Model-based 3D hand pose estimation from monocular video

Martin De de La Gorce La Gorce

ECOLE CENTRALE DE PARIS

P H D T H E S I S

to obtain the title of

PhD of Science

of Ecole Centrale de Paris
Specialty : Applied Mathematics

Defended by

Martin DE LA GORCE

Model-based 3D Hand Pose Estimation from Monocular Video

Thesis Advisor: Nikos PARAGIOS

prepared at Ecole Centrale de Paris, MAS laboratory

defended on December 14, 2009

Jury :

Reviewers :	Dimitri METAXAS	-	Rutgers University
	Pascal FUA	-	EPFL
Advisor :	Nikos PARAGIOS	-	Ecole Centrale de Paris
Examinators :	Radu Patrice HORAUD	-	INRIA
	Renaud KERIVEN	-	Ecole de Ponts Paritech
	Adrien BARTOLI	-	University d'Auvergne
	Bjorn STENGER	-	Toshiba Research
Invited :	David FLEET	-	University of Toronto.

Acknowledgments

I would like to thank the people that helped me during preparing my PhD these last four years.

Thank to my PhD advisor, Nikos Paragios, who has been supportive and had confidence in my work. He gave me the freedom I needed, proposed me very relevant directions and has been precious in is help to communicate and yield visibility to my research results. I also greatly appreciated his encouragements to establish international collaborations by visiting other prestigious research centers.

Thank to David Fleet for the extremely productive collaboration we started during a two month visit in the university of Toronto. While discussing the results I obtained with the method presented in the third chapter of this manuscript, he made a simple but sound remark that motivated the direction taken in the fourth chapter: "if you needed to add shading onto your hand model to get a good visualization of your results, that means that you should add shading in the generative model you use for the tracking". I deeply appreciated his strive for perfection in the explanation of scientific ideas and his enthusiasm to discuss about in-depth technical aspects as well as the great challenges of the field in general. His help as also be precious in the writing of this manuscript.

I am grateful to my thesis rapporteurs Dimitri Metaxas and Pascal Fua, for having kindly accepted to review this work. I appreciated their valuable comments. Thank to Radu Patrice Horaud, Renaud Keriven, Adrien Bartoly and Bjorn Stenger for examining it and for the constructive discussion during the thesis defense.

Thanks to all the current and former members of the Medical Imaging and Computer Vision Group at the Applied Mathematics Department in Ecole Centrale for the friendly international environment and the great working atmosphere. In particular I would like to thank Chaohui Wang and Mickaël Savinaud for our fruitful collaboration. Thank to Noura and Regis for their moral support. Thanks to my friends Geoffray and Romain for having confirming me by their example in the idea that having hobbies is a necessary condition to do a good PhD thesis. And finally, thank to all my other friend and my family members who have been supporting during these four years.

Index

Abstract

The objective of this thesis is to design an algorithm that allows to automatically recover a full description of the 3D motion of a hand from a monocular (i.e. single-view) video sequence. The solution we propose constitutes a preliminary step towards designing new natural human-computer interfaces or automatic gesture analysis. In this thesis, we introduce two model based methods that improve state-of-the-art techniques. To this end, an articulated hand model is fitted on each image by minimizing the discrepancy between the projection of the 3D model in the image plane and the observed image. This minimization is done efficiently using an iterative refinement technique that is based on quasi-Newton gradient-descents. The first method relies on a hand model composed of ellipsoids and convex polyhedra. The discrepancy measure is based on global color distributions of the hand and the background. The second method uses a triangulated surface model with texture and shading. In this case, the discrepancy measure is defined by comparing for each pixel the color intensities of the observed image with the synthetic image generated with the hand model. While computing the gradient of the discrepancy measure, a particular attention is given to terms related to the changes of visibility of the surface near self-occlusion boundaries that are neglected in existing formulations. Finally, a prior distribution of typical hand poses is learned from examples towards improving tracking performance.

Notations

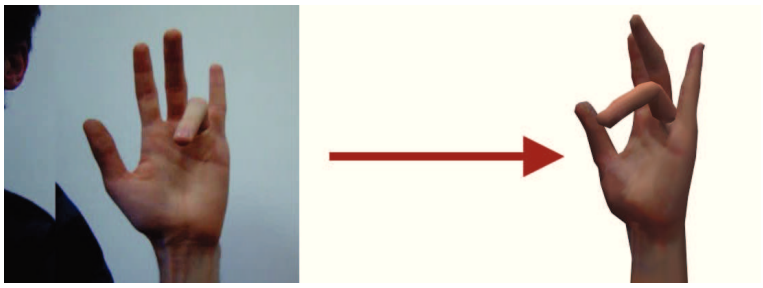
- $\mathbb{N}$ the set of integers
- $\mathbb{R}$ the set of reals
- $\overline{ab}$ the segment joining a and b
- A^T the transpose of matrix A
- A^{-1} the inverse of matrix A
- $\det(A)$ is the determinant of matrix A
- $A_{i,j}$ The element of A located at lines i and column j
- $A_{[i:j,k:l]}$ The submatrix of A obtained by selecting from lines i to j and columns k to l
- $A_{[:,k:l]}$ The submatrix of A obtained by selecting from lines lines and columns k to l
- $A_{[i:j,:]}$ The submatrix of A obtained by selecting from lines i to j and all columns
- $x_{i:j}$ sub vector by selecting component for i to j or (depending on the context) the set $\{x_i, \ldots, x_j\}$
- $p(x)$ probability of x
- $N(x|\mu, \Sigma)$ x is a random variable normally distribed, with mean $= \mu$, and covariance Σ.
- $0_{i\times j}$ is the i by j zero matrix
- $I_{n\times n}$ is the n by n identity matrix
- $SO(3)$ is the set of 3 by rotation matrix ($R \in SO(3) \Leftrightarrow R^tR = I_{3\times 3}$ and$det(R) = 1$)
- v^D is the diagonal matrix whose diagonal elements are components of the vector v
- ∇f Laplacian of a scalar function f
- f(x,.) is the one variable function that maps y to $f(x, y)$
- $\lfloor x \rfloor$ is the greatest integer n such that $n \leq x$.
- $\varepsilon(x)$ is the fractional part of x i.e $\varepsilon(x) \equiv x - \lfloor x \rfloor$

Caput 1

Introduction

1.1 General introduction

The objective of this thesis is to design an algorithm that would allow to recover automatically a full description of the 3d motion of a hand given a monocular (i.e. single-view) video sequence. Using the information provided by the video, our aim to is to determine the full set of kinematic parameters that are required to describe the pose of the skeleton of the hand. This set of parameters is composed of the angles associate to each joint/articulation and the global position and orientation of the wrist. Given these parameters, the 3D pose of the hand can then be fully reconstructed. This is illustrated in [Figure (1.1)] where the recovered hand pose parameters are used to synthesize the hand from a new view-point, from which the hand has not been seen before.



Descriptio 1.1: The 3D hand pose is inferred from the left image and is used to synthesize the hand from a new view-point in the right image

The method would preferably be sequential, in the sense that we could handle scenarios where the images are available one at a time and the hand pose is estimated after each new observation. We would like a method that can work in a single-view setting. In comparison with multi-view settings (using mirrors or several cameras), this setting is more affordable and convenient for the user but makes articulated hand tracking more challenging as the depth information cannot be retrieved. The motivation to solve the monocular hand tracking problem is twofold. One can view this problem both from practical and scientific interest. From practical point of view, many applications involving human-Computer interfaces or gesture analysis would benefit from such a development (see section 1.2). From practical point of view, this problem is extremely challenging in many aspects. One can cite for example the important number of degrees of freedom of the hand, the presence of

self-occlusions as well as the depth ambiguities inherent to monocular data. Such technical difficulties are not specific to hand tracking and elements of the solution could be of interest to other articulated-body tracking problems or other computer vision problem tasks where one has to deal with occlusions and depth ambiguities.

In this thesis, we introduce two novel methods of increasing complexity that improve to certain extend the state-of-the-art for monocular hand tracking problem. Both are model-based methods and are based on a hand model that is fitted to the image. This process is guided by an objective function that defines some image-based measure of the hand projection given the model parameters. The fitting process is achieved through an iterative refinement technique that is based on gradient-descent and aims a minimizing the objective function There are significant differences both in terms of modeling as well as in terms of inference between to the two proposed solutions.

The first method relies on a hand model made of ellipsoids and a simple discrepancy measure based on global color distributions of the hand and the background. The second method uses a triangulated surface model with texture and shading and exploits a robust distance between the synthetic and observed image as discrepancy measure. Like most of the state of the art monocular hand tracking methods, robustness and speed are requirements that would make it suitable for general use. Our hand tracking method is not real-time, which makes interactive applications not yet possible. Increase of computation power of computers and improvement of our method might make real-time attainable.

In this chapter we will first review the applications that motivate designing hand-tracking solutions, then we will discuss the different reasons that make hand-tracking a difficult problem to solve, especially in a monocular setting, and finally we will present of short overview of our contributions.

1.2 Applications of Hand tracking

Among different body parts, the hand is the most efficient, general-purpose interaction tool due to its dexterous functionality in communication and manipulation. This statement is supported physiologically by the fact that the area in the human brain dedicated to the control of the hand's movements is about as large as the total of the areas dedicated to the arms, the torso and the lower body [Mitobe 2007]. Probably due to the central role of the hands in gestural communication, a system able to make sense of hand motion have been envisioned by several science-fictions film directors in movies such as Johnny Mnemonic (1995,[Figure (1.2.a)] , Final Fantasy (2001) or Minority Report (Spielberg 2002 [Figure (1.2.b)]) .

Hand motion capture is actually not quite science-fiction but is already possible using physical systems such as data gloves that are equipped of sensors that can digitize the position the fingers (see section 2.5.1). These systems are efficient but intrusive, too expensive for most applications and generally their performance is decreasing quickly with extended use. Because human cognitive systems are

john mnemonic,1995

minority report,2002

Descriptio 1.2: Data glove in movies

able to infer the pose of a filmed hand, one would expect that alternative non-intrusive systems could be designed using cameras and Computer-Vision techniques. In comparison to glove-based systems, existing computer-vision (CV) systems have the advantage to be inexpensive but still lack of precision, robustness and speed for general uses. Before detailing further the existing hand motion capture systems and compare their respective advantages, we will first motivates development of CV hand tracking systems by presenting different scenarios where it can be used for.

The goal of this thesis is to obtain estimates related with the full 3D pose of the hand. However we will see that for some applications it might suffice to estimate only the 3D pose of some specific parts of the hand such as the fingertip(s) or the palm or even 2D position of fingertip(s) projections in the camera image plane. Despite not being essential for these applications, a full hand pose estimator would nevertheless remain of great interest. Indeed, rather than designing a new algorithm to extract needed information (that is specific to the task) directly from the image, it appears simpler to extract this information from the full hand pose estimates if already provided by a general-purpose hand pose estimator. Even if, for such applications, only a portion of the hand pose information is retained, it would eventually be easier to add new functionalities by retaining more information on the hand pose if it is already available. Let us now review application areas where such an inference system could be of great interest.

1.2.1 Animation

Motion capture (MoCap) systems digitalize actor's body movement and are more and more often used in the cinema and game industry to animate virtual characters with realism. Once the movement of the actor is captured, the estimated pose parameters are used to animate a virtual character. In the context of classical movies, this is a great help in situations where the performance would impractical or too dangerous for real actors or where the choice of camera angle would have make

the shooting difficult or impossible to do with real cameras. This is also amplified for scenes or movies involving no-human behaviors like animals for example. In such a context the motion capture system is not required to be real-time. Real-time motions capture systems are sometimes used for live television broadcast in order to place a virtual character interacting with real people within a real or virtual scene. Most character motion capture systems does not measure secondary motions like detailed hand motion. The resulting animations look unnatural due to the stiffness of hand motion. Systems dedicated to hand motion capture such as data-gloves (see section 2.5.1) are therefore useful to increase realism in character's full body animation or when doing close-up on hands in an animated sequence. Of course, we could mention also the importance of these techniques on the recent explosion of fully animated movies that go beyond children audience and become a main stream in cinematography.

1.2.2 Quantitative Motions analysis

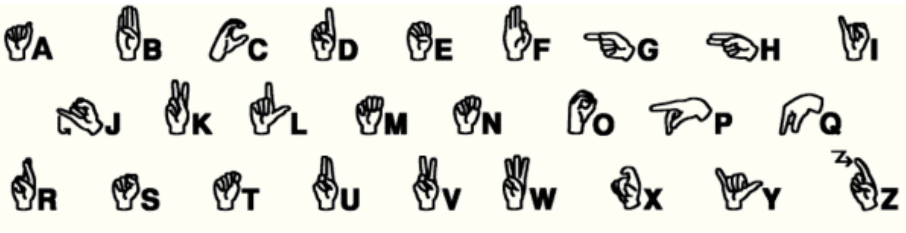
Motion capture systems can be use to study the human movements in general and the human locomotion in particular. We refer to gait analysis in the latter case. The interest of using motion capture for quantitative motions analysis lies mainly in its ability to produce repeatable results. In the field of clinical medicine, motion capture can be used to measure the extent of a client's disability as well as a client's progress with rehabilitation. By accurately measuring hand and finger movement one can quantitatively monitor the recovering from hand or nervous-system injury. Non invasive motion-capture systems are preferable for such medical applications, as it allows the clinicians to evaluate the motion without burdening patients with tracking devices. Motion capture can also be used for sportive gestures analysis towards helping athletes to improve their performances. However, capturing the detailed movements of the hands might not be of great interest for these applications.

Motion capture can help to design product that are ergonomically practical. For example, in the medical fields, motion capture can help to design effective prosthetic devices. One can cite for example aid systems for elder population [Xin 2007]. This subset of population is often physically challenged and motion capture can help to study the age-related physical limitations, especially the ones related with motion. Non invasive motion-capture systems can improve psychological confidence or freedom of the subjects over invasive motion-capture systems, and help the subjects to focus on performing their tasks.

1.2.3 Sign Language Recognition

Hand motion capture can be used as a central component in an automatic sign language recognition system. A sign language is a language that conveys meaning through the use of visual sign patterns combining hand shapes, orientation and movement of the hands, arms or body, lips movements and facial expressions. The

manual alphabet augments the vocabulary of a sign language when spelling individual letters of a word. Signs in the American Sign Language (ASL) correspond to dynamic hand movements and are associated to a full world while the letters of the American Manual Alphabet (AMA [Figure (1.3)]) correspond to static hand pose (the letters J and Z are exceptions). A system able to perform automatic sign language recognition would allow translating sign language into written text or synthetic speech and therefore enabling communications between deaf and normal people. In particular, the combination of an automatic speech recognition with ASL synthesis could produce a new bidirectional communication framework between hearing-impaired people and non-impaired people. There is certain prior work on this direction like for example the ones in [Takahashi 1991][1][Liang 1998], [Kadous 1995],[Murakami 1991][2] [Lee 1996][3] which rely on invasive motion capture systems - such as data-gloves - to obtain hand pose measurements which are used as features to perform automatic classification.



Descriptio 1.3: American Manual Alphabet (AMA)

However, as it is mentioned in section 2.5.1, data-gloves are expensive and not suited for frequent use. An alternative solution consists in using computer-vision techniques. A possible approach is to proceed in two steps: 1) estimate the 3D hand pose from the images using method like the ones developed in this thesis, and 2) use the estimated pose parameters as features for classification. Because 3D hand pose estimation is already a difficult problem, this is not the approach generally taken in the literature aiming at recognizing signs language. Most methods perform gesture classification using feature directly extracted from the images and do not explicitly estimate the 3D hand poses. In comparison with feature directly extracted from the image, 3D hand pose estimates present the advantage to contain all the useful information for sign recognition while being independent of the view-point, illumination etc.

For more details on gesture recognition in general, we refer the reader to several reviews [Wu 1999c, Wu 1999b]. Since the great majority of these methods do not explicitly estimate the 3D hand poses, the pose estimation problem is not detailed in these reviews.

[1] ï¿½ verifier
[2] ï¿½ verifier
[3] ï¿½ verifier

1.2.4 2D human-Computer interaction

Hand motion capture can be used as a central component to design 2D human-Computer interfaces. One can imagine numerous scenarios, like 2D virtual environments or augmented reality applications to manipulate electronic documents, presentations or other applications projected on desks, walls etc. We refer the reader to [Moeslund 2003] for complete review of such systems. The estimated index finger-tip position in the image plane can be used as a pointer like a virtual 2D mouse. This can be associated with the automatic recognition of gestures or poses among a finite set of predefined gestures/poses being associated with semantic meanings. The detected gestures/poses can be used to trigger actions (select/release for example) in a similar fashion to mouse buttons or keys on a keyboard. Using the tips position of two fingers (the thumb and the index for example) one can also perform simple 2D manipulation tasks such as translation, rotation and re-scale (the 2D position of two fingers provide a 4D parameters vector that matches the dimension of the space of transformation obtained by the above mentioned transformations). 3D hand pose provides a straightforward manner to obtain the 2D position of the finger tips in the image plane and to detect postures that trigger actions using some classifiers with the estimated hand parameters as features. The use of invasive motion capture systems is expensive and not suited for frequent use. Therefore computer-vision systems are a good alternative. The difficulty of the 3D hand pose estimation problem from images had as a result that most of the methods found in the literature do work directly with high-level features extracted from the images and/or with a simplified 2D hand models. The advantage in term of expressiveness of such 2D HCI over a mouse and a keyboard is not obvious. Furthermore, the user has to remember the set of command gestures which may contain ones that are not very natural to perform. The advantage of such 2D HCI systems over a mouse and a keyboard appears when non-invasive computer vision techniques are used. These methods are suitable for numerous scenarios where contacts between the user and the system are either not possible, or not allowed, such as in an operating room or in public places where we wish to avoid deterioration of the HCI by the users.

1.2.5 3D human-Computer interaction

Hand motion capture can be used as a central component to design 3D human-Computer interfaces in various scenarios including 3D Virtual Environments (VE) or augmented reality (AR) applications. If the 3D pose of a pointing index is recovered accurately, this can be used to select an object of interest in the 3D environment. The 6D rigid transformations of the palm can be estimated and applied on the selected object. Like in the 2D case, specific hand poses can be used to trigger actions. However this type of 6D interface does not fully exploit the full range of expression of the hand and such reduced functionalities could be obtained through simple 3D mouse devices. Another approach to manipulate 3D

virtual object consists in capturing the motion of the whole hand and simulating physically realistic reactions of the virtual object to the contact of the hand. Such approaches require haptic devices such as exoskeleton (see section 2.5.1) that can provide force feedback for the user to feel contact with the virtual object and object's inertia. Computer-Vision based hand motion trackers are therefore not suitable in this setting.

1.2.6 The hand as a high DOF control device

Hand motion capture can be used to devise high degree of freedom (DOF) control devices. The hand skeleton has about 28 degrees of freedom (24 joint angles + 6 DOF for global position and orientation of the wrist). By mapping the measure associated to each captured DOF into a real valued controller one can simultaneously control 26 continuous 1D inputs.

This has been experimented for live music performance in 1989, when the MIT Media-Lab composer Tod Machover included some form of whole-hand input in the performance of a piece he had been commissioned to write for the 50th anniversary of the Nippon University. The results had been promising however lag and imprecision on the measures prevented the consistency and accuracy needed by professional musicians [Sturman 1994, Sturman 1992].

To conclude, it is clear that there is an enormous and wide spread demand for vision-based hand-pose estimation problems

1.3 Hand Pose Estimation Scientific Challenges

Building a fast and effective vision-based hand pose tracker is quite challenging. Hand pose estimation from images is difficult due to various factors. One should separate these challenges into two categories, the ones due to the model and problem complexity from mathematical point of view and the ones due to the observations limitations.

- **Dimensionality**

  The hand skeleton has approximately 30 significant degrees of freedom, and therefore the state space of possible hand poses is enormous. The search for the best pose in such a high-dimensional state space is computationally demanding. The state space search can be improved by exploiting interdependence between fingers. This is generally done by constraining the hand pose to remain within a low dimensional sub-manifold of the Cartesian product of the individual parameter ranges. This dimensionality reduction is driven from considerations about the physiology of the hand or learned from a training set of poses that is assumed to be representative of the kind of gestures we aim to track. In the context of hand tracking several approaches have been proposed [Lin 2000, Wu 2001, Zhou 2003, Thayananthan 2003a](see section 2.2.2.2). In contrast with body tracking where the motion often fall on either the walking

or running class, it seems impractical to restraint the classical hand motion to a small set of activities that can be described by low dimensional manifolds in the pose space. In [Zhou 2003] the manifold as been obtained using a 6 dimensional PCA on training data collected from a CyberGlove. However such a dimensionality reduction of the pose space will lead to a slightly inaccurate representation of the hand that will be unable to fully explain the observed image. A solution might consist in progressively re-augmenting the dimensionality to refine the pose estimation. We refer to the section 4.5.2 for more details about the method we chose in order to learn the manifold of hand poses.

- **Range and Rate of Changes of the Model Parameters**

  The hand movements are often quick but also complex. According to [Erol 2007] its speed reach up to 5 m/s for translation and 300 degrees/s for wrist rotation. This has several consequences that make hand tracking difficult. First, predictions of the future state from estimated states at previous time instants inherit significant uncertainty. As a consequence we cannot focus the search of the hand pose in a narrow region of the space of hand pose parameters. Second, off-the-shelf cameras can generally support 30Hz frame rates. Therefore one can observe up to 10 degrees inter-frame wrist rotations. That may cause important motion blur which might deteriorates clues extracted in the image. In order to address the challenges associated to the speed of the hand motion and depending on the application, it might acceptable to ask the user to avoid rapid hand motions. However in his thesis Sturman [Sturman 1992] recommends at least 100 Hz tracking speed to allow effective human-computer interaction.

- **Presence of self occlusions**

  The hand is an articulated object and its projection often results in important self-occlusions. These occlusions make practically impossible to segment different parts of the hand and to extract reliably features such as optical flow or depth maps in stereos settings. Treating occlusion through an outlier process (or robust functions) might not be sufficient to treat large occlusion. The occlusion dependency on the hand pose should be modeled in the generative model towards avoiding the attribution of image measurements to occluded parts of the hand. Whenever parts are fully occluded (in all cameras, in multi-camera settings) their pose cannot be estimated based on image measurements. This introduces uncertainty to the pose estimate of the hidden parts. Without image support, it is generally not possible to predict the location where these parts are reappearing. This might cause the tracker to fail whenever the algorithm heavily relies on pose prediction to initialize local exploration in the pose space. Note that self-occlusions are less of a problem when using a wide base-line multi-camera setting (see section 2.1.3) because it is unlikely that a part of the hand is simultaneously occluded in all images

taken from very different viewpoints. A possibility to limit occlusions between fingers in a monocular setting is to restrict the palm to remain parallel to the image plane. However, such a constraint might be acceptable only for a limited set of applications.

- **Lack of Texture and color Discrimination Between Hand Parts**

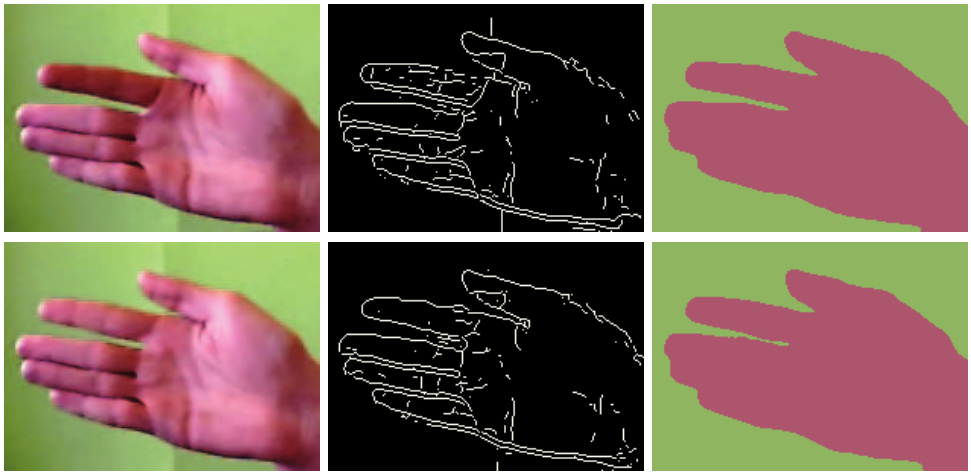
  Most of the hand is skin colored which makes difficult to discriminate one part of the hand from another based solely on color. This contrast with body tracking where hands, arms and legs have often different colors. The hands do not present contrasted texture, which makes optical flow, point tracking or stereo reconstruction often inefficient. Edges around fingers that are occluding other parts of the hand are difficult to discriminate from wrinkles.

- **Self collisions**

  Whenever a generative approach is used, one needs to ensure that the estimated hand pose is physically possible. One of the constraints is that two different part of the hand should not inter-penetrate. In the hand parameter space, the set of points representing admissible poses that do not present inter-penetration is a sub-volume of the initial hand parameter space. Unfortunately this set is not convex and confining the search in this set is far from trivial, especially for optimization-based methods.

- **Depth ambiguities**

  Working with a monocular setting is challenging compared to multi-camera settings because the depth information cannot be retrieved. The lack of depth information makes the estimation of the hand pose under-constrained or ill-posed. Distinct 3D hand poses hypothesis with different depth may lead to very similar 2D projection and thus could equally well explain 2D cues or features extracted from the observed image. Whenever the considered method is generative and the pose estimation problem is formulated as the minimization of some objective function based on the observed image cues, these ambiguities result into the existence of several equally low minima of the objective function. Similar to the variational approaches, in the case of Bayesian inference, these ambiguities produce multiple modes in the posterior distribution of the hand pose parameters.Such observation ambiguities depend on the nature of the features extracted from the images. This is illustrated in [Figure (1.4)] where the first column shows images of a hand with two different poses of the index finger. The other two columns show the corresponding edge maps and silhouettes based on color, both of which do not change significantly as the finger moves. The edges and the silhouette extracted in the image do not provide enough information to disambiguate between the two finger positions while inferring the hand pose. However, while looking the first column in [Figure (1.4)], an human is able to disambiguate between the two finger positions. This illustrates the fact that humans are generally able to guess



Descriptio 1.4: Two hand pictures in the first column and their corresponding edge map and segmented silhouette in the two other columns. Edge and Silhouette are little informative to disambiguate the two different index poses.

the 3D pose of a hand even with a single eye. In the example [Figure (1.4)] the visual information that allows disambiguation between the two poses is very likely to be conveyed by the shading. Because the shading depends on the orientation of the normal to the surface, it provides information about the local depth variations of the surface. This relation is exploited by general shape-from-shading techniques to infer depth from monocular data whenever the surface has uniform reflectance and the lightning is controlled.

- **Cluttered background**

  Edge detection in cluttered background provides many edges that do not relate to the hand and that are likely to distract the tracker. Furthermore, intensity separation between cluttered background (multi-modal unknown density) and skin color is not always straightforward. Dealing with arbitrary background is a challenging issue in computer vision, even for rigid objects tracking. A classical approach to overcome these problems it to assume that either it is uniform, static or with a color that differs significantly from the skin color. Note however that such assumptions might not be generally applicable for real-life applications.

To conclude, hand-pose estimation using computer vision techniques is a challenging problem from technical and practical perspective. In this thesis we made an effort to provide answers to some of the limitations of the existing methods through tow distinct contributions.

1.4 Contributions & outline

Our contributions consist of what are so-called model-based methods where candidate hand poses are evaluated using a synthetic model of the hand whose projection in the image is compared to the observed image. The methods do differ in terms of modeling as well as in terms of inference. The first one is a low complexity approach both in terms of the hand model as well as in terms of model-to-data association and inference [de La Gorce 2006, de La Gorce 2010b]. The second one introduces a more complex and richer model that is combined with a powerful image-based criterion and a term that account for prior knowledge of the hand-pose manifold [de La Gorce 2008, de La Gorce 2010a].

These two methods differ by the choice of the hand surface model and by the criterion used while comparing the model projection and the observed image. For each of the two method we propose solutions to some of the limitations of the previous methods we listed in section 2.6. Unfortunately, like for other monocular hand tracking methods proposed in the literature, robustness and speed are still two major concerns that make our methods not yet suitable for general use. We will briefly present the essence of these methods in the upcoming sections.

1.4.1 Part-based Hand Representation and Statistical Inference

In this context, the hand is modeled as an articulated body made of ellipsoids and polyhedra. For each frame, the model is fitted onto the observed image by minimizing a matching cost that is defined using color models of the background and of the hand. This cost is defined by summing the logarithm of the likelihood ratio (between hand and background likelihoods based on color) over all points within the synthetic silhouette. Ellipses are approximated by polygons leading to an approximate matching cost that can be numerically evaluated exactly while being a continuous and differentiable function of the hand pose parameters. The hand pose estimation is done by minimizing the matching cost using a trust-region method that exploits its exact gradient and an approximation of its Hessian. This local search method is combined with a particle filter in order to deal with occlusions and local minima through multiple hypotheses.

The main contributions of this method are the following:

- The methodology that consists in defining a continuous matching cost that can be computed exactly and whose gradient can also be computed exactly.
- The use of a polygonal discretization of the silhouette in order to define an approximate matching cost that is computationally tractable and differentiable.
- The design of a new algorithm that allows to compute quickly the exact integral of an interpolated gray-level image within a polygon with non integer vertex coordinates. The algorithm also computes the derivative of this integral with respect to the vertex positions.

- The introduction of a variable metric gradient descent and a efficient trust-region method for model fitting.

The main limitation of this method is its inability to disambiguate between different poses that produces the same hand silhouette. This is due to the lack of visual information extracted from the image, such as the information provided by the texture and the shading of the observed hand.

1.4.2 Triangular Mesh with Texture & Shading

The hand is modeled as deformable triangulated surface with an underlying skeleton controlling the deformations. In order to exploit shading information in the images, the hand model surface is shaded using some hypothesis on the direction and the strength of the lightning. Towards capturing dense information provided by fine asperities of the albedo, the fine texture of the hand is learned from the image sequence and mapped onto the hand model surface. Given an hypothesized hand pose and lightning condition, a synthetic image of the hand including texture and shading is generated and the matching cost is defined as the sum of individual pixel error between the synthetic image and the observed image. Due to the discretization on a pixel grid, this matching cost is discontinuous. We proposed a continuous version of the matching cost by considering the image domain as a continuous domain and we derive the analytical expression for the gradient of this continuous matching cost. The appropriate treatment of the occlusion boundary yield new *occlusion forces* along these occlusion boundary and a theoretical connection is made with active regions and active polygons. The pose estimation is done by finding hand pose and lightning parameters that minimize this matching error.

Like for the first method an effort has been made to propose an implementation of the matching cost that remains a continuous and differentiable function of the hand pose parameters, despite the discretization on the image grid.

Among the contribution of this approaches are

- The use of shading and texture in the model

- The derivation of the *occlusion forces* that account for the progressive change of visibility around occlusion boundary when the hand displace.

- The methodology that consist in 1) proposing an implementation of the matching cost that remains a continuous and differentiable function of the hand pose parameters, despite the discretization on the image grid 2) differentiating the implemented version of the matching cost.

- The introduction of a block-wise BFGS formula to exploit independences between hand parts and speed up the convergence rate over the standard BFGS formula.

1.4.3 Outline

The reminder of this document is organized in the following manner. In chapter two we briefly review the state of the art methods in hand pose estimation for monocular or stereo settings. We present related work both in terms of the modeling aspect as well as in terms of inference one. The main contributions of this thesis are presented in the next two chapters. First, we introduce a part-based representation and a corresponding inference method that aims to globally separate hand from the background through the use of regional statistics. The second method consists of a powerful richer model that includes texture, shading, lighting, etc... and is inferred through the definition of a global objective functions that minimizes local discrepancies between the image and the synthesized hand image. This framework is amended to account for prior knowledge. Discussion and future directions conclude this thesis.

Caput 2

State of the art

In this section we will review methods that have been proposed in the literature to design automatic hand pose estimation systems. We will use as basis the classification being proposed in [Erol 2007], which inspired in some aspect our own review. Prior in hand pose estimation could be classified in different ways, like for example according to the acquisition framework, the type of image features that are used, the presence/absence of constraints on the user's hand poses/movements or on the theoretical approach (model-based or discriminative) the method relies on. The two classification axes that seemed the most relevant are 1) the type of input provided by the acquisition system and 2) the theoretical approach considered to estimate the hand pose from the input. We will first present the different acquisition settings that have been considered to do hand tracking. We will then adopt a theoretical view point and treat separately the two main approaches that are the model-based approach and the classification/regression based approaches. Finally we will discuss the literature treating related problems such as non-optical hand tracking systems, automatic gesture recognition and body tracking.

2.1 Acquisition framework

Several optical acquisition systems can be considered to perform computer-vision hand tracking, In particular one can refer to the number of cameras, the use of markers etc. In this section, we briefly review the major optical acquisition systems that have been use for hand tracking. Non-optical acquisition systems are discussed in the section 2.5.1.

2.1.1 Monocular setting

The most affordable acquisition system is a system composed of a single classic camera or a webcam. This system is the most convenient for the user, of low cost and thus is accessible to a wide audience. A system based on a a single camera is eventually easy to transport and can be wear on a hat (see [Liu 2004, Starner 1998]). However having a single view-point of the hand makes 3D hand-tracking a very challenging problem. In comparison with multi-camera settings, the depth cannot be obtain by triangulation and is difficult to infer from the data. Subtle image information such as the information conveyed by shading is crucial to infer depth of the different parts of the hand (see section 1.3).

Among methods using a monocular setting to perform hand pose estimation are the methods proposed in [Athitsos 2002, Ouhaddi 1999a, Lee 2004, Stenger 2001c,

Stenger 2001b, Lin 2004, Wu 2001, Kuch 1995, Zhou 2005a, Zhou 2003, Wu 1999a, Sudderth 2004, Shimada 2001, Rosales 2001, Potamias 2008, Lu 2003, Kato 2006, Imai 2004, Heap 1996, Ge 2008, Cui. J-S 2004, Athitsos 2003a, Guan 1999, Chua 2002].

2.1.2 Small baseline stereo setting

Another classical setting consists in taking two cameras with a distance between them that is small in comparison with the distance to the hand. Thus the two cameras are approximatively pointing at the same direction and are capturing the hand with a slight change of the view-point. Using the two synchronized image sequences, some depth information can be inferred. Image-based stereo-reconstruction consists in (i) finding dense correspondences between the two images and (ii) performing triangulation towards recovering a "continuous" depth map. Because the hand has little texture and because for several configurations one observes significant occlusions, it is difficult to find dense correspondences if the resolution of the images is low to medium. The presence of occlusions cannot be handled very well by stereo reconstruction algorithms.

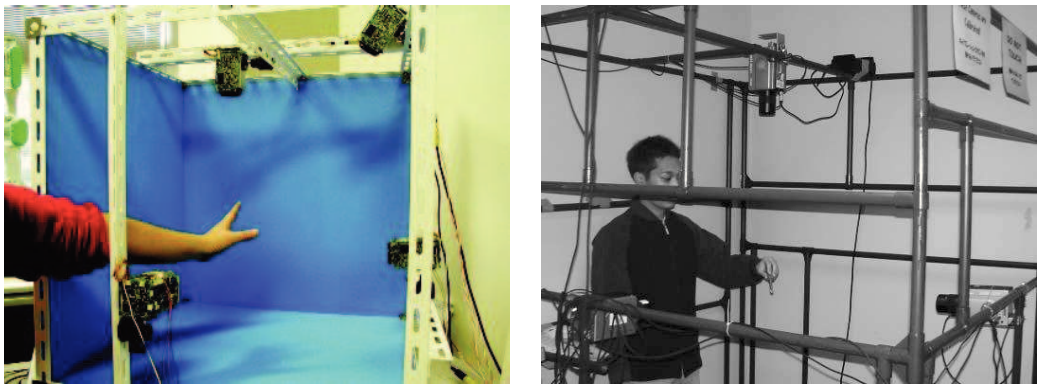
The triangulation process, requires some knowledge about the 3D positions of the two cameras and therefore an additional calibration process is to be considered [Clarke 1998]. If the main aim is to determine the 3D hand pose relatively to the cameras, then we only need to determine position of the second cameras relatively to the first camera (the system has to be calibrated each time the relative position of the two cameras is modified). Therefore it can be usefully to fix the two cameras on the same rigid frame. Another difficulty is to synchronize the two cameras such that the two acquired sequences refer to frames that are taken exactly in the same time (the maximal hand displacement in the images during the time discrepancy between the two cameras should be smaller than a pixel). This cannot be easily achieved using standard webcam but can be done with firewire cameras with a clock signal. Parameters of the cameras such as exposure and color balance should also be calibrated such that the acquired sequences refer to comparable pixel intensities. If the reliable depth information can be inferred using a stereo setting, then it would greatly facilitate tracking in comparison with monocular setting.

Among methods using a stereo setting to perform hand pose estimation are the methods proposed in [Delamarre 1998, Dewaele 2004, Chik 2007] (see section 2.2.3.3).

2.1.3 Wide baseline setting

Another muti-camera setting consists in taking two or more cameras with their relative distance comparable with their distance from the hand. Thus the two camera optical axis are pointing at very different directions (sometimes orthogonal one from an other when using 3 cameras [Ogawara 2003, Ueda 2005]) and the hand

is seen from different view-points. Two wide baseline setting that have been used for hand tracking are shown in [Figure (2.1)].



Descriptio 2.1: Wide baseline capture systems. Authors names from left to right: Ueda [Ueda 2005], Ogawara [Ogawara 2003]

The main challenge is to find correspondences between the two images. The local appearance of the image patches corresponding to the same physical hand point might be very different due to specularity for example, or because points visible in an image might not be visible in other images.

Nevertheless, some depth information can be inferred without finding correspondences but by computing the so-called visual hull (see section 2.2.3.3). The visual hull can then be used as a valuable cue for hand pose inference. Extracting the visual hull requires hand segmentation in images which can be facilitate using infrared cameras. Such a method requires the knowledge of the relative positions of the cameras. This can obtained through a multi-camera calibration process [Svoboda 2005] that has to be repeated each time a camera is moved with respect to the others. Because the cameras are not close from each other, it is difficult to fix them onto a single transportable rigid frame which make wide baseline setting difficult to move. Wide baseline methods were proposed for hand pose estimation in [Schlattmann 2007, Ogawara 2003, Nirei 1996, Ueda 2005, Guan 2007a, de Campos 2006, Tran 2008, Cheng 2006].

2.1.4 Other settings

Some specialized hardware-specific cameras can estimate a depth map of the scene at a short range without the need of any complex computation. Time-of-flight cameras are cameras that create distance data with help of the time-of-flight (TOF) principle. The scene is illuminated by short light pulses and the camera measures the time taken until the reflected light reaches each pixel of the camera. This time is directly proportional to the distance of the reflecting object. The camera therefore provides a range value for each pixel. Until recently available system have been too expensive for wide audience (as an example the SwissRanger^{TM}SR4000

by mesa imaging costs about 9000$). Recently the Israeli developer 3DV Systems announced a time-of-light camera at the cost of at approximately. Base on this technology, Microsoft will equip the next generation of XBox with a real-time full boy tracking system. In [Mo 2006] the depth is obtained using a camera developed by Canesta that allows the acquisition of low-resolution depth images at 30hz using a laser-base technology.

A different approach to obtain a depth map is through the projection of a structured light with a specially designed light pattern on the scene. In such a setting, the depth map can be obtained using only a single image of the reflected light and by performing triangulation. The structured light can be in the form of horizontal and vertical lines, points, or checker board patterns. Among methods using a structure to perform hand pose estimation one can refer to [Bray 2004b, Du 2008, Malassiotis 2008]

Finally, another approach to disambiguate the depth, proposed in the context of finger tracking by [Segen 1999], consists in using casted shadows of the object (the hand in our case) on a planar surface lying in its vicinity. The shadow provides information on the silhouette of the hand seen from another point of view. Extracting the information requires some knowledge about the positions of light source and the surface receiving the shadow. From a computational point of view this approach has some similarities with the wide baseline setting, where a second camera is virtually placed at the position of the light source.

2.1.5 Markers

Employing markers on the hand can help tracking. The use of markers could be considered intrusive but markers yield considerable technical advantages in terms of processing speed. A single ellipsoidal marker has been use in [Usabiaga 2006] to infer the 6D global pose of the hand. In [Kim 2004] white fingertip markers where used under black-light to detect fingertip locations, yielding to much richer set of gestures. In [Dorfmuller-Ulhaas 2001] a glove marked with retro-reflective balls has been used to track the index movements. In [Buchmann 2004] a glove with three markers (the index tip ,the thumb tip and the palm) has been used to ease the capture of grabbing movements between the index and the thumb. In [Holden 1997, Lien 1998, Lien 2005, Lien 1998, Dorner 1994] gloves with more than seven markers have been used to estimated the full 3D hand pose of the hand.

2.1.6 Context

The acquisition framework may also inherit contextual restrictions on the user or the environment. Concerning the environment, a classical approach is to use a uniform or static background in order to facilitate the hand segmentation (some examples are [Lin 2002, Wu 2001, Kuch 1995, Ouhaddi 1999a, Zhou 2005a, Athitsos 2002]). If the background is close to the user, a black background might be preferable because moving shadows due to the user are less perceptible on a dark background.

If the camera is fixed and the background is cluttered but static, one can learn a detailed model of the background that facilitates the hand segmentation. If the lightning is controlled the method does not need to be illuminant-independent. Concerning the user, it is often assumed that the set of hand pose that are to be estimated is restricted. A common restriction is to assure that the palm is parallel to the image plane. The purpose is to avoid out of plane rotations that cause fingers to occlude each other. Restricting the hand pose also helps to reduce the dimensionality of the hand pose space and facilitate the pose estimation problem. Finally, assuming the user is always facing the camera can help to reduce the space of possible hand poses during communicative gesturing such as in sign-language where the set of gesture depends on the observer/interlocutor position. With such a restriction the method does not need to be view-independent.

2.2 Model-based tracking

2.2.1 General principle

Model-based methods rely on an explicit model of the hand that allows, for arbitrary hand poses, to synthesize on-line hand model features. Given an image of the hand, the hand pose estimation consists in searching, among possible hand poses, the hand pose candidate that yield the best match between the synthesized model features and the ones extracted from the image. This search is generally (but not always, see section 2.2.4) formulated as an optimization problem. A scalar measure of the error matching is introduced, and the best hand pose is the one that minimize this error. Due to the high dimensionality of the hand pose space, this search cannot be exhaustive. The search is generally made locally, in the vicinity of an initial guess of the hand pose obtained by prediction from the pose estimated in the previous frames. The search is done by iteratively refining the initial pose or by sampling once a set of poses around the predicted pose (like in Bayesian filtering). Two terminologies are related to the model-based approach: the inverse problem formulation and the generative formulation.

- **Inverse problem formulation**. Model-based methods are sometimes referred as inverse problems. In an inverse problem formulation, one specifies first a (deterministic) function $f(\theta)$ that synthesize features from an hand model given an arbitrary pose parameter θ. The function $f(\theta)$ is defined using a model of the hand and eventually of the cameras. Then observed features O extracted from the image, one aims to estimate the hand pose by solving the inverse problem, that is evaluating the inverse function f^{-1} at O i.e finding θ such that $O = f(\theta)$. Due to noise and modeling errors this problem generally does not have any solution and we have $\{\theta \mid f\theta = O\} = \emptyset$. One can overcome the above mentioned limitations through the introduction of a discrepancy measure ρ between the observed features O and the synthetic features $f(\theta)$. We refer to these measure as the *objective function* or the

matching cost. The estimation of θ is then formalized as the minimization of this measure:

$$\hat{\theta} = \underset{\theta}{argmin}\, \rho(f(\theta), I) \tag{2.1}$$

The lowest potential of this objective function will provide best match between the synthesized model features and the features extracted from the image. This approach conforms to the description of the model-based approach we made.

- **Generative formulation.**

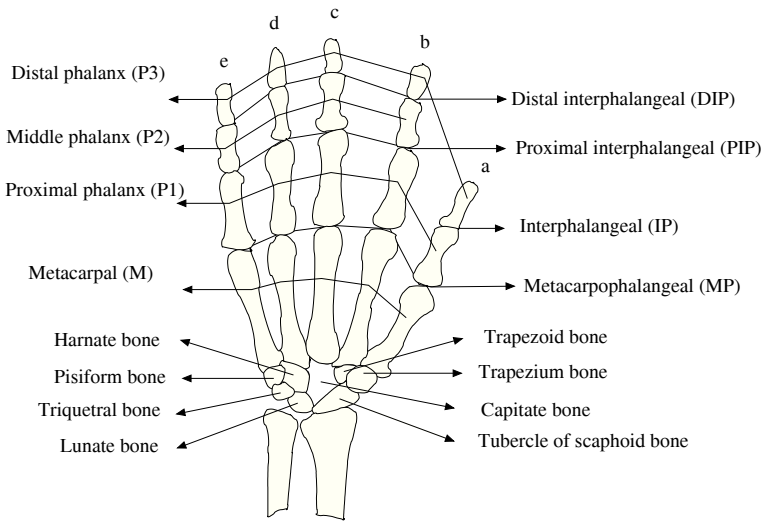
  The choice of the matching objective function might seem somewhat arbitrary, especially if one combine different features and need to weight their relative importance while evaluating the quality of the matching. A principled manner towards a model-based approach is to define a probabilistic generative model, i.e. to define the probability $P(O|\theta)$ with O the observed features and θ an arbitrary set of parameters that describe the configuration of the hand. The probability $P(O|\theta)$ is defined using the model of the hand, its projection onto the camera(s) and the noise measurements. Then, given an observed image, we use $P(O|\theta)$ to estimate the parameters θ. The most common framework driven from this principle is the *maximum-a-posteriori* estimation of θ that done by maximizing the posterior probability $P(\theta|O)$ with respect to θ. The posterior probability $P(\theta|O)$ can be computed up to a constant factor using the Bayes' rule:

  $$P(\theta|O) = \frac{P(O|\theta)P(O)}{P(O)} \tag{2.2}$$

  with $P(O) = \int_\theta P(I|\theta)P(\theta)$ that is constant and thus not taken into account during the maximization. If one does not define any prior on θ one can simply use *maximum likelihood* estimates by maximizing $P(O|\theta)$ with respect to θ. In both cases there no closed-form solution for the maximization and iterative search or Monte-Carlo techniques are considered. Such an attractive generative approach has not been used in the context of hand tracking due to the difficulty of modeling realistic noise on complex extracted features.

Model-based methods are good candidates for continuous tracking over consecutive frames with small or predictable inter-frame displacements. Another intersecting aspect of model-based methods is that multi-view inputs can be handled without solving any correspondence problem between the images. The matching errors with 2D features on all the cameras can simply be summed to define a single error that has to be minimized.

As we will see, the model-based approaches found in the literatures differs mainly on the choice of hand model, the choice of features, the evaluation of the matching, the prediction use to localize the search and the manner the search of the hand pose is made.



Descriptio 2.2: The right hand skeleton from palmar side

2.2.2 Hand models

A 3D hand model consists in a 3D surface that deforms and whose deformation is parameterized by a pose vector. This pose parameterization is often done using underlying skeleton that model the set of kinematic constraints between the bones of the hand. Thus, most 3D hand models comprises i) a model of the skeleton of the hand and its articulations and ii) a model of the hand surface and of its deformation being guided from bones displacements. Few methods do not follow this dual skeleton/surface modeling approach. As an example of such alternative approaches,in [Heap 1996] a linear PCA model of hand surface variations is obtained directly from a data-set of hand images acquired with a magnetic resonance imaging system. We will describe in this section the different skeleton and hand surface models.

2.2.2.1 Kinematic model/parameterization

The skeleton of the human hand comprises 27 bones [Figure (2.2)]. The palm (or metacarpus) comprises 5 bones (one for each finger) and the wrist (or carpus) 8 bones. The 14 remaining fingers are digital bones called phalanx.

Each finger but the thumb comprises 3 phalanxes: the distal phalanx, intermediate (middle) phalanx and proximal phalanx. The thumb as no middle phalanx but its associated metacarpal bone is well separated from the rest of the palm. The bones of the hand are connected together by joints with one or more degrees of freedom.

In the literature, the bones of the palm and the wrist are generally merged into a single rigid part. Few papers [Kuch 1995] uses non-fixed carpometacarpal joints (connecting the metacarpal bones to the wrist) to allow folding or curving of the palm. In [Kuch 1995] two internal Degrees of freedom (DOF) are located at the base of the forth and fifth (ring and pinky) metacarpals. Fixing the carpometacarpal joints might yield not very realistic palm rigidity.

For all fingers other than thumb, the two interphalangeal joints (Distal and Proximal interphalangeal) are modeled with one DOF and the metacarpophalangeal (MP) joint (connecting fingers to the palm) is generally modeled as a saddle joint with two DOFs [Kuch 1995, Dewaele 2004, Stenger 2001c]. Some authors model the MP joints as a spherical joint with three DOFs by adding a twist motion [Bray 2004b]. The thumb has only two phalanges and its single interphalangeal joint is generaly modeled with one DOF. The metacarpophalangeal joint of the thumb is modeled with one [Bray 2004b] or two DOF[Lee 1995, Kuch 1995, Dewaele 2004, Stenger 2001c]. The carpometacarpal articulation, connecting the thumb with the wrist, is the most difficult to model accurately. Biomechanical studies [Hollister 1992] have shown that this joint has two non-orthogonal and non-intersecting rotation axes. This joint has often been modeled as a saddle joint with two orthogonal intersecting rotation axes [Lee 1995, Kuch 1995, Bray 2004b, Dewaele 2004, Stenger 2001c], but this model is not very accurate. Predominant choice consists in 3 DOF spherical joint [Delamarre 1998], that is an over-relaxation of the model. A solution to get a 2 DOF model for this joint is to impose one of the 3 angles of the spherical joint to be linear combination of the two others [Griffin 2000].

To the best of our knowledge - except from our own work - none of the hand tracking paper modeled the forearm.

The pose the hand can be described by the angles associated to each articulations. Furthermore 6 additional parameters should be used to encode the global pose of the palm (3 for its global orientation and 3 for its global position). Depending on the choices made while modeling the joints the total number of parameters varies between 26 and 31 [Ueda 2005]. Describing the global orientation of the palm might be problematic due to the singularities appearing when parameterizing the set of rotation ($SO3$) by 3 angles (see section 3.2.1.2). In order to avoid such singularities the use of quaternions has been adopted to describe the global rotation in [Rehg 1994a].

Rather that describing the hand pose by the angles associated to each articulation, one can describe independently the global pose of each bone [Sudderth 2004]. Such a representation could in general yield to a dislocated hand with parts apart. One deal with the above mentioned limitation through the introduction of non-linear constraints on the parameters associated to adjacent bones (see section 3.2.1.4).

2.2.2.2 Pose constraints

The constraints implied by the kinematic model described in the preceding section are not sufficient to avoid unrealistic hand configurations. The structure of the bones, muscles and the tendons induces limitations on the articulation angles that should be embedded in the hand model. These constrains are essential to (i) avoid unrealistic hand configurations during tracking (ii) reduce the search space. Some structural constraints have been identified "by hand" in the literature [Lee 1995, Kuch 1995, Lin 2000]. Besides structural limitation, more subtle constraints are imposed by the naturalness of hand motions. Such constraints are difficult to identify by hand and have been learned automatically from a data-set of natural hand poses. Most papers consider only active movements (activated by tendons and muscles) and not passive movements (externally forced) for which joints generally have a greater range.

Previous works [Lee 1995, Kuch 1995, Lin 2000] distinguish two type of hand constraints: the *static* and the *dynamic* constraints. *Static constraints* are individual bounds on the range of each joint angles. The limits are fixed (static) and do not depend on other angles of the hand. These constraints are primarily derived from the hand's bone. If the pose is described by the set of joints angles, the set of hand pose that verifies these constraints is a hypercube in the parameters space. This hypercube is the Cartesian product of the individual angles ranges. The space of realistic hand poses is actually smaller than this hypercube. The tendon structure within the hand induces interdependence between fingers angles that are not modeled by static constraints. In order to take these interdependencies into account, a second kind of constraints has been defined in the literature. *Dynamic constraints* are linear or non-linear equalities or inequalities involving two or more joint angles. Linear inequalities involving two joint angles have been specified "by hand" in the literature (see [Lee 1995, Kuch 1995, Lin 2000]). These constraints are referred as *dynamic* because if we fix all angles of the hand with the exception of one, the range of this single angle now depends on some the other joint angles.

Dynamic constraints can be subdivided into intra-finger constraints and inter-finger constraints. The intra-finger constraints are constraints between joints angles of the same finger. As an example, the two interphalangeal joints of any finger other than the thumb are approximatively related by a linear equality (the DIP angle approximatively equals two third of the PIP angle). Inter-finger constraints are constraints between joint angles from different fingers. As an example of such constraints, the extension of a finger is hindered by the flexing of others.

The constraint related to the "naturalness" of poses are not due to structural limitation of the hand. These constraints are context-specific and eventually are difficult to identify. Different approaches have been proposed to learn such constraints from a data-set of natural hand poses we aim to track [Lin 2000, Wu 2001, Zhou 2003, Thayananthan 2003a, Lin 2004]. In most cases, a sub-manifold of the parameters space is learned from the data and the hand parameters vector during inference is constrained to remain within this sub-manifold.

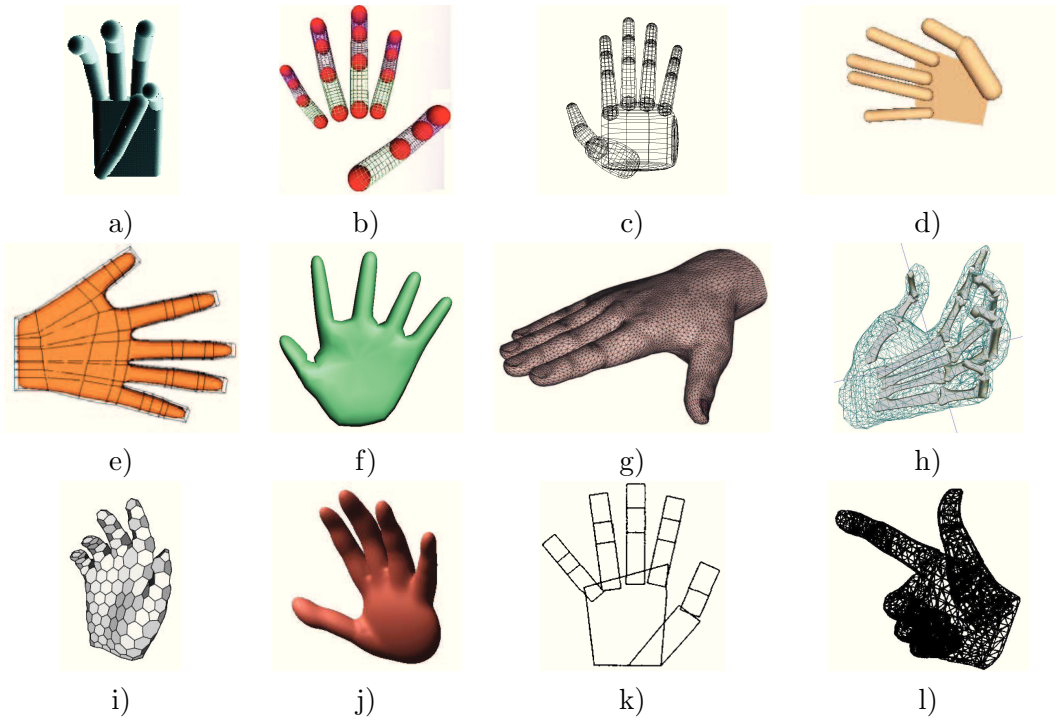
The global hand pose parameters (orientation and translation) are generally not included in the training set, and thus the hand remains free to rotate and translate during the inference. In [Wu 2001] and [Lin 2000] a principal component analysis (PCA) is applied on a data set of hand configurations, each represented by a set of joint angles. In both papers 95% of the variance is said to be captured within the first 7 principal components. They constrain, through re-parameterization of the hand pose, the configuration to lies in the affine space that corresponds to the mean added with any linear combination of the top 7 eigen vectors. Similar to that in [Zhou 2003] the same PCA based approach was used with 6 components. In [Lin 2000] the hand pose is further constrained to be expressed as a convex combination of 28 hand poses within this affine space. In [Lin 2002] the hand pose manifold is defined as the union of 1D manifolds joining each pairs in a set of 28 basis configurations. In [Lin 2004], hand pose estimation is constrained (in the first sage of the search) to be in a finite set of pose samples previously captured with a CyberGlove. Each sample corresponds to one set of joint angle parameter that defines the hand pose.

Whenever the pose estimation is formulated as the minimization of some objective function the naturalness of poses can be imposed using non-uniform penalization rather than constraints [Thayananthan 2003a]. If the method is formulated as a Bayesian method, one can use a non uniform prior on the hand parameters.

2.2.2.3 Hand surface

The second component of a hand model refers to the shape of the skin surface and to its deformation under bones displacement. The actual hand skin deformation involves quite complex elastic deformations. Sophisticated models have been introduced in the computer graphics literature to obtain realistic skin deformations [Sloan 2001, Kry 2002]. In the context of hand tracking, realism is not the main objective. The model has to be synthesized many times per analyzed frame, and therefore a rough model might be preferable to keep the computational load reasonable. Several hand surface models have been proposed in the hand tracking literature.

In [Delamarre 1998, Stenger 2001a, Stenger 2001b, Stenger 2003, Stenger 2004a, Stenger 2004b, Thayananthan 2003a, Lin 2004, Ouhaddi 1999a, Du 2008, Nirei 1996, Lee 2004] the hand surface has been modeled as the union of a small set of truncated quadric surfaces such as cylinders, cones, ellipsoids and spheres. The union of quadric primitives yields a surface that is smooth almost everywhere with the exception of the intersection of the primitives and the borders of the truncation. Conics are easy to project in the camera plane using the perspective geometry results [Stenger 2001c]. Each conic primitive has a fixed size and follows the movements of the bone it is associated with. The surface model that can be obtained using such an approach is not very accurate but computationally efficient. In [Lu 2003] the finger parts are modeled as cylinders and the palm is modeled as a six-rectangle-side-solid. In [Wu 2001, Lin 2002] the



Descriptio 2.3: Hand surface models. Authors names a) Ouhaddi [Ouhaddi 1999a] b) Lee [Lee 2004] c) Stenger [Stenger 2006] d) Lin [Lin 2004] e) Ueda [Ueda 2005] f) [Du 2008] g) Bray [Bray 2004b] h) Ogawara [Ogawara 2003]. i) Heap [Heap 1996] j) Dewaele [Dewaele 2004] k) Wu [Wu 2001] l) Kuch [Kuch 1995]

hand is supposed to be viewed from a direction orthogonal to the palm and the hand surface is modeled using a flat rectangle for each hand parts. This very simple view-dependent model is called *cardboard* and is shown in [Figure (2.3.k)]. The occlusions between parts are handled using a visibility map. A major drawback of this model is that it cannot capture motions with large out-of-plane rotations. In [Dewaele 2004] the hand is modeled as an implicit surfaces defined with meta-balls [Figure (2.3.j)]. The surface is defined as a level surface (a.k.a isosurface) of some a real-valued function of three variables referred as the implicit function. This implicit function is defined using pseudo-distances to ellipsoids, with one ellipsoid associated to each part of the hand. Each ellipsoid has a fixed size and follows the movements of the corresponding bone. The surface that is obtained in [Dewaele 2004] is smooth almost everywhere and the discontinuities of the normal vector are not noticeable. These study makes use of 3D features obtained from stereo vision ands thus eliminate the need to project the model into the image. In [Bray 2004a, Bray 2004b, Ogawara 2003, Du 2008, Chik 2007] the hand surface is modeled using a triangulated surface. Each vertex of the surface is associated to one or more bones. Using the Linear Blend Skinning technique (a.k.a Skeleton Subspace Deformation, see section 4.2.2), each vertex follows the movements of the

associate bone(s). For each pair vertex/bone, a weight allows to control influence of the bones on the vertex displacements. Using progressive transition while specifying the weights, one obtain smooth deformation of the surface around joints of the hand. These studies make use of 3D features obtained using structured light [Bray 2004a, Bray 2004b] or visual hull [Ogawara 2003] and thus eliminated the need to project the model into the image. In [Ueda 2005] a subdivision surface is used to model the hand surface at different resolutions in order to use a coarse to fine approach [Figure (2.3.e)]. In [Kuch 1994, Kuch 1995] the hand is modeled used a B-spline surface with 300 control points. Each of the control point is associated to a bone and follows its displacements [Figure (2.3.l)]. Note that not all model-based hand-pose estimations methods adopt a hand surface. One can cite for example the ones that use high-level features such as finger tip detected in the image [Nölker 1998, Nölker 1999, Rehg 1993, Shimada 1998] or markers [Holden 1997, Lien 1998, Lien 2005, Lien 1998, Dorner 1994]. Another example is the one presented in [Cheng 2004, Cheng 2006], in which a probability density of the hand is modeled in the 3D space using 16 anisotropic Gaussian distributions.

2.2.2.4 Adaptation to user morphology

The hand shape and the lengths of the bones vary among humans. Towards accurate model-based hand trackers, the hand model needs to match the size of the user's hand. This demand becomes more important in a monocular setting. For example, if a finger on the model is too long, its off-plane angle will be over-estimated such that the shortening due to the projection compensate for the over-estimation of the length. The off-plane angle is less likely to be over estimated if the hand is simultaneously observed from another viewpoint. The problem of calibrating the model to the user is usually solved manually in the literature. In [Kuch 1994] the hand is calibrated in a semi-supervised manner. Several landmark points are selected manually on images taken from different view points and the hand spline-based surface is fitted on the landmark points. In [Chua 2000] the index finger size is estimate from an image of the hand open and coplanar with the image plane. The size of each phalanx is estimated based on a fixed ratio between finger segments. In [Lu 2003] a shape correction step is employed in the first few frames to improve the hand shape. It consists in adapting the size of each rigid part using non-isotropic scaling deformations. The parameters are estimated using an iterative method based on physical forces.

2.2.3 Images features

The optimal hand pose corresponds to the model parameters that yield the best match between the synthesized model features and the ones extracted from the image. There are two classes of features, the low-level and high level ones. Low-level features are computed for each pixel using pixels intensities in a small neighborhood of the pixel. High-level features require non-local reasoning to be extracted, that

are generally specific to the kind of data we have and are often expensive to compute. Most model-based method combine different cues to estimate the hand pose. According to [Sigal 2007] "It is a general consensus in the body tracking community that combining features leads to better and more robust inference methods". We will describe some of the more common features used in the literature bellow.

2.2.3.1 Low level features

The simplest features coming from image are the pixels intensities. In [Rehg 1994b, Rehg 1995] pixel intensities are directly compared with those obtained from a model of the hand made of layered 2D templates. In [Rehg 1994b] the square differences between observed and synthesized intensities are summed over all pixels. If a background model is not available the summation is restricted to pixels that originates from one of the templates composing the hand model.

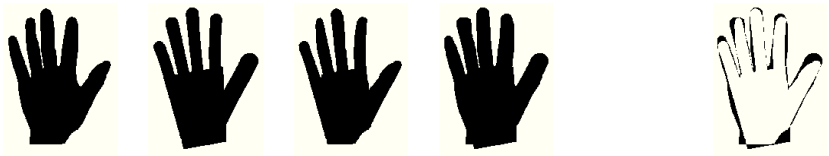
The observed image and the template are filtered with a Laplacian of Gaussian filter that emphases the edges and eventually helps to have results that are less sensitive to shading variations. Edges or contours are features that are used in most (if not all) model-based hand trackers. [Heap 1996, Lin 2002, Lin 2004, Stenger 2001a, Lu 2003, Thayananthan 2003a, Stenger 2001a, Stenger 2004a, Stenger 2004b, Sudderth 2004, Cui. J-S 2004]. An example from [Stenger 2006] of extracted edges is presented in [Figure (2.4)]. The edges in the observed image are compared with the edges that are synthesized from the model in with the hand pose candidates. If the hand surface is in 3D, the synthetic edges are obtained by finding point of the surface where the surface normal is orthogonal to the line passing thought the point and the optical center of the camera (in the case of triangulated surface the normal is not continuous and the definition slightly differs, see section 4.6.1.2). Among these points, the ones that are hidden by an other part of the hand should be discarded (which is not mentioned to be the case in [Heap 1996, Lu 2003]). Each point on the synthetic edge is associated to the nearest edge(s) in the observed image. The distances between associated edges are summed to define a matching measure. The edges orientations yield a valuable information in such approaches. One can (i) search the nearest edges along the direction orthogonal to the synthetic edges, (ii) use the observed edge directions when defining the matching cost [Thayananthan 2003a, Stenger 2001a, Stenger 2004a, Stenger 2004b, Sudderth 2004], (iii) use a 1D gradient filter on the image intensities along the line that is orthogonal to the synthetic edges [Heap 1996]. Edges are features that have the advantage not to be illumination-invariant. Their use is less evident when the background is cluttered, since background edges might be difficult to distinguish from hand edges. Wriggles and sharp cast shadows within the hand may also create edges that are generally not modeled.

The hand silhouette is a other common feature used by hand trackers [Lin 2002, Lin 2004, Wu 2001, Kuch 1995, Ouhaddi 1999a]. The silhouette corresponds to the segmentation of the hand in the observed image(s). The silhouette is more robust to clutter than edges or contours. The silhouette segmentation is generally based

Descriptio 2.4: Edges from [Stenger 2006]

on the assumption that the hand is the only skin colored object in the image. Generally, the background is either assumed to have a uniform color distribution or to be static. Whenever the background is static, a sequence of images can be used to learn a Gaussian per pixel (mean and covariance). The silhouette segmentation is done by comparing, for each pixel, the likelihoods of being part of the hand or the background given the hand color model and the background model (background substraction). Using a Markov-random-field formulation [Howe 2004], active contours or some morphological filters one can improve the quality of the segmentation over independent per-pixel classification.

Once the hand is segmented, the silhouette extracted form the image is compared with the ones that are synthesized with hand poses candidates. The comparison is generally done [Lin 2002, Lin 2004, Wu 2001, Kuch 1995, Ouhaddi 1999a] by measuring the symmetric area difference of the two hand silhouettes (observed and synthetic). The symmetric difference between two regions is the region made of the points that are within one of the two regions but not both.



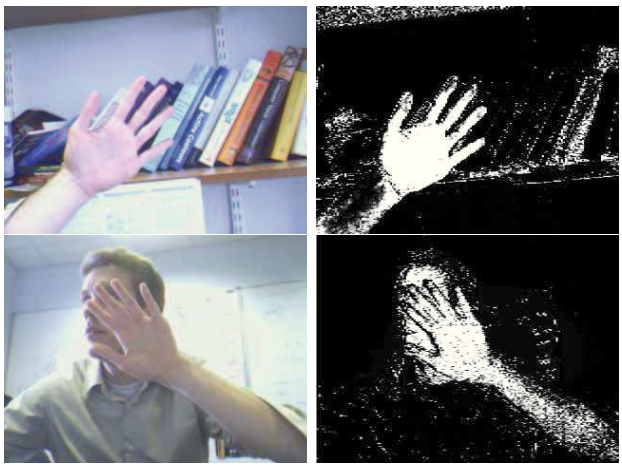
S_1 S_2 $S_1 \cup S_2$ $S_1 \cap S_2$ $S_1 \Delta S_2 \equiv (S_1 \cup S_2)/(S_1 \cap S_2)$

Descriptio 2.5: Symmetric difference between the segmented silhoutte (S_1) and the synthesized silhouette (S_2)

Denoting S_1 and S_2 the two regions, the symmetric difference between the two region writes $S_1 \Delta S_2 \equiv (S_1 \cup S_2)/(S_1 \cap S_2)$. This region is empty whenever the

two silhouette matches exactly [Figure (2.5)]. The area of this region is sometimes referred as the hamming distance. In [Ouhaddi 1999a] the chamfer distance between the two silhouettes is also tested as an alternative matching measure. In [Nirei 1996] the scalar product between the two normalized distance images associated to the segmented silhouette and the model silhouette is used as the matching criterion. In a monocular setting, the segmented hand silhouette generally does not provide sufficient information to capture the position of the fingers that occlude parts of the palm. These feature is generally combined with other features such as edges [Lin 2002, Lin 2004, Wu 2001].

The segmentation of the hand silhouette based on color distributions may be inaccurate, especially in the presence of a cluttered background, due to the lack of strong shape constraints when performing the segmentation. One would like to limit the impact of such segmentation error on the hand pose estimation. A possible solution could be to compute uncertainties on the segmentation and use these uncertainties while comparing the segmented silhouette with the silhouette obtained with the model. Another solution is to compute directly a "matching error" without performing segmentation by summing, over all pixels within the silhouette, the difference between the two log-likelihoods (which is equivalent to logarithm of the likelihood ratio) based respectively on the hand color model and the background color model. For each pixel, the logarithm of the likelihood-ratio is positive if the pixel is more likely to be part of the hand and negative otherwise. This has been done in [Stenger 2006, Sudderth 2004, Stenger 2004a, Stenger 2004b]. An example from [Stenger 2006] of likelihood-ratio image based on the color of the pixels is presented in [Figure (2.6)].



Descriptio 2.6: Hand/background likelihood ratio [Stenger 2006]

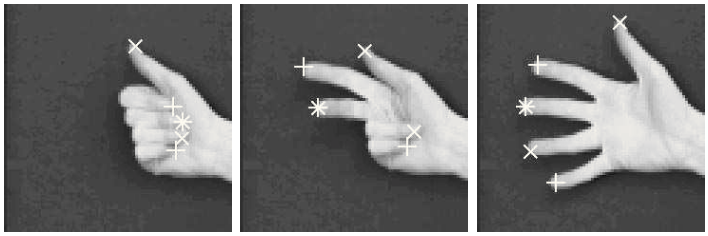
In [Zhou 2003] edges are detected by applying an edge filter on the log-likelihood ratio image which allows to detect the hand silhouette boundaries.

Some motion information can also be extracted from successive images as fea-

tures to estimate the hand displacement. In [Lu 2003, Nirei 1996] the optical flow is used to infer the displacement of the hand model between two successive frames. In [Lu 2003] A shading model is introduced in the formulation of the optical flow equation in order to cope with shading variation when the model displace. In [Dewaele 2004] points-of-interest are detected in the left image and matched with the right image using a correlation criterion in order to obtain a sparse 3D representation of the hand surface. For each frame the detected points-of-interest are matched [1] with points-of-interest detected in previous frame. This set of correspondences between two successive frames can be interpreted as a sparse 3D motion information that is updated at each iteration of the fitting procedure.

2.2.3.2 High level features

High level features carry the advantage of better contextual and global representation of the observation space but are in general computationally more expensive to extract from the image. They present some advantages over low level features: the matching cost for a hand pose candidate is generally much less expensive to compute. This could speed up the pose estimation process once the features are computed, and the resulting matching cost (if formalized) is generally less noisy and have less local minima. On the other hand, reliable high level features are difficult to extract in the general case due to self similarities in the hand and self-occlusions.



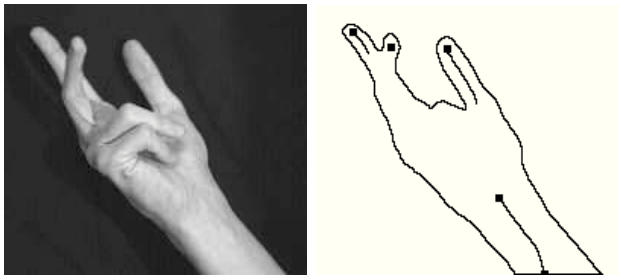
Descriptio 2.7: Detected finger tips in [Nölker 1999]

Finger tips 2D positions are valuable high-level features to estimate the 3D hand pose. Once the finger tips are detected, one can define the matching cost associated to these features to be the sum of the Euclidean distances between the position detected and model tip positions [Nölker 1998, Nölker 1999]. However detecting the tips is in general a difficult problem. In [Nölker 1998, Nölker 1999] a neural network was trained using a feature space derived from Gabor filter [Figure (2.7)]. In order to facilitate detection, the background is discarded by assuming it to be darker than any part of the hand. The performance of this detector is difficult to assess because the space of poses of the training and testing set seems quite limited (the palm interior is oriented to the camera, there is no occlusion of the finger

[1] the term "matching" is actually slightly abusive because the fitting procedure is based on EM-ICP where the matching variable are hidden variable and only their probabilities are computed (see section 2.2.4.2)

tips) and the illumination conditions do not vary in the dataset. In [Du 2008], finger tips are detected as local maxima of the curvature of the silhouette. This is sufficient for their virtual keyboard application where the fingers are extended and well separated. In [Rehg 1993] line and point features are extracted from images taken from one or more viewpoints. These features consist of points representing finger tips and link feature vectors that represent the central axis of each finger segment. Their extraction is done using local feature trackers that inspect locally the image intensities long lines that are nearly orthogonal to each finger. These features are not reliable in case the edges of the finger are hidden by an other part of the hand.

In [Shimada 1998] the protrusion of the hand silhouettes are detected [Figure (2.8)]. Each protrusion is assumed to correspond either to a finger tip or a finger joint. The main axis of each protrusion is computed and its direction is taken into account when defining the matching cost. The wrist position is also extracted from the silhouette by taking the location where the he width of the arm changes abruptly.



Descriptio 2.8: markers [Lien 2005]

In [Segen 1999] the hand is assumed to be nearby a planar white surface. It is also assumed that it is illuminated with a point source that creates a sharp shadows of the hand onto the surface that is well visible in the image. The tip and the main axis of both the pointing finger and its shadow are extracted from the hand silhouette and its shadow. The tips are assumed to be peaks on the silhouette contours where the local curvature is above some predefined threshold. Then, using the calibration parameters of the setting, one can recover the 3D axis aligned with the pointing finger. The same method is applied for an extended thumb.

The challenging aspect of reliable high-level feature detection has resulted into an extensive use of markers or colored gloves in order to facilitate the segmentation. In [Lien 1998], seven colored markers are pasted on the hand: one for each finger tip, one on the palm and one on the wrist. Similarly, in [Lee 1995], seven colored markers are painted on a white glove: one for each finger tip, one for the MP joint of the little finger and one for the wrist. Both [Lien 1998] and [Lee 1995] use a multi-camera setting such that the 3D position of these markers can easily be extracted. In [Lien 2005], seven non-colored markers [Figure (2.9)] are pasted on

the hand. Due to the lack of color, the marker identification process is done using the temporal coherence.

Descriptio 2.9: markers [Lien 2005]

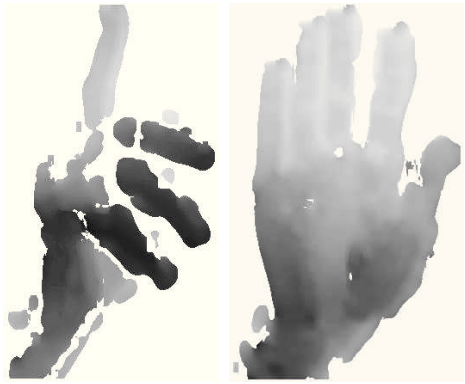
In [Holden 1997] thirteen colored markers are painted on a white glove: two ring markers for each finger (so that markers could still be detected from various viewing angles), two semi-ring markers for the MP joint of the index and the little finger, and one ring marker for the wrist. In [Lien 2005, Holden 1997] a single camera is used and thus the 3D position of the markers cannot be extracted directly. In [Dorner 1994] twenty-one colored markers are painted on a white glove one for each finger joint and tips and one for the wrist. Because it is difficult to find twenty-one different colors that are contrasted enough to allow robust discrimination between the colors, two different methods are proposed for coding the marker with colors. The first uses the same color for different markers (which may create some ambiguities) while the second is based on a combination of two colors around each location to uniquely encode each joint.

2.2.3.3 3D features

Using multi-camera settings or some specific hardware it is possible to obtain depth information about the hand being tracked. We refer to the features that encompass depth information as 3D features. We discussed in section 2.1 different acquisition frameworks that can recover depth information.

Using two cameras in a small base-line stereo setting (2.1.2), the depth can be determined by triangulation after matching corresponding points in the two images. This encompass both the hand and eventually the background geometry. Color information and the assumption that the hand is the closest object to the camera can be used to eliminate most of the points that should not be associated with the hand. Depending on the density of the matched points, sparse or dense stereo reconstruction can be achieved. In the context of hand tracking, sparse reconstruction has been used in [Dewaele 2004] (see section 2.2.3.1) and dense reconstruction in

[Delamarre 1998] [Figure (2.10)]. The hand has generally little texture and therefore reliable matching is not possible, which result in erroneous depth estimates.

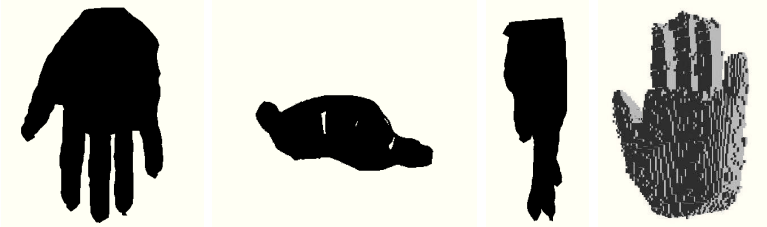


Descriptio 2.10: Reconstructed depth map from [Delamarre 1998]

Structured lights can be used to address the above mentioned limitation and recover the depth map using a single camera. This has been done for model-based hand tracking in [Bray 2004b, Bray 2004a].

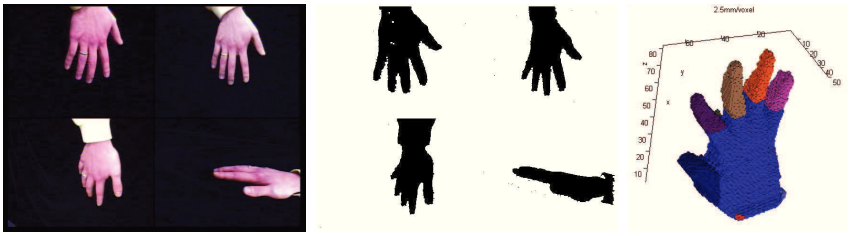
Multi-camera wide-base line setting (see section 2.1.3) can provide a volume called the *visual hull* that enclose the hand driven from the silhouette of the hand in each camera. The visual hull is the 3D volume defined as the set of all 3D points whose projection lies within the hand segmented regions of the images from all cameras. Intuitively one obtain the visual hull by carving parts of the space that are inconsistent with any silhouettes from the different image views. Correct hand segmentations will result to a visual hull that strictly contains the true hand volume. As the number of views grows more volume is carved away and the visual hull is getting closer to the true shape of the hand. Two examples of hand visual hull from [Ueda 2001a] and [Cheng 2004] are presented in [Figure (2.11)] and [Figure (2.12)].

The surface of the visual hull can then be compared with the surface of the



Descriptio 2.11: Visual hull from [Ueda 2001a]

model by matching each point of the model surface to its closest point on the visual hull [Ueda 2005, Ueda 2003, Schlattmann 2007, Ogawara 2003]. Another approach [Cheng 2004, Cheng 2006] consists in defining a hand-pose likelihood function based



Descriptio 2.12: Segmented silhouettes and visual hull from [Cheng 2004]

on a parameterized probability density of the hand using a mixture of 16 anisotropic Gaussian distributions. Finally when several camera and markers are used, the 3D position of the markers can be obtained by triangulation [Lien 1998, Lee 1995].

2.2.4 Fitting procedures

Once the features have been extracted, hand pose estimation consist in searching - among possible poses - the one that yields the best match between the synthesized model features and the ones extracted. This search is often formulated as an optimization problem. A scalar measure of the error matching is introduced, and the best hand pose is the one that minimize this error. When this measure is expressed as a sum of distances between points on the model and nearest feature points, then the minimization can be done using methods similar to the well known iterative closest point algorithm (ICP). Some approaches do not explicitly formulate a matching cost but rather use a metaphor of physical forces to pull or push the model toward the configuration of the model that is compatible with the observed features.

2.2.4.1 Local optimization approaches

Often, the matching error between the features extracted from the image and the ones obtained with the model can be formalized as a scalar objective function. In that case the hand pose estimation reduces to a continuous minimization problem and one can use classical continuous local optimization methods. In [Kuch 1995, Lin 2004, Ouhaddi 1999a, Lin 2004] the minimization is achieved using a local search method that does not require the first order derivatives(gradient) of the objective function. In [Kuch 1995] the matching cost is based on the symmetric difference between the silhouette extracted from the image on the one obtained with the model. The minimization is done using coordinate descent, that does not require the gradient. The set of parameters is updated one component at a time by solving a sequence of one dimensional minimization problems.

In [Lin 2004, Ouhaddi 1999a] the minimization is done using a variant of the downhill simplex method (a.k.a the Nelder-Mead method). This is a local search method that does not require the gradient as well. The function being minimized is

evaluated on the vertices of a simplex (a polytope of $N+1$ vertices in N dimensions) in the pose space. Then at each iteration, the vertex with the highest value is replaced with a point reflected through the centroid of the remaining N points. If the value at that vertex becomes the smallest value, then the simplex is expended otherwise the simplex is contracted. In [Ouhaddi 1999a] the matching cost is based on silhouettes. The minimization is done using a modified version of the downhill simplex method that takes the kinematic constraints into account. This is done by imposing the vertices of the simplex to remain in the the set of poses that verifies the kinematic constraints $A\theta \leq b$.

In [Lin 2004] the matching cost combines edges and silhouette terms. The minimization is done in two stages. The first stage alternates the estimation of the global palm position, using the downhill simplex method, and the estimation of the finger poses using a multi-start (30 particles) local search. This is achieved using the downhill simplex method with the constraint that the vertices of the simplex are taken in a set of sample points collected using a glove-based sensor. This constraint allows encoding kinematic limits of the hand, but imposes certain limits on the precision of the pose estimation. In the second stage, the constraints are relaxed and the simplex can converge towards a precise pose estimate.

In [Rehg 1994b, Rehg 1995, Bray 2004b] the minimization is done using local search methods that exploits the gradient of the function. The convergence rate of such method is faster compared to the ones that do not use the gradient information. In [Rehg 1994b, Rehg 1995] the matching error is defined as the sum of square differences between intensities of the images and the ones of the 2D templates modeling hand parts. The minimization is done using a classical gradient descent. In [Bray 2004b] the visible 3D hand surface is obtained using a structured light and the matching error is defined using an approximation of the chamfer distance between synthetic and measured surfaces. The minimization is done using a stochastic gradient descent. At each iteration, a small number of points (45 to 200) are selected randomly on the model surface in order to obtain a Monte-Carlo approximation of the energy and its gradient. This reduces the computational cost and also helps to avoid spurious local minima. The hand kinematic constraints were carefully taken into consideration by using an additional step at each iteration. The resulting algorithm was called Stochastic Meta Descent (SMD).

In [Chik 2007] the hand is captured with two cameras. The matching error combines a silhouette matching cost and a point stereo-consistency cost. The latter measure, for each point of a randomly chosen subset of point on the surface model, the visual consistency between the two image projections of this point (a criterion often used in stereo reconstruction [2]). A stochastic gradient descent approach using the Robbins-Monro method is proposed with proof of convergence, however the SMD [Bray 2004b] method is used for the experimentation because it appeared to be faster.

[2]Note that it is not said in the paper if and how the points that are hidden in one of the two camera are efficiently discarded

2.2.4.2 ICP like methods

The matching measure between the model features and the ones extracted from the image can often be expressed as a sum of squared Euclidean distances between points on the model and nearest feature points. In that case the minimization can be done using methods similar to the well known iterative closest point algorithm (ICP). The original ICP method iterates between two steps. The first step consists in building correspondences by matching each model feature with the nearest (using Euclidean distance) image feature. The second step consists in re-estimating the model pose by minimizing the sum of squared Euclidean distances between previously matched pairs. By iterating these two steps the method eventually converges to a local minimum of the error matching function.

The main advantage of the ICP method over the gradient descent is that a closed-form solution generally exists for estimating the optimal transformation once the correspondences are build. Last we should mention that ICP in its original formulation exhibits a linear convergence rate [Pottmann 2006].

In [Lin 2002] and [Zhou 2003] the pose estimation is done by alternating global hand pose (rotation and translation) and local finger pose estimation. The global pose is determined using ICP with the finger kept fixed, while the local finger poses are determined using a particle filter in [Lin 2002] (see section 2.2.4.4) and a factored sampling in [Zhou 2003]. In [Lin 2002] each model edges is associated with the nearest one in the image. The camera projection is supposed to be orthographic, and therefore the minimization with respect to the global pose parameters has a closed-form solution. In [Zhou 2003] the direction of the image gradient is taken unto account in order to improve the correspondences. In both papers, the matching criterion used for finger pose estimation slightly differs from the one used for the global pose estimation. The two criterion might compete one against another while alternating finger and global hand pose estimation, and thus their convergence is problematic.

In [Heap 1996] all hand pose parameters are estimated using a modified version of ICP. The correspondences are computed by seeking intensity changes along segments being orthogonal to the edges of the hand model. The pose estimation based on these correspondences is done using a weighted least square method. The use of weights aims at improving the convergence rate and robustness to outliers or false correspondences..

In [Dewaele 2004] a sparse stereo reconstruction of the 3D hand surface is obtained by detecting points-of-interest in the left image and matching them with the right ones. The model is fitted to this set of points using a method so-called Expectation-Maximization Iterative Closest Point and Surface (EM-ICPS). This method is inspired from the EM-ICP introduced in [Granger 2002]. The EM-ICP derives from a probabilistic formulation of ICP where the matches are considered as hidden random variables and the criterion is maximized using Expectation-Maximization (EM). According to [Granger 2002] the EM-ICP is more robust when compared with the conventional ICP.

In [Cheng 2004, Cheng 2006, Tran 2008] a kinematically constrained Gaussian mixture model is fitted to the set of voxel in the visual hull. This is done using an adaptation of the classical EM approach for Gaussian mixtures.

2.2.4.3 Generalized forces

An alternative fitting approach exploits physical forces applied on the model. Using a mechanical metaphor, the model is considered as a rigid body with inertia. The matching error between the model features and the ones extracted in the image is used to define external forces that are applied on the model. Based on the Newton second law or the Euler-Lagrange differential equations, these forces are numerically integrated to displace the model. If the forces are conservative (i.e. can be written as the gradient of some scalar potential function referred as potential energy by analogy to mechanics) and damping forces are introduced to dissipate the kinetic energy (or if the kinetic energy is set to zeros after each iteration as done in [Delamarre 1999]), then convergence of such approaches is guarantied. Note however that, for explicitly defined potential functions, optimization methods exploiting higher-order derivatives - such as Newton method or Quasi-Newton method - are likely to converge faster than the method consisting in applying the laws of mechanic.

In [Delamarre 1998] a 3D model based made of truncated cylinders is fitted to the dense 3D reconstruction of the visible part of the hand. Two kinds of forces are applied on the model. For each point on the reconstructed surface, a force is created to attract the closest model point. For each point on the reconstructed surface that is within the 3D model, a force in the direction of the normal of the reconstructed surface is considered. These forces are inspired by the Maxwell's demons introduced in [Thirion 1996], they are not conservative forces and therefore the convergence of the method is questionable. These forces are integrated using an efficient method for articulated bodies whose complexity is linear to the number of rigid parts. In [Lu 2003] the pose is estimated using a combination of 2D forces that drive the model to detected edges and other forces derived from the optical flow equation. These forces are transformed from 2D forces to 3D forces and integrated to displace a 3D hand model

In [Lee 1995] the fingers and the palm pose are alternatively estimated. The palm pose is estimated using forces that aim to attract the projected model markers toward colored ones detected in the image. The palm is incrementally rotated around the main axis of the resulting torque forces [3] until the torque vanishes. The finger poses are estimated using a classical inverse-kinematic method.

In [Ueda 2001b] the hand model is fitted to the visual hull. For each point of the model that lies outside the visual hull, a 3D forces is created. This force is oriented towards the main axis of the model phalanx it is associated with.

[3] Note that the definition of torque they use differs from the classical definition

2.2.4.4 Non-local methods

Iterative methods based on non-convex local optimization, ICP, or forces are likely to converge to the right pose estimate only for good initial configurations. However, in several applications, the motion of the hand can be very quick or the hand can present large occlusions. In such cases, it might be difficult to obtain a good initial pose from previous frames and the track could be lost. In order to circumvent such limitations methods that aim at exploring a wider region in the parameter space were proposed.

The use of multiple hypothesis testing during the tracking [Lin 2002] [Bray 2004a] using particle filtering is a promising direction to cope with local minima. Pose estimation is formalized using a recursive Bayesian filter where a set of weighted particles approximates the conditional probability density of the pose. Given a new observation, pose candidates are generated from the ones in the previous frame using a perturbation model. The importance associated to the new particles is modified by taking into account the likelihood of the observation in the new frame. This is done by measuring the matching error between the model features and the corresponding image features. In [Lin 2002] particle filters are only used for the hand parameters encoding the finger positions and the global hand rotation and translation are not sampled. In [Bray 2004a] the particle filter formulation is modified to allow local refinement of the hand pose estimates using local search. This allows to concentrate the particles on the peaks of the conditional probability density of the hand pose. The perturbation mode of the particles is very important to avoid degeneration of samples. We can cite for example [Wu 2001] where the set of particles is perturbed using factored importance sampling.

Another popular approach to explore a wide range of hand poses is to use global optimization methods. In [Nirei 1996] the hand pose is estimated in two steps. A rough estimation of the hand pose is achieved by minimizing a criterion based on the silhouette using a genetic algorithm (GA) that allows exploring a wide range of the hand parameter space. Then, the pose parameters are refined by minimizing a criterion based on the optical flow using a simulated annealing method (SA). In [Cui. J-S 2004] the initialization is done using a discriminative method and the hand pose is refined using a genetic algorithm. In [Sudderth 2004] the objective function is expressed using a graphical model. The factorization of the objective function into a factor graph requires each hand part to be parameterized independently from the other parts. Consequently an increase on the degrees of freedom is introduced. That induces the necessity of additional terms to impose the kinematic constraints between pairs of hand parts. Belief propagation, a non local inference technique for graphical models, is used to perform the pose inference. However certain restrictive approximation/assumptions have to be adopted to obtain a factorized form of the energy (in particular while modeling occlusions). Furthermore belief propagation involving high dimension continuous variables is yet an open problem.

Finally in [Stenger 2006, Thayananthan 2003a] a tree-based filter that performs Bayesian particle filter with a adaptive tree-based representation of the conditional

probability is considered. This approach is based on a hierarchical subdivision of the pose space and bears similarities with the branch and bound global optimization method. In [Thayananthan 2003a] the set of voxel is clustered using the Laplacian Embedding technique and the set of clusters is used to provide good initialization for the EM algorithm.

2.3 Discriminative methods / Learning-Based Methods

2.3.1 Principle

Discriminative methods aim at defining a direct mapping from images features to 3D hand pose estimates. The features are extracted either from a single image or multiple calibrated views. This mapping is learned directly from a large dataset of hand poses parameters with the associated image features. The dataset is either generated off-line with a synthetic model or acquired by a camera from a small set of poses. The mapping from the visual inputs to 3D poses is not necessarily a function. Due the occlusions for example, the same hand image could be generated by different hand configurations, which result in visual ambiguities. A one-to-many mapping, where the outputs is a set of possible candidates, is better suited to capture such ambiguities. Discriminative approaches have been proposed in the context of hand tracking in [Guan 2007b, Guan 2006, Athitsos 2002, Athitsos 2003b, Shimada 2001, Athitsos 2002, Athitsos 2003b, Potamias 2008, Stenger 2006, Zhou 2004, de Campos 2006, Rosales 2001, Nölker 1999, Nölker 2002, Nölker 2000].

We distinguish three classes among the discriminative methods: indexing, regression-based and classification-based methods.

- Database indexing methods can generally be interpreted as a simple nearest neighbor search in the database. The estimated hand pose belongs to the finite set of hand poses that are represented in the database.
- Regression-based methods consist in learning a continuous mapping from image features to hand pose parameters. The estimated hand poses is not necessarily represented in the database that is used to learn the mapping.
- Classification-based methods [Ziaie 2009] output a class label among a finite set of semantically meaningful hand poses classes, rather than a set of hand parameters. The number or pose classes is smaller than the set of exemplars in the database. In this thesis we are interested in estimating the hand pose as a set of hand parameters and thus we will not discuss more the classifications-based methods.

2.3.1.1 Discriminative versus Generative methods

Opposite to the generative methods that exploit local search, the discriminative ones do not require pose prediction from previous frame and allow estimating the hand pose from a single frame. One could apply such a discriminative method to

each frame of the video in order to do obtain the temporal hand trajectory. This approach would present a major advantage over the generative ones, based on a local search around some predicted hand pose. One can cite for example lower sensitivity with respect to the initial conditions. On the other hand, even if such a method ensures that poses parameters are not subject to drift (accumulation of error resulting from propagating estimates from frame to frame), it can produces very noisy results.

Unfortunately, due to memory limitation, and computational complexity, none of the existing methods allows yet to estimate accurately and quickly pose parameters for a tracking use. Due to the important dimensionality of the space spanned by possible hand poses (number of hand DOFs), it is not possible to perform dense sampling of the parameter space. As an example, if we discard the parameters that encode the global hand pose relatively to the camera, the set of hand poses was constrained to 26 poses in [Athitsos 2002, Athitsos 2003b] and 125 in [Shimada 2001]. The sampling of the pose space is either uniform but very sparse, which is likely to induce inaccuracy in the pose estimate, or concentrated around a small set of hand poses. Therefore, we can conclude that discriminative methods are well suited for recognition of a small set of predefined hand poses, such as the sign alphabet, or for performing rough (re-)initialization of the hand pose if combined with a generative method.

However, we should mention that the size of the database increases exponentially with the number of DOFs, while such complexity could be amplified due to other factors that influence the hand appearance in the image. If the features are sensitive to some external factors (such as the illumination) the possible variations of these factors have to be well represented in the training data set. The hand size varies across individuals and an erroneous assumption on the hand size yield wrong estimation of the hand. This is particularly the case in a monocular setting. Therefore, one would need to represent different hand sizes in the data-base in order to obtain good pose estimates, which can be very prohibitive. Another limitation for most of the discriminative methods (such that [Athitsos 2002]) is that the performance degrades significantly in cluttered scene where it is difficult to obtain good features. Few discriminative approaches, such as [Athitsos 2003b], are designed to cope with cluttered backgrounds. The generalization of discriminative method to multi view setting is in general difficult. Either the relative positions of the cameras have to remain the same during training and tracking, or the method requires the use of features that are invariant by changes of the relative positions.

Last but not least, discriminative methods refer to a “black box” solution and therefore provide little insight into the problem.

2.3.2 Database indexing methods

Given an input image of the hand, indexing methods consist in retrieving the most similar image(s) from a database by comparing image features. The hand pose parameters of the retrieved image(s) are used to determine the pose of the query image.

This can generally be interpreted as a Nearest-neighbor interpolation method where the output value is simply the value associated to the nearest point in the database. The resulting mapping from the image space to the hand pose one is a piecewise constant function.

The main challenge in these approaches is to find the most similar image(s) in a reasonable amount of time. Different strategies have been proposed to accelerate the search by avoiding exhaustive search and thus to obtain sub-linear computational complexity.

The choice of image features and the one of similarity measure have an tremendous impact in terms of speed and accuracy. There is a variety of methods doing pose estimation through image retrieval, like the ones proposed in [Guan 2007b, Guan 2006, Athitsos 2002, Athitsos 2003b, Potamias 2008, Stenger 2006, Zhou 2004].

In [Potamias 2008], the symmetric chamfer distance between the edges is used as similarity criterion. Such a measure cannot be expressed as an Euclidean distance between two vectors representing image features. As a consequence it is not possible to use fast nearest neighbor methods. Two approximate nearest-neighbor methods are proposed using this non-Euclidean distance. The first one approximates this distance by an Euclidean one through mapping into a lower dimensional space using boost map embedding. Then, a fast methods based on the Euclidean distance is used to quickly find the nearest neighbor in this lower dimensional space. The second one uses some distance-based hashing table to quickly select a small set of candidates and then perform exhaustive search within this small set. Both methods exhibit speed-up of order-two of magnitude over exhaustive search.

In [Athitsos 2002], the ranks obtained using four similarity measures between the input and the database images are combined. These measurements are derived according to edges location, edge orientation histograms, finger locations and the geometric moments of the silhouette. The search in the database is done using two passes. In the first pass the majority of the pose candidates is quickly rejected by combining the two distances that are the quickest to compute. In the second pass, a ranking among the remaining candidates is obtained by combining all similarity measures. The main limitation of this method is that it requires the hand segmentation to be precise in order to obtain reasonable estimates for the silhouette moments and finger locations.

In [Athitsos 2003b] an extension of the previous method is proposed where five similarity measures are combined. The first one is derived from a data-to-input directed chamfer distance while the second one approximates the input-to-data directed chamfer distance using Lipchitz embeddings with 200 reference images. The last three measurements comprise orientation histograms, symmetric chamfer distance and a metric derived from a probabilistic line matching analysis. The search in the database is done using exactly the same two-passes methodology. In comparison with [Athitsos 2002], the similarity measure yields better pose estimates while being more robust to clutters.

In [Guan 2006] a small set of representative hand poses is learned from the database using a Self-Organizing-Map [Kohonen 2001]. Each of the representative poses is associated to a neuron in the SOM [Guan 2007a]. Given the input image the k nearest neighbors are retrieved within the set of representatives. The similarity measure is derived from the depth edges and the silhouette. The depth edges are obtained using a multi-flash camera where flashes are positioned in such a way that shadows are casted along depth discontinuities. These edges are quite reliable and eliminate background clutter. The silhouette similarity is measured using the Hu moments (7 moments) and the Fourier coefficients (top 15) of the silhouette.

In [Zhou 2004] the similarity measure is defined using the Euclidean distance between low dimensional image descriptors. The descriptor is obtained in two steps. In the first step, a set of orientation histograms of the silhouette edges is computed and augmented with their relative image coordinates. In the second step, a small dimensional descriptor is computed by comparing the augmented histograms of the input image with those of a small set of representatives obtained through clustering of the database. Locality sensitive hashing is then used to retrieve efficiently the nearest candidate (based on the Euclidean distance between descriptors).In [Zhou 2005a] the similarity measure combines an okapi with a chamfer distance. A set of patches taken along the silhouette is constructed from the database. The local orientation histogram is computed for each patch and a lexicon is build by clustering them according to their orientation histograms. Given an input image, patches are extracted and the occurrence of each "world" from the lexicon is counted. The similarity measure between the input and an image in the database is defined as a combination of an okapi distance (world occurrence counts in the images) and a chamfer distance (2D location of patches). The search for the most similar images is not done exhaustively but accelerated using an inverted index method inspired from the text retrieval literature.

2.3.3 Regression techniques

The mapping from an image feature vector to hand pose can be interpreted as a nearest-neighbor piecewise constant interpolation in the case of indexing methods.

Consequently when the input image gets further and further from its nearest database exemplar, the error on the pose is likely to increase quickly. The true (but unavailable) mapping from the image to pose space is continuous and thus one can expect to obtain better results using a continuous mapping. This mapping can be learned from exemplars of the database using regression methods [de Campos 2006, Nölker 1999, Nölker 2002] or interpolation methods. To the best of our knowledge, the use of interpolation methods - that should pass trough the training points - has not been considered for discriminative hand pose estimation. This might be due to the fact that the training points are irregularly distributed in the image descriptor (input) space. Opposite to interpolation methods, regression techniques can deal with irregular distribution of the training points. Most regression techniques require the input to lie on a low-dimensional space. Therefore pose estimation based on

regression is generally composed of two steps. First, a descriptor is computed using the image features. Then, the hand pose is estimated using a mapping from the descriptor space to the hand pose space that is learned from the database.

In [Rosales 2001] the image descriptor corresponds to seven Hu moments computed from the segmented hand silhouette. The mapping from the descriptor to the hand pose is learned using the specialized mappings architecture. In [de Campos 2006] the hand is captured from 3 cameras. The silhouette is segmented, and shape context descriptors are computed on points sampled along the silhouette contour. These shape context descriptors are used as basis for a 30 dimensional global image descriptor using a codebook. The descriptors from the 3 different cameras are concatenated into a single 90-dimensional one. The mapping from the descriptor to the hand pose is learned using a regression method proposed in [Agarwal 2006] that is based on Relevant-Vector-Machine [Tipping 2001] and that implicitly performs feature selection. The descriptors are invariant to 2D image rotations and scaling, and therefore results are invariant to camera motion along their optical axis.

In [Nölker 1999, Nölker 2002], the palm is assumed to be coplanar with the image plane and the 2D finger tips position relatively to the hand palm are considered as descriptors. The finger tips are detected using Gabor filters and two neural networks. The mapping from the descriptors to the hand pose is done for each finger individually. By assuming that each finger has only two degrees of freedom (flexion-extension and adduction-abduction), a finger-dependent mapping from the relative 2D tip position to the 2D angles is learned using a self-organizing map. The main limitation of this system is that it assumes that the hand palm is coplanar with the image plane and that the background is easy to subtract. This method was extended in [Nölker 2000] to account for stereo information where two small cameras are used in order to obtain the 3D position of the finger tips using triangulation.

2.4 Other approaches

Let us now review methods that present similarities with the indexing or the regression ones but cannot be classified to either of them.

[Stenger 2006] proposes a combination of hierarchical detection with Bayesian filtering. The output of the Bayesian filtering is a piecewise constant approximation of the posterior probability over the entire pose space. This posterior is computed by combining the probability obtained in the previous frame with a model of the hand pose transition and the observation likelihood associated to the current frame. The pose parameter space is partitioned by recursively splitting regions with the highest posterior probability density. For each partition the observation likelihood is determined through the evaluation of the pose corresponding to the region center.

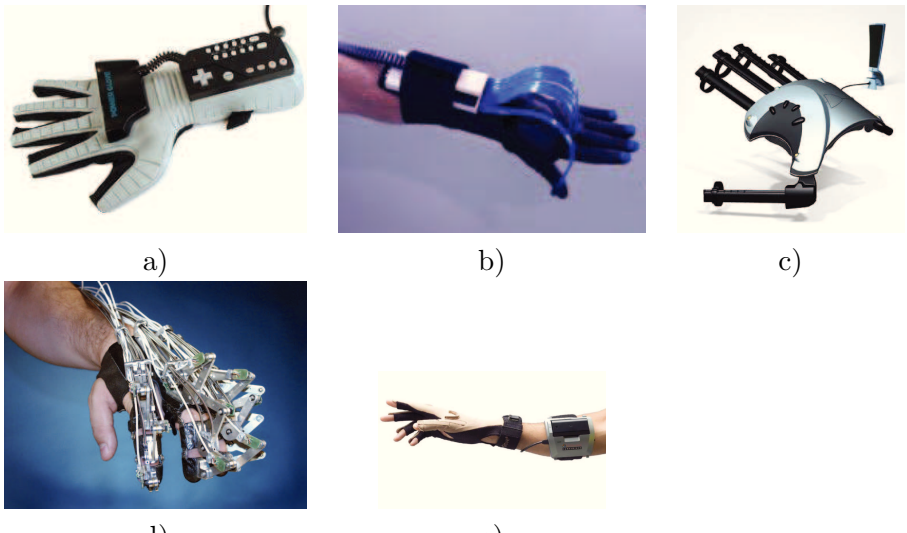
This observation likelihood combines edges and color. Edges similarity is measured using a quadratic chamfer distance function while for color the hand-like likelihood is considered according to the hand color distribution. The partition

centers that are obtained recursively are pre-computed and organized in a tree. Sub-trees are not explored when the hand image associate to the root of the branch is not similar enough to the input one. This approach appears to be quite similar to nearest neighbor search that uses a tree structure in order to quickly prune full branch of candidates.

In [Shimada 2001] hand pose estimation is done in two phases. In the first phase the search is constrained among pre-computed poses in a database, that is somehow similar to indexing methods. However, by using an adjacency map in the database, the search area is reduced to the neighborhood of a small set of the estimated pose in the previous frame and therefore global optimality is not guarantied. The mapping from image to pose is thus not well defined because it depends on the set of estimated poses in the previous frame. In the second pass the hand pose is refined using a generative approach where a model is used to synthesized new images at each iteration

2.5 Related literature

2.5.1 Non-optical systems



a) b) c)

d) e)

Descriptio 2.13: Data glove and exoskeleton: a) the power-glove b) the shapeHandPlus c) the P5 glove d) TU Berlin exoskeleton e) The cyberGlove II

Several efficient hand motion capture systems that are not based on images measurements exist. We can cite for example physical measurements using electromechanical or magnetic sensors. Data gloves offer probably the most reliable hand pose measurements. Equipped of electromechanical captors and magnetic sensing devices, these gloves can digitize in real-time and the fingers joint angles

and the global position of the hand. On the other hand these devices are intrusive and thus hinder the naturalness of hand motion while being quite expensive. Furthermore, they require complex calibration procedures toward precise measurements and generally deteriorate quickly with extended use. Several virtual reality data gloves are evaluated in [Koji 2002]. The CyberGlove II (by Immersion) allows to measure 22 joint-angle with high-accuracy (0.5 degree). The P5 Glove (by Essential Reality) and the ShapeHandPlus (by Measurand) allow measuring only 5 finger flexion angles (one per finger) and the global position of the hand.

Exoskeletons, such as the one developed at the TU Berlin [Figure (2.13.d)], are an alternative to data-gloves that provide mechanical tracking of user limbs, by aligning mechanical linkage between bone joints. Exoskeleton are typically heavy (can weigh up to 12 ounces) and consequently unstable, especially when the hand is shaken or moves rapidly. An a positive note exoskeletons can provide haptic feedback. According to [DiZio 2002] cutaneous contact cues "contribute to perception and adaptation of limb position and force".

2.5.1.1 Body tracking

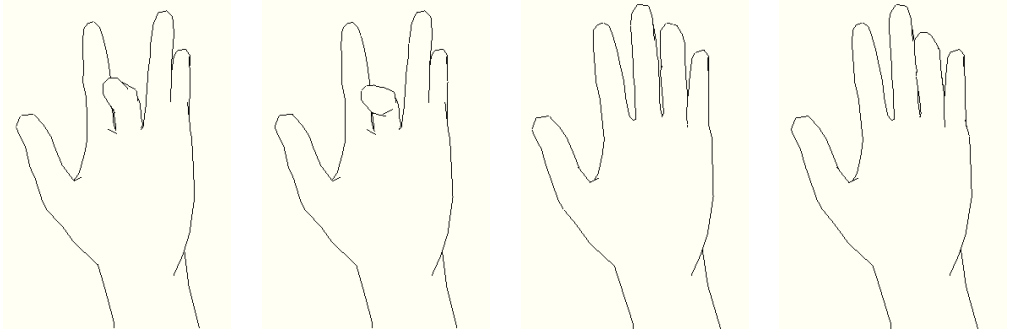
The problem of tracking the human body is similar to the one of tracking the hand. Human body tracking is a more popular research field than hand tracking and several techniques that have been developed for it could be applied to hand tracking. The review of body tracking systems is out of the scope of this thesis and we refer the reader to the following surveys [Weingaertner 1997, Aggarwal 1999, Moeslund 2001, Moeslund 2006]. Despite their similarity, human body tracking and hand tracking differ in several crucial aspects. Clothing induces unpredictable variations of the body shape and appearance. On a positive note clothes make color or texture features reliable for tracking and often allows to discriminate body parts based on the color. The human hand has a uniform color which does not allows separating or identifying hand parts based solely on color. The human body is often vertical which facilitates the identification of the legs and the head. Due to the higher inertia of the human body in comparison with the hand, the human body motions are usually slower than hand motions. The set of typical human motion (walking and running and sitting) seems more restricted than the set of typical hand motion. For this reason, predicting body motion seems to be easier than predicting hand motion.

2.6 Limitations of existing methods

None of the methods that have been proposed in the literature allows to perform robust and accurate real-time tracking of free hand movements in a monocular setting using regular computers. Methods that use multi-cameras seem more efficient but generally cannot be adapted to the monocular setting. Discriminative methods are not suitable for general hand tracking since they are able to estimate only a very restrictive set of poses. Model-based approaches also suffers from several limitations

that are related either to the image features or the definition of the matching errors, or due to the strategy that is used to minimize the matching criterion. Let us first discuss issues related with the feature space and the definition of the matching cost.

- **Absence of shading models**. In a monocular setting the depth ambiguities are a major cause of tracking failure. The information conveyed by the shading can help to disambiguate between hand poses hypotheses. Nevertheless, it is interesting that shading has not been used widely for hand tracking or body tracking (but see [Lu 2003, Balan 2007]). One reason is that shading require an accurate model of surface shape that is difficult to obtain. Another reason might be that the error measures based on the shading are quite expensive to compute in comparison with edge-based or color-based error measures.



a) b) c) d)

Descriptio 2.14: Synthetic edges. A small change on the pose may induce abrupt changes in the synthetic edge image. When comparing a) and b) we observe new edges appearing on the middle finger. When comparing c) and d) we observe that the edges at the right of the middle finger are suddenly occluded by the ring finger

- **Discontinuities due to new boundary edges**. Matching costs based on the symmetric chamfer distance between edges may present discontinuities when the candidate hand pose varies. This is illustrated in [Figure (2.14.a&b)] where a small change on the middle finger induces the sudden appearance of new edges, and in [Figure (2.14.c&d)] where a small change on the ring fingers induce the sudden occlusion of a large fraction of the middle finger edges. This respectively creates a discontinuity and a sharp variation in the matching cost that may cause the tracker to fail.

- **Discontinuities due to the occlusions**. Model-based methods that take into account the self-occlusions generally rely on a formalization that introduce discontinuities in the matching error (see section 4.4.4).

- **Cues weighting**. Model-based approaches generally use a combination of cues (edges, silhouette etc) while defining the matching cost. The costs associated to the cues are summed with a set weights controlling their importance.

Finding the optimal set of weighs might be difficult and the use of not-reliables feature might actually deteriorate the quality of the poses estimates.

- **Lack of appearance information**. The hand albedo is not perfectly constant. Wrinkles and other small asperities create small albedo variations. If the albedo of the hand and its fine variations are known this can be used to provide dense information to register the model onto the image. This dense information has not been explicitly used for model-based hand tracking.

Let us now discuss limitations being associated with the inference procedure:

- **Local minima**. None of the existing model-based methods guaranties global optimality of the solution. Non-local method have been proposed but the dimension of the pose space is too high for these method to be really efficient.
- **Convergence problems due to the occlusions**. Common occlusion handling induces discontinuities in the matching error function. This discontinuities could preclude the convergence of the fitting procedure (see section 4.4.4).
- **Low convergence rate**. Model-based methods based on local optimization or generalized forces have low convergence rates. In a monocular setting, the matching error being minimized is not well conditioned and the convergence rate of gradient-free or simple gradient descent methods is low in comparison with Newton or quasi-Newton methods. Method derived from the Iterative-closest-point (ICP) might have better convergence rates.

ï»¿

Caput 3

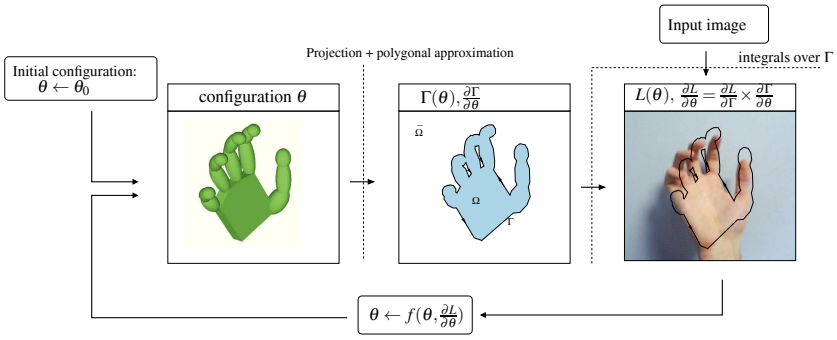
Silhouette Based Method

In this chapter, we propose a model-based approach to recover 3D hand pose from 2D images. To this end, we describe the hand structure using a compact 3D articulated model and reformulate pose-estimation as a binary image segmentation problem aiming to separate the hand from the background. We propose generative models for hand and background pixels leading to a log-likelihood objective function which aims to enclose hand-like pixels within the projected silhouette of the 3D model while excluding background-like pixels. Segmentation and hand pose estimation are jointly addressed through the minimization of a single likelihood function. The pose is determined through gradient descent in the hand parameter space of such an area-based objective function. Furthermore, we propose a new constrained variable metric gradient descent and a trust-region method to speed up convergence. Finally we use the so called *smart particle filter* to deal with occlusions and local minima through multiple hypotheses. Promising experimental results demonstrate the potentials of our approach.

3.1 Method overview

The hand is modeled as an articulated body made of ellipsoids and polyhedra. Limit constraints are defined on each degree of freedom in order to avoid unrealistic hand poses. The model is fitted onto the observed image by minimizing, through gradient descent, a matching cost that quantifies the discrepancy between the projected model and the observed image. The matching cost we propose is derived from a Bayesian formulation according to generative models of the color distribution of the hand and the background pixels. These distributions are used to compute, for each pixel, the likelihood ratio of being part of the hand or background. In order to evaluate the matching cost associated to pose candidate, we first synthesize the hand silhouette by projecting each part in the image plane and computing the union of these projections. Then we evaluate the matching cost by integrating the logarithm of the likelihood ratio based on pixel colors over all points within the projection silhouette.

Assuming the image support to be a continuous subset of $\mathbb{R}^2$ and the observed image to be interpolated between discrete location using nearest neighbor or bilinear interpolation, the matching error is is defined using continuous integrals over points within the projection silhouette. This matching cost is a continuous and differentiable function and we derive the expression of the corresponding gradient. The estimation of the exact value of the matching cost and its gradient is a difficult



Descriptio 3.1: Overview of the hand pose refinement through iterative minimization. Given a candidate hand pose with parameters θ, we compute the silhouette $\Omega(\theta)$, its contour $\Gamma(\theta) \equiv \partial\Omega(\theta)$ and its first order variation $\frac{\partial \Gamma(\theta)}{\partial \theta}$. The pose likelihood $L(\theta)$ is computed by integrating, within the silhouette $\Omega(\theta)$, the pixel likelihood ratio obtained from the input image using the color distribution models. Using the first order variation of the hand pose likelihood $\frac{\partial L(\theta)}{\partial \theta}$, an update of the hand pose is computed using some operator f

task due to the fact that the silhouette is partly made of ellipsoids. This is dealt with through the approximation of the ellipses by polygons toward the definition of an approximate matching cost that we refers as the *polygonal matching cost*. It can be numerically evaluated precisely (up to round-off errors) and yet is a continuous and differentiable function of the hand pose parameters. We derive the exact gradient of the *polygonal matching cost* and a connection is made with the equation of *active polygons* derived in [Unal 2005]. We devise an new efficient algorithms to evaluate exactly both the polygonal matching cost and its gradient.

The hand pose estimation is done by iteratively minimizing the polygonal matching cost using its exact gradient. One can see a graphical overview of the iterative minimization method for hand pose refinement in [Figure (3.1)]. We propose a new constrained variable metric gradient descent and a trust-region method to improve the convergence rate over steepest descent methods. This local search method is combined with a particle filter in order to deal with occlusions and local minima through multiple hypotheses.

Our method bears some similarities with previous approaches [Lin 2002, Lin 2004, Wu 2001, Kuch 1995, Ouhaddi 1999a] which have considered the segmented hand silhouette or edges as image cues. However such cues may be inaccurate, especially with a cluttered background, due to the lack of strong shape constraints. Using directly the likelihood ratio of the pixel, we avoid segmenting the silhouette before matching the model. Our approach somehow unifies the segmentation and hand pose estimation through the minimization of a single matching cost. A similar matching cost based on likelihood ratio has been used in combination with edges-based costs in [Stenger 2006, Sudderth 2004, Stenger 2004a, Stenger 2004b], but these methods do not make use of the gradient of the matching cost to perform

fast-converging local search.

In regards with limitations of previous method we listed in section 2.6, our method:

- Avoids the use of chamfer distance maps from detected edges that may create spurious minima in the vicinity of the global minima. Furthermore, it does not require edges extraction and silhouette pre-segmentation.
- Uses a single term in the definition of the matching cost and thus avoids the problem of weighting different terms. The matching cost directly derives from a Bayesian formulation and thus is principled.
- Relies on a fitting procedure that converges since it is obtained through a gradient descent on a continuous function that can be evaluated numerically.
- Uses a variable metric or a trust-region method to gain speed over steepest descent

On the other hand, it exhibits two main limitations. Our method

- Ignores internal edges and therefore cannot distinguish to hand poses with the same silhouette. In some configurations, the positions of the fingers that occlude the palm might not have any influence on the silhouette and thus cannot be estimated.
- Ignores shading and dense information provided by small albedo asperities.

In the remainder of this chapter, first we describe in detail the hand kinematic structure, the surface model and its projection onto the image plane. Then, we will introduce the matching cost and discuss numerical difficulties due to the use of ellipses. Then a *polygonal matching cost* that approximates each ellipse in the silhouette by a polygon will be considered to address this numerical difficulties. We will present a numerical method to compute exactly this *polygonal matching cost*. Finally will present experimental results and compare to the method that is based on an inexact evaluation of the gradient.

3.2 Articulated model

Let use now introduce some mathematical formalism to describe the structure of articulated bodies in general.

3.2.1 Forward kinematic

3.2.1.1 Solid reference frames

We assume the system to be made of N rigid parts called links (or bones). We associate a rigid reference frame to each part. The reference frame is defined by

three orthogonal unit vectors $x_i, y_i, z_i \in \mathbb{R}^3$ and an origin $o_i \in \mathbb{R}^3$ We define R_i the 3 by 3 matrix obtained by taking the three unit vector $\vec{x_i}$,$\vec{y_i}$ and $\vec{z_i}$ as columns.

$$R_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}$$

The matrix R_i is in SO_3 which is the special orthogonal group of degree 3 over the reals (i.e the set or orthogonal matrices with determinant equal to 1). We can associate a 4x4 matrix K_i to each solid

$$K_i = \begin{bmatrix} \vec{x_i} & \vec{y_i} & \vec{z_i} & o_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_i & o_i \\ 0_{1\times 3} & 1 \end{bmatrix}$$

With $0_{1\times 3}$ the 1 by 3 zero matrix. Let denote $\mathbb{K}$ the set of such transformation matrices:

$$\mathbb{K} = \{K | K \in M(4,\mathbb{R}),\ K_{[1:3,1:3]} \in SO_3,\ K_{[4,1:3]} = 0_{1\times 3},\ K_{4,4} = 1\}$$

with $M(4,\mathbb{R})$ being the set of 4×4 matrices with entries in $\mathbb{R}$. $\mathbb{K}$ is a 6-dimensional sub-manifold of $M(4,\mathbb{R})$. By construction we have $K_i \in \mathbb{K}$.

Furthermore, the matrices K_i allows to obtain the homogeneous coordinates $y \in \mathbb{R}^4$ (homogeneous coordinates are obtained from 3D coordinates by adding 1 for the forth coordinate) in the world reference frame from its homogeneous coordinates $x \in \mathbb{R}^4$ in the solid frame:

$$K_i x = y$$

The matrices K_i are invertible and we have:

$$K_i^{-1} = \begin{bmatrix} R_i^t & -R_i^t o_i \\ 0_{1\times 3} & 1 \end{bmatrix}$$

The local coordinates x_j of a point in the reference K_j can be obtained from its local coordinates x_i in the reference K_i through

$$x_j = K_j^{-1} K_i x_i$$

This comes directly from $K_j x_j = K_i x_i$.

3.2.1.2 Articulated body pose parameterization

In an articulated body the parts do not move independently. Each part is attached to a least another part and their relative positions are constrained. In a tree structured articulated body each part is attached to a father and its reference frame is expressed using the father's reference frame and some transformation.

One can define a tree structured articulated body as a tree[1] made of N nodes with a matrix-valued function of vectors associated to each edge. We denote $G = (V, E)$ the tree where the nodes $V = \{1, \dots N\}$ represent the solid parts (i.e. a bones), and the edges E represent the articulated joints between them. The tree

[1] a tree is connected graph with no cycles

G is oriented from the root to the leaves. Without loss of generality, we suppose the bones are indexed such that $i < j$ for all $(i, j) \in E$. The bone with index 1 is the root. For each node $j \in \{2, \ldots, N\}$ we denote $p(j) \in V$ its ancestor in the tree i.e. the unique element in $\{i|(i, j) \in E\}$. We parameterize the pose of each bone $j \in \{2, \ldots, N\}$ with respect to its father $p(j)$ using a matrix valued function F_j which writes :

$$F_j : \begin{array}{ccc} \Theta_j & \rightarrow & \mathbb{K} \\ \theta_j & \mapsto & F_j(\theta_j) \end{array} \tag{3.1}$$

Where Θ_j is the input parameter space of the function F_j. Θ_j is a manifold of dimension d_j embedded in $\mathbb{R}^{d_j^*}$ with $d_j^* \geq d_j$. In practice we have $d_j^* = d_j$ for all $j \geq 2$ in our model. The parameter $\theta_j \in \Theta_j$ parameterizes the relative position of parts $p(j)$ and j:

$$K_j(\theta) = K_{p(j)}(\theta) F_j(\theta_j) \tag{3.2}$$

The matrix $F_j(\theta_j)$ is defined as a combination of parameterized rotation matrices and a translation matrix (see equations 3.22,3.21, 3.24 and 3.25) where the type of transformation matrices specifies the nature of the articulation. The dimension of the manifold Θ_j - denoted d_j - corresponds to the number of degrees of freedom (DOF) associated to the joint between $p(j)$ and j.

Given a set of parameters $\theta = (\theta_j)_{j \in \{2,\ldots,N\}}$ and the pose matrix K_1 associated to the part that corresponds to the root of the tree, the position of the N-1 remaining bones are obtained using eqn.3.2 from the root to the leafs. We assume that the root of the articulated object can freely move and rotate in the three directions. For notation convenience we suppose the matrix K_1 to be parameterized by $\theta_1 \in \Theta_1$ with Θ_1 a manifold of dimension $d_1 = 6$ (as $\mathbb{K}$ is of dimension 6) and write $K_1 = F_1(\theta_1)$. We choose $\Theta_1 = Q \times \mathbb{R}^3$ with Q the set of unit quaternions. The set of unit quaternions is topologically isomorphic to the sphere S_3 and can be seen as three dimensional sub-manifold of $\mathbb{R}^4$ i.e. $d_1 = 6$ and $d_1^* = 7$. We could parameterize the rotational part of the matrix using Euler but this could lead to singularities [2] that may cause the search to stop at saddle points that would not exist with an other parameterization of the rotations.

We define the internal parameter vector θ_{int} to be the concatenation of the parameters associated to the joints $\theta_{int} = (\theta_2, \ldots, \theta_N)$ and the internal parameter space space $\Theta_{int} = \Theta_2 \times \cdots \times \Theta_N$. Similarly we define the total parameter vector to be $\theta = (\theta_1, \ldots, \theta_N)$ and the total parameter space to be $\Theta = \Theta_1 \times \cdots \times \Theta_N = \Theta_1 \times \Theta_{int}$ Furthermore we define as the *forward kinematic function* f_K, the one that computes the position of the N rigid parts:

$$F_K : \begin{array}{ccc} \Theta & \rightarrow & \mathbb{K}^N \\ \theta & \mapsto & (K_1(\theta), \ldots, K_N(\theta)) \end{array} \tag{3.3}$$

[2] A continuous parameterization of the matrix K_1 is singular around θ_1 if the first order derivative of the matrix with respect to one of the vector components equals θ_1 $0_{4\times4}$ at θ_1 i.e. if the representation of the matrix K_1 is not unique for the chosen parameterization

This function is defined by applying the equation 3.2 from the root to the leaves. It can be implemented using this algorithm 1.

Algorithm 1: Forward kinematic function F_K

Data: $\theta_1, \ldots, \theta_N$
Result: $K_1, \ldots, K_N$
$K_1 \leftarrow F_1(\theta_0)$;
for $j \leftarrow 2$ **to** N **do**
  $i \leftarrow p(j)$;
  $K_j \leftarrow K_i F_j(\theta_j)$;

We denote D the dimensionality of manifold Θ. D is the sum of the number degrees of freedom associate to each joint plus the dimension of the root pose parameters i.e

$$D = \sum_{i=1}^{N} d_i.$$

Θ is a D-dimensional manifold embedded in $\mathbb{R}^{D^*}$ with $D^* = \sum_{i=1}^{N} d_i^*$. Note that, in the literature, the root is fixed in the world coordinate system and the forward kinematic function is generally defined as a function that provides only the position of the *end effectors*. These *end effectors* are points attached to the extremities of the kinematic tree that can be used to reach some desired target. In the context of tracking we do not have such end effectors and the positions of all visible links are of interest in order to compute the matching cost.

The image of Θ by f_K, $f_K(\Theta)$, is a D-dimensional sub manifold of $\mathbb{K}^N$ and by construction the mapping f_K is an explicit parameterization of this manifold. Following [Demirdjian 2003], we will call this manifold the *articulated motion space* or *articulated motion manifold*.

3.2.1.3 Displacements in the parameter space

In order to iteratively estimate the hand pose, we need to perform small displacements in the space of pose parameters Θ. For this purpose, we define a *retraction*, denoted S, on the manifold Θ. The function s writes

$$S : \begin{array}{ccc} \Theta, \mathbb{R}^{D^*} & \rightarrow & \Theta \\ (\theta, \Delta_\theta) & \mapsto & \theta^* \end{array} \tag{3.4}$$

Where the parameter vector θ^* corresponds to the position that is reached while moving within the manifold Θ from θ in the direction $\Delta_\theta / \|\Delta_\theta\|$ with a step length $\|\Delta_\theta\|$ (we refer the reader to [Absil 2008] for a detailed explanation of the concept of *retraction*). If Θ was a linear manifold we could use $\theta^* = \theta + \Delta_\theta$. However Θ is not linear due to the use of quaternions to describe the global orientation of the hand and $\theta + \Delta_\theta \notin \Theta$. For all bones $i \geq 2$ we have Θ_i linear and we can use:

$$\theta_i^* = \theta_i + \Delta_{\theta_i} \tag{3.5}$$

The case of the root bone ($i = 1$) is slightly more complex. We use a unit quaternion to parameterize R_1, the rotation part of $K_1 \equiv F_1(\theta_1)$. During the search of the best pose candidate, this quaternion has to remain a unit one because we want $\theta_1^* \in \Theta_1$. The simplest solution is to reproject $\theta_1 + \Delta_{\theta_1}$ on the manifold Θ_1 by normalizing the quaternion after the displacement. A better solution consists in using *exponential maps* (we refer the reader to [Absil 2008] for more details). For notation convenience we decompose θ_1 and Δ_{θ_1} into $\theta_1 = [q_{1:4}, t_{1:3}]$ and $\Delta_{\theta_1} = [\Delta_{q_{1:4}}, \Delta_{t_{1:3}}]$. Then, we have:

$$F_1(\theta_1) = K_1 = \begin{bmatrix} R_1 & t_{1:3}^T \\ 0_{3\times 1} & 1 \end{bmatrix} \tag{3.6}$$

We assume that the trajectory of the rotational part of K_1, denoted $R_1'(t)$ follows a geodesic path on the manifold of the rotations matrices denoted $SO(3)$:

$$R_1'(t) \equiv R_1 \exp(At) \equiv R_1 \sum_{k=0}^{\infty} \frac{1}{k!}(At)^k \tag{3.7}$$

where *exp* corresponds to the matrix exponential and A a skew matrix (antisymetric). We want the initial "speed" along the path to match the "unrestricted" (displacement $\sum_{l=1}^{4} (\partial R_1/\partial q_i)\Delta_{q_i}$ that is tangential to the manifold) i.e.:

$$\left.\frac{\partial R_1'}{\partial t}\right|_0 = \sum_{i=1}^{4} \frac{\partial R_1}{\partial q_i}\Delta_{q_i} \tag{3.8}$$

By differentiating eqn.3.7 we obtain

$$\left.\frac{\partial R_1'}{\partial t}\right|_0 = R_1 A \tag{3.9}$$

Therefore we can identify the matrix A to be:

$$A = R_1^T \left[\sum_{i=1}^{4} \frac{\partial R_1}{\partial q_i}\Delta_{q_i}\right] \tag{3.10}$$

We can verify that A is a skew matrix by differentiating $R_1^T R_1 = I_{3\times 3}$ with respect to q. The rotation matrix R_1^* is obtained by evaluating $R_1'(t)$ at time $t = 1$:

$$R_1^* = R_1'(1) = R_1 \exp(A) \tag{3.11}$$

Finally we can return to the quaternion representation by evaluating the inverse function of F_1:

$$\theta_1^* = F_1^{-1}\left(\begin{bmatrix} R_1^* & (t_{1:3} + \Delta_{t_{1:3}})^T \\ 0_{3\times 1} & 1 \end{bmatrix}\right) \tag{3.12}$$

An advantage related with the use of a geodesic path is that the displacement on the manifold of rotation matrices SO_3 is made invariant of its parameterization. Consequently the use of other parameterization of the rotational part of K_1 (using the Euler angles or the Rodriguez representation) would provide the same sequence

of matrices K_1 during the iterative minimization. We refer the reader to [Absil 2008] p 58 for a discussion on several retractions on the orthogonal group SO(3). Note that, even if we do not use a quaternion to parameterize the relative position of the bones, we can use a "geodesic path" on the manifold SO(3) if a joint has two or more degrees of freedom. The main drawback is that limit constraints on joint angles would be harder to take into account while computing a displacement Δ_θ within the local optimization method (while the constraint $(\theta + \Delta_\theta) \leq b$ is sufficient if we use $\theta_i^* = \theta_i + \Delta_{\theta_i}$, see equations 3.173 and 3.181).

3.2.1.4 The articulated motion manifold using nonlinear constraints

The vector pose parameter θ of the articulated object can also be represented using directly the set of matrices $K_1, \ldots, K_N$. However, an arbitrary choice of matrices $(K_1, \ldots, K_N)$ is very likely to yield to a configuration that does not belongs to the *articulated motion space* $f_K(\Theta)$ and cannot be obtained with any vector pose parameter. The condition $(K_1, \ldots, K_N) \in f_K(\Theta)$ can be expressed using the joint functions F_{ij}:

$$\begin{aligned}(K_1, \ldots, K_N) \in f_K(\Theta) &\Leftrightarrow K_1 \in \mathbb{K} \text{ and } \forall j \in \{2, \ldots, N\}, \exists \theta_j \in \Theta_j : K_j = K_{p(j)} F_j(\theta_j) \\ &\Leftrightarrow K_1 \in \mathbb{K} \text{ and } \forall j \in \{2, \ldots, N\}, (K_j K_{p(j)}^{-1}) \in F_j(\Theta_j)\end{aligned} \tag{3.13}$$

$F_j(\Theta_j)$ is a sub-manifold of $\mathbb{K}$ whose dimension equals to d_j. It refers to the number of DOF associated to the joint between bones $p(j)$ and j. Each constraint $(K_j K_{p(j)}^{-1}) \in F_j(\Theta_j$ is a *scleronomous constraint*. Depending of the method used to minimize the matching cost, each of this constraints can be reformulated using a implicit representation of the manifolds $F_j(\Theta_j)$. This can be achieved by introducing a positive scalar function f_j such that

$$(K_j K_{p(j)}^{-1}) \in F_j(\Theta_j) \Leftrightarrow f_j(K_{p(j)}, K_j) = 0 \tag{3.14}$$

Minimizing the matching cost between the observed image and the model by working directly on the matrices $K_1, \ldots, K_N$ can be of interest in order to exploit independences between disjoint hand parts while computing the matching cost. In that case it is important to ensure the obtained configuration does not violate any of the joint constraints. This can be done by re-projecting $(K_1, \ldots, K_N)$ onto the *articulated motion manifold* $f_K(\Theta)$ at each iteration or by penalizing poses that violate articulated constraints using the functions f_j as done in [Sudderth 2004].

In our approach we choose to perform local search using the parameter θ. In section 4.8.2 we propose a method that exploits some of the independences in the matching cost between disjoint hand parts while working with the reduced representation $\theta \in \Theta$. Note that the use of joints constraints to describe the kinematic structure, allows the introduction of loops in the kinematic chain. However, this not a concern in our approach since the hand does not present loops.

3.2.2 Forward Kinematic Differentiation

The pose estimation involves the minimization of a matching cost function $L(\theta)$. We perform a gradient-descent optimization and thus we will need to differentiate the matching cost with respect to the pose parameter vector θ. The computation of this gradient will involve the differentiation of the forward kinematic function f_K. The hand pose is fully determined by the matrices K_i therefore we can introduce a new function L_K that computes the matching cost given the matrices $(K_1, \ldots, K_N)$. In simple word, the matching cost $L(\theta)$ can be rewritten as:

$$L(\theta) = L_K \circ F_K(\theta) \tag{3.15}$$

Given the derivative of L_K with respect to the matrices K_i, the derivatives of L with respect to θ can be obtained using the chain rule. From a computational point of view there exist several manners to compute these derivatives. Let us, without loss of generality, express in an artificial manner the forward kinematic function as a composition of N simple functions:

$$F_K = f_K^N \circ \cdots \circ f_K^1 \tag{3.16}$$

The first function f_K^1 takes the pose parameters vector θ as input and provides as output the pose parameters vector unchanged (because it will be needed by successive functions) and the pose matrix K_1:

$$f_K^1 : \begin{array}{ccc} \Theta & \rightarrow & \Theta \times \mathbb{K}^N \\ \theta & \mapsto & (\theta, K_1 = F_1(\theta_1)) \end{array} \tag{3.17}$$

Similar to the previous reasoning, each function f_K^j with $j \in \{2, \ldots, N-1\}$ takes as input the full pose parameters vector θ and the matrices K_i for all bones with index i smaller than j, computes the pose matrix K_j of the bone j, and provides as output the concatenation of the input and the newly computed matrix K_j :

$$f_K^j : \begin{array}{ccc} \Theta \times \mathbb{K}^{N-1} & \rightarrow & \Theta \times \mathbb{K}^N \\ (\theta, K_1, \ldots, K_{j-1}) & \mapsto & \big(\theta, K_1, \ldots, \ldots, K_{j-1}, K_j = K_{p(j)} F_{p(j)j}(\theta_{p(j)j})\big) \end{array} \tag{3.18}$$

Note that a single matrix product is performed at each evaluation of a function f_K^j.

Finally, for the last operation, θ should be removed and we should provide as output only pose matrices:

$$f_K^N : \begin{array}{ccc} \Theta \times \mathbb{K}^{N-1} & \rightarrow & \mathbb{K}^N \\ (\theta, K_1, \ldots, K_{N-1}) & \mapsto & \big(K_1, \ldots, K_{N-1}, K_N = K_{p(N)} F_{p(N)N}(\theta_{p(N)N})\big) \end{array} \tag{3.19}$$

Using the sequence of function f_K^i, the matching cost can be rewritten as:

$$L(\theta) = L_K \circ f_K(\theta) = L_1 \circ f_K^N \circ \cdots \circ f_K^1(\theta) \tag{3.20}$$

Then, the Jacobian of the function L can be obtained by multiplying the Jacobian of each function f_K^i through the chain rule. Each Jacobian is a rectangular matrix whose square upper part is the identity matrix, since the input remains unchanged. The matrix product is associative thus there exist at least two strategies to perform the product of Jacobians. The multiplication of the Jacobians can be done from left to right or from right to left. Because the actual computation of L is done by successively applying the function f_k^j from right to left, we refer to *forward differentiation* when we multiply Jacobians from right to the left and to *reverse differentiation* when we multiply Jacobians from left to the right.

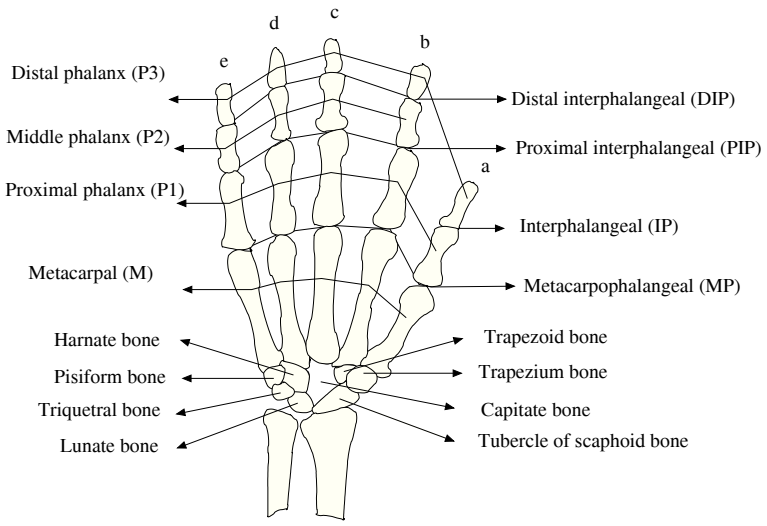
Because the function L_K is scalar, the Jacobian of the function L_K is a line vector. When we multiply Jacobian from the left to the right (*reverse differentiation*) each intermediary product is a vector and thus each multiplication is a vector-matrix product. As a consequence the cost of computing the derivative in the backward order is small. When the multiplication is done from the right to the left (*forward differentiation*), each multiplication is a matrix-matrix product, which is computationally expensive (in comparison with vector-matrix product). The algorithms that correspond to each of these computation orders are presented in Alg.3 and Alg.4. The notation $\theta_{i,l}$ with $l \in \{1, \ldots, d_i^*\}$ refers to the scalar that corresponds to the l^{th} coordinate of the parameters $\theta_i \in \Theta_i$ in its embedding space $\mathbb{R}^{d_i^*}$. Since the decomposition of f_K into elementary functions is not explicit the correspondence with the Jacobian product formulation might not be straightforward to understand.

The computational cost of the forward algorithm is in $O(N^2)$ (lines 14-16 in alg.3) while the reverse algorithm is in $O(N)$ with N the number of bones.

These two approaches for computing the derivatives of complex functions are thoroughly discussed in the *automatic differentiation* literature. The *automatic differentiation* methods aim at computing the derivatives of any function once this function has been implemented in the desired language. We refer to [Griewank 2000] for a detailed introduction to automatic differentiation. This method yields the exact gradient of the function (up to machine precision) and should not be confound with the divided difference method that approximate the derivative by $f'(x) \approx f(x+\varepsilon)-f(x))/\varepsilon$ with a small ε and that is not numerically precise due to truncation errors. In our implementation we use the *forward differentiation* approach, despite being a slower method. The computation of $\partial K_j/\partial\theta_i$ is required in order to estimate the metric in our variable-metric and trust-region method used for the optimization.

3.2.3 Hand anatomy terms

The human hand is a complex mechanical structure comprising bones, muscles serving as tension motors, tendons acting as cables connecting muscles to bone, and a covering of protective soft tissue and skin. The bones are linked at the joints and do not change in size. Muscles produce torque and/or joint movements trough tension. For every muscle there exists one or more muscles that oppose to it through counter-torque and/or opposing motion. The skeleton of the right hand

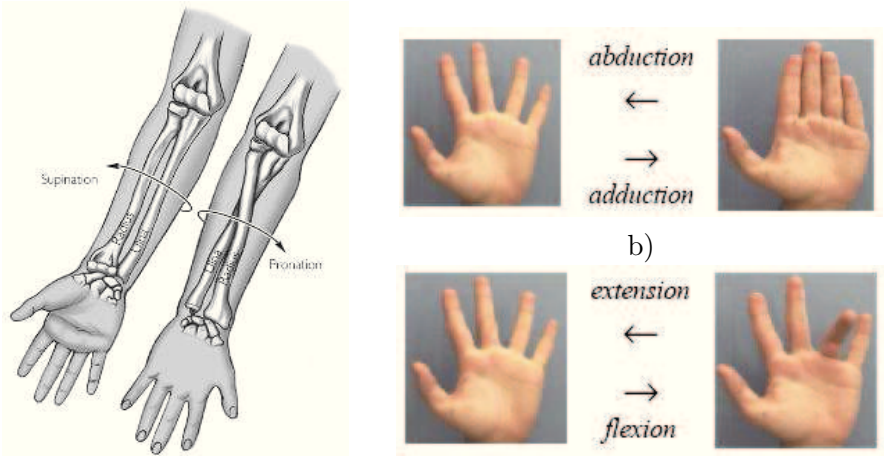


Descriptio 3.2: The right hand skeleton from palmar side

observed from the palmar side is shown in [Figure (3.2)]. We use the terminology and abbreviations from that figure hereafter.

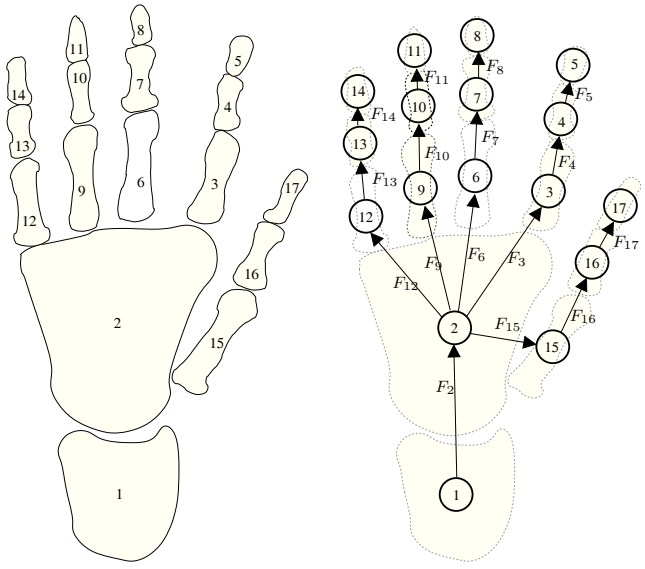
The skeleton of the human hand comprises 27 bones [Figure (2.2)]. The palm (or metacarpus) comprises 5 bones (one for each finger) and the wrist (or carpus) 8 bones. The 14 remaining bones are digital bones called phalanx. Each finger other than the thumb comprises 3 phalanxes: the distal phalanx, intermediate (middle) phalanx and proximal phalanx. The thumb has no middle phalanx but the corresponding metacarpal bone is well separated from the rest of the palm. Due to the geometry of the bones head, two adjacent bones cannot move freely. For joints such as the distal-proximal articulations, the centers of the head and of the hemispherical shape remain matched during the bones displacement. The relative orientation is also restricted and this can be roughly modeled by a pivot joint with one degree of freedom. Depending on the special geometry of the bone, bone pairs present one or two degrees of freedom.

Some hand movement are presented in [Figure (3.3)]. Adduction consists in moving the body parts towards the central axis, which in the case of the hand is between middle and ring fingers. Therefore, it consists in gathering the fingers together. Flexing and extension are well known terms. Abduction is the opposite of adduction and thus consists in spreading the fingers. The rotation of the hand along the forearm axis (called Pronation/supination) is not made at the wrist/forearm interface but within the forearm using the radius bone.



a) b) c)

Descriptio 3.3: a) pronation/supination b) abduction/adduction b) flexing/extension



(a) (b)

Descriptio 3.4: The hand skeleton model and associated graph from palmar side

3.2.4 The hand skeleton model

Inspired by existing pose estimation literature, we merge the bones of the palm and the wrist into a single rigid part. Additionally, we model the forearm, and thus our articulated model is composed of 17 bones. The bones indexes and the resulting graph are shown in [Figure (3.4)]

For each pair of connected bones $(p(i), i)$, we have a function F_i that models the articulation and its parameterization. $F_i(\theta_i)$ is expressed using rotations and translations. We denote $Rx(\theta)$ the 4×4 matrix associated to the rotation around the axis x with the angle θ when using the homogeneous coordinates. Similarly we define rotations Ry,Rz, and the translations Tx,Ty,Tz,Txy.

Each finger other than the thumb is modeled with the same kinematic chain involving 4 DOF [Figure (3.5)].

- Each of the two interphalangeal joints (Distal and Proximal interphalangeal) is modeled with 1 DOF for flexing/extension. We orient the solid axes such that the rotation is done around the axis x.

$$\forall j \in \{4, 5, 7, 8, 10, 11, 13, 14\} : F_j(\theta_j) = Ty(l_{p(j)})Rx(\theta_j) \tag{3.21}$$

  Where, for $i \in \{3, \ldots, 17\}$, l_i is the length of the bone i (these bones have a elongated structure and thus their length are well defined). The angles are equal to 0 when the hand is fully extended.

- The metacarpophalangeal joint is modeled with 2 DOFs for flexing/extension and adduction/abduction. We combine two rotations, a rotation of angle θ_j^xaround the x axis for flexing/extension and a rotation of angle θ_j^z around the z axis for abduction/adduction:

$$\forall j \in \{3, 6, 9, 12\} : F_j(\theta_j) = M_j Rz(\theta_j^z)Rx(\theta_j^x) \tag{3.22}$$

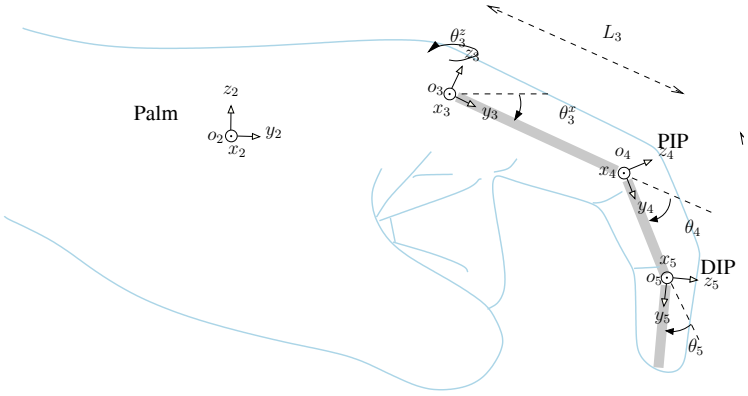
  We have $\theta_j = [\theta_j^x, \theta_j^z]$. For $j \in \{3, 6, 9, 12, 15\}$, M_j is the fixed translation matrix associated to the translation vector $o_j - o_2$ where o_j is the origin of the reference frame of the bone i when the bone is at the rest position. This translation allows attaching each finger at the right position onto the palm.

The thumb is modeled as a kinematic chain with 4 DOF [Figure (3.6)]:

- The distal interphalengeal and the metacarpophalangeal joints are modeled with a one-DOF for flexing/extension. We orient the solid axes such that the rotation is done around the axis z:

$$F_{17}(\theta_{17}) = Ty(l_{16})Rz(\theta_{17}) \tag{3.23}$$
$$F_{16}(\theta_{16}) = Ty(l_{15})Ry(-\pi/4)Rz(\theta_{16}) \tag{3.24}$$

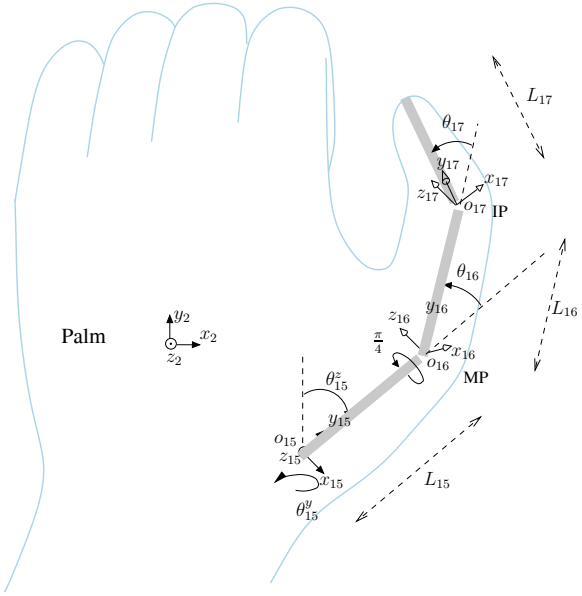
  The rotation of $-\pi/4$ allows to model the fact that the rotation axis of the joints is not aligned with any of the three axis of the metacarpal bone(15).



Descriptio 3.5: Left hand index kinematic chain

- The carpometacarpal articulation is modeled with 2 DOF using two rotations for flexing/extension and abduction/adduction:

$$F_{15}(\theta_{15}) = M_{15} \times Ry(\theta^y_{15}) \times Rz(\theta^z_{15}) \tag{3.25}$$



Descriptio 3.6: Left hand thumb kinematic chain

Several thumb models have been proposed in the past. Most of them assume 5 DOF with orthogonal axes, using two DOF at the MP instead of one (see [Lee 1995],[Kuch 1995]). However, we think that our 4 DOF model which uses non-orthogonal axes ($\pi/4$ rotation in [Figure (3.6)]) is accurate enough to capture the thumb movements.

We model the joint between the hand palm of the forearm using 2 DOF using two rotations:

$$F_2(\theta_2) = M_2 \times Ry(\theta_2^x) \times Rz(\theta_2^z) \tag{3.26}$$

Finally the global pose of the forearm relatively to the absolute frame can be described by a unit quaternion θ_1^q and a three dimensional translation vector $[\theta_1^x, \theta_1^y, \theta_1^z]$:

$$F_1(\theta_2) = Tx(\theta_1^x)Ty(\theta_1^y)Tz(\theta_1^z)R(\theta_1^q) \tag{3.27}$$

where $R(q)$ is the 4×4 matrix corresponding to the normalized quaternion q.

The accumulation of the kinematic parameters for all the fingers provides a vector θ of dimension $D^* = 29$ while the dimension of the manifold Θ is $D = 28$ which corresponds to the number of degrees of freedom (see section 3.2.1.2 for formal definitions of D^* and D).

3.2.5 Linear constraints on joint angles

Bones, muscles and tendons structures lead to a number of natural constraints that should be embedded in the hand model. Prior art [Lee 1995, Kuch 1995, Lin 2000] distinguishes two type of constraints: the *static* and the *dynamic* ones.

The *static constraints* correspond to bounds on a single parameter while the dynamic ones correspond to linear or non-linear equalities or inequalities that involve a combination of two or more parameters. The term *dynamic* does not come from the fact that the time is taken into account, but from the fact that the limit on one DOF depends on the values taken by other ones. The adopted constraints are inspired from [Lee 1995].

3.2.5.1 Static constraints:

For all fingers other than the thumb we have:

$$\begin{aligned} \forall j \in \{3,6,9,12\} : & \theta_{j+1} \in [0^\circ, 110^\circ] \\ & \theta_{j+2} \in [0^\circ, 90^\circ] \\ & \theta_j^z \in [-15^\circ, 15^\circ] \\ & \theta_j^x \in [0^\circ, 100^\circ] \end{aligned} \tag{3.28}$$

The angles are equal to 0 when the hand is fully extended. For the thumb we have:

$$\begin{aligned} \theta_{17} &\in [0^\circ, 110^\circ] \\ \theta_{16} &\in [0^\circ, 80^\circ] \\ \theta_{15}^z &\in [-15^\circ, 80^\circ] \\ \theta_{15}^y &\in [-30, 130^\circ] \end{aligned} \tag{3.29}$$

The angles between the arm and the palm are also restricted

$$\begin{aligned} \theta_2^x &\in [-80^\circ, 80^\circ] \\ \theta_2^z &\in [-20^\circ, 60^\circ] \end{aligned} \tag{3.30}$$

3.2.5.2 Dynamic constraints on flexing of the interphalangeal joints:

We notice that, for all fingers, it is nearly impossible to move the DIP $(F_5, F_8, F_{11}, F_{14})$ without moving the adjacent PIP joint $(F_4, F_7, F_{12}, F_{13})$ or forcing one of them to move in an unnatural manner. This dependency as been modeled in [Lee 1995] by $3\theta_{j+2} - 2\theta_{j+1} = 0, \forall j\{3, 6, 9, 12\}$. However we noticed that when the hand is clenched into a fist $(\theta_{j+2} = \theta_{j+1} = 90^\circ, j \in \{3, 6, 9, 12\})$ it is easy to have the DIP joints extended $(\theta_{j+2} = 0, j \in \{3, 6, 9, 12\})$, which clearly violates the linear relation given above. Therefore we choose to replace the constraints defined by:

$$\begin{aligned} \forall j \in \{3, 6, 9, 12\} : & 3\theta_{j+2} - 2\theta_{j+1} \leq 0 \\ & 3\theta_{j+2} - 2\theta_{j+1} \geq -2\theta_j^x \end{aligned} \tag{3.31}$$

3.2.5.3 Dynamic constraint on flexing of the metacarpophalangeal joints:

The MP joints have an individual flexing range of 110 degrees. However, since isolated flexing of a finger is restricted by accompanying tension in the palmar interdigital ligament, such a flexing might cause flexing of the adjacent fingers. In the same way, a finger's extension is hindered by the flexing of others. This can be modeled using the following linear inequalities:

$$\theta_3^x \leq \theta_6^x + 25 \tag{3.32}$$
$$\theta_3^x \geq \theta_6^x - 55 \tag{3.33}$$
$$\theta_6^x \leq inf(\theta_3^x + 55, \theta_9^x + 20) \tag{3.34}$$
$$\theta_6^x \geq sup(\theta_3^x - 25, \theta_9^x - 45) \tag{3.35}$$
$$\theta_9^x \leq inf(\theta_6^x + 45, \theta_{12}^x + 50) \tag{3.36}$$
$$\theta_9^x \geq sup(\theta_6^x - 20, \theta_{12}^x - 45) \tag{3.37}$$
$$\theta_{12}^x \leq \theta_9^x + 45 \tag{3.38}$$
$$\theta_{12}^x \geq \theta_9^x - 50 \tag{3.39}$$
$$\tag{3.40}$$

3.2.5.4 Dynamic constraint on adduction and abduction of the metacapophalangeal joints:

In their natural position, the fingers can freely carry out adduction or abduction. However when clenched into a fist the abduction/adduction range is greatly reduced. The range of the angle of rotation around the axis z at the MP joint progressively decreases as the angle of the rotation around x increases. This can be expressed mathematically as follows:

$$\forall j\{3,6,9,12\} : |\theta_j^z| \leq 15(1-\alpha) + 5\alpha,\ \alpha = \frac{\theta_j^x}{50} \tag{3.41}$$

3.2.5.5 Inequality system

All these static and dynamic constraints can be resumed in a single inequality system

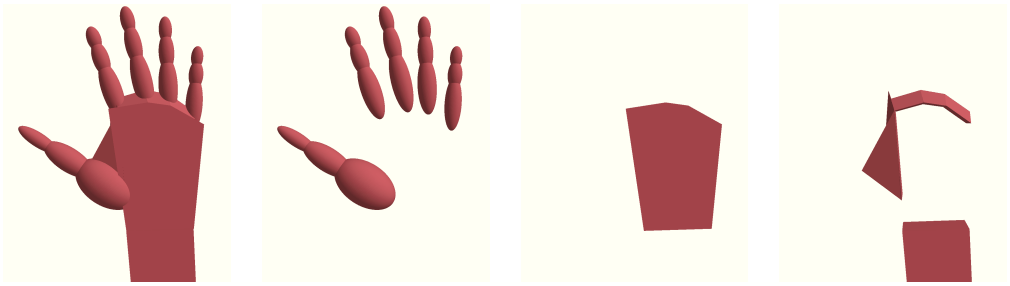
$$A\theta \leq b \tag{3.42}$$

With A a sparse matrix with 29 columns and 68 rows (there are 68 inequality constraints). Because we use linear inequalities the authorized configuration space is the intersection of 52 half spaces (which are convex spaces) and thus is a convex set. This set of constraints should be taken into account during the model fitting process. This is detailed in the section relative to the optimization (3.7.2).

These linear inequalities are not exhaustive. We could define many other inequalities constraints. However it could be cumbersome to identify most of them without a systematic approach. One systematic approach could be to calculate the N-dimension convex hull of a large set of real configurations (each configuration corresponding to a point in $\mathbb{R}^N$). Note that such a methodology does not allow identifying non linear constraints and may lead to many inequalities constraints.

3.2.6 Model calibration

The sizes of the bones of the hand vary across individuals. The size of the phalanx bones are controlled by the length parameters l_j with $j \in \{3, \ldots, 17\}$. The locations of the points where each finger is attached to the palm are controlled by the parameters o_j with $j \in \{3, 6, 9, 12, 15\}$. In order to calibrate these parameters to the user's hand, we ask the user to place his hand in a flat position in a plane that is parallel to the camera image plane. The model is calibrated manually: the user is asked to click on the each finger joint and each finger extremity and at two points at each side of the wrist (where the width of the forearm starts increasing wile going from the elbow to the hand). The width (the two smaller axis length) of the ellipse that is used to model each finger (see next section) is also calibrated. This is done by clicking for each phalanx two points at each side of the finger. The manual calibration process is important to reduce model-related error and get accurate hand positions.



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

Descriptio 3.7: (a) The hand Surface model (b) the phalanxes (c) the palm (d) the interdigital skin and the wrist

3.3 Hand surface model and projection

3.3.1 surface model

The model of the hand surface is composed of ellipsoids and polyhedrons. Each phalanx is modeled using a *prolate spheroid* i.e. an elongated ellipsoid whose two smallest axes have the same length and whose greater axis is aligned with the bone.

The palm is modeled with a rigid convex polyhedron. The forearm and the skin between fingers (called interdigital skin) are modeled using deformable convex polyhedra. The parameters of the ellipsoids and of the polyhedra are estimated during the calibration stage. This surface model yields a good speed/accuracy compromise for the silhouette computation that is critical in our approach. Note that the use of a triangulated surface does not seem to be appropriate for our purpose. It would require the calculation of the position of many points which will be ignored since they do not lie on the silhouette.

3.3.1.1 Ellipsoids

Each phalanx is modeled by a *prolate spheroid* that is a special type of ellipsoid. An ellipsoid is a quadric surface that can be expressed implicitly as follows:

$$\{X \in \mathbb{R}^3 | (X-C)^T A (X-C) = 1\} \tag{3.43}$$

With A a 3×3 positive symmetric definite matrix and $C \in \mathbb{R}^3$. The point C is the center of the ellipsoid.

Since A is positive semi definite, there exists a rotation matrix R and $(a, b, c) \in \mathbb{R}^{+3}$ with $a \geq b \geq c$ such that:

$$A = R \begin{bmatrix} 1/a^2 & 0 & 0 \\ 0 & 1/b^2 & 0 \\ 0 & 0 & 1/c^2 \end{bmatrix} R^T \tag{3.44}$$

Each column of R corresponds to a principal direction of the ellipsoid and the values a, b and c correspond to the radii of the ellipsoid in the three principal directions.

Prolate spheroid are a special type of ellipsoid where the two smallest radii b and c are equal. Let denote T the matrix defined as follow:

$$T \equiv \begin{bmatrix} R^T & -R^T C \\ 0_{3\times 1} & 1 \end{bmatrix} \tag{3.45}$$

This matrix corresponds to a rigid Euclidean transform in the homogeneous coordinates system. We obtain an equivalent equation for the ellipsoid:

$$[x, y, z, 1]^T Q [x, y, z, 1]^T = 0 \tag{3.46}$$

with

$$Q = T^T \begin{bmatrix} 1/a^2 & 0 & 0 & 0 \\ 0 & 1/b^2 & 0 & 0 \\ 0 & 0 & 1/c^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} T \tag{3.47}$$

The hand surface is modeled by attaching a prolate spheroid to each phalanx bone. The shape of each prolate spheroid is specified by a 4×4 matrix Q_0 that remains unchanged. If the spheroid is attached to the bone i, then its equation in the local frame $(\vec{x_i}, \vec{y_i}, \vec{z_i}, o_i)$ of the bone i can be expressed as:

$$[x_i, y_i, z_i, 1]^T Q_0 [x_i, y_i, z_i, 1]^T \tag{3.48}$$

with (x_i, y_i, z_i) being the coordinate of the points in the local frame $(\vec{x_i}, \vec{y_i}, \vec{z_i}, o_i)$. Then, the equation of the ellipsoid in the world coordinate system becomes:

$$[x, y, z, 1]^T Q [x, y, z, 1]^T = 0 \tag{3.49}$$

with

$$Q = (K_i^{-1})^T Q_0 K_i^{-1} \tag{3.50}$$

3.3.1.2 Convex polytopes

The palm [Figure (3.7.c)], the forearm and the interdigital skin [Figure (3.7.d)] are modeled using 3D convex polytopes. Different definitions of polytopes exist in the literature. We define a convex polytope to be the convex hull of a finite set of points. Given a set of points $X \equiv \{x_1, \ldots, x_N\}$, the convex hull $H(X)$ of these points is defined to be the minimal convex set enclosing all these points. The convex hull is the intersection of all convex sets containing X, which constitutes an alternative definition. One can show that the convex hull can be described as the set of convex combinations of the points in X:

$$H(X) = \Big\{ \sum_{i=1}^{N} \lambda_i x_i \mid (\lambda_1, \ldots, \lambda_N) \in \mathbb{R}^{+N}, \sum_{i=1}^{N} \alpha_i = 1 \Big\} \tag{3.51}$$

Each part of the hand that is modeled as a convex polytope is described using a small set of 3D points X. Given a set of points X, the boundary of its convex hull

$\partial H(X)$ can be described as a triangulated surface whose vertices are a subset of X. Such a representation of the convex hull boundary can be computed using standard convex hull algorithms [Barber 1996] and is useful if one want to visualize the 3D model. However, as we will see in section 3.3.4, this triangulated representation of the convex hull boundary is not needed if we only aim at projecting the convex polytope in the image plane.

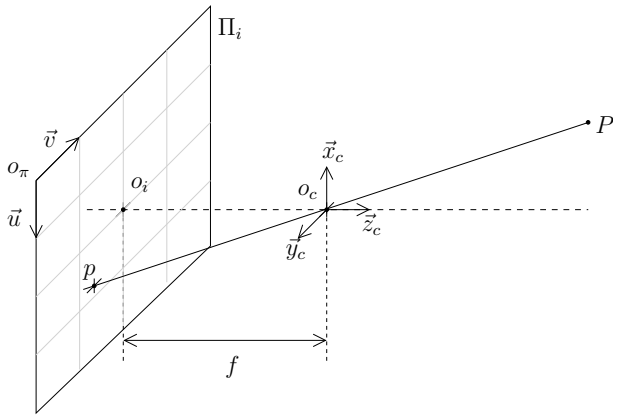
For the interdigital skin and the forearm [Figure (3.7.d)] the set of the points that define each polytope undergo a non-rigid transformation. The points of X that define the polytope are eventually attached to different bones. Let $a(i) \in \{1, \ldots, K\}$ be the index of the bone corresponding to the i^{th} point of the polytope. Given a hand pose θ we compute the matrices $(K_j)_{j=1}^{N}$ defining the rigid frame attached to each bone using forward kinematic (see eqn.3.3). Then the position of a vertex $[X_i, Y_i, Z_i]$ of a polyhedra is obtained from its local coordinates $[X_0^i, Y_0^i, Z_0^i]$ in the reference frame of the bone $a(i)$ using the following equation:

$$[X_i, Y_i, Z_i]^T = K_{a(i)}[X_i^0, Y_i^0, Z_i^0] \tag{3.52}$$

The deformable polytope associated to each hand part is defined as the convex hull of the points being transformed according the bone displacements. Because the convex hull is computed after transformation of the points the polytope remains convex.

In order to evaluate a matching cost between the model in the candidate pose and the observed image, we first need to project the 3D model into the image plane.

3.3.2 Camera model



Descriptio 3.8: The pinhole camera model

In order to project the 3D model into the image we need a model of the camera. We use the simple pinhole model shown in [Figure (3.8)]. A normalized coordinate system $(\vec{x}_c, \vec{y}_c, \vec{z}_c, o_c)$ is associated to the camera with o_c the optical center (where the pinhole is) of the camera. Given a point P and $[X, Y, Z]$ its world coordinates,

its coordinates in the camera coordinate system $[X_c, Y_c, Z_c]$ are obtained using a matrix $K \in \mathbb{K}$ whose columns are made of the coordinates of $\vec{x}_c$,$\vec{y}_c$,$\vec{z}_c$ and o_c in the world coordinate system:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.53}$$

We denote f the focal length of the camera. The image plane Π_i is orthogonal to $\vec{z}_c$ and at a distance f of the optical center i.e. $\Pi_i = \{x|(x - o_c).\vec{z}_c = -f\}$. We denote o_i the *principal point* that is defined as the orthogonal projection of the optical center o_c onto the image plane Π_i. The perspective projection p of a point P onto the image plane Π_i is the intersection of the line (o_c, P) and the plane Π. If we denote (x_c, y_c) 2D the coordinates of p in the coordinate system $(\vec{x}_c, \vec{y}_c, o_i))$ using the Thales theorem we get:

$$x_c = -\frac{fX_c}{Z_c} \tag{3.54}$$

$$y_c = -\frac{fY_c}{Z_c} \tag{3.55}$$

$$\tag{3.56}$$

The coordinates (x_c, y_c) do not correspond to the ones of the point in the pixel coordinate system $(\vec{u}, \vec{v}, o_\pi)$. Let us denote (u_0, v_0) being the coordinates of the principal point in the pixel coordinate system. We suppose the axes $\vec{u}$ and $\vec{v}$ respectively parallel to $\vec{x}$ and $\vec{y}$ with opposite directions i.e $\vec{u} = -|\vec{u}|\vec{x}$ and $\vec{v} = -|\vec{v}|\vec{y}$. The norms $|\vec{u}|$ and $|\vec{v}|$ correspond to the width and the height of a single pixel on the captor.Then the coordinates (x, y) of the point p in the pixel coordinate system are given by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -|u|^{-1} & 0 & u_0 \\ 0 & -|v|^{-1} & v_0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{3.57}$$

That can be expressed as follows::

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \hat{x}/\hat{z} \\ \hat{y}/\hat{z} \end{bmatrix} \tag{3.58}$$

with

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = P_c \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.59}$$

And P_c the camera projection matrix being defined as:

$$P_c = \begin{bmatrix} f/|u| & 0 & u_0 \\ 0 & f/|v| & v_0 \\ 0 & 0 & 1 \end{bmatrix} [I_{3\times3}|0]K \tag{3.60}$$

Finally we can define the projection on the camera through the function Π:

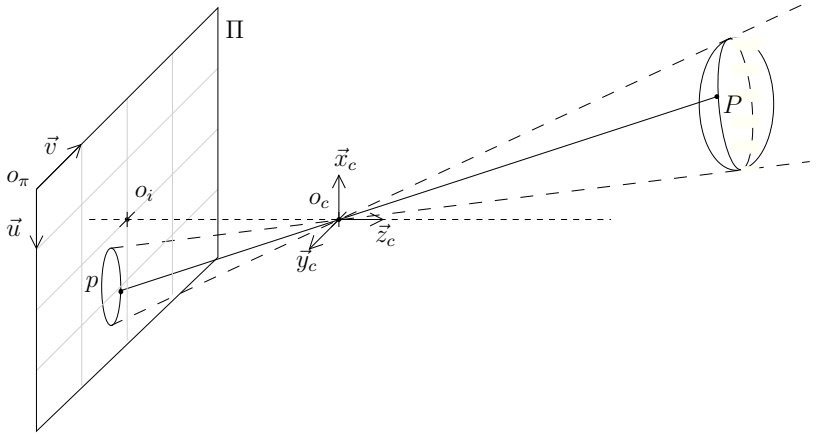
$$\Pi : \begin{array}{ccc} \mathbb{R}^3 & \rightarrow & \mathbb{R}^2 \\ \mathbf{x} & \rightarrow & (P_{c[3,:]}\mathbf{x})^{-1} P_{c[1:2,:]}\mathbf{x} \end{array} \tag{3.61}$$

The use of a single camera makes more convenient to set the camera coordinate system to match the world coordinates system, or $K = I_{4\times 4}$

3.3.3 Ellipsoid projection

Each phalanx is modeled using a 3D ellipsoid that is composed of the points whose coordinates in the world coordinate system satisfy $[x, y, z, 1]Q[x, y, z, 1]^T = 0$.

The perspective projection model will project each 3D ellipsoid into a sub region of the image plane whose boundary is a 2D ellipse [Figure (3.9)].



Descriptio 3.9: Ellispoid projection

Given a point p with coordinates (x, y) in the image coordinate system $(o_\pi, \vec{u}, \vec{v})$, the set of points that project onto (x, y) (i.e. the line $(o_c, p) \equiv \Pi^{-1}(\{(x, y)\})$) are points of coordinate (X, Y, Z) such that $P[X, Y, Z, 1] \sim [x, y, 1]$ i.e. such that there exist an $\alpha \in \mathbb{R}$ such that $P_c[X, Y, Z, 1]^T = \alpha[x, y, 1]^T$ The matrix P is 3 by 4 and therefore is not invertible. We augment the matrix by a forth row to obtain a new matrix P_s:

$$P_s \equiv \begin{bmatrix} P_c \\ 0\ 0\ 0\ 1 \end{bmatrix} \tag{3.62}$$

Using this matrix we have:

$$\begin{aligned} P_c[X, Y, Z, 1]^T = \alpha[x, y, 1]^T &\Leftrightarrow P_s[X, Y, Z, 1]^T = [\alpha x, \alpha y, \alpha, 1]^T \\ &\Leftrightarrow [X, Y, Z, 1]^T = P_s^{-1}[\alpha x, \alpha y, \alpha, 1]^T \end{aligned} \tag{3.63}$$

A point of the image belongs to the projection of the ellipsoid if and only if the line (o_c, p) intersects the ellipsoid. The set of intersections of the line (o_c, p) with the ellipsoid is the set of solutions of the following system:

$$\begin{cases} [X,Y,Z,1]Q[X,Y,Z,1]^T = 0 \\ (X,Y,Z,1)^T = P_s^{-1}[\alpha x, \alpha y, \alpha, 1]^T \end{cases} \tag{3.64}$$

We replace $[X,Y,Z,1]$ in the first equation and introduce $\hat{Q}$ defined as:

$$\hat{Q} \equiv (P_s^{-1})^T Q P_s^{-1} \tag{3.65}$$

The first equation of the system becomes:

$$[\alpha x, \alpha y, \alpha, 1]^T \hat{Q} [\alpha x, \alpha y, \alpha, 1]^T = 0 \tag{3.66}$$

This equation is quadric with respect to α and can be expressed as

$$\alpha^2 [x,y,1]\hat{Q}_{[1:3,1:3]}[x,y,1]^T + 2\alpha \hat{Q}_{[4,1:3]}[x,y,1]^T + \hat{Q}_{4,4} = 0 \tag{3.67}$$

The line (o_i, p) intersects the ellipsoid if this equation has at least one solution i.e. if its discriminant is greater or equal to zero:

$$[x,y,1]\hat{Q}^T_{[4,1:3]}\hat{Q}_{[4,1:3]}[x,y,1]^T - \hat{Q}_{4,4}[x,y,1]t\hat{Q}_{[1:3,1:3]}[x,y,1]^T \geq 0 \tag{3.68}$$

which rewrites

$$[x,y,1]C[x,y,1]^T \leq 0 \tag{3.69}$$

with C being defined as:

$$C = \hat{Q}_{44}\hat{Q}_{[1:3,1:3]} - \hat{Q}^T_{[4,1:3]}\hat{Q}_{[4,1:3]} \tag{3.70}$$

Therefore the region A in the image that corresponds to the ellipsoid projection is defined as:

$$A = \{x \in \mathbb{R}^2 \mid [x,y,1]C[x,y,1]^T \leq 0\} \tag{3.71}$$

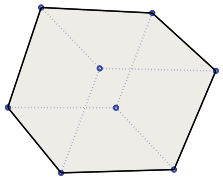
the region A is bounded and its boundary ∂A of this region is an ellipse whose equation is

$$\partial A = \{x \in \mathbb{R}^2 \mid [x,y,1]C[x,y,1]^T = 0\} \tag{3.72}$$

For convenience we will refer to any closed region bounded by an ellipse as a *filled ellipse*. Therefore A is a filled ellipse.

3.3.4 Convex polytope projection

The palm and the interdigital skin are modeled with convex polytopes that correspond to convex hulls of finite sets of points being attached to bones of the skeleton. One could project a polytope by computing the triangulated surface that corresponds to its boundary and then project each triangular facet. Instead, we can use the fact that the perspective projection of a convex hull of a set of points is the convex hull of the perspective projection of that set of points. Therefore we simply project the points used to define the polytope and then compute the 2D



Descriptio 3.10: projecting a cube. The contour of the projection is obtained by computing the convex hull of the projected vertices

convex hull of the projected points, which is a convex polygon. This is illustrated in [Figure (3.10)] where the projection of a cube is obtained as the 2D convex hull of the projected vertices of the cube.

The boundary of a convex polygon can be described as the union of a finite set of connected segments and is homeomorphic with the unit circle. The union of segments constitutes a closed polyline also known as a close polygonal chain. A polyline (a.k.a. a polygonal chain) is specified by a sequence of points $(p_j)_{j=1}^{N}$ called vertices. The corresponding open polyline is defined as the union of the segments joining two successive vertices:

$$P_o = \bigcup_{j=1}^{N-1} \overline{p_j p_{j+1}} \tag{3.73}$$

The corresponding closed polyline is obtained by adding the segment joining the first and last vertex.

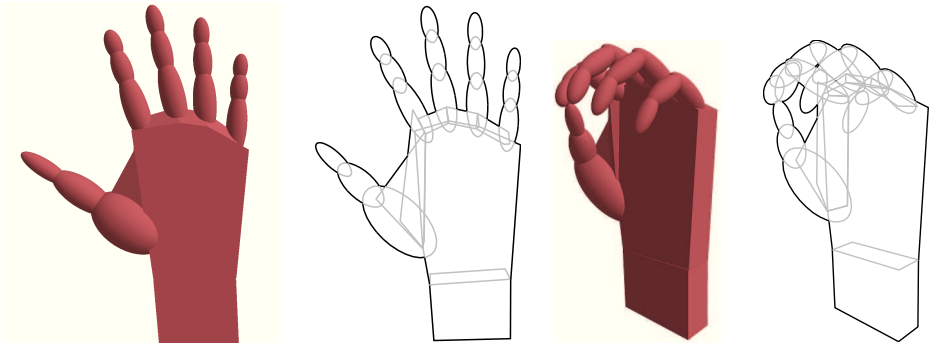
$$P_c = \big(\bigcup_{j=1}^{N-1} \overline{p_j p_{j+1}} \big) \cup \overline{p_1 p_N} \tag{3.74}$$

The problem of testing whether a point lies inside, outside or on the boundary of a polygon is known as the *Point-in-polygon* problem. A simple method called the *even-odd rule algorithm* is to count how many times a ray starting at the point intersects the polygon boundary. Non-boundary points will produce an even number of intersections when the point is outside, and odd otherwise.

3.3.5 Filled ellipses/polygons union

Once we projected each ellipsoids into a filled ellipse and each convex polytope into a convex polygon we can compute the hand silhouette. We define the *hand silhouette* Ω to be the union of the filled ellipses and the polygon. We define the *hand silhouette outline* $\Gamma \equiv \partial\Omega$ as the boundary of the hand silhouette. There are several manners to compute the silhouette and obtain a description of its outline.

Given a point x in the image plane, one can test whether it is in the silhouette by iterating trough each primitive and testing if x is within any projected primitive using the inequality eqn.3.71 for filled ellipses and the *even-odd rule algorithm* for



Descriptio 3.11: The 3D hand model, its projection and the silhouette outline

the convex polygons. Another approach consists in computing the silhouette on a on a set of points that corresponds to a regular pixel grid. Existing 3D engines, such as OpenGL, allows to project and to rasterize a polyhedra on a pixel grid. By converting ellipses into polyhedra, we can use OpenGL to compute a binary image representing the silhouette on a pixel grid (for example 1 if the center of the pixel fall within the silhouette, 0 else). It is then possible to test if a point with non integer coordinates lies on the silhouette boundary by testing if, among the four neighboring points on the pixel grid, two of them have a different label. However, due to the finite resolution of the pixel grid, this test might fail when two silhouette edges fall in the same pixel. These two methods to compute the silhouette do not provide a full, concise and analytical description of the hand silhouette. An analytical description of the silhouette is needed if one aim to compute a continuous matching cost as defined in section 3.4.3.1 and whose practical interest over a discontinuous matching cost is discussed in section 3.7.5.

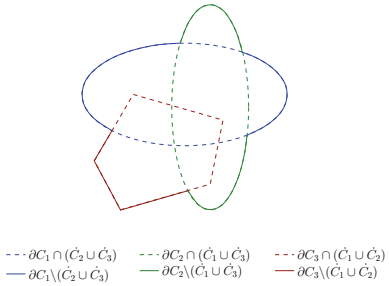
The hand silhouette outline can be described as the union of a finite set of segments and parameterized arcs of ellipses. Therefore while "computing the silhouette" we aim to obtain a compact description of its boundary as the union of segments and arcs.

The necessity to obtain a description of the hand silhouette outline depends on the definition of matching cost that is used to estimate the quality of a hand pose hypothesis. The analytical description of the boundary is necessary to compute a continuous and differentiable matching cost (see section 3.4.3.1).

Several methods for computing the union of two or more polygons have been proposed in the literature [Chazelle 1992, Mairson 1988]. Methods to compute boolean operations (union,intersection and difference) with conic polygons have been proposed in [Gong 2009, Berberich 2002]. The computational geometry library CGAL [Fogel 2006] allows the computation of the union of general 2D curves composed of segments and conic arcs. However this library does not provide the derivatives of the primitive intersections with respect to primitive parameters, which is needed to compute the gradient of our matching cost. The handling of all degenerated cases with ellipses is cumbersome and since we will circumvent this problem by approxi-

mating ellipses by convex polygons (see 3.5.3), we provide only sketch of a method that can be used to solve this problem

Let us consider a simple example with two filled ellipses C_1 and C_2 and a single polygon C_3 shown in [Figure (3.12)]. In order to refer indifferently to a filled ellipse or a polygon we will refer to them as *primitives*.



Descriptio 3.12: Computing the boundary of the union of primitives

We aim to describe the closed curve $\Gamma \equiv \partial\Omega$ that corresponds to the boundary of their union $\Omega \equiv C_1 \cup C_2 \cup C_3$. We assume that at every point where the boundaries of two primitives are intersecting, the boundaries are crossing each other. This assumption can provide the following condition: With this assumption we could show that we have:

$$\Gamma = (\partial C_1 \cup \partial C_2 \cup \partial C_3) \backslash (\dot{C}_1 \cup \dot{C}_2 \cup \dot{C}_3) \tag{3.75}$$

The boundary Γ is the union of portions of the original primitives boundaries. For each primitive boundary we keep the portion that does not lies in the interior of any other primitive [Figure (3.12)]:

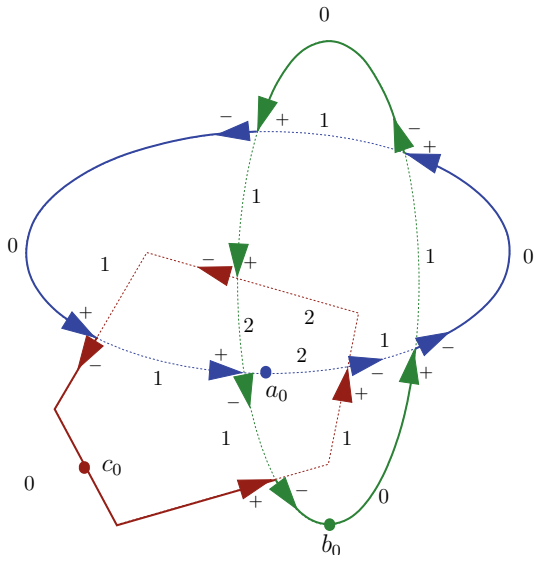
$$\Gamma = \left(\partial C_1 \backslash (\dot{C}_2 \cup \dot{C}_3)\right) \cup \left(\partial C_2 \backslash (\dot{C}_1 \cup \dot{C}_3)\right) \cup \left(\partial C_3 \backslash (\dot{C}_1 \cup \dot{C}_2)\right) \tag{3.76}$$

In order to compute Γ, we first detect all intersections between pairs of primitive boundaries. Each primitive boundary C_i is a closed curve homomorphic with the unit circle and. Using the angular coordinate for example, we can parameterize it using a continuous mapping f_i from $[0, 1]$ to $\mathbb{R}^2$ such that

$$f([0,1]) = C_i, f(0) = f(1) \text{ and } f(x) = f(y) \Rightarrow x = y \text{ or } (x, y) \in \{(0, 1), (1, 0)\} \tag{3.77}$$

For each primitive boundary we store a list of intersections with the corresponding coordinates in the curve. We also associate a binary label specifying if, when walking the boundary counterclockwise, we are going in or out the other intersected primitive. This is illustrated in [Figure (3.13)]. Note that a similar idea is used to compute the union of polygons in [Kui Liu 2007, Greiner 1998].

For each primitive boundary we also count how many other primitives cover the point with coordinate 0 (a_0,b_0 and c_0 in [Figure (3.13)]). Once all intersections are computed, we walk counterclockwise along each primitive boundary starting from



Descriptio 3.13: Computing the boundary of the union of primitives. The head (respectively the tail) of the arrows are positioned at the intersection when we are entering (respectively exiting) the other intersected primitive.

the point with coordinate 0 on the curve. We initialize a counter at 0 and, each time we cross an intersection, we increment or decrement this counter. We keep only portions of the boundary curves where the counter equals 0.

The hand silhouette outline is the union of these portions of polygon boundaries and ellipses arcs. We can connect the portions that share an intersection point and we obtain a set of closed curves. We orient these curves such that the direction corresponds the counterclockwise direction on each primitive. As a consequence the hand silhouette outline is composed of a single curve oriented counterclockwise (that we refer as the external outline) and a set (generally empty) of curves oriented clockwise that define holes in the silhouette (see holes in the silhouette [Figure (3.11)]).

In order to further explain the method, we need to specify how we compute intersections between pairs of ellipses, pairs of polygon boundaries and between an ellipse and a polygon boundary. Because we will ultimately approximate ellipses by convex polygons (see section 3.5.3) we will not explain in detail the methods that can be used to intersect a pairs of ellipses or an ellipse with a polygon boundary. We presented these methods only because we want to follow an intuitive succession of steps and because the justification for approximating ellipses by polygons will only appear later, when computing the matching cost.

3.3.6 Intersecting two ellipses

Several method to compute ellipses intersections have been proposed in the literature ([Hill 1995, Gong 2009, Berberich 2002],[Schneider 2002] p255-256). Despite their different formulations these methods essentially reduce to the same computations. We briefly explain the method proposed in [Hill 1995] and refer the reader to the original material for more details.

Let us consider two ellipses E_0 and E_1 respectively defined as the set of solutions of equations $X^T S_0 X = 0$ and $X^T S_1 X = 0$. Any point in their intersection $E_0 \cap E_1$ should simultaneously satisfy both equations. The points in $E_0 \cap E_1$ should satisfy any linear combination of the two ellipses equations and thus for any $\mu \in \mathbb{R}$:

$$X \in E_0 \cap E_1 \Rightarrow X^T S(\mu) X = 0 \text{ with } S(\mu) \equiv (S_0 + \mu S_1) \tag{3.78}$$

Therefore, for any $\mu \in \mathbb{R}$ the set $E(\mu) \equiv \{X | X^T S(\mu) X = 0\}$ is a conic which contains the intersection of the two ellipses i.e we have $(E_0 \cap E_1) \subset E(\mu)$. We can choose μ such that the conic $E(\mu)$ is degenerate which happens when $\det(S(\mu)) = 0$. The equation $\det(S(\mu)) = 0$ is a cubic equation of μ and thus we can find at most three real solutions μ_1, μ_2 and μ_3 (which are the eigen values of the matrix $-S_0 S_1^{-1}$). For each real solution, the corresponding conic is degenerated i.e. is composed of two lines. The problem of finding intersection between the two ellipses reduces to the problem of finding intersections between a set of lines and a conic. It is possible to get some extraneous intersections, and one need to check them against both ellipses.

We illustrate the method on a example of two ellipses with

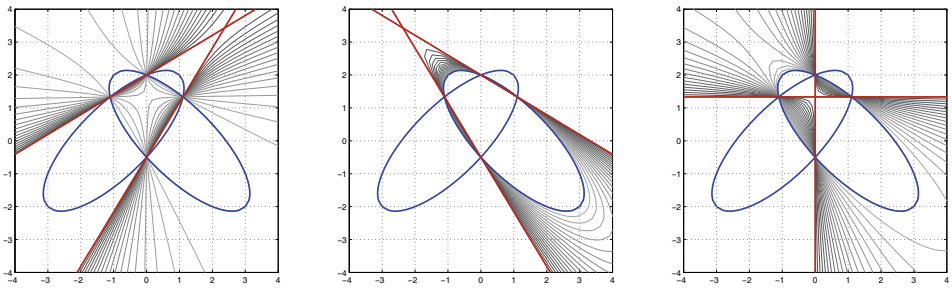
$$S_0 = \begin{bmatrix} 4 & 3 & -4 \\ 3 & 4 & -3 \\ -4 & -3 & -4 \end{bmatrix} \quad S_1 = \begin{bmatrix} 4 & -3 & 4 \\ -3 & 4 & -3 \\ 4 & -3 & -4 \end{bmatrix} \tag{3.79}$$

The eigen values of the matrix $-S_0 S_1^{-1}$ are $\mu_1 = -4.9404$, $\mu_2 = -0.2024$, and $\mu_3 = -1$. For each eigen value $\mu \in \{\mu_1, \mu_2, \mu_3\}$ we have $det(S(\mu)) = 0$ and the 2D conic $E(\mu)$ degenerates into a set of two lines. We illustrate in [Figure (3.14)] the conic obtained for each of the tree eigen values. For each eigen value μ_i we also display intermediary conics $E(\mu)$ with μ sampled uniformly between 0 and μ_i.

3.3.7 Intersecting an ellipse with a polyline

The boundary of a polygon is the union of a set of connected segments that can be described by a closed polygonal chain (a.k.a. closed polyline). A possible method [Schneider 2002] to intersect an ellipse with a polyline consists in intersecting each segment of the polyline against the ellipse. Given the homogeneous coordinates X_1 and X_0 of the two extremities of a segment, a point X belongs to the segment if and only if there exists $\lambda \in [0, 1]$ such that:

$$X = X_0 + \lambda(X_1 - X_0) \tag{3.80}$$



(a) (b) (c)

Descriptio 3.14: intersect two ellipses. a) $\mu = -0.2024$ b) $\mu = -49404$ c) $\mu = -1$

This point belongs to the ellipse of equation $X^T Q X = 0$ if and only if

$$\lambda^2 \underbrace{(X_1 - X_0)^T Q (X_1 - X_0)}_{\varepsilon_1} + \lambda \underbrace{X_0^T Q (X_1 - X_0)}_{\varepsilon_2} + \underbrace{X_0^T Q X_0}_{\varepsilon_3} = 0 \tag{3.81}$$

The discriminant of this quadratic equation is given by

$$\delta = \varepsilon_2^2 - 4\varepsilon_1\varepsilon_3 \tag{3.82}$$

If $\delta < 0$, then the the equation 3.81 does not have a solution. The line and the ellipse do not intersect. If $\delta = 0$ there exists a single solution. The line is tangent to the ellipse. Otherwise $\delta > 0$ and there exist two solutions. The line intersects the ellipse twice and we can compute these intersections using $\lambda \in \{(-\varepsilon_2 + \sqrt{\delta})/(2\varepsilon_1), (-\varepsilon_2 - \sqrt{\delta})/(2\varepsilon_1)\}$. Each solution in the interval $[0, 1]$ corresponds to an intersection of the ellipse with the segment. If none of the solution lies in the interval $[0, 1]$ then the segment does not intersect the ellipse.

Because we test each segment, the complexity of this method is linear with the number of segments in the polygon. However the polygons we use are not arbitrary but convex polygons, and thus it might be possible to accelerate the computation of the intersections with the ellipse toward sub-linear complexity.

3.3.8 Intersecting boundaries of two polygons

The computation of intersections between the boundaries of two polygons is a classical problem in computational geometry [Boissonnat 1995, Preparata 1985]. If the polygons have respectively m and n edges, the complexity of a brute force method, that consist in testing the intersection of each edge of the first polygon with each edge of the second polygon, would be of $O(nm)$. In some cases, like for example where all the edges of the first polygon intersect all edges of the second polygon, this complexity cannot be reduced. Fortunately, the number of intersections between polygon edges is, in general, much smaller than $n \times m$ and it is possible to avoid testing all pairs of edges. The two algorithms proposed in [Chazelle 1992] and [Mairson 1988] allow to compute all intersections between pairs of edges with a complexity of $O((m + n)log(n + m + k))$ with k being the

number of intersections. A simple algorithm in $O((m+n+k)log(n+m+k))$ has been proposed in [Žalik 2000]. When the polygons are convex the number of intersection is at most $n+m$. Linear time algorithms, i.e with complexity $O(m+n)$, have been proposed two solve the problem of intersecting two convex polygons [O'Rourke 1982a, Toussaint 1985, O'Rourke 1982b, Yang 2006]. We propose an algorithm [3] that is of order $O(m+n)$ in the case of convex polygons, but that can also handle arbitrary polygons with a worst complexity of $O(mn)$. We think our method is conceptually simpler than the methods proposed in [O'Rourke 1982a, Toussaint 1985, O'Rourke 1982b, Yang 2006].

3.3.8.1 Intersecting two polylines with increasing abscisse

Let first consider the problem of computing all intersection between two open polylines P_1 and P_2 whose vertex coordinates are increasing in x. Each polyline P_i (a.k.a a polygonal chain) is defined by a sequence of points $(p_j^i)_{j=1}^{N_i}$ called vertices. Each open polyline P_i is then defined as the union of the segments joining two successive vertices:

$$P_i = \bigcup_{j=1}^{N_i} \overline{p_j^i p_{j+1}^i} \tag{3.83}$$

We denote (x_j^i, y_j^i) the two coordinates of the point p_j^i i.e. the j^{th} point of polyline P_i. As we will see it is possible to compute all intersections between these two polylines without testing all pairs of segments. We suppose the vertex coordinates of both polylines to be increasing in x i.e.

$$\forall i \in \{1,2\}, j \in \{1,\dots,N_i-1\} : x_{j+1}^i \geq x_j^i \tag{3.84}$$

If two edges intersect then their corresponding intervals on the x axis should also intersects. Therefore we aim to list S of all pairs $(i,j) \in \{1,\dots,N_1\} \times \{1,\dots,N_2\}$ such that $[x_i^1, x_{i+1}^1] \cap [x_j^2, x_{j+1}^2] \neq \emptyset$. Using the fact that $x_1^i < \dots < x_{N_i}^i$ This can be done in a linear time using the algorithm 2.

The number of elements in S is less than N_1+N_2 and the complexity of this algorithm is of order $O(N_1+N_2)$. In order to intersect the two polyline we test for each $(i,j) \in S$ the intersection between the two segments $\overline{p_i^1, p_{i+1}^1}$ and $\overline{p_j^2, p_{j+1}^2}$. Testing the intersection between two segments can be done easily using additions and multiplications. If the two segments intersect one needs to perform a division to compute the location of the intersection. Robust methods to test and compute the intersection of two line segments have been proposed in the literature [Gavrilova 2000, Zhu 2005]. Because we will need to differentiate the location of

[3] We found out that our method has some strong similarities with a method implemented in Matlab by Sebastian Hï¿½lz. The Matlab file *curveintersect.m* can be downloaded here `http://www.mathworks.co.kr/matlabcentral/fileexchange/8908`. however their method is not linear with the number of edges because the Matlab mex function *histc.c* called line 300 of *interp1.m* does binary search for each element and does not takes advantage of the fact that the vectors *xiCol* are already ordered.

Algorithm 2: listIntersectingSegments

Data: Two increasing sequences $(x_1^i, \dots, x_{N_i})_{i=2}^2$ with $x_1^i < \dots < x_{N_i}^i$
Result: $S = \{(i,j) \in \{1,\dots,N_1\} \times \{1,\dots,N_2\} \mid [x_i^1, x_{i+1}^1] \cap [x_j^2, x_{j+1}^2] \neq \emptyset\}$

```
i ← 1;
j ← 1;
S = ∅
while x^1_{i+1} < x^2_j do
    i ← i + 1;
while x^2_{j+1} < x^1_i do
    j ← j + 1;
while (i < N_i) and (j < N_j) do
    if x^1_{i+1} < x^2_{j+1} then
        i ← i + 1;
        S ← S ∪ {(i, j)};
    else
        j ← j + 1;
        S ← S ∪ {(i, j)};
```

the intersection point with respect to the hand pose parameter, we adopt simple equations to compute the intersection between two line segments [O'Rourke 1999]. For clarity of the notation we rename the two extremities of the two segments: $a \equiv p_i^1, b \equiv p_{i+1}^1, c \equiv p_j^2$ and $d \equiv p_{j+1}^2$. The two line segments write:

$$\overline{ab} \equiv \{ua + (1-u)b | u \in [0,1]\} \tag{3.85}$$
$$\overline{cd} \equiv \{vc + (1-v)d | v \in [0,1]\} \tag{3.86}$$

The intersection point of the two lines is obtained by solving

$$ua + (1-u)b = vc + (1-v)d \tag{3.87}$$

The solution writes:

$$u = [a_x(d_y - c_y) + c_x(a_y + d_y) + d_x(c_y - a_y)]/D \tag{3.88}$$
$$v = -[a_x(c_y - b_y) + b_x(a_y - c_y) + c_x(b_y - a_y)]/D \tag{3.89}$$
$$D = (a_x - b_x)(d_y - c_y) + (d_x - c_x)(b_y - a_y) \tag{3.90}$$

If the two lines are parallel, then the denominator D equals to zero ans this require special handling. The intersection point x is obtained by substituting u or v i.e using either $x = ua + (1-u)b$ or $x = vc + (1-v)d$. The intersection belongs to the two segments if and only if the solution verifies $u \in [0,1]$ and $v \in [0,1]$. One can perform these two tests without performing the division by comparing the numerators appearing in the computation of u and v with 0 and D.

3.3.8.2 Intersecting two closed polylines

We now consider the problem of intersecting the boundaries of two polygons. The boundary of a polygon is a closed polyline. We decompose the closed polyline into a minimal set of open polylines monotonic in x (i.e increasing or decreasing) and re-index vertices in each extracted polyline to obtain increasing open chains. For convex polygons we get only two polylines per polygons. We compare each pair of polylines extracted from the two polygons. We get only four pairs of polylines if the two polygons are convex. The method is thus linear with the number of vertices of the polygons if the polygons are convex.

3.4 Matching cost

We described in the previous section how to compute the hand silhouette Ω and its boundary $\Gamma = \partial\Omega$ given the hand pose θ. Given an observed image I, we need to assess the quality of a candidate hand pose, described by the parameter vector θ. This is done by measuring the consistency between the synthesized hand silhouette Ω and the observed image. In a Bayesian framework, this is done by recovering the probability density $p(I|\theta)$ of getting the observed image I given a hand configuration θ. This probability is built from a generative models for pixels of the hand, the background and other regions. Those regions are ordered in depth and may occlude each other, leading to a so called 2.1D sketch model [Nitzberg 1990]. Our model leads to a log-likelihood $L(\theta) = -log(p(I|\theta))$ that can be interpreted as a cost function with the lowest value corresponding to the hand configuration that is most consistent with the observed image. Despite our specific modeling choice on background and foreground separation, our approach can be adapted to other well-known contour-based or area based segmentation functional $f(\Gamma, I)$. This could be done by taking the Gibbs distribution $p(I|\theta) \propto exp(-f(\Gamma(\theta), I)/T)$ with T a temperature parameter. Our generative model supposes that the observation is made of four classes representing four different elements of the image: 1) the static background, 2) the skin, 3) foreground which might occlude the hand and 4) parts of the body behind the hand. Unlike most silhouette matching methods [Delamarre 1999, Ouhaddi 1999b], we do not assume pre-segmentation of the image into those four classes in order to match the hand to a segmented silhouette. Our method unifies segmentation and hand pose estimation in a single optimization problem thus improving overall robustness.

3.4.1 Generative colors models

Under the assumption of a static camera, we assume that the background is stationary or changing in a gradual fashion. The background model presented in [Stauffer 1999] that uses a mixture of Gaussian Distributions for each pixel is an excellent compromise between low complexity and fairly good approximation of stationary signals. This yields the following background log-likelihood:

$$f_{\text{bk}}(x) = -\log(\sum_i w_x^i N(\mu_x^i, \Sigma_x^i)(I(x)) \tag{3.91}$$

We model the three other classes (the hand skin, the parts of the body behind the hand and the foreground which might occlude the hand) using a kernel-based approximation (Parzen windows) of the RGB histograms. Some minimal interaction is required from the user in order to recover an initial form of this non-parametric approximation. We denote d_{hd}, d_{for} and d_{bd} the respective approximated distributions on the RGB space. The histograms are thresholded such that no-zero probability is given to any color. In the absence of spatial inter-pixel dependencies within each part, we obtain the following observation log-likelihoods:

$$\begin{aligned} f_{\text{hd}}(x) &= -\log(d_{\text{hd}}(I(x))) \\ f_{\text{bd}}(x) &= -\log(d_{\text{bd}}(I(x))) \\ f_{for}(x) &= -\log(d_{for}(I(x))) \end{aligned} \tag{3.92}$$

We illustrate these function for an image composed of the hand, a red moving foreground and a moving background composed of the hips region with a dark brown shirt and blue jean in [Figure (3.15)].

3.4.2 The discontinuous likelihood

We suppose the image to be of size W by H with W the width and H the height in pixels. Let $L_d \equiv \{0, \ldots, W-1\} \times \{0, \ldots, H-1\}$ denote the set of discrete pixel coordinates. The image I, the functions f_{hd},f_{bd} and f_{for} are defined on set on discrete pixel locations L_d.

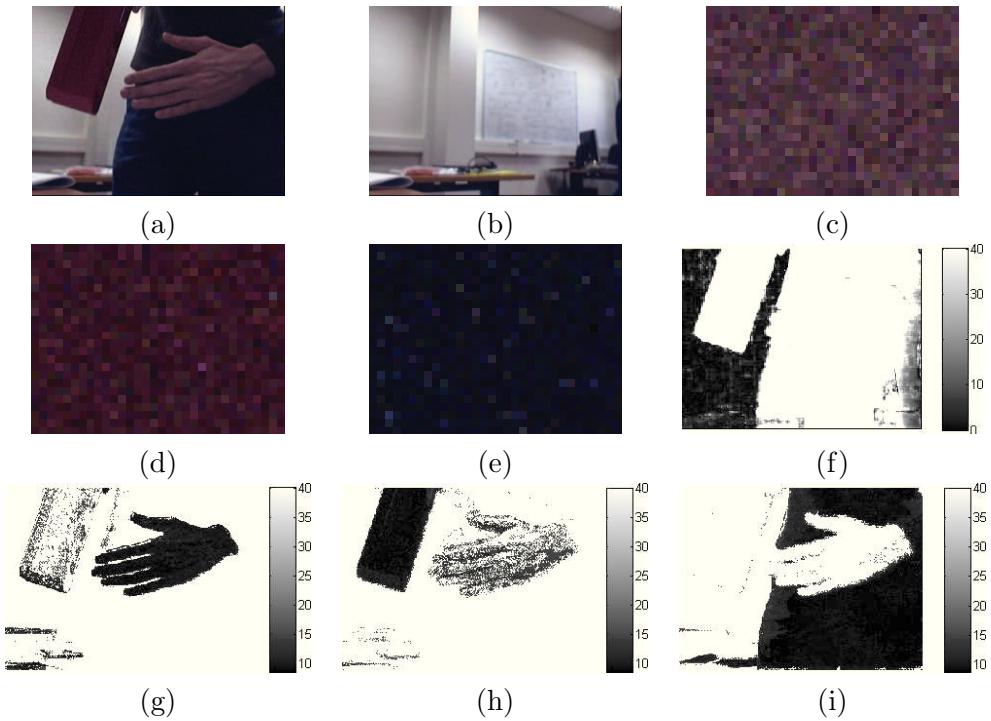
We denote M_{for}, M_{hd}, M_{bd} and M_{bk} respectively the characteristic functions (or masks) of the regions (including occluded parts) corresponding to foreground, hand, body and background. These functions are defined on the discrete pixel grid, therefore a mask can be associated to an element of $\{0,1\}^{M \times N}$. The continuous characteristic function of the hand silhouette is defined as follows:

$$\chi_\Omega : \begin{array}{ccc} [0,H] \times [0,W] \times \Theta & \rightarrow & \{0,1\} \\ (x,y;\theta) & \mapsto & 1 \text{ if } (x,y) \in \Omega(\theta),\ 0 \text{ else} \end{array} \tag{3.93}$$

The discrete mask of the hand is obtained by restricting the function to the integer coordinates:

$$M_{\text{hd}} : \begin{array}{ccc} \{0, \ldots W-1\} \times \{0, \ldots H-1\} \times \Theta & \leftarrow & \{0,1\} \\ (i,j;\theta) & \mapsto & 1 \text{ if } (i,j) \in \Omega(\theta),\ 0 \text{ else} \end{array} \tag{3.94}$$

This mask can be computed by testing for each x in L_d if it is inside the silhouette Ω. For each x this can be done by iterating trough each primitive and testing if x is within any projected primitive - using the inequality eqn.3.71 for filled ellipses and the *even-odd rule algorithm* for the convex polygons. This could be done easily



Descriptio 3.15: negative Log-likelihood maps: (a) video frame (b) mean image of the background (c) random samples of the hand distribution (d) random samples of the foreground distribution (e) random samples of the moving background distribution (f) background minus log-likelihood $f_{\mathrm{bk}}(thedarkerthemostlikely)$ (g) hand background minus log-likelihood f_{hd} (h) foreground background minus log-likelihood f_{for} (i) moving background minus log-likelihood f_{bd}

by rasterizing the hand into a binary image using a library such as OpenGL. For the two mask M_{for} and M_{bk} we assume the prior probabilities to be uniform i.e. $p(M_{\mathrm{for}}) = 2^{-M\times N}$ and $p(M_{\mathrm{bk}}) = 2^{-M\times N}$. The background mask $M_{\mathrm{bk}}(x)$ remains equal to 1 on the entire image. We do not require the regions to form a partition of the image, but we give a depth order: Foreground, hand, body and background. Each part may occlude deeper parts. The probability of observing I given the hand parameter θ is obtained by marginalization over the possible configurations of the masks M_{for} and M_{bd}. The background mask $M_{\mathrm{bk}}(x)$ remains equal to 1 on the entire image support and it is therefore not marginalized.

$$p(I|\theta) \equiv p(I|M_{\mathrm{hd}}(\theta)) \propto \int_{M_{\mathrm{for}}} \int_{M_{\mathrm{bd}}} p(I|M_{\mathrm{hd}}(\theta), M_{\mathrm{for}}, M_{\mathrm{bd}}) \\ p(M_{\mathrm{for}})p(M_{\mathrm{bd}})dM_{\mathrm{bd}}dM_{\mathrm{for}} \tag{3.95}$$

Defining $\prec$ the depth order relation we get the ordering for $\prec$ hd $\prec$ bd $\prec$ bk and we can write the observation likelihood taking occlusions into account:

$$p(I|M_{\text{hd}}, M_{\text{for}}, M_{\text{bd}}) = \sum_{x \in L_d} \sum_{i} d_j(I(x)) \big[M_i(x) \prod_{j \prec i} (1 - M_j(x)) \big] dx \tag{3.96}$$

In order to simplify the computation we approximate by maximizing rather than marginalizing over the other masks M_{for} and M_{bd}. Maximizing the expression in (eqn.3.96) with respect to M_{for} and M_{bd} and taking the negative logarithm of the maximum leads to an approximate log-likelihood of the hand configuration given the observed image. This functional, also referred to as the data cost function, is finally expressed as:

$$\begin{aligned} L_b(\theta) &= \sum_{x \in L_d} M_{\text{hd}}(x;\theta) f_{\text{in}}(x) + (1 - M_{\text{hd}}(x;\theta)) f_{\text{out}}(x) && (3.97) \\ &= \underbrace{\sum_{x \in L_d} f_{\text{out}}}_{K} + \sum_{x \in L_d} M_{\text{hd}}(x;\theta) \underbrace{(f_{\text{in}}(x) - f_{\text{out}}(x))}_{f(x)} && (3.98) \end{aligned}$$

with

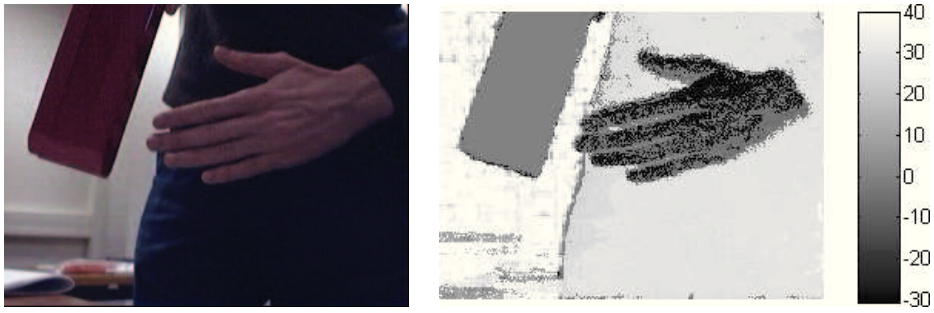
$$\begin{aligned} f_{\text{in}}(x) &= min(f_{\text{hd}}(x), f_{\text{for}}(x)) && (3.99) \\ f_{\text{out}}(x) &= min(f_{\text{bk}}(x), f_{\text{bd}}(x), f_{\text{for}}(x)) && (3.100) \end{aligned}$$

Using the silhouette characteristic function $\chi_\Omega(x, y; \theta)$ the discontinuous matching cost can be rewritten as follows:

$$L_b(\theta) = K + \sum_{(i,j) \in L_d} f(i,j) \chi_\Omega(i,j;\theta) \tag{3.101}$$

The value K does not depend on the synthesized silhouette and can be calculated only once for a given frame. We refer to f as the image of differences of log-likelihoods (which is equivalent to the logarithm of likelihood-ratios). This image equals zeros whenever the likelihood of being inside the hand silhouette equals the one of being outside the silhouette. Pixels that looks like the moving foreground may or may not cover the hand, and this cannot be decided based on the pixel color, therefore $f(x) = 0$ for these points, which is illustrated in [Figure (3.16)] where the region of the red object is gray.

Unfortunately the matching cost L_b is a discontinuous function of θ. Because M_{hd} is a binary image, the matching cost L_b cannot take more than $2^{W \times H}$ distinct values and thus it is piecewise constant. When θ varies, the boundary Γ of the silhouette Ω continuously displace in the image plane. When Γ crosses a point (i, j) in L_d, i.e. with integer coordinates, the binary values of $M_{\text{hd}}(i, j; \theta)$ suddenly change from 0 to 1 or from 1 to 0 which results in a discontinuous change in the matching cost. This matching cost is not suitable for local optimization simply because it is piecewise constant and therefore its gradient is null almost everywhere. Local search



(a) (b)

Descriptio 3.16: (a) the observed image (b) the image of differences of log-likelilhoods (or logarithm of likelihood-ratio)

could be performed using gradient free optimization routines such as the downhill simplex method. However the convergence rate of such methods is slow. One can overcome this limitation through the definition of a continuous and differentiable matching cost, for which we can derive the analytical expression of the gradient and the Hessian. This will allow to perform the local search efficiently.

3.4.3 The continuous likelihood

3.4.3.1 Using a continuous integration

In order to define a continuous and differentiable matching cost, we define a new function f_c that interpolates the functions f on the continuous image domain L_c. Then we replace, in the equartion eqn.3.101, the finite sum over the finite set of discrete pixels L_d by a continuous integrals over the continuous image domain $L_c \equiv [0, W] \times [0, H]$. The continuous matching cost is defined by:

$$L(\theta) = K + \iint f_c(x, y)\chi_\Omega(x, y; \theta)dxdy, \tag{3.102}$$

and can be rewritten as:

$$L(\theta) = K + \iint_{\Omega(\theta)} f_c(x, y)dxdy \tag{3.103}$$

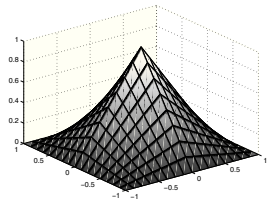
We formulate the interpolation of f as a convolution by a kernel k:

$$f_c(x, y) = \sum_{(i,j)\in L_d} f(i, j)\ k(x - i, y - j) \tag{3.104}$$

The kernel depends on the type of interpolation we use:

- The nearest neighbor (shifted by 0.5) interpolation is obtained using the kernel that is constant and equal to 1 within the square $[0, 1]^2$ and zeros outside the square:

$$k_n(x, y) = 1 \text{ if } (x, y) \in [0, 1]^2,\ 0 \text{ else} \tag{3.105}$$



Descriptio 3.17: kernel for bilinear interpolation

Some basic variables manipulation will lead to:

$$f_c(x,y) = \sum_{(i,j)\in L_d} f(i,j)k_n(x-i,y-j) = f(\lfloor x \rfloor, \lfloor y \rfloor) \tag{3.106}$$

with $\lfloor x \rfloor$ the greatest integer that is smaller or equal to x.

- The bilinear interpolation is obtained using the kernel k_b defined as follows:

$$k_b(x,y) = (1-|x|)(1-|y|) \text{ if } (x,y) \in [-1,1]^2,\ 0 \text{ else} \tag{3.107}$$

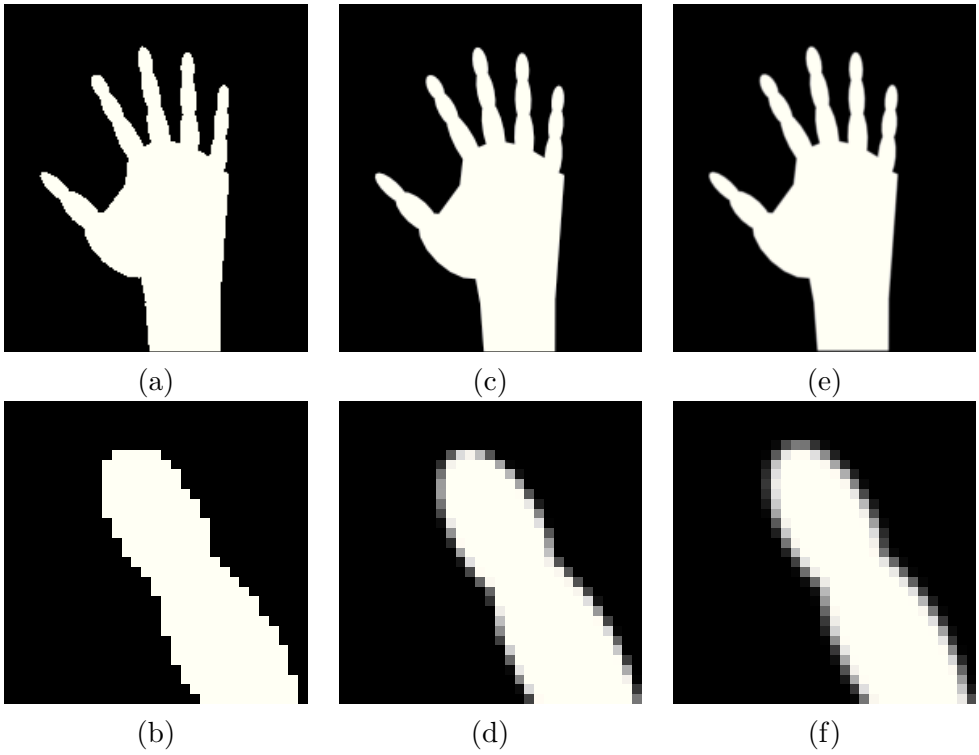
The kernel k_b is shown in [Figure (3.17)]. We define the function $\varepsilon(x) = x - \lfloor x \rfloor$. With some rewriting we get:

$$\begin{aligned} f_c(x,y) =& (1-\varepsilon(x))(1-\varepsilon(y))f(\lfloor x \rfloor, \lfloor y \rfloor) \\ &+ (\varepsilon(x))(1-\varepsilon(y))f(\lfloor x \rfloor + 1, \lfloor y \rfloor) \\ &+ (1-\varepsilon(x))(\varepsilon(y))f(\lfloor x \rfloor, \lfloor y \rfloor + 1) \\ &+ (\varepsilon(x))(\varepsilon(y))f(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1) \end{aligned} \tag{3.108}$$

The advantage of the continuous matching cost $L(\theta)$ over the matching cost $L_b(\theta)$ (that was defined in the previous section) is that $L(\theta)$ can be differentiated.

3.4.3.2 Relationship with antialiasing

There is a strong relationship between the problem of computing *exactly* the continuous matching cost and performing anti-aliasing. In the context of computer graphics and signal processing in general the aliasing problem appears when a signal with high frequencies (spacial frequencies for an image) is sampled at a frequency smaller than $f_e/2$ (f_e being the highest frequencies with no negligible energy in the signal). In computer graphics this appears when rendering geometric primitives with sharp boundaries on a discrete pixel grid. The abrupt intensity discontinuities at the boundaries cause high amplitude spectral components at extremely high frequencies. In practice if one discretizes an image composed of object with sharp boundaries, this result in jaggies or stair-like artifacts (see first column in [Figure (3.18)]. In real cameras the optics creates some low-pass filtering (blurring) and pixel have spacial extension, which removes the aliasing effect.



Descriptio 3.18: Masks of the hand and a zoom on the index extremity: (a-b) the binary mask M_{hd} (c-d) the antialiased mask $\tilde{M}$ (see eqn.3.109) corresponding to nearest neighbor interpolation (e-f) the antialiased mask $\tilde{M}$ corresponding to bilinear interpolation

Three techniques are now commonly used to reduce aliasing: pre-filtering, uniform super-sampling, and stochastic sampling. Pre-filtering consists in filtering the image in order to remove high spatial frequencies before sampling at pixel rates. For each pixel, the combination of the filtering and sampling operations can be interpreted as the computation of the scalar product of the filter kernel centered at the pixel with the continuous image. The super-sampling methods consist in increasing the sampling rates (and hence the Nyquist rates) to some small multiple of pixel rates and then form each pixel intensity of the discrete image from a weighted average of neighboring samples. The super-sampling methods do not remove the discontinuous intensity changes, but reduce their magnitude, when the objects being drawn move continuously. Stochastic super-sampling methods randomly displace the super-sampling positions so that any aliased components appear as uncorrelated noise in the discrete image. The noise induces temporal discontinuities in the discrete image when the objects being drawn displace continuously. Only the pre-filtering method can provide a discrete image that varies continuously when the rendered object displace continuously.

When the filter is the characteristic function of a unit square and the object being drawn are polygons with uniform colors, the antialiasing process can be re-

interpreted as follows: Each polygon is clipped to the extend of the pixel and the contribution of each polygon to the pixel is weighted by area of the clipped region. This method is referred as the unweighted area sampling in [Foley 1996](p133-134) and exact algorithms have been proposed in [Catmull 1978] and [Duff 1989] to compute the weights. In [Catmull 1978], each polygon is clipped to each pixel using the Weiler-Atherton method which is quite expensive. In [Duff 1989] the approach is extended by replacing the area computation with a contour integral. Several optimizations that exploit the coherence of scan-conversion are also proposed. A method to compute exact antialiased triangles with any *prisme spline* filter (which include the 2D box filter and the filter k_b that is used for bilinear interpolation) has been proposed in [McCool 1995].

In order to see the relationship between the problem of computing the continuous matching cost and performing anti aliasing we rewrite the integral defining $L(\theta)$ as follows:

$$\begin{aligned} L(\theta) &= K + \iint f_c(x,y)\chi_\Omega(x,y;\theta)dxdy \\ &= K + \iint \left[\sum_{(i,j)\in L_d} f(i,j)k(x-i,y-j)) \right] \chi_\Omega(x,y;\theta)dxdy \\ &= K + \sum_{(i,j)\in L_d} f(i,j) \underbrace{\iint k(x-i,y-i))\chi_\Omega(x,y;\theta)dxdy}_{\tilde{M}(i,j)} \end{aligned} \tag{3.109}$$

The matching cost is rewritten as a finite sum over the image pixels where the contribution of each pixel is weighted by $\tilde{M}(i,j)$. The image $\tilde{M}$ can be interpreted as an anti-aliased version of the hand silhouette binary $maskM_{\text{hd}}$ or the characteristic function $\chi_\Omega(x,y)$. Indeed $\tilde{M}(i,j)$ corresponds to the evaluation at the integer location $(i,j) \in L_d$ of χ_Ω convolved with the kernel k. This convolution filters out (or reduce) high spatial frequencies. The obtained images $\tilde{M}$ corresponding to the nearest neighbor interpolation and the bilinear interpolation are shown in [Figure (3.18)].

If we use the nearest neighbor interpolation shifted by 0.5 (eqn.3.106) one can easily interpret $\tilde{M}$ as the antialiased version to the silhouette characteristic function using the *unweighted area sampling* method. We rewrite the equation using the corresponding kernel k:

$$\begin{aligned} \tilde{M}(i,j) &= \iint k(x-i,y-j)\chi_\Omega(x,y)dxdy \\ &= \iint_{(x,y)\in[i,i+1]\times[j,j+1]} \chi_\Omega(x,y)dxdy \\ &= \iint_{\Omega(\theta)\cap[i,i+1]\times[j,j+1]} 1dxdy \end{aligned} \tag{3.110}$$

For each pixel (i, j) the value $\tilde{M}(i, j)$ is the measure of the area of intersection of the silhouette Ω with the pixel (i, j) that is spatially extended into a unit square $[i, i+1] \times [j, j+1]$. The matching cost is a finite sum over the image pixels where the contribution of each pixel is weighted according to the surface of the intersection between the pixel and the silhouette.

In order to compute the exact matching cost we could use the antialiasing algorithms proposed in [Catmull 1978] and in [Duff 1989]. However, as shown in the next section, it is not actually needed to compute the antialiased image M towards computing $L(\theta)$. Our method uses the Green theorem to reduce the computation complexity by replacing the integral within the silhouette by an integral along the silhouette outline.

3.5 Numerical computation of the matching cost

Hand pose estimation through iterative minimization of the matching cost requires the ability to compute the gradient of matching cost for some hand pose candidate. The need of computing exactly the continuous matching cost itself during the optimization is not obvious. The computation of the discontinuous cost L_d could be sufficient. This could be done easily by i) rasterizing the hand into a binary image using a library such as OpenGL ii) summing the function f at location that have been labeled as the hand silhouette in the binary image. This approach is simple and avoids the need of computing the intersections between projected primitives. However, in section 3.7.5, we compared the use of the *approximate* (discontinuous) and the *exact* (continuous) matching cost while performing local optimization. This comparison demonstrated that the use of the exact matching cost is preferable.

The *exact* or analytical computation of the continuous cost can be be done using the analytical description of the silhouette contour $\Gamma \equiv \partial\Omega$ using the green's theorem (the two-dimensional special case of the more general stokes' theorem). The formulas are in closed form, thus providing the exact area in terms of real-valued arithmetic. Given a 2-valued function of two real variables $F(x, y) = (F_x(x, y), F_y(x, y))$ such that its divergence $\nabla \cdot F \equiv \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y}$ equals f_c. The Green's divergence theorem states that we have the equality:

$$\iint_{\Omega} f_c(x, y)dxdy = \oint_{\Gamma=\partial\Omega} < F(s), \hat{n}(s) > ds \tag{3.111}$$

where $\hat{n}$ denotes the outward unit normal to Γ, ds the Euclidean arc length element and the circle in the right-hand integral indicates that the curve is closed. We choose $F_y = 0$ and

$$F_x(x, y) = \int_{t=0}^{x} f_c(t, y)dt \tag{3.112}$$

We have $F(s) = F_x(s).\hat{x}$. If we use the shifted nearest neighbor interpolation then F_x can be rewritten as:

$$F_x(x,y) = \sum_{u=0}^{\lfloor x \rfloor - 1} f(u, \lfloor y \rfloor) + (x - \lfloor x \rfloor) f(\lfloor x \rfloor, \lfloor y \rfloor) \tag{3.113}$$

Using this function the matching cost becomes:

$$L(\theta) = K + \iint_{\Omega} f_c(x,y)dxdy = K + \oint_{\Gamma=\partial\Omega} F_x(s) < \hat{n}(s).\vec{x} > ds \tag{3.114}$$

The integral over Ω is reduced to an integral over the silhouette outline Γ, which is composed of line segments and arcs of ellipses. We will now detail an efficient method to compute the contributions to the integrals due to line segments. Because we will ultimately avoid the use of ellipses by converting them into polygons, we will only sketch a method that could be used to compute the contributions due to arcs of ellipses.

3.5.1 Line segments

Let consider a segment $\overline{q_k q_{k+1}}$ whose extremities are $q_k \equiv (x_k, y_k)$ and $q_{k+1} \equiv (x_{k+1}, y_{k+1})$ (like the segment $\overline{ab}$ in [Figure (3.19.a)]) The contribution of the segment $\overline{q_k q_{k+1}}$ to the matching cost is:

$$C_k \equiv \int_{\overline{q_k q_{k+1}}} F_x(s) < \hat{x}, \hat{n_k} > ds \tag{3.115}$$

The normal vector of the segment $\hat{n_k}$ writes $\hat{n_k} = J(q_{k+1} - q_k)/l_k$ with $J = \left[\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right]$ and $l_k \equiv |q_k - q_{k+1}|$ the length of the segment.

$$< \hat{x}, \hat{n_k} >= (y_{k+1} - y_k)/l_k$$

We can parameterize each segment linearly using a parameter $t \in [0,1]$:

$$\overline{q_k q_{k+1}} = \{(1-t)q_k + tq_{k+1} \mid t \in [0,1]\} \tag{3.116}$$

Using this parameterization for each segment, the matching cost rewrites:

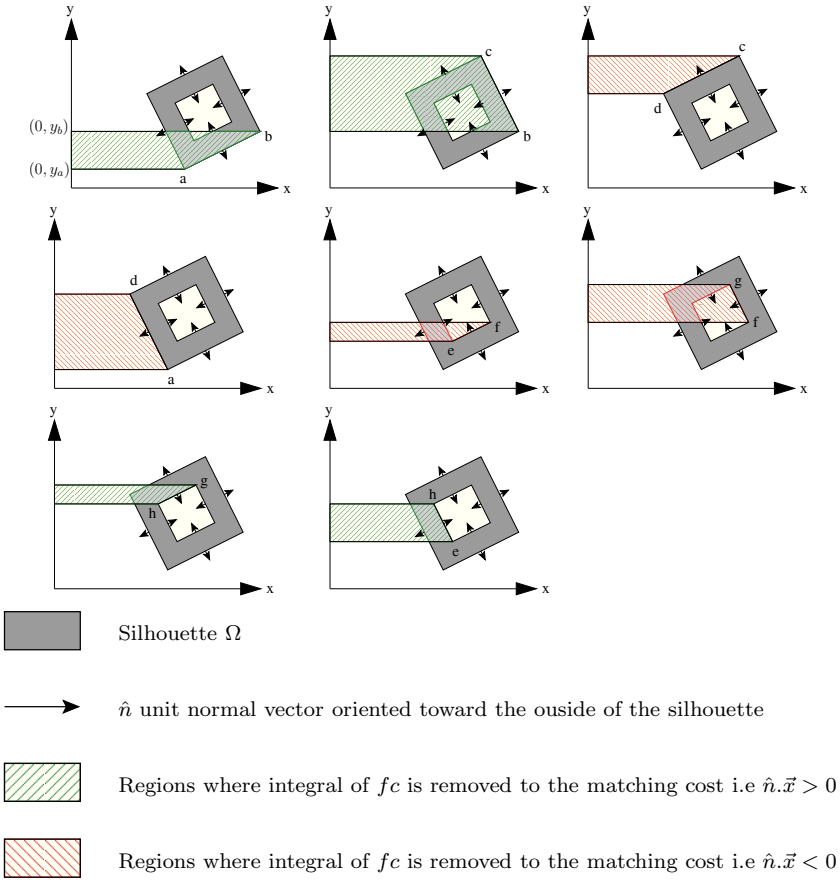
$$C_k = (y_{k+1} - y_k) \int_0^1 F_x((1-t)q_k + tq_{k+1})dt \tag{3.117}$$

Supposing $y_k < y_{k+1}$, the segment can also be parameterized using the following curve

$$r : \begin{array}{ccc} [y_k, y_{k+1}] & \to & \mathbb{R}^2 \\ v & \mapsto & (\alpha v + \beta, v) \end{array} \tag{3.118}$$

with $\alpha = (x_k - x_{k+1})/(y_k - y_{k+1})$ and $\beta = x_k - \alpha y_k$. We get $\overline{q_k q_{k+1}} = r([y_k, y_{k+1}])$. Using this parameterization of the segment, its contribution rewrites:

$$C_k = \int_{v=y_k}^{y_{k+1}} F_x(r(v)) < \hat{n}_k, \vec{x} > |r'(v)|dv \tag{3.119}$$



Descriptio 3.19: trapezoid integrals

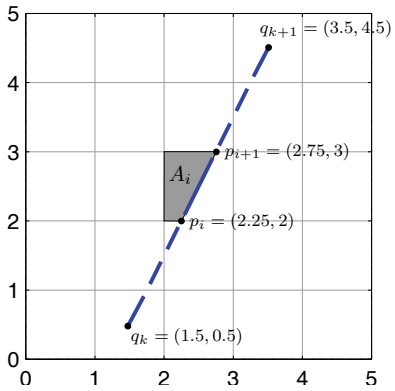
Assuming that we are walking counterclockwise around the silhouette outline, the outside normal $\hat{n}$ points towards the right while going from q_k to q_{k+1} (i.e. the inside of the silhouette is on the left when going from q_k to q_{k+1}). Because $y_k < y_{k+1}$ one can show that the outside normal points toward the right and we have $< \hat{n}_k, \vec{x} >$ positive. With some calculus one can show that we have $< \hat{n}_k, \vec{x} > |r'(v)| = 1$ and we get:

$$\begin{aligned} C_k &= \int_{v=y_k}^{y_{k+1}} F_x(r(v))dv \\ &= \int_{v=y_k}^{y_{k+1}} F_x(\alpha v + \beta, u)du \\ &= \int_{v=y_k}^{y_{k+1}} \int_{u=0}^{\alpha v+\beta} f_c(u, v)dudv \end{aligned} \tag{3.120}$$

The last equation allows to re-interpret easily the contribution of the segment to the contour integral. It corresponds to the integral of f_c in the trapezoidal region on the left of the segment [Figure (3.19)] i.e with corners of coordinates

(x_k, y_k),(x_{k+1}, y_{k+1}),$(0, y_{k+1})$, and $(0, y_k)$. If the edge is on a hole boundary, the contribution of the segment to the contour integral is the integral of $-f_c$ in the trapezoidal region on the left of the segment.

Depending on the interpolation scheme (shifted-nearest neighbor or bilinear) the function f_c is either constant or quadric on each pixel. In order to compute exactly the integral of F_x along the segment $\overline{q_k q_{k+1}}$ one needs to split this line segment into sub-pixel fragments. An example of such a fragmentation in shown in [Figure (3.20)]. Each subpixel fragment is contained in an unit square pixel $[x, x+1] \times [y, y+1]$, with $(x, y) \in L_d$. This is done by clipping the segment against each pixel using for example the method introduced in [Sutherland 1974]. However this is likely to be quite slow. We propose an new efficient algorithm to clip a line segment against a pixel grid whose pseudo code is provided in 5. The segment $\overline{q_k q_{k+1}}$ is clipped by computing its intersections with the pixel grid that is composed of vertical lines with integer x coordinates and horizontal lines with integer y coordinates. The intersections are detected in an order such that their distance from q_k is increasing. Each segment whose extremities are two successive intersections with the pixel grid is contained in a pixel square.



Descriptio 3.20: Sub-pixel fragmentation of a segment. Example with $q_k = (1.5, 0)$ and $q_{k+1} = (3.5, 4.5)$

Once the segment is partitioned into sub-pixel fragments we can easily perform integration on each fragment. Let $p_i \equiv (x_i, y_i)$ and $p_{i+1} \equiv (x_{i+1}, y_{i+1})$ be the extremities of a subpixel segment of $\overline{q_k q_{k+1}}$. We have $\overline{p_i p_{i+1}} \subset \overline{q_k q_{k+1}}$. The two extremities p_i and p_{i+1} belong to the same pixel which means that we have

$$\begin{cases} (x_i, y_i) & \in & [k, k+1] \times [l, l+1] \\ (x_{i+1}, y_{i+1}) & \in & [k, k+1] \times [l, l+1] \end{cases} \tag{3.121}$$

with $k = \lfloor (x_i + x_{i+1})/2 \rfloor$ and $l = \lfloor (y_i + y_{i+1})/2 \rfloor$. Note that we may have $\lceil x_i \rceil \neq \lceil x_{i+1} \rceil$ of $\lceil y_i \rceil \neq \lceil y_{i+1} \rceil$ if x_i or y_i are integers. We denote $c^i \equiv (x_c^i, y_{ci})$ the center of the i^{th} subpixel segment. we have $x_{ci} = (x_i + x_{i+1})/2$ and $y_{ci} = (y_i + y_{i+1})/2$. We introduce the two variables $\Delta_{yi} \equiv y_{i+1} - y_i$ and $\Delta_{xi} \equiv x_{i+1} - x_i$.

3.5.1.1 Nearest neighbor interpolation

If we use the shifted nearest neighbor interpolation with the kernel k_n, then the function f_c is constant within each unit square corresponding to a pixel. Furthermore, the function F_x is linear within each of these squares. Because p_i and $p_i + 1$ belong to the same pixel and we use the shifted nearest neighbor interpolation, the function $u \mapsto F_x(r(u))$ is linear on the interval $[y_i, y_{i+1}]$ and we can evaluate the integral by simply evaluating the center point c^i of the interval:

$$\begin{aligned} C_i &= (y_{i+1} - y_i) F_x\big(r((y_i + y_{i+1})/2)\big) \\ &= \Delta_{yi} F_x(x_{ci}, y_{ci}) \end{aligned} \tag{3.122}$$

We get from eqn 3.113:

$$C_i = (y_{i+1} - y_i) \sum_{u=0}^{\lfloor x_c \rfloor - 1} f(u, \lfloor y_c \rfloor) + \underbrace{(y_{i+1} - y_i)(x_c - \lfloor x_c \rfloor)}_{A_i} f(\lfloor x_c \rfloor, \lfloor y_c \rfloor) \tag{3.123}$$

The value of A_i is the area of the intersection of the trapezoidal region on the left of the fragment $\overline{p_i p_{i+1}}$ and the pixel as shown in [Figure (3.20)].

Consequently, our algorithm could be easily adapted to implement the antialiasing method we referred as the "unweighted area sampling" in the previous section. After meticulous search of similar algorithm in the literature, it appeared that our method bears similarities with the one proposed in [Cheng 1992] that adapts the Bresenham [Bresenham 1965] line rasterization method using two error variables. However, our method that clips segment on pixel grids appears to be simpler. The source code can be found on the Graphic Gems Repository [4]. Our method also bears similarities with antialiasing methods proposed in [Pitteway 1980] [5],[Wu 1991] and [Fujimoto 1983]. These algorithms adapt the Bresenham [Bresenham 1965] rasterization line algorithm to perform antialiasing. These algorithms differ from our contribution because they consists in clipping the line segment only against vertical lines of the pixel grid or only against horizontal lines, depending on the direction of the line segment. The resulting approximate coverage agrees with the exact coverage only when the edge does not cross a line on the pixel grid that is orthogonal to the set of lines chosen for the clipping. Furthermore these algorithms do not handle line ends properly.

3.5.1.2 Bilinear interpolation

From the equation eqn.3.108 that defines the bilinear interpolation and the equation eqn.3.112 that defines F_x we get:

[4] http://tog.acm.org/resources/GraphicsGems/gemsiii/edgeCalc.c

[5] A pseudo code can be found here http://www710.univ-lyon1.fr/~jciehl/Public/educ/ENS/2003/antialiassage.pdf

$$\begin{aligned} F_x(x,y) =& \varepsilon(y) \left(\sum_{u=0}^{\lfloor x \rfloor} f(u, \lceil y \rceil) - \frac{1}{2} f(0, \lceil y \rceil) - \frac{1}{2} f(\lfloor x \rfloor, \lceil y \rceil) \right) \\ &+ (1 - \varepsilon(y)) \left(\sum_{u=0}^{\lfloor x \rfloor} f(u, \lfloor y \rfloor) - \frac{1}{2} f(0, \lfloor y \rfloor) - \frac{1}{2} f(\lfloor x \rfloor, \lfloor y \rfloor) \right) \\ &+ \varepsilon(x) f_c(\lfloor x \rfloor + \frac{\varepsilon(x)}{2}, y) \end{aligned} \tag{3.124}$$

The contribution of the i^{th} sub-pixel fragment can be shown to be:

$$\begin{aligned} C_i = \Delta_{yi} F_x(\lfloor x_{ci} \rfloor, y_{ci}) + c_{00} f(\lfloor x_{ci} \rfloor, \lfloor y_{ci} \rfloor) + c_{10} f(\lfloor x_{ci} \rfloor + 1, \lfloor y_{ci} \rfloor) \\ + c_{01} f(\lfloor x_{ci} \rfloor, \lfloor y_{ci} \rfloor + 1) + c_{11} f(\lfloor x_{ci} \rfloor + 1, \lfloor y_{ci} \rfloor + 1) \end{aligned} \tag{3.125}$$

with c_{00},c_{10},c_{01} and c_{11} defined as:

$$c_{11} \equiv \frac{\Delta_{xi} \Delta_{yi}^2}{12} \varepsilon(x_{ci}) + \frac{\Delta_{yi} \Delta_{xi}^2}{24} \varepsilon(y_{ci}) \tag{3.126}$$

$$c_{10} \equiv -c_{11} + \frac{\Delta_{yi} \Delta_{xi}^2}{24} \tag{3.127}$$

$$c_{01} \equiv -c_{11} + \frac{\Delta_{xi} \Delta_{yi}^2}{12} \tag{3.128}$$

$$c_{00} \equiv -c_{10} - \frac{\Delta_{xi} \Delta_{yi}^2}{12} \tag{3.129}$$

These equations are quite tedious to derive and we used the Maple symbolic computation tool. Note that a method to integrate 2D polynomials into polygons have been proposed in [Tumblin 2006, Steger 1996, Liggett 1988, Singer 1993, Strachan 1990] and their result might be useful to extend the integration in the silhouette for other polynomial interpolation schemes such as bi-cubic interpolation The method proposed in [McCool 1995] to render antialiased triangles is also of interest.

3.5.2 Ellipsoid arcs

Like for line segment edges we need to clip the ellipse arcs to the pixel grid in order to compute their contribution to the matching cost. An arc is defined by a ellipse matrix Q and a ordered couple of extremities (a, b) with a and b 2D points on the ellipse. We assume that the two extremities are ordered such that the arc is joining a and b by traversing the ellipse in the counterclockwise direction. In order to clip the arc onto the pixel grid we need first to decompose the arc into a set of monotonic arcs. This is done by computing the extremal points of the ellipse in the x and y directions. The left-most point, (x_l, y_l), and right-most point, (x_r, y_r), of the ellipse are points on the ellipse where the normal to the curve is aligned with the x axis:

$$\frac{\partial}{\partial y}([x, y, 1] Q [x, y, 1]^T) = 0 \Leftrightarrow Q_{12} x + Q_{22} y + Q_{23} = 0 \tag{3.130}$$

The extremal point in the x direction belongs to the ellipse and the two points point should verify the following system:

$$\begin{cases} Q_{11}x^2 + Q_{22}y^2 + 2Q_{12}xy + 2Q_{13}x + 2Q_{23}y + Q_{33} = 0 \\ Q_{12}x + Q_{22}y + Q_{23} = 0 \end{cases} \tag{3.131}$$

Using the second line we can eliminate y from the first line.

$$\underbrace{(Q_{22}Q_{11} - Q_{12}^2)}_{\alpha}x^2 + \underbrace{2(Q_{13}Q_{22} - Q_{12}Q_{23})}_{\beta}x + \underbrace{Q_{33}Q_{22} - Q_{23}^2}_{\gamma} = 0 \tag{3.132}$$

The two solutions of the system are the found easily:

$$x_l = (-\beta - \sqrt{\beta^2 - 4\alpha\beta})/(2\alpha) \tag{3.133}$$
$$y_l = (-Q_{12}x_l + Q_{23})/Q_{22} \tag{3.134}$$
$$x_r = (-\beta + \sqrt{\beta^2 - 4\alpha\beta})/(2\alpha) \tag{3.135}$$
$$y_r = (-Q_{12}x_r + Q_{23})/Q_{22} \tag{3.136}$$

The two extremal points in the y direction are obtained with the same equation by swapping the x and y coordinates.

Using the extremal points of the ellipse in the x and y direction we can decompose the ellipse arc in monotonic subparts. Then we clip each monotonic part of the ellipsoid with the pixel grid using an algorithm that is very similar to the algorithm that clips a segment on a grid. We need exact floating point intersection and thus cannot use methods proposed in [McIlroy 1992, Fellner 1993a, Fellner 1993b]. The pseudo code of the method that clips the ellipse arc onto the pixel grid is depicted in alg.6.

Then we can compute the contribution of each subpixel fragment. This require to compute area A_i of the intersection of the region at the left of the i^{th} ellipse fragment with the pixel. We denote c the center of the segment $\overline{ab}$. Using the nearest neighbor interpolation kernel k_n the cost associated to the i^{th} sub-pixel arc writes

$$C_i = (y_{i+1} - y_i) \sum_{u=0}^{\lfloor x_c^i \rfloor - 1} f(u, \lfloor y_c^i \rfloor) + A_i f(\lfloor x_c^i \rfloor, \lfloor y_c^i \rfloor) \tag{3.137}$$

the region A_i can be decomposed into a trapezoidal region at the left of the line segment $\overline{ab}$ connecting the two endpoints of the arc and the region bounded by the elliptical arc and the line segment $\overline{ab}$.

3.5.3 Approximating filled-ellipses by polygons

As one can see, the use of ellipses make the computation of the silhouette contour Γ and of the matching cost difficult. The computation of the contribution of ellipse arcs to the matching cost is slow due to repeated use of the square-root function (see alg.6).

But the main difficulties using ellipses are yet to come, when deriving the gradient of the matching cost with respect to the hand pose parameters. The differentiation of the terms in the matching cost due to the ellipses arcs is quite challenging. Two solutions can be considered. The first one consists in converting each ellipsoid in to a convex polyhedron. The second solution is to convert each ellipse (resulting from projection of the ellipsoids) into a convex polygon, before computing the silhouette Γ. Compared to the second solution, the first is simpler could produce a matching cost with more local minima. This can be understood if we consider the case of a prolate spheroid that is discretized into a polyhedron. When the prolate spheroid rotates about its main axis, the projected ellipse remains unchanged. That is also the case for its polygonal approximation and the matching cost. When the polyhedron rotates about the main axis, the silhouette (a convex polygon) slightly moves and the matching cost is not constant but oscillates.

Let consider an ellipse:

$$A = \{(x, y) \in \mathbb{R}^2 | [x, y, 1] C [x, y, 1]^T \leq 0\} \tag{3.138}$$

This ellipse could be approximated with a N edges polygon. To this end, we find the center μ of the ellipse. The center corresponds to the extrema of the 2-variables scalar function $x, y \mapsto [x, y, 1] C [x, y, 1]$ and is found by solving the following system:

$$C_{1:2,1:2}\mu = -C_{1:2,3} \tag{3.139}$$

We find the direction and length of main axis of the ellipse by diagonalizing the submatrix $C_{1:2,1:2}$ using eigenvalue decomposition:

$$C_{1:2,1:2} = VDV^T \text{ avec } VV^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ D = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix},\ d_1 \geq d_2 \tag{3.140}$$

The directions of the main axis correspond to the columns of the matrix V. the 2-dimensional vector $V_{1:2,1}$ corresponds to the direction of the small axis and $V_{1:2,1}$ corresponds to the one of the great axis. The lengths a and b of the small and great axes are obtained by:

$$a = \sqrt{-\beta/d_1},\ b = \sqrt{-\beta/d_2} \tag{3.141}$$

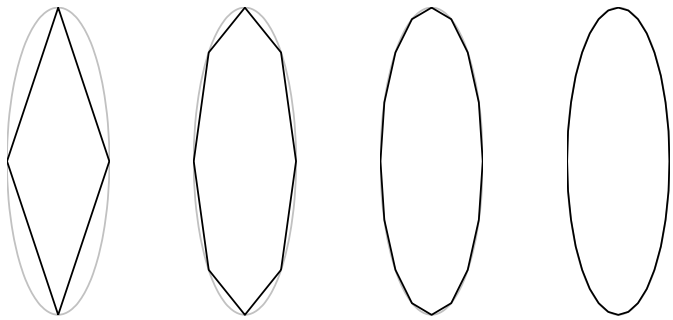
With $\beta = Q_{3,3} + \mu^T Q_{1:2,3}$.

We approximate the ellipse with a polygon with N vertices defined by:

$$p_n = V \times [a\ cos(2\pi n/N), b\ sin(2\pi n/N), 1]^T + \mu,\ n \in \{0, \ldots, N-1\} \tag{3.142}$$

The approximations of the ellipse $x^2 + \left(\frac{y}{3}\right)^2 = 1$ with $N = 4$, $N = 8$, $N = 16$ and $N = 32$ are illustrated in [Figure (3.21)]. We choose $N = 16$ during experimentations.

Note that this approximation is continuous (when $d_1 \neq d_2$), and therefore if the ellipse moves continuously, then the polygon vertices move continuously as well. The case $d_1 = d_2$ (when the ellipsoid projects onto a circle) remains problematic. Another discretization scheme based on the extremal points in a set of fixed direction



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

Descriptio 3.21: Approximating an ellipse by a polygon with (a) 4 edges (b) 8 edges (c) 16 edges (d) 32 edges

could help to avoid such degenerated case. Given a direction $\vec{d} \in \mathbb{R}^2$ the extremal point of the ellipse is the point p^* defined by

$$p* \equiv argmax_p < \vec{d}, p > \text{ s.t } p \in A \tag{3.143}$$

The approximation of ellipses by polygons induces an approximation of the matching cost. This is not problematic for the optimization procedure because such an approximation does not result in discontinuity and non-differentiability of the matching cost (this is not true for the degenerated case $d_1 = d_2$).

Once the ellipses are converted into polygons, we need to compute the silhouette outline defined as the boundary of the union of these polygons. For each polygon we add the list of intersection with other polygon as vertices and remove all edges that are not part of the silhouette outline. The hand silhouette outline is composed of a polyline oriented counterclockwise (that we referred as the external outline) and a set (generally empty) of polylines oriented clockwise that define the boundaries of holes in the silhouette.

$$\Gamma = \bigcup_{k=1}^{K} \overline{q_k q_{k+1})} \tag{3.144}$$

We use the notation q_{k+1} to denote the next vertex succeeding the vertex q_k in the polyline it belongs to (which might be the silhouette external outline or the boundary of a hole in the silhouette). The vertices $(q_k)_{k=1}^K$ of these polylines are either vertices of the polygons obtained after polyhedra projection and ellipse-to-polygon conversions or are intersection points between two of such polygons.

3.6 The Matching Cost Derivatives

In order to use efficient local optimization techniques we need the gradient of the matching cost L with respect to the hand pose parameters θ. The gradient can

be obtained by performing numerical differentiation using the definition of the directional derivative:

$$\frac{\partial L}{\partial \theta_i} = \lim_{\varepsilon \to 0} \frac{L(\theta + \varepsilon e_i) - L(\theta)}{\varepsilon} \tag{3.145}$$

where θ_i is the i^{th} scalar component of the vector the vector θ and e_i is the vector composed of zeros with the exception of the i^{th} components that equals 1. Note that θ_i differs from $\theta_{\mathbf{i}} \in \Theta_{\mathbf{i}}$ which is a vector of dimension d_i that specifies the pose of bone i with respect to its parent in the kinematic tree. In practice one does not compute the limit but choose ε very (but not too) small. This method is quite slow since it requires $D^* + 1$ evaluations of the matching cost with $D^* = 29$ being the number of component in the vector θ. Furthermore, the approximated gradient becomes imprecise when ε is too small due to the limitation of the floating point representation (roundoff errors). It can be difficult to determine which ε yield a good compromise between the roundoff error and the error due to fact ε is too large for the limit to be well approximated.

A better approach is to compute the gradient exactly using the chain rule. For each successive operation performed in order to obtain the matching cost from a given θ, we need to propagate the derivatives with respect to θ.

3.6.1 Differentiation of the polytope transformation and projection

The 3D word coordinates $[X_j, Y_j, Z_j]$ of a point defining a polytope are obtained from its coordinates $[X_j^0, Y_j^0, Z_j^0]$ in the local frame attached to the bones $a(j)$ using eqn.3.52. This equation is easily differentiated:

$$\frac{\partial [X_j, Y_j, Z_j]^T}{\partial \theta_{\mathbf{i}}} = \frac{\partial K_{a(j)}}{\partial \theta_{\mathbf{i}}} [X_j^0, Y_j^0, Z_j^0]^T \tag{3.146}$$

Then the coordinates of its projection onto the camera are obtained using eqn.3.59. We differentiate the projection equations:

$$\frac{\partial [\hat{x}, \hat{y}, \hat{z}]^T}{\partial \theta_{\mathbf{i}}} = P \frac{\partial [X_j, Y_j, Z_j]^T}{\partial \theta_{\mathbf{i}}} \tag{3.147}$$

and

$$\frac{\partial \left[x_j, y_j\right]}{\partial \theta_{\mathbf{i}}} = \frac{1}{\hat{z}^2} \left[\frac{\partial \hat{x}}{\partial \theta_{\mathbf{i}}} \hat{z} - \hat{x} \frac{\partial \hat{z}}{\partial \theta_{\mathbf{i}}}, \frac{\partial \hat{y}}{\partial \theta_{\mathbf{i}}} \hat{z} - \hat{y} \frac{\partial \hat{z}}{\partial \theta_{\mathbf{i}}} \right] \tag{3.148}$$

3.6.2 Differentiation of the ellipsoid transformation and projection

While projecting an ellipsoid composing the hand surface model, the first step is to compute the matrix of the ellipsoids Q from Q_0 after applying the rigid transformation of the bone it is associated with, using the equation 3.50. We differentiate this equation as follows:

$$\frac{\partial Q}{\partial \theta_{\mathbf{i}}} = \frac{\partial (K_j^{-1})}{\partial \theta_{\mathbf{i}}}^T Q_0 K_j^{-1} + (K_j^{-1})^T Q_0 \frac{\partial (K_j^{-1})}{\partial \theta_{\mathbf{i}}} \tag{3.149}$$

With

$$\frac{\partial(K_j^{-1})}{\partial\theta_{\mathbf{i}}} = -K_j^{-1}\frac{\partial K_j}{\partial\theta_{\mathbf{i}}}K_j^{-1} \tag{3.150}$$

The matrix C defining the ellipse that corresponds the projection of the ellipsoid is obtain in two step using equations eqn. 3.65 and eqn.3.70 which are straightforward to differentiate.

3.6.3 Differentiation of ellipses to convex polygons conversion

Once each matrices C defining an ellipse is computed, each ellipse is approximated by a polygon. The center of the ellipse μ is obtained using eqn.3.139. The derivative $\frac{\partial\mu}{\partial\theta_{\mathbf{i}}}$ is obtained by solving the system:

$$C_{1:2,1:2}\frac{\partial\mu}{\partial\theta_{\mathbf{i}}} = -\frac{\partial C_{1:2,3}}{\partial\theta_{\mathbf{i}}} - \frac{\partial C_{1:2,1:2}}{\partial\theta_{\mathbf{i}}}\mu \tag{3.151}$$

Then the direction and lengths of the main axis are obtained by using eigenvalue decomposition of the matrix $C_{1:2,1:2}$ (eqn.3.140). Differentiation of the eigenvalue decomposition (computing the derivative of eigen values and eigen vectors with respect to the entries of the matrix $C_{1:2,1:2}$) is somehow technical. For simplicity we will use $B \equiv C_{1:2,1:2}$. We need the first order variation of the eigen vectors and eigen values of B. Given the definition of eigen vector we have:

$$(B - d_k I)V_{1:2,k} = [0,0]^T \tag{3.152}$$

Let us determine the derivative with respect to B_{ij}. We have $\frac{dB}{\partial B_{ij}} = e_i e_j^T$ with (e_1, e_2, e_3) being the canonic base of $\mathbb{R}^3$ and therefore we can get:

$$(e_i e_j^T - \frac{\partial d_k}{\partial B_{ij}}I)V_{1:3,k} + (B - d_k I)\frac{dV_{1:3,k}}{\partial B_{ij}} = 0 \tag{3.153}$$

Furthermore the constraint that the eigen vector are of unit length can be differentiated, which lead to $\frac{\partial V_{1:2,k}}{\partial B_{ij}}^T V_{1:2,k} = 0$. This produce a linear system made of four equations with four unknowns. We consider $V_{1:2,k}^T \times$ (3.153):

$$V_{1:2,k}^T(e_i e_j^T - \frac{\partial d_k}{\partial B_{ij}}I)V_{1:2,k} + V_{1:2,k}^T(B - d_k I)\frac{\partial V_{1:2,k}}{\partial B_{ij}} = [0,0]^T \tag{3.154}$$

as $V_{1:2,k}^T(B - d_k I) = [0,0]$ we get:

$$\frac{\partial d_k}{\partial B_{ij}} = \frac{V_{1:2,k}e_i e_j V_{1:2,k}}{V_{1:2,k}{}^T V_{1:2,k}} = V_{ik}V_{jk} \tag{3.155}$$

The first order variation of the eigen vectors is obtained by solving, for each eigen value, the following linear system:

$$\begin{bmatrix} B - d_k I \\ (V_{1:2,k})^T \end{bmatrix}\frac{\partial V_{1:2,k}}{\partial B_{ij}} = \begin{bmatrix} \frac{\partial d_k}{\partial B_{ij}}V_{1:2,k} - e_i V_{jk} \\ 0 \end{bmatrix} \tag{3.156}$$

The derivative of the axes lenghts a and b (eqn.3.141) and the points of the polygon $(p_n)_{n=1}^N$ (eqn.3.142) are easily obtain using the chain rule.

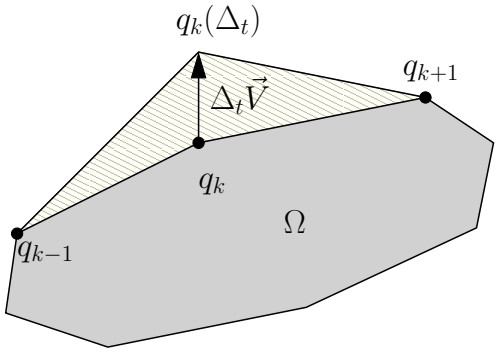
3.6.4 Differentiation of segment intersections

Once ellipses are converted into polygons we need to compute the silhouette outline that is the boundary of the union of the polygons. The vertices q_i of the polylines defining this outline are either vertices of the polygons or are intersection points between two of such polygons. Given the derivatives with respect to the pose parameters vector θ of the extremities of two intersecting segments, we need to compute the derivative of the position of the intersection point. This is done by differentiating eqn.3.90 which is straightforward. Once all intersection are computed we obtain the vertices q_j of the polylines composing the silhouette outlines and their derivatives $\partial q_j / \partial \theta_{\mathbf{i}}$.

3.6.5 Force on silhouette vertices

In order to compute the gradient matching cost with respect to θ we still need the derivatives $\partial L / \partial q_j$. For this, we follow [Unal 2005] but provide two alternative demonstrations. An intuitive demonstration is given here while a more formal demonstration is given in the appendix A.1. Given a vertex of the outline q_k we use q_{k-1} and q_{k+1} to denote the preceding and succeeding vertices in the polyline (which might be the external silhouette outline or a hole in the hand silhouette)

In order to derive the gradient of the matching cost with respect to a vertex of the silhouette, let consider a displacement of the vertex q_k in a direction V with a scaling factor Δ_t i.e $q_k(\Delta_t) = q_k + \vec{V}\Delta_t$ as shown in [Figure (3.22)].



Descriptio 3.22: Displacement of a single vertex q_k and the two triangles that are added (and could be removed for other directions $\vec{V}$) from the silouette

The matching cost is defined as the integral of f_c within the silhouette polygon. When the vertex q_k is displaced, the variation $\Delta_L(V, \Delta_t)$ of the matching cost is obtained by integrating f_c in two triangles $(q_{k-1}, q_k, q_k(\Delta_t)$ and $(q_k, q_{k+1}, q_k(\Delta_t)$

(dashed in [Figure (3.22)] or:

$$\Delta_L(V,\Delta_t) = \int_{t=0}^{1}\int_{u=0}^{t} f_c(q_{k-1}+t(q_k-q_{k-1})+u\Delta_t\vec{V})\det([\Delta_t\vec{V},(q_k-q_{k-1})])dtdu + \int_{t=0}^{1}\int_{u=0}^{t} f_c(q_{k-1}+t(q_k-q_{k-1})+u\Delta_t\vec{V})\det([\Delta_t\vec{V},(q_{k+1}-q_k)])dtdu \tag{3.157}$$

Using $J = \left[\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right]$, we can rewrite the infinitesimal surfaces element in the triangles as follows:

$$\det([\Delta_t\vec{V},(q_k-q_{k-1})])dtdu = \Delta_t\left\langle \vec{V}, J(q_k-q_{k-1})\right\rangle dtdu \tag{3.158}$$

$$\det([\Delta_t\vec{V},(q_{k+1}-q_k)])dtdu = \Delta_t\left\langle \vec{V}, J(q_{k+1}-q_k)\right\rangle dtdu \tag{3.159}$$

and then we can obtain the following derivative:

$$\frac{\Delta_L(\vec{V},\Delta_t)}{\Delta_t} = \left\langle \vec{V}, J(q_k-q_{k-1})\int_{t=0}^{1}\int_{u=0}^{t} f_c(q_{k-1}+t(q_k-q_{k-1})+u\Delta_t\vec{V})dtdu + J(q_{k+1}-q_k)\int_{t=0}^{1}\int_{u=0}^{t} f_c(q_{k-1}+t(q_k-q_{k-1})+u\Delta_t\vec{V})dtdu\right\rangle \tag{3.160}$$

By definition of directional derivatives, for any $\vec{V}$ in $\mathbb{R}^2$ we have:

$$\left\langle \vec{V}, \frac{\partial L}{\partial q_k}\right\rangle = \lim_{\Delta_t\to 0}\frac{\Delta_L(\vec{V},\Delta_t)}{\Delta_t} \tag{3.161}$$

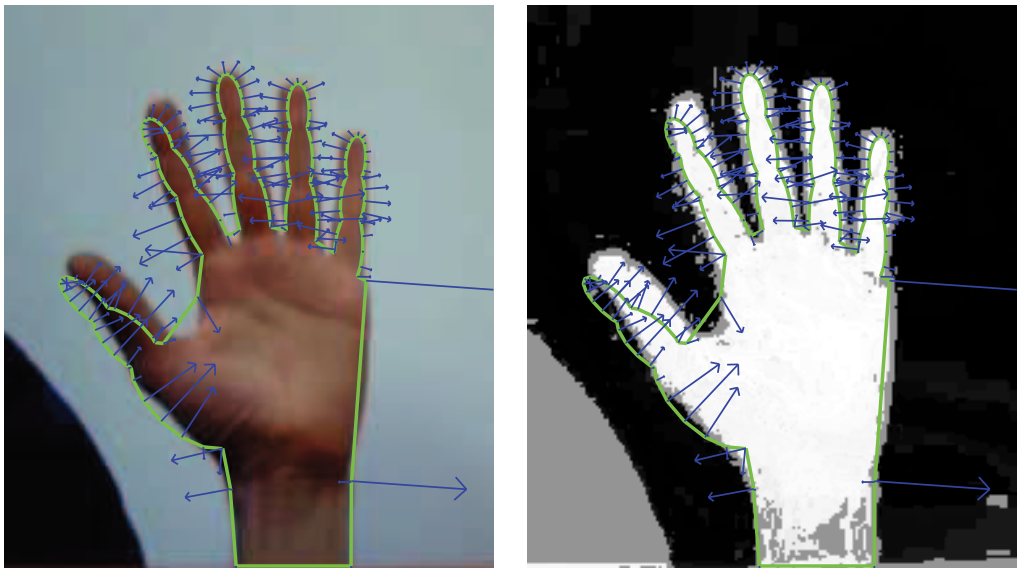
Therefore by identification, the derivative of the functional $L(\theta)$ with respect to a vertex q_k of the polygon is:

$$\begin{aligned}\frac{\partial L}{\partial q_k} =& J(q_k-q_{k-1})\int_0^1 f_c(tq_k+(1-t)q_{k-1})tdt \\ &+ J(q_{k+1}-q_k)\int_0^1 f_c(tq_k+(1-t)q_{k+1})tdt\end{aligned} \tag{3.162}$$

Let us introduce l_k and $\hat{n}_k$ being the length and the normal of the segment $\overline{q_{k-1}q_k}$. The the following condition holds:

$$J(q_k-q_{k-1}) = \hat{n}_k l_k \tag{3.163}$$

One should point out that, for each vertex q_k, the information of the functional f_c is being integrated along its adjacent edges. Using a mechanical metaphor, the resulting vector $\partial L/\partial q_k$ can be interpreted as a 2D data force acting on the silhouette vertex q_k. An example of the resulting 2D force given the silhouette and the image/function f_c is shown in [Figure (3.23)]. One can notice that the forces



Descriptio 3.23: Derivative of the matching cost with respect to the vertices positions that can be interpreted as Forces on the vertices using a mechanical metaphor (a) over-layered on the observed image (b) over-layered on the cost image f

have greater magnitude when adjacent edges are long. This is due to the fact that these forces are proportional to the length l_k and l_{k+1} of the two adjacent edges.

Note that f_c does not have to be continuous everywhere for the matching cost $L(\theta)$ to be differentiable. When using shifted-nearest-neighbor interpolation, f_c is discontinuous for any point that lies on a vertical or an horizontal line of the pixel grid. However the first order derivative $\frac{\partial L}{\partial q_k}$ of the matching cost is defined and continuous as long as none of the two adjacent silhouette edge matches a vertical line or an horizontal line of the pixel grid.

In order to compute the *exact* derivative given by equation 3.162, we need to clip each segment onto the pixel grid and to integrate on each subpixel fragment. By stating that the gradient is *exact* we mean that, given a definition of f_c as nearest neighbor or bilinear interpolation of f, we are able to compute exactly the equation 3.162. The method to compute exactly the gradient is very similar to the method that is used to compute exactly the matching cost. The pseudo code for shifted nearest neighbor interpolation is at alg.9 and the pseudo code for the bilinear interpolation at alg.10 (this second algorithm also compute second order derivatives whose derivation is explained in the section 3.6.6).

Note that one could approximate the integral by sampling uniformly points on each segments and evaluate f_c at each point. The results obtained using this approximation are compared with the ones obtained with the exact computation of the gradient in section 3.7.5.

3.6.6 Second order derivatives

We can approximate the Hessian (second order derivatives) of the matching cost in order to uses quasi-Newton optimization techniques. The second order derivative of the matching cost writes:

$$\frac{\partial^2 L}{\partial\theta_k\partial\theta_l} = \sum_{ij} \frac{\partial q_i}{\partial\theta_k}\frac{\partial^2 L}{\partial q_i\partial q_j}\frac{\partial q_j}{\partial\theta_l} + \sum_i \frac{\partial L}{\partial q_i}\frac{\partial^2 q_i}{\partial\theta_k\partial\theta_l} \tag{3.164}$$

The second term $\sum_i \frac{\partial L}{\partial q_i}\frac{\partial^2 q_i}{\partial\theta_k\partial\theta_l}$ is tedious to derive and can be neglected. The approximation of the Hessian is then defined as:

$$\tilde{H}_\theta(k,l) \equiv \sum_{ij} \frac{\partial q_i}{\partial\theta_k}\frac{\partial^2 L}{\partial q_i\partial q_j}\frac{\partial q_j}{\partial\theta_l} \tag{3.165}$$

This approximation might not be very precise, but the optimization method is robust to approximations errors on the Hessian. The experimental results also tend to validate that the approximated Hessian remains useful to speed up the convergence rate, despite it is not the exact Hessian.

The second order derivative of the functional $L(\theta)$ with respect to a vertex of the polygon is derived by differentiating the equation eqn.3.162. For any pair of vertices (q_i, q_j) of the silhouette polygon the second order derivative $\partial^2 L/\partial q_i q_j$ is a 2 by 2 matrix. If the two vertices are not connected by an edge then $\partial^2 L/\partial q_i q_j = 0_{2\times 2}$. As a consequence, if we gather all the derivatives $\frac{\partial^2 L}{\partial^2 q_j}$ into a single matrix M composed of 2×2 sub-matrices $M_{2i:2i+1,2j:2j+1}$ corresponding to the 2×2 matrices $\partial^2 L/\partial q_i q_j$, then this matrix M is expected to be sparse and symmetric.

If two vertices are connected ($i = j+1$), then by setting $k = j$ in the equation eqn.3.162 and differentiating, we obtain:

$$\begin{aligned}\frac{\partial^2 L}{\partial q_{j+1}\partial q_j} =& + J\int_0^1 f_c((1-t)q_{j+1} + tq_j)tdt \\ &+ J(q_{j+1} - q_j)\int_0^1 \nabla f_c((1-t)q_{j+1} + tq_j)(1-t)tdt\end{aligned} \tag{3.166}$$

Because of the symmetry of derivatives we also get $\left[\frac{\partial^2 L}{\partial q_j\partial q_{j+1}}\right] = \left[\frac{\partial^2 L}{\partial q_{j+1}\partial q_j}^T\right]$. Note that the matrix $\left[\frac{\partial^2 L}{\partial q_j\partial q_{j+1}}\right]$ can also be derived from eqn.3.162 by taking $k = j+1$. The equivalence between the expression we would obtain can be shown using integration by parts. Note that the function f_c should be differentiable almost everywhere along the two adjacent edges in order to have the second order derivative being properly defined. One should recall the bilinear interpolation f_c is not differentiable for any point that lies on a vertical or an horizontal line of the pixel grid. Therefore the second order derivative $\partial^2 L/\partial q_{j+1}\partial q_j$ is defined for the bilinear interpolation if and only if the silhouette edge $\overline{q_j q_{j+1}}$ does not match a vertical or an horizontal line of the image grid.

The second order derivative when $i = j$ writes

$$\begin{aligned}\frac{\partial^2 L}{\partial^2 q_j} =& J\int_0^1 f_c((1-t)q_{j-1}+tq_j)tdt - J\int_0^1 f_c((1-t)q_{j+1}+tq_j)tdt \\ &+ J(q_j - q_{j-1})\int_0^1 \nabla f_c((1-t)q_{j-1}+tq_j)t^2dt \\ &+ J(q_{j+1} - q_j)\int_0^1 \nabla f_c((1-t)q_{j+1}+tq_j)t^2dt\end{aligned} \tag{3.167}$$

The symmetry of the 2×2 matrix $\partial^2 L/\partial^2 q_j$ is not obvious when considering this equation but can be proved using integration by parts. The second order derivative $\partial^2 L/\partial^2 q_j$ is defined for bilinear interpolation and if none of the two adjacent silhouette edge $\overline{q_{j-1}q_j}$ $\overline{q_j q_{j+1}}$ matches a vertical or an horizontal line of the image grid.

The pseudo-code to compute first and second order derivative for bilinear interpolation of f is provided in alg.10

Note that we could approximate the integrals by finite sums through uniform sampling of points along each segment and evaluation of f_c and ∇f_c at these locations. However the equation 3.167 does not exhibit obvious symmetry and the approximate Hessian we would obtain is not symmetric. Therefore we would have to impose the symmetry of the Hessian by averaging the computed Hessian matrix and its transpose.

The Hessian we obtain using the exact integration along the edges using the algorithm alg.10 is symmetric up to round-off errors. The experimental comparison done in section 3.7.5 to proves that the use of the exact integration improves the results. Having exact computation is also very useful from a practical point of view when it comes to test if there are not error in the manner the equations have been implemented. This point is also discussed in section 3.7.5.

Note that in the context of active contour, several authors [Hintermüller 2007, Burger 2003, Hintermüller 2004, Bar 2009] proposed the use of second order derivatives to accelerate the convergence. Their formulations are done for general continuous and differentiable curves, generally require Gï¿½teaux derivatives in the derivation and are implemented using level set methods. In our case we have an explicit formulation of the silhouette outline Γ as a polygon and, as a consequence, the second order derivatives are simpler to obtain.

3.7 Pose estimation

Given an observed image we can formulate the problem of estimating the most-likely hand pose to be the vector θ_{optim} that minimize the matching cost $L(\theta)$ while satisfying the constraints on the pose parameters i.e:

$$\begin{aligned}\theta_{\text{optim}} &= \underset{\theta}{argmin}\, L(\theta) \\ &\text{s.c } A\theta \leq b\end{aligned} \tag{3.168}$$

with $A\theta \leq b$ the set of linear inequalities defined in eqn.3.42. The fact that the function $L(\theta)$ is neither linear nor quadratic results to a nonlinear programming problem. Solving exactly this problem, i.e. finding the global minimizer of $L(\theta)$ restricted to the set of valid hand pose cannot be done in the general case. Due to the dimensionality of θ one cannot perform exhaustive search or perform exact global minimization.

In the context of tracking, one can assume that a rough approximation of the hand pose from previous frame is available. Then iterative refinement through local exploration of the matching cost function could produce satisfactory results. The problem of estimating the hand pose in the first frame remains problematic and is not withing the scope of this thesis, therefore it is done through user aided initialization.

Despite the fact that a rough initialization can generally be derived from estimates in previous frames, the iterative search may lead to local minima. In order to improve the robustness we perform local search starting from several pose candidates. This is done by combining the particle filter with local search, which has been proposed in [Bray 2004b]. Given an initial hand pose that is provided by the particle filter, we iteratively refine the pose estimate by exploring locally the matching cost function toward reducing its value. Several methods exist in order to perform local minimization of differentiable functions under linear constraints. In the context of our method, the use of the gradient of the matching cost and its approximate Hessian . For this purpose we adopt three methods and compare their performances. We will first describe efficient local search procedures given an initial estimate, and then describe the *smart particle filter* that allows to propagate multiple hypothesis.

3.7.1 Sequential Quadratic Programing with BFGS update

The first method we test is a classic quasi newton method implemented in the function *fmincon* from the Matlab®optimization toolbox (Matlab®release 7.5.0.342). Two algorithms are implemented in *fmincon*: a *medium scale* and a *large scale* method. We use the *medium-scale* method because the *large scale* method does not handle linear inequality constraints of the form $A\theta \leq b$. Unfortunately, the medium scale method does not allow using an approximate Hessian provided by the user. The Matlab optimization functions do not allows to constraint the input vector to be restricted in a manifold Θ that differs from $\mathbb{R}^{D*}$ (with D^* the size of the vector θ) without resorting to non-linear inequalities. If we use a quaternion to parameterize the global orientation of the hand, then it has to remain a unit quaternion. This can be formalized using a non-linear inequality constraint $\|q\| = 1$. Such a non-linear inequality can be handled by the Matlab's minimization method with the drawback that it reduces the convergence rate. Therefore we use the Euler angles to describe the global orientation of the hand instead of using a quaternion and we obtain $\Theta = \mathbb{R}^{D*}$. The *medium-scale* method uses a sequential quadratic

programming (SQP) method. At each iteration it minimizes a convex quadratic form $m_k(\theta)$ that approximates locally the function being minimized

$$\theta^* \leftarrow \underset{\theta}{argmin}\, m_k(\theta) \text{ s.c } A\theta \leq b \tag{3.169}$$

with:

$$m_k(\theta) = L(\theta_k) + \langle \nabla L(\theta_k), \theta - \theta_k \rangle + \frac{1}{2} \langle \theta - \theta_k), H_k(\theta - \theta_k) \rangle \tag{3.170}$$

Here θ_k denote the estimated hand pose at the k^{th} iteration and not its k^{th} component of the parameter describing the pose of the k^{th} bone. The matrix H_k is a positive definite matrix approximate of the Hessian d^2L/d^θ of L at θ_k. The matrix H_k is updated at each iteration using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula 3.171 [Broyden 1970, Fletcher 1970, Goldfarb 1970, Shanno 1970]. The minimization of a quadratic function under linear constraints is referred as a quadratic programing (QP) problem in the literature. Because a QP problem is solved at each iteration, the method is referred as a sequential quadratic programing (SQP) method. Because H_k is not the exact Hessian, the quadratic model $m_k(\theta)$ is only valid up to the first order in a small region around θ_k and therefore the minimizer θ^* is likely to be a poor candidate for the next iteration. In order to obtain a good candidate, a one-dimensional minimization, referred as *line search* is performed along the line ray starting from θ_k passing thought θ^*. Using a *merit function*, this *line search* aims a finding a positive scalar $\alpha_k \in \mathbb{R}^+$ such that the candidate vector $\theta_{k+1} \equiv \theta_k + \alpha_k(\theta^* - \theta_k))$ *sufficiently reduces* the function L. In Matlab®, this line search is done using a method similar to [Powell 1978].

At each iteration the matrix H_k is obtained from the matrix H_{k-1} and the the function gradient at location θ_k using the BFGS formula:

$$H_{k+1} = H_k + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} - \frac{H_k \mathbf{s}_k (H_k \mathbf{s}_k)^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} H_k \mathbf{s}_k}. \tag{3.171}$$

With $\mathbf{s}_k = \alpha_k(\theta^* - \theta_k)$ and $\mathbf{y}_k = \nabla L(\theta_{k+1}) - \nabla L(\theta_k)$. Using this formula, the equation $H_{k+1}(\theta_{k+1} - \theta_k) = \nabla L(\theta_{k+1}) - \nabla L(\theta_k)$ is verified at each iteration, which is what would be expected with the exact Hessian. The Hessian matrix is actually updated only if $\mathbf{y}_k \mathbf{s}_k > 0$), which ensures that the matrix remains positive definite through iterations. A choice has to be made for H_0. Even in the best case, i.e when the function being minimized is a convex quadratic, the quasi-newton requires as many iterations as the dimensions of θ in order to obtain a good approximation of the true Hessian. The most common choice for H_0 is the identity matrix (or a multiple of it) which produces the first search direction to be the steepest one. Such a choice results in very poor Hessian approximations during the first iterations whenever he problem is not well conditioned, i.e when the ratio between the biggest and lowest eigen values of the Hessian is important. Opposite to exact Newton methods, whose behavior does not change when a linear re-parameterization is done on the input vector, the quasi newton method has a behavior that depends on

the parameterization through the choice of the initial matrix H_0. Matlab®does not allows to specify an initial matrix H_0 and so we have to re-parameterize the problem $\theta = Sx$ with S a diagonal matrix. We chose S empirically such that changes on each component of x has about the same amount of variation on the 3D hand position. Near a local minima, the BFGS-SQP method is super-linear [Chen 1996, Liu 2007]. However, when the hand pose is far from a local minima the BFGS method suffers from the fact that the approximation of the cost function cannot be done accurately with a convex quadratic function.

3.7.2 Variable metric descent

The second optimization method is a *variable-metric* method that we proposed in [de La Gorce 2006]. This method bears similarities with the Matlab®method described in the previous section. It also involves the resolution of a convex quadratic program at each iteration, and thus can be considered as a SQP method. It differs from the previous method in two aspects: 1) we do not use linear search anymore and 2) the the quadratic model m_k now approximate the function f_k defined by:

$$f_k : \Delta_\theta \mapsto L\big(S(\theta_k, \Delta_\theta)\big) - L(\theta_k) \tag{3.172}$$

with S the manifold *reduction function* we introduced in section 3.2.1.3. The function f_k models the decrease of the matching cost we would observe if a displacement in the direction $\Delta_\theta/\|\Delta_\theta\|$ with a length $\|\Delta_\theta\|$ is performed. The use of S allows to handle properly a quaternion representation of the global orientation of the hand. The model m_k combines the linear approximation of the cost function (based on the gradient) with a quadratic penalization of the step.

At each iteration we solve the following quadratic programing problem:

$$\Delta_\theta \leftarrow \underset{\theta}{argmin}\, m_k(\Delta_\theta) \text{ s.t } A(\theta + \Delta_\theta) \leq b \tag{3.173}$$

With:

$$m_k(\Delta_\theta) = \langle \nabla L(\theta_k), \Delta_\theta \rangle + \frac{1}{\rho} \langle \Delta_\theta, C_\theta \Delta_\theta \rangle \tag{3.174}$$

The positive coefficient ρ and the symmetric definite matrix C_θ will be defined later. The solution of (eqn.3.173) is obtained using a standard quadratic programming method like the function *quadprog* in Matlab®. The model $m_k(\Delta_\theta)$ approximates to the first order the function f_k only for small $\|\Delta_\theta\|$ and thus $\theta^* \equiv S(\theta_k, \Delta_\theta)$ might be a bad candidate for the next iteration. Instead of using a linear search in order to cope with this problem, we use the positive scalar coefficient ρ in eqn.3.174 that is adapted to control the step length. Indeed, without the linear constraints, the solution of (eqn.3.173) would be given by $\Delta_\theta = \rho C_\theta^{-1} \nabla L(\theta_k)$. The coefficient ρ is adapted such that the decrease of the objective function is close enough to the predicted one. We should have:

$$\frac{f_k(\Delta_\theta)}{m_k(\Delta_\theta)} \geq \mu \tag{3.175}$$

with $\mu = 1/2$. At each iteration, the coefficient ρ is progressively until θ^* sufficiently decreases the function being minimized. The difference with the method described the previous section also appears in the choice of the matrix defining the quadratic term in $m_k(\Delta_\theta)$ i.e H_k in eqn.3.170 and C_θ in eqn.3.174. This matrix is obtain using a BFGS approximation of the Hessian in the previous method. Here we use a quadratic penalization on the step based on the chamfer distance between silhouettes. More precisely, we choose C_θ such that the quadratic term in m_k locally matches (up to the second order) the quadratic chamfer distance between the silhouettes for small variations, i.e.:

$$D_{qc}(\Gamma(\theta), \Gamma(\theta + \Delta_\theta)) = \Delta_\theta^T C_\theta \Delta_\theta + o(\|\Delta_\theta\|^2) \tag{3.176}$$

with $D_{qc}(\Gamma_1, \Gamma_2)$ the quadratic Chamfer distance between two parametric curves that is defined as:

$$D_{qc}(\Gamma_1, \Gamma_2) \equiv \int_{\Gamma_1} min\ _t(\Gamma_1(s) - \Gamma_2(t))^2 ds \tag{3.177}$$

where s a curvilinear coordinate. One can calculate C_θ given the set $Q = (q_k)$ of silhouette vertices. With some calculation, the second order Taylor expansion of D_{qc} around $\Delta_\theta = 0_{29\times 1}$ writes:

$$\begin{aligned} D_{qc}(\Gamma(\theta), \Gamma(\theta + \Delta_\theta)) &= \sum_k l_k \int_0^1 \left((1-t)\Delta_{q_k}.\hat{n}_k + t\Delta_{q_{k+}}.\hat{n}_k\right)^2 dt + o(\|\Delta_\theta\|^2) \\ &= \frac{1}{3}\sum_k l_k \left[(\Delta_{q_k}.\hat{n}_k)^2 + (\Delta_{q_{k+}}.\hat{n}_k)^2 \right. \\ &\left. +(\Delta_{q_k}.\hat{n}_k \times \Delta_{q_{k+}}.\hat{n}_k)^2\right] + o(\|\Delta_\theta\|^2) \end{aligned} \tag{3.178}$$

with $\Delta_{q_k} = \sum_j \frac{\partial q_k}{\partial \theta_j}\Delta_{\theta_j}$, $l_k = \|q_{k+1} - q_k\|$ and $\hat{n}_k = J(q_{k+1} - q_k)/l_k$

The second term on the right side of (eqn.3.178) is quadratic with respect to Δ_θ and therefore can be rewritten under the form of (eqn.3.174). This term penalizes large steps and leads to a natural scaling between rotations and translations. Directions with small influence on the silhouette are penalized less than directions with a greater influence. This penalization can be interpreted as a "natural" metric that measure the amount of change on the silhouette induced by a change in the parameters space. The metric depends on the the pose but not on the parameterization and the penalization is not defined as the euclidean norm of the step but using a metric that may change at each iteration. Therefore our new optimization method can be assimilated to a variable metric gradient descent under linear constraints. Using this alternative norm, we define a "natural" metric and the behavior of our method does not change when a linear re-parameterization is done on the input vector. Probably due to the limitation of the BFGS update method when the search is not close enough to the local minima, this variable-metric method yields

faster convergence rates [Figure (3.25)]. We believe that it could be possible to combine the BFGS approximation of the Hessian with the proposed metric in order to improve efficiency.

3.7.3 Trust-Region method

Unlike the two methods we presented in the two previous section, the third method uses the approximate Hessian $\tilde{H}$ that we defined in the equation 3.165 and implemented in the algorithm 10. We adopt a SQP approach in order to be able to handle easily the linear constraints. We use the approximate Hessian $\tilde{H}$ instead of the BFGS or the variable metric to define the quadratic term in $m_k(\theta)$. We obtain a quadratic form $m_k(\Delta_\theta)$ that locally approximates the function $f_k : \Delta_\theta \mapsto L(S(\theta_k, \Delta_\theta)) - L(\theta_k)$:

$$m_k(\Delta_\theta) = \langle \nabla L(\theta_k), \Delta_\theta \rangle + \left\langle \Delta_\theta, \tilde{H}_\theta \Delta_\theta \right\rangle \tag{3.179}$$

Instead of using a linear search to handle the fact that the model is only local (see the two previous methods) we use the *trust region* approach [Conn 2000]. In this approach, the *trust region* corresponds to the region in which we believe the quadratic model m_k to approximates sufficiently well the function f_k. It is defined as the set of points

$$B_k = \{\Delta_\theta \in \mathbb{R}^{D*} |\ \|\Delta_\theta\|_k \leq \Delta_k\} \tag{3.180}$$

where Δ_k is called the trust-region radius, and $\|.\|_k$ is an iteration-dependent norm.

At each iteration we find the minimum of the approximation m_k in the trust region by solving the problem

$$\Delta_\theta \leftarrow argmin\, m_k(\theta) \text{ s.t } A(\theta + \Delta_\theta) \leq b \text{ and } \Delta_\theta \in B_k \tag{3.181}$$

This problem is denoted as the *trust-region sub-problem* and the technique used to solve it will be detailed latter. Like in the previous methods, the model $m_k(\Delta_\theta)$ approximates to the first order the function f_k only for small $\|\Delta_\theta\|$ and thus $\theta^* \equiv S(\theta_k, \Delta_\theta)$ might be a bad candidate for the next iteration if the trust region radius Δ_k is large. The radius is adapted (increased or decreased) according to whether θ^* is a good candidate or not. If *sufficient decrease* is achieved for the objective function, then the trial point is accepted and is considered in the next iteration. If the model turns out to be a poor predictor of the actual behavior of the objective function, the trial point θ^* is rejected and the trust region is contracted, with the hope that the model provides better prediction in the smaller region. One advantage of the trust region approach over the previous method is the fact that $m_k(\Delta_\theta)$ is not required to be convex, which is the case when $\tilde{H}_\theta$ has negative eigen values. If $\tilde{H}_\theta$ has directions of negative curvature, then the trust region algorithm can take advantage of them directly.

An important aspect of any trust-region method is the shape of the trust region itself which is determined by the norm $\|.\|_k$. Ideally, this shape should reflect the

region where we believe the model approximate the objective function well, that is where the ratio r defined by

$$r(\Delta_\theta) = \frac{f_k(\Delta_\theta)}{m_k(\Delta_\theta)}, \tag{3.182}$$

is close to 1. As explained in [Conn 2000] the ideal shape of trust region can be extremely complicated. The variable metric defined in the previous section, through the matrix C_θ defined equation 3.176, seems to be a relevant choice to define the norm in the trust region definition eqn.3.180, or

$$\|\Delta_\theta\|_k = \Delta_\theta^T C_{\theta_k} \Delta_\theta \tag{3.183}$$

Using this metric the trust region would be an ellipsoid centered around θ_k. Unfortunately the trust region sub-problem with such an ellipsoidal trust region becomes a difficult in the presence of a set of linear constraints ($A(\theta + \Delta_\theta) \leq b$). In order to use standard quadratic programing to solve the trust-region subproblem we define the metric using the L_∞ norm after re-parameterization. To this end we first rewrite the term $\Delta_\theta^T C_k \Delta_\theta$ as a L_2 norm after some re-parametrization. The matrix C_k is positive definite and can be decomposed using an eigen values decomposition with an orthogonal set of normalized vectors V_i and positive eigen values λ_i such that $C_k = \sum_i \lambda_i V_i V_i^T$. We define the matrix T such that each line $T_{i,:}$ writes:

$$T_{i,:} \equiv \sqrt{\lambda_i} V_i^T \tag{3.184}$$

using this matrix we have $\Delta_\theta^T C_k \Delta_\theta = \|T\Delta_\theta\|_2$. We now approximate the L_2 by and L_∞ norm and define the trust region to be:

$$\|\Delta_\theta\|_k \equiv \|T\Delta_\theta\|_\infty \tag{3.185}$$

The trust regionB_k is now a box centered at θ_k whose axis are aligned with the main axis of the ellipsoid $\Delta_\theta^T C_k \Delta_\theta < \Delta_k$ and whose side lengths equal the main axis lengths of the ellipsoids. The advantage of such a trust region is that the constraint $\|\Delta_\theta\|_k < \Delta_k$ now writes as a set of linear constraints $\Delta_k \leq \sqrt{\lambda_i} V_i^T \Delta_\theta \leq \Delta_k$ that are simply added to the constraints $A(\theta + \Delta_\theta) \leq b$ while calling the quadratic program solver.

Note however that solving exactly an indefinite quadratic programing is NP hard in the general case. For example $-\|x\|^2$ subject to $-1 < x_i < 1$ has 2^n minimizers. When $\tilde{H}_{\theta_k}$ is positive semidefinite there exist polynomial time interior point to solve problem

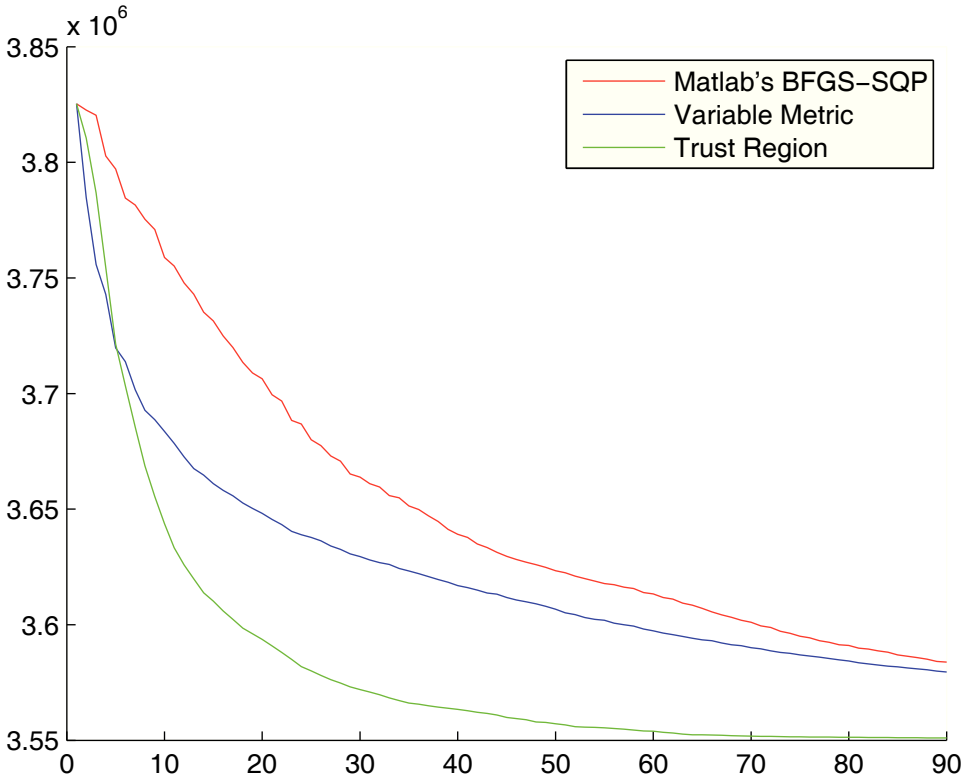
Despite the fact that solving indefinite quadratic programming is NP in the general case, the solutions provided by the Matlab®function *quadprog* are good enough for this third approach to converge faster than the two previous ones.

3.7.4 Comparing the three Optimization methods

We compared the three optimization methods using an image shown in [Figure (3.25)] where the hand is flat with the finger stretched.

We initialized the hand pose manually in order to define a reference hand pose and then we generated 100 hand pose configurations $\tilde{\theta}_1, \ldots, \tilde{\theta}_{100}$ in the vicinity of this reference hand pose by adding noise on the 29 hand parameters. For each sample we reprojected the hand poses in the set of valid hand constraints by solving the quadratic problem:

$$\theta_k \leftarrow \underset{\theta}{argmin} \left\langle \theta - \tilde{\theta}_k), C_\theta(\theta - \tilde{\theta}_k) \right\rangle \text{ s.t } A\theta \leq b \tag{3.186}$$



Descriptio 3.24: Comparison of convergence rates for the Matlab®*fmincon* function that implements a BFGS-SQP method, the variable-metric method and the trust region method

We used each of these samples to initiate a local search using each of the three suggested optimization techniques. For each method we average the hundred curves of the matching cost versus the number of iterations. The resulting curves are shown in [Figure (3.24)]. As we can see, the trust region method is the fastest to converge while the Matlab®*fmincon* (BFGS-SQP) method is the slowest to converge.

By considering the columns 1 and 3 in [Figure (3.25)] we see that the Trust-Region and the Matlab®method seems better at not falling into bad local minima.

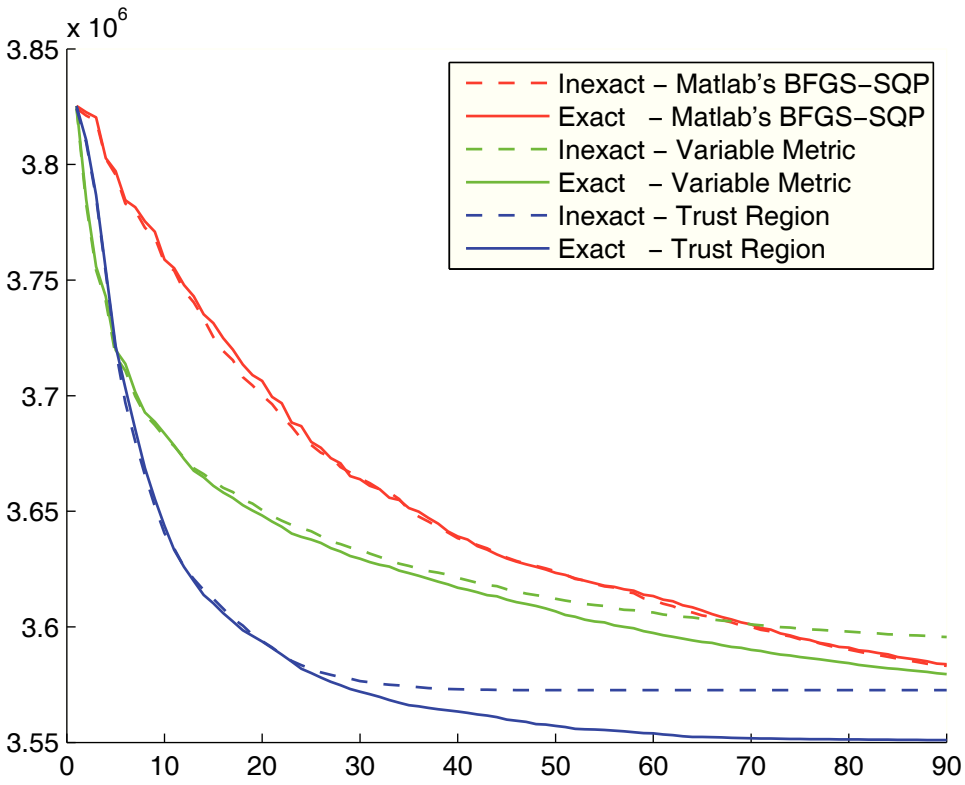
Descriptio 3.25: Each raw corresponds respectively to 1) the initial pose 2) the optimal pose found with the Matlab®function fmincon (BFGS-SQP) after 90 iteration 2) the optimal pose found with the variable-metric method after 90 iterations 3) the optimal pose found with the trust-region method after 90 iterations.

3.7.5 Exact versus Approximate Matching cost and derivatives

3.7.5.1 Performance comparisons

In section 3.4.2 we introduced the continuous matching cost in order to formally define its gradient whose expression is derived in section 3.6.5. Furthermore, for the continuous matching cost and its derivatives, we provided practical methods to compute them exactly (using respectively the algorithms 5 and 10). The adjective *exactly* means that, if we assume f_c to be defined using the bilinear interpolation, then the continuous matching cost and its derivatives are computed exactly without the need to approximate integrals by finite sums or the derivatives by finite differences. The bilinear interpolation of f and the approximation of ellipses by polygons are approximations that preceded the definition of the matching cost, and thus are already included in the continuous matching cost. We will refer to this approach as the *exact* approach.

The need of actually implementing the exact evaluation of the continuous matching cost and its derivatives is not obvious. Indeed, a classical approach in the lit-



Descriptio 3.26: Comparison of convergence rates for the Matlab®*fmincon* function that implements a BFGS-SQP method, the variable-metric method and the trust region method using either the inexact computation or the exact computation of the matching cost and its derivatives

erature consists in 1) formulating a continuous objective function using continuous integrals 2) express its derivatives using the continuous expression and finally 3) implement an approximative evaluation of the energy and its derivatives by replacing integrals by finite sums and derivative by finite differences. Details on the third step are often omitted in the computer vision literature, maybe because these are considered as technical details of minimal interest.

We compare in this section our approach with an approach that we refer to the *inexact* method and that consists in using finite sums and finite differences to replace respectively the integrals and the derivatives. In order to implement an *inexact* evaluation of the matching cost we simply use the Matlab®function *poly2mask* that compute a binary image given a simple polygon. This image equals 1 for the pixel that lies inside the polygon. While computing the derivatives with respect to the silhouette vertices (eqn.3.162), we approximate integrals by finite sums on uniformly distributed points. For each segment $\overline{q_k q_{k+1}}$ we sample N_k

points with N_k a number that roughly equals the length of the segment expressed in pixels and that is defined by $N_k \equiv \lceil \|q_{k+1} - q_k\| \rceil$. We obtain:

$$\begin{aligned}\frac{\partial L}{\partial q_k} \approx & J(q_k - q_{k-1}) \frac{1}{N_{k-1}} \sum_{n=0}^{N_{k-1}-1} f_c((n/N_k)q_k + (1 - n/N_k)q_{k-1})(n/N_k) \\ & + J(q_{k+1} - q_k) \frac{1}{N_k} \sum_{t=0}^{N_k-1} f_c((n/N_k)q_k + (1 - n/N_k)q_{k+1})(n/N_k)\end{aligned} \tag{3.187}$$

The integrals appearing in the second order derivatives (equations 3.166 and 3.167) are replaced by finite sums in the same manner.

We now compare the performance between the *exact* and the *inexact* approach using the same methodology as in section 3.7.4. When we visualize the curves we obtain by averaging the hundred curves corresponding to different random initialization around the reference pose, it clearly appears that the *exact* approach performs better than the *inexact* approach when using the variable metric or the trust region method. This could be explained by the fact that these optimizations method expect that, for a very small step, the first order model is valid i.e the predicted change in the energy should be close the the scalar product of the step and the gradient. When using the *inexact* approach, a small step in the parameters space induces a change of the silhouette binary mask on very few pixels. Therefore the variation on the matching cost is not well predicted by the first order model. For small steps in the parameter space, the silhouette binary mask will not even change. Because the coefficient ρ in eqn.3.174 and Δ_k in eqn.3.180 are decreased until the first model becomes valid, this might cause the method to stop too early during the optimization.

While using the *exact* approach, small step will always induce a small change on the antialiased silhouette image $\tilde{M}$ defined in section 3.109 and thus will induce a change on the matching cost that is well predicted by the first order approximation based on the exact gradient.

When using the *inexact* computation of the second order derivative, we can numerically observe that the Hessian is not symmetric. This due to the approximation made in the discretization. We have to impose its symmetry by averaging the approximate Hessian with its transpose. Otherwise the trust-region method could fail due to the fact that non symmetric matrices have complex eigen values. With the *exact* computation of the Hessian we numerically observe that the Hessian is almost symmetric, and the dissymmetry is only due to the limitation of the floating point representation (round-off errors). We also enforce the symmetry of the Hessian in order to avoid any numerical error due to complex eigen values.

Note that a recent paper [Wilke 2009] proposed an optimization method that do not require to evaluate the matching cost but only its gradient. In particular, the line search uses only the gradient information. It would be interesting to see how well this method performs with the *inexact* gradient in comparison with our *exact* methods.

In the case of the Matlab®method based on the BFGS-SQP method we did not observe a change in the performance. We believe that this is due to the fact that the Matlab®method converges quickly enough to reach regions where the step length has to be decreased, i.e close to a local minima or a critical point. It seems also possible that the Matlab®function has a greater tolerance to errors on the gradient than our variable metric and our trust region implementations.

3.7.5.2 Methodological advantages of the Exact computation

As we said in the previous section, the first order model $m_k(\theta)$ - provided around some pose vector θ_k by the *exact approach* - is exact up the first order. This means that, for a very small step, the variation predicted by the model and the actual matching cost will be very close. Besides being desirable for the optimization routines, this is also very useful to assess if no coding errors have been made during the implementation of the matching cost and its derivative. By using small increments on θ_k on can compute a numerical approximation $\tilde{\nabla} L(\theta)$ for the gradient of the objective function. More precisely, one can approximate the limit in the definition of the derivative eqn.3.145 by evaluating the ratio for a small (but not too small) ε. Due to round-off errors, the approximate gradient is not precise if ε is taken too small. We tested several values for ε in order to obtain some measure of uncertainty on the approximate gradient This numerical gradient can be compared component by component with the gradient we implemented in the algorithm 10. If the ratio between two corresponding components is not close to 1, then an other ε is used to compute a second approximation of the gradient. If this two approximated gradient are close to each other then this is a indicator of the fact that there is an error in the computation of the exact gradient. Using the same methodology we can also verify that the Hessian $\partial^2 L/\partial q_i \partial q_j$ is valid using differences of gradients.

This was achieved for the *exact method*. Furthermore, we verified that the Hessian matrix is symmetric for the exact method, which could not be verified for the *inexact* approach.

3.7.6 Smart Particle Filtering

The local optimization methods presented in the previous section allow reaching a nearby minimum quite efficiently. However occlusions and depth ambiguities tend to create multiple local minima. Consequently, given the absence of temporal constraints in our approach, the method could fail after several frames. Such a limitation can be dealt with through multiple hypotheses testing. Particle filters are a common approach to implement such a framework.

Particle filtering is proposed to tackle the problem of Bayesian estimation of the hand trajectory.

Given the set of frames $I_{1:t} \equiv (I_1, \ldots, I_t)$ we aim to recover the hand pose and speed $X_t = [\theta_t, \dot{\theta}_t]^T$. We aim to compute the probability density function $p(X_t|I_{1:t})$ of present state X_t, based on observations from time 1 to time t i.e. $I_{1:t}$. This

probability density function can be computed in a recursive manner: given our previous probability density function $p(X_{t-1}|I_{1:t-1})$ and a new observation I_t, the updated *a posteriori* probability density function $p(X_t|I_{1:t})$ can be computed using Bayes' rule:

$$p(X_t|I_{1:t}) = \frac{p(I_t|X_t)p(X_t|I_{1:t-1})}{p(I_t|I_{1:t-1})} \tag{3.188}$$

We model the dynamic process X_t using a simple constant speed model:

$$P(X_{t+1}|X_t) = \frac{1}{Z}exp(-\frac{1}{2}(DX - X)^T\Sigma_d^{-1}(DX - X)) \text{ if } A\theta_{t+1} < b, \text{ 0 else} \tag{3.189}$$

Where $D = \left[\begin{smallmatrix}1 & \delta t\\ 0 & 1\end{smallmatrix}\right]$ (δt being the time interval between two successive frames), Z is a normalizing factor, and Σ_d a covariance matrix which we defined manually (by taking a diagonal matrix). Note that this matrix could be estimated from training data using the method proposed in [Ghahramani 1996]. The observation probability $P(I_t|X_t)$ is obtained using the matching cost $L(\theta)$ defined in equation 3.103. We defined the matching cost as a log-probability of the observed image given the hand pose and thus we should take $P(I_t|X_t) = exp\big(-L(\theta_t)\big)$ (remember $X_t = [\theta_t, \dot{\theta}_t]^T$). The use of the observation probability in the particle filter framework is problematic from a practical point of view. The mass of the resulting observation probability is concentrated in very narrow regions of the hand parameter space. As a consequence, a single particle will get a weight (see eqn. 3.192) close to 1, while the other particle will get a quasi-null weight after weight renormalization, even if the number of particle is large. In order to smooth and flatten the observation distribution we use a temperature parameter T and redefine $P(I_t|X_t)$ to be $P(I_t|X_t) \equiv \frac{1}{Z}exp(-L(\theta_t/T)$ with Z an normalization factor (we took T=1000 in the experiments).

Because the process is Markovian (i.e $p(X_t|X_{t-1}) = p(X_t|X_{1:t-1})$ we have:

$$p(X_t|I_{1:t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|I_{t-1})dX_{t-1} \tag{3.190}$$

The normalization factor in [eqn.3.188] is given by:

$$p(I_t|I_{1:t-1}) = \int p(I_t|X_t)p(X_t|I_{1:t-1})dX_{t-1}$$

The recursive computation of the prior and the posterior probability density function leads to the exact computation of the posterior density. Nevertheless, due to the dimension of the hand configuration space, it is impossible to compute the posterior probability density function $p(X_t|I_{1:t})$, which must be approximated. Particle filters, which are sequential Monte-Carlo techniques, estimate the Bayesian posterior probability density function with a set of samples. Sequential Monte-Carlo methods have been first introduced in [Gordon 1993]. For a more complete review of particle filters, one can refer to [Gordon 2002]. Particle filtering methods approximate the posterior pdf $p(X_t|I_{1:t})$ by a weighted sum of M Dirac distributions centered at $\{X_t^m, m = 1..M\}$ with the weights $\{w_t^m, m = 1..M\}$:

$$p(X_t|I_{1:t}) \approx \sum_{n=1}^{N} w_t^n \delta(X_t - X_t^n). \tag{3.191}$$

The set of pairs $\{X_t^n, w_t^n\}_{n=1}^N$ is called the weighted particle set. Each weight w_t^n reflects the importance of the sample X_t^n in the pdf.

Different methods exist for updating the approximated posterior probability density function, i.e the set of particles, each time a new frame is observed. One possible method is called Sequential Importance Sampling (SIS) and consists in two steps:

- Draw each new state X_t^m from the previous state X_{t-1}^m using a proposal distribution $q(X_t|X_{t-1}^m, I_t)$
- Update the importance of each sample w_t^m according to the fitness measure between the generated solution and the observed data.

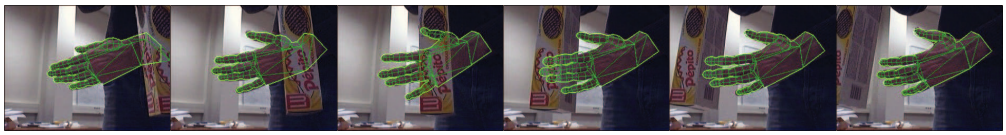
$$w_t^m \propto w_{t-1}^m \frac{p(I_t|X_t^m)p(X_t^m|X_{t-1}^m)}{q(X_t^m|X_{t-1}^m, I_t)}. \tag{3.192}$$

The algorithm's performance depends on the choice of the importance distribution. Even if the the transition prior distribution is not the optimal choice, it can be used as the importance function. In order to keep a reasonable number of particles and avoid degeneracy of the algorithm one could re-sample the set of particles after each probability density function update. The approximated distribution with N particles of heterogeneous weights $\{X_t^n, w_t^n\}_{n=1}^N$ is re-sampled into an approximation with N particles of same weight $\{X_t^n, \frac{1}{N}\}_{n=1}^N$. The combination of SIS with re-sampling is called Sampling Importance Resampling (SIR).

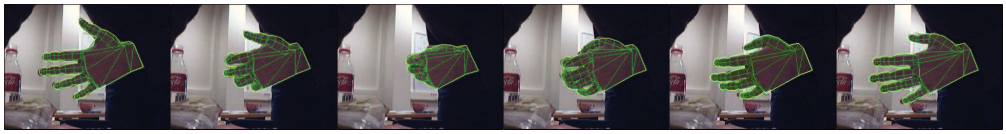
Hand pose estimation aims to recover parameters in a high-dimensional space. Therefore, a huge number of particles should be considered in order to get a reasonable approximation of the posterior probability. Unfortunately, this is not practical due to the heavy computational cost of evaluating $p(I_t|X_t)$ for a single particle. Therefore, particle filter with a reasonable number of hypotheses is likely to fail. In order to address this limitation while being able to deal to some extent with the presence of multiple local minima, we adopt the concept of "smart particle filter" [Bray 2004b]. Such an approach combines multiple hypotheses generation with local search. After propagating the particles, each particle is duplicated with half of the original weight for both particles and a local maximization (which is the variable-metric method in our case) of $p(I_t|X_t^n)$ with respect to X_t^n is performed (by minimizing $L(\theta)$ using Variable-Metric gradient-descent). Because we change some particles' positions, the represented probability density function $p(X_t|I_{1:t})$ would be altered if the weights were kept unchanged. The resulting new particle set is therefore re-weighted such that the original Bayesian distribution is not altered. This helps to get more particles near the center of each mode of the distribution and allows efficient particle filtering using far fewer samples. We refer the reader to the original paper [Bray 2004b] for a fully detailed explanation.

3.8 Discussion

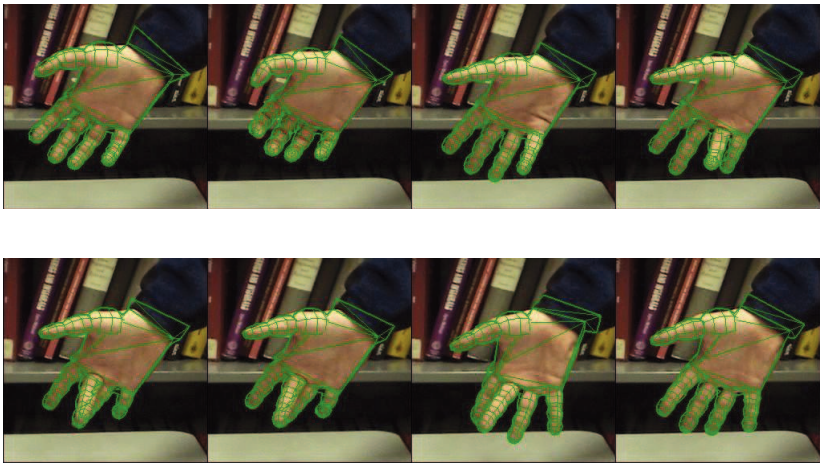
3.8.1 Validation



Descriptio 3.27: The hand is clutched and extended without loosing track



Descriptio 3.28: Robustness to occlusion is demonstrated in this sequence



Descriptio 3.29: Eight frames from a tracking sequence provided by authors of [Sudderth 2004] in which the hand makes grasping motions and individual finger movements. The result are qualitatively equivalent

In order to validate the proposed technique, several sequences with important variations of the hand configurations were considered. We also tested our method on two sequences provided by authors [Stenger 2003] and [Sudderth 2004]. Experimental tracking results are shown in [Figures (3.27,3.28,3.29,3.30)]. User-aided initial hand configuration and calibration were provided for the first frame, as well as an estimate of the color distributions of the hand and the foreground using user-aided labeling. For the sequences in [Figures (3.27,3.28)], we constrained the number of particles to 30 and the number of iteration to 10, thus limiting to 300 the number of function evaluations per frame. For the sequences [Figures (3.29,3.30)], we used

a single particle with 300 iterations. The runtime per frame is proportional to the number of function evaluations and is about 3 minutes on a 3Ghz intel Xeon^{TM}CPU Matlab®and not vectorized. Note that it is common to obtain a speedup by at last two orders of magnitude by rewriting non-vectorized Matlab®code in C. In the first sequence [Figure (3.27)], the hand is clutched and extended. In the second sequence [Figure (3.28)], robustness to occlusion is demonstrated as tracking does not fail when the foreground object occludes parts of the hand. In the third sequence [Figure (3.29)], the hand makes grasping motions and individual finger movements, as been provided by authors of [Sudderth 2004]. We show the same frame as the ones shown in [Sudderth 2004]. We obtain results that are qualitatively as good. Unfortunately, ground truth is not available in order to perform finer comparisons of the results.

Descriptio 3.30: Tracking results with a sequence provided by authors in [Stenger 2003] with a grasping movement of the hand. the first and second are obtained with some additional linear constraint between fingers angles. The third and forth row are obtained without additional linear constraints.

In the forth sequence [Figure (3.30)], the hand is closed and then opened, as been provided by authors of [Stenger 2003]. For computational reasons, the results

presented in [Stenger 2003] were obtained with important reduction in the dimension of the hand pose-space, adapted to each sequence individually (8D movements for second sequence - 2 for articulation and 6 for global motion - and 6D rigid movement for third sequence). We tested our algorithm both with and without such reductions (with reduction rows 1 and 2, without reduction rows 3 and 4). To reduce the space of poses, linear inequalities were defined between pairs or triplets of angles. Inequalities were preferred to equalities toward reducing the range of possible poses while locally keeping enough freedom of pose variation. As one would expect, the results are better when the pose space is reduced (see the first and third images from the left in the forth row). A limitation of our approach appears while inspecting the results on these sequences. When the hand is closed, the positions of the phalanges that fully project within the hand palm are not well estimated. This can be explained by that fact that these phalanges do not contribute to the synthetic hand/background boundary and thus their positions do not affect the objective function being minimized. In other words, our objective function cannot capture information relative to edges of fingers are not part of the hand/background silhouette.

Descriptio 3.31: Tracking results comparisons: each row corresponds respectively to the smart particle filter, the classical particle filter and the single hypothesis method

We compared the results between the classic particle filter method (where no local search is performed to update the particle set) and a single hypothesis method where the quasi-newton local search is initialized with the best pose found in the previous frame. We limited the number of particles to 300 for the former and the number of iteration to 300 for the latter, thus obtaining the same overall number of function evaluations per frame.

Some selected frame are presented in [Figure (3.31)]. The classical particle filter

method fails after few iterations. The dimensionality of the pose space is too high for the particle filter to be stable with only 300 particles. The single-hypothesis method fails the first time the hand gets occluded by an object. This is due to the fact that the single-hypothesis tracker is unable to escape local minima.

3.8.2 Summary

In this chapter we have proposed a novel hand model along with two novel optimization methods for hand pose estimation from monocular images. Our method is based on a hand model with 28 degrees of freedom for the articulations, while fingers refer to a succession of ellipsoids. Anatomical conditions are considered through constraints within the minimization process. Pose is estimated in a Bayesian manner through a particle filter combined with local optimization of the observation-likelihood function. In contrast to prior work that use region based terms, we estimate the gradient of the cost function with respect to the model parameters and an approximation of its Hessian. We propose a new constrained variable metric gradient descent method and a trust-region method that exploit the approximate Hessian. Both methods improves the convergence to the optimal hand parameters when compared with the standard BFGS-SQP method implement in the Matlab®function *fmincon*. The particle filter framework addresses the limitations of local optimization methods as it introduces multiple hypotheses in the process, reducing the risk of convergence to local minima.

The observed shortcoming of such an approach are multiple.

- The smart particle filter increases the chance to fall in a good local minima but this might still not be sufficient to guarantee convergence to a good estimate in the case of large inter-frame movements.

- The method requires a manual initialization, which is the case of most model-based methods

- The matching cost is only a function of the silhouette Ω and thus does not capture some important visual informations. The fitting process only tends to align the silhouette of the model with the silhouette in the observed image. The internal edges (i.e the edges that do not contribute to the silhouette) are removed while computing the silhouette and do not participate to the matching cost. As a consequence nothing in the matching cost favors hand pose candidate whose internal edges matches the edges in the observed image. The matching cost do not allows to discriminate between two pose that lead to similar silhouettes, which causes failures such as those in the second columns of [Figure (3.30)]. Using edges, one might be able to disambiguate, but such measurements are often ambiguous due to clutter. Furthermore the edge information is sometimes not sufficient to disambiguate depth ambiguities, which is illustrated in [Figure (1.4)], and might create sharp variation of the matching cost as illustrated in [Figure (2.14)]

Introducing a more sophisticated model of the hand appearance as well as an appropriate cost function that goes beyond simple silhouette matching might helps us to address some of the aforementioned limitations. The use of a detailed triangulated skin surface with texture and shading along with an objective function that exploits this model refinements is the most prominent direction to help to disambiguate configurations that lead to the same silhouette. This is the direction taken in the following chapter.

Caput 4

Method with texture & shading

In this chapter, we propose a model-based approach to recover 3D hand pose from 2D images. The hand pose, the hand texture and the illuminant are dynamically estimated through minimization of an objective function. This function is composed of a *data-fidelity term*, measuring discrepancy between the observed image and the synthetic image of the model, and a *prior term* defined using the kernel principal component analysis (KPCA) of some training set of hand poses. Derived from an inverse problem formulation, the *data term* enables explicit use of texture temporal continuity and shading information, while handling important self-occlusions and time-varying illumination. The *prior term* aims at improving the robustness of the tracking by enforcing the inferred hand pose to remain close to the training set. As a non linear extension of the principal component analysis, KPCA allows to define an adequate measure of closeness to the training set despite its nonlinearity The minimization is done efficiently using a quasi-Newton method, for which we propose a rigorous derivation of the objective function gradient. Particular attention is given to terms related to the change of visibility near self-occlusion boundaries that are neglected in existing formulations. Toward this end we introduce new occlusion forces and show that using all gradient terms greatly improves the performance of the method. Experimental results demonstrate the potential of the formulation.

4.1 Overview

As mentioned in the introduction, the key problems in 3D monocular hand tracking consist of the high dimensional search problem, noisy or missing measurements due to occlusion, and the existence of depth ambiguities. To minimize uncertainty and cope with ambiguities, one must effectively exploit the available constraints coming from the image measurements.

As we explained in section 3.8.2, the first method we proposed in this thesis does not make an optimal use of the image measurements. In particular, the only source of depth information that is exploited in this method derives from sizes of the hand parts in the images. According to the pinhole camera model presented in 3.3.2, these sizes are inversely proportional to the depth in the camera coordinate system. If the estimated depth of a hand part is erroneous, then its size will be too large or to small in comparison with the image one. By minimizing the matching cost, the fitting procedure will eventually find a depth for each part such that their projected sizes matches the image ones. Unfortunately the depth estimate will be

reliable only if one knows the orientations of the hand parts, and one has a very accurate deformable 3D model, which is not the case.

In section 1.3 we illustrated the fact that using edges is not always sufficient to resolve depth ambiguities [Figure (1.4)]. We also mentioned that shading is a powerful cue for surface orientation and relative depth. Nevertheless, shading has not been used widely for articulated tracking (but see [Lu 2003, Balan 2007]). The main reason is that shading constraints require an accurate model of surface shape and the estimation of the illuminant. Models of hand geometry where hand parts are approximated using simple ellipsoidal or cylindrical solids might not be detailed enough to obtain useful shading constraints. Surface occlusions also complicate the use of shading constraints.

In this chapter we advocate the use of a model-based approach with a richer generative models of the hand, both in terms of geometry and appearance. In terms of geometry, the hand surface is now modeled as a fine triangulated surface that is deformed to follow the articulations of the underlying skeleton. In terms of appearance, the model is now shaded and textured.

The hand pose is estimated by minimizing a matching cost that is derived from a principled analysis-by-synthesis formulation. Given a parametric hand model, and a well-defined image formation process, we seek the hand pose parameters which produce the synthetic image that is most similar to the observed image. Our similarity measure (referred as the *data term*) simply comprises the sum of residual errors, taken over the image domain. This measure is composed of a single term and thus we avoid the problem of weighting different terms related to different cues extracted in the image. We define our *matching cost* as the combination of this *data term* with a second one, referred as the *prior term*, that is independent of the observation and that penalizes poses estimates that are presupposed to be less likely in some specific context or unnatural. This second term is learned from examples using Kernel Principal Component Analysis.

The use of a fine triangulated mesh-based model allows for finer modeling of the surface than allowed by ellipsoids and thus allows for a good shading model. During the tracking process we determine, for each frame, the hand and illumination parameters by minimizing the objective function. The hand texture model is updated after convergence in each frame (using the pose objective function), and then remains static while fitting the model pose and illumination in the next frame. By explicitly modeling texture of the hand, we obtain a method that naturally captures the key visual cues without the need to add new ad-hoc terms to the objective function. This can be understood by the following reasoning: if the pose of the hand or the estimated orientation are wrong, the shading in the synthetic image will not agree with the observed shading and then the discrepancy measure will be large. While minimizing the discrepancy measure we will favor poses that generates the right shading, without the need for extracting explicitly the shading information from the image using some ad-hoc methods. Using the same reasoning, the use of texture allows to exploit the dense information provided by fine asperities of the albedo without the need for performing optical flow or patch matching. In

contrast to the approach described in [Lu 2003], which relies on optical flow, our objective function does not assume small inter-frame displacements in its formulation; it therefore allows large displacements and depth discontinuities.

Finally we do not require extra terms related to the distance between synthetic edges and edges in the observed image. If the edges in the synthetic image are not aligned with the edges in the observed image, then this induces a large residual error at the pixels in the vicinity of the synthetic and of observed edges. By minimizing the residual error we tend to automatically align the synthetic and observed edges.

The optimal hand pose is determined through a quasi-Newton descent using the gradient of the objective function. The presence of discontinuities in the synthetic image at occlusion boundaries is somehow problematic when deriving the gradient of the matching cost. We provide a novel, detailed derivation of the gradient in the vicinity of depth discontinuities, showing that there are specific terms in the gradient due to occlusions; we call such terms *occlusion forces*.

While describing our method it is natural to start by introducing the generative model that allows synthesizing an image of the hand. This generative model includes the hand geometry, appearance, and its projection onto the image.

4.2 Hand geometry

4.2.1 The choice of triangulated surface

From section 2.2.2.3 we know that several techniques have been considered in the hand tracking or body tracking literature to model the surface of an articulated body. Each of these techniques has its advantages and drawbacks in comparison with others.

Implicit surfaces [Dewaele 2004, Plänkers 2001] are smooth almost everywhere which can be useful to obtain a smooth matching cost. However we did not choose to use an implicit surface for three reasons 1) it is difficult to define an accurate surface model using implicit surfaces 2) it is difficult to perform texture mapping on an implicit surface and 3) There is generally no closed-form expression for the silhouette boundary, which complicates the derivation of continuous approximation of the objective function as done in section 4.7.3. In particular it appears to be difficult to obtain the full set of occlusion boundaries Γ_θ (see section 4.6.1.2). The solution proposed in [Schmidt 2006] and [Bremer 1998] are approximative. These approximation errors may induce temporal discontinuities of the silhouette representation when the surface continuously deforms through time, which is not suitable for our continuous optimization approach.

The hand surface models based on convex polyhedral and ellipsoids, such as the one we used in the previous chapter, are easy to project onto the image plane using the perspective geometry. However such surface models also suffer in term of accuracy and make the use of texture difficult, especially near the joints where the ellipsoids are inter-penetrating.

Following [Bray 2004a, Bray 2004b, Ogawara 2003, Du 2008, Chik 2007] we used a triangulated surface. This appeared to be the most convenient choice. Rendering a triangulated surface is quite simple in comparison with other representations, and this allows us to reach a good trade-off between the computational cost associated to the rendering and the precision of the surface. Having an accurate surface model is critical for shading because normal directions are quite sensitive to surface shape. Furthermore, triangulated surfaces are also convenient for texture mapping. For these reasons, the triangulated surface representation is the most common surface representation in computer graphics. Nevertheless, triangulated surfaces present some inherent drawbacks. The surface is continuous but not smooth along edges of the triangular facets. Depending on the definition of the matching cost, this may induce its non-differentiability with respect to the location of the surface vertices. This limitation has been addressed in [Ilic 2003] but the proposed solution is not easily adaptable to our problem : Similarly to general implicit surfaces it seems difficult to combine these implicit meshes with texture mapping and shading. Furthermore the fact that there is no close-form expression of the occlusion boundary makes difficult the numerical implementation of a differentiable approximation as done in section 4.7.3.

Given our matching cost, non-differentiability occurs only when the projection of a triangular facet degenerates into a segment. We did not experience trouble in the fitting procedure due to such non-differentiability. However we do not exclude the possibility that using a smoother surfaces model - using B-splines, non-uniform rational basis splines (NURBS) or subdivision surfaces for example - would allow faster convergence rates of the iterative fitting process.

Our hand surface model is made of 1000 facets [Figure (4.1)]. In order to obtain this model we started from a fine triangular mesh of the right hand with stretched fingers that has been acquired with a 3D scanner. With the freeware Blender we simplified this mesh and reduced the number of facets to 1000. Using the method explained in the next section we deform this triangulated surface when the hand pose and morphological parameters, respectively denoted θ and ϑ, are modified. Our hand surface is an orientable closed manifold. A formal definition of the triangulated surface will be used to define equations of the image formation process, to discuss the problem of texture continuity, and to derive the texture smoothness term in the texture update formulation. The surface is defined by:

- $N_v = 500$ is the number of vertices of the triangulated surface
- $(V(i;\theta,\vartheta))_{i=1}^{N_v} \in \mathbb{R}^{3\times N_v}$ is the parametrized list of 3D vertex positions with $V(j;\theta,\vartheta) \in \mathbb{R}^3$ being the j^{th} 3D vertex.
- $N_f = 1000$ is the number of faces in the triangulated surface.
- $F_V \in \{1,\dots,N_v\}^{3\times N_f}$ is the list of vertex indices associated to each of the N_f faces. $F_V(i,j)$ is the index of the i^{th} vertex for face j. We denote S_j the j^{th} facet of the surface. S_j is the triangle whose three extremities are

$V(F_V(1,j))$,$V(F_V(2,j))$ and $V(F_V(3,j))$. We assume that for each face these three vertices are ordered such that the vector $(V(F_V(2,j)) - V(F_V(1,j))) \wedge (V(F_V(3,j)) - V(F_V(1,j)))$ is pointing toward the exterior of the surface

We describe the position of a point in the surface using a four component vector $w \equiv (w_1, w_2, w_2, w_4)$. The component w_4 is the index of the facet whose point belongs and (w_1, w_2, w_3) are the three barycentric coordinates of the point within this facet. We have $w_4 \in \{1, \ldots, N_f\}$ and $(w_1, w_2, w_2) \in \mathbb{B}$ with $\mathbb{B} \equiv \{w \in \mathbb{R}^3 \mid w_i \geq 0, \sum_{i=1}^3 w_i = 1\}$. We denote $\mathbb{W} \equiv \mathbb{B} \times \{1, \ldots, N_f\}$ the set of such point coordinates. Using $\mathbb{B}$, the j^{th} facet, denoted S_j, is written as:

$$S_j \equiv \left\{ \sum_{i=1}^{3} w_i V(F_V(i,j); \theta, \vartheta) \mid w \in \mathbb{B} \right\}, \tag{4.1}$$

Given a pose configuration θ and morphological parameters ϑ, we define the mapping from mesh coordinates to $\mathbb{R}^3$ as follows:

$$g : \begin{array}{ccc} \mathbb{W} \times \Theta \times \Upsilon & \rightarrow & \mathbb{R}^3 \\ (w; \theta, \vartheta) & \rightarrow & \sum_{i=1}^3 w_i V(F_V(i, w_4); \theta, \vartheta) \end{array} \tag{4.2}$$

Once such a mapping has been defined, the j^{th} facet can be re-expressed as:

$$S_j \equiv g([\mathbb{B}, j]; \theta, \vartheta) \equiv \{g([b, j]; \theta, \vartheta) \mid b \in \mathbb{B}\} \tag{4.3}$$

The full 3D hand surface can be expressed as:

$$S(\theta, \vartheta) \equiv g(\mathbb{W}; \theta, \vartheta) \subset \mathbb{R}^3 \tag{4.4}$$

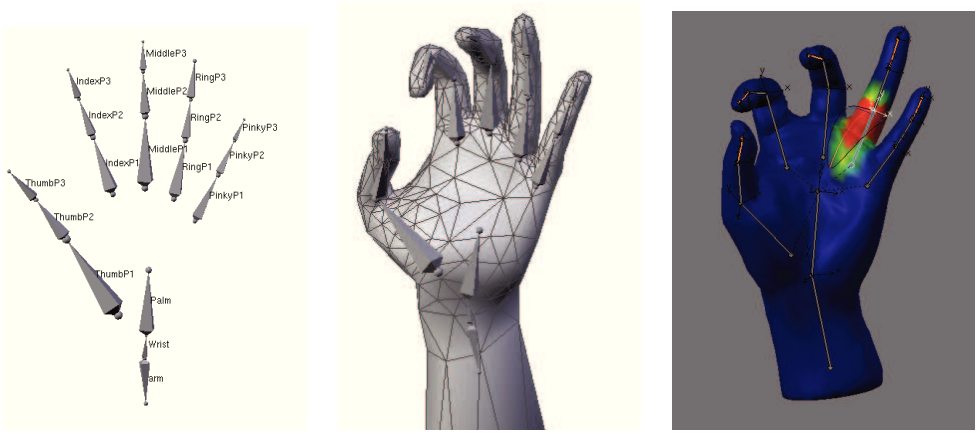
Note that, given some parameters θ and ϑ, the one-variable function $w \mapsto g(w; \theta, \vartheta)$ is not an injective function. If a vertex $V(i)$ is adjacent to k facets then the pre-image of the singleton $\{V(i)\}$ denoted

$$g^{-1}_{\theta,\vartheta}(\{V(i)\}) \equiv \{x \in \mathbb{W} | g(w; \theta, \vartheta) = V(i)\} \tag{4.5}$$

has exactly k distinct elements. This means that the vertex has k corresponding points in $\mathbb{W}$. Similarly a point along an edge is represented twice in $\mathbb{W}$. We could remove some elements of $\mathbb{W}$ to solve this problem, however we keep this redundant representation which will facilitate the discussion on the continuity of the texture. We will now describe how we define the vertex positions $(V(i; \theta, \vartheta))_{i=1}^{N_v}$ given the pose and morphological parameters θ and ϑ.

4.2.2 Linear Blend Skinning

In order to deform the triangulated surface when the hand pose parameters are modified we need to define a method to compute the new position of the surface vertices. Several methods exist in the literature. Because we ultimately want to use a gradient descent approach to estimate the hand pose, we need to use a deformation



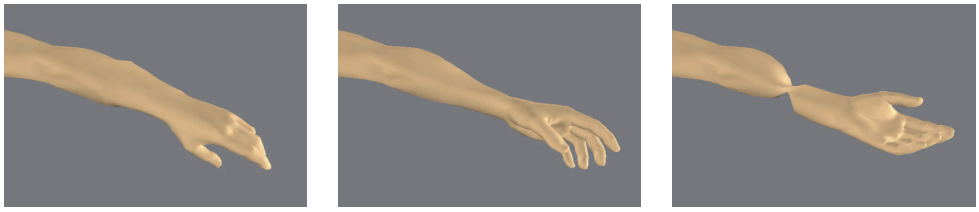
(a) (b)

Descriptio 4.1: (a) The skeleton (b) The deformed hand triangulated surface c) blending weights associated to the ring finger proximal phalanx

model that is not too complicated to differentiate with respect to the pose parameters. Following [Bray 2004a, Bray 2004b, Ogawara 2003, Du 2008, Chik 2007], we use the skeleton subspace deformation method (SSD) [Magnenat-Thalmann 1988, Lewis 2000]. This technique is widely used for interactive applications and is also known as enveloping, smooth skinning (in Autodesk Maya®) or linear blend skinning. This method is used in the freeware Blender to deform a mesh under the influence of a skeleton.

Using Blender, we start by placing an articulated skeleton inside the hand surface with all fingers extended. While defining the skeleton, Blender does not provide a simple method to define the $\Pi/4$ rotation we used in the definition of the thumb articulations in the previous chapter (see equation 3.24). As a consequence we have to add an extra degree of freedom to the thumb. We also replace the bone in the forearm by two bones that are rigidly fixed. This doubles the set of morphological parameters for the forearm and allows better adaptation of the model to the morphology of the user (see section 4.2.3). The new skeleton comprises 18 bones with 29 degrees of freedom (DOF). The pose is fully determined by a vector θ that comprises the 22 articulation parameters, 3 translational parameters and a quaternion which specifies the global position and orientation of the wrist with respect to the camera's coordinate frame.

Then, each vertex of the triangulated surface is associated to one or more bones. The influence of i^{th} bone on the j^{th} vertex is determined according to the *blending weight* w_{ij}. For each vertex the set of weights is normalized i.e. $\sum_i w_{ij} = 1$. Using Blender we have been able to manually edit these blending weights [Figure (4.1.c)]. Each vertex $V(j)$ follows the movement of the bone(s) it is attached to, i.e. of the bone(s) i such that $w_{ij} > 0$. Let denote θ_0 the parameter vector that corresponds to the reference pose with all fingers extended. Then, for any pose configuration θ,



Descriptio 4.2: candy-wrapper artifact when the forearm rotates with an the angle π

the new position of each vertex is computed as follows:

$$V(j;\theta) = \sum_i w_{ij} K_i(\theta) K_i(\theta_0)^{-1} V_j^0 \tag{4.6}$$

Where V_j^0 is the homogeneous coordinate f vertex j for the reference pose θ_0 in the world coordinate system. The two 4 by 4 matrices $K_i(\theta)$ and $K_i(\theta_0)$ are the transformation matrix that maps from i^{th} joint coordinates to world coordinates respectively for poses θ and θ_0. These matrices are obtained using the forward kinematic equation (see eqn.3.3). Using progressive transitions while specifying the blending weights w_{ij}, we obtain smooth deformations of the surface around joints of the hand [Figure (4.1.b)].

This skinning algorithm has some well known limitations. When the skeleton deformation is large, it suffers from characteristic artifacts such as the candy-wrapper [Figure (4.2)]. The artifacts occur because vertices are transformed by a linear combination of rigid transformations matrices $K_i(\theta)K_i(\theta_0)^{-1}$. Because the set of rigid transformation matrices is not convex, the linear combination of these matrices does not defines a rigid transform. If the matrices $K_i(\theta)K_i(\theta_0)^{-1}$ are very dissimilar, as in a rotation of nearly π, the interpolated transformation degenerates and the geometry collapses. Some methods to overcome this limitations exist in the literature [Kry 2002, Mohr 2003]. The artifacts are not too visible on our model and the SSD appeared to be sufficient for hand tracking purposes.

4.2.3 Morphological variations

We expect the sizes of the palm and fingers to vary across individuals. In order to allow morphological variations between users, we introduce 54 additional scaling parameters (3 per bone), called *morphological parameters*. Those parameters are estimated during the calibration process explained in Section 4.9.1. Let denote the morphological parameter vector $\vartheta \in \Upsilon$ with $\Upsilon \equiv \mathbb{R}^3 \times N$. Those parameters are not susceptible to temporal change for the same person, but rather from one person to another. Because we use a triangulated surface, modifying the length of a bone is not as simple as in the method presented in the previous chapter. When we modify the scale of bone, we also need to displace the vertices that are associated to this bone. For notation convenience we decompose the parameters θ_0 into $\theta_0 \equiv (\theta_1^0, \ldots, \theta_N^0)$,

where θ_j parameterize the articulation between the bone j and its parent. We also decompose the vector of morphological parameters into N three-dimensional subvectors $\vartheta \equiv (\vartheta_1, \ldots, \vartheta_N)$. For each bone $j \geq 2$ we define the matrix $M_j \equiv F_j(\theta_j^0)$ that express the pose of he bone j in the coordinated system associated to parent bone $p(j)$ in the reference position. For each distal and middle phalanx the matrix M_j is a translation matrix $T_y(l_{p(j)})$ (see eqn.3.21) with $l_{p(j)}$ the length of the bone j before morphological rescaling. The vector $M_{j,[1:3,4]}$ is the coordinates of the center of the joint between the bones $p(j)$ and j expressed in the local coordinate system of the bone $p(j)$. If the parent bone $p(j)$ undergoes rescaling with parameters $\vartheta_{p(j)} \in \mathbb{R}^3$ then the homogeneous coordinates of the center of the joint expressed in the parents coordinate system should be $[\vartheta_{p(j)}, 1]^D M_{j,[1:4,4]}$ with $[\vartheta_{p(j)}, 1]^D$ the diagonal matrix whose diagonal elements are the three components of $\vartheta_{p(j)}$ and 1. Therefore for each bone $j \geq 2$ we define the new matrix valued functions F_j^s which is a modified version of F_j (see eqn3.1) that take the morphological parameters into account:

$$F_j^s : \begin{array}{ccc} \Theta_j \times \Upsilon \times \mathbb{R}^3 & \to & \mathbb{K} \\ \theta_j, \vartheta & \mapsto & \left[M_{j,[1:4,1:3]}, [\vartheta_{p(j)}, 1]^D M_{j,[1:4,4]}\right] (M_j)^{-1} F_j(\theta_j) \end{array} \tag{4.7}$$

Given a pose parameter θ and morphological parameters ϑ construct $K_1, \ldots, K_N$ using the recursive equations

$$K_j(\theta, \vartheta) = K_{p(j)} F_j^s(\theta_j, \vartheta) \tag{4.8}$$

Then we need to redefine the SSD equation (see eqn4.6) in order to take the morphological variation into account:

$$V(j; \theta, \vartheta) = \sum_i w_{ij} K_i(\theta, \vartheta) [\vartheta_{p(i)}, 1]^D K_i(\theta_0)^{-1} V_j^0 \tag{4.9}$$

Now that we define the hand geometry, given the hand pose parameters and the morphological parameters, we need to precise the hand appearance. This is done by texturing and shading the hand.

4.3 Hand appearance & projection

Since most hands have relatively little texture, shading has a major impact on hand appearance. We therefore incorporate illumination and shading (using the Gouraud shading algorithm) in our synthesis model. We assume a Lambertian surface for the hand reflectance with a simple model for the illuminant, and an adaptive model for the albedo function over the surface of the hand (to model texture and other otherwise unmodelled appearance properties).

4.3.1 Shading the hand

The illuminant model includes a distinct point source and an ambient term. Its parameters are specified by a 4D vector denoted by L, comprising three elements

for a directional component, and one for an ambient component. The irradiance at each vertex of the surface mesh is obtained by the scalar product between the homogenized surface normal at the vertex and this 4D vector. The irradiance across each face is then obtained through bilinear interpolation. Multiplying the reflectance and the irradiance yields the appearance for any point on the surface.

In order to formalize the shading computation we define normal vectors on vertices $\big(\hat{n}_i(\theta,\vartheta)\big)_{i=1}^{N_v} \in \mathbb{R}^{3\times N_v}$ by averaging the normal vectors on adjacent faces. We compute the luminance associated with each vertex given L as follows:

$$L_v(i;\theta,\vartheta,L) = \big\langle \hat{n}_i(\theta,\vartheta), L_{[1:3]} \big\rangle + L_4 \tag{4.10}$$

Once the luminance of the vertices is computed, we linearly interpolate the luminance on the surface using the barycentric coordinates:

$$L : \begin{array}{ccc} \mathbb{W}\times\Theta\times\Upsilon\times\mathbb{R}^4 & \rightarrow & \mathbb{R} \\ (w;\theta,\vartheta,L) & \rightarrow & \sum_{i=1}^{3} w_i L_v\big(F_V(i,w_4);\theta,\vartheta,L\big) \end{array} \tag{4.11}$$

4.3.2 Texturing the hand

Texture (albedo variation) can be handled in two ways. The first associates an RGB triplet with each vertex of the surface mesh, from which one can linearly interpolate over mesh facets. This approach is conceptually simple but computationally inefficient as it requires many small facets to accurately model smooth surface radiance. The second approach, widely used in computer graphics, involves mapping an RGB reflectance (texture) image onto the surface. This technique guarantees details preservation with a reasonably small number of faces.

In contrast with previous methods in the field of computer vision that used textured models (such as the morphable model [Blanz 1999]), our pose estimation formulation (Sec. 4.6) requires that the surface reflectance is continuous over the entire surface. Using bilinear interpolation of the discretized texture we ensure continuity of the reflectance properties within each face.

However, since the hand is a closed surface it is mathematically impossible to define a continuous bijective mapping between the whole surface and a 2D planar surface. As a consequence there is no simple way to ensure continuity of the texture over the entire surface.

Following [Soucy 1996] and [Hernández 2004], we use patch-based texture mapping. Each facet is associated with a triangle (also refered as *patch*) in the texture map. These triangles are uniform in size and have integer vertex coordinates. As depicted in [Figure (4.3)], each facet with an odd (respectively even) index is mapped onto a triangle that is the left-upper-half (respectively right-lower-half) of a square of size K_T by K_T with $K_T \in \mathbb{N}$ divided along its diagonal. In practice we used $K_T = 5$. Because we use bilinear interpolation we need to reserve some pixels in the texture map (a.k.a texels) outside the diagonal edges for points with non-integer coordinates [Figure (4.3)].

The formal definition of the hand texture mapping can be done as follows:

- T is a color texture image. For each texel (texture pixel)of coordinate (i, j) its color is defined by $T_{i,j} \in \mathbb{R}^3$.
- N_t is the number of texture vertices in the texture map. Because we use separated triangular patch for each facet of the hand surface we have $N_t = 3N_f$.
- $(V_T(i))_{i=1}^{N_t} \in [0,1]^{2\times N_t}$, a list of N_t 2D vertex positions in the texture map (usually referred as UV coordinates vertices in the computer graphic community).
- $F_T \in \{1,\dots,N_t\}^{3\times N_f}$ associates texture vertices to the extremities of each of the N_f faces. This differs from the most common method in computer graphics where a single UV coordinate is associated with each 3D vertex of the triangulated surface, independently of the faces. As a consequence a single vertex $V(j)$ that is adjacent to k facets is mapped into k vertices in the texture map.

We define the mapping from mesh coordinates to UV coordinates as follows:

$$h : \begin{array}{ccc} \mathbb{W} & \to & [0,1]^2 \\ w & \to & \sum_{i=1}^{3} w_i V_T(F_T(i, w_4)) \end{array} \tag{4.12}$$

Given a point on the surface $s \in S(\theta, \vartheta)$ its color is defined through three steps by 1) taking a coordinate $w \in g^{-1}_{\theta,\vartheta}(\{s\})$ 2) finding the corresponding point $h(w)$ in the texture image and 3) computing its color $r(h(w); T)$ with r the function that bilinearly interpolates the texture image :

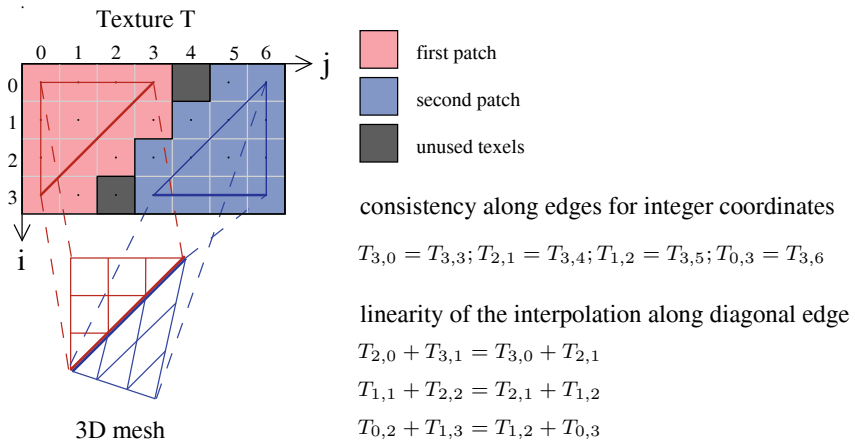
$$r : \begin{array}{ccc} \mathbb{R}^2, \mathbb{T} & \to & \mathbb{R}^3 \\ (p_x, p_y; T) & \to & \sum_i \sum_j T_{i,j} k_b(p_x - i, p_y - j) \end{array} \tag{4.13}$$

Where k_b is the bilinear interpolation kernel defined in eqn.3.107. We combine these three steps into a single function f_T such that $f_T(s)$ represents the color of the point s on the surface: $f_T(s) \in \mathbb{R}^3$ with

$$f_T(s) \equiv r(h(w); T) \text{ with } w \in g^{-1}_{\theta,\vartheta}(\{s\}). \tag{4.14}$$

The function g is actually not an injective which means that $g^{-1}(\{s\})$ might not be a singleton. This is due to the fact that our representation of the surface based on $\mathbb{W}$ is redundant for vertices of the triangulated surface or points along edges of the triangulated surface. Each point on an edge occurs twice in the texture map, while each vertex occurs an average of 6 times (according to the Eulerian properties of the mesh). Because of this redundancy, some constraints must be introduced to ensure consistency between RGB values of texture map points that map to the same edge or vertex of the 3D mesh. If we denote $\{w_1,\dots,w_k\}$ the k elements of the set $g^{-1}_{\theta,\vartheta}(\{s\})$, that is the pre-image of the singleton $\{s\}$, then this constraint can be formalized as

$$r(h(w_1); T) = \cdots = r(h(w_k); T) \tag{4.15}$$



Descriptio 4.3: Two adjacent facets of the triangulated surface project in two separated triangles in the 2D texture map T. Since the shared edge projects in two distinct segments in the texture map, it is necessary to specify constraints that the texture elements along the shared edge must be consistent. This is done here using 7 linear equality constraints.

By enforcing this consistency we also ensure the continuity of the texture across edges of the triangulated mesh. Due to the use of bilinear texture mapping, this consistency is straightforwardly enforced with two sets of linear constraints on the RGB intensities of the texture pixels. The first set of linear constraints specifies that the intensities of points along mesh edges with integer-coordinates in the texture map must be consistent. The second set of constraints enforces the interpolation of the texture to be linear along edges of the triangular patches. As long as the texture interpolation is linear along each edge and is texture intensities are consistent at points with integer texture coordinates, the mapping will be consistent for all points along the edge. The bilinear interpolation is already linear along vertical and horizontal edges, so we only need to consider the diagonal edges while defining the second set of constraints.

Let $(i+1, j)$ and $(i, j+1)$ denote two successive points with integer coordinates along a diagonal edge of some triangular patch in the texture map. Using bilinear texture mapping, the texture intensities of a point with non-integer coordinates $(i+1-\lambda, j+\lambda), \lambda \in (0,1)$ along this edge is expressed as

$$\lambda(1-\lambda)(T_{i,j} + T_{i+1,j+1}) + \lambda^2 T_{i,j+1} + (1-\lambda)^2 T_{i+1,j} \ . \tag{4.16}$$

By twice differentiating this expression with respect to λ, we find that it is linear with respect to λ if and only if the following linear constraint is satisfied:

$$T_{i,j} + T_{i+1,j+1} = T_{i+1,j} + T_{i,j+1} \ . \tag{4.17}$$

By considering all points on diagonal edges, this constitutes the second set of linear constraints. Finally, let $\mathbb{T}$ denote the linear subspace of valid textures, i.e. textures

whose RGB values satisfy the linear constraints that ensure continuity over the surface.

The combination of the surface geometry, the texture, the lighting condition as well as the shading can now be used to produce realistic appearances of the surface. The appearance of a point s of the surface S_θ is define by

$$A(s) = L(w; \theta, \vartheta, L) r\big(h(w); T\big) \text{ where } w \in g^{-1}_{\theta,\vartheta}(s) \tag{4.18}$$

While defining the texture we have ensured that the choice of w in the set $g^{-1}_{\theta,\vartheta}(s)$ does not change the appearance of a point given our texture and shading models. We need now to describe how the surface along with its appearance projects into the image domain to produce the synthetic image that we aim to compare to the observed image.

4.3.3 Hand Model projection & Occlusions

Given the illuminant, the texture mapping, and the mesh geometry, the formation of the model image at each point $\mathbf{x}$ in the 2D image plane can be determined in a two step procedure. First, as in ray-tracing, we determine the first intersection between the triangulated surface mesh and the half-line (or ray) starting at the camera center and passing through $\mathbf{x}$. Second, the appearance of this intersection point is computed given the illuminant and the information at the vertices of the intersected face. If no intersection exists, then the image is determined by the background. Therefore, in addition to the hand, a model for the background is also considered. When it is static, we simply assume that an image of the background is available. Otherwise, we assume a probabilistic model for which background pixel intensities are drawn from a background distribution $p_{bg}(\cdot)$ (e.g., it can be learned in the first frame with some user interaction).

The two step procedure to estimate the synthetic color at a given point $\mathbf{x}$ can be formalized as follows. Using the camera projection function Π defined in eqn.3.61 and the function g defined in eqn.4.2, the mapping from the mesh coordinate to the image coordinate is is expressed as $\Pi \circ g$. This mapping is homographic for each planar triangular face. A point with coordinates (i, j) in the texture image is mapped onto the point $\Pi \circ g \circ h^{-1}(i, j)$. This yields to the so called perspective texture mapping in computer graphics [Heckbert 1989]. A common assumption in computer graphics is that the mapping from mesh to image coordinates can be fairly well approximated by a linear function rather than homographic for each triangular face, when the size of projected triangles is small enough. Once the projection of the vertices $\bar{V}(i; \theta\vartheta) \equiv \Pi\big(V(i; \theta, \vartheta)\big)$ is computed, we can define the approximated projection of a point on the surface directly from its mesh coordinates:

$$\hat{\Pi} : \begin{array}{ccc} \mathbb{W} \times \Theta \times \Upsilon & \to & \mathbb{R}^2 \\ (w; \theta, \vartheta) & \to & \sum_{i=1}^{3} w_i \bar{V}(F_V(i, w_4); \theta, \vartheta) \end{array} \tag{4.19}$$

In order to compute the appearance of a point $\mathbf{x}$ in the image domain, we first recover the closest point (with respect to the camera) of the surface that projects

onto it and then attribute its appearance to the point $\mathbf{x}$. This is formalized by the back-projection function:

$$\hat{\Pi}^{-1}_{\theta,\vartheta} : \begin{array}{ccc} \mathbb{R}^2 & \leftarrow & P(\mathbb{W}) \\ \mathbf{x} & \mapsto & \arg\inf \Big(\big\langle \hat{z}_c, \big(g(w;\theta,\vartheta)\big) \big\rangle \,|\, w \in \mathbb{W}, \hat{\Pi}(w;\theta,\vartheta) = \mathbf{x} \Big) \end{array} \tag{4.20}$$

where $\hat{z}_c$ is the direction of the camera principal axis and $P(\mathbb{W})$ is the power set of $\mathbb{W}$ i.e the set of all subsets of $\mathbb{W}$. This maps any point that does not back project on the surface, i.e that belongs to the background, to the empty set $\emptyset$. Let denote by $\chi(\theta,\vartheta) \subset \Omega$ the set of 2D points within the hand silhouette i.e. that back-project onto the hand surface. We have $\chi(\theta,\vartheta) \equiv \Pi(S(\theta,\vartheta))$. The mapping $\hat{\Pi}^{-1}_{\theta,\vartheta}$ is linear within the visible part of the projection of each facet, and thus is piecewise linear. One should note that the set $\hat{\Pi}^{-1}_{\theta,\vartheta}(\mathbf{x})$ has more than one element when $\mathbf{x}$ lies projected edges or vertices because $g(\theta,\vartheta,.)$ is not an injective function. This is not problematic due to the choice made on the on shading and texturing models which will produce the same appearance for all points in this set.

Now that we have defined the inverse projection function, we can write a formal expression for the synthesized image. Let $I_{syn}(\mathbf{x};\theta,\vartheta,L,T)$ denote the synthetic image comprising the hand and the background, at the point $\mathbf{x}$ for a given pose θ, texture T and illuminant L.

$$I_{syn}(\mathbf{x};\theta,\vartheta,L,T) = \begin{cases} L(w;\theta,\vartheta,L)\ r(h(w),T) \text{ with } \ w \in \hat{\Pi}^{-1}_{\theta,\vartheta}(\mathbf{x}) \text{ if } \mathbf{x} \in \chi(\theta,\vartheta) \\ I_{bk}(\mathbf{x}) \text{ else} \end{cases} \tag{4.21}$$

where I_{bk} is an image of the background that is obtain in a semi-supervised manner.

In practice, the image is computed on a discrete grid and the image synthesis can be done efficiently using the triangle rasterization technique in combination with a depth buffer

This completes the definition of the model synthesis process for images of the hand. We next consider the estimation problem that consist in inferring the surface from real 2D observations.

4.4 Data-fidelity function

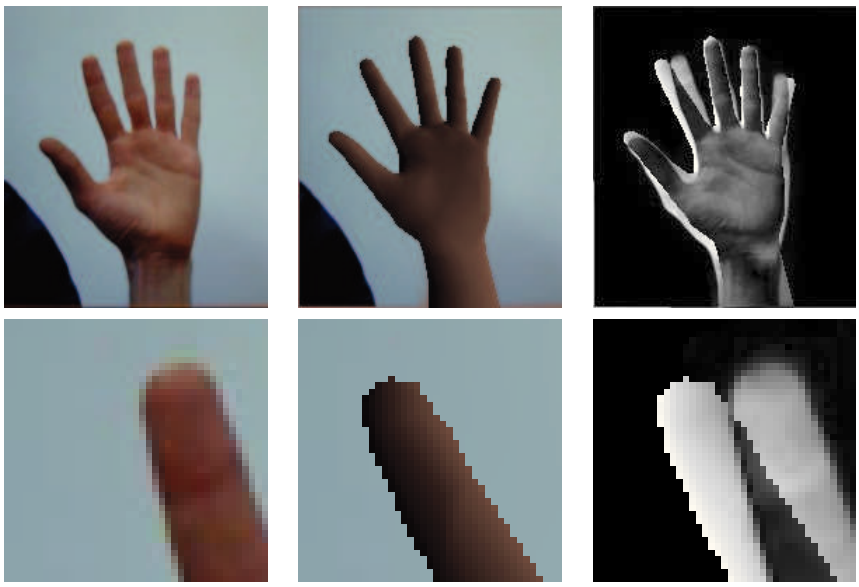
4.4.1 Discrete image domain

Our task is to estimate the pose parameters θ, the texture T, and the illuminant L that produce a synthesized image which best matches the observed image, denoted by $I_{obs}()$. Our objective function is based on a simple discrepancy measure between these two images. First, we define a residual image R_d [Figure (4.4.c)] as

$$R_d(\mathbf{x};\theta,\vartheta,T,L) \;=\; \rho\big(\|I_{syn}(\mathbf{x};\theta,L,T) - I_{obs}(\mathbf{x})\|\big)\ , \tag{4.22}$$

where ρ is either the classical squared-error function or a robust error function such as the Huber function used in [Sidenbladh 2000]. With ρ being the classical

squared-error function, the pixels errors are implicitly assumed to be IID Gaussian. Similarly, with a robust error function, the pixels errors are implicitly assumed to be heavy-tailed IID. Contrary to this simple IID assumption, modeling errors tend to produce highly correlated errors between nearby pixels. But as the quality of the generative model improves (as we strive to do in this chapter) the significance of this correlation is lessened.



Descriptio 4.4: The first raw represents from the left to the righ: the observed image I_{obs}, the synthetic image I_{syn} and the residual image R_d. The second raw represents zooms on index extremity.

When the background is not static, but rather defined probabilistically, we can separate the image domain Ω (a continuous subset of $\mathbb{R}^2$) into the region $\chi(\theta, \vartheta)$ covered by hand, given the pose, and the background region $\Omega \backslash \chi(\theta, \vartheta)$. Then the residual can be expressed using the background color distribution as:

$$R_d(\mathbf{x}; \theta, \vartheta, L, T) \;=\; \begin{cases} \rho\big(\|I_{syn}(\mathbf{x}; \theta, L, T) - I_{obs}(\mathbf{x})\|\big), \;\; \forall \mathbf{x} \in \chi(\theta, \vartheta) \\ -\log\big(p_{bg}(I_{obs}(\mathbf{x}))\big), \;\; \forall \mathbf{x} \in \Omega \backslash \chi(\theta, \vartheta) \end{cases} \tag{4.23}$$

Examples of the residual obtained using p_{bf} to model the background are shown in the third and sixth rows of [Figure (4.19)].

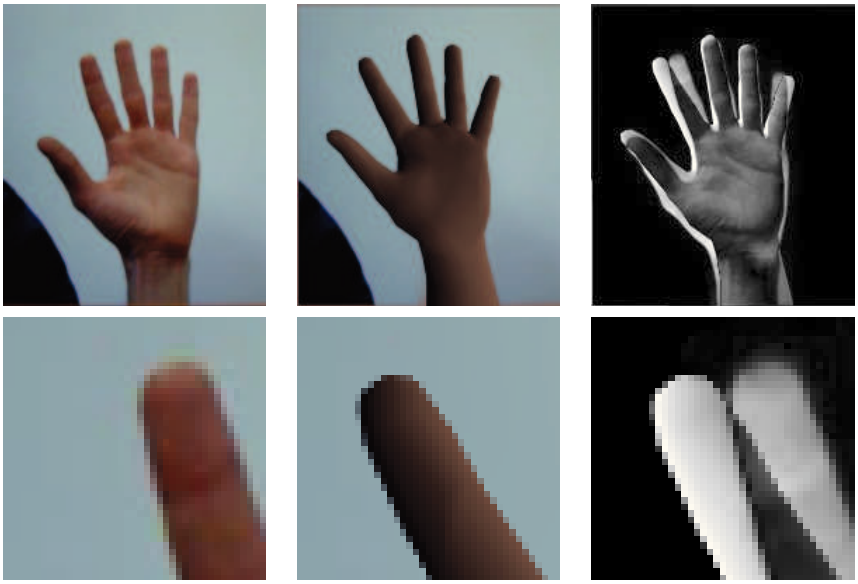
Our discrete discrepancy measure, E_d, is defined to be the sum of the residual errors over the discrete image domain Ω_d:

$$E_d(\theta, \vartheta, T, L) \;=\; \sum_{\mathbf{x} \in \Omega_d} R_d(\mathbf{x}; \theta, L, T) \tag{4.24}$$

Unfortunately, due to the aliasing along edges of the occluding contour (i.e., discretization noise), $E_d(\theta, T, L)$ is not a continuous function of θ (e.g., when an

occlusion boundary crosses the center of a pixel). The aliasing artifacts are visible in images [Figure (4.4)]. As a consequence the gradient $\nabla_\theta E_d$ is not defined everywhere and the first order approximation of the objective function based on its gradient is only valid in a very small (sub-pixel) displacement range. Based on this discontinuous objective function and its gradient it is not possible to perform efficient continuous optimization to estimate the hand pose. Some tests based on this objective function and its gradient have provided very poor results even if initialized in the vicinity of the right solution. As a consequence we define another version of this objective function that is continuous everywhere and whose gradient provide a first-order approximation that is valid in a wider range of displacements.

4.4.2 Antialiasing formulation



Descriptio 4.5: The first raw represents from the left to the righ: the observed image I_{obs}, the antialiased synthetic image $\bar{I}_{syn}$ and the residual image R_a. The second raw represents zooms on index extremity.

In actual cameras, the optical system filters out high frequencies above half the spacial frequency of the captor. This helps to avoid aliasing in the captured image. As a consequence, the intensity measured at a given pixel is a weighted integral of the image I_{syn} in the vicinity of the pixel's center. This can be modeled by a convolution of the image I_{syn} by a kernel. Using an interpolation kernel k as the ones introduced in eqn.3.105 or eqn.3.107 this yields:

$$\bar{I}_{syn}(i,j;\theta,\vartheta,T,L) = \int_x \int_y k(x-i,y-j) I_{syn}(x,y;\theta,\vartheta,T,L) \tag{4.25}$$

This discrete antialiased image $\bar{I}_{syn}$ [Figure (4.5.b)] is a continuous function of θ,ϑ, T and L. When the boundary of a finger displace continuously, the intensities of this image changes continuously. Based on this antialiased image we define a new residual R_a [Figure (4.5.c)]

$$R_a(i,j;\theta,\vartheta,T,L) = \rho\|\bar{I}_{syn}(i,j;\theta,\vartheta,T,L) - I_{obs}(i,j)\|) \tag{4.26}$$

And an objective function

$$E_a(\theta,\vartheta,T,L) = \sum_{i,j} R_a(i,j;\theta,\vartheta,T,L) \tag{4.27}$$

This objective function is a continuous function of θ,ϑ, T and L. However this function is not very well adapted to efficient local search methods. In order to illustrate this we consider a simple one-dimensional case where the parameter θ is a scalar and the synthetic image is a white rectangle $[\theta,\theta+L]\times[0,H]$. L is an integer that represent the width of the rectangle. This rectangle is fully extended in the Height direction. The synthetic image is:

$$I_{syn}(x,y;\theta) = \begin{cases} 1 \text{ if } (x,y)\in[\theta,\theta+L]\times[0,H] \\ 0 \text{ otherwise} \end{cases} \tag{4.28}$$

Using the shifted nearest neighbor interpolation kernel k_n (eqn.3.105) we have the antialiased image:

$$\bar{I}_{syn}(i,j;\theta) = \begin{cases} 1 & \text{if} \quad i\in[\lfloor\theta\rfloor+1,\lfloor\theta\rfloor+L-1] \\ 1-\varepsilon(\theta) & \text{if} \quad i=\lfloor\theta\rfloor \\ \varepsilon(\theta) & \text{if} \quad i=\lfloor\theta\rfloor+L \\ 0 & \text{else} \end{cases} \tag{4.29}$$

Supposing the observed image to be constant and equal to zero, and $\rho : x \mapsto x^2$, we have

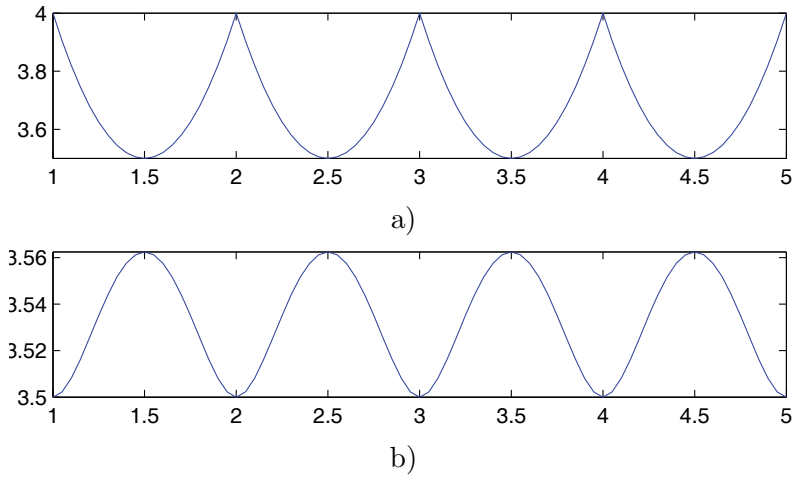
$$E_a(\theta) = H(L-1+(1-\varepsilon(\theta))^2+\varepsilon(\theta)^2) \tag{4.30}$$

Using the bilinear interpolation kernel k_b (eqn.3.107) we have the antialiased image:

$$\bar{I}_{syn}(i,j;\theta) = \begin{cases} 1 & \text{if} \quad i\in[\lfloor\theta\rfloor+2,\lfloor\theta\rfloor+L-1] \\ \frac{1}{2}-\varepsilon(\theta)+\frac{1}{2}\varepsilon(\theta)^2 & \text{if} \quad i=\lfloor\theta\rfloor \\ 1-\frac{1}{2}\varepsilon(\theta)^2 & \text{if} \quad i=\lfloor\theta\rfloor+1 \\ \frac{1}{2}+\varepsilon(\theta)-\frac{1}{2}\varepsilon(\theta)^2 & \text{if} \quad i=\lfloor\theta\rfloor+L \\ \frac{1}{2}\varepsilon(\theta)^2 & \text{if} \quad i=\lfloor\theta\rfloor+L+1 \\ 0 & \text{else} \end{cases} \tag{4.31}$$

Supposing again the observed image to be constant and equal to zero, and $\rho : x \mapsto x^2$ we have

$$E_a(\theta) = H(L-\frac{1}{2}+(1-\varepsilon(\theta))^2\varepsilon(\theta)^2) \tag{4.32}$$



Descriptio 4.6: Objective function $E_a(\theta)$ for the rectangle example with a) the shifted nearest-neighbor interpolation kernel k_n and b) the bilinear interpolation kernel k_b

A plot of the function $E_a(\theta)$ for the two kernels k_n and k_b are shown in [Figure (4.6)] for $H = 1$ and $L = 4$.

While using the nearest neighbor kernel k_n, the function E_a is not differentiable everywhere. While using the bilinear kernel, this function is continuous and differentiable but also bumpy. It "oscillates" and thus the local gradient provides a first order approximation that is valid only in a very limited range. In order to avoid this problem and obtain a constant value for this simple one-dimensional case involves the interpolation of the observed image in the continuous image domain.

4.4.3 Continuous Image domain formulation

In order to define a continuous matching cost we denote by $\tilde{I}_{obs}$ the continuous image obtained using bilinear interpolation.

$$\tilde{I}_{obs}(x, y) = \sum_{ij} I_{obs}(i, j) k_b(x - i, y - j) \tag{4.33}$$

where k_b is the bilinear interpolation kernel defined in eqn.3.107. We define the residual image R based on the interpolated observed image:

$$R(\mathbf{x}; \theta, \vartheta, T, L) \; = \; \rho\big(\|I_{syn}(\mathbf{x}; \theta, L, T) - \tilde{I}_{obs}(\mathbf{x})\|\big) \; , \tag{4.34}$$

When the background is not static, but rather defined probabilistically, we express the residual as follows:

$$R(\mathbf{x}; \theta, \vartheta, L, T) \; = \; \begin{cases} \rho\big(\|I_{syn}(\mathbf{x}; \theta, L, T) - \tilde{I}_{obs}(\mathbf{x})\|\big), \; \forall \mathbf{x} \in \chi(\theta, \vartheta) \\ -\log\big(p_{bg}(\tilde{I}_{obs}(\mathbf{x}))\big), \; \forall \mathbf{x} \in \Omega \backslash \chi(\theta, \vartheta) \end{cases} \tag{4.35}$$

Our discrepancy measure, E_c, is now defined to be the continuous integral of the residual errors over the image domain Ω:

$$E_c(\theta, \vartheta, T, L) = \int_{\Omega} R(\mathbf{x}; \theta, L, T) d\mathbf{x} \tag{4.36}$$

The resulting objective function is less realistic from a generative point of view than the objective function formulate in the previous section (eqn.4.27) but is smoother. On the example of the constant size rectangle with a constant zero observation, this objective function will remain constant. Note that unlike in the previous chapter we do not have an equivalence between the pre-filtered antialiasing formulation (eqn.4.27) and the continuous image-domain formulation (4.36). This is a very simple discrepancy measure which is, in several ways, preferable to more sophisticated measures that combine heterogeneous cues such as optical flow, silhouettes, or chamfer distances between detected and synthetic edges. First, the measure in (4.36) avoids the tuning of weights associated with different cues that is often problematic; in particular, a simple weighted sum of errors from different cues implicitly assumes (usually incorrectly) that errors in different cues are statistically independent. Second, the measure in (4.36) avoids early decisions about the relevance of edges through thresholding, about the area of the silhouette by segmentation, and about the position of discontinuities in optical flow. Third, the equation (4.36) is a continuous function of θ, that is not the case for measures based on distances between edges like the symmetric chamfer distance. These measures usually present discontinuities or sharp variations when an edge of one finger is suddenly occluded by another finger.

4.4.4 Surface domain formulation

By changing the domain of integration, the integral of the residual error within the hypothesized hand region can be re-expressed as a continuous integral over the visible part of the surface. We define E_χ the contribution of the hand surface to the data-fidelity term, i.e. the integral of the residual within the hand silhouette:

$$E_\chi(\theta, \vartheta, T, L) \equiv \int_{\chi(\theta,\vartheta)} \rho\big(\|I_{syn}(\mathbf{x}; \theta, L, T) - \tilde{I}_{obs}(\mathbf{x})\|\big) d\mathbf{x} \tag{4.37}$$

Note that E_χ and E_c differ. We have:

$$E_c(\theta, \vartheta, T, L) \equiv E_\chi(\theta, \vartheta, T, L) + \int_{\Omega/\chi(\theta,\vartheta)} R(\mathbf{x}; \theta, L, T) d\mathbf{x} \tag{4.38}$$

In order to reformulate this integral as an integral on the surface $S(\theta, \vartheta)$ we need to introduce an visibility function v with $v(s; \theta, \vartheta)$ the visibility of point s that is binary variable that equals 1 if the point s is visible and 0 if the point is hidden by other parts of the hand. This visibility function can be formalized as:

$$\begin{aligned} v(s; \theta, \vartheta) = & 1 \text{ if } \Pi^{-1}(\Pi(s)) = \{s\} \\ & 0 \text{ otherwise} \end{aligned} \tag{4.39}$$

Using this visibility function we can rewrite E_χ using an the integral onto the surface:

$$E_\chi(\theta,\vartheta,T,L) = \int_{S(\theta,\vartheta)} v(s;\theta,\vartheta)\rho\big(\|I_{syn}(\Pi(s);\theta,L,T) - \tilde{I}_{obs}(\Pi(s))\|\big)w(s;\theta,\vartheta)ds \tag{4.40}$$

where $(s;\theta,\vartheta)$ is the ratio between the surface element in the image and the corresponding surface element on the surface $s(\theta,\vartheta)$. Using some calculation one can show that we have:

$$w(s;\theta,\vartheta) \equiv \det\big([J_s\Pi^T, \hat{n}(s;\theta,\vartheta)]\big) \tag{4.41}$$

where $J_s\Pi$ is the 2 by 3 Jacobian matrix of the projection function evaluated at s and $\hat{n}(s)$ the normal vector to the surface at point s.

E_χ can then be approximated by a discretized approximation, denoted $\bar{E}_\chi$, using a finite weighted sum over centers of all visible faces:

$$\bar{E}_\chi(\theta,\vartheta,T,L) \approx \sum_{k=1}^{K} \|S_k\| v(c_k;\theta,\vartheta)\rho\big(\|I_{syn}(\Pi(c_k);\theta,L,T) - \tilde{I}_{obs}(\Pi(s_k))\|\big)w(c_k) \tag{4.42}$$

with c_k the barycenter of the k_{th} facet and $\|S_k\|$ its surface area. Much of the literature on 3D deformable models adopts this approach, and assumes the visibility of each face to be a binary state (fully visible or fully hidden) that can be obtained from a depth buffer (e.g., see [Blanz 1999]). Unfortunately, such a discretization introduces discontinuities in the approximate functional when θ varies. When the surface moves or deforms, the binary visibility state of a face near self-occlusion is likely to change. This would cause a discontinuity in the sum of residuals. Such discontinuities are undesirable if one aims to use gradient-based optimization methods. In order to preserve continuity of the discretized functional with respect to θ, the visibility state should not be binary. Rather, it should be real-valued between zero and one, and should behave smoothly as the surface becomes occluded or unoccluded. The term $v(c_k;\theta,\vartheta)$ could be replace by the ratio of the area of the visible part of the facet over the area of the facet (i.e 0.5 is half of the facet is visible). In practice, this appears rather cumbersome to implement and the analytic derivation of the functional gradient may be complicated. Furthermore the term corresponding to the integral of the residual in $\Omega/\chi(\theta,\vartheta)$ seems important to take into account if we want the silhouette of the hand model to align with the silhouette in the observed image. To address the continuity problem due to change of visibility of the facets, in contrast with much of the literature on 3D deformable models, we keep the formulation in the continuous image domain when deriving the expression of functional gradient. Deriving the analytical gradient of $E_\chi(\theta,\vartheta)$ using the surface domain formulation would also be possible but seems more complex.

4.5 Pose Prior

4.5.1 Motivation & existing methods

Hand pose estimation in monocular setting is often under-determined. Due to presence of occlusions and lack of depth information, 3D hand poses hypothesis can equally well explain the observed image. *Model-based* methods formulate the pose estimation as the minimization of some objective function based on the observed image. The above mentioned ambiguities induce the objective function to have several equally low minima or even flat regions. Reflection ambiguities are a reason for such minima multiplicity. While performing tracking, the search is generally localized around some predicted pose from previous frames. Therefore, making at some instant an erroneous choice among these ambiguous hypotheses might quickly lead to loose track. Several methods could be considered to improve robustness of hand tracking:

- Keeping track of several hypotheses and taking deferred decisions on the hand pose (e.g. using particle filtering smoothing [Stenger 2004a, Bray 2004b]). Unfortunately the number of hypothesis to track may become quickly unmanageable whenever the observations become highly ambiguous.

- Making extensive use of the available image information in order to reduce ambiguities (see [de La Gorce 2008, Lu 2003]). Two distinct hand hypotheses could explain the same segmented silhouette but are less likely to explain the same pixel intensities due to the shading for example.

- Exploiting prior knowledge about the expected hands poses. This could be done by penalizing poses estimates that are presupposed to be less likely in some specific context or unnatural (i.e. using a prior in the Bayesian formulation). An alternative is to constrain the hand pose to remain within a low dimensional sub-manifold of the Cartesian product of the individual parameter ranges. This could be derived from physical considerations on the hand or learned from a training set of poses that is assumed to be representative of the kind of gestures we aim to track.

Exploiting prior knowledge on expected articulated poses has been extensively used for human body tracking where motions generally correspond to either walking or running. Autoregressive models [Agarwal 2004], hidden Markov models [Pavlovic 2000], Gaussian process dynamical models [Urtasun 2006], piecewise linear models in the form of mixture of factor analyzers [Li 2007] or manifold learning based on spectral [Sminchisescu 2004] or Isomap [Wang 2003] embedding are example of method to model prior knowledge.

In the context of hand tracking several approaches has been proposed [Lin 2000, Wu 2001, Zhou 2003, Thayananthan 2003a]. In [Wu 2001] the pose manifold as been modeled as the union of low dimensional manifolds obtained from training data collected with a CyberGlove. In [Zhou 2003] the manifold has been obtained

using a 6 dimensional PCA on training data collected with a CyberGlove. The authors proposed a model of the dynamic of the hand using a structured dynamic model of order 10 within this 6 dimensional space, where each fingers bends and extends nearly independently.

Methods that constrain the hand to remain within some sub manifold [Sminchisescu 2004, Wang 2003, Wu 2001, Zhou 2003] of the Cartesian product of the individual parameter ranges are likely to be inaccurate because the variability needed to fully explain the observed image is inevitably lost. A better approach could be to increasingly penalize hand configuration as its Euclidean distance from the sub-manifold increases. However computing such a distance or the Euclidean projection onto the manifold is generally intractable for non linear cases. Some approximations could be derived using the embedding transformation and its inverse mapping but this would generally require an iterative scheme (out-of-sample problem and pre-image problem) and would result in discontinuities in the approximating distance. Existing approach using non linear pose prior are Monte-Carlo methods that do not specifically require the objective function to be continuous. Our approach relies on efficient continuous optimization and therefore requires the prior term to be continuous and differentiable.

In order to fulfill such requirements, we advocate the use of Kernel Principal Component analysis. It allows to define a continuous and tractable penalization function in the pose space that measure closeness of the hypothesized hand pose from a training set. The combination of this prior term with a data-fidelity one would improve performance over the use of the data-fidelity term alone. As we aim to use efficient Quasi-Newton minimization of the objection function, we derive the gradient of both the data-fidelity term and the prior term.

4.5.2 Prior based on Kernel PCA

In order to define the hand pose prior we apply KPCA onto a set of 22-dimensional vectors α_i. Each of these vector corresponds to a set of 22 angular parameters of an hand poses . This training set has been obtained with a specific multi-view setting described in the experimental section. For notation convenience we replace α_i by x_i in the sequel of the section.

Introduced in [Schölkopf 1998], Kernel PCA is a dimensionality reduction method that can be considered as a generalization of linear components analysis using the *kernel trick* [Aizerman 1964].

Given a non linear operator ϕ that maps the training data points x_i in an higher dimensional space (with ϕ defined explicitly or implicitly using a mercer kernel k such that $k : I \times I \rightarrow \mathbb{R}$ such that $k(x, y) =< \phi(x), \phi(y) >$), this method implicitly amounts to

- map the training data points x_i in an higher dimensional space through the non linear operator ϕ

- perform PCA analysis of the mapped points $\phi(x_i)$ in order to find the affine sub-space of dimension l (where l is either fixed in advance , or compute such that some percentage of the variance is kept) that best approximate this set of feature points in a least square sense

- define the reduced coordinates of x_i to be the coordinates (within some orthonormal base of the plane) of the projection of the mapped point onto the plane.

Using the *kernel trick*, this can performed using the kernel k without the need to explicitly compute $\phi(x_i)$ (see Appendix). This eventually allows the use of non linear mapping into infinite dimensional space, using Gaussian kernels for example.

An interesting aspect of the kernel PCA approach is that one can directly use the squared distance of the mapped point to the PCA plane in the feature space as a measure of closeness to the training set. Denoting P^l the projection operator onto the kernel PCA space this measure writes

$$D_F^2(\phi(x), P^l(\phi(x))) = \|\phi(x) - P^l(\phi(x))\| \tag{4.43}$$

Again the kernel trick allows to computed this distance measure without actually mapping the points with ϕ.

Note that this distance is expressed in the feature space and thus is not homogeneous with respect to the Euclidean distances in the input space. This might be problematic while combining this measure with a data-fidelity term.

With the kernel PCA approach, we can use the variance along each principal direction in the feature space to define a Mahalanobis distance between the projection of the point $P^l(\phi(x))$ and the mean point denoted ϕ_0 [Cremers 2003]. This results in another closeness measure, whose derivation is detailed in the appendix (eqn. A.49), and that is a dimensionless quantities and thus might not need to be rescaled. We choose this closeness measure as our prior term during our tracking experiments and refer to it as $E_{prior}(x)$.

During the tracking process, the model is registered to each new frame by minimizing the objective function that combine both the data term E_c and the pose prior E_{prior} with respect to the pose θ and the illuminant L. We denote E_2 the combined objective function:

$$E(\theta, L, T) = E_c(\theta, L, T) + \beta E_{prior}(\theta) \tag{4.44}$$

where β is a coefficient that provides control on the relative importance of the prior term over the data term.

Because we advocate the use of Quasi-Newton local optimization, we need to compute the derivate or gradient of the prior term with respect to the hand pose parameter θ. This derivation is described in the appendix.

4.6 Pose and Lighting Estimation

In order to estimate the pose θ and the lighting condition L for each new incoming frame, or to update the texture T, we look for minima of the objective function. During tracking, the optimization procedure involves two steps. First we minimize (4.36) with respect to θ and L to register the model with respect to the new image frame. Then we minimize the error function with respect to the texture T to find the optimal texture update. This alternating minimization can be interpreted as form of coordinate descent, limited to a single iteration per frame. We first consider the problem of estimating θ and L given a new frame as it constitutes the core of the tracking problem.

The simultaneous estimation of the pose parameters θ and the illuminant L is a challenging non-linear, non-convex, high-dimensional optimization problem. Global optimization is impractical and therefore we resort to an efficient quasi-Newton, iterative local optimization. This method requires that we are able to efficiently compute the gradient of E_c in (4.36) with respect to θ and L.

The gradient of E_c with respect to lighting L is straightforward to derive. At any point $\mathbf{x}$ of the image the synthetic intensity is differentiable with respect to L, i.e the function function $L \mapsto R(\mathbf{x}; \theta, L, T)$ is differentiable. Therefore differentiation and integration operators commute, and we obtain

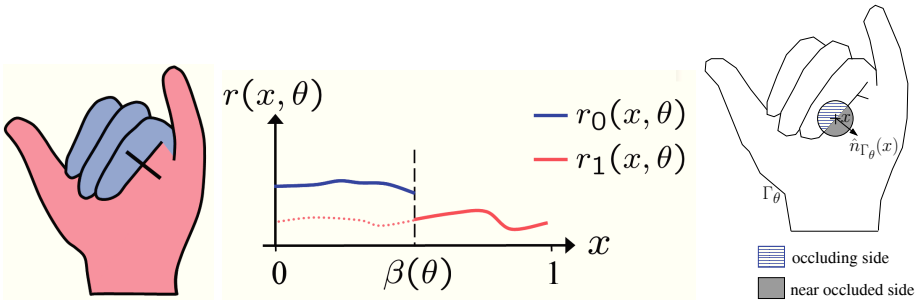
$$\begin{aligned} \frac{\partial E_c}{\partial L}(\theta, L, T) &= \frac{\partial}{\partial L} \int_\Omega R(\mathbf{x}; \theta, L, T) d\mathbf{x} \\ &= \int_\Omega \frac{\partial R}{\partial L}(\mathbf{x}; \theta, L, T) d\mathbf{x} \ . \end{aligned} \tag{4.45}$$

Computing $\frac{\partial R}{\partial L}(\mathbf{x}; \theta, L)$ is straightforward with application of the chain rule to the synthesis process.

Formulating the gradient of E_c with respect to pose θ is not straightforward to derive. For any point $\mathbf{x}$ along an occlusion boundaries, the residual is a discontinuous function of the pose parameters, i.e $\theta \mapsto R(\mathbf{x}; \theta, L, T)$ is not differentiable with respect to θ As a consequence, we cannot commute differentiation and integration. However, while the residual is a discontinuous function of the pose, the data-fidelity function remains continuous in θ and therefore the gradient with respect to θ can still be specified analytically. Its derivation is the focus of the next section.

4.6.1 Gradient with respect to Pose and Lighting

The synthesis process defined above was carefully formulated so that scene radiance is continuous over the hand surface (using Gouraud shading and patch-based texture mapping with bilinear interpolation). We further assume that the background and the observed image are known and continuous. Therefore the residual error is spatially continuous everywhere except at the self-occlusion (hand/hand) and occlusions boundaries (hand/background) denoted by Γ_θ. Because these boundaries move when θ varies, the spatial discontinuity of the residual implies a lack of differentiability with respect to θ along Γ_θ. The occlusion boundaries require special



(a) (b) (c)

Descriptio 4.7: a) Example of a segment that crosses an occlusion boundary. b) A curve representing the residual along this segment and c) The occlusion contour Γ_θ

attention when deriving the gradient of E_c with respect to pose θ. To sketch the main idea behind the gradient derivation, we first consider a 1D residual function on a line segment that crosses a self-occlusion boundary.

4.6.1.1 1D illustration

We consider the residual function and the discrepancy measure along a 1D line segment that crosses a self-occlusion boundary as shown in [Figure (4.7)]. The residual is continuous everywhere on the segment except at the self-occlusion boundary location, denoted by β. The residual function on the line can be written in two parts, namely, the residual to the left of the boundary (r_0) and the residual on the right of the boundary (r_1):

$$r(x,\theta) = \begin{cases} r_0(x,\theta), & x \in [0,\beta(\theta)) \\ r_1(x,\theta), & x \in (\beta(\theta),1] \end{cases} . \tag{4.46}$$

The data-fidelity term is then the sum of the integral of r_0 on the left and the integral of r_1 on the right side.

$$E_c = \int_0^{\beta(\theta)} r_0(x,\theta)dx + \int_{\beta(\theta)}^1 r_1(x,\theta)dx . \tag{4.47}$$

Note that β is a function of the pose parameter θ. Intuitively, when the pose parameter θ varies the integral E_c is affected in two ways. First each of the residual functions r_0 and r_1 will vary, e.g. due to shading changes. Second, the boundary location β will also move as a function of the pose θ.

Mathematicaly the total derivative of E_c with respect to θ is the sum of two terms:

$$\frac{dE_c}{d\theta} = \frac{\partial E_c}{\partial \theta} + \frac{\partial E_c}{\partial \beta}\frac{\partial \beta}{\partial \theta} . \tag{4.48}$$

The first term, the partial derivative of E_c with respect to θ with β fixed, corresponds to the integration of the residual derivative everywhere except at the

discontinuity.

$$\frac{\partial E_c}{\partial \theta} = \int_{[0,1]\setminus\{\beta\}} \frac{\partial r}{\partial \theta}(x,\theta)\, dx \ . \tag{4.49}$$

The second term (4.48) is obtained by differentiating E_c with respect to the location of the boundary β. Using the fundamental theorem of calculus, one can show that this term reduces to the difference between the residuals at both sides of the occlusion boundary, multiplied by the derivative of the boundary location β with respect to the pose parameters θ; i.e., from (4.47) it follows that

$$\frac{\partial E_c}{\partial \beta}\frac{\partial \beta}{\partial \theta} = [r_0(\beta,\theta) - r_1(\beta,\theta)]\,\frac{\partial \beta}{\partial \theta} \ . \tag{4.50}$$

Therefor, while the residual $r(x,\theta)$ is a discontinuous function of θ at the occlusion location β, the data-fidelity term E_c is still a continuous and differentiable function of θ.

4.6.1.2 General 2D case

The 2D case is a generalization of the 1D example. As in the 1D case, the residual image is spatially continuous almost everywhere, except for points at occlusion and self-occlusion boundaries; i.e., those points where the hand starts occluding the background or other parts of the hand [Figure (4.7.c)]. Let Γ_θ be the set of boundary points. One can interpret Γ_θ to be the support of depth discontinuities in the image domain.

Because we are working with a triangulated surface mesh, Γ_θ can be decomposed into a set line segments. More precisely, Γ_θ is the union of the projections of all visible portions of edges of the triangulated surface that separate front-facing facets from back-facing facets. For any point $\mathbf{x}$ along Γ_θ, the corresponding edge projection locally separates the image domain into two sub-regions of $\Omega\setminus\Gamma_\theta$ that we call the *occluding* side and the *near-occluded* side.

Let $\bar{V}_k$ be the 2D image projection of the k^{th} vertex of the 3D hand mesh. For any image point $\mathbf{x}$ on the self-occlusion boundary Γ_θ, there exist projections of two vertices indexed by $m_\mathbf{x}$ and $n_\mathbf{x}$ such that $\mathbf{x}$ belongs to the image line segment with end points $\bar{V}_{m_\mathbf{x}}$ and $\bar{V}_{n_\mathbf{x}}$. Further, we define $t_\mathbf{x} \in [0,1]$ to be the scalar satisfying $\mathbf{x} = (1-t_\mathbf{x})\bar{V}_{m_\mathbf{x}} + t_\mathbf{x}\bar{V}_{n_\mathbf{x}}$.

The normal to the line segment is given by

$$\hat{n}_{\Gamma_\theta}(\mathbf{x}) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{\bar{V}_{n_\mathbf{x}} - \bar{V}_{m_\mathbf{x}}}{\|\bar{V}_{n_\mathbf{x}} - \bar{V}_{m_\mathbf{x}}\|} \tag{4.51}$$

By ordering $n_\mathbf{x}$ and $m_\mathbf{x}$ properly we can ensure that $\hat{n}_{\Gamma_\theta}(\mathbf{x})$ is the *outward* normal to the line segment; that is, it faces toward the *near-occluded* side [Figure (4.7.c)]. Finally, the partial derivative of the curve Γ_θ with respect to a given pose parameter, say θ_j in θ, is given by

$$v_j(\mathbf{x}) = (1-t_\mathbf{x})\frac{\partial \bar{V}_{m_\mathbf{x}}}{\partial \theta_j} + t_\mathbf{x}\frac{\partial \bar{V}_{n_\mathbf{x}}}{\partial \theta_j} \ . \tag{4.52}$$

Along Γ_θ, the residual image $R()$ is discontinuous both with respect to $\mathbf{x}$ and θ. Thus $\frac{\partial R(\theta,\mathbf{x})}{\partial \theta_j}$ is not defined on Γ_θ. To deal with such discontinuities, we introduce a new residual image $R^+(\theta,\mathbf{x})$ that extends $R()$ by continuity in Γ_θ. This is done by replicating the content in $\Omega\backslash\Gamma_\theta$ of $R()$ from the *near-occluded* side. In forming R^+, we are recovering the residual of *occluded faces* in the vicinity of the occluding boundary points, $\mathbf{x}$. Formally, this is done using a infinite sequence of points that approaches $\mathbf{x}$ from the *near-occluded* side:

$$R^+(\theta,\mathbf{x}) = \lim_{k\to\infty} \left(R\left(\theta, \mathbf{x}+\frac{\hat{n}_{\Gamma_\theta}(\mathbf{x})}{k}\right)\right) \tag{4.53}$$

One can also interpret $R^+(\theta,\mathbf{x})$ on Γ_θ as the residual we would have obtained if the near-occluded surface had been visible instead of the occluding surface.

When the pose parameter θ_j is changed with an infinitesimal step $d\theta_j$, the occluding contour Γ_θ in the neighborhood of $\mathbf{x}\in\Gamma_\theta$ moves with an infinitesimal step $v_i(\mathbf{x})d\theta$ and the residual in the vicinity of $\mathbf{x}$ will change in a discontinuous manner, *jumping* between $R^+(\theta,\mathbf{x})$ and $R(\theta,\mathbf{x})$. However, the surface area where this jump occurs is also infinitesimal and proportional to $\big(v_j(\mathbf{x})\hat{n}_{\Gamma_\theta}(\mathbf{x})\big)d\theta_j$. Therefore this jump induces an infinitesimal change in the objective functional after integration over the continuous image domain Ω.

Like the 1D case, one can formally derive the exact functional gradient $\nabla_\theta E_c \equiv (\frac{\partial E_c}{\partial\theta_1},\ldots,\frac{\partial E_c}{\partial\theta_n})$ using two terms:

$$\frac{\partial E_c}{\partial\theta_j} = \int_{\Omega\backslash\Gamma_\theta} \frac{\partial R}{\partial\theta_i}(\mathbf{x};\theta,L,T)d\mathbf{x} - \int_{\Gamma^\theta} \underbrace{\big[R^+(\mathbf{x};\theta,T,L-R(\mathbf{x};\theta,L,T))\big]\hat{n}_{\Gamma_\theta}(\mathbf{x})}_{f_{oc}} v_i(\mathbf{x})d\mathbf{x} \tag{4.54}$$

The first term captures the data-fidelity term variation due to the smooth variation of the residual in $\Omega\backslash\Gamma_\theta$. It integrates the partial derivative of the residual R with respect to θ everywhere but at the occluding contours Γ_θ, where it is not defined. The analytical derivation of these partial derivatives simply requires application of the chain rule to the functions that define the full synthesis process.

The second term captures the data-fidelity term variation due to the displacement of the occluding contours. It integrates the difference between the residual function on both sides of the occluding contour, multiplied by the normal speed of the boundary when the pose varies. The term f_{oc} is a vector field that associates a 2D vector to each point on the occluding contour:

$$\begin{aligned} f_{oc} :&\Gamma_\theta \to \mathbb{R}^2 \\ f_{oc}(\mathbf{x}) =&\big[R^+(\mathbf{x};\theta,T,L) - R(\mathbf{x};\theta,L,T)\big]\hat{n}_{\Gamma_\theta}(\mathbf{x}) \end{aligned} \tag{4.55}$$

The vector field directions are normal to the curve, and their norms are proportional to the difference of the local cost at each side of the curve. We call these vectors *occlusion forces* by analogy with mechanics. These forces account for the displacement of occlusion and self-occlusion contours when θ varies. They bear some similarities to the forces obtained with 2D active regions [Paragios 1999]. This similarity with

2D active regions derives from the fact that we kept the image domain Ω continuous while computing the functional gradient. Because the surface is triangulated, Γ_θ can be decomposed into a set of image line segments and we can rewrite expressions similar to the ones reported in [Unal 2005] for active polygons.

Our *occlusion forces* also bear similarity to the gradient flows of [Gargallo 2007, Delaunoy 2008] for multi-view reconstruction, where some terms account for the change of visibility at occlusion boundaries. However, in their formulation no shading and texture are attached to the reconstructed surface, which results in substantial differences in the derivation.

4.7 Numerical computation of E_c and $\nabla_\theta E_c$

Computing the data term in the objective function E_c and its gradient with respect to the pose parameters must be done carefully. We devote the next section to explaining the implementation of the objective function and its gradient computation in greater detail.

4.7.1 Exact computation of the matching cost

Computing the discontinuous matching cost E_d is straightforward. One simply needs to synthesize the image for each the discrete location of the pixel grid. As in the previous chapter, computing the continuous matching cost E_c is very desirable because this could be used to numerically test the validity of the analytical derivation and implementation of the gradient (which became quite complex to implement due to the complexity of the rendering process) using small increments on the pose parameters. This is also desirable for the descent-gradient optimization routine that requires the minimized function to be continuous and derivable.

We have mentioned in section 4.4.3 that the objective functions derived from a pre-filtered anti-aliasing formulation (eqn.4.27) and the continuous image-domain formulation are not equivalent (eqn.4.36). However we could still derive some theoretical connection between the problem of computing the exact matching cost and the one of performing exact pre-filtered anti-aliasing. Unfortunately, computing the exact matching cost or performing exact pre-filtered anti-aliasing is much more difficult than in the previous chapter. In order to perform exact (up to the machine precision) we can decompose the integral of the residual within the silhouette $\chi(\theta, \vartheta)$ as the sum of integral on the visible part of each facet. To this aim we can re-express the integral in equation 4.40 as a sum on integral on each facet:

$$E_\chi(\theta, \vartheta, T, L) = \sum_{j=1}^{K} \int_{S_j(\theta,\vartheta)} v(s;\theta,\vartheta) R(\mathbf{x};\theta,\vartheta,L,T) w(s;\theta,\vartheta) ds \tag{4.56}$$

We define the visible part of the j^{th} facet the set:

$$S_j^v(\theta,\vartheta) \equiv \{x \in S_j(\theta,\vartheta) | v(x;\theta,\vartheta = 1)\}, \tag{4.57}$$

which can be equivalently be defined as:

$$S_j^v(\theta,\vartheta) \equiv \Pi^{-1}(\Pi(S_j(\theta,\vartheta))) \cap S_j(\theta,\vartheta). \tag{4.58}$$

The integral on the surface domain rewrites

$$E_\chi(\theta,\vartheta,T,L) = \sum_{j=1}^{K} \int_{S_j^v(\theta,\vartheta)} R(\mathbf{x};\theta,\vartheta,L,T) w(s;\theta,\vartheta) ds \tag{4.59}$$

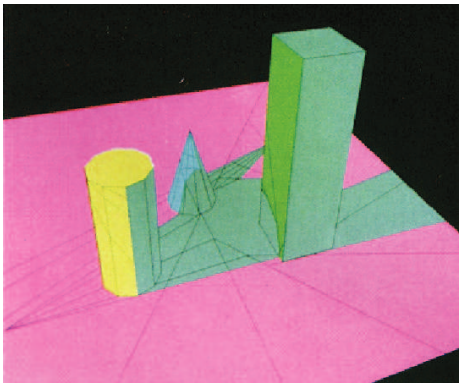
We define the projected visible part of the j^{th} triangular facet $\bar{S_j^v} \equiv \Pi(S_j^v(\theta,\vartheta))$. With this notation, the cost function rewrites:

$$\begin{aligned} E_\chi(\theta,\vartheta,T,L) &= \sum_{j=1}^{K} \int_{\bar{S_j^v}} R(\mathbf{x};\theta,\vartheta,L,T) d\mathbf{x} \\ &= \sum_{j=1}^{K} \int_{\bar{S_j^v}} \rho\big(\|I_{syn}(\mathbf{x};\theta,L,T) - \tilde{I}_{obs}(\mathbf{x})\|\big) d\mathbf{x} \end{aligned} \tag{4.60}$$

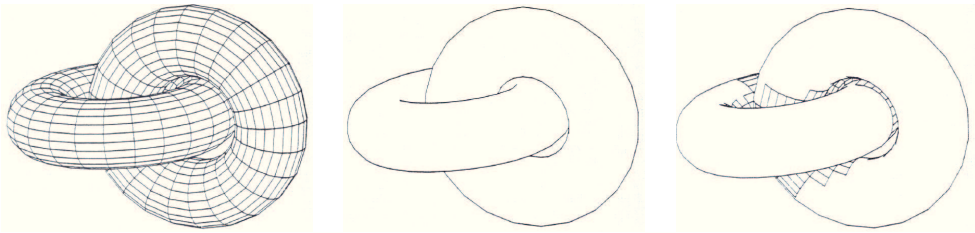
Each projected visible part of a triangular facet can be described as empty set (if the facet is completely hidden), a single polygon or a set of polygons with eventually polygonal holes. In order to compute exactly (up to machine precision) the cost function, we need an analytical description of these polygons, i.e a list of real valued vertices and lists of edges. Within the computer graphics literature this is referred as *model-space* hidden surface removal, by opposition to *image-space* hidden surface removal that works on the discrete pixel lattice on the projection plane, deciding visibility point by point a each discrete pixel position, like the well known depth-buffer (a.k.a z-buffer) algorithm. The methods described in [Roberts 1963, Appel 1967, Galimberti 1969, Nakamae 1972, Weiler 1977, Hornung 1984, Markosian 1997] are examples of such *object-space* methods. The problem of removing hidden surface is related to hard shadow casting, and some shadow casting method [Nishita 1974, Crow 1977, Chin 1989, Atherton 1978, Chrysanthou 1995, Lischinski 1992, Stewart 1994] actually perform model-space hidden surface removal from the light point of view as a first pass in order to subdivide polygon into sub-polygons that are fully visible from the light source (lit) and ones that are fully hidden (shadowed) as seen in figure 4.8.

In contrast with most methods found in the literature, the methods proposed in [Appel 1967, Hornung 1984, Markosian 1997] take advantage of the connectivity of polyhedral objects to reduce the computational complexity. These methods are of great interest in our context as the hand surface is composed of a single connected triangulated surface. We use the method presented in [Markosian 1997] in the context of non-photo-realistic rendering. The method used for hidden-surface removal can be summarized in two steps.

- The first step consists in computing the set of segments composing the silhouette and self-occlusions (denoted Γ_θ in section 4.6.1.2) by performing



Descriptio 4.8: Scene with split polygons from [Chin 1989]



a) b) c)

Descriptio 4.9: Image extracted from [Hornung 1984]: a) the scene composed of two torus b) the set of silhouette and self-occlusion segments c) The visible part of the partially occluded polygons

hidden-line removal on the set of edges connected to both a front-facing and a back facing facet (referred as *silhouette edges* in the paper). This is done using a modified version of the Appel's algorithm [Appel 1967] that decrease the number of ray-surface intersection tests by using the topology of singular maps of a surface into the plane. The original method use a randomized search of *silhouette edges* to gain speed. Instead, we perform an exhaustive search as we aim at computing exactly the cost function.

- The second step consists in deriving visibility information across surface regions. This implies subdividing each partially visible facet into fully visible ($\{\bar{S}_j^v\}_{j=1}^K$) or fully hidden polygons (see fig.4.9). In [Markosian 1997] this is done using a method proposed in [Hornung 1984] that takes the connectivity of the mesh into account in order to reduce the number of of ray-surface intersections tests.

Once we obtained the polygonal regions $\{\bar{S}_j^v\}_{j=1}^K$ we can tackle the problem of computing the integral of the residual in each region. Within each polygonal region $\bar{S}_j^v$, thanks to the use of our approximate linear projection $\hat{\Pi}$ defined in eqn.4.19, the

mapping of from the texture coordinate systme the image plan coordinate system is a linear function. As a consequence, I_{syn} is piecewise quadratic with respect to $\mathbf{x}$, on a grid that is composed of uniform parallelograms (texels grid linearly mapped in the image plan). The function $\tilde{I}_{obs}$ is piecewise bilinear on the square grid defined by the pixels. Therefore, using $\rho(t) = t^2$, the residual is a piecewise polynomial of order four in $\mathbf{x}$, where each element of the associate plane partition is an intersection of a square pixel region and a parallelogram. By splitting each segment of the polygons defining $\bar{S}_j^v$ against both grids and using a method that allows to integrate general polynomial functions within polygonal regions [Tumblin 2006, Steger 1996, Liggett 1988, Singer 1993, Strachan 1990] (as already mentioned in section 3.5.1.2), the exact integral could eventually be computed. However this seems a bit too complex for our purpose. In section we justified the choice of using texture rather than using of a single albedo associated to each vertex (and linearly interpolated across each facet) by that fact that retaining albedo details would require very fine triangulation. However this choice lead to the technical difficulty we are facing here. We will therefore temporally resort to per-vertex definition of the albedo, where the albedo is linearly interpolated on each facet. In order to get I_{syn} to be linear in the region $\bar{S}_j^v$, and not homographic, we continue using the approximate projection $\hat{\Pi}$ defined in 4.19 while mapping the surface albedo into the image plan. Using $\rho(t) = t^2$ and a nearest neighbor interpolation to define I_{obs}, the residual is now piecewise linear on the pixel grid. The exact integral of this piecewise linear function into the polygonal region $\bar{S}_j^v$ could be computed using a method similar to the one detailed in section 3.5.1.1. If I_{obs} is defined using a bilinear interpolation, the residual is a piecewise polynomial of order three and it could be possible to use the methods in [Tumblin 2006, Steger 1996, Liggett 1988, Singer 1993, Strachan 1990], as already mentioned in section 3.5.1.2. A method proposed in [McCool 1995] allows to perform exact antialiasing for linearly shaded triangles could also be of interest.

The computation of the gradient as expressed in eqn.4.54 could also be computed exactly by decomposing the 2D integrals within the silhouette as a sum of integrals over the projection of the visible parts of the facets, and the 1D integrals as a sum of integral on visible part of silhouette edges. One advantage of such an approach is that the analytical expression of the Hessian of the cost function could also be formulated and exactly computed, which allow to use advance trust region methods for the optimization. Such an optimization approach led to a important increase of the convergence rate of the hand fitting process using the silhouette-base cost function (see section 3.7.4), and one could expect such in improvement using the texture and shading based cost function.

Note that the exact visibility computation could be also be use to introduce hard cast shadows in our model by splitting each facet into lighted and shaded regions (using the visibility from the light point of source). The speed of the displacement of the shadow boundaries in the image plane as the hand pose change could be formulated easily using this object-space approach of the visibility computation, and again exact computation of the cost function including projected shadows and

its derivatives could be carried out. This would yield to shadow forces obtained by integrating the difference of the residual between the two sides of each cast shadow boundary.

At the time of the writing of this manuscript, the method discussed in this section to compute the visible regions projections $\{\bar{S}_j^v\}_{j=1}^K$ has been only partially implemented and thus further experiment based on the exact computation of the cost function have not been carried out. However, as explained in section 4.7.3, this does not mean that we had to settle for the discontinuous matching cost E_d in the optimization procedure.

4.7.2 Direct discretization of the gradient

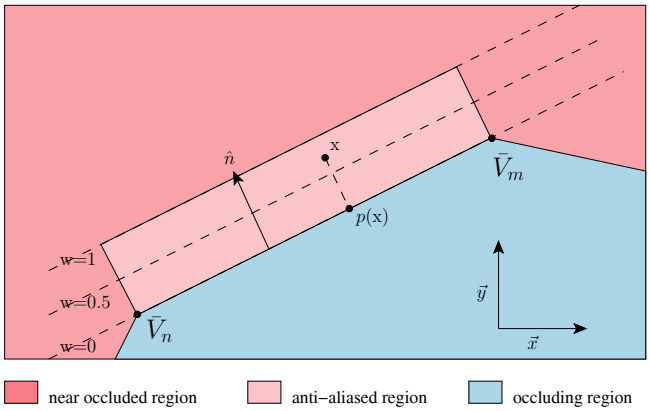
A possible approach to computing the gradient $\nabla_\theta E_c$ involves the discretization of the two terms in the (4.54). The integral over $\Omega \backslash \Gamma_\theta$ could be approximated by a finite sum over image pixel in $\Omega \cap \mathbb{N}^2$ while the integral along contours Γ_θ could be approximated by sampling uniformly points along each segment.

Let denote $\tilde{\nabla}_\theta E_c$ the numerical approximation to the gradient we obtain this way.There are two practical problems with this approach.

The first difficulty occurs when using $\tilde{\nabla}_\theta E_c$ in conjunction with E_d while using standard gradient-descent optimization methods. Most[1] iterative gradient-based optimization methods require direct access to a routine that allows the numerical evaluation of the objective function (here $E_d + \beta E_{prior}$). This routine is used in order to ensure that matching cost decreases at each iteration. Such optimization methods expect the energy to be continuous and the computed gradient to be valid i.e to be consistent with the objective function up to first-order. Unfortunately this is not the case while using $\tilde{\nabla}_\theta E_c$. Using $\nabla_\theta E_d$ would provide a valid the first order approximation of E_d but this approximation is only valid for sub pixel displacements where none of the occlusion boundary crossed the center of a pixel.

The second difficulty stems from the fact that the gradient is intricate to implement. It would be highly desirable to have a method that allows checking the validity of the implemented gradient. One could compare the implemented gradient with the gradient obtained by numerical differentiation of E_d (E_c being not computable). Unfortunately the two gradients are not comparable: Due to the image aliasing, the numerical derivative of E_d does not capture the variation of the visibility around the occlusion boundaries. The gradient $\tilde{\nabla}_\theta E_c$ does not allow one to approximate E_d even at the first order and thus $\nabla_\theta E_d$ cannot be used to test the validity of the gradient computation.

[1] A recent paper [Wilke 2009] proposes some optimization methods that do not require to evaluate the matching cost but require only its gradient. It could be of interest to test how robust are these methods when using an approximate gradient such as the one defined here



Descriptio 4.10: Antialiasing weights for points in the vicinity of the segment

4.7.3 Differentiable discretization of the energy

Another approach is to introduce a discretization of the energy, denoted $\bar{E}_c$, which is a continuous function of θ, and then derive its exact gradient $\nabla_\theta \bar{E}_c$ using the chain rule. Using a special anti-aliasing technique inspired by the *discontinuity edge overdraw* method proposed by [Sander 2001], we adopt a new residual $\bar{R}$ that is a continuous a differentiable function of θ. The summation $\bar{E}_c$ of this residual over pixels is also a continuous function of θ.

Our anti-aliasing technique is applied to pixels along occlusion boundaries and progressively blends the residuals at both sides of the occlusion boundary. If the point $\mathbf{x}$ is close to an edge segment that corresponds to an occlusion boundary we compute a blending weight that is proportional to the signed distance to the edge.

$$w(\mathbf{x}) = \frac{(\mathbf{x} - p(\mathbf{x})) \cdot \hat{n}}{max(|\hat{n}_x|, |\hat{n}_y)|)} , \tag{4.61}$$

where $\hat{n} = (\hat{n}_x, \hat{n}_y)$ is the unit normal to the segment pointing toward the *near-occluded* side of the occlusion boundary. Dividing the signed distance by $max(|\hat{n}_x|, |\hat{n}_y|)$ will help to identify terms of the gradient $\nabla_\theta \bar{E}_c$ with the *occlusion forces* (see the Appendix section A.3).

Given a segment $\overline{\bar{V}_m \bar{V}_n}$ on the occluding contour, we define an associated *anti-aliased region*, denoted $\mathcal{A}$, to be a set of points that lie in the vicinity of the segment on the occluded side [Figure (4.10)]. In particular, it is defined to be the set of points that projects within the segment with weights in $[0, 1)$:

$$\mathcal{A} = \{\mathbf{x} \in \mathbb{R}^2 | w(\mathbf{x}) \in [0, 1), p(\mathbf{x}) \in \overline{\bar{V}_m \bar{V}_n}\} \tag{4.62}$$

For each point in this anti-aliased region we define the anti-aliased residual to be a linear combination of the residual on the occluded side and the residual on the occluding side:

$$\bar{R}(\mathbf{x}; \theta, L, T) = w(\mathbf{x}) R(\mathbf{x}; \theta, L, T) + (1 - w(\mathbf{x})) R(p(\mathbf{x}); \theta, L, T) \tag{4.63}$$

The blending is non-symmetric with respect to the line segment, but shifted toward the occluded side. This allows the use of a Z-buffer to handle occlusions (see [Sander 2001] for more detail). For all points outside the anti-aliased regions we simply take$\bar{R}(\mathbf{x};\theta,L,T) = R(\mathbf{x};\theta,L,T)$.

Using this new anti-aliased residual image, $\bar{R}$, we define a new approximation to the objective function, denoted $\bar{E}_c$:

$$\bar{E}_c(\theta,T,L) \;=\; \sum_{\mathbf{x}\in\Omega\cap\mathbb{N}^2} \bar{R}(\mathbf{x};\theta,L,T) \tag{4.64}$$

This anti-aliasing technique makes $\bar{R}$ and thus $\bar{E}_c$ continuous with respect to θ even along occlusion boundaries. When θ varies continuously, the blending weights w and the residual also vary continuously.

One can consider other standard anti-aliasing methods such as over-sampling [Crow 1981] or A-buffer [Carpenter 1984] for example. These techniques would reduce the magnitude of the jump in the residual when an occlusion boundary crosses the center of a pixel as θ changes, making this jump visually imperceptible, but the residual would numerically remain a discontinuous function of θ. The edge overdraw method ensures the continuity of the residual with respect to θ. $\bar{E}_c$ is now continuous and differentiable almost everywhere.

We derive $\nabla_\theta \bar{E}_c$ by differentiating $\bar{R}$ using the chain rule.

$$\nabla_\theta \bar{E}_c \;=\; \sum_{\mathbf{x}\in\Omega\cap\mathbb{N}^2} \frac{\partial}{\partial\theta}\bar{R}(\mathbf{x};\theta,L,T)\;. \tag{4.65}$$

Using a backward ordering of derivative multiplications (called adjoint coding in the algorithm differentiation literature [Griewank 2000]), we obtain an evaluation of the gradient with a computational cost that is comparable to the evaluation of the objective function.

The gradient of $\bar{E}_c$ with respect to θ, i.e., $\nabla_\theta \bar{E}_c$, is one of the possible discretization of the analytical $\nabla_\theta E_c$. The differentiation of the anti-aliasing process with respect to θ yielded terms that sum along the edges and that can been interpreted as a possible discretization of the *occlusion forces* in equation 4.55). This is demonstrated in the appendix A.3. The advantage of this second approach over the first one is that now $\nabla_\theta \bar{E}_c$ is the exact gradient of a continuous function $\bar{E}_c$ and this function can be numerically evaluated. This also allows one to check validity of the gradient $\nabla_\theta \bar{E}_c$ obtained using the chain rule by direct comparison with the gradient obtained by divided differences on $\bar{E}_c$. Divided differences are known to be prone to round-off errors and computationally inefficient, but they provide one way to check the validity of the gradient computation.

4.8 Model Registration

4.8.1 Sequential Quadratic Programming

During the tracking process, the model of the hand is registered to each new frame by minimizing the objective function E_c with respect to the pose θ and the illuminant L. Let $P = [\theta, L]^T$ comprise the unknown pose and lighting parameters, and assume that we are given an initial guess by extrapolating estimates from the previous frames. We refine the pose and lighting estimates by minimizing E_c with respect to X, subject to linear constraints that model the articulated joint limits.

$$\tilde{P} = \underset{P}{argmin} \quad E_c(P) \qquad \text{such that } AP \leq b \tag{4.66}$$

Because the gradient of the objective function is available, we minimize (4.36) efficiently using a sequential quadratic programming method (SQP) [Conn 2000] with an adapted Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian approximation. This allows us to combine the well-known BFGS quasi-Newton method with the linear constraints that limit the range of the articulation angles of the hand pose.

The method comprises four steps:

- First, a quadratic program is used to determine a descent direction. This makes use of the energy gradient and an approximation to the Hessian based on a modified BFGS procedure denoted $\tilde{H}_t$:

$$\Delta_P = \underset{\Delta_P}{argmin} \quad \left(\frac{dE_c}{dP}(P_t)\Delta_P + \frac{1}{2}\Delta_P^t \tilde{H}_t \Delta_P \right) \qquad \text{s.t. } A(P_t + \Delta_P) \leq b \tag{4.67}$$

- A line search is then performed in that direction:

$$\lambda^* = \underset{\lambda}{argmin}\, E_c(P_t + \lambda\Delta_P) \;\text{ s.t. } \lambda \leq 1\ . \tag{4.68}$$

  The inequality $\lambda \leq 1$ ensures that we stay in the linearly constrained subspace.

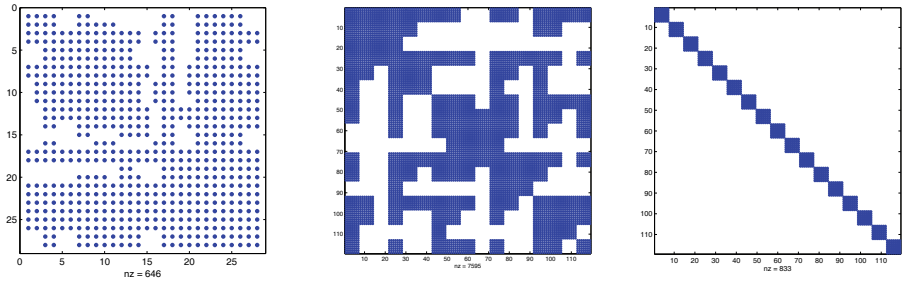
- The pose and lighting parameters are updated with the line optimum

$$P_{t+1} = P_t + \lambda^* \Delta_P\ . \tag{4.69}$$

- And finally the approximate Hessian $\tilde{H}_{t+1}$ is updated using the adapted BFGS formula.

These four steps are iterated until convergence.

Note that the BFGS update method was adapted slightly to take advantage of the partial independence of separated fingers regarding the data term. With such



(a) (b) (c)

Descriptio 4.11: Sparsity structure: (a) $\theta \in \mathbb{R}^{28}$ (b) $\theta \in \mathbb{R}^{119}$ (c) approximated structure

independence, some sparseness of the Hessian of the data term can be exploited. We adapted the BFGS update method to define a *Block-wise BFGS update* that takes advantage of this structure explained in the next section. The Hessian of the prior term E_{prior} does not exhibit such sparsity pattern we choose to maintain separately the Hessian approximation of each of the two terms, applying the BFGS update formula independently for the data-fidelity term and the prior term. Using this method we increased the convergence rate by a factor ranging from 3 to 6.

Because our optimization method is local, we need to initialize the search. To obtain reasonable starting points we first perform a one-dimensional search along the line that linearly extrapolates the two previous pose estimates. Each local minima obtained during this 1D search is used as a starting point for our SQP method in the full pose space. The number of starting points found is usually just one, but when there are several local minima, the results of the SQP method are compared and the best solution is chosen. Once the model has been fit to a new image frame using this optimization method, we update the texture model that is used for registration with the next frame.

4.8.2 Blockwise BFGS update

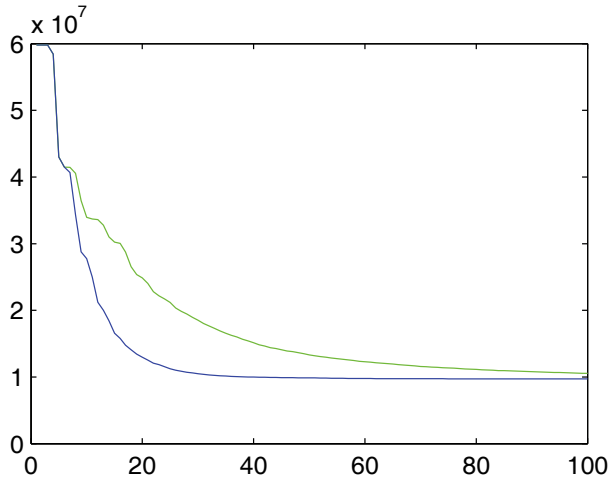
The better the approximation of the Hessian is, the faster a Quasi-Newton optimization method will converge. To obtain a good convergence rate even during the first iterations, it is also important to carefully initialize the approximated Hessian. Therefore, rather than using the identity matrix as often done, we used a scaled version of the matrix ${J_\theta^{\bar{V}}}^t J_\theta^{\bar{V}}$ where $J_\theta^{\bar{V}}$ is the Jacobian of the projected vertices with respect to θ. This initialization favors the displacements in the depth direction for which the gradient is small due to weak support from the image data.

Because the contributions to the overall data cost of two well-separated fingers are independent, the true Hessian of E_c will not be fully populated but will exhibit blocks of zeros [Figure (4.11.a)]. The sparsity of the Hessian of the data term E_c is accentuated if, instead of using joints angles, we parameterize individually the

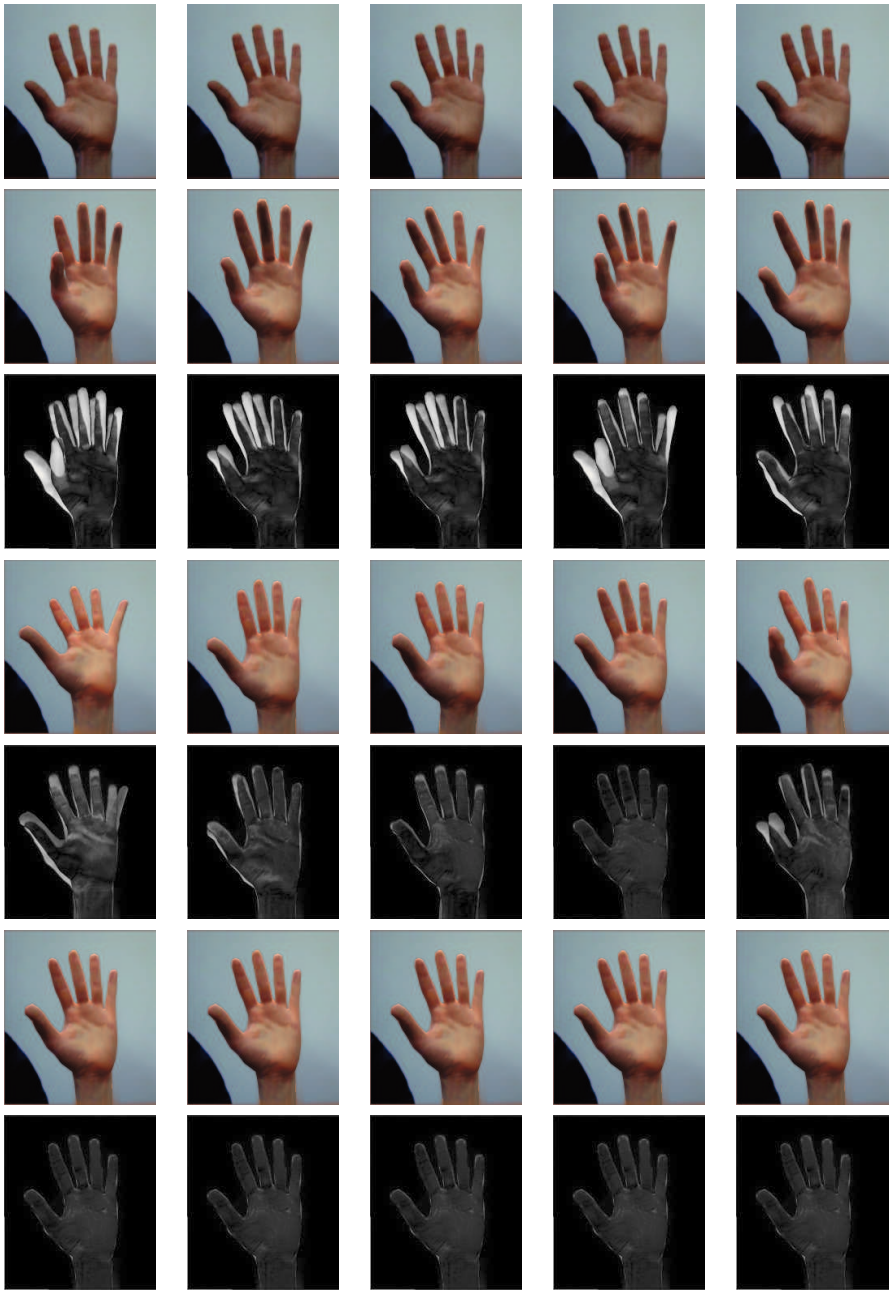
pose of each of the 18 bones using a 7D vector consisting of a quaternion and a translation vector such that $\theta \in \mathbb{R}^{119}$ [Figure (4.11b)]. Non-zero entries of the 119 by 119 Hessian appear on 7 by 7 blocks. Each block that is not on the diagonal corresponds to a pair of hand parts that either occlude each other or share some facets in their influence area when the pose space deformation method is used. Using quaternions for the hand pose parameterization would facilitate the use of finger independences in the minimization process. Unfortunately this would require additional non-linear equality constraints between quaternions to enforce validity of relative poses between linked bones. Such non-linear equality constraints are difficult to handle in a continuous optimization framework and are likely to decrease the convergence rate.

To exploit the sparsity of the Hessian of the data term when quaternions are used for the pose parameterization without the need for additional non-linear constraints, we first decompose the function $E_c(\theta)$ into $E_c(\theta) = E_q(Q(\theta))$, where Q maps the joints-angles pose representation to the quaternion representation. The Hessian $\frac{\partial^2 E_q}{\partial^2 \theta}$ is then approximated by $(\frac{\partial Q}{\partial \theta})^t H_q (\frac{\partial Q}{\partial \theta})$ where $H_q = (\frac{\partial^2 E_q}{\partial^2 Q})$. At each step, we refine the approximation to the Hessian H_q with an adapted BFGS update.

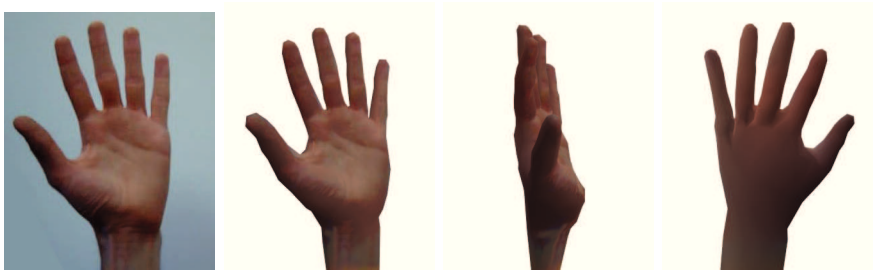
We approximate the structure of H_q by assuming complete independence between parts of the hand. This results in a block-diagonal structure [Figure (4.11.c)] where non-zero entries are restricted to the 7 by 7 blocks along the diagonal. The standard BFGS update does not exploit the sparsity structure of the approximated Hessian and would populate the entire matrix with non-zeros values. Using the BFGS formula, we do not update the whole matrix H_q, but we update each non-zero 7 by 7 block on the diagonal of the matrix independently. About 7 gradient evaluations are then necessary to obtain a reasonable local approximation of the Hessian of the data term whereas the standard BFGS method would require about 28 evaluations. This has a direct impact on the convergence rate of the minimization. The method induces more zeros than in the true Hessian but still leads to significant improvement over the standard BFGS update. As we keep performing increments on the θ vector during the optimization, we do not need to add non-linear constraints to enforce validity of relative poses between linked bones that would be necessary if increments where done in the quaternion representation space. The improvement in the minimization process, in terms of the number of calls to the objective function, is illustrated in [Figure (4.12)] where we estimated the pose for a single frame and we average the matching cost curve for 100 trials with random initialization around the optimal pose. We also present in [Figure (4.13)] five results obtained after 100 iteration with the classic BFGS method and the Blockwise BFGS method.



Descriptio 4.12: Averaged (100 trials) functional decrease with respect to number of functional evaluations. blue: adapted BFGS. green: normal BFGS



Descriptio 4.13: Each raw corresponds respectively to 1) the observed image 2) the initial pose 3) the residual for the initial pose 4) the optimal pose found with BFGS after 100 iteration 5) The residual for BFGS after 100 iterations 6) the optimal pose found with Blockwise BFGS after 100 iterations 3)The residual of Blockwise BFGS after 100 iterations



Descriptio 4.14: Observed image and extracted texture mapped on the hand under the camera view point and two other view points. The texture is smoothly diffused into regions that are hidden from the camera view point

4.8.3 Texture update

4.8.3.1 Formulation

Various methods for mapping images onto a static 3D surface have been proposed in the literature [Zhou 2005b, Wang 2001]. Perhaps the simplest method involves, for each 3D surface point, (1) computing its projected coordinates in the image plane, (2) checking its visibility by comparing its depth with the depth buffer at those image coordinates, and (3) if the point is visible, interpolating image intensities at those coordinates and then recovering the albedo by dividing the interpolated intensity by the model irradiance at that point. This approach is not suitable for our formulation for several reasons. First, we need to interpolate values in hidden parts of the hand, as those regions may become visible in the next frame. Second, we need to update the texture robustly to avoid progressive contamination by the background color, or self-contamination between different parts of the hand.

Here, we formulate texture estimation as the minimization of the same objective functional as that used for tracking, in combination with a smoothness regularization term (which does not depend on pose or lighting). That is, for texture $T \in \mathbb{T}$ we minimize:

$$\hat{T} = \underset{T}{arg\,min}\, E_{text}(T) \text{ s.t } T \in \mathbb{T} \tag{4.70}$$

With:

$$E_{text}(T) = E_d(\theta, L, T) + \beta E_{sm}(T) \tag{4.71}$$

Where β control the weight of a smoothing term $E_{sm}(T)$ that is defined in eqn4.72 and that penalize the local variations of the surface color. Due to the smoothness term, the albedo is smoothly diffused (inpainted) to the texels that do not contribute to the image i, either because they are associated to a part that is hidden from the camera [Figure (4.14)] or because of texture aliasing artifacts. Due to the image domain discretization some texels that belongs to visible parts of the surface may not contribute to any pixel of the image. This is likely to happen when image resolution is coarser than projected texture resolution.

4.8.3.2 The smoothness term

In order to ease the minimization of E_{text} we use a smoothness term that is a convex quadratic function of the texture image T. We use the notation f_T defined in eqn.4.14.

We write this smoothing term as:

$$E_{sm}(T) \equiv \frac{1}{2} \int_{S(\theta,\vartheta)} \|J^S_{f_T}(s)\|^2 ds \tag{4.72}$$

where $J^S_{f_T}(s)$ is the 3 by 3 intrinsic Jacobian matrix of f_T on the manifold $S(\theta, \vartheta)$. Denoting $f_T(x) = [f_{T[1]}(x), f_{T[2]}(x), f_{T[3]}(x)]^T$, the i^{th} line of this Jacobian matrix $J^S_{f_T}(s)$ corresponds to the intrinsic gradient of the function $x \mapsto f_{T[i]}(x)$ on the manifold $S(\theta, \vartheta)$. Therefore the Frobenius norm $\|J^S_{f_T}(s)\|^2$ can be written as

$$\|J^S_{f_T}(s)\|^2 = \sum_{i=1}^{3} \|\nabla_S f_{T[i]}(s)\|^2 \tag{4.73}$$

with $\|\nabla_S f_{T[i]}(s)\|^2$ the first differential parameter of Beltrami [Kreyszig 1991] of the function $x \mapsto f_{T[i]}(x)$. The surface is composed of triangular facets ($S(\theta, \vartheta) = \bigcup_{j=1}^{N_f} S_j(\theta, \vartheta)$) and thus the smoothing term can be decomposed into a sum over the facets:

$$E_{sm}(T) \equiv \sum_{j=1}^{N_f} E^j_{sm}(T) \tag{4.74}$$

with

$$E^j_{sm}(T) = \int_{S_j(\theta,\vartheta)} \|J^S_{f_T}(s)\|^2 ds \tag{4.75}$$

In order to compute $J^S_{f_T}$ for the points in the facet S_j we consider the local parameterization of the facet using two dimensional variable $\alpha = (\alpha_1, \alpha_2) \in \mathbb{R}^2$ with $\alpha_1 \geq 0$ and $0 \leq \alpha_2 \leq 1 - \alpha_1$. In order to be able to use the function h defined with barycentric coordinates, we define the mapping w:

$$w : (\alpha_1, \alpha_2) \mapsto (1 - \alpha_1 - \alpha_2, \alpha_1, \alpha_2, j) \tag{4.76}$$

The parameterization is of the surface is done using p defined as:

$$p(\alpha) \equiv g \circ w(\alpha) = v_1 + \alpha_1(v_{2j} - v_{1j}) + \alpha_2(v_{3j} - v_{1j}) \tag{4.77}$$

with $v_{ij} \equiv V(F(i, j))$

From 4.14 we obtain

$$f_T \circ p(\alpha) = r(h \circ w(\alpha), T) \tag{4.78}$$

Following [Jin 2003, Delaunoy 2009] we rewrite the term using the fundamental forms.

The Jacobian matrix of p is a 3 by 2 matrix that remain constant over the domain spanned by α and writes

$$J_p = [v_{2j} - v_{1j}, v_{3j} - v_{1j}] \tag{4.79}$$

We denote $J_{f_T \circ p}(\alpha)$ the 3 by 2 Jacobian matrix of $f_T \circ p$ evaluated at α. The intrinsic Jacobian matrix $J^S_{f_T}(p(\alpha))$ can be defined as the minimum norm (Frobenius) matrix that verifies $J^S_{f_T}(p(\alpha))J_p = J_{f_T \circ p}(\alpha)$ i.e.:

$$J^S_{f_T}(p(\alpha)) = \text{argmin}_X \|X\| \text{ s.t } XJ_p = J_{f_T \circ p}(\alpha) \tag{4.80}$$

This is solved using the left inverse of J_p defined by $B_j \equiv (J_p^T J_p)^{-1} J_p^T$:

$$J^S_{f_T}(p(\alpha)) = J_{f_T \circ p}(\alpha) B_j \tag{4.81}$$

Note that the matrix $(J_p^T J_p)$ is a 2 by 2 matrix whose coefficients are called the first fundamental form coefficients.

The smoothness term is quadratic with respect to T and we can compute a sparse matrix H_{sm} such that

$$E_{sm}(T) = [T]_l^T H_{sm} [T]_l \tag{4.82}$$

Whose derivation is detailed in the next section.

4.8.3.3 Texture Smoothness Matrix

In section 4.3.2 we assumed that a facet in the triangulated surface is associated with a triangle in the texture image that has is either the upper-left or the lower-right half part of a square whose size is K_T by K_T and whose vertices have integer coordinates. Without loss of generality let consider a facet indexed by j whose associated triangle is the upper-left part of a square. For notations convenience we also assume that we have:

$$\begin{aligned} V_T(F_T(1,j)) &= [0,0]^T \\ V_T(F_T(2,j)) &= [K_T,0]^T \\ V_T(F_T(3,j)) &= [0,K_T]^T \end{aligned} \tag{4.83}$$

We have $h \circ w(\alpha) = K_T \alpha$. The 2 by 2 Jacobian matrix of $h \circ w$ is constant for the facet j and writes

$$J_{h \circ w} = K_T I_{2\times 2} \tag{4.84}$$

The 3 by 2 Jacobian matrix of $f_T \circ p$ writes

$$\begin{aligned} J_{f_T \circ p}(\alpha)) &= J_r(h \circ w(\alpha)) J_{h \circ w} \\ &= K_T J_r(h \circ w(\alpha)) \end{aligned} \tag{4.85}$$

With r defined in eqn.4.13 and J_r its 3 by 2 Jacobian matrix. The smoothing term for the facet $S_i(\theta, \vartheta)$ writes

$$\begin{aligned} E^j_{sm}(T) &= c_j \int_{\alpha_1=0}^{1} \int_{\alpha_2=0}^{1-\alpha_1} \|J^S_{f_T}\big(p(\alpha_1,\alpha_2)\big)\|^2 d\alpha_1 d\alpha_2 \\ &= c_j \int_{\alpha_1=0}^{1} \int_{\alpha_2=0}^{1-\alpha_1} \|J_{f_T \circ p}(\alpha_1,\alpha_2))B_j\|^2 d\alpha_1 d\alpha_2 \\ &= c_j K_T \int_{\alpha_1=0}^{1} \int_{\alpha_2=0}^{1-\alpha_1} \|J_r(h \circ w(\alpha_1,\alpha_2))B_j\|^2 d\alpha_1 d\alpha_2 \end{aligned} \tag{4.86}$$

With $B_j = (J_p^T J_p)^{-1} J_p^T$ and c_i the ration of surface elements on the surface and $d\alpha$ and corresponds to twice the surface of the facet. $c_j = 2A_j$ with

$$A_j \equiv \frac{1}{2}\det([J_p, \hat{n}_j]) = \frac{1}{2}\|J_{p[:,1]} \wedge J_{p[:,2]}\| = \frac{1}{2}\|(v_{2j} - v_{1j}) \wedge (v_{3j} - v_{1j})\| \tag{4.87}$$

Because we use a bilinear interpolation of the texture while defining r (see eqn.4.13), the jacobian J_r is piecewise linear. In order to simplify the computation of eqn.4.86, we reformulate r using the following linear interpolation:

$$r(u,v) = \begin{cases} A(u,v) \text{ if } \varepsilon(u) + \varepsilon(v) \leq 1 \\ B(u,v) \text{ else} \end{cases} \tag{4.88}$$

With

$$\begin{aligned} A(u,v) &= (1 - \varepsilon(u) - \varepsilon(v))T_{\lfloor u \rfloor, \lfloor v \rfloor} + \varepsilon(u)T_{\lfloor u \rfloor + 1, \lfloor v \rfloor} + \varepsilon(v)T_{\lfloor u \rfloor, \lfloor v \rfloor + 1} \\ B(u,v) &= +(1 - \varepsilon(u))T_{\lfloor u \rfloor, \lfloor v \rfloor + 1} + (1 - \varepsilon(v))T_{\lfloor u \rfloor + 1, \lfloor v \rfloor} \\ &\quad +(\varepsilon(u) + \varepsilon(v) - 1)T_{\lfloor u \rfloor + 1, \lfloor v \rfloor + 1} \end{aligned} \tag{4.89}$$

The function r is now linear on each triangle that is either the upper-left or the lower-right part of a texel. The Jacobian matrix J_r is now constant on each texel subpart:

$$J_r(u,v) = \begin{cases} J_A(u,v) \text{ if } \varepsilon(u) + \varepsilon(v) \leq 1 \\ J_B(u,v) \text{ else} \end{cases} \tag{4.90}$$

With

$$\begin{aligned} J_A(u,v) &= \left[T_{\lfloor u \rfloor + 1, \lfloor v \rfloor} - T_{\lfloor u \rfloor, \lfloor v \rfloor}, T_{\lfloor u \rfloor, \lfloor v \rfloor + 1} - T_{\lfloor u \rfloor, \lfloor v \rfloor}\right] \\ J_B(u,v) &= \left[T_{\lfloor u \rfloor + 1, \lfloor v \rfloor + 1} - T_{\lfloor u \rfloor + 1, \lfloor v \rfloor}, T_{\lfloor u \rfloor + 1, \lfloor v \rfloor + 1} - T_{\lfloor u \rfloor, \lfloor v \rfloor + 1}\right] \end{aligned} \tag{4.91}$$

We change the integration variables in the integral eqn.4.86 by $(u,v) = h \circ w(\alpha_1, \alpha_2) = K_t(\alpha_1, \alpha_2)$:

$$E^j_{sm}(T) = 2A_j \int_{u=0}^{K_T} \int_{v=0}^{K_T - u} \|J_r(u,v)B_j\|^2 \tag{4.92}$$

$J_r(u,v)$ is piecewise constant on we can decompose the integrals into a sum. We obtain

$$E^j_{sm}(T) = A_j \sum_{u=0}^{K_T} \left[\sum_{v=0}^{K_T-u} \|J_A(u,v)B_j\|^2 + \sum_{v=0}^{K_T-u-1} \|J_B(u,v)B_j\|^2 \right] \tag{4.93}$$

J_A and J_B are linear with respect to the texture image T and thus this smoothing term is quadratic with respect to T. As a consequence we can define derive from this equation a symmetric positive matrix H^i_{sm} such that

$$E^j_{sm}(T) = [T]^T_l H^j_{sm} [T]_l \tag{4.94}$$

We denote H_{sm} the smoothing matrix defined by

$$H_{sm} = \sum_{j=1}^{N_f} H^j_{sm} \tag{4.95}$$

From eqn.4.74 we obtain

$$E_{sm}(T) = \sum_{j=1}^{N_f} E^j_{sm}(T) = [T]^T_l H_{sm} [T]_l \tag{4.96}$$

where $[T]_l$ corresponds to the vectorized version of T that is obtained by concatenating all columns of T.

Note that in [de La Gorce 2008] we defined the smoothness measure $\tilde{E}_{sm}(T)$ to be the sum of squared differences between colors associated to pixels (texels) that are adjacent in the texture and that are associated to the same facet. That is,

$$\tilde{E}_{sm}(T) = \sum_i \sum_{j \in \mathcal{N}_T(i)} \|T_i - T_j\|^2 \tag{4.97}$$

where $\mathcal{N}_T(i)$ represent the neighborhood of the texel i that is associated to the same facet as $T(i)$. The smoothness term $\tilde{E}_{sm}(T)$ is a good approximation of $E_{sm}(T)$ whenever all facet are about the same size and nearly equilateral.

4.8.3.4 Minimization with Augmented Half-Quadric form

Once L θ and ϑ fixed, the RGB value of each pixel of the synthesized image I_{syn} is a linear combination of the RGB values of the four texels in the texture image T that are used in the bilinear interpolation. We assume the texture image T to contain N_t texels and denote $[T]_l$ the one-column ($3N_t$ by 1) matrix obtained by concatenating all RGB values in T. For each pixel $x \in \Omega_d$ in the synthetic image we can write:

$$I_{syn}(x) = A_x [T]_l$$

with A_x a 3 by $3N_t$ matrix with only 12 non-zeros values (3 per texel that contribute to the bilinear interpolation). The data-term $E_d(\theta, \vartheta, L, T)$ rewrites:

$$E_d(\theta, \vartheta, L, T) = \sum_{ij} \rho(\|A_{ij}[T]_l - I_{obs}(i,j)\|) \tag{4.98}$$

If one chooses ρ to be the Huber function or a truncated quadratic function in (4.36), then the minimization of the function $E_{text}(T)$ with $T \in \mathbb{T}$ can be done efficiently by introducing an auxiliary variables w_x for each pixel $x \in \Omega_d$ in the image. We assume that ρ verifies $x \mapsto \rho(\sqrt{(}x))$ is concave, $x \mapsto \rho(x)$ is increasing, ρ is C^1 and $\rho'(0) = 0$ and $\rho''(0) > 0$ is finite. Using the theory of convex conjugacy, it has been established in [Geman 1992] that if we denote β the function defined on $\mathbb{R}^{+*}$ by

$$\beta(\alpha) = \sup_x [\rho(x) - \alpha \|x\|^2] \tag{4.99}$$

The we can rewrite ρ as

$$\rho(x) = \min_{\alpha > 0} [\alpha \|x\|^2 + \beta(\alpha)] \tag{4.100}$$

an the minim of the side term of this equation is reached for

$$\hat{\alpha} = \frac{1}{2|x|^2} \left\langle x \ , \ \left. \frac{d\rho(x)}{dx} \right|_x \right\rangle \tag{4.101}$$

If $\rho(x) = x$ then $\beta(\alpha) = \frac{1}{4\alpha}$. If $\rho(x) = min(x^2, \tau)$ then $\beta(\alpha) = \tau(1 - \alpha)$.

Using the function β and auxiliary variables $W \equiv (w_x)_{x \in \Omega_d}$, the texture estimation rewrites

$$\hat{T} = \arg \min_T (\min_W \bar{E}_{text}(T, W)) \text{ s.t } T \in \mathbb{T} \tag{4.102}$$

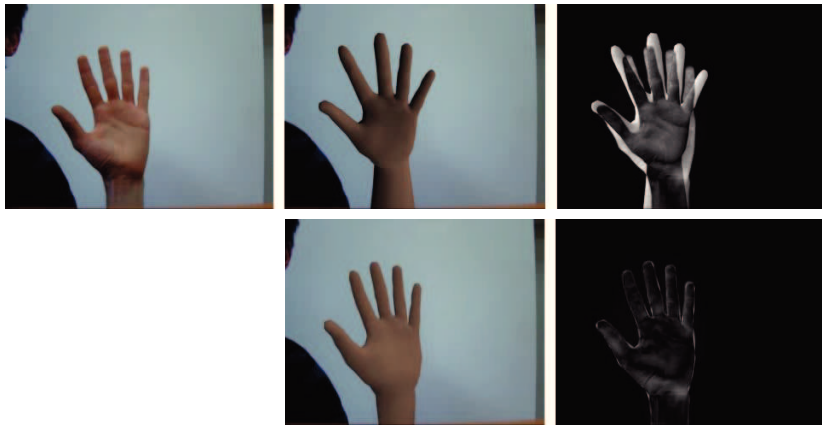
with the augmented function:

$$\bar{E}_{text}(T, W) \equiv [T]_l^T H_{sm} [T]_l + \sum_{x \in \Omega_d} w_x \| A_x [T]_l - I_{obs}(i, j)) \|^2 + \beta(w_x) \tag{4.103}$$

The function $\bar{E}_{text}$ is quadric with respect to T and thus is an *half-quadric* function. The minimization of $\bar{E}_{text}$ is done by minimizing alternatively with respect to T and with respect to the auxiliary variables W. The minimization with respect to T is done by solving a linear system. The minimization with respect to the auxiliary variables W is done using eqn.4.101. One can draw a parallel between this method and the iterative-reweighted-least square (IRLS) method as done in [Nikolova 2007]. The choice of ρ as a truncated quadratic function gave the best results.

To improve robustness we also remove pixels near the occlusion boundary from the summation domain Ω_d when computing the term $E_d(\theta, L, T)$, and we bound the difference between the texture in the first frame and the subsequent texture estimates.

Finally, we note that cast shadows are not modeled directly in our approach, and are therefore modeled as texture, as any other color information. Introducing cast shadows in our continuous optimization framework is not straightforward as it would require computation of related terms in the functional gradient. Shadows would then constitute an additional source of information to guide the registration of the model. Nevertheless, despite lack of cast shadows in our model, our results show adequate, robust tracking.



Descriptio 4.15: Estimation of the pose light and morphologic parameters in the first frame: the first line show the first observed frame, the synthetic image corresponding to rough initialization given by the user and the corresponding residual. The second line show the result obtained after convergence

While fitting the hand model onto the first frame, very little is known about color and texture of the hand and the background. A simple prior is to assume that the texture can be fairly approximated by a single color. This is detailed din the section on initialization

4.9 Experimental Results

4.9.1 Initialization

The hand pose tracker requires a reliable initial guess in the first frame. Estimating the hand pose in a single frame without a strong prior of the hand pose is challenging. In our case the morphological parameters are also estimated in the first frame.

One could attempt to use a discriminative method to obtain a rough initialization (e.g., [Rosales 2001]). However, since this is outside the scope of our approach at present, we assume prior information about the initial hand pose.

In particular, the hand is assumed to be parallel to the image plane at initialization [Figure (4.15)] and linear constraints were defined on the relative lengths of the parts within each finger. Furthermore, since we do not yet have a texture estimate in the first frame, we suppose the model hand color albedo to be constant across the surface. With this assumption the appearance of the model hand is largely due to shading. The three RGB values of the hand color, along with the hand pose, the morphological parameters and the illuminant are estimated simultaneously using

the quasi-Newton method (see the results in second row [Figure (4.15)][2]). We suppose that the background image or its histogram are provided by the user. Once the morphological parameters are estimated in the first frame, they remain fixed for the remainder of the image sequence.

4.9.2 Tracking without pose prior

We test our tracking algorithm on various image sequences. In the first sequence [Figure (4.16)] each finger bends in sequential order, eventually occluding parts of other fingers and the palm [3]. The cluttered background image is static, and was obtained from a frame where the hand was not visible. The resolution of each frame is 640 by 480 pixels. We can notice that the edges of the synthetic hand image edges matches the edges in the observed image, despite we did not use any explicit edge-related term in our objective functional. Misalignment of the edges in the synthetic and observed images creates a large residual in the area between these edges and produces occlusion forces on the hand edges pointing in the direction that reduces the gap between these edges.

To illustrate the improvements provided by self-occlusion forces, we simply removed the occlusion-forces while computing the functional gradient. The effect is dramatic as the resulting algorithm is unable track any displacement.

The comparisons with the conventional sum-on-surface approach outlined in Section 4.4.4 is more relevant. This alternative approach involves summing errors on 3D points that remain fixed on the triangulated surface throughout the iterations. Those points are uniformly distributed on the hand surface and their binary visibility is computed at each step of the quasi-Newton minimization process. To account for the change of summation domain (from image to the surface), the error associated to each point is weighted by the inverse of the ratio of surfaces between the 3D triangular face and its projection. The errors associated with the background are also taken into account, in order to remain consistent with the initial cost function. Furthermore, this will prohibit the model from shrinking in the image domain by increasing its distance to the camera. We kept occlusion forces between the hand and the background to account for variations of the background visibility while computing the functional gradient. The functional computed in both approaches would ideally be equal and their difference is bounded by the error induced by the discretization of integrals. For both methods we limited the number of iterations to 100.

This alternative approach produces overall good results [Figure (4.16) rows 5-7] but fails to recover the precise location of the finger extremities when they bend and occlude the palm. This is most significant in the third column where a large portion of the little finger is missing. Our approach (rows 2-5) compares favorably, yielding accurate tracking through the entire sequence. The alternative approach

[2] the video Video1.avi encoded with the xvid codec can be downloaded here:
http://www.mas.ecp.fr/vision/Personnel/martin/PAMI09/

[3] see Video2.avi and Video3.avi

fails because the hand/background silhouette is not particularly informative about the position of fingertips when fingers are bending. The residual error is mostly localized near the outside extremity of the synthesized finger, and self-occlusion forces are necessary to pull the finger toward this region.

We further validate our approach by choosing the erroneous estimated hand pose in [Figure (4.16) column 3 rows 5-7] as an initialization of the new tracking method that uses the occlusion forces [Figure (4.17)][4]. The initial hand pose and the corresponding residual are shown in the first column. The pose obtained after 25 and 50 iterations are in the second and the third column. After 30 iterations the hand pose is properly recovered. This illustrates the eventual inability of the alternative approach to converge to a local minima of the cost function as a consequence of its poor treatment of occlusions.

The second and third sequences [Figure (4.18)] and [Figure (4.19)] were provided by the authors of the tree-based Monte Carlo method that constitutes the state-of-the-art in monocular hand tracking [Stenger 2004a]. Both sequences have a resolution of 320 by 240 pixels. In the second sequence the hand is closing and opening while rotating. In the third sequence the index finger is pointing and the hand rotates in a rigid manner. Both sequences present important self-occlusion and large inter-frame displacements. The background in the third sequence is not static and the associated cost has been expressed using a histogram (see Section 4.4). For computational reasons, the results presented in [Stenger 2004a] were obtained with a reduction in the dimension of the hand pose-space, adapted to each sequence individually (8D movements for second sequence - 2 for articulation and 6 for global motion - and 6D rigid movement for third sequence).

We tested our algorithm both with [5] and without [6] such reductions (respectively rows 2 and 3). To do so, linear inequalities were defined between pairs or triplet of angles. Inequalities were preferred to equalities because this limits the range of possible poses while locally keeping enough freedom of pose variation to make fine registration possible. We did not update the texture for those sequences after the first frame. As one would expect the results are better when the pose space is reduced. Nevertheless, the results obtained with the full pose space are still satisfying. The loss of accuracy during tracking in the second and third sequences, in comparison with the first sequence, can be attributed to two key factors. The interframe movement in the second and third sequences is large. This challenges our local search approach as a starting point for the minimization (i.e., it has to be predicted from previous estimated poses). Also, in the second and third sequences the fingers are often touching each other. This challenges our method because collision avoidance has not been incorporated to prohibit parts from penetrating one another in our optimization framework.

The last sequence [Figure (4.20)] illustrates robustness to self-occlusion using

[4] see Video4.avi
[5] See Video5.avi and Video7avi
[6] See Video6.avi and Video8.avi

two hands [7]. The left hand occludes progressively the right hand. Each column corresponds successively to the observed images, the best synthesized images, the difference images, and the images obtained with shading only (without texture). One can notice that, due to the shading and texture, the synthetic image look very alike the observed image. In the second row, the ring finger and the little finger of the right hand are still properly registered despite the large occlusion. Despite the translational movement of the hands, we kept all 28 DOFs on each hand while performing tracking on this sequence.

4.9.3 Tracking with pose prior

In order to validate the qualitative improvement while using the prior term, we test our tracker on a sequence of signed letter from the American manual alphabet where "CVPR is spelled several times. A cyberGlove could be used to acquire the set of training and poses. However number of DOF that can be tracked is smaller than the number of DOF in our model, and the error on joint angles is often large. Instead, we record a video of an hand spelling "CVPR simultaneously from two different view point (front and side) using calibrated cameras. We perform multi-view tracking by simply minimizing the sum of two data terms that respectively correspond to the residual in the front view and the side view. This illustrates the fact that our approach can easily be extended to the multi-view settings. The combination of the two views reduces depth ambiguities and significantly improves the quality of the inferred pose. Toward improving further the accuracy of the tracking, we help the tracker by manually labeling the position of each finger tip in both sequences, and add a third term in the objective function that measure the distance between each finger tips projection and its manually labeled position in the image. We track four occurrence of the "CVPR with this multi view setting and obtained a total of about 600 training poses. The corresponding set of joint angles parameters vectors $\alpha_1, \ldots, \alpha_{600}$ is used to learn the kernel PCA plane using a radial basis function kernel of the form $exp(-\|x-y\|^2/r^2)$. We center and normalize each angle parameter by its range in the extracted tracks, thus obtaining scaled values ranging from -1 to 1. The bandwidth r of the kernel is chosen to be 1.5 times the mean of the Euclidean distance to the closest scaled point in the training set, after having removed points that temporally closer than some duration τ :

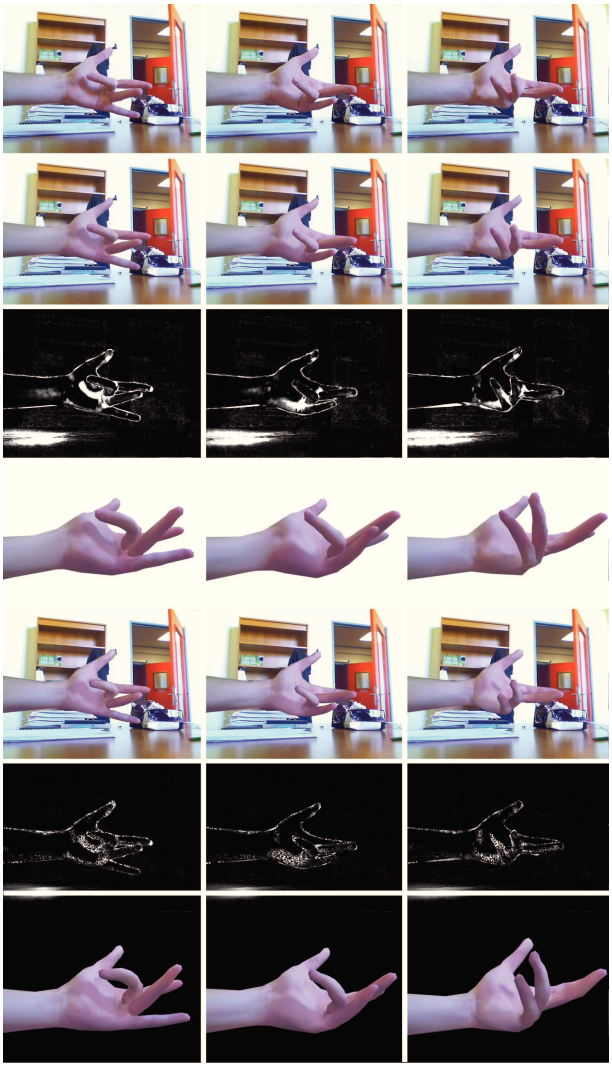
$$r^2 = \frac{1}{N} \sum_i min_{j,\|i-j\|>\tau} \|x_i - x_j\| \tag{4.104}$$

This avoid comparisons with poses from the same letter occurrence, which would lead to very small bandwidth and thus lead to over fit the training set. We chose to keep 90% of the variance in the kernel PCA plane and took $\lambda_\perp = \lambda_l$ (see eqn. A.49).

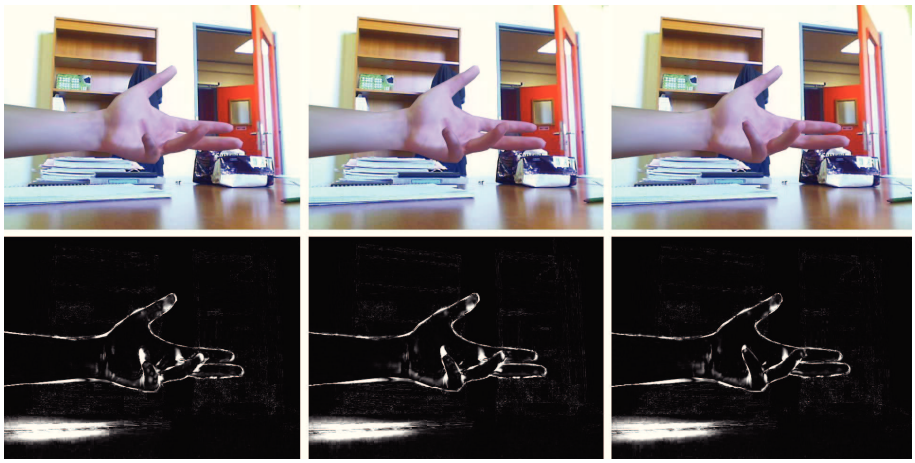
We test the performance of the tracking on three new occurrences of the world "CVPR using the front images only, thus performing monocular tracking. We

[7] see Video9.avi

estimated the texture of the hand in the first frame and did not update the texture of the model. We use the side view to assess the quality of the inferred pose. Without the prior term, the methods loose track of the actual pose of the hand after the first letter C [Figure (4.21)]. This is due to the fact that the inter-frame motion is quite important during the transition between the letters. With the KPCA prior term [Figure (4.22)] the methods is quantitatively able to track the hand pose during the three occurrence of the word CVPR. We tested several weighting factor β for the prior term and found the best results with $\beta = 5.10^4$. Note that the inferred depth of the hand is not very precise (this is visible in the side view images), but this is inevitable in the monocular setting. This validation clearly illustrates the gain in robustness imputable to the prior term. Note that after the first letter the method without the prior term yielded to pose where finger self-collide. No collision avoidance is explicitly incorporated in the optimization framework to prohibit parts from penetrating one another. However, since the set of trained configuration do not exhibit self-collision, the inferred position are also mainly free of collisions.

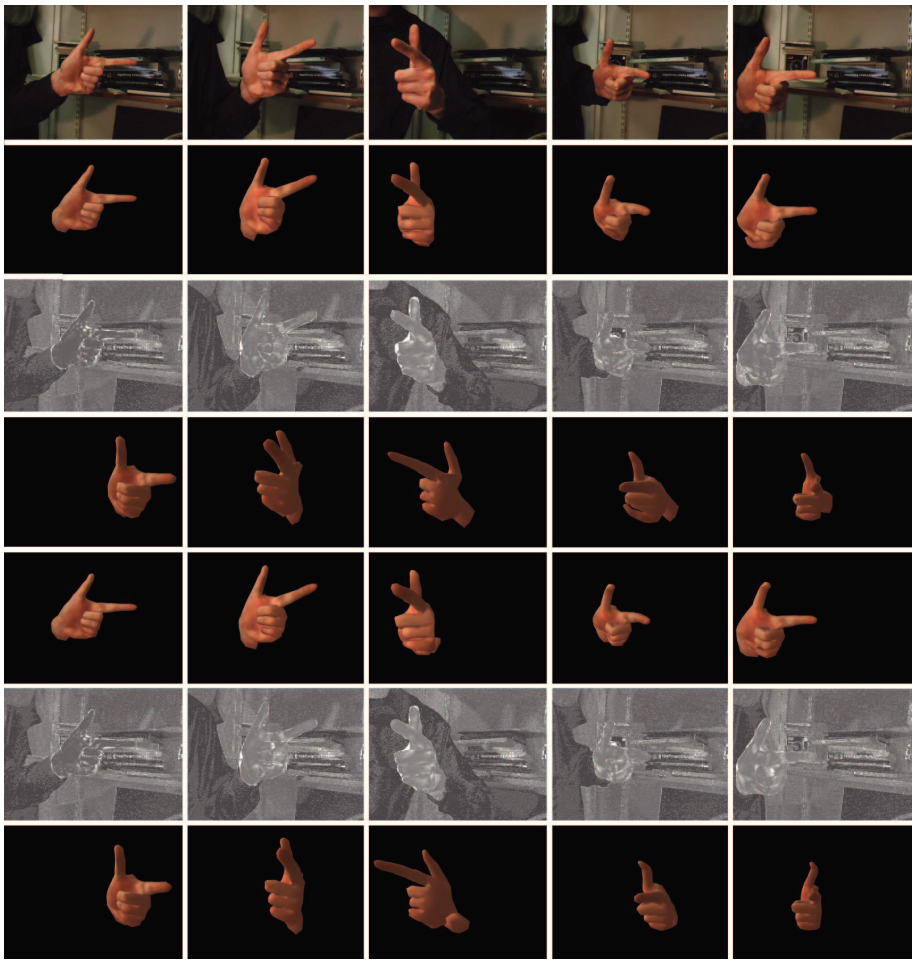


Descriptio 4.16: First sequence illustrating improvement due to self-occlusion forces. The three columns contain results from three separate frames in the sequence. Each row shows (from top to bottom) (1) the observed image, (2) the final synthetic image, (3) the final residual image, (4) a synthetic side view at 45°, (5) the final synthetic image with residual summed on surface, (6) the residual for visible points on the surface, and (7) the synthetic side view

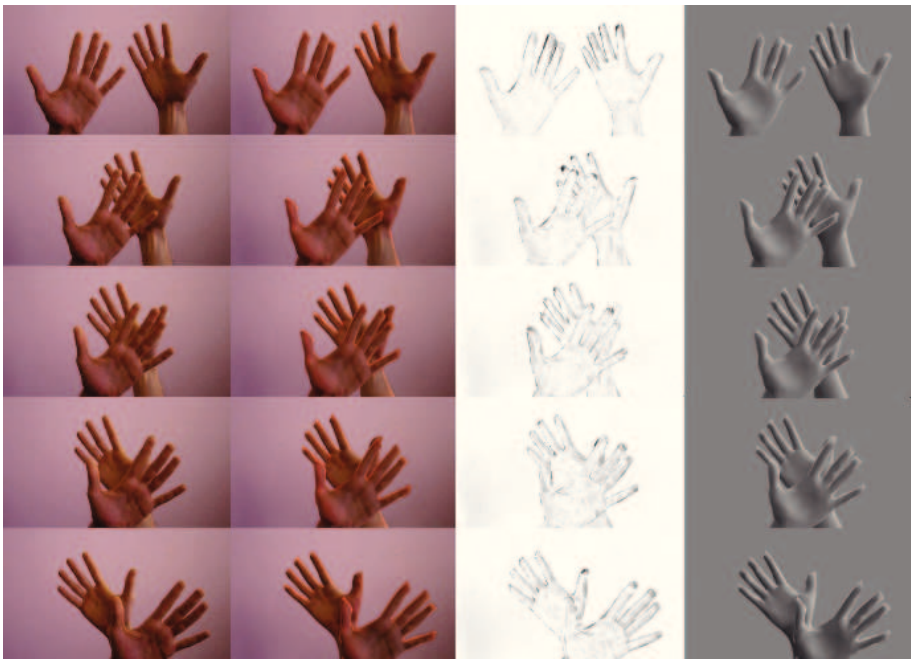


Descriptio 4.17: Recovery of the failure mode of the sum-on-surface method (section 4.9.2) by our method with occlusion forces.

Descriptio 4.18: Second sequence. Each row shows (from top to bottom) (1) the observed image, (2) the final synthetic image with limited pose space, (3) the final residual image, (4) the synthetic side view with an angle of 45°, (5) the final synthetic image with full pose space, (6) the residual image and (7) the synthetic side view

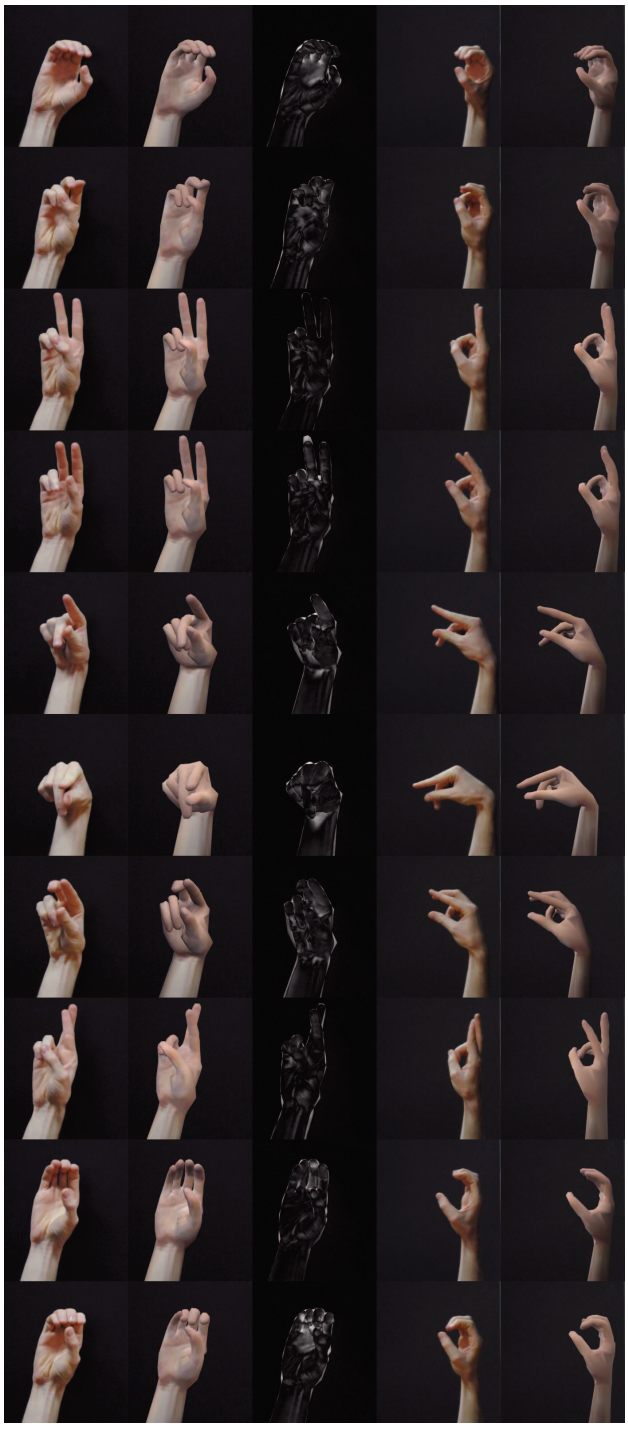


Descriptio 4.19: Third sequence. Each row shows (from top to bottom) (1) the observed image, (2) the final synthetic image with limited pose space, (3) the final residual image, (4) the synthetic side view with an angle of 45°, (5) the final synthetic image with full pose space and (6) the residual image, the synthetic side view



Descriptio 4.20: Sequence with two hands illustrating robustness to large self-occlusions

Descriptio 4.21: “CV tracking without any hand pose prior. Each column correspond successively to the observe front image, the synthetic front image of with the estimated hand pose, the residual image , the observed image from side view (unused will estimating the hand pose), and the synthetic image front the side view. Lines 1 and 3 respectively to letters C and V and the other lines correspond to transition between letters. The track is lost during the transition between C and V.



Descriptio 4.22: “CVPR tracking with the KPCA prior, Lines 1,3,6,8 and 10 corresponds respectively to letters C,V,P,R and C. The other lines correspond to transition between these letters.

Caput 5

Conclusion & Perspectives

5.1 Contributions

In this thesis we studied the problem of hand pose estimation in a sequence of monocular images. We have introduced two model-based approaches where the hand pose is estimated through the minimization of an objective function.

The first method is based on a simple hand surface model made of polyhedra and ellipsoids. The objective function aims to separate the characteristics of the skin with the ones of the cluttered background. This function depends only on the hand silhouette once projected to the image plane and thus does not allows to discriminate between pose that yield to the same hand silhouette. In order to address this limitation we presented a second approach that exploits the shading and textural information of the observed hand. This is done using a detailed triangulated surface model whose appearance encompass shading and texture. Because the generative model is realistic enough, we have been able to define the objective function directly as the sum of errors within the observed image at each pixel.

For both approaches we have devised a differentiable objective function and we have derived the exact expression of its gradient. For the silhouette method we have devised numerical method to compute exactly (up to machine precision) the matching cost and its gradient. For the method that combine texture and shading, The exact computation of the matching cost has not been devised. We devised an approximate matching cost that can be numerically evaluated exactly and yet is be continuous and differentiable function of the hand pose parameters. The gradient of the approximate matching cost has been derived an can be computed exactly. This contrasts with most hand model-based approaches in the literature where the matching cost is not computed exactly or is a discontinuous function of the hand pose parameters

Based on the gradient of the objective function we have been able to use efficient local optimization methods to estimate the hand pose. For the method based on ellipsoids, we introduced a variable metric gradient descent and a efficient trust-region method for model fitting. We also have devised an approximation of the Hessian that further increase the convergence rate. For the model based on a triangulated surface, we introduced a block-wise BFGS formula to exploit independences between hand parts and speed up the convergence rate over the standard BFGS formula. The exact derivation of the gradient in the case of the triangulated surface has yield to *occlusions forces* that allow to deal with the change of visibility around occlusion, which has not been done in previous methods.

Towards addressing limitations of local optimization methods we have considered a multiple-hypotheses testing algorithm. Introducing multiple hypotheses in the process eliminates the risk of convergence to local minima that is often the case of gradient descent optimization techniques. Furthermore we have introduced a prior on the hand pose base on the Kernel principal-component-analysis toward improving the robustness of the tracking whenever the context allows to have strong assumption on the possible hand poses.

5.2 Perspectives

Promising qualitative results, as well as comparisons with the state of the art methods demonstrated the potentials of our contributions. Nevertheless, our method does fails in long sequence with large self-occlusions and is not real-time. Several improvements could be considered:

- **Self-Collisions**. It is desirable to avoid self-collision towards getting realistic hand poses.Furthermore, self-occlusion may introduce a change of visibility at surface intersections. The facet that do intersect might be partially visible and change of their visibility should also be treated when computing the exact gradient of the matching cost. This would further complicate the expression of the gradient and is not desirable. On the other hand, self-collision avoidance not straightforward to handle in the pose estimation procedure. One need first to define the set valid pose in regard to self-collision, which we denote $\Theta_c \subset \Theta$. Unfortunately, and that is not a convex set. Most efficient optimization routines requires nonlinear(constraints to be define through a set of continuous functions $f_i : \theta \mapsto \mathbb{R}$ such that $\Theta_c \equiv \{\theta \in \Theta | f_i(\theta) \leq 0 \ \forall i\}$. Defining such function f_i to model the self-collision is very challenging. Most collision-distance measure found in the literature are either defined for pairs of convex objects [Cameron 1986, Ong 2000] or not differentiable around the set $\{\theta \in \Theta \mid f_i(\theta) = 0\}$ [Fisher 2001].

- **Cast shadows**. Regions that correspond to cast shadow often produce a large residual. By modeling cast-shadows in the generative model, we could reduce this residual if the hand pose is well estimated. Meanwhile we would take advantage of the information conveyed by the cast shadows to improve the pose estimate. Furthermore, if the hand casts a shadow on a simple flat surface in the background , then it could be exploited to improve the depth estimation of the hand pose. However, adding cast-shadow to our generative model is not straightforward. The self cast-shadows continuously displace when the hand moves and this as to be taken into account when deriving the gradient of the objective function. Similarly to the *occlusion forces* deriving from the variation of visibility, new forces would result from the variation of the boundaries of the shadows. In order to compute these forces, ones would need to describe the shadows boundaries analytically as polylines. Most

existing cast-shadow algorithms uses some discretization on the pixel grid and therefore do not provide such an exact description of the shadow boundaries. The model-space method to compute visibility discussed in section 4.7.1 is a promising approach to include cast shadows in the model.

- **Automatic initialization**. The proposed methods are not fully automatic. For both methods we either need to provide a full image of the static background or to select a large region in the background in order to learn some color backgrounds statistics. The hand pose color distribution is obtained in a supervised manner for the method based on ellipses. The mean hand color is provided by the user for the method based on the triangulated surface. For both methods, a rough estimate of the pose is requested from the user in the first frame. Methods that localize a 2D flat hand with extended finger in the image [Thayananthan 2003b, Wang 2008], can be considered to automatically obtain a rough initial pose estimate. These estimated location could facilitate the estimation of the color statistics of the hand and the background.

- **Second order derivatives**. We devised second and approximation of the Hessian of the objective function for the method based on ellipsoids. Similarly we could derive an approximation of the Hessian for the method based on the triangulated surface. Using the trust-region method where the approximate Hessian is not required to be positive, it is likely that we would improve the convergence rate of the method.

- **Globalizing the search**. We advocate the use of a smart particle filter in order to cope with the presence of several local minima in the objective function. Even if it allows to improve robustness, it might be insufficient to recover the right pose in case of important ambiguity. Methods based on a very limited set of 2D poses such as [Thayananthan 2003b, Wang 2008] could allow to perform re-initialization of the tracker if the user is informed in real-time that the tracking has failed. In order to robustly pose estimation one would either need to use an efficient discriminative approach or improve the search using global optimizations methods. Several direction could be envisioned. One possibility is to decompose the energy into a factor graph and use message passing methods. An important difficulty while re-writing the objective function as a sum of of factor come from the fact that the occlusion is difficult to decompose into a graph. The method proposed in [Wang 2009] is a first step toward modeling occlusion on discrete graphs. An additional challenge is inherited from the dimensionality of the problem, since each phalanx is represented by a node in the graph, the associated variable has 6 dimension (rotation and translation). Message passing using 6D variable remain a difficult problem. Method based on Monte-Carlo integration [Sudderth 2004] do not provide accurate results in such high dimensional spaces. An other possibility is to perform some Branch and bound optimization, where the pose space is iteratively sectioned and large parts are discarded using some lower

bound of the actual objective function. The method proposed in , thought not being rigorously stated as a branch and bound approach, is very close to the idea. Unfortunately, like the discriminative methods, this method requires to compute a huge set of hand images of the hand in different poses in order to obtain precise hand pose estimates.

- **Gesture analysis**.Modeling and understanding hand gestures as a succession of articulation parameters through autoregressive models could be a natural extension of the proposed framework. Such an extension could lead to sign-language recognition that is one of the most challenging tasks of gesture analysis. The method based on the kernel-PCA prior allows to recognize letter of the American-manual-alphabet but does not allows to recognize a succession of hand pose as a dynamic gesture like those of the America-sign-language. Note however that the information conveyed by the hand hand might not be sufficient because lips also convey crucial information while signing.

Other perspectives are the extension of some results to other applications. It could be of interest, for medical applications for example, to extend the exact integration of interpolated image into a polygon to the three dimensional case. This would require to clip a polyhedra against the voxels of the 3D image and then the integration of polynomials into each sub-voxel polyhedra using results from [Li 1993, Sheynin 2001, Shu 2001, Grant 1985] for example. Then first order and second derivatives of this integral could also be computed exactly. This would allow to use efficient trust region schemes with active polyhedra [Slabaugh 2005].

Additamentum A

Appendix

A.1 forces on the silhouette : alternative proof

In order to demonstrate the equation eqn.3.162 in a more rigorous but less intuitive manner than previously done, we first derive some equalities using integration by parts along the line segment. Given a continuous function g we have:

$$[g(t)t^n]_0^1 = \int_0^1 \frac{\partial g}{\partial t}(t)t^n dt + \int_0^1 g(t)nt^{n-1}dt \tag{A.1}$$

if we take $g(t) = h(q(t))$ with h a scalar function of two variable and $q(t) = (q_x(t), q_y(t))$ with $q_x(t) = tx_k + (1-t)x_{k+1}$ and $q_y(t) = ty_k + (1-t)y_{k+1}$ we get :

$$\begin{aligned} h(q_k) - 0^n h(q_{k+1}) &= \int_0^1 \left(\frac{\partial h}{\partial x}(q(t))\frac{\partial q_x}{\partial t}(t) + \frac{\partial h}{\partial y}(q(t))\frac{\partial q_y}{\partial t}(t)\right) t^n dt \quad \text{(A.2)} \\ &\quad + \int_0^1 h(q(t))nt^{n-1}dt \\ &= \left[(x_k - x_{k+1})\int_0^1 \frac{\partial h}{\partial x}(q(t))t^n dt\right] \quad \text{(A.3)} \\ &\quad + \left[(y_k - y_{k+1})\int_0^1 \frac{\partial h}{\partial y}(q(t))t^n dt\right] \\ &\quad + \int_0^1 h(q(t))nt^{n-1}dt \quad \text{(A.4)} \end{aligned}$$

From 3.144 and 3.117 we can rewrite the matching cost as :

$$L(\theta) = \sum_k C_k = \sum_k (y_{k+1} - y_k)\int_0^1 F_x((1-t)q_k + tq_{k+1})dt \tag{A.5}$$

By replacing t by $(1-t)$ in equation 3.117 we get :

$$C_k = (y_{k+1} - y_k)\int_0^1 F_x(q(t))dt \tag{A.6}$$

We differentiate C_k with respect to x_k and y_k :

$$\frac{\partial C_k}{\partial x_k} = (y_{k+1} - y_k) \int_0^1 \frac{\partial F_x}{\partial x}(q(t)) \frac{\partial q_x}{\partial x_k} dt \tag{A.7}$$

$$= (y_{k+1} - y_k) \int_0^1 f_c(tq_k + (1-t)q_{k+1}) t dt \tag{A.8}$$

$$\frac{\partial C_k}{\partial y_k} = -\int_0^1 F_x(q(t))dt + (y_{k+1} - y_k) \int_0^1 \frac{\partial F_y}{\partial x}(q(t)) \frac{\partial q_x}{\partial y_k} dt \tag{A.9}$$

$$= -\int_0^1 F_x(q(t))dt + (y_{k+1} - y_k) \int_0^1 \frac{\partial F_y}{\partial y}(q(t)) t dt \tag{A.10}$$

Using eqn.A.4 with $h = F_x$ and $n = 1$ we get :

$$\frac{\partial C_k}{\partial y_k} = -F_x(x_k, y_k) - (x_{k+1} - x_k) \int_0^1 f_c(tq_k + (1-t)q_{k+1}) t dt \tag{A.11}$$

Finally we can differential L with respect to q_k. The displacement of a vertex q_k has an influence only on the cost associated to the two neighboring segments i.e we have $\frac{\partial C_l}{\partial q_k} = 0 \forall l \notin \{k-1, k\}$. As a consequence we get:

$$\frac{\partial L}{\partial q_k} = \left[\frac{\partial C_{k-1}}{\partial x_k} + \frac{\partial C_k}{\partial x_k}, \frac{\partial C_{k-1}}{\partial y_k} + \frac{\partial C_k}{\partial y_k} \right]^T \tag{A.12}$$

$$= J(q_k - q_{k-1}) \int_0^1 f_c(tq_k + (1-t)q_{k-1}) t dt \tag{A.13}$$

$$+ J(q_{k+1} - q_k) \int_0^1 f_c((tq_k + (1-t)q_{k+1}) t dt \tag{A.14}$$

A.2 Algorithms

Algorithm 3: Forward differentiation

Data: $\theta, (\frac{\partial E}{\partial K_i})_{i=1,\dots,N}$
Result: $\frac{\partial E}{\partial \theta}$
// Compute all derivatives $\frac{\partial K_n}{\partial \theta_j}$ *with* $(n,j) \in \{1,\dots,N\}^2$
$K_1 \leftarrow F_1(\theta_1)$;
for $l = 1$ **to** d_1^* **do**
  $\frac{\partial K_1}{\partial \theta_{1,l}} \leftarrow \left.\frac{\partial F_1}{\partial \theta_{1,l}}\right|_{\theta_1}$;
for $j = 2$ **to** N **do**
  for $l = 1$ **to** d_1^* **do**
    $\frac{\partial K_1}{\partial \theta_{j,l}} \leftarrow 0_{4\times 4}$;
for $j \leftarrow 2$ **to** N **do**
  $i \leftarrow p(j)$;
  $K_j \leftarrow K_i F_j(\theta_j)$;
  for $l = 1$ **to** d_1^* **do**
    $\frac{\partial K_j}{\partial \theta_{j,l}} \leftarrow K_i \left.\frac{\partial F_j}{\partial \theta_{j,l}}\right|_{\theta_j}$;
  for $k \in V \backslash \{(j)\}$ **do**
    for $l = 1$ **to** d_1^* **do**
      $\frac{\partial K_j}{\partial \theta_{k,l}} \leftarrow \frac{\partial K_i}{\partial \theta_k} F_j(\theta_{j,l})$;
// Combine $\frac{\partial K_n}{\partial \theta_j}$ *with* $\frac{\partial E}{\partial K_n}$ *to obtain the final results*
for $j = 1$ **to** N **do**
  for $l = 1$ **to** d_1^* **do**
    $\frac{\partial E}{\partial \theta_{j,l}} \leftarrow \sum_{n=1}^{N} \frac{\partial E}{\partial K_n} \frac{\partial K_n}{\partial \theta_{j,l}}$;

Algorithm 4: Backward differentiation

Data: $\theta, (\frac{\partial E}{\partial K_i})_{i=1,\dots,N}$
Result: $\frac{\partial E}{\partial \theta}$
// Recompute matrices $K_1, \dots, K_N$ if not stored in memory
$K_1 \leftarrow F_1(\theta_1)$;
for $j \leftarrow 2$ **to** N **do**
  $i \leftarrow p(j)$;
  $K_j \leftarrow K_i F_j(\theta_j)$;
// Initialize matrices $\bar{K}_1, \dots, \bar{K}_N$
for $i \leftarrow 1$ **to** N **do**
  $\bar{K}_i \leftarrow \frac{\partial E}{\partial K_i}$;
// Back-propagate derivatives
for $j \leftarrow N$ **to** 2 **do**
  $i \leftarrow p(j)$;
  $\bar{K}_i \leftarrow \bar{K}_i + \bar{K}_j F_i^T(\theta_j)$;
  for $l = 1$ **to** d_1^* **do**
    $\frac{\partial E}{\partial \theta_j} \leftarrow K_i^T \bar{K}_j \left.\frac{\partial F_j}{\partial \theta_{j,l}}\right|_{\theta_{j,l}}$;
for $l = 1$ **to** d_1^* **do**
  $\frac{\partial E}{\partial \theta_{1,l}} \leftarrow \bar{K}_1 \left.\frac{\partial F_1}{\partial \theta_1}\right|_{\theta_{1,l}}$;

Algorithm 5: clipSegmentOnPixelGrid

Data: Two segment extremities $a \equiv (x_a, y_a)$ and $b \equiv (x_b, y_b)$

Result: a list of Point $p_1, \ldots, p_N$ and their normalized curvilinear coordinate $t_1, \ldots t_N \in [0, 1]$ on the segment such that $p_1 = (x_a, y_a)$, $p_N = (x_b, y_b)$, $\overline{ab} = \bigcup_{i=1}^{N} \overline{p_i, p_{i+1}}$ and for each i the two segments extremities p_i and p_{i+1} lies inside the same pixel

// get the first intersection with a vertical line

if $x_b > x_a$ **then**

  $\delta_v^x \leftarrow 1$; $x_v \leftarrow \lfloor x_a \rfloor + 1$;

else

  $\delta_v^x \leftarrow -1$; $x_v \leftarrow \lceil x_a \rceil - 1$;

if $\delta_x x_v < \delta_x x_b$ **then**

  $N_v \leftarrow \lceil \delta_x (x_b - x_v) \rceil$; $\delta_v^T \leftarrow \delta_v^x / (x_b - x_a)$; $\delta_v^y \leftarrow (y_b - y_a)\delta_v^T$;

  $y_v \leftarrow \delta_v^x \delta_v^y (x_v - x_a) + y_a$; $t_v \leftarrow \delta_v^x \delta_v^T (x_v - x_a)$;

else

  $N_v = 0$;$t_v = 1$;

// get the first intersection with an horizontal line

if $y_b > y_a$ **then**

  $\delta_h^y \leftarrow 1$; $y_h \leftarrow \lfloor y_a \rfloor + 1$;

else

  $\delta_h^y \leftarrow -1$; $y_h \leftarrow \lceil y_a \rceil - 1$;

if $\delta_y y_h < \delta_y y_b$ **then**

  $N_h \leftarrow \lceil \delta_y (y_b - y_h) \rceil$; $\delta_h^T \leftarrow \delta_h^y / (y_b - y_a)$; $\delta_h^x \leftarrow (x_b - x_a)\delta_h^T$;

  $x_h \leftarrow \delta_h^x \delta_h^y (y_h - y_a) + x_a$; $t_h \leftarrow \delta_v^y \delta_v^T (y_v - y_a)$;

else

  $N_h \leftarrow 0$;$t_h = 1$;

$p_1 = (x_a, y_a)$; $t_1 = 0$;

// Loop until it reaches the extremity b

$N \leftarrow N_h + N_v + 2$;

for $k \leftarrow 2$ **to** N **do**

  if $t_v < t_h$ **then**

    // the next intersection is with a vertical line

    $p_k \leftarrow (x_v, y_v)$; $t_k \leftarrow t_v$;

    // get the next intersection with an vertical line

    $(x_v, y_v, t_v) \leftarrow (x_v + \delta_v^x, y_v + \delta_v^y, t_v + \delta_v^T)$;

  else

    // The next intersection is with an horizontal line

    $p_k \leftarrow (x_h, y_h)$; $t_k \leftarrow t_h$;

    // get the next intersection with an horizontal line

    $(x_h, y_h, t_h) \leftarrow (x_h + \delta_h^x, y_h + \delta_h^y, t_h + \delta_h^T)$;

$p_N \leftarrow (x_b, y_b)$; $t_N = 1$;

Algorithm 6: clipMonotonicEllipseArcOnPixelGrid

Data: a 3 by 3matrix Q associate to the ellipse and two arc extremities (x_a, y_a) and (x_b, y_b)

Result: a list of Point $P_1, \ldots, P_N$ on the ellipse arc such that $p_1 = (x_a, y_a)$, $p_N = (x_b, y_b)$ and for each i the ellipse arc between p_i and p_{i+1} lies inside the same pixel

```
// Check if the two extremities belong to the ellipse
if [x_a, y_a, 1]Q[x_a, y_a, 1]^T ≠ 0 or [x_b, y_b, 1]Q[x_b, y_b, 1]^T ≠ 0 then
    error
// get the first intersection with a vertical line
if x_b > x_a then
    δ_x ← 1; x_v ← ⌊x_a⌋ + 1;
else
    δ_x ← −1; x_v ← ⌈x_a⌉ − 1;
if δ_x x_v < δ_x x_b then
    y_v ← intersectV(Q, x_v, −δ_x); N_v ← ⌈δ_x(x_b − x_v)⌉;
else
    N_v = 0;
// get the first intersection with an horizontal line
if y_b > y_a then
    δ_y ← 1; y_h ← ⌊y_a⌋ + 1;
else
    δ_y ← −1; y_h ← ⌈y_a⌉ − 1;
if δ_y y_h < δ_y y_b then
    x_h ← intersectH(Q, y_h, δ_y); N_h ← ⌈δ_y(y_b − y_h)⌉;
else
    x_h ← δ_x + x_b; N_h ← 0;
p_1 = (x_a, y_a);
// Loop until it reaches the extremity b
N ← N_h + N_v + 2;
for k ← 2 to N do
    if δ_x x_v < δ_x x_h then
        // the next intersection is with a vertical line
        p_k ← (x_v, y_v);
        // get the next intersection with an vertical line
        x_v ← x_v + δ_x;  y_v ← intersectV(Q, x_v, −δ_x);
    else
        // The next intersection is with an horizontal line
        p_k ← (x_h, y_h);
        // get the next intersection with an horizontal line
        y_h ← y_h + δ_y; x_h ← intersectH(Q, y_h, δ_y);
p_N ← (x_b, y_b);
```

Algorithm 7: intersectV(Q,x,s_v)

Data: a 3 by 3matrix Q associate to the ellipse, a vertical line coordinate x and $s_v \in \{-1, 1\}$ that specify if we are in the upper(1) of lower (-1)part of the ellipse

Result: The coordinate of the intersection (x, y) of the line and the ellipse upper or lower part

$\alpha \leftarrow Q_{22}$;
$\beta \leftarrow 2Q_{12}x + 2Q_{23}$;
$\gamma \leftarrow Q_{11}x^2 + 2Q_{13}x + Q_{33}$;
$\Delta = \beta^2 - 4\alpha\gamma$;
$y \leftarrow (-\beta + s_v\sqrt{\Delta})/(2\alpha)$;

Algorithm 8: intersectH(Q,y,s_h)

Data: a 3 by 3matrix Q associate to the ellipse,an horizontal line coordinate y and $s_h \in \{-1, 1\}$ that specify if we are in the right(1) of left (-1) part of the ellipse

Result: The coordinate of the intersection (x, y)

$\alpha \leftarrow Q_{11}$;
$\beta \leftarrow 2Q_{12}y + 2Q_{13}$;
$\gamma \leftarrow Q_{22}y^2 + 2Q_{23}y + Q_{33}$;
$\Delta = \beta^2 - 4\alpha\gamma$;
$x \leftarrow (-\beta + s_h\sqrt{\Delta})/(2\alpha)$;

Algorithm 9: integrateSegmentDerivativeNearestNeighbor

Data: The polygon vertices $(q_i)_{i=1}^N$ and a discretized function $f(x, y)$
Result: The derivatives $\dot{q}_i \equiv \frac{\partial I}{\partial q_i}$ of the integral I of f_c inside the polygon with f_c the shifted nearest neighbor interpolation of f

for $i \leftarrow 1$ **to** N **do**
  $\dot{q}_i \leftarrow [0, 0]^T$;
// loop over edges of the polygon
for $i \leftarrow 1$ **to** N **do**
  $j = mod(i, N) + 1$;
  // clip the segment $\overline{q_i, q_j}$ *on the pixel grid*
  $(p_k, t_k)_{k=1}^K \leftarrow$ intersectSegment(q_i, q_j);
  // loop over subpixel fragments
  for $k \leftarrow 1$ **to** $K - 1$ **do**
    // compute the middle point of the subpixel fragment
    $x_c = (x_k + x_{k+1})/2$;
    $y_c = (y_k + y_{k+1})/2$;
    $t_c = (t_k + t_{k+1})/2$;
    $\Delta_t = (t_{k+1} - t_k)$;
    // add contribution of fragment to the derivative
    $\dot{q}_i \leftarrow \dot{q}_i + \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right] (q_j - q_i)(t_{k+1} - t_k) t_c f(\lfloor x_c \rfloor, \lfloor y_c \rfloor)$;
    $\dot{q}_j \leftarrow \dot{q}_j + \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right] (q_j - q_i)(1 - t_c) f(\lfloor x_c \rfloor, \lfloor y_c \rfloor)$;

In order to make the pseudo-code 10 comprehensive, we explain some of the intermediary variables:

$$\begin{aligned} x(t) &= (1-t)x_i + tx_j = A_x[1,t]^T \\ y(t) &= (1-t)y_i + ty_j = A_x[1,t]^T \\ x(t)y(t) &= A_x y[1,t,t^2]^T \end{aligned} \tag{A.15}$$

$$v_d = \int_{t_k}^{t_{k+1}} t^{d-1} dt \tag{A.16}$$

$$\begin{aligned} c_0 &= \int_{t_k}^{t_{k+1}} t^{d-1} dt \\ c_x &= \int_{t_k}^{t_{k+1}} \varepsilon(x(t)) t^{d-1} dt \\ c_y &= \int_{t_k}^{t_{k+1}} \varepsilon(y(t)) t^{d-1} dt \\ c_{xy} &= \int_{t_k}^{t_{k+1}} \varepsilon(x(t))\varepsilon(y(t)) t^{d-1} dt \end{aligned} \tag{A.17}$$

After summation over all subpixel fragments of a segment we have:

$$\begin{aligned} \beta &= \int_0^1 [1,t,t^2]^T f_c(x(t),y(t)))dt \\ \gamma &= \int_0^1 [1,t,t^2]^T \nabla f c(x(t),y(t)))dt \end{aligned} \tag{A.18}$$

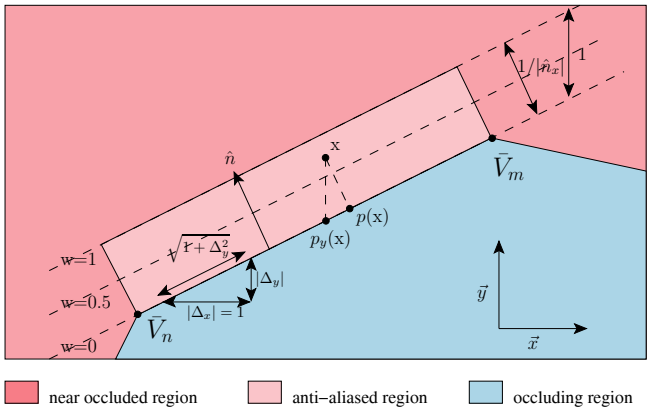
And we finally obtain.

$$\begin{aligned} \alpha_1 &= \int_0^1 f_c((1-t)q_i + tq_j)t dt \\ \alpha_2 &= \int_0^1 f_c((1-t)q_i + tq_j)(1-t) dt \\ \alpha_3 &= \int_0^1 \nabla f_c((1-t)q_i + tq_j)\ t^2 dt \\ \alpha_4 &= \int_0^1 \nabla f_c((1-t)q_i + tq_j)(1-t)^2 dt \\ \alpha_5 &= \int_0^1 \nabla f_c((1-t)q_i + tq_j)(1-t)\ t\ dt \end{aligned} \tag{A.19}$$

Algorithm 10: integrateSegmentDerivativesBilinear

Data: The polygon vertices $(q_i)_{i=1}^N$ and a discretized function $f(x, y)$
Result: The derivatives $\dot{q}_i \equiv \frac{\partial I}{\partial q_i}$ and $H_{ij} \equiv \frac{\partial^2 I}{\partial q_i q_j}$ of the integral I of f_c inside the polygon with f_c the bilinear interpolation of f

// loop over edges of the polygon
for $i \leftarrow 1$ **to** N **do**
  $j = mod(i, N) + 1$;
  $\beta \leftarrow 0_{3\times 1}$; $\gamma \leftarrow 0_{3\times 2}$
  $(p_k, t_k)_{k=1}^K \leftarrow$ intersectSegment(q_i, q_j);*// clip the segment $\overline{q_i, q_j}$ on the pixel grid*
  // loop over subpixel fragments
  for $k \leftarrow 1$ **to** $K - 1$ **do**
    $x_c = (x_k + x_{k+1})/2$; $y_c = (y_k + y_{k+1})/2$; $t_c = (t_k + t_{k+1})/2$;
    $\Delta_x = x_{k+1} - x_k$; $\Delta_y = y_{k+1} - y_k$; $\Delta_t = t_{k+1} - t_k$;
    $A_x \leftarrow [\varepsilon(x_c) - (\Delta_x t_c/\Delta_t),\ \Delta_x/\Delta_t]$; *// see eqn.A.15*
    $A_y \leftarrow [\varepsilon(y_c) - (\Delta_y t_c/\Delta_t),\ \Delta_y/\Delta_t]$;
    $A_{xy} = [A_x(1)A_y(1),\ A_x(1)A_y(2) + A_x(2)A_y(1),\ A_x(2)A_y(2)]$;
    for $d \leftarrow 1$ **to** 5 **do**
      $v_d = ((t_{k+1})^d - (t_k)^d)/d$; *// see eqn.A.16*
    $f_{00} \leftarrow f(\lfloor x_c \rfloor, \lfloor y_c \rfloor)$; $f_{01} \leftarrow f(\lfloor x_c \rfloor, \lfloor y_c \rfloor + 1)$;
    $f_{10} \leftarrow f(\lfloor x_c \rfloor + 1, \lfloor y_c \rfloor)$; $f_{11} \leftarrow f(\lfloor x_c \rfloor + 1, \lfloor y_c \rfloor + 1)$;
    $b_0 \leftarrow f_{00}$; $b_x \leftarrow f_{10} - f_{00}$; $b_y \leftarrow f_{01} - f_{00}$;
    $bxy \leftarrow f_{11} - f_{10} + f_{00} - f_{01}$;
    for $d \leftarrow 1$ **to** 3 **do**
      $c_0 \leftarrow v_d$;
      $c_x \leftarrow A_x[v_d, v_{d+1}]^T$; *// see eqn.A.17*
      $c_y \leftarrow A_y[v_d, v_{d+1}]^T$;
      $c_{xy} \leftarrow A_{xy}[v_d, v_{d+1}, v_{d+2}]^T$;
      $\beta_d \leftarrow \beta_d + b_0 c_0 + b_x c_x + b_y c_y + b_{xy} c_{xy}$; *// see eqn.A.18*
      $\gamma_{[d,:]} \leftarrow \gamma_{[d,:]} + b_{xy} c_y + [b_x, b_y]^T c_0$;
  $\alpha_1 \leftarrow [0, 1, 0]\beta$; *// see eqn.A.19*
  $\alpha_2 \leftarrow [1, -1, 0]\beta$; $\alpha_3 \leftarrow [0, 0, 1]\gamma$; $\alpha_4 \leftarrow [1, -2, 1]\gamma$; $\alpha_5 \leftarrow [0, 1, -1]\gamma$;
  $J \leftarrow \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$;
  $\dot{q}_j \leftarrow \dot{q}_j + J(q_j - q_i)\alpha_1$;
  $\dot{q}_i \leftarrow \dot{q}_i + J(q_j - q_i)\alpha_2$;
  $H_{ii} \leftarrow H_{ii} - J\alpha_2 + J(q_j - q_i)\alpha_4$;
  $H_{jj} \leftarrow H_{jj} + J\alpha_1 + J(q_j - q_i)\alpha_3$;
  $H_{ji} \leftarrow J\alpha_2 + J(q_j - q_i)\alpha_5$;
  $H_{ji} \leftarrow H_{ji}^T$;



Descriptio A.1: Antialiasing weights for points in the vicinity of the segment

A.3 identifying occlusion forces

Let consider the residual along a segment joining two successive vertices $\bar{V_m}$ and $\bar{V_n}$ along the occlusion boundary. Let assume this segment to be rather horizontal i.e $|\hat{n}_y| > |\hat{n}_x|$ and that we have $\hat{n}_y > 0$. We denote in the sequel $\bar{V}_m = (x_m, y_m)$ and $\bar{V}_n = (x_n, y_n)$. Let assume without loss of generality that $\hat{n}_y$ is positive i.e that the occluding side lies below the segment. While computing $\nabla_\theta \bar{E}$ we need to differentiate $\bar{R}$ (eqn.4.63) with respect to θ. Some term are due to changes in the residuals R while other are due to changes in the anti-aliasing weights w. We consider here the terms due to the change in weights and thus assume the residuals R to remain constant as θ varies. For notation concision we remove θ,L and T of R from the list of parameters of R. We aim at identifying the term due to change in weights with the second line in the equation 4.54. We denote this term C

$$C = \sum_{\mathbf{x} \in \mathcal{A} \cap \mathbb{N}^2} \frac{\partial w(\mathbf{x})}{\partial \theta_j} [R(\mathbf{x}) - R(p(\mathbf{x}))] \tag{A.20}$$

We assumed the segment to be rather horizontal ($|\hat{n}_y| > |\hat{n}_x|$) and that $\hat{n}_y > 0$, therefore w can be rewritten as follow:

$$w(\mathbf{x}) = \frac{(\mathbf{x} - p(\mathbf{x})) \cdot \hat{n}}{\hat{n}_y} \tag{A.21}$$

We denote $p_y(\mathbf{x})$ the projections along the vertical direction of the point $\mathbf{x}$ onto the line that extend the considered segment. We denote $(\vec{x}, \vec{y})$ the coordinate system of the image plane. For any point $\mathbf{x}$ in $\mathcal{A}$ we can show that we have

$$w(\mathbf{x}) = (\mathbf{x} - p_y(\mathbf{x})).\vec{y} \tag{A.22}$$

Differencing w with respect to θ_i gives:

$$\frac{\partial w(\mathbf{x})}{\partial \theta_j} = -\frac{\partial p_y(\mathbf{x})}{\partial \theta_j}.\vec{y} \tag{A.23}$$

the vertically projected point $p_y(\mathbf{x})$ lies in the segment joining $\bar{V}_m$ and $\bar{V}_n$, therefore there exist $t \in [0,1]$ such that

$$p_y(\mathbf{x}) = (1-t)\bar{V}_m + t\bar{V}_n \tag{A.24}$$

We can then differentiate $p_y(\mathbf{x})$ with respect to θ_j:

$$\frac{\partial p_y(\mathbf{x})}{\partial \theta_j} = \frac{\partial t}{\partial \theta_j}(\bar{V}_n - \bar{V}_m) + (1-t)\frac{\partial \bar{V}_m}{\partial \theta_j} + t\frac{\partial \bar{V}_n}{\partial \theta_j} \tag{A.25}$$

Because $(\bar{V}_n - \bar{V}_m).\hat{n} = 0$ and given the definition of the curve speed v_j (eqn.4.52), we get:

$$\hat{n}.\frac{\partial p_y(\mathbf{x})}{\partial \theta_j} = \hat{n}.v_j(p_y(\mathbf{x})) \tag{A.26}$$

Because $\vec{x}.\frac{\partial p_y(\mathbf{x})}{\partial \theta_j} = 0$ we get :

$$\hat{n}.\frac{\partial p_y(\mathbf{x})}{\partial \theta_j} = (\hat{n}_x\vec{x} + \hat{n}_y\vec{y}).\frac{\partial p_y(\mathbf{x})}{\partial \theta_j} = \hat{n}_y(\frac{\partial p_y(\mathbf{x})}{\partial \theta_j}.\vec{y}) \tag{A.27}$$

Therefore

$$\frac{\partial w(\mathbf{x})}{\partial \theta_j} = -\frac{\partial p_y(\mathbf{x})}{\partial \theta_j}.\vec{y} = -\frac{\hat{n}}{\hat{n}_y}.\frac{\partial p_y(\mathbf{x})}{\partial \theta_j} = -\frac{\hat{n}.v_j(p_y(\mathbf{x}))}{\hat{n}_y} \tag{A.28}$$

We can rewrite A as follow:

$$C = \frac{1}{\hat{n}_y}\sum_{\mathbf{x}\in\mathcal{A}\cap\mathbb{N}^2} -[R(\mathbf{x}) - R(p(\mathbf{x}))]\hat{n}.v_j(p_y(\mathbf{x})) \tag{A.29}$$

We show now that, with some approximation we could identify this term with the second term in the equation 4.54. We assume R and R^+ to be smooth and we get

$$R(p(\mathbf{x})) \approx R(p_y(\mathbf{x})) \tag{A.30}$$
$$R(\mathbf{x}) \approx R^+(p_y(\mathbf{x})) \tag{A.31}$$

Therefore, given the definition of the *occlusion forces* (eqn.4.55), C can be approximated by $\tilde{C}$ defined as :

$$\tilde{C} = \frac{1}{\hat{n}_y}\sum_{\mathbf{x}\in\mathcal{A}\cap\mathbb{N}^2} -f_{oc}(p_y(\mathbf{x})).v_j(p_y(\mathbf{x})) \tag{A.32}$$

The division by $max(|\hat{n}_x|, |\hat{n}_y|)$ in our definition of the weight [eqn.4.61] ensure that (for a segment that is rather horizontal) within each vertical line there is a single point $\mathbf{x}$ with integer coordinates and its weight in $[0,1)$. This appears more clearly in [eqn.A.22]. Given a vertical line with constant x, this point has the coordinates $\mathbf{x} = (x, \lceil\bar{y}(x)\rceil)$ where

$$\bar{y}(x) = y_m + (x - x_m)\Delta_y \tag{A.33}$$

with

$$\Delta_y = (y_n - y_m)/(x_n - x_m) \tag{A.34}$$

Δ_y is slope of the segment or the increment we need to perform on the y coordinate when x is incremented by 1 if one want to stay on the segment. We obtain from [eqn.4.62]:

$$\mathcal{A} \cap \mathbb{N}^2 = \{(x, \lceil \bar{y}(x) \rceil) | x \in \mathbb{N}, p((x, \lceil \bar{y}(x) \rceil) \in \overline{\bar{V}_m \bar{V}_n}\} \tag{A.35}$$

We assume now that the condition that the point x should orthogonally project into the segment (i.e $p((x, \lceil \bar{y}(x) \rceil) \in \overline{\bar{V}_m \bar{V}_n}$) can be approximated by the condition $x \in [x_n, x_m]$ as the line is rather vertical. The resulting approximate anti-aliased region is no more a rectangle but a but parallelogram with two vertical sides. After discretization on the grid this might result in some cases in neglecting a single point near the extremities of the segment.

$$\mathcal{A} \cap \mathbb{N}^2 \approx \{(x, \lceil \bar{y}(x) \rceil) | x \in \{\lceil x_n \rceil, \ldots, \lfloor x_m \rfloor\}\} \tag{A.36}$$

This approximation of the anti-aliased region and the fact that $p_y(x, \lceil \bar{y}(x) \rceil) = (x, \bar{y}(x))$ allows us to approximate $\tilde{C}$ by $\tilde{C}_2$ as follow:

$$\tilde{C}_2 = \frac{1}{\hat{n}_y} \sum_{x=\lceil x_n \rceil}^{\lfloor x_m \rfloor} -f_{\text{oc}}\big((x, \bar{y}(x))\big).v_j\big((x, \bar{y}(x))\big) \tag{A.37}$$

We introduce $\Delta_V = (1, \Delta_y)$ to denote the 2D displacement along the segment $\overline{\bar{V}_m \bar{V}_n}$ when x is incremented by one. We have

$$|\Delta_V| = \sqrt{1 + \Delta_y^2} = \frac{1}{\hat{n}_y} \tag{A.38}$$

We also introduce $t = x - \lceil x_n \rceil$, $N = \lfloor x_m \rfloor - \lceil x_n \rceil$ and $\varepsilon = \lceil x_n \rceil - x_n$. We obtain after some derivation:

$$\tilde{C}_2 = |\Delta_V| \sum_{t=0}^{N-1} -f_{\text{oc}}(\bar{V}_n + (t + \varepsilon)\Delta_V).v_j(\bar{V}_n + (t + \varepsilon)\Delta_V) \tag{A.39}$$

This last approximation of C can be easily identified as an discrete approximation of the integral along the segment $\overline{\bar{V}_m \bar{V}_n}$ of the function

$$f(\mathbf{x}) = -f_{\text{oc}}(\mathbf{x}).v_j(\mathbf{x}) \tag{A.40}$$

This term corresponds to the contribution of the considered segment in the second term of the equation 4.54. We demonstrated that the implementation based on a discrete image domain with anti-aliasing yield, after differentiation, terms that are consistent (up to some approximations) with the *occlusion forces* that where obtained by differentiating the objective function defined with a continuous image domain.

A.4 Kernel PCA

We briefly describe the Kernel PCA method [Schölkopf 1998] and it use to define the pose prior. Let $x_1, \ldots, x_n$ the set of training points land ϕ a non linear mapping from the input space to a higher dimensional space F called the $f$eature space. The mapping ϕ can either be defined explicitly or implicitly through the use of Mercer kernel. In the latter case one first define a Mercer kernel $k(.,.)$ such that for all set of data points $(x_i)_{i=1}^n$, the kernel matrix with elements $K_{ij} = k(x_i, x_j)$ is symmetric positive definite. Then using Mercer's theorem [Schölkopf 1998] it can been shown that there exist a mapping ϕ into some high dimensional feature such that $k(x, y) =< \phi(x), \phi(y) >$.

Given the training point we define the kernel matrix $K_{ij} = k(x_i, x_j)$ we denote $\phi_0 := \frac{1}{N}\sum_{i=1}^N \phi(x_i)$ the mean of the mapped points, and $\bar{\phi}(x_i) := \phi(x_i) - \phi_0$ the set of points after centering. Let Σ_ϕ denote the covariance matrix of the elements of the training set mapped by ϕ.

$$\Sigma_\phi = \frac{1}{N}\sum_i \bar{\phi}(x_i).\bar{\phi}(x_i)^t \tag{A.41}$$

The centered kernel function is defined as $\bar{k}(x, y) =< \bar{\phi}(x).\bar{\phi}(y) >$ And can be rewritten using the kernel functionï¿½:

$$\begin{aligned}\bar{k}(x, y) =& k(x, y) - \frac{1}{n}\sum_{i=1}^n (k(x, x_i) + k(y, x_i)) \\ &+ \frac{1}{n^2}\sum_{i,j=1}^n k(x_i, x_j)\end{aligned} \tag{A.42}$$

The centered kernel matrix is defined byï¿½$\bar{K}_{ij} = k(x_i, x_j)$. It can easily be shown that $\bar{K} = HKH$ where $H = I - \frac{1}{N}\mathbf{1}\mathbf{1}^t$ and $\mathbf{1} = [1, \ldots, 1]^t$ is a $N \times 1$ vector. We denote λ_k the eigen values of the covariance matrix in feature space Σ_ϕ and the V_k associated eigen vectors. These vector lie in the span of the mapped training data and thus decompose as follow:

$$V_k = \sum_{i=1}^n \alpha_{ik}\bar{\phi}(x_i) \tag{A.43}$$

Scholkopf et al. showed that the eigen values λ_k of the covariance matrix Σ_ϕ and the associated expansion coefficients $\alpha_i k$ can be obtained using eigen decomposition of the centered kernel matrix $\bar{K}$. Using eigen value decomposition

$$\bar{K} = USU \tag{A.44}$$

where $S = \mathrm{diag}(\gamma_1, \ldots, \gamma_n)$ is a diagonal matrix containing eigen values of $\bar{K}$. the matrix U is an orthonormal matrix whose columns are eigen vectors. The element u_{ik} represent the i^{th} coordinates of the $k^t h$ eig en vector. The eigen values λ_k are given by $\lambda_k = \gamma_k/n$ and the expansion coefficients α_{ik} are given by : $\alpha_{ik} = u_{ik}/\sqrt{\gamma_k}$;

Given a dimension l we define the l-dimensional kernel PCA plane to be the affine space containing ϕ_0 and spanned by the l eigen vector associated to the l largest eigen values of the covariance matrix. We denote P^l the projection onto the kernel PCA plane. Given a point x in the input space, the squared distance of its corresponding feature point to the kernel PCA plane writes:

$$\begin{aligned} D_F^2(\phi(x), P^l(\phi(x)) &= \|\phi(x) - P^l(\phi(x))\| \\ &= \|\bar{\phi}(x)\|^2 - \sum_{k=1}^{l} < V_k, \bar{\phi}(x) >^2 \\ &= \bar{k}(x,x) - \sum_{k=1}^{l}(\sum_{i=1}^{n} \alpha_{ik}\bar{k}(x_i,x))^2 \\ &= \bar{k}(x,x) - \bar{k}_x^t M \bar{k}_x \end{aligned} \tag{A.45}$$

where $k_x = [k(x,x_1), \ldots, k(x,x_n)]^t$, $\bar{k}_x = H(k_x - \frac{1}{n}K\mathbf{1})$, $\bar{k}(x,x) = k(x,x) + \frac{1}{N}\mathbf{1}^t K\mathbf{1} - \frac{2}{N}\mathbf{1}^t k_x$ and the elements of the matrix M defined by:

$$M_{ij} = \sum_{k=1}^{l} \alpha_{ik}\alpha_{jk} = \sum_{k=1}^{l} \frac{1}{\gamma_k} u_{ik}u_{jk} \tag{A.46}$$

Following [Cremers 2003] we can use the eigen values λ_k to compute the squared Mahalanobis distance within the kernel PCA plane between the projected point $P^l(\phi(x))$ and the mean ϕ_0:

$$D_M^2(P^l(\phi(x)), \phi_0) = \sum_{k=1}^{l} \frac{1}{\lambda_k}(\sum_{i=1}^{n} \alpha_i^k \bar{k}(x_i,x))^2 = \bar{k}_x^t D \bar{k}_x \tag{A.47}$$

where elements of the matrix D are

$$D_{ij} = \sum_{k=1}^{l} (\alpha_{ik}\alpha_{jk})/\lambda_k \tag{A.48}$$

One can finally choose the closeness measure to the set of training points $x_1, \ldots, x_n$ to be a combination of the distance to the plan and the Mahalanobis distance within the plane :

$$\begin{aligned} E_{prior}(x) &= (x) D_F^2(\phi(x), P^l(\phi(x))/\lambda_\perp + D_M(P^l(\phi(x)), \phi_0) \\ &= \frac{1}{\lambda_\perp}(\bar{k}(x,x) - \bar{k}_x^t M \bar{k}_x) + \bar{k}_x^t D \bar{k}_x \end{aligned} \tag{A.49}$$

where $0 < \lambda_\perp \leq \lambda_l$ such that distance to the kernel PCA plane is more penalized than the distance along any direction within the plane (see [Cremers 2003]).

The gradient of the energy can simply be obtained using chain derivation rule :

$$\frac{\partial E_{prior(x)}}{\partial x} = \frac{1}{\lambda_\perp}\left(\frac{\partial}{\partial x}(\bar{k}(x,x)) - 2\bar{k}_x^t M \frac{\partial \bar{k}_x}{\partial x}\right) + 2\bar{k}_x^t D \frac{\partial \bar{k}_x}{\partial x} \tag{A.50}$$

where

$$\frac{\partial k_x}{\partial x} = \left[\frac{\partial k(x,x_1)}{\partial x},\ldots,\frac{\partial k(x,x_n)}{\partial x}\right]^t \tag{A.51}$$

$$\frac{\partial}{\partial x}(\bar{k}(x,x)) = \frac{\partial}{\partial x}k(x,x) - \frac{2}{N}\mathbf{1}^t\frac{\partial k_x}{\partial x} \tag{A.52}$$

$$\frac{\partial \bar{k}_x}{\partial x} = H\frac{\partial k_x}{\partial x} \tag{A.53}$$

Conspectus librorum

[Absil 2008] P.A. Absil, R. Mahony and R. Sepulchre. Optimization algorithms on matrix manifolds. Princeton University Press, Princeton, NJ, 2008. 58, 59, 60

[Agarwal 2004] A. Agarwal and B. Triggs. *Tracking Articulated Motion Using a Mixture of Autoregressive Models.* In ECCV, volume 3, pages 54–65, 2004. 146

[Agarwal 2006] A. Agarwal and B. Triggs. *Recovering 3D human pose from monocular images.* PAMI, vol. 28, no. 1, pages 44–58, 2006. 47

[Aggarwal 1999] J.K. Aggarwal and Q. Cai. *Human Motion Analysis: A Review.* CVIU, vol. 73, pages 428–440, 1999. 49

[Aizerman 1964] A. Aizerman, E.M. Braverman and L.I. Rozoner. *Theoretical foundations of the potential function method in pattern recognition learning.* Automation and Remote Control, vol. 25, pages 821–837, 1964. 147

[Appel 1967] A. Appel. *The notion of quantitative invisibility and the machine rendering of solids.* In ACM National Conference, pages 387–393, 1967. 154, 155

[Atherton 1978] P. Atherton, K. Weiler and D. Greenberg. *Polygon shadow generation.* In SIGGRAPH, pages 275–281, 1978. 154

[Athitsos 2002] V. Athitsos and S. Sclaroff. *An Appearance-Based Framework for 3D Hand Shape Classification and Camera Viewpoint Estimation.* In AFGR, pages 45–48, 2002. 20, 22, 43, 44, 45

[Athitsos 2003a] V. Athitsos and S. Sclaroff. *Database Indexing Methods for 3D Hand Pose Estimation.* In Gesture Workshop, pages 288–299, 2003. 20

[Athitsos 2003b] V. Athitsos and S. Sclaroff. *Estimating 3D Hand Pose from a Cluttered Image.* In CVPR, pages 432–442, 2003. 43, 44, 45

[Balan 2007] A.O. Balan, M.J. Black, H. Haussecker and L. Sigal. *Shining a Light on Human Pose: On Shadows, Shading and the Estimation of Pose and Shape.* In ICCV, pages 1–8, 2007. 50, 128

[Bar 2009] L. Bar and G. Sapiro. *Generalized Newton-Type Methods For Energy Formulations In Image Processing.* SIIMS, vol. 2, no. 2, pages 508–531, 2009. 107

[Barber 1996] C.B. Barber, D.P. Dobkin and H. Huhdanpaa. *The quickhull algorithm for convex hulls.* TMS, vol. 22, no. 4, pages 469–483, 1996. 72

[Berberich 2002] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn and E. Schömer. *A Computational Basis for Conic Arcs and Boolean Operations on Conic Polygons.* In Annual European Symposium on Algorithms, pages 174–186, 2002. 77, 80

[Blanz 1999] V. Blanz and T. Vetter. *A Morphable Model for the Synthesis of 3D Faces.* In SIGGRAPH, pages 187–194, 1999. 135, 145

[Boissonnat 1995] J.D. Boissonnat and M. Yvinec. Géométrie algorithmique. Ediscience international, 1995. 81

[Bray 2004a] M. Bray, E. Koller-Meier and L.J. Van Gool. *Smart particle filtering for 3D hand tracking.* In AFGR, pages 675–680, 2004. 29, 30, 37, 42, 130, 132

[Bray 2004b] M. Bray, E. Koller-Meier, L.J. Van Gool and N.N. Schraudolph. *3D Hand Tracking by Rapid Stochastic Gradient Descent Using a Skinning Model.* In European Conference on Visual Media Production (CVMP), pages 59–68, 2004. 22, 26, 29, 30, 37, 39, 108, 120, 130, 132, 146

[Bremer 1998] D. Bremer and J.F. Hughes. *Rapid Approximate Silhouette Rendering of Implicit Surfaces.* In Proceesings of Implicit Surfaces, pages 155–164, 1998. 129

[Bresenham 1965] J.E. Bresenham. *Algorithm for Computer Control of a Digital Plotter.* IBM System Journal, vol. 4, no. 1, pages 25–30, 1965. 96

[Broyden 1970] C.G. Broyden. *The Convergence of a Class of Double-rank Minimization Algorithms.* Journal of the Institute of Mathematics and Its Applications, vol. 6, pages 76–90, 1970. 109

[Buchmann 2004] V. Buchmann, S. Violich and A. Billinghurst M.and Cockburn. *FingARtips: gesture based direct manipulation in Augmented Reality.* In 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE), pages 212–221, 2004. 22

[Burger 2003] M. Burger. *Levenberg-Marquardt Level Set Methods for Inverse Obstacle Problems.* Inverse Problems, vol. 20, pages 20–259, 2003. 107

[Cameron 1986] S.A. Cameron and R.K. Culley. *Determining the minimum translational distance between two convex polyhedra.* In International Conference on Robotics and Automation, pages 591–596, 1986. 184

[Carpenter 1984] L. Carpenter. *The A-buffer, an antialiased hidden surface method.* In SIGGRAPH, pages 103–108, 1984. 159

[Catmull 1978] E. Catmull. *A hidden-surface algorithm with anti-aliasing.* In SIGGRAPH, pages 6–11, 1978. 91, 92

[Chazelle 1992] B. Chazelle and H. Edelsbrunner. *An optimal algorithm for intersecting line segments in the plane.* ACM, vol. 39, no. 1, pages 1–54, 1992. 77, 81

[Chen 1996] X. Chen. *Convergence of the BFGS Method for LC1 Convex Constrained Optimization.* SIAM J. Control Optim., vol. 34, no. 6, pages 2051–2063, 1996. 110

[Cheng 1992] R.C.H Cheng. *Fast Linear Color Rendering.* Graphics gems, vol. 3, pages 343–354, 1992. 96

[Cheng 2004] S.Y Cheng and M.M. Trivedi. *Hand Pose Estimation Using Expectation-Constrained-Maximization From Voxel Data.* Rapport technique, Computer Vision and Robotics Research (CVRR) Laboratory, University of California, San Diego, 2004. 30, 37, 38, 41

[Cheng 2006] S.Y. Cheng and M.M. Trivedi. *Multimodal Voxelization and Kinematically Constrained Gaussian Mixture Models for Full Hand Pose Estimation: An Integrated Systems Approach.* In CVS, page 34, 2006. 21, 30, 37, 41

[Chik 2007] D. Chik, J. Trumpf and N.N. Schraudolph. *3D Hand Tracking in a Stochastic Approximation Setting.* In HUM, pages 136–151, 2007. 20, 29, 39, 130, 132

[Chin 1989] N. Chin and S. Feiner. *Near real-time shadow generation using BSP trees.* In SIGGRAPH Comput. Graph., pages 99–106, 1989. 154, 155

[Chrysanthou 1995] Y. Chrysanthou and M. Slater. *Shadow volume BSP trees for computation of shadows in dynamic scenes.* In symposium on Interactive 3D graphics, pages 45–50, 1995. 154

[Chua 2000] C.S. Chua, H.Y. Guan and Y.K. Ho. *Model-based finger posture estimation.* In ACCV, pages 43–48, 2000. 30

[Chua 2002] C.S. Chua, H.Y. Guan and Y.K. Ho. *Model-based 3D hand posture estimation from a single 2D image.* IVC, vol. 20, no. 3, pages 191–202, 2002. 20

[Clarke 1998] T.A. Clarke and J.G. Fryer. *The Development of Camera Calibration Methods and Models.* The Photogrammetric Record, vol. 16, no. 91, pages 51–66, 1998. 20

[Conn 2000] A.R Conn, N.I.M. Gould and P.L. Toint. Trust-region methods. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 112, 113, 160

[Cremers 2003] D. Cremers, T. Kohlberger and C. Schnorr. *Shape statistics in kernel space for variational image segmentation.* PR, vol. 36, no. 9, pages 1929–1943, 2003. 148, 201

[Crow 1977] F.C. Crow. *Shadow algorithms for computer graphics.* In SIGGRAPH, pages 242–248, 1977. 154

[Crow 1981] F.C. Crow. *A Comparison of Antialiasing Techniques.* Computer Graphics and Applications, vol. 1, no. 1, pages 40–48, 1981. 159

[Cui 2004] J-S Cui and Z-Q Sun. *Vision-Based Hand Motion Capture Using Genetic Algorithm.* Lecture Notes in Computer Science, vol. 3005, pages 289–300, 2004. 20, 31, 42

[de Campos 2006] T.E. de Campos and D.W. Murray. *Regression-based Hand Pose Estimation from Multiple Cameras.* In CVPR, pages 782–789, 2006. 21, 43, 46, 47

[de La Gorce 2006] M. de La Gorce and N. Paragios. *Monocular Hand Pose Estimation Using Variable Metric Gradient-Descent.* In BMVC, volume 3, page 1269, 2006. 15, 110

[de La Gorce 2008] M. de La Gorce, D. Fleet and N. Paragios. *Model-Based Hand Tracking with Texture, Shading and Self-occlusions.* In CPVR, pages 1–8, 2008. 15, 146, 169

[de La Gorce 2010a] M. de La Gorce, D.J. Fleet and N. Paragios. *Model-based 3D Hand Pose Estimation from Monocular Video.* PAMI, 2010. 15

[de La Gorce 2010b] M. de La Gorce and N. Paragios. *A variational approach to monocular hand-pose estimation.* CVIU, vol. 114, pages 363–372, 2010. 15

[Delamarre 1998] Q. Delamarre and O. Faugeras. *Finding Pose of Hand in Video Images: A Stereo-Based Approach.* In AFGR, page 585, 1998. 20, 26, 28, 37, 41

[Delamarre 1999] Q. Delamarre and O. Faugeras. *3D Articulated Models and Multi-View Tracking with Silhouettes.* In ICCV, pages 716–721, 1999. 41, 84

[Delaunoy 2008] A. Delaunoy, E. Prados, P. Gargallo, J-P Pons and P. Sturm. *Minimizing the Multi-view Stereo Reprojection Error for Triangular Surface Meshes.* In BMVC, pages 1–10, 2008. 153

[Delaunoy 2009] A. Delaunoy, K. Fundana, E. Prados and A. Heyden. *Convex Multi-Region Segmentation on Manifolds.* In ICCV, pages 662–669, 2009. 166

[Demirdjian 2003] D. Demirdjian. *Enforcing constraints for human body tracking.* In CVPR Workshop on Multi-Object Tracking, page 102, 2003. 58

[Dewaele 2004] G. Dewaele, F. Devernay and R.P. Horaud. *Hand Motion from 3D Point Trajectories and a Smooth Surface Model.* In ECCV, volume 1, pages 495–507, 2004. 20, 26, 29, 34, 36, 40, 129

[DiZio 2002] P. DiZio and J. Lackner. *Proprioceptive adaptation and aftereffects.* Handbook of Virtual Environments, pages 751–777, 2002. 49

[Dorfmuller-Ulhaas 2001] K. Dorfmuller-Ulhaas and D. Schmalstieg. *Finger Tracking for Interaction in Augmented Environments.* Rapport technique TR-186-2-01-03, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, February 2001. human contact: technical-report@cg.tuwien.ac.at. 22

[Dorner 1994] B. Dorner. Chasing the colour glove : Visual hand tracking. Master's thesis, Simon Fraser Univ. (Canada), 1994. 22, 30, 36

[Du 2008] H. Du and E. Charbon. *A Virtual Keyboard System based on Multi-Level Feature Matching.* In HIS, pages 170–175, 2008. 22, 28, 29, 35, 130, 132

[Duff 1989] T. Duff. *Polygon Scan Conversion by Exact Convolution.* Raster Imaging & Digital Typography, 1989. 91, 92

[Erol 2007] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle and X. Twombly. *Vision-based hand pose estimation: A review.* CVIU, vol. 108, no. 1-2, pages 52–73, 2007. 12, 19

[Fellner 1993a] D.W. Fellner and C. Helmberg. *Best Approximate General Ellipses on Integer Grids.* Rapport technique, University of Bonn, 1993. 98

[Fellner 1993b] D.W. Fellner and C. Helmberg. *Robust rendering of general ellipses and elliptical arcs.* ACM Trans. Graph., vol. 12, no. 3, pages 251–276, 1993. 98

[Fisher 2001] S. Fisher and M.C. Lin. *Deformed distance fields for simulation of non-penetrating flexible bodies.* In Eurographic workshop on Computer Animation and Simulation, pages 99–111, 2001. 184

[Fletcher 1970] R. Fletcher. *A New Approach to Variable Metric Algorithms.* Computer Journal, vol. 13, pages 317–322, 1970. 109

[Fogel 2006] E. Fogel, R. Wein, B. Zukerman and D. Halperin. *2D Regularized Boolean Set-Operations.* CGAL-3.2 User and Reference Manual, 2006. 77

[Foley 1996] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes. Computer graphics (2nd ed. in c): principles and practice. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996. 91

[Fujimoto 1983] A. Fujimoto and K. Iwata. *Jag-Free Images on Raster Displays.* Computer Graphics and Applications, vol. 3, no. 9, pages 26–34, 1983. 96

[Galimberti 1969] R. Galimberti. *An algorithm for hidden line elimination.* Commun. ACM, vol. 12, pages 206–211, 1969. 154

[Gargallo 2007] P. Gargallo, E. Prados and P. Sturm. *Minimizing the Reprojection Error in Surface Reconstruction from Images.* In ICCV, pages 1–8, 2007. 153

[Gavrilova 2000] M.L. Gavrilova and J.G. Rokne. *Reliable line segment intersection testing.* Computer-Aided Design, vol. 32, no. 12, pages 737–745, 2000. 82

[Ge 2008] S.S. Ge, Y. Yang and T.H. Lee. *Hand gesture recognition and tracking based on distributed locally linear embedding.* Image and Vision Computing, vol. 26, no. 12, pages 1607 – 1620, 2008. 20

[Geman 1992] D. Geman and G. Reynolds. *Constrained Restoration and the Recovery of Discontinuities.* PAMI, vol. 14, no. 3, pages 367–383, 1992. 170

[Ghahramani 1996] Z. Ghahramani and G. Hinton. *Parameter estimation for linear dynamical systems.* Rapport technique CRG-TR-96-2, Department of Computer Science, University of Toronto, 1996. 119

[Goldfarb 1970] D. Goldfarb. *A Family of Variable Metric Updates Derived by Variational Means.* Mathematics of Computation, vol. 24, pages 23–26, 1970. 109

[Gong 2009] Y-X. Gong, Y. Liu, L. Wu and Y-B. Xie. *Boolean Operations on Conic Polygons.* Journal Of Computer Science And Technology, vol. 24, pages 568–577, 2009. 77, 80

[Gordon 1993] N. Gordon. *Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation.* Radar and Signal Processing, vol. 140, pages 107–113, 1993. 119

[Gordon 2002] N. Gordon. *A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking.* Signal Processing, vol. 50, pages 174–188, 2002. 119

[Granger 2002] S. Granger and X. Pennec. *Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration.* In ECCV, pages 418–432, 2002. 40

[Grant 1985] C.W. Grant. *Integrated analytic spatial and temporal anti-aliasing for polyhedra in 4-space.* In SIGGRAPH, pages 79–84, 1985. 186

[Greiner 1998] G. Greiner and K. Hormann. *Efficient clipping of arbitrary polygons.* ACM Trans. Graph., vol. 17, no. 2, pages 71–83, 1998. 78

[Griewank 2000] A. Griewank. Evaluating derivatives: principles and techniques of algorithmic differentiation. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 62, 159

[Griffin 2000] W.B. Griffin, R.P. Findley, M.L. Turner and M.R. Cutkosky. *Calibration and Mapping of a Human Hand for Dexterous Telemanipulation*. In Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems, 2000. 26

[Guan 1999] H.Y. Guan, C.S. Chua and Y.K. Ho. *Hand posture estimation from 2D monocular image*. In 3DIM, pages 424–429, 1999. 20

[Guan 2006] H. Guan, R.S. Feris and M. Turk. *The Isometric Self-Organizing Map for 3D Hand Pose Estimation*. In AFGR, pages 263–268, 2006. 43, 45, 46

[Guan 2007a] H. Guan. *Vision-based three-dimensional hand posture estimation using hierarchical-isosom*. PhD thesis, University of California at Santa Barbara, Santa Barbara, CA, USA, 2007. 21, 46

[Guan 2007b] H.Y. Guan and M. Turk. *The Hierarchical Isometric Self-Organizing Map for Manifold Representation*. In CVPR, pages 1–8, 2007. 43, 45

[Heap 1996] T. Heap and D. Hogg. *Towards 3D hand tracking using a deformable model*. In AFGR, page 140, 1996. 20, 25, 29, 31, 40

[Heckbert 1989] P.S. Heckbert. *Fundamentals of Texture Mapping and Image Warping*. Rapport technique UCB/CSD-89-516, EECS Department, University of California, Berkeley, 1989. 138

[Hernández 2004] C. Hernández. *Stereo and Silhouette Fusion for 3D Object Modeling from Uncalibrated Images Under Circular Motion*. PhD thesis, ENST, May 2004. 135

[Hill 1995] K.J. Hill. *Matrix-based Ellipse Geometry*. In Graphics gems V, pages 75–77. 1995. 80

[Hintermüller 2004] M. Hintermüller and W. Ring. *A Second Order Shape Optimization Approach for Image Segmentation*. SIAM Journal of Applied Mathematics, vol. 64, no. 2, pages 442–467, 2004. 107

[Hintermüller 2007] M. Hintermüller. *A combined shape Newton and topology optimization technique in real time image segmentation*. In Real-Time PDE-Constrained Optimization, volume 3, pages 253–274. 2007. 107

[Holden 1997] E. Holden. *Visual Recognition of Hand Motion*. PhD thesis, University of Western Australia, 1997. 22, 30, 36

[Hollister 1992] A. Hollister, W. L. Buford, L. M. Myers, D. J. Giurintano and A. Novick. *The axes of rotation of the thumb carpometacarpal joint*. J. Orthop. Res., vol. 10, no. 3, pages 454–460, 1992. 26

[Hornung 1984] C. Hornung. *A Method for Solving the Visibility Problem*. Computer Graphics and Applications, vol. 4, pages 26–33, 1984. 154, 155

[Howe 2004] N.R. Howe and A. Deschamps. *Better Foreground Segmentation Through Graph Cuts*. Rapport technique, Smith College, 2004. 32

[Ilic 2003] S. Ilic and P. Fua. *Implicit Meshes for Modeling and Reconstruction*. In CVPR, volume 2, page 483, 2003. 130

[Imai 2004] A. Imai, N. Shimada and Y. Shirai. *3-D Hand Posture Recognition by Training Contour Variation*. In AFGR, pages 895 – 900, 2004. 20

[Jin 2003] H. Jin, A.J. Yezzi, Y-H. Tsai, L-T. Cheng and S. Soatto. *Estimation of 3D Surface Shape and Smooth Radiance from 2D Images: A Level Set Approach*. J. Sci. Comput., vol. 19, no. 1-3, pages 267–292, 2003. 166

[Kadous 1995] M.W. Kadous. *GRASP: Recognition of Australian Sign Language Using Instrumented Gloves*. Honours Thesis. School of Computer Science and Engineering, University of New South Wales, 1995. 9

[Kato 2006] M. Kato, Y-W. Chen and G. Xu. *Articulated Hand Motion Tracking Using ICA-based Motion Analysis and Particle Filtering*. Journal of Multimedia, vol. 1, no. 3, pages 52–60, 2006. 20

[Kim 2004] H. Kim and D.W. Fellner. *Interaction with Hand Gesture for a Back-Projection Wall*. In Computer Graphics International, pages 395–402, Washington, DC, USA, 2004. IEEE Computer Society. 22

[Kohonen 2001] T. Kohonen, M.R. Schroeder and T.S. Huang, editeurs. Self-organizing maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd édition, 2001. 46

[Koji 2002] T. Koji and Y Michiaki. *Ubi-Finger: Gesture Input Device for Mobile Use*. Transactions of Information Processing Society of Japan, vol. 43, pages 3675–3684, 2002. 49

[Kreyszig 1991] E. Kreyszig. Differential geometry. Dover, 1991. 166

[Kry 2002] P.G. Kry, D.L. James and D.K. Pai. *EigenSkin: real time large deformation character skinning in hardware*. In SSIGGRAPH, pages 153–159, New York, NY, USA, 2002. ACM. 28, 133

[Kuch 1994] J.J Kuch and T.S. Huang. *Human computer interaction via the human hand: a hand model*. In Asilomar Conference on Signals, Systems, and Computers, volume 2, pages 1252–1256, 1994. 30

[Kuch 1995] J.J. Kuch and T.S. Huang. *Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration*. In ICCV, page 666, 1995. 20, 22, 26, 27, 29, 30, 31, 32, 38, 54, 66, 67

[Kui Liu 2007] Y. Kui Liu, X. Qiang Wang, S. Zhe Bao, Ma. Gomboši and B. Alik. *An algorithm for polygon clipping, and for determining polygon intersections and unions*. Comput. Geosci., vol. 33, no. 5, pages 589–598, 2007. 78

[Lee 1995] J. Lee and T.L. Kunii. *Model-Based Analysis of Hand Posture.* Computer Graphics and Applications, vol. 15, no. 5, pages 77–86, 1995. 26, 27, 35, 38, 41, 66, 67, 68

[Lee 1996] C. Lee and Y. Xu. *Online, Interactive Learning of Gestures for Human/Robot Interfaces.* In International Conference on Robotics and Automation, volume 4, pages 2982–2987, 1996. 9

[Lee 2004] S.U. Lee and I. Cohen. *3D hand reconstruction from a monocular view.* In ICPR, volume 3, pages 310–313, 2004. 20, 28, 29

[Lewis 2000] J.P. Lewis, M. Cordner and N. Fong. *Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation.* In SIGGRAPH, pages 165–172, 2000. 132

[Li 1993] B. Li. *The Moment Calculation of Polyhedra.* PR, vol. 26, no. 8, pages 1229–1233, 1993. 186

[Li 2007] R. Li, T.P. Tian and S. Sclaroff. *Simultaneous Learning of Nonlinear Manifold and Dynamical Models for High-dimensional Time Series.* In ICCV, pages 1–8, 2007. 146

[Liang 1998] R.H. Liang and M. Ouhyoung. *A Real-Time Continuous Gesture Recognition System for Sign Language.* In AFGR, pages 558–567, 1998. 9

[Lien 1998] C.C. Lien and C.L. Huang. *Model-Based Articulated Hand Motion Tracking For Gesture Recognition.* IVC, vol. 16, no. 2, pages 121–134, 1998. 22, 30, 35, 38

[Lien 2005] C.C. Lien. *A scalable model-based hand posture analysis system.* MVA, vol. 16, no. 3, pages 157–169, 2005. 22, 30, 35, 36

[Liggett 1988] J.A. Liggett. *Exact formulae for areas, volumes and moments of polygons and polyhedra.* Communications in applied numerical methods, vol. 4, pages 815–820, 1988. 97, 156

[Lin 2000] J.Y. Lin, Y. Wu and T. S. Huang. *Modeling the Constraints of Human Hand Motion.* In Workshop on Human Motion, pages 121–126, 2000. 11, 27, 28, 67, 146

[Lin 2002] J. Lin, Y. Wu and T.S. Huang. *Capturing human hand motion in image sequences.* In Workshop on Motion and Video Computing, pages 99–104, 2002. 22, 28, 31, 32, 33, 40, 42, 54

[Lin 2004] J.Y. Lin, Y. Wu and T.S. Huang. *3D model-based hand tracking using stochastic direct search method.* In AFGR, pages 693–698, 2004. 20, 27, 28, 29, 31, 32, 33, 38, 39, 54

[Lischinski 1992] D. Lischinski, F. Tampieri and D.P. Greenberg. *A Discontinuity Meshing Algorithm for Accurate Radiosity*. Computer Graphics and Applications, vol. 12, pages 25–39, 1992. 154

[Liu 2004] Y. Liu and Y. Jia. *A Robust Hand Tracking and Gesture Recognition Method for Wearable Visual Interfaces and Its Applications*. In International Conference on Image and Graphics, pages 472–475, 2004. 19

[Liu 2007] T-W. Liu and D-H. Li. *A practical update criterion for SQP method*. Optimization Methods Software, vol. 22, no. 2, pages 253–266, 2007. 110

[Lu 2003] S. Lu, D. Metaxas, D. Samaras and J. Oliensis. *Using Multiple Cues for Hand Tracking and Model Refinement*. In CVPR, volume 2, pages 443–450, 2003. 20, 28, 30, 31, 34, 41, 50, 128, 129, 146

[Magnenat-Thalmann 1988] N. Magnenat-Thalmann, R. Laperrière and D. Thalmann. *Joint-dependent local deformations for hand animation and object grasping*. In Graphics Interface, pages 26–33, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society. 132

[Mairson 1988] H.G. Mairson and J. Stolfi. *Reporting and counting inter-sections between two sets of line segments*. Theoretical Foundations of Computer Graphics and CAD, pages pp. 307–325, 1988. 77, 81

[Malassiotis 2008] S. Malassiotis and M. G. Strintzis. *Real-time hand posture recognition using range data*. Image Vision Comput., vol. 26, no. 7, pages 1027–1037, 2008. 22

[Markosian 1997] L. Markosian, M.A. Kowalski, D. Goldstein, S.J. Trychin, J.F. Hughes and L.D. Bourdev. *Real-time nonphotorealistic rendering*. In SIGGRAPH, pages 415–420, New York, NY, USA, 1997. 154, 155

[McCool 1995] M.D. McCool. *Analytic antialiasing with prism splines*. In SIGGRAPH, pages 429–436, New York, NY, USA, 1995. ACM. 91, 97, 156

[McIlroy 1992] M.D. McIlroy. *Getting raster ellipses right*. ACM Trans. Graph., vol. 11, no. 3, pages 259–275, 1992. 98

[Mitobe 2007] K. Mitobe, J. Sato, T. Kaiga, T. Yukawa, T. Miura, H. Tamamoto and N. Yoshimura. *Development of a high precision hand motion capture system and an auto calibration method for a hand skeleton model*. In SIGGRAPH, page 159, New York, NY, USA, 2007. ACM. 6

[Mo 2006] Z. Mo and U. Neumann. *Real-time Hand Pose Recognition Using Low-Resolution Depth Images*. In CVPR, pages 1499–1505, Washington, DC, USA, 2006. IEEE Computer Society. 22

[Moeslund 2001] T.B. Moeslund and E. Granum. *A survey of computer vision-based human motion capture*. CVIU, vol. 81, no. 3, pages 231–268, 2001. 49

[Moeslund 2003] T.B. Moeslund and L. Nørgaard. *A Brief Overview of Hand Gestures Used in Wearable Human Computer Interfaces*. Rapport technique, Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark, 2003. 10

[Moeslund 2006] T.B. Moeslund, A. Hilton and V. Krüger. *A survey of advances in vision-based human motion capture and analysis*. CVIU, vol. 104, no. 2, pages 90–126, 2006. 49

[Mohr 2003] A. Mohr and M. Gleicher. *Building efficient, accurate character skins from examples*. ACM Trans. Graph., vol. 22, no. 3, pages 562–568, 2003. 133

[Murakami 1991] K. Murakami and H. Taguchi. *Gesture recognition using recurrent neural networks*. In SIGCHI Conference on Human factors in Computing Systems, pages 237–242, New York, NY, USA, 1991. ACM. 9

[Nakamae 1972] E. Nakamae and T. Nishita. *An Algorithm for Hidden Line Elimination of Polyhedra*. Information Processing in Japan, vol. 12, pages 134–141, 1972. 154

[Nikolova 2007] M. Nikolova and R.H. Chan. *The Equivalence of Half-Quadratic Minimization and the Gradient Linearization Iteration*. IP, vol. 16, no. 6, pages 1623–1627, 2007. 170

[Nirei 1996] K. Nirei, H. Saito, M. Mochimaru and S. Ozawa. *Human Hand Tracking from Binocular Image Sequences*. In International Conference on Industrial Electronics, Control, and Instrumentation, pages 297 – 302, 1996. 21, 28, 33, 34, 42

[Nishita 1974] T. Nishita and E. Nakamae. *An algorithm for Half-Toned Representation of three Dimensional objects*. Information Processing in Japan, vol. 14, pages 93–99, 1974. 154

[Nitzberg 1990] M. Nitzberg and D. Mumford. *The 2.1-D Sketch*. In ICCV, pages 138–144, 1990. 84

[Nölker 1998] C. Nölker and H. Ritter. *Detection of Fingertips in Human Hand Movement Sequences*. In International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction, pages 209–218, 1998. 30, 34

[Nölker 1999] C. Nölker and H. Ritter. *GREFIT: Visual Recognition of Hand Postures*. In International Gesture Workshop, page 61, 1999. 30, 34, 43, 46, 47

[Nölker 2000] C. Nölker. *Parametrized SOMs for Hand Posture Reconstruction.* In International Joint Conference on Neural Networks, volume 4, pages 139–144, 2000. 43, 47

[Nölker 2002] H. Nölker C. Ritter. *Visual recognition of continuous hand postures.* In Neural Networks, pages 2983– 994, 2002. 43, 46, 47

[Ogawara 2003] K. Ogawara, J.Takamatsu, K. Hashimoto and K. Ikeuchi. *Grasp recognition using a 3D articulated model and infrared images.* In Intelligent Robots and Systems, volume 2, pages 1590 – 1595, 2003. 20, 21, 29, 30, 37, 130, 132

[Ong 2000] C-J. Ong, E. Huang and S-M. Hong. *A fast growth distance algorithm for incremental motions.* IEEE Transactions on Robotics, vol. 16, no. 6, pages 880–890, 2000. 184

[O'Rourke 1982a] J. O'Rourke, C-B. Chien, T. Olson and D. Naddor. *A New Linear Algorithm for Intersecting Convex Polygons.* CGIP, vol. 19, no. 4, pages 384–391, 1982. 82

[O'Rourke 1982b] J. O'Rourke, C-B. Chien, T. Olson and D. Naddor. *A new linear algorithm for intersecting convex polygons.* Comput. Graph. Image Process., vol. 19, pages 384–391, 1982. 82

[O'Rourke 1999] J. O'Rourke. Computational geometry in C, second edition, cup, cambridge, uk, 1998, 376 pages, volume 17. Cambridge University Press, New York, NY, USA, 1999. 83

[Ouhaddi 1999a] H. Ouhaddi and P. Horain. *3D hand gesture tracking by model registration.* In International Workshop on Synthetic-Natural Hybrid Coding and 3D Imaging, pages 70–73, 1999. 20, 22, 28, 29, 31, 32, 33, 38, 39, 54

[Ouhaddi 1999b] H. Ouhaddi and P. Horain. *Hand tracking by 3D model registration : a comparison of optimisation methods.* In International Scientific Workshop on Virtual Reality and Prototyping, pages 51–59, 1999. 84

[Paragios 1999] N. Paragios and R. Deriche. *Geodesic Active Regions for Supervised Texture Segmentation.* In ICCV, volume 2, pages 926–932, 1999. 152

[Pavlovic 2000] V. Pavlovic, J.M. Rehg and J. MacCormick. *Learning Switching Linear Models of Human Motion.* In NIPS, pages 981–987, 2000. 146

[Pitteway 1980] M.L.V. Pitteway and D.J. Watkinson. *Bresenham's algorithm with Grey scale.* Commun. ACM, vol. 23, no. 11, pages 625–626, 1980. 96

[Plänkers 2001] R. Plänkers and P. Fua. *Tracking and Modeling People in Video Sequences.* CVIU, vol. 81, no. 3, 2001. 129

[Potamias 2008] M. Potamias and V. Athitsos. *Nearest neighbor search methods for handshape recognition.* In International conference on Pervasive Technologies Related to Assistive Environments, pages 1–8, 2008. 20, 43, 45

[Pottmann 2006] H. Pottmann, Q-X. Huang, Y-L. Yang and S-M. Hu. *Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes.* IJCV, vol. 67, no. 3, pages 277–296, 2006. 40

[Powell 1978] M.J.D. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, 1978. 109

[Preparata 1985] F.P Preparata and M.I Shamos. Computational geometry. Springer-Verlag - New York, 1985. 81

[Rehg 1993] J. Rehg and T. Kanade. *DigitEyes: Vision-Based Human Hand Tracking.* Rapport technique CMU-CS-93-220, Computer Science Department, Carnegie Mellon Universit, Pittsburgh, PA, December 1993. 30, 35

[Rehg 1994a] J. Rehg and T. Kanade. *Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking.* In ECCV, volume 2, pages 35–46, 1994. 26

[Rehg 1994b] J. Rehg and T. Kanade. *Visual Tracking of Self-Occluding Articulated Objects.* Rapport technique CMU-CS-94-224, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1994. 31, 39

[Rehg 1995] J. Rehg and T. Kanade. *Model-Based Tracking of Self-Occluding Articulated Objects.* In ICCV, pages 612–617, 1995. 31, 39

[Roberts 1963] L.G. Roberts. Machine perception of three-dimensional solids. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1963. 154

[Rosales 2001] R. Rosales, V. Athitsos, L. Sigal and S. Scarloff. *3D Hand Pose Reconstruction Using Specialized Mappings.* In ICCV, volume 1, pages 378–385, 2001. 20, 43, 47, 171

[Sander 2001] P.V. Sander, H. Hoppe, J. Snyder and S.J. Gortler. *Discontinuity edge overdraw.* In Symposium on Interactive 3D graphics, pages 167–174, New York, NY, USA, 2001. ACM Press. 158, 159

[Schlattmann 2007] M. Schlattmann, F. Kahlesz, R. Sarlette and R. Klein. *Markerless 4 gestures 6 DOF real-time visual tracking of the human hand with automatic initialization.* Computer Graphics Forum, vol. 26, no. 3, pages 467–476, 2007. 21, 37

[Schmidt 2006] R. Schmidt, T. Isenberg and B. Wyvill. *Interactive pen-and-ink rendering for implicit surfaces.* In SIGGRAPH, 2006. 129

[Schneider 2002] D.H. Schneider P.J. Eberly. Geometric tools for computer graphics. Elsevier Science Inc., New York, NY, USA, 2002. 80

[Schölkopf 1998] B. Schölkopf, A. Smola and K-R. Müller. *Nonlinear component analysis as a kernel eigenvalue problem.* Neural Comput., vol. 10, no. 5, pages 1299–1319, 1998. 147, 200

[Segen 1999] J. Segen and S. Kumar. *Shadow Gestures: 3D Hand Pose Estimation Using a Single Camera.* In CVPR, volume 1, pages 479–485, 1999. 22, 35

[Shanno 1970] D.F. Shanno. *Conditioning of Quasi-Newton Methods for Function Minimization.* Mathematics of Computation, vol. 24, pages 647–656, 1970. 109

[Sheynin 2001] S.A. Sheynin and A.V. Tuzikov. *Explicit formulae for polyhedra moments.* PRL, vol. 22, no. 10, pages 1103–1109, 2001. 186

[Shimada 1998] N. Shimada, Y. Shirai, Y. Kuno and J. Miura. *Hand Gesture Estimation and Model Refinement Using Monocular Camera - Ambiguity Limitation by Inequality Constraints.* In AFGR, page 268, Washington, DC, USA, 1998. IEEE Computer Society. 30, 35

[Shimada 2001] N. Shimada. *Real-time 3-D Hand Posture Estimation based on 2-D Appearance Retrieval Using Monocular Camera.* In RATFG, page 23, 2001. 20, 43, 44, 48

[Shu 2001] H.Z. Shu, L.M. Luo, W.X. Yu and J.D. Zhou. *Fast computation of Legendre moments of polyhedra.* PR, vol. 34, no. 5, pages 1119–1126, 2001. 186

[Sidenbladh 2000] H. Sidenbladh, M.J. Black and D.J. Fleet. *Stochastic Tracking of 3D Human Figures Using 2D Image Motion.* In ECCV, volume 2, pages 702–718, 2000. 139

[Sigal 2007] L. Sigal. *Continuous-state Graphical Models for Object Localization, Pose Estimation and Tracking.* Phd. manuscript, Brown University, September 2007. 31

[Singer 1993] M.H. Singer. *A General Approach to Moment Calculation for Polygons and Line Segments.* PR, vol. 26, no. 7, pages 1019–1028, 1993. 97, 156

[Slabaugh 2005] G. Slabaugh and G. Unal. *Active Polyhedron: Surface Evolution Theory Applied to Deformable Meshes.* In CVPR, pages 84–91, Washington, DC, USA, 2005. IEEE Computer Society. 186

[Sloan 2001] P-P. J. Sloan, C.F. Rose and M. F. Cohen. *Shape by example.* In Symposium on Interactive 3D graphics, pages 135–143, New York, NY, USA, 2001. ACM. 28

[Sminchisescu 2004] C. Sminchisescu and A. Jepson. *Generative modeling for continuous non-linearly embedded visual inference.* In ICML, page 96, 2004. 146, 147

[Soucy 1996] M. Soucy, G. Godin and M. Rioux. *A texture-mapping approach for the compression of colored 3D triangulations.* The Visual Computer, vol. 12, no. 10, pages 503–514, 1996. 135

[Starner 1998] T. Starner, A. Pentland and J. Weaver. *Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video.* PAMI, vol. 20, no. 12, pages 1371–1375, 1998. 19

[Stauffer 1999] C. Stauffer and W. Grimson. *Adaptive Background Mixture Models for Real-time Tracking.* In CVPR, volume 2, pages 246–252, 1999. 84

[Steger 1996] C. Steger. *On the Calculation of Moments of Polygons.* Rapport technique FGBV–96–04, Forschungsgruppe Bildverstehen (FG BV), Informatik IX, Technische Universität München, August 1996. 97, 156

[Stenger 2001a] B. Stenger, P.R.S. Mendonça and R. Cipolla. *Model Based 3D Tracking of an Articulated Hand.* In CVPR, volume 2, pages 310–315, 2001. 28, 31

[Stenger 2001b] B. Stenger, P.R.S. Mendonça and R. Cipolla. *Model-Based Hand Tracking Using an Unscented Kalman Filter.* In BMVC, volume 1, pages 63–72, 2001. 20, 28

[Stenger 2001c] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee and J.M. Buhmann. *Topology Free Hidden Markov Models: Application to Background Modeling.* In ICCV, volume 1, pages 294–301, 2001. 20, 26, 28

[Stenger 2003] B. Stenger, A. Thayananthan, P.H.S. Torr and R. Cipolla. *Filtering Using a Tree-Based Estimator.* In ICCV, volume 2, pages 1063–1070, 2003. 28, 121, 122, 123

[Stenger 2004a] B. Stenger. *Model-Based Hand Tracking Using A Hierarchical Bayesian Filter.* PhD thesis, University of Cambridge, 2004. 28, 31, 33, 54, 146, 173

[Stenger 2004b] B. Stenger, A. Thayananthan, P.H.S. Torr and R. Cipolla. *Hand Pose Estimation Using Hierarchical Detection.* In Intl. Workshop on Human-Computer Interaction, pages 105–116, 2004. 28, 31, 33, 54

[Stenger 2006] B. Stenger, A. Thayananthan, P.H.S Torr and R. Cipolla. *Model-Based Hand Tracking Using a Hierarchical Bayesian Filter.* PAMI, vol. 28, no. 9, pages 1372–1384, 2006. 29, 31, 32, 33, 42, 43, 45, 47, 54

[Stewart 1994] A.J. Stewart and S. Ghali. *Fast computation of shadow boundaries using spatial coherence and backprojections.* In SIGGRAPH, pages 231–238, 1994. 154

[Strachan 1990] N.J.C. Strachan, P. Nesvadba and A.R. Allen. *A Method for Working out the Moments of a Polygon.* PRL, vol. 11, pages 351–354, 1990. 97, 156

[Sturman 1992] D.J. Sturman. *Whole-hand input.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1992. 11, 12

[Sturman 1994] D.J. Sturman and D. Zeltzer. *A Survey of Glove-based Input.* Computer Graphics and Applications, vol. 14, no. 1, pages 30–39, 1994. 11

[Sudderth 2004] E. Sudderth, M. Mandel, W. Freeman and A. Willsky. *Visual Hand Tracking Using Nonparametric Belief Propagation.* In CVPR, page 189, 2004. 20, 26, 31, 33, 42, 54, 60, 121, 122, 185

[Sutherland 1974] I.E. Sutherland and G.W. Hodgman. *Reentrant polygon clipping.* Commun. ACM, vol. 17, no. 1, pages 32–42, 1974. 95

[Svoboda 2005] T. Svoboda, D. Martinec and T. Pajdla. *A convenient multicamera self-calibration for virtual environments.* Presence: Teleoper. Virtual Environ., vol. 14, no. 4, pages 407–422, 2005. 21

[Takahashi 1991] T. Takahashi and F. Kishino. *Hand gesture coding based on experiments using a hand gesture interface device.* SIGCHI Bull., vol. 23, no. 2, pages 67–74, 1991. 9

[Thayananthan 2003a] A. Thayananthan, B. Stenger, P.H.S. Torr and R. Cipolla. *Learning a Kinematic Prior for Tree-Based Filtering.* In BMVC, volume 2, pages 589–598, 2003. 11, 27, 28, 31, 42, 43, 146

[Thayananthan 2003b] A. Thayananthan, B. Stenger, P.H.S. Torr and R. Cipolla. *Shape Context and chamfer matching in cluttered scenes.* In CVPR, volume 1, pages 127–133, 2003. 185

[Thirion 1996] J-P. Thirion. *Non-Rigid Matching Using Demons.* In CVPR, page 245, Washington, DC, USA, 1996. IEEE Computer Society. 41

[Tipping 2001] M.E. Tipping. *Sparse bayesian learning and the relevance vector machine.* J. Mach. Learn. Res., vol. 1, pages 211–244, 2001. 47

[Toussaint 1985] G. Toussaint. *A simple linear algorithm for intersecting convex polygons.* The Visual Computer, vol. 1, no. 2, pages 118–123, 1985. 82

[Tran 2008] C. Tran and M.M. Trivedi. *Hand modeling and tracking from voxel data: An integrated framework with automatic initialization.* In ICPR, pages 1–4, 2008. 21, 41

[Tumblin 2006] J. Tumblin. *Exact Two-Dimensional Integration inside Quadrilateral Boundaries.* Journal of Graphics, GPU and Game Tools, vol. 11, no. 1, pages 61–71, 2006. 97, 156

[Ueda 2001a] M. Ueda, M. Imai and T. Ogasawara. *Hand Pose Estimation for Vision-based Human Interface.* In Workshop on Robot and Human Commnunication, pages 473–478, 2001. 37

[Ueda 2001b] M. Ueda, M. Imai and T. Ogasawara. *Hand Pose Estimation Using Multi-Viewpoint Silhouette Images.* In International Conference on Intelligent Robots and Systems, pages 1989–1996, 2001. 41

[Ueda 2003] M. Ueda, M. Imai and T. Ogasawara. *Hand Pose Estimation for Vision-based Human Interface.* IEEE Transactions on Industrial Electronics, vol. 50, pages 676–684, 2003. 37

[Ueda 2005] E. Ueda, Y. Matsumoto and T. Ogasawara. *Virtual Clay Modeling System Using Multi-Viewpoint Images.* In International Conference on 3-D Digital Imaging and Modeling, pages 134–141, 2005. 20, 21, 26, 29, 30, 37

[Unal 2005] G. Unal, A. Yezzi and H. Krim. *Information-Theoretic Active Polygons for Unsupervised Texture Segmentation.* IJCV, vol. 62, no. 3, pages 199–220, 2005. 54, 103, 153

[Urtasun 2006] R. Urtasun, D.J. Fleet and P. Fua. *3D People Tracking with Gaussian Process Dynamical Models.* In CVPR, volume 1, pages 238–245, 2006. 146

[Usabiaga 2006] J. Usabiaga, A. Erol, G.N. Bebis, R. Boyle and X. Twombly. *Global Hand Pose Estimation by Multiple Camera Ellipse Tracking.* In ISVC, volume 1, pages 122–132, 2006. 22

[Žalik 2000] B. Žalik. *Two efficient algorithms for determining intersection points between simple polygons.* Comput. Geosci., vol. 26, no. 2, pages 137–151, 2000. 82

[Wang 2001] L. Wang, S.B. Kang, R. Szeliski and H.Y. Shum. *Optimal Texture Map Reconstruction from Multiple Views.* In CVPR, pages 347–354, 2001. 165

[Wang 2003] Q. Wang, G.Y. Xu and H.Z. Ai. *Learning object intrinsic structure for robust visual tracking.* In CVPR, volume 2, pages 227–233, 2003. 146, 147

[Wang 2008] J. Wang, V. Athitsos, S. Sclaroff and M. Betke. *Detecting Objects of Variable Shape Structure With Hidden State Shape Models.* PAMI, vol. 30, no. 3, pages 477–492, 2008. 185

[Wang 2009] C. Wang, M. de La Gorce and N. Paragios. *Segmentation, Ordering and Multi-Object Tracking using Graphical Models.* In ICCV, 2009. 185

[Weiler 1977] K. Weiler and P. Atherton. *Hidden surface removal using polygon area sorting.* In SIGGRAPH, volume 11, pages 214–222, 1977. 154

[Weingaertner 1997] T. Weingaertner, S. Hassfeld and R. Dillmann. *Human Motion Analysis: A Review.* Workshop on Motion of Non-Rigid and Articulated Objects, page 90, 1997. 49

[Wilke 2009] D. Wilke, S. Kok and A. Groenwold. *The application of gradient-only optimization methods for problems discretized using non-constant methods.* Structural and Multidisciplinary Optimization, 2009. 117, 157

[Wu 1991] X. Wu. *An efficient antialiasing technique.* In SIGGRAPH, volume 25, pages 143–152, 1991. 96

[Wu 1999a] Y. Wu and T.S. Huang. *Capturing Articulated Human Hand Motion: A Divide-and-Conquer Approach.* In ICCV, pages 606–611, 1999. 20

[Wu 1999b] Y. Wu and T.S. Huang. *Human hand modeling, analysis and animation in the context of human computer interaction.* IEEE Signal Processing Magazine, Special issue on Immersive Interactive Technology, pages 6–10, 1999. 9

[Wu 1999c] Y. Wu and T.S. Huang. *Vision-Based Gesture Recognition: A Review.* In International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction, pages 103–115, 1999. 9

[Wu 2001] Y. Wu, J.Y. Lin and T.S. Huang. *Capturing Natural Hand Articulation.* In ICCV, pages 426–432, 2001. 11, 20, 22, 27, 28, 29, 31, 32, 33, 42, 54, 146, 147

[Xin 2007] X. Xin, H. Ma and C. Vogel. *Motion Capture as a User Research Tool in Dynamic Ergonomics.* International Association of Societies of Design Research, 2007. 8

[Yang 2006] C. Yang, P. Shi, W. Zao, L. Wang, X. Meng and J. Wang. *New Intersection Algorithm of Convex Polygons Based on Voronoi Diagrams.* In International Symposium on Voronoi Diagrams in Science and Engineering, pages 224–231, 2006. 82

[Zhou 2003] H. Zhou and T.S. Huang. *Tracking Articulated Hand Motion with Eigen Dynamics Analysis.* In ICCV, page 1102, 2003. 11, 12, 20, 27, 28, 33, 40, 146, 147

[Zhou 2004] H. Zhou, D.J. Lin and T.S. Huang. *Static Hand Gesture Recognition based on Local Orientation Histogram Feature Distribution Model.* In CVPR Workshop, volume 10, page 161, 2004. 43, 45, 46

[Zhou 2005a] H.N. Zhou and T.S. Huang. *Okapi Chamfer Matching for Articulated Object Recognition.* In ICCV, volume 2, pages 1026–1033, 2005. 20, 22, 46

[Zhou 2005b] K. Zhou, X. Wang, Y. Tong, M. Desbrun, B. Guo and H-Y. Shum. *TextureMontage: Seamless Texturing of Arbitrary Surfaces From Multiple Images.* SIGGRAPH, pages 1148–1155, 2005. 165

[Zhu 2005] Y-K. Zhu, J-H. Yong and G-Q. Zheng. *Line Segment Intersection Testing.* Computing, vol. 75, no. 4, pages 337–357, 2005. 82

[Ziaie 2009] P. Ziaie, T. Müller, M.E. Foster and A. Knoll. *A Naïve Bayes Classifier with Distance Weighting for Hand-Gesture Recognition.* Advances in Computer Science and Engineering, vol. 6, pages 308–315, 2009. 43