating a reduced set of postures, with an approach that is similar in spirit to (Oikonomidis et al. 2011a). One advantage of their method is the higher geometric fidelity of their convex bodies hand model compared to our cylinder model. Furthermore, our evaluation demonstrated how their more precise representation of the hand's Thenar eminence, as well as the thumb articulation, can result in more natural fitting in these regions.

Convolutional networks (Tompson et al. 2014)

Figure 6.23 shows a quantitative comparison with the appearance-based method of (Tompson et al. 2014) on a dataset provided by the authors of that paper. Overall, the tracking quality is comparable, with a somewhat lower average error for our method. However, our solution avoids many of the high-error peaks of (Tompson et al. 2014) where tracking is lost completely. An additional advantage of our approach in comparison to any of the existing appearance-based methods is that we can handle more complex interactions of two hands, since such configurations are not part of the training data sets of existing methods; see Figure 6.22.

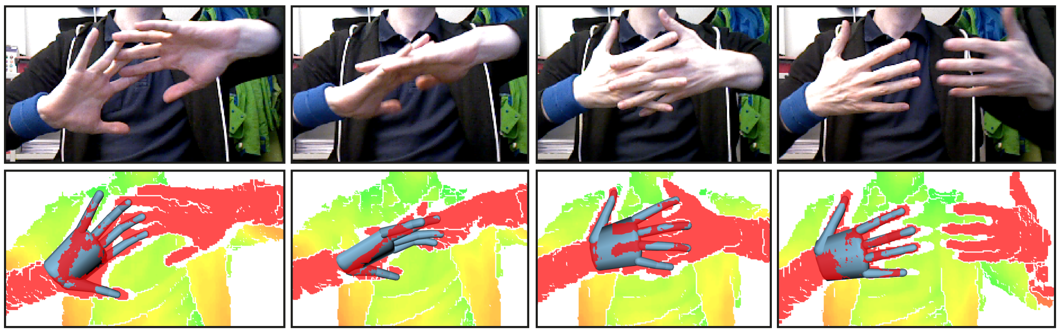


Figure 6.22: Developing robust model-based tracking is essential to enable tracking of hands interacting with each other or with other objects in the environment. Here we illustrate that for our method tracking accuracy is not significantly affected even though we are not modeling the second hand. Note that such motion cannot be tracked successfully by appearance-based methods such as (Tompson et al. 2014).

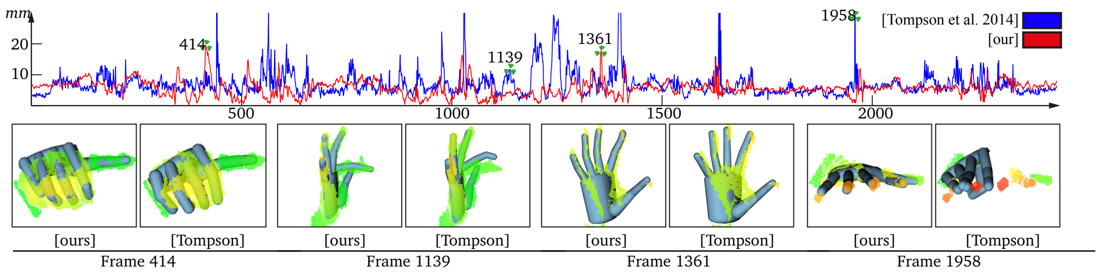


Figure 6.23: Quantitative comparison to (Tompson et al. 2014). The graph shows the average root mean square tracking error w.r.t. ground truth across 2440 frames. Some frames where the accuracy of the two methods differs significantly are highlighted in the bottom row.

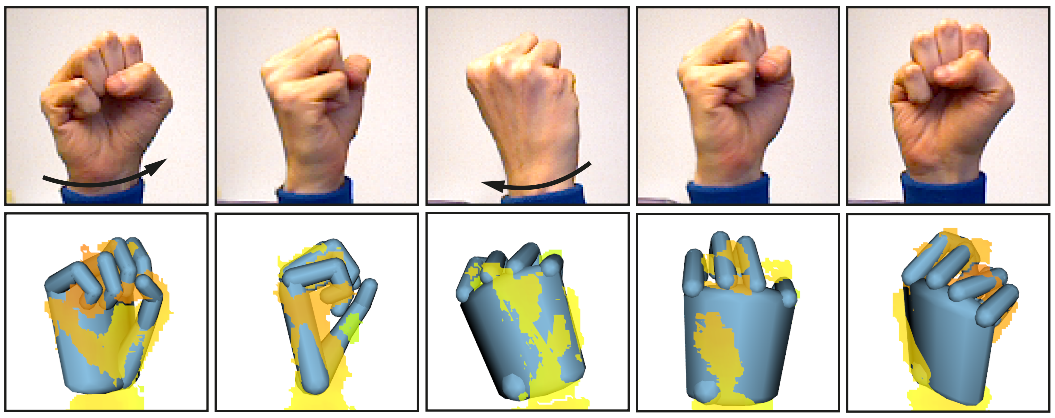


Figure 6.24: Our algorithm relies on the presence of salient geometric features in the depth map. Challenging sequences like a rotating fist lack such features when acquired with current commodity depth sensors, which can result in loss of tracking.

Limitations

Single-camera depth acquisition yields incomplete data and as such the posture reconstruction problem is inherently ill-posed. Tracking errors can occur in certain situations as explained above when insufficient data is acquired due to occlusions or fast motion. Similarly, the resolution of the sensor limits tracking accuracy. As shown in Figure 6.24, when geometric features become indiscriminate, our registration approach fails. Integrating color and shading information could potentially address this issue (de La Gorce et al. 2011). While our current system requires the user to wear a wristband for detection and stabilization, this could be replaced by automatic hand labeling, e.g. using forest classifiers as in (Tompson et al. 2014).

Our cylinder model proved adequate for the data quality of current commodity sensors, but is overall limited in geometric accuracy, and hence might not scale with increasing sensor resolution. Also, in our current implementation the model needs to be manually adapted to the user through simple scaling operations. Without such adaptation, tracking accuracy degrades as shown in Figure 6.25. This user-specific adaption could be automated (Taylor et al. 2014) and potentially even performed simultaneously with the real-time tracking as recently proposed for face tracking (Bouaziz, Wang and Pauly 2013).

The PCA model used in the prior energy is an efficient, but rather simplistic representation of the posture space. We currently do not consider the temporal order in which the hand postures of the database have been acquired, which could potentially be exploited for more sophisticated temporal priors.

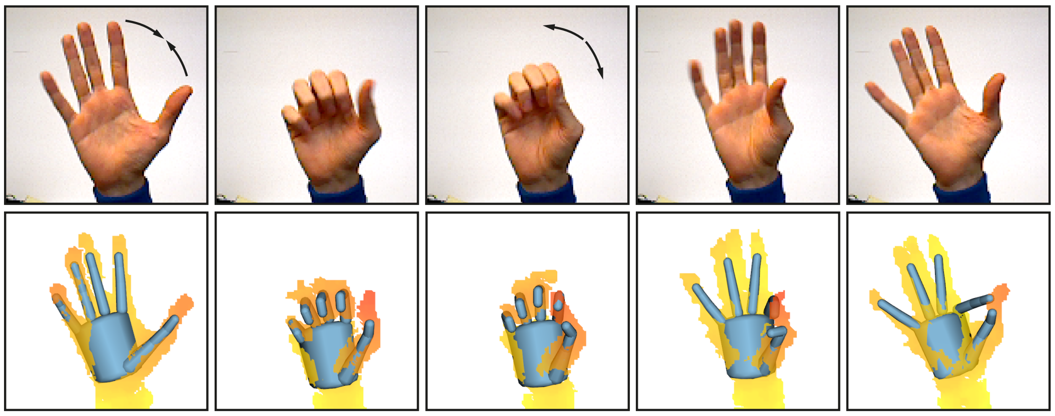


Figure 6.25: When tracking with an uncalibrated model, tracking correspondences can map to data belonging to erroneous portions of the model. In the figure, the index finger remains attached to samples associated with the thumb.

6.6 Discussion

We have introduced a new model-based approach to real-time hand tracking using a single low-cost depth camera. This simple acquisition setup maximizes ease of deployment, but poses significant challenges for robust tracking. We have demonstrated that accurate real-time hand tracking can be performed at 60FPS on a low-cost RGBD device using a model-based approach. Robust tracking is achieved by combining 2D and 3D information extracted from the sensor data with a set of carefully designed priors. We compared our method with multiple state of the art approaches and showed that our system outperforms current model- and appearance-based approaches.

Our analysis revealed that a major source of error when tracking articulated hands are erroneous correspondences between the hand model and the acquired data, mainly caused by outliers, holes, or data popping in and out during acquisition. We demonstrate that these problems can be resolved by our new formulation of correspondence search. In combination with suitable 2D/3D registration energies and data-driven priors, this leads to a robust and efficient hand tracking algorithm that outperforms existing model- and appearance-based solutions.

In our experiments we show that our system runs seamlessly for sensors capturing data at 60 Hz. However, we can even support higher frame rates of up to 120 fps in anticipation of future sensors that have recently been announced. By fully disclosing our source code and data we ensure that our method and results are reproducible, as well as facilitating future research and product development.

Examples of such future efforts are automatic personalization of the tracking model to the acquired user, robust two-hand tracking with object interactions, combinations of hand tracking with full body tracking algorithms, and integrating our hand tracking solution to new interfaces and real-time applications.

Chapter 7

Conclusion

In this thesis several different approaches for hand tracking were explored, including marker-based optical motion capture, data-driven discriminative visual tracking and generative tracking based on articulated registration. Furthermore, the dimensionality of hand articulations was analyzed and models for low-dimensional hand posture representations were derived, adapted and integrated into hand posture estimation approaches. Real-time hand tracking systems using RGBD sensors were developed and were shown to achieve high quality results.

We designed a method for automatically generating functional reduced marker layouts for marker-based optical motion capture. Such layouts containing a low number of markers simplify the hand motion capture and reconstruction process. Based on a subspace-constrained inverse kinematics (IK) posture reconstruction approach, we defined specific criteria for reduced marker layouts that combine the numerical stability of the IK problem and the practical geometric feasibility of the layout. Optimizing for these criteria produces reduced marker layouts that are well-suited for usage in optical motion capture of hands. We developed a specialized surface-constrained particle swarm optimization (PSO) process to generate such marker layouts. Our approach established a qualitative relationship between subspace representations of hand articulations and marker placement, and showed that plausible hand posture reconstructions can be achieved from sparse marker input.

Based on previous work on discriminative hand tracking with a color glove, we built a real-time hand tracking system and used it for interactive teleoperation of an anthropomorphic robot hand. We modified the method by employing a simplified, efficient nearest neighbor search algorithm for color-labeled images based on cascaded image matching using chamfering to efficiently approximate color class distance transforms. Using an RGBD sensor allowed us to extend the discriminative approach by a generative pose estimation step that aligns a textured hand model with the color-classified input point cloud in a color-sensitive iterative closest point (ICP) process.

Analyzing motion captured human hand motions using principal component analysis (PCA) facilitated the construction of low-dimensional representations of hand articulations. We integrated this subspace concept into an inverse kinematics hand posture estimation process and developed a tracking method that combined this inverse kinematics fitting with ICP in an articulated registration approach. To overcome the limitations associated with dimension reduction, we proposed an online adaptive PCA model that complements an initial hand posture subspace with a continuously updated local linear model. This approach continuously captures new incoming data in an adaptive subspace model by performing efficient incremental covariance computations based on rank-one updates. Integrating this adaptive scheme in the inverse kinematics hand tracking system accomplished robust and flexible real-time posture estimation by combining subspace-anchoring with continuous refinements.

We formulated model-based hand tracking as a regularized articulated registration process, in which a template hand model is aligned with point cloud input data obtained from an RGBD sensor. The developed method combines geometric fitting with statistical, kinematic and temporal priors in a unified registration framework. We devised a combined 2D/3D registration and correspondence computation approach that explicitly takes visibility and occlusions into account. Our optimized GPU implementation allows for efficient processing of the entire input point cloud and the silhouette pixels of the rendered hand model during the correspondence computations. Since the method represents the model fitting and regularization priors in a unified way, the posture estimation itself is efficiently done in single linear solve. We demonstrated that highly robust and accurate real-time hand tracking is achievable using a gradient-based ICP approach, when designed with explicit occlusion handling and combined with carefully designed regularizers, overcoming the propensity of gradient-based methods to fall into local minima.

Various avenues of future work based on the contributions of this thesis are conceivable. Our model-based tracking approaches show promise for being extended towards robust two-hand tracking with object interactions, as well as combination with and extension towards real-time full body tracking. Our analysis of subspace representations of hand articulations based on PCA simplifies the hand tracking problem and increases realism as well as robustness. Beyond that, more sophisticated hand articulation priors, particularly data-driven temporal motion priors, could further improve the hand posture estimation. The accuracy achieved by our combination of data-driven discriminative estimation with an additional generative refinement step indicates the potential of hybrid approaches that combine the strengths of discriminative per-frame initialization and generative continuous refinement. Other works on hand tracking also often utilize the concept of method hybridization to maximize robustness and accuracy. An important contribution of our work is that gradient-based registration methods can achieve outstanding performance for real-time tracking. As the various different hand tracking approaches get more refined and hardware capabilities increase, the combination of gradient-based generative registration, discriminative detection and reinitialization, as well as stochastic perturbation has the potential to advance technological development towards seamless and pervasive integration of hand tracking and gestural interaction technologies in every day life.

Bibliography

Agur, A. M. R. and Lee, M. J. (1999). *Grant's Atlas of Anatomy*, tenth edn, Lippincott Williams and Wilkins.

Albrecht, I., Haber, J. and Seidel, H.-P. (2003). Construction and animation of anatomically based human hand models, *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 98–109.

Aristidou, A. and Lasenby, J. (2010). Motion capture with constrained inverse kinematics for real-time hand tracking, *Proc. International Symposium on Communications, Control and Signal Processing*, pp. 1–5.

ASUS Xtion PRO (2015).
URL: *http://www.asus.com/Multimedia/Xtion_PRO/*

Ballan, L., Taneja, A., Gall, J., Van Gool, L. and Pollefeys, M. (2012). Motion capture of hands in action using discriminative salient points, *Proc. IEEE European Conference on Computer Vision*, pp. 640–653.

Barrow, H. G., Tenenbaum, J. M., Bolles, R. C. and Wolf, H. C. (1977). Parametric correspondence and chamfer matching: two new techniques for image matching, *Proc. 5th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 659–663.

Bernstein, N. (1967). *The Co-ordination and Regulation of Movements*, Pergamon Press Ltd.

Bicchi, A., M, M. G. and Santello, M. (2011). Modelling natural and artificial hands with synergies, *Philosophical Transactions of the Royal Society B: Biological Sciences* **366**(1581): 3153–3161.

Biosyn (2015).
URL: *http://www.biosynsystems.net/*

Bouaziz, S., Deuss, M., Schwartzburg, Y., Weise, T. and Pauly, M. (2012). Shape-up: Shaping discrete geometry with projections, *Computer Graphics Forum (Proc. Symposium on Geometry Processing)* **31**(5): 1657–1667.

Bouaziz, S., Tagliasacchi, A. and Pauly, M. (2013). Sparse iterative closest point, *Computer Graphics Forum (Proc. Symposium on Geometry Processing)* **32**(5): 113–123.

Bouaziz, S., Tagliasacchi, A. and Pauly, M. (2014). Dynamic 2D/3D registration, *in* N. Holzschuch and K. Myszkowski (eds), *Eurographics (Tutorials)*.

Bouaziz, S., Wang, Y. and Pauly, M. (2013). Online modeling for realtime facial animation, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **32**(4): 40:1–40:10.

Brand, M. (2006). Fast low-rank modifications of the thin singular value decomposition, *Linear Algebra and its Applications* **415**(1): 20–30. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

Bray, M., Koller-Meier, E. and Gool, L. V. (2007). Smart particle filtering for high-dimensional tracking, *Computer Vision and Image Understanding* **106**(1): 116–129.

Bunch, J., Nielsen, C. and Sorensen, D. (1978). Rank-one modification of the symmetric eigenproblem, *Numerische Mathematik* **31**(1): 31–48.

Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods, *Technical report*, University of California.

Buss, S. R. and Kim, J.-S. (2004). Selectively damped least squares for inverse kinematics, *Journal of Graphics Tools* **10**(3): 37–49.

Chai, J. and Hodgins, J. K. (2005). Performance animation from low-dimensional control signals, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **24**(3): 686–696.

Chan, T. F. and Dubey, R. V. (1995). A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators, *IEEE Transactions on Robotics and Automation* **11**(2): 286–292.

Chang, L., Pollard, N., Mitchell, T. and Xing, E. (2007). Feature selection for grasp recognition from optical markers, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2944–2950.

Creative Senz3D (2015).
URL: *http://us.creative.com/p/web-cameras/creative-senz3d*

Cutkosky, M. (1989). On grasp choice, grasp models, and the design of hands for manufacturing tasks, *IEEE Transactions on Robotics and Automation* **5**(3): 269–279.

CyberGlove (2015).
URL: *http://www.cyberglovesystems.com/*

Daffertshofer, A., Lamoth, C. J., Meijer, O. G. and Beek, P. J. (2004). PCA in studying coordination and variability: A tutorial, *Clinical Biomechanics* **19**(4): 415–428.

de La Gorce, M., Fleet, D. and Paragios, N. (2011). Model-based 3D hand pose estimation from monocular video, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(9): 1793–1805.

de La Gorce, M., Paragios, N. and Fleet, D. J. (2008). Model-based hand tracking with texture, shading and self-occlusions, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Do, M., Romero, J., Kjellström, H., Azad, P., Asfour, T., Kragic, D. and Dillmann, R. (2009). Grasp recognition and mapping on humanoid robots, *Proc. IEEE/RAS International Conference on Humanoid Robots*, pp. 465–471.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D. and Twombly, X. (2007). Vision based hand pose estimation: A review, *Computer Vision Image Understanding* **108**(1-2): 52–73.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance transforms of sampled functions, *Theory of Computing* **8**(1): 415–428.

Fischer, M., van der Smagt, P. and Hirzinger, G. (1998). Learning techniques in a dataglove based telemanipulation system for the DLR hand, *Proc. IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1603–1608.

Gabiccini, M., Bicchi, A., Prattichizzo, D. and Malvezzi, M. (2011). On the role of hand synergies in the optimal choice of grasping forces, *Autonomous Robots* **31**(2-3): 235–252.

Ganapathi, V., Plagemann, C., Koller, D. and Thrun, S. (2012). Real-time human pose tracking from range data, *Proc. IEEE European Conference on Computer Vision*, pp. 738–751.

Gourret, J.-P., Thalmann, N. M. and Thalmann, D. (1989). Simulation of object and human skin formations in a grasping task, *Proc. Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pp. 21–30.

Griffin, W. B., Findley, R. P., Turner, M. L. and Cutkosky, M. R. (2000). Calibration and mapping of a human hand for dexterous telemanipulation, *Proc. ASME*

International Mechanical Engineering Congress and Exposition, Dynamics Systems and Controls, Vol. 69, pp. 1145–1152.

Grochow, K., Martin, S. L., Hertzmann, A. and Popović, Z. (2004). Style-based inverse kinematics, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **23**(3): 522–531.

Guerra-filho, G. B. (2005). Optical motion capture: Theory and implementation, *Journal of Theoretical and Applied Informatics* **12**(2): 61–90.

Hamer, H., Schindler, K., Koller-Meier, E. and Gool, L. V. (2009). Tracking a hand manipulating an object, *Proc. IEEE International Conference on Computer Vision*, pp. 1475–1482.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions, *Journal of the Optical Society of America A* **4**(4): 629–642.

Hoyet, L., Ryall, K., McDonnell, R. and O'Sullivan, C. (2012). Sleight of hand: Perception of finger motion from reduced marker sets, *Proc. Symposium on Interactive 3D Graphics and Games*, pp. 79–86.

Huttenlocher, D., Klanderman, G. and Rucklidge, W. (1993). Comparing images using the hausdorff distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(9): 850–863.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. and Fitzgibbon, A. (2011). KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera, *Proc. ACM Symposium on User Interface Software and Technology*, pp. 559–568.

Jacka, D., Reid, A. and Merry, B. (2007). A comparison of linear skinning techniques for character animation, *Proc. Afrigraph*, pp. 177–186.

Kang, C., Wheatland, N., Neff, M. and Zordan, V. B. (2012). Automatic hand-over animation for free-hand motions from low resolution input., *in* M. Kallmann and K. E. Bekris (eds), *Motion in Games*, Vol. 7660 of *Lecture Notes in Computer Science*, pp. 244–253.

Kato, M., Chen, Y.-W. and Xu, G. (2006). Articulated hand motion tracking using ICA-based motion analysis and particle filtering, *Journal of Multimedia* **1**(3): 52–60.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization, *Proc. IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942–1948.

Kennedy, J. and Eberhart, R. C. (2001). *Swarm Intelligence*, Morgan Kaufmann Publishers Inc.

Keskin, C., Kıraç, F., Kara, Y. E. and Akarun, L. (2012). Hand pose estimation and hand shape classification using multi-layered randomized decision forests, *Proc. IEEE European Conference on Computer Vision*, pp. 852–863.

Kitagawa, M. and Windsor, B. (2008). *MoCap for Artists: Workflow and Techniques for Motion Capture*, Focal Press.

Krupka, E., Vinnikov, A., Klein, B., Hillel, A. B., Freedman, D. and Stachniak, S. (2014). Discriminative ferns ensemble for hand pose recognition, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3670–3677.

Le, B. H., Zhu, M. and Deng, Z. (2013). Marker optimization for facial motion acquisition and deformation., *IEEE Transactions on Visualization and Computer Graphics* **19**(11): 1859–1871.

Leap Motion (2015).
URL: *http://www.leapmotion.com/*

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

Lee, Y.-H. and Tsai, C.-Y. (2009). Taiwan sign language (TSL) recognition based on 3D data and neural networks, *Expert Systems with Applications* **36**(2): 1123–1128.

Li, H., Adams, B., Guibas, L. J. and Pauly, M. (2009). Robust single-view geometry and motion reconstruction, *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* **28**(5): 175:1–175:10.

Li, H., Sumner, R. W. and Pauly, M. (2008). Global correspondence optimization for non-rigid registration of depth scans, *Computer Graphics Forum (Proc. Symposium on Geometry Processing)*, pp. 1421–1430.

Li, H., Yu, J., Ye, Y. and Bregler, C. (2013). Realtime facial animation with on-the-fly correctives, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **32**(4): 42:1–42:10.

Lin, J., Wu, Y. and Huang, T. S. (2000). Modeling the constraints of human hand motion, *Proc. Workshop on Human Motion*, pp. 121–126.

Liu, G., Zhang, J., Wang, W. and McMillan, L. (2006). Human motion estimation from a reduced marker set, *Proc. Symposium on Interactive 3D Graphics and Games*, pp. 35–42.

Loper, M. M., Mahmood, N. and Black, M. J. (2014). MoSh: Motion and shape capture from sparse markers, *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* **33**(6): 220:1–220:13.

Lozano-Perez, T., Gallagher, G., Kaelbling, L. P. and Tedrake, R. (2010). Kinect hand detection, http://www.csail.mit.edu/videoarchive/research/hci/kinect-detection. Accessed: June 2015.

Lütkebohle, I., Peltason, J., Schillingmann, L., Elbrechter, C., Wrede, B., Wachsmuth, S. and Haschke, R. (2009). The Curious Robot – Structuring Interactive Robot Learning, *Proc. IEEE International Conference on Robotics and Automation*, pp. 4156–4162.

Magnenat-Thalmann, N., Laperrière, R. and Thalmann, D. (1988). Joint-dependent local deformations for hand animation and object grasping, *Proceedings on Graphics Interface*, pp. 26–33.

Maycock, J., Dornbusch, D., Elbrechter, C., Haschke, R., Schack, T. and Ritter, H. (2010). Approaching manual intelligence, *Künstliche Intelligenz – Issue Cognition for Technical Systems* **24**(4): 287–294.

Maycock, J., Steffen, J., Haschke, R. and Ritter, H. (2011). Robust tracking of human hand postures for robot teaching, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2947–2952.

Melax, S., Keselman, L. and Orsten, S. (2013). Dynamics based 3D skeletal hand tracking, *Proc. Graphics Interface 2013*, pp. 63–70.

Meyer, J., Kuderer, M., Müller, J. and Burgard, W. (2014). Online marker labeling for fully automatic skeleton tracking in optical motion capture, *Proc. IEEE International Conference on Robotics and Automation*, pp. 5652–5657.

Microsoft Kinect (2015).
URL: *http://www.microsoft.com/en-us/kinectforwindows/*

Mulatto, S., Formaglio, A., Malvezzi, M. and Prattichizzo, D. (2013). Using postural synergies to animate a low-dimensional hand avatar in haptic simulation, *IEEE Transactions on Haptics* **6**(5): 106–116.

Napier, J. (1980). *Hands*, Princeton University Press.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S. and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking, *IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136.

Nimble VR (2015).
URL: *http://nimblevr.com/*

Oikonomidis, I., Kyriazis, N. and Argyros, A. A. (2011a). Efficient model-based 3D tracking of hand articulation using Kinect, *British Machine Vision Conference*, pp. 101.1–101.11.

Oikonomidis, I., Kyriazis, N. and Argyros, A. A. (2011b). Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints, *Proc. IEEE International Conference on Computer Vision*, pp. 2088–2095.

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

Oikonomidis, I., Lourakis, M. I. and Argyros, A. A. (2014). Evolutionary quasi-random search for hand articulations tracking, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3422–3429.

OptiTrack (2015).
URL: *http://www.optitrack.com/*

Organic Motion (2015).
URL: *http://www.organicmotion.com/*

PhaseSpace (2015).
URL: *http://www.phasespace.com/*

Qian, C., Sun, X., Wei, Y., Tang, X. and Sun, J. (2014). Realtime and robust hand tracking from depth, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1106–1113.

Qualisys (2015).
URL: *http://www.qualisys.com/*

Rhee, T., Neumann, U. and Lewis, J. P. (2006). Human hand modeling from surface anatomy, *Proc. Symposium on Interactive 3D Graphics and Games*, pp. 27–34.

Rijpkema, H. and Girard, M. (1991). Computer animation of knowledge-based human grasping, *Proc. Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pp. 339–348.

Romero, J., Kjellström, H., Ek, C. H. and Kragic, D. (2013). Non-parametric hand pose estimation with object context, *Image and Vision Computing* **31**(8): 555–564.

Romero, J., Kjellström, H. and Kragic, D. (2010). Hands in action: Real-time 3D reconstruction of hands in interaction with objects, *Proc. IEEE International Conference on Robotics and Automation*, pp. 458–463.

Roweis, S. (1998). EM algorithms for PCA and SPCA, *Proc. Conference on Neural Information Processing Systems*, pp. 626–632.

Roweis, S., Saul, L. K. and Hinton, G. E. (2002). Global coordination of local linear models, *Proc. Conference on Neural Information Processing Systems*, pp. 889–896.

Safonova, A., Hodgins, J. K. and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces, *ACM Transactions on Graphics* **23**(3): 514–521.

Santello, M., Flanders, M. and Soechting, J. F. (1998). Postural hand synergies for tool use, *Journal of Neuroscience* **18**(23): 10105–10115.

Schlömer, T., Heck, D. and Deussen, O. (2011). Farthest-point optimized point sets with maximized minimum distance, *Proc. ACM SIGGRAPH Symposium on High Performance Graphics*, pp. 135–142.

Schneider, P. J. and Eberly, D. (2002). *Geometric Tools for Computer Graphics*, Elsevier Science Inc.

Schröder, M. (2011). *Data-driven real-time hand tracking for interactive applications and robot control*, Master's thesis, Universität Bielefeld.

Schröder, M. and Botsch, M. (2014). Online adaptive PCA for inverse kinematics hand tracking, *Proc. Vision, Modeling and Visualization*, pp. 111–118.

Schröder, M., Elbrechter, C., Maycock, J., Haschke, R., Botsch, M. and Ritter, H. (2012). Real-time hand tracking with a color glove for the actuation of anthropomorphic robot hands, *Proc. IEEE/RAS International Conference on Humanoid Robots*, pp. 262–269.

Schröder, M., Maycock, J. and Botsch, M. (2015). Reduced marker layouts for optical motion capture of hands. submitted to *Motion in Games*.

Schröder, M., Maycock, J., Ritter, H. and Botsch, M. (2013). Analysis of hand synergies for inverse kinematics hand tracking, *Proc. IEEE International Conference on Robotics and Automation*. Workshop: Hand synergies – how to tame the complexity of grasping.

Schröder, M., Maycock, J., Ritter, H. and Botsch, M. (2014). Real-time hand tracking using synergistic inverse kinematics, *Proc. IEEE International Conference on Robotics and Automation*, pp. 5447–5454.

Shadow Robot Company (2015). The Shadow Dextrous Hand.
URL: *http://www.shadowrobot.com/hand/overview.shtml*

Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y., Freedman, D., Kohli, P., Krupka, E., Fitzgibbon, A. and Izadi, S. (2015). Accurate, robust, and flexible real-time hand tracking, *Proc. ACM Conference on Human Factors in Computing Systems*, pp. 3633–3642.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A. and Blake, A. (2011). Real-time human pose recognition in parts from single depth images, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304.

Sridhar, S., Mueller, F., Oulasvirta, A. and Theobalt, C. (2015). Fast and robust hand tracking using detection-guided optimization, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3221.

Sridhar, S., Oulasvirta, A. and Theobalt, C. (2013). Interactive markerless articulated hand motion tracking using RGB and depth data, *Proc. IEEE International Conference on Computer Vision*, pp. 2456–2463.

Sridhar, S., Rhodin, H., Seidel, H.-P., Oulasvirta, A. and Theobalt, C. (2014). Real-time hand tracking using a sum of anisotropic gaussians model, *Proc. International Conference on 3D Vision*, pp. 319–326.

Steffen, J., Elbrechter, C., Haschke, R. and Ritter, H. (2010). Bio-inspired motion strategies for a bimanual manipulation task, *Proc. IEEE/RAS International Conference on Humanoid Robots*, pp. 625–630.

Steffen, J., Maycock, J. and Ritter, H. (2011). Robust dataglove mapping for recording human hand postures, *Proc. International Conference on Intelligent Robotics and Applications*, pp. 34–45.

Stenger, B., Mendonca, P. and Cipolla, R. (2001). Model-based 3D tracking of an articulated hand, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 310–315.

Sudderth, E. B., Mandel, M. I., Freeman, W. T. and Willsky, A. S. (2004). Distributed occlusion reasoning for tracking with nonparametric belief propagation, *Proc. Conference on Neural Information Processing Systems*, pp. 1369–1376.

Sueda, S., Kaufman, A. and Pai, D. K. (2008). Musculotendon simulation for hand animation, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **27**(3): 83:1–83:8.

Tagliasacchi*, A., Schröder*, M., Tkach, A., Bouaziz, S., Botsch, M. and Pauly, M. (2015). Robust articulated-ICP for real-time hand tracking, *Computer Graphics Forum (Proc. Symposium on Geometry Processing)* p. to appear. (* equal contributors).

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

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

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

Thiery, J.-M., Tierny, J. and Boubekeur, T. (2012). CageR: Cage-based reverse engineering of animated 3D shapes, *Computer Graphics Forum (Proc. Eurographics)* **31**(8): 2303–2316.

Tompson, J., Stein, M., Lecun, Y. and Perlin, K. (2014). Real-time continuous pose recovery of human hands using convolutional networks, *ACM Transactions on Graphics* **33**(5): 169:1–169:10.

Torralba, A., Fergus, R. and Freeman, W. T. (2007). Tiny images, *Technical Report MIT-CSAIL-TR-2007-024*, Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology.

Torralba, A., Fergus, R. and Weiss, Y. (2008). Small codes and large image databases for recognition., *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Tresch, M. C., Cheung, V. C. and d'Avella, A. (2006). Matrix factorization algorithms for the identification of muscle synergies: evaluation on simulated and experimental data sets, *Journal of Neurophysiology* **95**(4): 2199–2212.

Tsang, W., Singh, K. and Fiume, E. (2005). Helping hand: An anatomically accurate inverse dynamics solution for unconstrained hand motion, *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 319–328.

Turner, M. (2001). *Programming Dexterous Manipulation by Demonstration*, PhD thesis, Stanford University, Department of Mechanical Engineering.

Urtasun, R. and Darrell, T. (2008). Sparse probabilistic regression for activity-independent human pose inference, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Vaillant, R., Barthe, L., Guennebaud, G., Cani, M.-P., Rohmer, D., Wyvill, B., Gourmel, O. and Paulin, M. (2013). Implicit skinning: Real-time skin deformation with contact modeling, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **32**(4): 125:1–125:12.

Vaillant, R., Guennebaud, G., Barthe, L., Wyvill, B. and Cani, M.-P. (2014). Robust iso-surface tracking for interactive character skinning, *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* **33**(6): 189:1–189:11.

Vicon (2015).
URL: *http://www.vicon.com/*

Wang, R., Paris, S. and Popović, J. (2011). 6d hands: Markerless hand-tracking for computer aided design, *Proc. ACM Symposium on User Interface Software and Technology*, pp. 549–558.

Wang, R. Y. and Popović, J. (2009). Real-time hand-tracking with a color glove, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **28**(3): 63:1–63:8.

Wang, Y., Min, J., Zhang, J., Liu, Y., Xu, F., Dai, Q. and Chai, J. (2013). Video-based hand manipulation capture through composite motion control, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **32**(4): 43:1–43:14.

Wei, X., Zhang, P. and Chai, J. (2012). Accurate realtime full-body motion capture using a single depth camera, *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* **31**(6): 188:1–188:12.

Weise, T., Bouaziz, S., Li, H. and Pauly, M. (2011). Realtime performance-based facial animation, *ACM Transactions on Graphics (Proc. SIGGRAPH)* **30**(4): 77:1–77:10.

Wheatland, N., Jörg, S. and Zordan, V. (2013). Automatic hand-over animation using principle component analysis, *Proc. Motion in Games*, pp. 175:197–175:202.

Wheatland, N., Wang, Y., Song, H., Neff, M., Zordan, V. and Jörg, S. (2015). State of the art in hand and finger modeling and animation, *Computer Graphics Forum (Proc. Eurographics)* **34**(2): 735–760.

Wu, Y., Lin, J. Y. and Huang, T. S. (2001). Capturing natural hand articulation, *Proc. IEEE International Conference on Computer Vision*, pp. 426–432.

Xsens (2015).
URL: *http://www.xsens.com/*

Ye, M., Zhang, Q., Wang, L., Zhu, J., Yang, R. and Gall, J. (2013). A survey on human motion analysis from depth data., *Time-of-Flight and Depth Imaging*, Vol. 8200 of *Lecture Notes in Computer Science*, pp. 149–187.

Zhang, P., Siu, K., Zhang, J., Liu, C. K. and Chai, J. (2014). Leveraging depth cameras and wearable pressure sensors for full-body kinematics and dynamics capture, *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* **33**(6): 221:1–221:14.

Zhao, W., Chai, J. and Xu, Y.-Q. (2012). Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data, *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 33–42.

Zheng, J., De La Rosa, S. and Dollar, A. (2011). An investigation of grasp type and frequency in daily household and machine shop tasks, *Proc. IEEE International Conference on Robotics and Automation*, pp. 4169–4175.

Zhu, L., Hu, X. and Kavan, L. (2015). Adaptable anatomical models for realistic bone motion reconstruction, *Computer Graphics Forum (Proc. Eurographics)* **34**(2): 459–471.