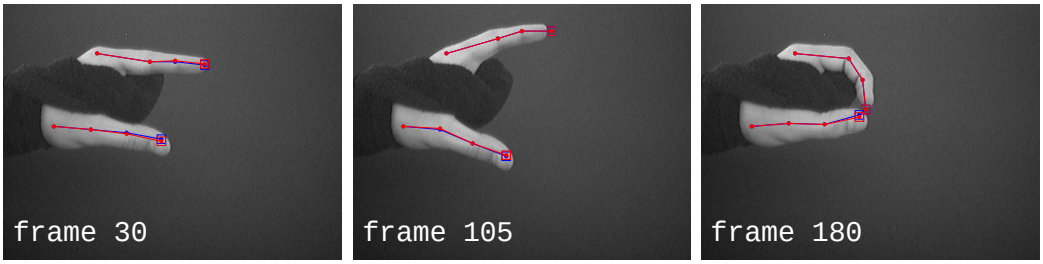
UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO



VISUAL TRACKING OF ARTICULATED OBJECTS: AN APPLICATION TO THE HUMAN HAND

RICARDO ANDRÉ DA SILVA MARRANITA

(Licenciado)

Dissertação para obtenção de Grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador:	Doutor José Alberto Rosado dos Santos Victor
Júri:	
Presidente:	Doutor José Alberto Rosado dos Santos Victor
Vogais:	Doutor Arnaldo Joaquim de Castro Abrantes
	Doutor Alexandre José Malheiro Bernardino

Julho de 2005

Agradecimentos

Aos meus pais e irmão pelo apoio e constante motivação.

Ao professor João Paulo Costeira pela sugestão de como abordar o problema.

Ao professor Alexandre Bernardino pelas discussões proporcionadas e por me ajudar a perceber um pouco melhor os filtros de Kalman.

Ao professor José Santos Victor pela sua orientação e em particular pelas discussões proporcionadas, pelas ideias e por incutir algum método no meio do caos.

Aos que me emprestaram livros, alguns deles por mais tempo do que desejariam.

A todos os que directa ou indirectamente, pelas suas acções, me tornaram mais forte, persistente e determinado em chegar ao fim.

Acknowledgements

To my parents and brother for their support and constant motivation.

To professor João Paulo Costeira for his suggestion on how to approach the problem.

To professor Alexandre Bernardino for the discussions and for helping me understand a bit more of Kalman filters.

To professor José Santos Victor for his supervision and in particular for the discussions, the ideas and for shedding some method upon chaos.

To those who lent me books, some of them longer than they had wished.

To all those who directly or indirectly, for their actions, made me stronger, persistent and determined to reach the end.

Resumo

O seguimento de objectos articulados é importante em áreas como a Medicina, Robótica ou Computação Gráfica. Uma aplicação concreta é o seguimento de movimento humano.

Esta tese propõe um método baseado em visão para o seguimento de movimentos de dedos numa sequência de imagens. O objectivo é determinar o valor dos ângulos nas articulações dos dedos.

A abordagem compreende vários passos. Um modelo cinemático da mão é usado para descrever algumas das limitações físicas dos movimentos dos dedos. Primeiro, o modelo é usado para guiar a extracção de medidas locais de imagem correspondentes a partes do dedo. Esta análise de imagem pode ser feita de quatro maneiras diferentes, onde a melhor está concebida para lidar com oclusões nos dedos. Um algoritmo de estimação de pose ajusta o modelo cinemático às observações extraídas da imagem. Finalmente, usa-se integração temporal para reduzir a incerteza de estimação e prever o aspecto da mão na próxima imagem.

Testes efectuados aos diferentes passos mostram o desempenho e limitações destes. Os resultados mostram que o seguimento tem sucesso em imagens reais e sintéticas usando as duas técnicas de melhor desempenho. Estes testes abrangem desde modelos simplificados a completos da mão.

No fim discutem-se as conclusões e direcções de trabalho futuro.

Palavras-chave: Seguimento de mãos, oclusão, modelo cinemático, seguimento não invasivo, sequência de imagens, filtro de Kalman.

Abstract

Tracking articulated objects is important in areas such has Medicine, Robotics or Computer Graphics. A particular application is tracking human movement.

This thesis proposes a vision-based method to track finger movements on a sequence of images. The goal is to determine the value of the angles in the articulations of the fingers.

The approach comprises several steps. A kinematic model of the hand is used to describe some of the physical limitations of finger movements. The model is first used to guide the extraction of local image measurements corresponding to the finger parts. This image analysis can be done in four different ways, where the best one is designed to handle finger occlusion. A pose estimation algorithm adjusts the kinematic model to the observations retrieved from the image. Finally, temporal integration is used to reduce estimation uncertainty and predict the hand appearance in the next image frame.

Tests performed on different steps show their performance and limitations. Results show successful tracking on real and synthetic images using the two best performing techniques. These tests range from simplified to complete hand models.

Conclusions and further direction of work are described in the end.

Keywords: Hand tracking, occlusion, kinematic model, non-invasive tracking, image sequence, Kalman filter.

Contents

Figure List

Index of Tables

1 Introduction

The goal of this thesis is to present a method which allows estimating the pose of an articulated object over time, by analysis of a sequence of images. The pose is characterized by a set of degrees of freedom (DOF) which generally corresponds to joint angles and displacements.

The proposed method is designed to work in a very specific context of articulated objects: the human hands. Thus, the idea is to take a sequence of images with moving fingers and determine their position and orientation. The pose estimation problem is to determine the angles in the articulations of the fingers.

Tracking articulated objects has multiple applications, which can simply concern checking if the articulated object is working as desired or capture its movement as to mimic it somewhere else or learn from it.

In the context of tracking articulated objects there is a field which dedicates to studying the motion in animals. More specifically, this study can concern the ranges of movement of the human body. Even more specifically, that study may not focus on the whole body, but just a region, such as the human hand.

Tracking the movement of a human hand can have several degrees of complexity, depending on the desired application, the means available, the context in which tracking will take place. Some of the more basic and reliable ways of determining the movement of a human hand rely on invasive techniques, such as the use of sensory gloves. However, the use of gloves may prove superfluous if one is only interested in tracking the position of the hand in space, rather than knowing the angles in the articulations of the hand and fingers. The use of gloves is also not a good choice when dealing with agents who may not feel comfortable with them, such has human babies and simians, which may try to remove the glove.

When dealing with such agents while focusing on the aspects of tracking the

movement of the hand, including its position, orientation, and those of its fingers, one has to consider using non-invasive techniques. The answer often is using cameras to capture the movement and then process the images to determine the desired information.

But even within this scope there are many ways of determining the configuration of the tracked hand, some of which are presented shortly ahead for reference.

1.1 Previous Work

In [1], Rehg and Kanade proceed to the analysis of a sequence of monocular images with the purpose of estimating the DOF of the finger joints and wrist. The goal is to use the hand as a substitute for a computer mouse, and at the same time provide this new pointer with new abilities, such as a third displacement direction. They use a kinematic model of the hand to help searching for features in the image. The features correspond to lines aligned with the phalanges and to points indicating the location of the finger tips. The image is sampled perpendicularly to the projection of the kinematic model on camera, where the model is configured with a predicted pose. These samples are then used to search for edge points, which will then be used to find the line and point features.

This conception however, didn't enable tracking in the presence of occlusions, so in [2], Rehg and Kanade improved this procedure. They stop using perpendicular sampling and edge search to start using templates describing the appearance of the phalanges on image. To enable tracking with occlusions, they present a method based on the kinematic model which describes which finger phalanges are occluding the other ones. They use this estimate of occlusion to mask the templates to form representations of the occluded phalanges.

Both in [1] and [2], the pose is estimated by aligning the projection of the kinematic model with the image data. This is done using a gradient descent technique to minimize a cost function which accounts for the differences between the image data and the features in question.

In [3], Cham and Rehg introduce 2D scaled prismatic models (SPM) to perform figure registration. They also resort to a bank of filters to provide multiple state predictions, in order to enable multiple-hypothesis tracking (MHT). The goal is to be

able to track animated figures in an image sequence, where these figures present some problematic self occlusions, which could not be solved using a single-hypothesis scheme.

The same principle of SPM and MHT is used in [4] by Rehg to perform the same type of figure tracking. The novelty is the possibility of using that knowledge to perform background reconstruction and mosaicing, or even introduce the animated agent in a different background.

A volumetric model of the human body is used by Sminchisescu and Triggs in [5] as a way of generating intensity and edge features in the image frame, which are then compared with the features of the image itself to determine the state. Multiple hypothesis is used to add robustness to the tracking process.

In [6], Delamarre presents a way of using a volumetric representation of the hand and active contour techniques to locate a hand in an image and determine its pose. The author shows that the pose estimates can be found with no need of having an initial prediction of the model projected near the object in the image (as in the above documents). He also suggests using stereo vision and disparity maps as a way of improving this method.

The concept of stereo vision and disparity maps is further explored by Delamarre and Faugeras in [7] as a way of retrieving the hand pose, even with occlusions, resorting to physical attraction forces to attract a 3D volumetric model towards the disparity map.

In [8] and [9], Athitsos and Sclaroff present a way of estimating the hand pose based on finger detectors and a database with trained hand configurations.

In [10], Sidenbladh and Black develop a method to learn image statistics to use in Bayesian tracking. Image features used include optical flow, edges and ridges. The authors show how the combined use of the three statistics is much more robust than using a single statistic. They also show successful tracking in the presence of occlusion.

In [11] and [12], Li *et al* implement a 3D mouse using monocular images of a hand. However, they resort to a mirror so the monocular image does in fact present two points of view of the tracked hand. They show how their mouse can be used for 3D positioning and attribute actions given the states of the hand, which are based on the

contour of the hand rather than in the determination of the joint values. They try to model the background, which generally presents a lot of clutter, as a way of better determining the contour.

In [13], Houhaddi and Horain propose a method for hand tracking based on image features. The goal is to adjust a model of the hand to features regarding the sides of the fingers, which are found after a segmentation and image analysis process. The authors show the results when evaluating a hand facing the camera and spreading its fingers. These concepts are also used in [14], which shows some additional results from the same point of view but with the fingers being bent independently.

In [15], Shimada and Shirai propose a method to precisely estimate the shape and pose of articulated objects like a human hand. Their initial rough estimates are based on silhouette matching and are then fitted to an image using an extended Kalman filter. The problem of ambiguities in shape and pose due to the fact of monocular images possessing no depth information is overcome by modifying the filter solution. The estimates are achieved by incrementally reducing the possible solution space with informative observations. The authors show rough estimation results based on the silhouette method and show advantages in using their modification of the filter solution when tracking a finger in a side-view image sequence

The method proposed by Nirei *et al* in [16] uses binocular images of a hand. Tracking is performed by minimizing estimation error of an optical flow and maximizing the overlap between a projected model and a silhouette image. The problems are solved using stochastic optimization. They present results using synthetic images as a base of comparison between the estimates and the ground truth and also present results using real images.

Regarding full body tracking, in [17] Fua *et al* develop a framework for 3D shape and motion recovery of articulated deformable objects. They incorporate implicit surfaces into earlier robotics approaches designed to handle articulated structures. They demonstrate its effectiveness for human body modelling from video sequences, with a method both robust and generic since it could easily be applied to other shape and motion recovery problems. This line of work is also developed by Plankers and Fua in [18], showing some additional results.

1.2 Approach

Independently of the particular choices in the approach, the previous procedures present a common basis. They all resort to sequences of images as a source of data. These images are analyzed to retrieve information which can be used to determine the pose. Those which deal with aspects of pose determination generally employ a model of the object to track. This model can be used, for example, to implement the physical restrictions of the object or to use it as a way of generating data on image. Then, there generally is an algorithm responsible for configurating the model accordingly to the retrieved data, as to determine the pose of the object in the image. As an extra, some works employ filters to add robustness to the determination process over time.

The proposed method is a compromise between [1] and [2]. The approach to this problem consists of the following key blocks:

- articulated model;
- pose estimation;
- image analysis;
- temporal integration (optional).

The articulated model describes the articulations and connections of the tracked object: the hand and its fingers. It allows the geometric configuration of the object to be known in function of the angles in the articulations. It is used to restrict the image analysis search space.

The pose estimation block deals with the aspect of determining the values of the articulations angles which align the articulated model with the object in the image. Hence, it needs to know the articulated model and how to relate it to the image. The estimates are found with a recursive method which tries to determine the angles which minimize the distances between model and image data.

Image analysis is the block which evaluates an image for features which can be used by the pose estimation block. These image features are extracted from the image in specific locations resorting to image samplers. These image samplers are positioned on the image accordingly to an estimate of the pose. This estimate is the geometric configuration of the model given an estimate of the articulations angles.

Temporal integration is an optional block since it is used as a way of statistically providing estimates for the initial state of the articulated model at each new image. It is also used to statistically improve the pose estimates determined by the pose estimation process, which has some associated estimation noise. Finally, it is a way of statistically limiting wide variations in the angles over time, which could lead to loss of tracking or a high uncertainty in the estimated pose.

1.3 Choices and Assumptions

In order so simplify the estimation process some choices and assumptions are made for the several key blocks. In general, they are only made as a way of simplifying the blocks and not as a way of limiting the tracking process. This means that the blocks can be easily upgraded to encompass more complex situations.

1.3.1 Articulated Model

In the articulated models used, a finger is usually modelled as a sequence of three bodies (the phalanges) and three joints (the articulations). The articulations only allow the model to mimic the bending movement (flexion/extension) of a finger but don't allow a side-to-side movement (adduction/abduction). Bear in mind though, that this is no limitation to any of the blocks, it is only a way of simplifying their design.

The geometric configuration of the model is projected on the image, to allow positioning of the samplers and visualization of the tracked pose. The projection process is included in the articulated model, since all measurements will be made in the image frame. To avoid camera calibration, the axes of the image frame are considered to be aligned and in the same units as the root frame of the articulated model. Furthermore, the perspective camera model is used throughout the thesis. These two simplifications allow the projection matrix to be very simple, which enables focusing on the real issue (tracking) without concerning too much with camera calibration aspects.

Still regarding camera aspects, the whole thesis is built around sequences of monocular images. The concept though, is very easily extendable to binocular images or even more complex situations. It's basically a matter of repeating the same

procedures to all cameras and account for all errors at the same time.

1.3.2 Pose Estimation

The pose estimation process is based on a recursive method which tries to minimize a cost function by iteratively updating the articulations angles. Whenever possible the basic implementation of the Gauss-Newton (GN) algorithm is used. When facing a more complex hand model, which generally leads to a cost function with many peaks and high amplitudes, a more robust implementation of the GN algorithm is used in order to restrict the updates in each recursive step.

Since a single camera is used (monocular images), the cost function only accounts for the differences between model data and image data in that single camera. Should more cameras be in use, the cost function should account for all differences at the same time.

1.3.3 Image Analysis

This is a very complex block in the proposed tracking method. As a simplification to image sampling and further evaluation of the samples, it is assumed that the images present high contrast between foreground (the hand) and the background. This allows not worrying about segmentation issues. To avoid using background extraction and estimation, the background is black (or as dark as possible). This means that hands with light skin will still stand out in greyscale images, so the images used are thus in greyscale.

To allow the image samples to contain all necessary information, it helps that the frame rate of the image sequence is fast enough for the movement of the tracked object. This means that it helps if the differences between consecutive images are as minimal as possible.

1.3.4 Temporal Integration

The temporal integration methods are based on the Kalman filter and account for first and second order aspects (velocity and acceleration). The temporal integration is used on the estimates of the articulations angles determined by the pose estimation. Since

the hand movement can change very rapidly in a short amount of time, the filter always considers new measurements, instead of relying solely on history from a certain time instant and on.

1.4 Organization of the Thesis

Chapter 2 explains how to describe an articulated model and the anatomy of the human hand. These concepts are then used in two examples which illustrate basic articulated models which can be applied to hand tracking. The chapter ends with a description of the camera model used and how model data is projected on the image.

Chapter 3 concerns the aspects of model parameters estimation. It describes how to measure the differences between model data and image data and then how to use them to estimate the pose. In the end, four examples show how to estimate the pose in basic situations and illustrate the performance of the estimation in each of them.

Chapter 4 describes the image analysis block, the most complex of the blocks in this thesis. It starts by showing how to position the image trackers and how to sample the image. Then it explains how feature search is performed and how the features locations are estimated. It then ends by describing how to use those features to determine lines aligned with the phalanges of the fingers in the image. This block evolved from a series of techniques which are shown in Appendix D.

Chapter 5 deals with the aspects of temporal integration, which is based on the Kalman filter. It starts by describing the basic Kalman filter, as a reference, then evolving to the g-h-k Kalman filter. It then shows how to apply the filter to the refinement and prediction of pose estimates.

Chapter 6 shows the results of using the proposed method and an alternative one. Conclusions about the performance of the methods can be found in Chapter 7 as well as some guidelines for future work.

Appendices A to E cover some additional aspects and detailed explanations, such as relevant facts for modelling, simulation, image analysis or estimation.

2 Articulated Model

The articulated model plays an important role in the presented tracking method and is used in different ways. Its basic function is to describe the geometrical aspect of the tracked object given a set of degrees of freedom (DOF) and some fundamental distances and angles. However, it is also used to help reduce the image features search space, and can be expanded to include information such as shape, visual aspect, and width of the links (in this case the model refers to a hand or part of it).

In the present case the articulated model is described by a kinematic chain, which can be simple or have ramifications depending on the complexity of the tracked object or the wished complexity for the model which describes it. This kinematic chain describes how each link connects to the others. Separation between links and relative orientations only get defined in the modelling process.

A set of DOF encodes the variables of the articulated model, which are distances and angles allowed to change. In the end, the geometric interpretation is attained by spatial transformations, created using all distances and angles, so to position and orient the frames attached to each link with their neighbours along the chain, down to a base frame.

In order to build and choose the complexity of the model it is helpful to know the intrinsic characteristics of the tracked object, so a brief study of the human hand is presented for reference.

The last section of this chapter shows a couple of examples of how to model kinematic chains of basic complexity.

2.1 Kinematic Definition

The kinematic definition is useful to deal with the complex geometry (position and orientation) of the manipulator linkages. A manipulator may be thought of as a set of

bodies connected in a chain by joints. These bodies are called links. Joints form a connection between a neighbouring pair of links. For convenience, frames are attached to the various parts of the mechanism and the relationship between these frames is described (see Figure 2.1).

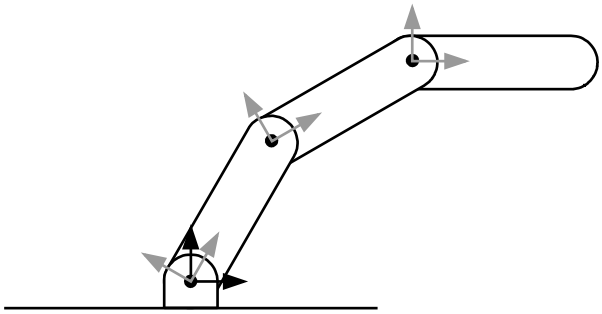


Figure 2.1 - Example of a kinematic chain with three bodies connected by revolute joints. A reference frame is attached to the base frame (black arrows) and to each of the joints (grey arrows).

The kinematic chain is modelled by the Denavit-Hartenberg (DH) notation, which is widely used in robotics [19]. Basic articulated objects, such as the finger, can be modelled by a single kinematic chain, while more complex objects, such as the hand, can be modelled by a tree-like kinematic chain, with several basic chains (the fingers) attached to a common node (the end of the wrist).

The links are numbered starting from the immobile base of the chain, which might be called link 0. The first moving body is link 1, and so on, out to the free end of the chain (the end-effector), which is link n.

In robotics, due to mechanical design considerations, manipulators are generally constructed from joints which exhibit just one degree of freedom (DOF). Most manipulators have revolute joints or sliding joints (called prismatic joints). In the rare case that a mechanism is built with a joint having n DOF, it can be modelled as n joints of one DOF connected with $n-1$ links of zero length. Therefore, without loss of generality, only manipulators which have joints with a single DOF will be considered. The reference frame is attached to the base of the chain, or link 0, and is called frame {0}. The position of all other link frames may be described in terms of this frame.

By definition of the adopted Denavit-Hartenberg notation [19], the z-axis is always aligned with the axis of rotation of a revolute joint and with the axis of dislocation of a prismatic joint. The convention for the x-axis and y-axis depends on the context.

The following is a summary of the procedure to follow when faced with a new mechanism in order to properly attach the link frames:

1. Identify the joint axes and imagine (or draw) infinite lines along them. For steps 2 through 5 below, consider two of these neighbouring lines (at axes i and $i+1$);
2. Identify the common perpendicular between them, or point of intersection. At the point of intersection, or at the point where the common perpendicular meets the i-th axis, assign the link frame origin;
3. Assign the $\hat{Z}_i$ axis pointing along the i-th joint axis;
4. Assign the $\hat{X}_i$ axis pointing along the common perpendicular, or if the axes intersect, assign $\hat{X}_i$ to be normal to the plane containing the two axes;
5. Assign the $\hat{Y}_i$ axis to complete a right-hand coordinate system $\left(\hat{X}_i \times \hat{Y}_i = \hat{Z}_i\right)$;
6. Assign {0} to match {1} when the first joint variable is zero. For {N} choose an origin location and $\hat{X}_N$ direction freely, but generally so as to cause as many linkage parameters as possible to become zero.

If the link frames have been attached to the links according to this DH convention, the following definitions of the link parameters are valid:

α_i: the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$;
a_i: the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$;
d_i: the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$;
θ_i: the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$.

Usually $a_i > 0$ since it corresponds to a distance; however, α_i, d_i and θ_i are signed quantities.

If the assignment of link frames and parameters is performed according to this

procedure, the general form of the transformation of link i relative to link $i-1$ is given by

$$ {}^{i-1}_{i}\mathbf{T} = \begin{bmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i & 0 & a_{i-1} \\ \mathrm{s}\theta_i \mathrm{c}\alpha_{i-1} & \mathrm{c}\theta_i \mathrm{c}\alpha_{i-1} & -\mathrm{s}\alpha_{i-1} & -\mathrm{s}\alpha_{i-1} d_i \\ \mathrm{s}\theta_i \mathrm{s}\alpha_{i-1} & \mathrm{c}\theta_i \mathrm{s}\alpha_{i-1} & \mathrm{c}\alpha_{i-1} & \mathrm{c}\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1} $$

where $\mathrm{c}\theta_i$, $\mathrm{s}\theta_i$, $\mathrm{c}\alpha_{i-1}$ and $\mathrm{s}\alpha_{i-1}$ denote respectively the functions $\cos(\theta_i)$, $\sin(\theta_i)$, $\cos(\alpha_{i-1})$ and $\sin(\alpha_{i-1})$.

The link transformations can be multiplied to find the single transformation that relates frame {N} to frame {0},

$$ {}^{0}_{N}\mathbf{T} = {}^{0}_{1}\mathbf{T}\ {}^{1}_{2}\mathbf{T}\ {}^{2}_{3}\mathbf{T} \cdots {}^{N-1}_{N}\mathbf{T}, \tag{2.2} $$

which is useful for geometric interpretation.

In order to describe the model without attaching a frame to the terminal tip of a chain, the length of terminal link i is represented by A_i. This will only be useful for geometric interpretation of the tip and does not interfere with the transformation matrices.

Once the articulated object is known and the complexity of the articulated model is specified it is possible to start dimensioning the model. For convenience, it is usual to start by drawing the reference frames and then store all link parameters in a table.

Sometimes, the way the link frames are laid down makes it harder to find the values of the link parameters. In such cases, and if an alternative configuration for the link frames is not easy to find, it is possible to resort to auxiliary frames, as if they were assigned to joints without moving capability. The auxiliary frames will help to derive the link parameters without increasing complexity.

As soon as the link parameters have been defined, it is important to understand which are constant and which are variable. The variable link parameters correspond to the DOF and are important because they contain information about the state of the model. In practice, the state is encoded by an array $\mathbf{q}$ of DOF. The constant link

parameters are also important since they manifest the immutable idiosyncrasies of the model.

The term “pose” will be used loosely to describe the geometrical manifestation of the current state. So in the end, “pose” is related to the Cartesian space and “state” to the state space.

2.2 Human Hand Anatomy and Physiology

To build a coherent articulated model of a human hand it is important to understand how many bones compose it, their dimensions, which articulations connect them, and the articulations range and type of movement. This evaluation later allows simplifying the model and understanding how seriously it compromises the tracking problem or the range of movement of the tracked object.

From the kinematic modelling point of view some facts are important, others are just curiosities. If the goal was to build a robotic model of the hand, some other aspects would gain more relevance. The anatomy and physiology of the human hand however, provide the main guidelines for the modelling process.

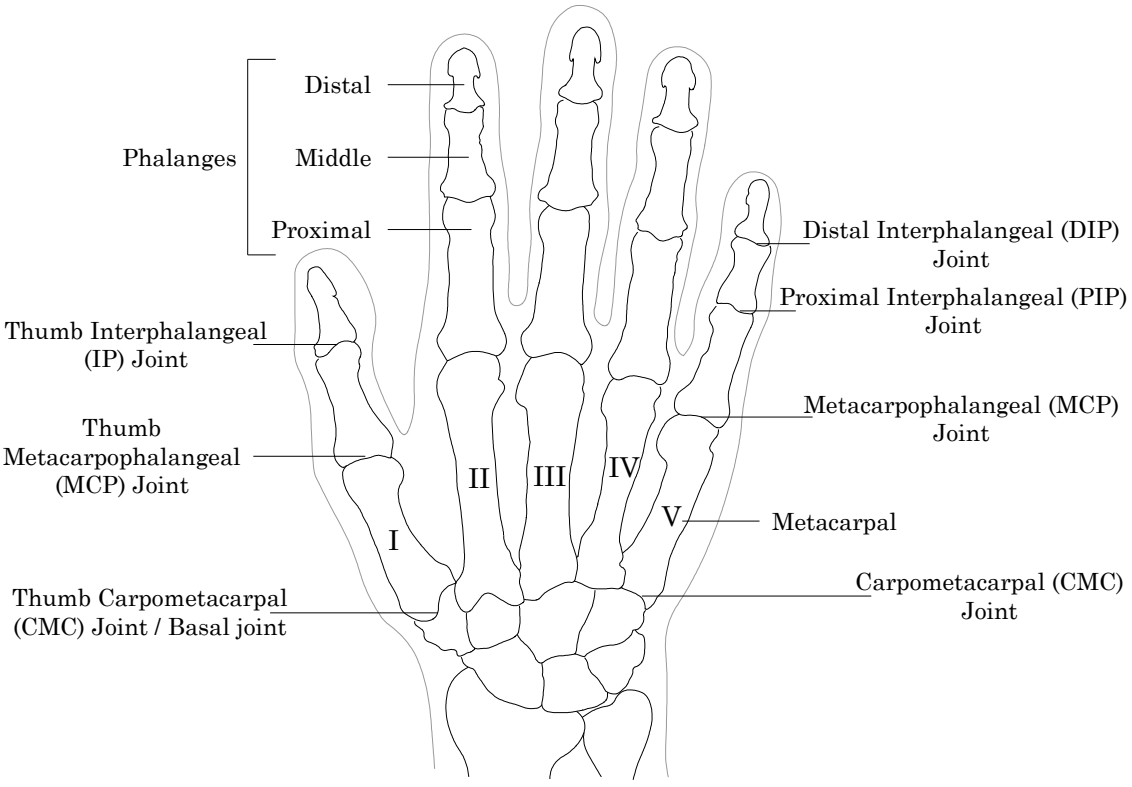


Figure 2.2 - Relevant human hand bones and joints names.

For the modelling problem at hand, only basic characteristics referring to links and joints will be considered. These include the fact that a hand has 29 major and minor bones and 29 major joints. If the model was to be more accurate some of the aspects in Appendix A should also be considered. Though fingers are never perfectly straight the models are built assuming they are, as a simplification.

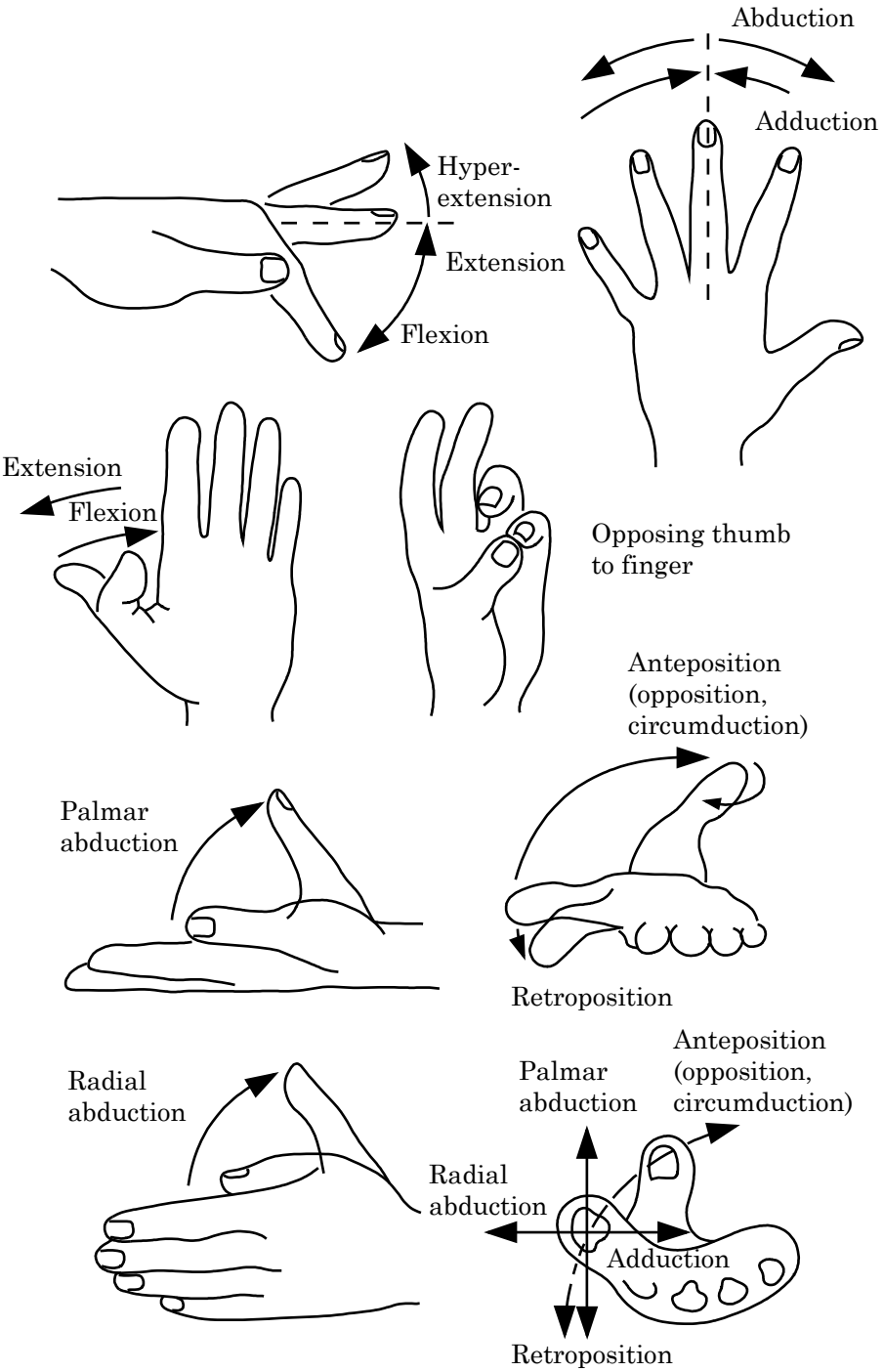


Figure 2.3 - The human hand is able to perform a wide number of movements.

To understand the anatomy and physiology of the hand, the relevant contents of [20] and [21] were summarized into Figure 2.2, Figure 2.3 and Table 2.1. These sources provide the basic understanding of the human hand.

Typical Range of Motion		
Thumb basal joint	Palmar Adduction/Abduction	Contact/45
	Radial Adduction/Abduction	Contact/60
Thumb Interphalangeal	Hyperextension/Flexion	15H/80
Thumb Metacarpophalangeal	Hyperextension/Flexion	10/55
Finger DIP joints	Extension/Flexion	0/80
Finger PIP joints	Extension/Flexion	0/100
Finger MCP joints	Hyperextension/Flexion	(0-45H)/90

Table 2.1 - Typical range of motion of thumb joints and finger joints in degrees.

2.3 Examples

The following examples were gathered in order to convey the difference between a single kinematic chain and a tree-like kinematic chain, as well as the difference of complexity behind the creation of articulated models based on those chains. These are only basic examples since the articulated models are simplifications, in terms of degrees of freedom (DOF), of the articulated objects they intend to portray. These examples also intend to illustrate how to use the Denavit-Hartenberg (DH) notation.

2.3.1 Index Finger

The index finger may be described by a single kinematic chain. In this example, the finger only has the three revolute DOF responsible for the extension/flexion movement. The adduction/abduction movement is not portrayed in this model. This low level of complexity is enough to allow some typical movements to be analysed, however, it renders the base of the finger immobile and restricts the range of movements to a plain.

Once the complexity of this model is specified it is possible to draw the link frames according to the six step procedure introduced earlier. As a way of helping doing so, an

index finger is visualized from a side view as shown in Figure 2.4. This aid immediately leads to the scheme of interconnecting joints of Figure 2.5, from which the link frames are assigned. The link frames of the single kinematic chain and the link parameters are shown in Figure 2.6.

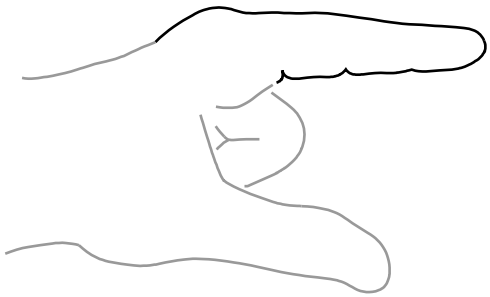


Figure 2.4 - Side view of a hand (grey curve) with highlight of an index finger (black curve).

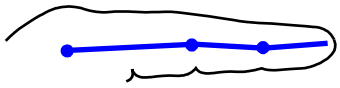


Figure 2.5 - Index finger joints (blue circles) and links (blue lines) on a side view of an index finger (black curve).

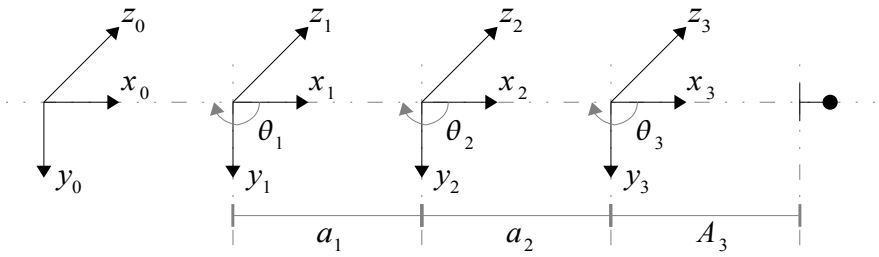


Figure 2.6 - Example of link frames assignment in a single kinematic chain representing an index finger with three revolute joints.

This is a situation where the determination of the link parameters is easily achieved by direct application of the definitions. The link parameters are shown in Table 2.2.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	a_1	0	θ_2
3	0	a_2	0	θ_3

Table 2.2 - Link parameters of a single kinematic chain representing an index finger.

From the link parameters in Table 2.2, only the θ_i are variable, all other link parameters are constant. This is so because in this model only the joint angles are allowed to change. Hence, the link transformations are given by

$$ {}^{0}_{1}\mathbf{T}=\begin{bmatrix} \mathrm{c}\theta_1 & -\mathrm{s}\theta_1 & 0 & 0 \\ \mathrm{s}\theta_1 & \mathrm{c}\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; {}^{1}_{2}\mathbf{T}=\begin{bmatrix} \mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & a_1 \\ \mathrm{s}\theta_2 & \mathrm{c}\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } {}^{2}_{3}\mathbf{T}=\begin{bmatrix} \mathrm{c}\theta_3 & -\mathrm{s}\theta_3 & 0 & a_2 \\ \mathrm{s}\theta_3 & \mathrm{c}\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Finally, the vector which encodes the state of the articulated model is

$$ \mathbf{q}=\begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^{\mathrm{T}}=\begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^{\mathrm{T}}. $$

2.3.2 Index Finger and Thumb

The index finger and thumb form a structure of two opposing fingers and may be described by a tree-like kinematic chain with two branches of single kinematic chains. In this example, the index finger and thumb only have three revolute DOF each and are each a single kinematic chain. This simplified model allows both index finger and thumb to perform the extension/flexion movement, which can then be combined to carry out a grip, release or dextrous manipulation, even if in a rather simple way. The adduction/abduction movement of the index finger as well as both the palmar and radial adduction/abduction movements of the thumb are not portrayed in this model.

Once the complexity of this model is specified it is possible to draw the link frames according to the six step procedure introduced earlier. As in the previous example, a side view of the object is used to help doing this, except this time there are two fingers, as can be seen in Figure 2.7. Albeit the image does not portray this situation, the base joint of the index finger will be vertically aligned with the base joint of the thumb to

aid in link frames assignment, making the whole chain symmetrical to a horizontal axis. With this simplification in mind, the scheme of interconnecting joints of Figure 2.8 is created, leading to the link frames of the tree-like kinematic chain and link parameters shown in Figure 2.9.

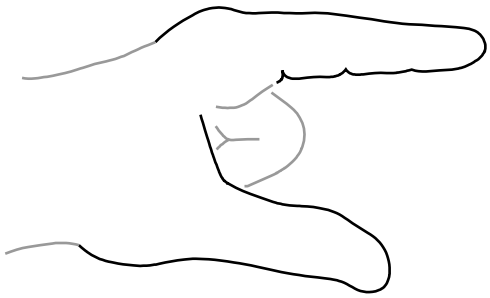


Figure 2.7 - Side view of a hand (grey curve) with highlight of an index finger (top black curve) and thumb (bottom black curve).

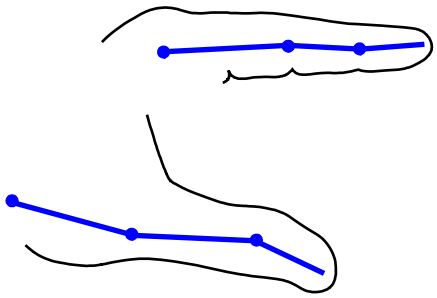


Figure 2.8 - Index finger joints (top blue circles) and links (top blue lines) on a side view of an index finger (top black curve) and thumb joints (bottom blue circles) and links (bottom blue lines) on a side view of a thumb (bottom black curve).

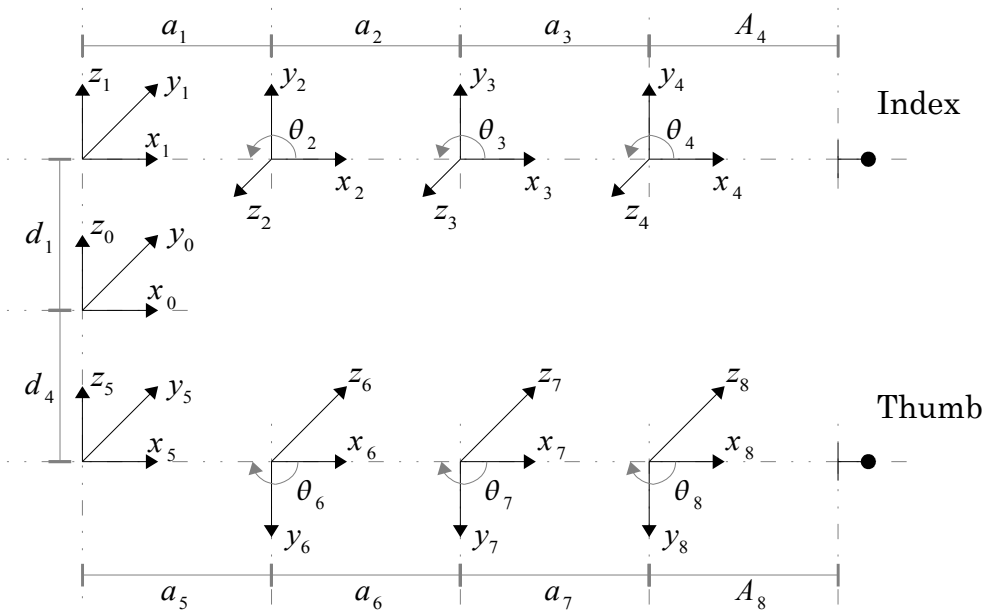


Figure 2.9 - Example of link frames assignment in a tree-like kinematic chain representing an index finger and thumb with three revolute joints each. Frame {1} is the base frame for the index finger and frame {5} the base frame for the thumb. Frame {0} is the base frame of the whole chain.

This is a situation where the determination of the link parameters is not so easily achieved by direct application of the definitions. As depicted in Figure 2.9, this is a case where including two auxiliary frames, {1} and {5}, helps to determine the link parameters without compromising the definitions. The link parameters are shown in Table 2.3.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	0
2	$\pi/2$	a_1	0	θ_2
3	0	a_2	0	θ_3
4	0	a_3	0	θ_4
5	0	0	$-\lvert d_5 \rvert$	0
6	$-\pi/2$	a_5	0	θ_6
7	0	a_6	0	θ_7
8	0	a_7	0	θ_8

Table 2.3 - Link parameters of a tree-like kinematic chain with an index finger and thumb.

From the link parameters in Table 2.3, only the non-zero θ_i are variable, all other

link parameters are constant. This is so because in this model only the joint angles are allowed to change.

Hence, the link transformations are given by

$$
{}^{0}_{1}\mathbf{T}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&d_1\\0&0&0&1\end{bmatrix},\ {}^{1}_{2}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_2&-\mathrm{s}\theta_2&0&a_1\\0&0&-1&0\\\mathrm{s}\theta_2&\mathrm{c}\theta_2&0&0\\0&0&0&1\end{bmatrix},\ {}^{2}_{3}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_3&-\mathrm{s}\theta_3&0&a_2\\\mathrm{s}\theta_3&\mathrm{c}\theta_3&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},
$$

$$
{}^{3}_{4}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_4&-\mathrm{s}\theta_4&0&a_3\\\mathrm{s}\theta_4&\mathrm{c}\theta_4&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},\ {}^{0}_{5}\mathbf{T}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&-|d_5|\\0&0&0&1\end{bmatrix},\ {}^{5}_{6}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_6&-\mathrm{s}\theta_6&0&a_5\\0&0&1&0\\-\mathrm{s}\theta_6&-\mathrm{c}\theta_6&0&0\\0&0&0&1\end{bmatrix},
$$

$$
{}^{6}_{7}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_7&-\mathrm{s}\theta_7&0&a_6\\\mathrm{s}\theta_7&\mathrm{c}\theta_7&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\ \text{and}\ {}^{7}_{8}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_8&-\mathrm{s}\theta_8&0&a_7\\\mathrm{s}\theta_8&\mathrm{c}\theta_8&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.
$$

Finally, the vector which encodes the state of the articulated model is

$$
\mathbf{q}=\begin{bmatrix}q_1&q_2&q_3&q_4&q_5&q_6\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}\theta_2&\theta_3&\theta_4&\theta_6&\theta_7&\theta_8\end{bmatrix}^{\mathrm{T}}.
$$

2.4 Graphic Visualization / Geometric Configuration

Interpretation of data stored in the state vector generally proves to be a difficult task. It is much more intuitive to evaluate the geometric configuration of the model given a state vector. The geometric configuration will be extensively used in the following two chapters, which is why it is important to specify the common basis.

First of all, assume there is an entity $\mathrm{M}(\mathbf{q})$ which projects geometric model data in the image. This data can be visible or invisible depending on the application:

- Invisible:
  - Position and orientation of the links (pose);
  - Location of end points of the links.
- Visible:

- Pose visualization;
- Creation of synthetic image.

This entity always needs a parameterization of the links of the model, which can go from simple line segments to elaborate 3D shapes. It also needs the spatial transformations derived with the DH notation, in order to position and orient each link in a 3D space described by the base frame: frame {0}. Finally, in order to describe this 3D data in 2D image units, it also needs the camera projection matrix. In short, this entity works as summarized in Figure 2.10.

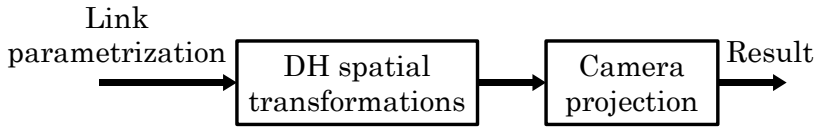


Figure 2.10 - Entity M(q) projects model data on the image frame according to this block diagram.

2.4.1 Camera Model

The camera model explains how data in a 3D space is projected on a 2D space. For convenience of manipulation, a point defined by a set of Euclidian coordinates $[X\,Y\,Z]^T$ is described by a set of projective coordinates $[X\,Y\,Z\,1]^T$. This enables performing rotations and translations with a single matrix multiplication.

This matrix is called the camera projection matrix and is given by

$$\mathbf{P}=\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix},$$

generally assuming that $p_{34}=1$, since the matrix is defined up to a non-zero scale factor.

This matrix combines informations about the intrinsic and extrinsic parameters of the camera, as well as the perspective projection. The intrinsic parameters relate the coordinate system of the camera with the coordinate system of the projection plain,

describing aspects such as the dimension of the image pixels and the position of the image centre. The extrinsic parameters relate the coordinate system of the camera to a fixed world coordinate system, specifying its position and orientation in space.

In the two extremes of projective geometry are the perspective camera model and the orthographic camera model. The orthographic model is actually a particular case of the perspective model. The major difference resides in the fact that in the orthographic model any point in space is projected orthogonally on the image plain, while in the more generic perspective model the projection corresponds to the intersection of the image plain with a line defined by the camera centre and the point in space. Thus, the orthographic model can be thought of as a perspective model where the camera centre is at an infinite distance from the camera plain.

The projection is computed with

$$\begin{bmatrix} \lambda x_p \\ \lambda y_p \\ \lambda \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

where the vector on the right hand side represents the coordinates of the 3D point and the coordinates on the left the point in the image, where λ is a scaling factor.

The perspective model is characterized by a matrix $\mathbf{P}=\mathbf{K}[\mathbf{R}|\mathbf{t}]$, where $\mathbf{K}$ is a (3×3) matrix accounting for the intrinsic parameters, $\mathbf{R}$ a matrix accounting for the rotation between camera and world and $\mathbf{t}$ a vector accounting for the coordinates of the optical centre in the world reference frame.

In the particular case of the orthographic projection, $[\mathbf{R}|\mathbf{t}]=[\mathbf{I}|0]$.

2.4.2 Image Projection

For coherence, the projection matrix $\mathbf{P}$ is now represented by ${}^{C}_{0}\mathbf{T}$ in order to use the same notation used in the kinematic model. Any point ${}^{i}\mathbf{p}$ described in its own reference frame {i} is projected on the image frame with

$${}^{C}\mathbf{p}={}^{C}_{i}\mathbf{T}\,{}^{i}\mathbf{p}, \tag{2.3}$$

where

$$ {}^{C}_{i}\mathbf{T} = {}^{C}_{0}\mathbf{T}\,{}^{0}_{i}\mathbf{T} \tag{2.4} $$

is the matrix resulting from multiplying the (3×4) projection matrix ${}^{C}_{0}\mathbf{T}$ by the single transformation which relates frame {i} to frame {0}.

For details on link parameterization and creation of synthetic images refer to Appendix C.

2.5 Chapter Outcome

This chapter presented how the articulated model is designed and how it is represented on the image reference frame.

The articulated model is useful for description of the tracked object and for positioning the image local analysis tools on the image reference frame.

The problem of pose determination is to estimate the model parameters (joint values) which make the projected articulated model overlap the collected image data. The next chapter shows how to estimate the pose once the image data is known.

3 Model Parameters Estimation

The creation of an articulated model discussed in the previous chapter is based on constant and variable link parameters. The variable link parameters correspond to degrees of freedom (DOF) and are important because they contain information about the state of the model. These are the unknowns in the pose estimation problem and have to be determined by a suitable method in order to describe the configuration of the targeted articulated object.

In the present tracking problem, the model parameters are estimated adjusting the articulated model to features extracted from an image. First, a cost function is defined, accounting for the distance between the model projected on the image plain and the observed features. The cost function is then minimized by iteratively modifying the model parameters.

The last section of this chapter presents some examples which illustrate how the estimation process works.

3.1 Pose Estimation Algorithm

The algorithm for estimating the pose of the tracked object is carried out in two interleaved phases. One involves measuring the error between the articulated model and the extracted features while the other aims to minimize that error.

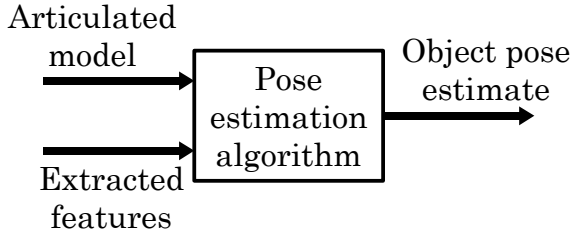


Figure 3.1 - Block diagram showing pose estimation algorithm inputs and outputs.

3.1.1 Phase 1: Defining the Residuals

The residuals are each of the distances between geometric model and extracted features. Prior to explaining how they are measured it helps to mention the elements in use, as to simplify the overall understanding of the process. Assume there is an image analysis routine with one component returning lines aligned with the finger phalanges and other returning the finger tips location. Also assume the state vector $\mathbf{q}$ has a certain value (e.g. a prediction) and $\mathrm{M}(\mathbf{q})$ indicates the end coordinates of each link in the image frame.

Tip Residuals

The tip residual measures the distance in image units between predicted, $\mathbf{t}_i(\mathbf{q})$, and measured, $\mathbf{t}_i^{obs}$, tip positions. Each tip feature is represented by a vector $[x \quad y]^T$ which gives the tip feature position in image coordinates. The residual for the i-th tip feature is a vector in the image plane defined by

$$\boldsymbol{\varepsilon}_i^{tip}(\mathbf{q}) = \mathbf{t}_i(\mathbf{q}) - \mathbf{t}_i^{obs}, \tag{3.1}$$

where $\mathbf{t}_i(\mathbf{q})$ is the projection of the tip centre into the image as a function of the hand state $\mathbf{q}$. This projection is achieved with (2.3).

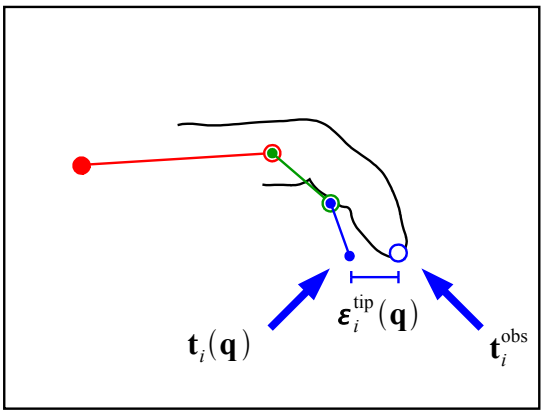


Figure 3.2 - Tip residual measured between predicted model tip (blue dot) and measured tip feature (blue circle). The coloured segments represent the finger model, the remaining dots and circles its joints and the black curve the boundary of the tracked finger.

Link Residuals

The link residual $\varepsilon_i^{link}(\mathbf{q})$ is a scalar which measures the distance from a point in the projected link axis to the measured feature line. This distance is measured perpendicularly to the feature line. As a simplification, the lines are described in the image plane using an orthographic projection. The orthographic camera model is incorporated into the residual equation by setting $\mathbf{m}=[a \quad b \quad 0]^T$ and writing

$$\varepsilon_i^{link}(\mathbf{q})=\mathbf{m}^T\mathbf{p}_i(\mathbf{q})-\rho\,, \tag{3.2}$$

where $\mathbf{p}_i(\mathbf{q})$ is the 3D position, $[x \quad y \quad z]^T$, of a point on the link axis in camera coordinates, and $[a \quad b \quad \rho]$ are the line parameters. Each link feature is represented by a vector $[a \quad b \quad \rho]$ which gives the parameters of the 2D line equation $ax+by-\rho=0$. The total link residual consists of one or more point residuals along the link axis, each given by (3.2).

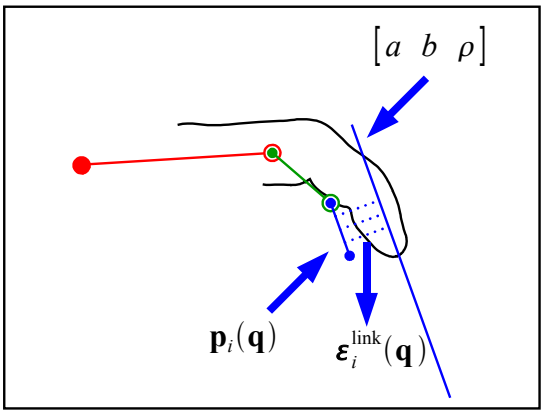


Figure 3.3 - Link residual measured between point in predicted model link (intersection of dashed segment with blue segment) and measured line feature (blue line). The coloured segments represent the finger model, the remaining dots and circles its joints and the black curve the boundary of the tracked finger.

Total Residual

The total deviation between model data and image data is achieved by gathering all residuals for the link and tip features into a single vector of residuals, for example

$$\boldsymbol{\varepsilon}(\mathbf{q})=\left[\varepsilon_{1,x}^{tip}(\mathbf{q}) \quad \varepsilon_{1,y}^{tip}(\mathbf{q}) \quad \varepsilon_1^{link}(\mathbf{q}) \quad \varepsilon_2^{link}(\mathbf{q}) \quad \cdots\right]^T,$$

where $\varepsilon_{1,x}^{tip}(\mathbf{q})$ and $\varepsilon_{1,y}^{tip}(\mathbf{q})$ are the components of the tip residual $\boldsymbol{\varepsilon}_1^{tip}(\mathbf{q})$ along the x-axis and y-axis respectively. If the magnitude of the residual vector is zero, the articulated model is perfectly aligned with the image data.

3.1.2 Phase 2: State Vector Estimation

The goal of this phase is to estimate the state vector $\mathbf{q}$ which minimizes the magnitude of the vector of residuals. This is done iteratively through successive incremental corrections.

The state correction is obtained from the vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ by minimizing the cost function

$$H(\mathbf{q}) = \frac{1}{2}\left\|\boldsymbol{\varepsilon}(\mathbf{q})\right\|^2 = \frac{1}{2}\boldsymbol{\varepsilon}(\mathbf{q})^T\boldsymbol{\varepsilon}(\mathbf{q}) . \tag{3.3}$$

The vector of residuals generally has nonlinearities. The nonlinearity in the state model for articulated mechanisms is caused by the trigonometric terms in the forward kinematic model. The other source of nonlinearity, inverse depth coefficients in the perspective camera model, does not exist since an orthographic formulation is being used. The Gauss-Newton (GN) algorithm is used to solve this nonlinear least squares problem.

Thus, the estimated state vector corresponds to

$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q}} \|H(\mathbf{q})\|^2 .$$

Update Equation

Since a different vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ is minimized at each image j, the notation should be altered to $\boldsymbol{\varepsilon}(\mathbf{q}_j)$, but for the sake of readability, this notation shall be further shortened to $\boldsymbol{\varepsilon}_j$. The GN state update equation, from iteration k to $k+1$, is given by

$$\mathbf{q}_{k+1} = \mathbf{q}_k - [\mathbf{J}_k^T\mathbf{J}_k]^{-1}\mathbf{J}_k^T\boldsymbol{\varepsilon}_k , \tag{3.4}$$

where $\mathbf{J}_k$ is the Jacobian matrix for the residual $\boldsymbol{\varepsilon}_k$, both of which are evaluated at $\mathbf{q}_k$, and is formed from the link and tip residual Jacobians.

The update equation in (3.4) can be carried out in, at least, two ways, which can be used depending on the desirable performance of the state estimation:

- For a faster, yet less accurate, state estimation, the state is updated only once per image, so that $\mathbf{q}_{j+1}$ is the result of running (3.4) only once.
- For better state estimation at the cost of some computational time, the state is updated a variable number of times per frame, so that $\mathbf{q}_{j+1}$ is the result of running (3.4) until a certain convergence criterion is met (e.g. the stabilization of the cost function).

The GN algorithm is preferred over other methods since it is faster and more accurate near an error minimum. An alternative to the GN algorithm would be the Levenberg-Marquardt algorithm, with the state update equation

$$\mathbf{q}_{k+1} = \mathbf{q}_k - [\mathbf{J}_k^T \mathbf{J}_k + \mu \mathbf{I}]^{-1} \mathbf{J}_k^T \boldsymbol{\varepsilon}_k$$

which uses a constant diagonal regularization matrix $\mu \mathbf{I}$. When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size.

Jacobian

The Jacobian matrix is related to the direction of fastest growth of a function. The GN state update equation (3.4) uses this information to reach a minimum of the cost function by changing the sign to a minus on the second term of the right hand side of (3.4), thus originating the correction

$$\Delta \mathbf{q}_k = -[\mathbf{J}_k^T \mathbf{J}_k]^{-1} \mathbf{J}_k^T \boldsymbol{\varepsilon}_k \,. \tag{3.5}$$

The term $[\mathbf{J}_k^T \mathbf{J}_k]^{-1} \mathbf{J}_k^T$ corresponds to the pseudo-inverse of the Jacobian $\mathbf{J}_k$, and is

used because the Jacobian generally is not a square matrix, and cannot be inverted. It is important to make sure that $\mathbf{J}_k^T \mathbf{J}_k$ is not singular.

By definition, if the vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ has length m, and the state vector $\mathbf{q}$ has length n, the Jacobian will be a matrix of size $(m \times n)$ of partial derivatives represented by

$$\mathbf{J} = \frac{\partial \boldsymbol{\varepsilon}(\mathbf{q})}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial \varepsilon_1}{\partial q_1} & \cdots & \frac{\partial \varepsilon_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_m}{\partial q_1} & \cdots & \frac{\partial \varepsilon_m}{\partial q_n} \end{bmatrix}, \tag{3.6}$$

where ε_i is the i-th element of $\boldsymbol{\varepsilon}(\mathbf{q})$ and q_j the j-th element of $\mathbf{q}$.

It is important to remember the vector of residuals can have two types of elements: tip residuals, in the form of two scalar entries, and link residuals, in the form of a single scalar entry.

The link Jacobian is achieved by differentiation of (3.2) with respect to the state vector, obtaining

$$\frac{\partial \varepsilon_i^{link}(\mathbf{q})}{\partial \mathbf{q}} = \mathbf{m}^T \frac{\partial \mathbf{p}_i(\mathbf{q})}{\partial \mathbf{q}}, \tag{3.7}$$

with the above gradient vector for link i being one row of the total Jacobian matrix.

Similarly, the tip Jacobian is obtained from (3.1) as

$$\frac{\partial \boldsymbol{\varepsilon}_i^{tip}(\mathbf{q})}{\partial \mathbf{q}} = \frac{\partial \mathbf{t}_i(\mathbf{q})}{\partial \mathbf{q}}, \tag{3.8}$$

with the above gradient matrix for link i being two rows of the total Jacobian matrix, one regarding the tip residual along x and the other along y.

In practice, the total Jacobian is computed off-line with symbolic mathematics. The resulting expression is then embedded in the source code for faster processing (see the upcoming examples).

3.2 Examples

The following examples show the application of the model of residuals to different kinematic chains. Each example is composed of a theoretical introduction, where the kinematic model and model of residuals are formulated, and an experimental part to convey the performance of the estimation algorithm. The last example also includes an additional experimental part dedicated to the bypass of a situation where the process does not converge to the optimal solution.

As a simplification in the following examples, the base frame of the kinematic chain is aligned with the image frame and the units are the same.

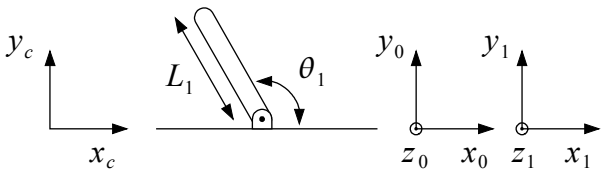
Remember that all examples have the same structure. This helps relating a new example with a previous one. Also pay attention to the schematic figures, since they portray the situation at hand.

3.2.1 Revolute Joint (R)

In this case, a single link is attached to a revolute joint which allows it to move in a plane parallel to the image plane.

Theoretical formulation

The articulated model is described in Figure 3.4.



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1

Figure 3.4 - An articulated model with one revolute joint and its DH parameters. Featuring from left to right: image frame, object description, DH frames and DH parameters.

The transformations between reference frames are given by

$$ {}^{C}_{0}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & p_0^x \\ 0 & 1 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } {}^{0}_{1}\mathbf{T} = \begin{bmatrix} \mathrm{c}\theta_1 & -\mathrm{s}\theta_1 & 0 & 0 \\ \mathrm{s}\theta_1 & \mathrm{c}\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where p_0^x and p_0^y represent the location of the origin of {0} in the image frame, along the x-axis and y-axis respectively.

A generic point on the link is represented on the link reference frame by

$$\mathbf{p}=[P \quad 0 \quad 0 \quad 1]^T,$$

where P is the position along the x-axis. At the end of the link $P=L_1$.

The application of the measurement of residuals to this case yields the situation depicted in Figure 3.5.

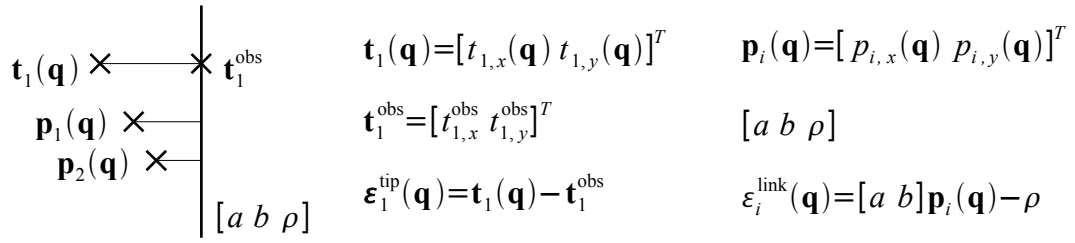


Figure 3.5 - Application of the model of residuals to the case of an articulated object with one revolute joint. Featuring (left to right): predicted features and extracted features; tip data; link data.

Consequently the vector of residuals is given by

$$\boldsymbol{\varepsilon}(\mathbf{q})=\begin{bmatrix} t_{1,x}^{tip}(\mathbf{q})-t_{1,x}^{obs} \\ t_{1,y}^{tip}(\mathbf{q})-t_{1,y}^{obs} \\ a\,p_1^x(\mathbf{q})+b\,p_1^y(\mathbf{q})-\rho \\ a\,p_2^x(\mathbf{q})+b\,p_2^y(\mathbf{q})-\rho \end{bmatrix},$$

where the scripts x and y represent the components of each vector along the x-axis and y-axis respectively.

A point on the link is projected on the image frame using (2.3), yielding

$${}^C\mathbf{p}_i(\mathbf{q})={}^C_0\mathbf{T}\,{}^0_1\mathbf{T}\,{}^1\mathbf{p}_i(\mathbf{q})=\begin{bmatrix} 1 & 0 & 0 & p_0^x \\ 0 & 1 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \mathrm{c}\theta_1 & -\mathrm{s}\theta_1 & 0 & 0 \\ \mathrm{s}\theta_1 & \mathrm{c}\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} P_i \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

where ${}^j\mathbf{p}_i(\mathbf{q})$ is the representation of point i in frame j. This equation leads to

$${}^C\mathbf{p}_i(\mathbf{q})=\begin{bmatrix} P_i\,\mathrm{c}\theta_1+p_0^x \\ P_i\,\mathrm{s}\theta_1+p_0^y \\ 1 \end{bmatrix}.$$

Replacing these values in the vector of residuals returns

$$\boldsymbol{\varepsilon}(\mathbf{q})=\begin{bmatrix} L_1\,\mathrm{c}\theta_1+p_0^x-t_{1,x}^{obs} \\ L_1\,\mathrm{s}\theta_1+p_0^y-t_{1,y}^{obs} \\ a(P_1\,\mathrm{c}\theta_1+p_0^x)+b(P_1\,\mathrm{s}\theta_1+p_0^y)-\rho \\ a(P_2\,\mathrm{c}\theta_1+p_0^x)+b(P_2\,\mathrm{s}\theta_1+p_0^y)-\rho \end{bmatrix}.$$

Hence, by application of (3.6), (3.7) and (3.8) the Jacobian of the vector of residuals

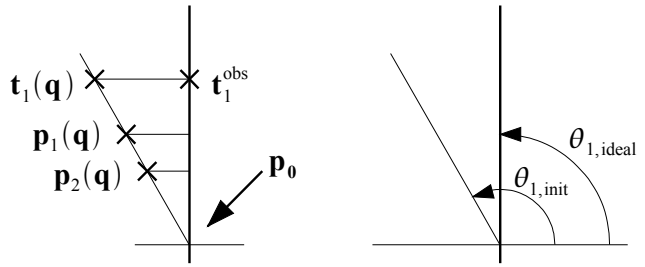
$\boldsymbol{\varepsilon}(\mathbf{q})$ regarding the state $\mathbf{q}$ is given by

$$\mathbf{J}=\begin{bmatrix}\dfrac{\partial \varepsilon_1}{\partial q_1}\\ \vdots \\ \dfrac{\partial \varepsilon_4}{\partial q_1}\end{bmatrix}=\begin{bmatrix}-L_1 \mathrm{s}\theta_1\\ L_1 \mathrm{c}\theta_1\\ a(-P_1 \mathrm{s}\theta_1)+b(P_1 \mathrm{c}\theta_1)\\ a(-P_2 \mathrm{s}\theta_1)+b(P_2 \mathrm{c}\theta_1)\end{bmatrix},$$

thus concluding the theoretical formulation of the problem.

Practical application

To test the convergence to the ideal solution in a problem of this kind, the experience shown in Figure 3.6 was performed.



$\theta_{1,\mathrm{ideal}}=\pi/2$ $\quad L_1=5$

$\theta_{1,\mathrm{init}}=\pi/2+\pi/6$ $\quad P_1=3$

$P_2=2$

$\mathbf{p_0}=[3\ 0]^T$ $\quad [a\ b\ \rho]=[1\ 0\ 3]$

Figure 3.6 - Practical application of the model of residuals and convergence algorithm to the case of an articulated object with one revolute joint.

The state update equation is ran until the convergence criterion

$$|\mathbf{q}_{k+1}-\mathbf{q}_k|\leqslant 10^{-30}\ \ [rad]$$

is reached. The reduced magnitude of the criterion is not really necessary but is that insignificant to demonstrate the fast convergence of the GN algorithm under such strict conditions.

The process converges to the ideal configuration, as can be seen in Table 3.1, Figure 3.7 and Figure 3.8.

k	0	1	2	3	4	ideal
$q_1=\theta_1$	2.0944	1.5727	1.5708	1.5708	1.5708	1.5708
$H(\mathbf{q})$	4.9744	0.0001	0.0000	0.0000	0.0000	0

Table 3.1 - Convergence of the joint parameter and cost function towards the ideal configuration in a single kinematic chain with one revolute joint.

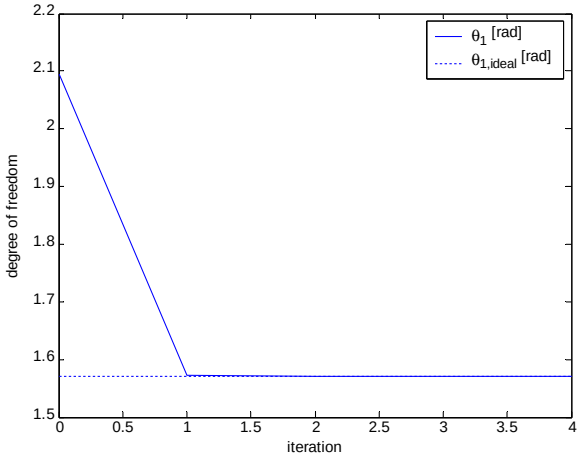


Figure 3.7 - Convergence of the joint parameter towards the ideal configuration in a single kinematic chain with one revolute joint.

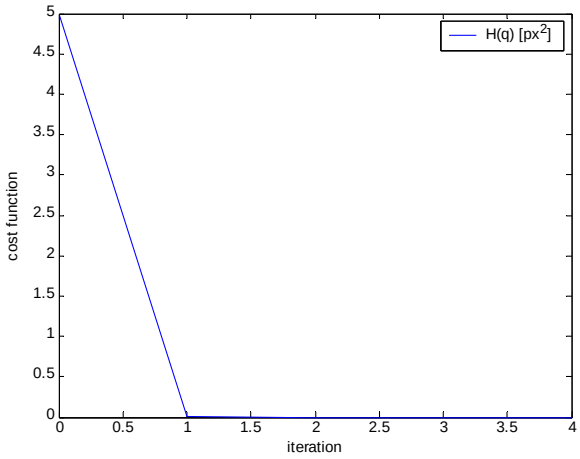


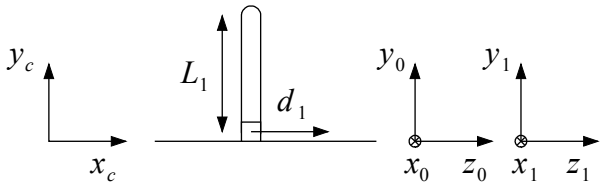
Figure 3.8 - Convergence of the cost function towards the ideal configuration in a single kinematic chain with one revolute joint.

3.2.2 Prismatic Joint (P)

In this case, a single link is attached to a prismatic joint which allows it to move in an axis parallel to the image frame x-axis.

Theoretical formulation

The articulated model is described in Figure 3.9.



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	0

Figure 3.9 - An articulated model with one prismatic joint and its DH parameters. Featuring from left to right: image frame, object description, DH frames and DH parameters.

The transformations between reference frames are given by

$$ {}^{C}_{0}\mathbf{T}=\begin{bmatrix} 0 & 0 & 1 & p_0^x \\ 0 & 1 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } {}^{0}_{1}\mathbf{T}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where p_0^x and p_0^y represent the location of the origin of {0} in the image frame, along the x-axis and y-axis respectively.

A generic point on the link is represented on the link reference frame by

$$ \mathbf{p}=[0 \quad P \quad 0 \quad 1]^T, $$

where P is the position along the y-axis. At the end of the link $P=L_1$.

The application of the measurement of residuals to this case yields a situation similar to the case of a revolute joint, hence the vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ has the same general aspect as in that situation.

A point on the link is projected on the image frame using (2.3), yielding

$$ {}^{C}\mathbf{p}_i(\mathbf{q})={}^{C}_{0}\mathbf{T}\ {}^{0}_{1}\mathbf{T}\ {}^{1}\mathbf{p}_i(\mathbf{q})=\begin{bmatrix} 0 & 0 & 1 & p_0^x \\ 0 & 1 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ P_i \\ 0 \\ 1 \end{bmatrix}, $$

where ${}^{j}\mathbf{p}_i(\mathbf{q})$ is the representation of point i in frame j. This equation leads to

$$ {}^{C}\mathbf{p}_i(\mathbf{q})=\begin{bmatrix} d_1+p_0^x \\ P_i+p_0^y \\ 1 \end{bmatrix}. $$

Replacing these values in the vector of residuals returns

$$ \boldsymbol{\varepsilon}(\mathbf{q})=\begin{bmatrix} d_1+p_0^x-t_{1,x}^{obs} \\ L_1+p_0^y-t_{1,y}^{obs} \\ a(d_1+p_0^x)+b(P_1+p_0^y)-\rho \\ a(d_1+p_0^x)+b(P_2+p_0^y)-\rho \end{bmatrix}. $$

Hence, by application of (3.6), (3.7) and (3.8) the Jacobian of the vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ regarding the state $\mathbf{q}$ is given by

$$ \mathbf{J}=\begin{bmatrix} \dfrac{\partial \varepsilon_1}{\partial q_1} \\ \vdots \\ \dfrac{\partial \varepsilon_4}{\partial q_1} \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \\ a \\ a \end{bmatrix}, $$

thus concluding the theoretical formulation of the problem.

Practical application

To test the convergence to the ideal solution in a problem of this kind, the experience shown in Figure 3.10 was performed.

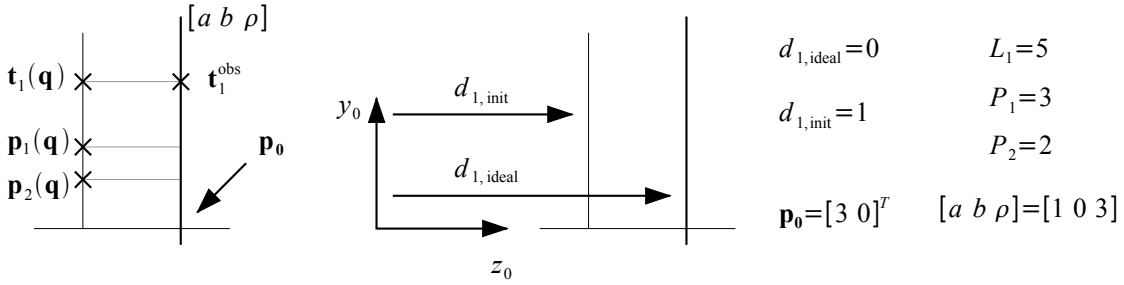


Figure 3.10 - Practical application of the model of residuals and convergence algorithm to the case of an articulated object with one prismatic joint.

The state update equation is ran until the same convergence criterion of the case with one revolute joint is reached.

The process converges to the ideal configuration, as can be seen in Table 3.2, Figure 3.11 and Figure 3.12.

k	0	1	2	ideal
$q_1 = d_1$	1.0000	0	0	0
$H(\mathbf{q})$	1.5000	0	0	0

Table 3.2 - Convergence of the joint parameter and cost function towards the ideal configuration in a single kinematic chain with one prismatic joint.

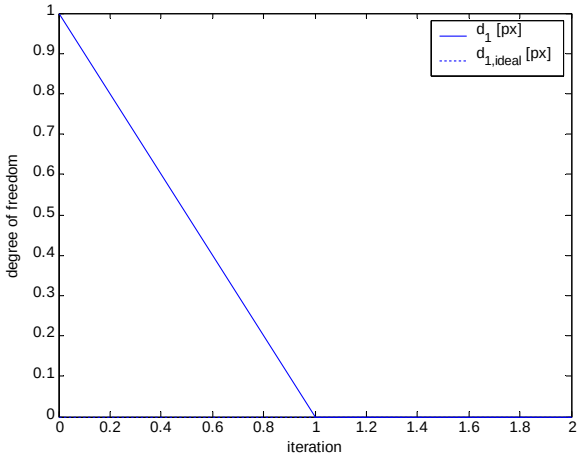


Figure 3.11 - Convergence of the joint parameter towards the ideal configuration in a single kinematic chain with one prismatic joint.

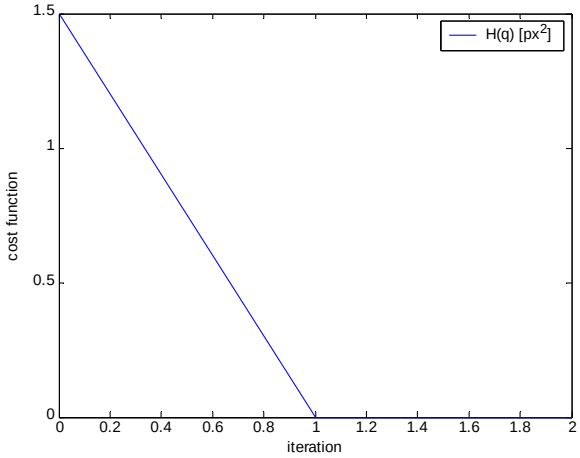


Figure 3.12 - Convergence of the cost function towards the ideal configuration in a single kinematic chain with one prismatic joint.

Note that in this case, the Jacobian does not depend of the state, so the problem could easily be solved by estimating the solution of the linear least squares problem. Since the goal is to determine the state vector which minimizes the cost function, then

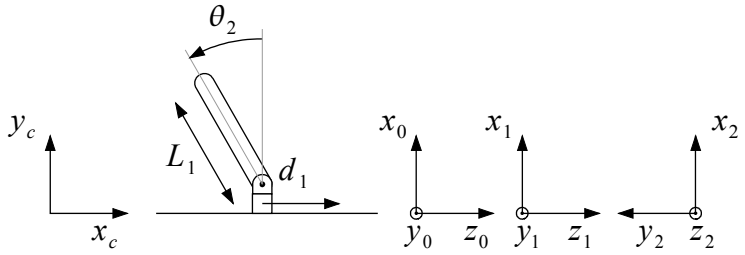
$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q}} H(\mathbf{q}) = \arg\min_{\mathbf{q}} \frac{1}{2}\|\boldsymbol{\varepsilon}(\mathbf{q})\|^2 = \arg\min_{\mathbf{q}} \|\boldsymbol{\varepsilon}(\mathbf{q})\|^2 = \arg\min_{\mathbf{q}} \boldsymbol{\varepsilon}(\mathbf{q})^T \boldsymbol{\varepsilon}(\mathbf{q})$$
$$= \arg\min_{d_1} 3\,d_1^{\,2} \Rightarrow d_1 = 0 \quad .$$

3.2.3 Prismatic and Revolute Joints (PR)

In this case, a single link is attached to a revolute joint which is attached to a prismatic joint. This configuration allows the link to move in an axis parallel to the image frame x-axis and to revolve in a plane parallel to the image plane.

Theoretical formulation

The articulated model is described in Figure 3.13.



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	0
2	$-\pi/2$	0	0	θ_2

Figure 3.13 - An articulated model with one prismatic joint and a revolute joint and its DH parameters. Featuring from left to right: image frame, object description, DH frames and DH parameters.

The transformations between reference frames are given by

$$ {}^{C}_{0}\mathbf{T}=\begin{bmatrix}0 & 0 & 1 & p_0^x\\ 1 & 0 & 0 & p_0^y\\ 0 & 0 & 0 & 1\end{bmatrix},\ {}^{0}_{1}\mathbf{T}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_1\\ 0 & 0 & 0 & 1\end{bmatrix} \text{ and } {}^{1}_{2}\mathbf{T}=\begin{bmatrix}\mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & 0\\ 0 & 0 & 1 & 0\\ -\mathrm{s}\theta_2 & -\mathrm{c}\theta_2 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, $$

where p_0^x and p_0^y represent the location of the origin of {0} in the image frame, along the x-axis and y-axis respectively.

A generic point on the link is represented on the link reference frame by

$$ \mathbf{p}=[P \quad 0 \quad 0 \quad 1]^T, $$

where P is the position along the x-axis. At the end of the link $P=L_1$.

The application of the measurement of residuals to this case yields a situation similar to the case of a revolute joint, hence the vector of residuals has the same general aspect as in that situation.

A point on the link is projected on the image frame using (2.3), yielding

$$ {}^{C}\mathbf{p}_i(\mathbf{q})={}^{C}_{0}\mathbf{T}\,{}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T}\,{}^{2}\mathbf{p}_i(\mathbf{q})=\begin{bmatrix}0 & 0 & 1 & p_0^x\\ 1 & 0 & 0 & p_0^y\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_1\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & 0\\ 0 & 0 & 1 & 0\\ -\mathrm{s}\theta_2 & -\mathrm{c}\theta_2 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}P_i\\ 0\\ 0\\ 1\end{bmatrix}, $$

where ${}^{j}\mathbf{p}_i(\mathbf{q})$ is the representation of point i in frame j. This equation leads to

$$ {}^{C}\mathbf{p}_i(\mathbf{q})=\begin{bmatrix}d_1+p_0^x-P_i\mathrm{s}\theta_2\\ p_0^y+P_i\mathrm{s}\theta_2\\ 1\end{bmatrix}. $$

Replacing these values in the vector of residuals returns

$$ \boldsymbol{\varepsilon}(\mathbf{q})=\begin{bmatrix}d_1+p_0^x-L_1\mathrm{s}\theta_2-t_{1,x}^{obs}\\ p_0^y+L_1\mathrm{c}\theta_2-t_{1,y}^{obs}\\ a(d_1+p_0^x-P_1\mathrm{s}\theta_2)+b(p_0^y+P_1\mathrm{c}\theta_2)-\rho\\ a(d_1+p_0^x-P_2\mathrm{s}\theta_2)+b(p_0^y+P_2\mathrm{c}\theta_2)-\rho\end{bmatrix}. $$

Hence, by application of (3.6), (3.7) and (3.8) the Jacobian of the vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ regarding the state $\mathbf{q}$ is given by

$$ \mathbf{J}=\begin{bmatrix}\frac{\partial\varepsilon_1}{\partial q_1} & \frac{\partial\varepsilon_1}{\partial q_2}\\ \vdots & \vdots\\ \frac{\partial\varepsilon_4}{\partial q_1} & \frac{\partial\varepsilon_4}{\partial q_2}\end{bmatrix}=\begin{bmatrix}1 & -L_1\mathrm{c}\theta_2\\ 0 & -L_1\mathrm{s}\theta_2\\ a & a(-P_1\mathrm{c}\theta_2)+b(-P_1\mathrm{s}\theta_2)\\ a & a(-P_2\mathrm{c}\theta_2)+b(-P_2\mathrm{s}\theta_2)\end{bmatrix}, $$

thus concluding the theoretical formulation of the problem.

Practical application

To test the convergence to the ideal solution in a problem of this kind, the experience shown in Figure 3.14 was performed.

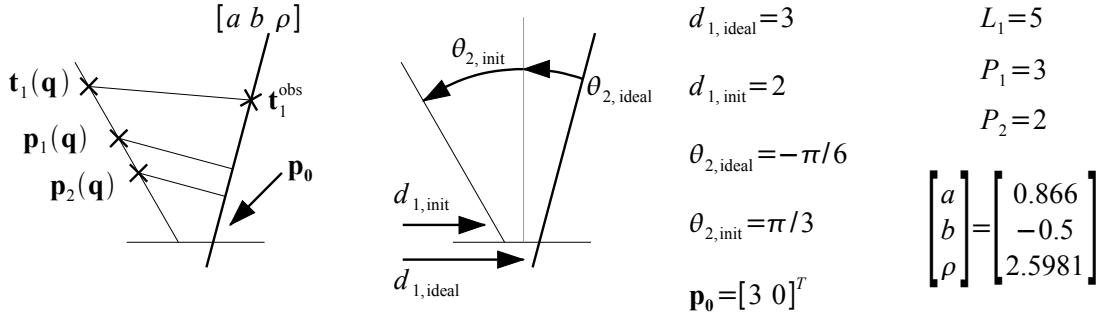


Figure 3.14 - Practical application of the model of residuals and convergence algorithm to the case of an articulated object with one prismatic joint and one revolute joint.

The state update equation is ran until a convergence criterion is reached, which in this case, because $\mathbf{q}$ has 2 elements, is given by

$$\sum|\mathbf{q}_{k+1}-\mathbf{q}_k|\leqslant 10^{-5}\ \ [rad]\ .$$

Notice that the magnitude of the criterion is much smaller than in the previous examples. However, it is still a small quantity.

The process converges to the ideal configuration, as can be seen in Table 3.3, Figure 3.15 and Figure 3.16.

k	0	1	2	⋯	5	6	7	ideal
$q_1=d_1$	2.0000	6.8490	5.1707	⋯	2.9998	3.0000	3.0000	3
$q_2=\theta_2$	1.0472	0.4321	0.0480	⋯	−0.5237	−0.5236	−0.5236	−0.5236
$H(\mathbf{q})$	43.9103	2.1344	0.7339	⋯	0.0005	0.0000	0.0000	0

Table 3.3 - Convergence of the joint parameters and cost function towards the ideal configuration in a single kinematic chain with one prismatic joint and one revolute joint.

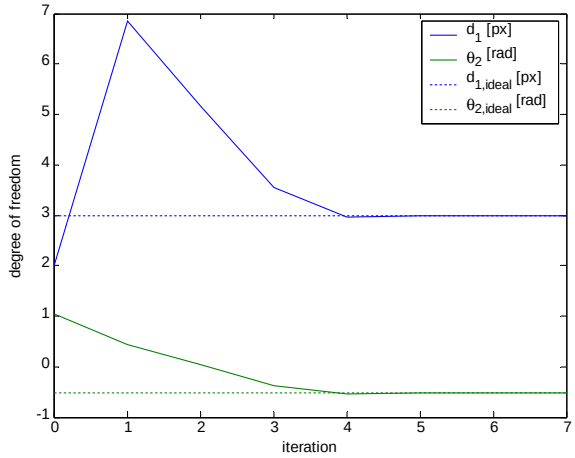


Figure 3.15 - Convergence of the joint parameters towards the ideal configuration in a single kinematic chain with one prismatic joint and one revolute joint.

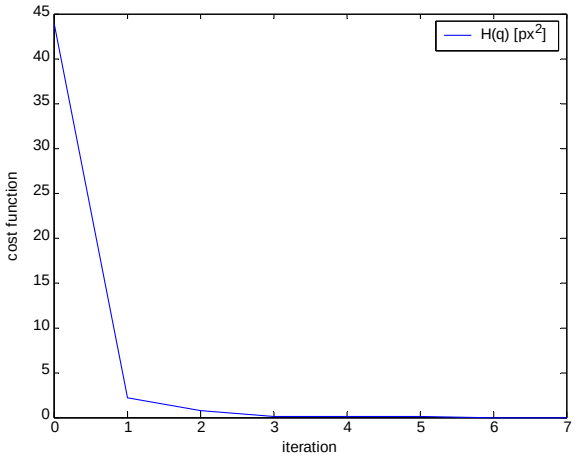


Figure 3.16 - Convergence of the cost function towards the ideal configuration in a single kinematic chain with one prismatic joint and one revolute joint.

Since it is difficult to understand from the table and figures above how the link configuration evolves with the iterations, an auxiliary animation was created to show this evolution from the geometric point of view. In order to condense this information, all frames of the animation were overlaid into a single figure, as shown in Figure 3.17.

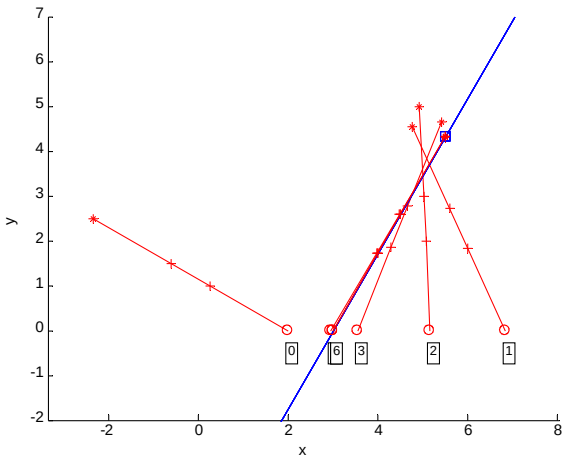


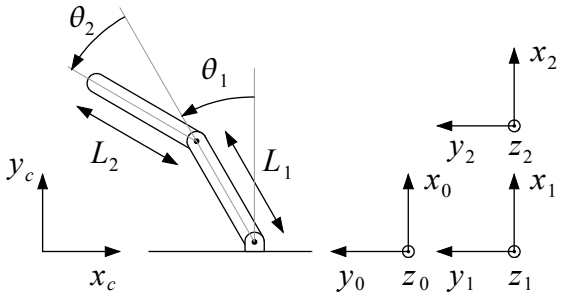
Figure 3.17 - Evolution of the link configuration with the iterations, from the geometric point of view: after some initial oscillation, the link (red line segment) converges to the link feature (blue line); the link and tip residuals minimization is visible and ends with the link points (red plus-signs) and link tip (red asterisk) overlaying the link feature and tip feature (blue square) respectively; the numbers in rectangles below the base of the link (red circle) are the iterations.

3.2.4 Two Revolute Joints (RR)

In this case, the configuration with one revolute joint is expanded with another configuration of the same type attached at tip level. This configuration allows both links to revolve in a plane parallel to the image plane. Furthermore, this combination of revolute joints also allows the new tip to perform limited linear movement.

Theoretical formulation

The articulated model is described in Figure 3.18.



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2

Figure 3.18 - An articulated model with two revolute joints and its DH parameters. Featuring from left to right: image frame, object description, DH frames and DH parameters.

The transformations between reference frames are given by

$$ {}^{C}_{0}\mathbf{T}=\begin{bmatrix} 0 & -1 & 0 & p_0^x \\ 1 & 0 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^{0}_{1}\mathbf{T}=\begin{bmatrix} \mathrm{c}\theta_1 & -\mathrm{s}\theta_1 & 0 & 0 \\ \mathrm{s}\theta_1 & \mathrm{c}\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } {}^{1}_{2}\mathbf{T}=\begin{bmatrix} \mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & L_1 \\ \mathrm{s}\theta_2 & \mathrm{c}\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where p_0^x and p_0^y represent the location of the origin of {0} in the image frame, along the x-axis and y-axis respectively.

A generic point on the link is represented on the link reference frame by

$$\mathbf{p}=[P \quad 0 \quad 0 \quad 1]^T,$$

where P is the position along the x-axis. At the end of the first link $P=L_1$ and at the end of the second link $P=L_2$.

Since now there are two links, the application of the measurement of residuals to this case yields a situation a bit more complex than the previous, as depicted in Figure 3.19.

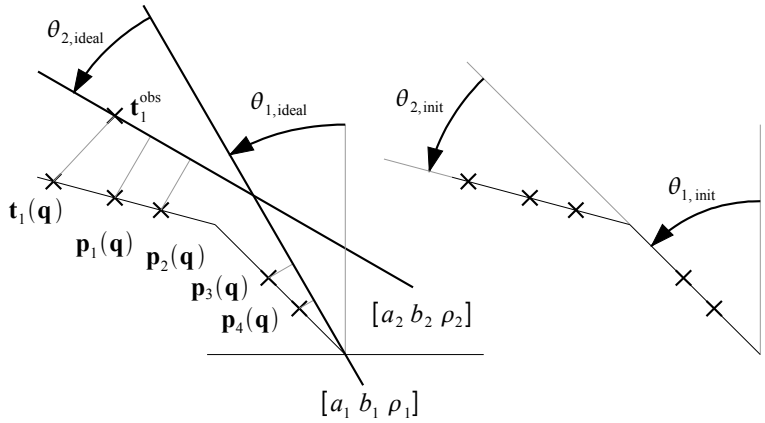


Figure 3.19 - Application of the model of residuals to the case of an articulated object with two revolute joints.

The existence of two line features leads to the need of changing the notation, so that line r_i is represented by vector $[a_i \quad b_i \quad \rho_i]$.

Consequently, the vector of residuals has the same general aspect as in the case of a revolute joint, but has more elements, due to the existence of points over the two links.

A point on a link is projected on the image frame using (2.3), yielding for a point on the terminal link

$$
\begin{aligned}
{}^{C}\mathbf{p}_i(\mathbf{q}) &= {}^{C}_{0}\mathbf{T}\,{}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T}\,{}^{2}\mathbf{p}_i(\mathbf{q}) \\
&= \begin{bmatrix} 0 & -1 & 0 & p_0^x \\ 1 & 0 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta_1 & -\mathrm{s}\theta_1 & 0 & 0 \\ \mathrm{s}\theta_1 & \mathrm{c}\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & L_1 \\ \mathrm{s}\theta_2 & \mathrm{c}\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ 0 \\ 0 \\ 1 \end{bmatrix},
\end{aligned}
$$

and for a point on the base link

$$
{}^{C}\mathbf{p}_i(\mathbf{q}) = {}^{C}_{0}\mathbf{T}\,{}^{0}_{1}\mathbf{T}\,{}^{1}\mathbf{p}_i(\mathbf{q}) = \begin{bmatrix} 0 & -1 & 0 & p_0^x \\ 1 & 0 & 0 & p_0^y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta_1 & -\mathrm{s}\theta_1 & 0 & 0 \\ \mathrm{s}\theta_1 & \mathrm{c}\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ 0 \\ 0 \\ 1 \end{bmatrix},
$$

where ${}^{j}\mathbf{p}_i(\mathbf{q})$ is the representation of point i in frame j. These equations lead respectively to

$$
{}^{C}\mathbf{p}_i(\mathbf{q}) = \begin{bmatrix} -P_i\mathrm{s}(\theta_1+\theta_2) - L_1\mathrm{s}\theta_1 + p_0^x \\ P_i\mathrm{c}(\theta_1+\theta_2) + L_1\mathrm{c}\theta_1 + p_0^y \\ 1 \end{bmatrix} \text{ and } {}^{C}\mathbf{p}_i(\mathbf{q}) = \begin{bmatrix} -P_i\mathrm{s}\theta_1 + p_0^x \\ P_i\mathrm{c}\theta_1 + p_0^y \\ 1 \end{bmatrix}.
$$

Replacing these values in the vector of residuals returns

$$
\boldsymbol{\varepsilon}(\mathbf{q}) = \begin{bmatrix} -L_2\mathrm{s}(\theta_1+\theta_2) - L_1\mathrm{s}\theta_1 + p_0^x - t_{1,x}^{obs} \\ L_2\mathrm{c}(\theta_1+\theta_2) + L_1\mathrm{c}\theta_1 + p_0^y - t_{1,y}^{obs} \\ A_{1,1} \\ A_{1,2} \\ A_{2,1} \\ A_{2,2} \end{bmatrix},
$$

where $A_{1,i}$ and $A_{2,i}$ are given by

$$
A_{1,i} = \begin{bmatrix} a_1 \\ b_1 \\ \rho_1 \end{bmatrix} \cdot \begin{bmatrix} -P_i\mathrm{s}(\theta_1+\theta_2) - L_1\mathrm{s}\theta_1 + p_0^x \\ P_i\mathrm{c}(\theta_1+\theta_2) + L_1\mathrm{c}\theta_1 + p_0^y \\ -1 \end{bmatrix} \text{ and } A_{2,i} = \begin{bmatrix} a_2 \\ b_2 \\ \rho_2 \end{bmatrix} \cdot \begin{bmatrix} -P_i\mathrm{s}\theta_1 + p_0^x \\ P_i\mathrm{c}\theta_1 + p_0^y \\ -1 \end{bmatrix}.
$$

Note that the orthographic projection is still used since the result of the dot product of the vectors above leads to the same result as (3.2) would.

Hence, by application of (3.6), (3.7) and (3.8) the Jacobian of the vector of residuals $\boldsymbol{\varepsilon}(\mathbf{q})$ regarding the state $\mathbf{q}$ is given by

$$\mathbf{J}=\begin{bmatrix} \frac{\partial \varepsilon_1}{\partial q_1} & \frac{\partial \varepsilon_1}{\partial q_2} \\ \vdots & \vdots \\ \frac{\partial \varepsilon_6}{\partial q_1} & \frac{\partial \varepsilon_6}{\partial q_2} \end{bmatrix}=\begin{bmatrix} -L_2\mathrm{c}(\theta_1+\theta_2)-L_1\mathrm{c}\theta_1 & -L_2\mathrm{c}(\theta_1+\theta_2) \\ -L_2\mathrm{s}(\theta_1+\theta_2)-L_1\mathrm{s}\theta_1 & -L_2\mathrm{s}(\theta_1+\theta_2) \\ B_{1,1} & C_{1,1} \\ B_{1,2} & C_{1,2} \\ B_{2,1} & 0 \\ B_{2,2} & 0 \end{bmatrix},$$

with $B_{1,i}$, $B_{2,i}$ and $C_{1,i}$ given by

$$B_{1,i}=\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}.\begin{bmatrix} -P_i\mathrm{c}(\theta_1+\theta_2)-L_1\mathrm{c}\theta_1 \\ -P_i\mathrm{s}(\theta_1+\theta_2)-L_1\mathrm{s}\theta_1 \end{bmatrix},\quad B_{2,i}=\begin{bmatrix} a_2 \\ b_2 \end{bmatrix}.\begin{bmatrix} -P_i\mathrm{c}\theta_1 \\ -P_i\mathrm{s}\theta_1 \end{bmatrix} \text{ and } C_{1,i}=\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}.\begin{bmatrix} -P_i\mathrm{c}(\theta_1+\theta_2) \\ -P_i\mathrm{s}(\theta_1+\theta_2) \end{bmatrix}.$$

This concludes the theoretical formulation of the problem.

Practical application

To test the convergence to the ideal solution in a problem of this kind, the experience shown in Figure 3.20 was performed.

$$\theta_{1,\text{ideal}}=-\pi/3 \qquad \mathbf{p_0}=[3\ 0]^T \qquad \begin{bmatrix} a_1 \\ b_1 \\ \rho_1 \end{bmatrix}=\begin{bmatrix} 0.5 \\ -0.866 \\ 1.5 \end{bmatrix}$$

$$\theta_{1,\text{init}}=-\pi/3+\pi/10 \qquad L_1=5 \qquad L_2=5$$

$$\theta_{2,\text{ideal}}=-\pi/2-\pi/6 \qquad P_1=3 \qquad P_3=3 \qquad \begin{bmatrix} a_2 \\ b_2 \\ \rho_2 \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \\ 7.3301 \end{bmatrix}$$

$$\theta_{2,\text{init}}=-\pi/2-\pi/6+\pi/10 \qquad P_2=2 \qquad P_4=2$$

Figure 3.20 - Practical application of the model of residuals and convergence algorithm to the case of an articulated object with two prismatic joints.

The state update equation is ran until the same convergence criterion of the case with one prismatic joint and one revolute joint is reached.

The process converges to the ideal configuration, as can be seen in Table 3.4, Figure 3.21 and Figure 3.22.

k	0	1	2	⋯	4	5	6	ideal
$q_1=\theta_1$	−0.7330	−0.8616	−1.0617	⋯	−1.0472	−1.0472	−1.0472	−1.0472
$q_2=\theta_2$	−1.7802	−2.2315	−2.0927	⋯	−2.0943	−2.0944	−2.0944	−2.0944
$H(\mathbf{q})$	5.2081	0.7270	0.0038	⋯	0.0000	0.0000	0.0000	0

Table 3.4 - Convergence of the joint parameters and cost function towards the ideal configuration in a single kinematic chain with two revolute joints.

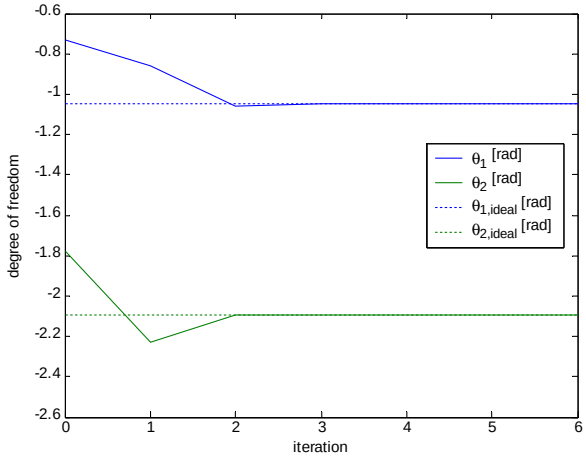


Figure 3.21 - Convergence of the joint parameters towards the ideal configuration in a single kinematic chain with two revolute joints.

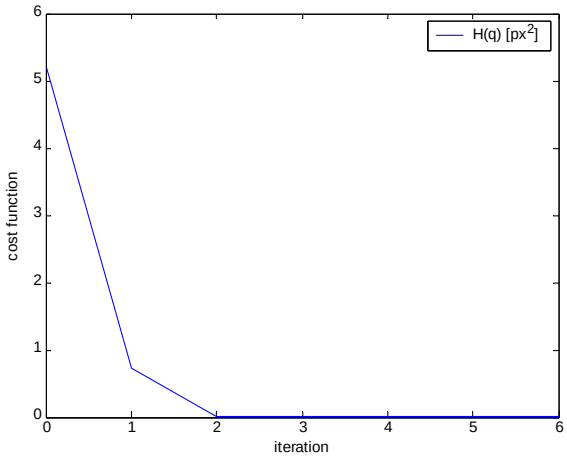


Figure 3.22 - Convergence of the cost function towards the ideal configuration in a single kinematic chain with two revolute joints.

As in the previous case, it is difficult to understand from the table and figures above how the link configuration evolves with the iterations. Once again an auxiliary animation was created to show this evolution from the geometric point of view. In order to condense this information, all frames of the animation were overlaid into a single figure, as shown in Figure 3.23.

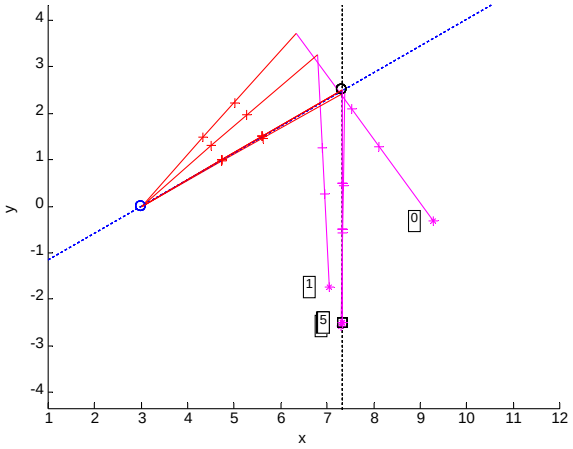


Figure 3.23 - Evolution of the link configuration with the iterations, from the geometric point of view: after some initial oscillation, the links (red and magenta line segments) converge to their link features (blue and black lines); the link and tip residuals minimization is visible and ends with the link points (red and magenta plus-signs) and link tip (magenta asterisk) overlaying their link features and tip feature (black square) respectively; the numbers in rectangles to the left of the link tip are the iterations. The rotational joints of the tracked articulated object are represented by the blue and black circles.

Additional Situation: Convergence Instability Bypass

In some situations the process does not converge to the ideal solution. In fact, in a situation where the initial pose is sufficiently distinct from the true pose, the initial value of the cost function also is very high. Because the iterative algorithm only converges if the corrections made to the state vector are incremental, the process may diverge if those corrections cannot be considered incremental.

In this example, the update equation and convergence criterion remain the same, however, the situation now is defined by the parameters of Figure 3.24.

$$\theta_{1,\text{ideal}} = -\pi/3 \qquad \mathbf{p_0} = [3\ 0]^T \qquad \begin{bmatrix} a_1 \\ b_1 \\ \rho_1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.866 \\ 1.5 \end{bmatrix}$$

$$\theta_{1,\text{init}} = \pi/6 \qquad L_1 = 5 \qquad L_2 = 5$$

$$\theta_{2,\text{ideal}} = -\pi/2 - \pi/6 \qquad P_1 = 3 \qquad P_3 = 3 \qquad \begin{bmatrix} a_2 \\ b_2 \\ \rho_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 7.3301 \end{bmatrix}$$

$$\theta_{2,\text{init}} = -\pi/3 \qquad P_2 = 2 \qquad P_4 = 2$$

Figure 3.24 - Practical application of the model of residuals and convergence algorithm to the case of an articulated object with two prismatic joints (additional situation). Note: only the initial angles changed.

As a consequence of an initial divergence, the process converges to a solution which is not the ideal, as can be seen in Table 3.5, Figure 3.25 and Figure 3.26.

k	0	1	2	⋯	14	15	16	ideal
$q_1=\theta_1$	0.5236	1.2304	1.1705	⋯	3.3975	3.3974	3.3974	−1.0472
$q_2=\theta_2$	−1.0472	−4.4211	−1.9139	⋯	−4.5082	−4.5082	−4.5082	−2.0944
$H(\mathbf{q})$	109.3510	131.2587	105.1625	⋯	7.9345	7.9345	7.9345	0

Table 3.5 - Convergence of the joint parameters and cost function towards a non-ideal solution, after an initial divergence, in a single kinematic chain with two revolute joints.

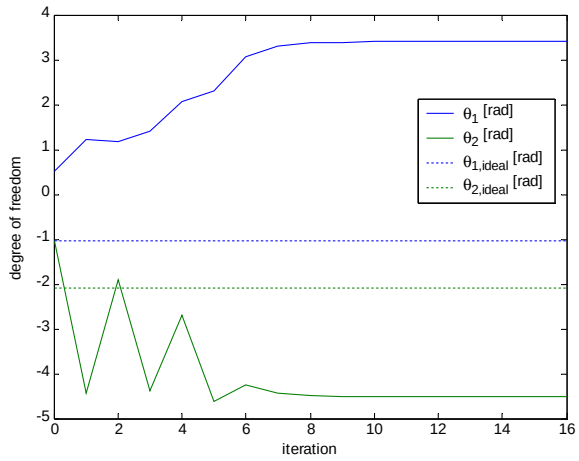


Figure 3.25 - Convergence of the joint parameters towards a non-ideal solution, after an initial divergence, in a single kinematic chain with two revolute joints.

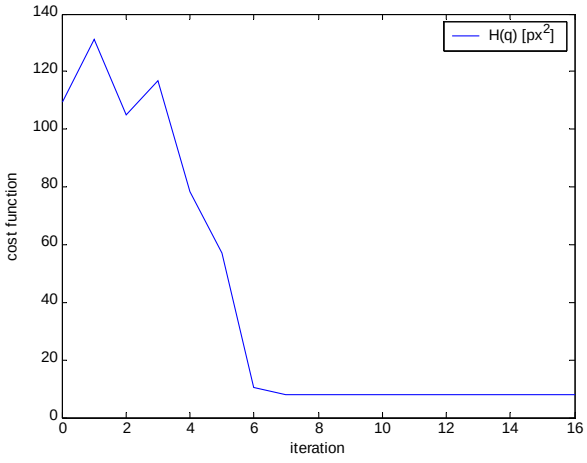


Figure 3.26 - Convergence of the cost function towards a non-ideal solution, after an initial divergence, in a single kinematic chain with two revolute joints.

Once again an auxiliary animation was created to show this evolution from the geometric point of view. In order to condense this information, all frames of the animation were overlaid into a single figure, as shown in Figure 3.27. The initial divergence is clear as well as the convergence to a non-ideal solution.

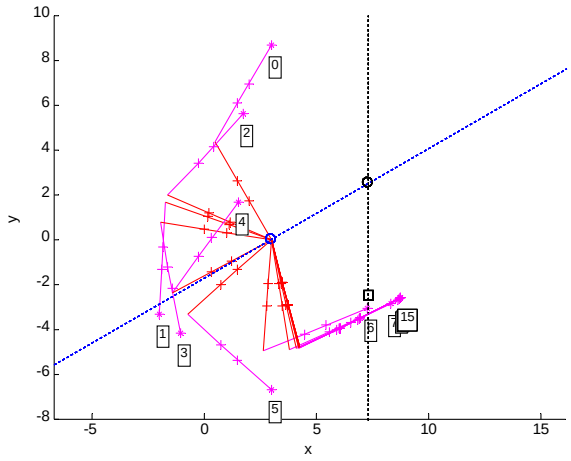


Figure 3.27 - Evolution of the link configuration with the iterations, from the geometric point of view: after some initial oscillation, the links (red and magenta line segments) converge towards a non-ideal configuration, failing to overlay their link features (blue and black lines); the link points (red and magenta plus-signs) and link tip (magenta asterisk) should overlay their link features and tip feature (black square) respectively; the numbers in rectangles below the link tip are the iterations. The rotational joints of the tracked articulated object are represented by the blue and black circles.

It was verified experimentally that with this configuration of parameters the divergence can be avoided if the state update equation is changed to

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \eta \left[\mathbf{J}_k^T \mathbf{J}_k\right]^{-1} \mathbf{J}_k^T \boldsymbol{\varepsilon}_k ,$$

where an adaptive step η is used to ensure that all corrections to the state vector can still be considered incremental. Note that the state update equation from (3.4) is a particular case of this one, where $\eta = 1$. Using a smaller adaptive step, like for example $\eta = 0.1$, the process converges at the cost of a greatest number of iterations, as can be seen in Table 3.6, Figure 3.28 and Figure 3.29.

k	0	1	2	⋯	96	97	ideal
$q_1 = \theta_1$	0.5236	0.5943	0.6476	⋯	−1.0471	−1.0471	−1.0472
$q_2 = \theta_2$	−1.0472	−1.3846	−1.6683	⋯	−2.0944	−2.0944	−2.0944
$H(\mathbf{q})$	109.3510	93.2694	79.9565	⋯	0.0000	0.0000	0

Table 3.6 - Convergence of the joint parameters and cost function towards the ideal solution, after including an adaptive step in the state update equation to bypass the problems of convergence, in a single kinematic chain with two revolute joints.

Note that the value determined for q_1 still is 10^{-4} radians away from the ideal solution, however, that only corresponds to 0.0573 degrees.

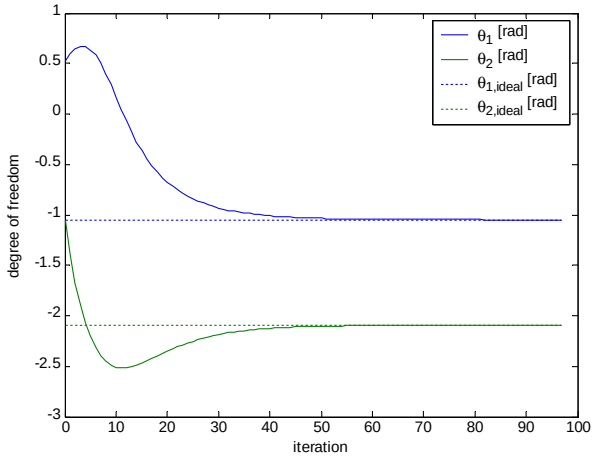


Figure 3.28 - Convergence of the joint parameters towards the ideal solution, after including an adaptive step in the state update equation to bypass the problems of convergence, in a single kinematic chain with two revolute joints.

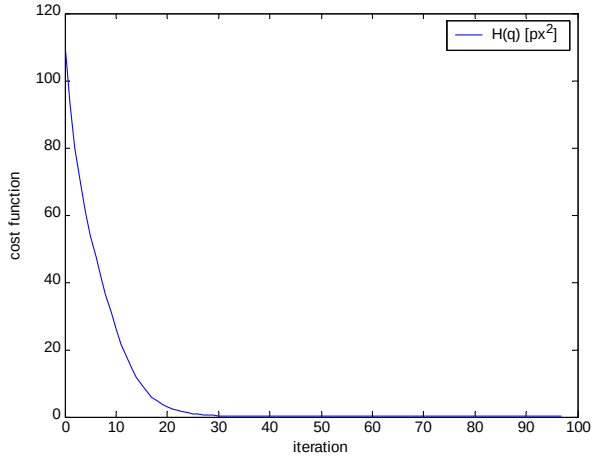


Figure 3.29 - Convergence of the cost function towards the ideal solution, after including an adaptive step in the state update equation to bypass the problems of convergence, in a single kinematic chain with two revolute joints.

The animation that shows the convergence from the geometric point of view, with all frames overlaid, can be seen in Figure 3.30.

The difference of evolution of these two convergences from the point of view of descending the surface of the cost function can be observed in Figure 3.31.

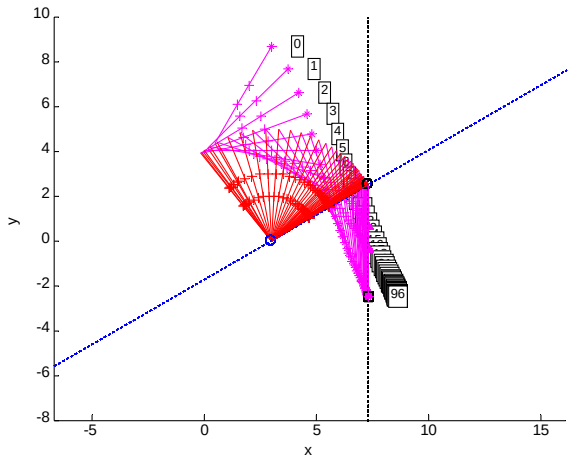


Figure 3.30 - Evolution of the link configuration with the iterations, from the geometric point of view: the links (red and magenta line segments) converge to their link features (blue and black lines); the link and tip residuals minimization is visible and ends with the link points (red and magenta plus-signs) and link tip (magenta asterisk) overlaying their link features and tip feature (black square) respectively; the numbers in rectangles to the right of the link tip are the iterations. The blue and black circles represent the rotational joints of the tracked articulated object.

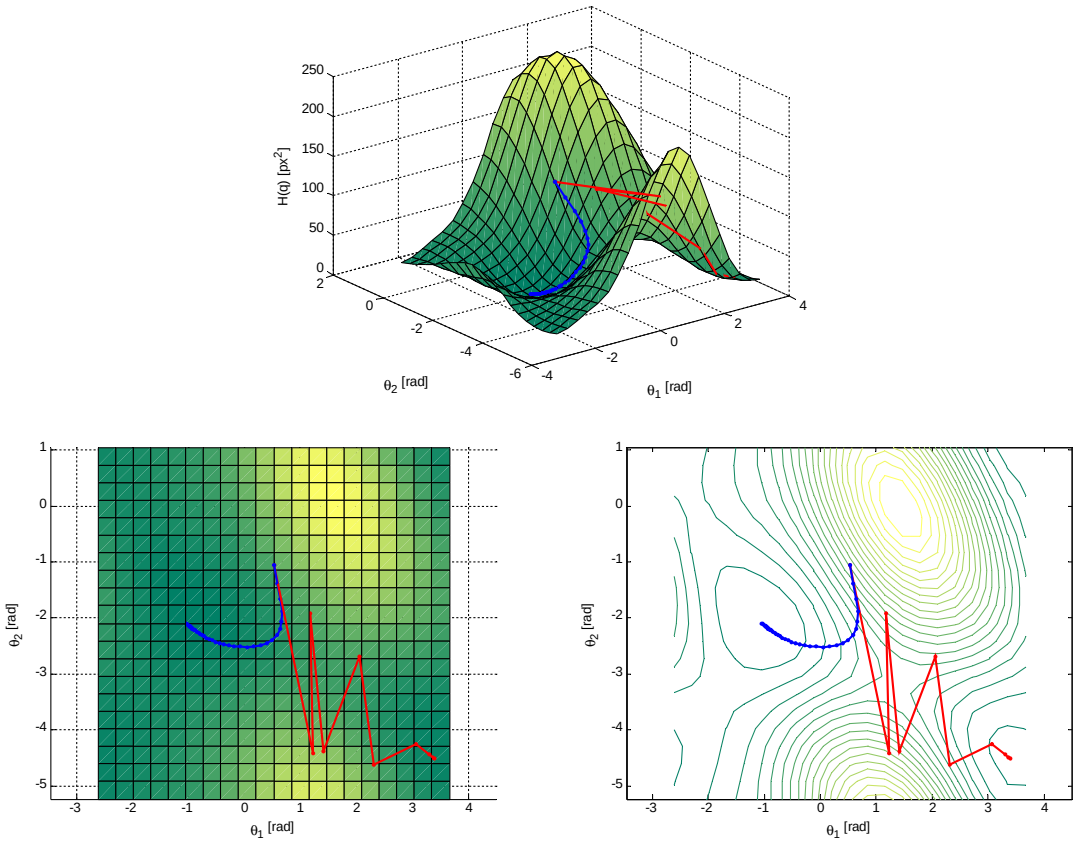


Figure 3.31 - The cost function H(q) as a function of the DOF (top: isometric view; left: top view; right: level lines); the red and blue plots represent the evolution of the DOF from a common starting point towards the non-ideal solution (red) and the ideal solution (blue). The cost function has a period of 2π radians along each DOF since both are revolute joints.

3.3 Chapter Outcome

So far, both the articulated model and the pose estimation algorithm have been presented. The pose estimation algorithm uses a projection of the articulated model and data collected from the image to estimate the model parameters which make the projected articulated model overlap the data collected from the image. However, there is still no understanding of how data is collected from the image, which is shown in the next chapter.

In this chapter it has also been shown that it is critical that the Gauss-Newton method converges to a solution as close as possible to the real one. As the complexity of the model increases, good convergence may become more difficult to attain. In such situations the update step may have to be limited using additional methods.

4 Image Analysis

The estimation of model parameters described in the previous chapter relies on data extracted from a sequence of images. The image analysis routine returns this data as axial lines aligned with the phalanges (link features) and as points related with the finger tips location (tip features). This data is estimated from image samples retrieved by specialized image analysis entities working on a local context, the image trackers.

The location of the image trackers is set by the model $\mathrm{M}(\mathbf{q})$ using a predicted state vector $\mathbf{q}$. There are two types of image trackers: link trackers, which collect data from the finger phalanges in the image, and tip trackers, which collect data from the finger tips in the image. Data is retrieved from the images with the help of these trackers, where link and tip trackers originate link and tip features, respectively.

In order to analyze an image and return data in a certain format it is necessary to go over a series of steps, all of them always present in the image analysis routine. The process can be separated into four distinct tasks:

1. Positioning image trackers;
2. Image sampling;
3. Feature search;
4. Line fitting.

The third task is the most critical and this is where most of the effort was put. Not surprisingly, it lead to a larger amount of related work, including a wide range of variations, which are presented in Appendix D for reference. The remaining tasks are more trivial.

4.1 Positioning the Image Trackers

This is the step in charge of positioning the image trackers over the image so the local analysis can then be performed. In the following explanation a single-finger situation is considered.

The model $M(\mathbf{q})$ provides estimates of the end coordinates of each link in the image frame using (2.3), where $\mathbf{q}$ is the same prediction used in the pose estimation algorithm. This prediction can be the state estimated in the previous image frame or a prediction considering the recent range of movement, understood from the estimations produced along a set of recent frames. For convenience of both feature extraction and pose estimation algorithm, the initial state of the articulated model should be as close as possible to the state of the tracked object.

As a simplification to the initialization process, the initial state is set by hand on the first frame of the image sequence, so the projection of the articulated model overlays the tracked object in the image. This avoids using more complex techniques to estimate the initial state.

Projecting the articulated model helps placing the image trackers, which are responsible for extracting contour points from the image. Each link tracker is located on the projection of its corresponding finger link, while each tip tracker is located on the projection of its corresponding finger tip.

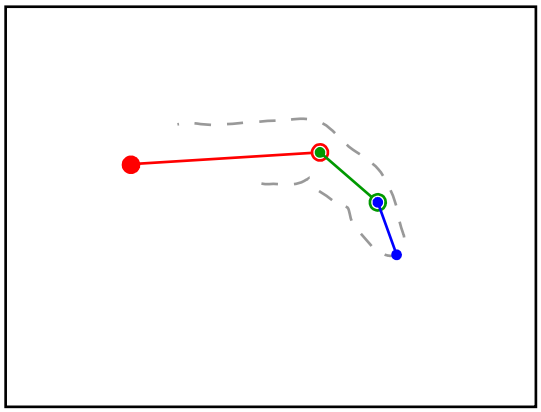


Figure 4.1 - Projection of the articulated model on the image frame showing links (coloured segments), joints (coloured circles), tip (blue dot) and silhouette of corresponding finger (dashed curve).

A link tracker is positioned and oriented the same way an articulated model link is projected on the image, therefore it is not surprising that the algebra and storing

methods involved in this process resemble those involved in simulating edge points (check Appendix B for details), as shown in Table 4.1.

Local Image Trackers	***Simulated Edge Points***
Positioning of link tracker	Projection of model link
Positioning of link tip	Projection of model tip
Positioning of samplers perpendicularly over link tracker axis	Spreading of marks over the projected link
Computation of sampling extremes for link samplers	Generation of edge points to both sides of projected link

Table 4.1 - The methods involved in positioning the image trackers resemble those involved in simulating edge points.

The derivation as how to place, orient and store the link trackers is immediate once this algebraic parallelism is noticed. Link trackers are placed directly over the projected links, and link samplers with limited scope are positioned perpendicularly along these trackers.

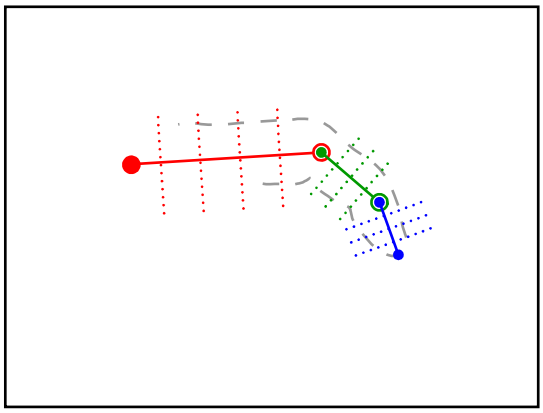


Figure 4.2 - Link samplers (dotted segments) positioned perpendicularly over the projected links. The samplers have limited scope. This example shows four samplers being used on the proximal phalanx and three samplers on each of the other two phalanges.

The positioning of the tip tracker is slightly more complex and can be described as a two-step proceeding: first a rude estimate and then a refinement. The previously discussed initial position of the tip tracker is refined using a line aligned with the distal phalanx of the finger. Such a line, also referred as a terminal axial line, is determined by the line fitting algorithm which is described later. The position of a tip

tracker is refined by projecting the position of the initial tip tracker on the corresponding terminal axial line. The tip sampler is then oriented with the direction of the axial line and sampling extremes are determined as with a tip sampler (see Figure 4.3).

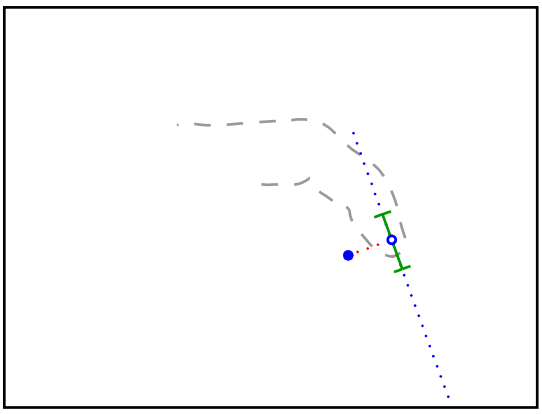


Figure 4.3 - The refined tip position (blue circle) results from projecting the predicted finger tip (blue dot) on the extracted link feature (dotted blue line), perpendicularly (dotted red segment) to the line. The tip sampler (green segment) is placed on the line and centred on the refined tip.

Given a terminal axial line defined by point $\mathbf{p}_{ax}$ and vector $\mathbf{v}_{ax}$ (headed towards the tip), the projection of model tip $\mathbf{p}_{tip}$ on the axial line corresponds to $\tilde{\mathbf{p}}_{tip}$, the point on that line closest to the model tip, which is computed with

$$\tilde{\mathbf{p}}_{tip} = \mathbf{p}_{ax} + \left(\left(\mathbf{p}_{tip} - \mathbf{p}_{ax} \right)^T \mathbf{v}_0 \right) \mathbf{v}_0 \,. \tag{4.1}$$

A directional unitary vector $\mathbf{v}_0$ is computed dividing $\mathbf{v}_{ax}$ by its norm. The sampling extremes for the tip sampler are computed relatively to the refined tip position. This is similar to defining the sampling extremes for a link sampler relatively to the point where it intersects the link tracker.

4.2 Image Profile Sampling

Once the image trackers have been placed and the extremes for each sampler specified, the image is sampled along each of the specified line segments. The sampling procedure computes the intensity values along a line or a multiline path in an image. This sampling routine selects equally spaced points along the specified path, and then uses interpolation to find the intensity value for each point. The path is

specified by a parameterization of endpoints along the x-axis and y-axis, which in this case is given by the coordinates of the sampling extremes since the path is a line segment. The resulting image intensity profile along each line segment can be thought of as a slice of the image intensity surface.

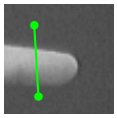


Figure 4.4 - Finger tip image and link sampler (green segment).

In the case of image profile sampling, each profile results from sampling the image in specific points along a line segment. As said before, this segment has a certain length. However, when an image is sampled along line segments with different directions a variable number of pixels is crossed, mainly because the image surface is discrete. To avoid this, the sampling routine is set to always sample a fixed amount of points. Generally, the number of pixels the path traverses does not match the requested number of points, so the sampling routine must use interpolation to find the intensity value for each of the requested points.

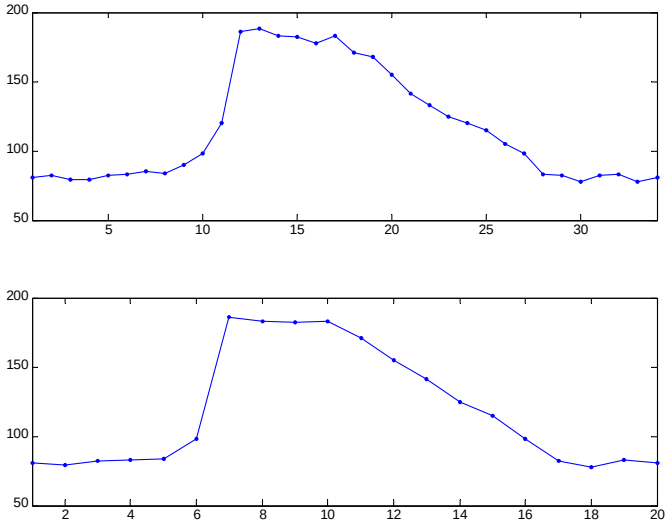


Figure 4.5 - Image profile returned by a link sampler: number of pixels in the sample not specified (top) and fixed on 20 points (bottom). Notice how the sample with less points is smoother.

In essence, all the interpolation methods work in a similar way. In each case, the

desired interpolated pixel is a point in the image. Since the coordinates of this point generally are not integers, its intensity value is determined by computing a weighted average of some set of pixels in the vicinity of the point. The weightings are based on the distance each pixel is from the point.

The methods differ in the considered set of pixels used for the weighting:

- For nearest neighbour interpolation the desired point is assigned the value of the pixel it falls within. No other pixels are considered.
- For bilinear interpolation, the point value is a weighted average of pixels in the nearest 2-by-2 neighbourhood.
- For bicubic interpolation, the point value is a weighted average of pixels in the nearest 4-by-4 neighbourhood.

Do bear in mind that this description is merely for elucidation purposes. The sampling routines only require the path parameterization and the interpolation method to use.

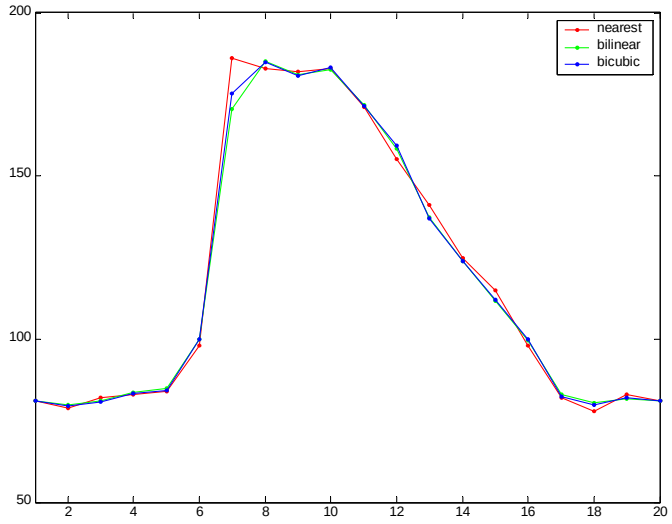


Figure 4.6 - Comparative plot of an image profile with 20 points using nearest neighbour (red), bilinear (green) and bicubic (blue) interpolations.

The number of pixels considered in the interpolation affects the complexity of the computation. Therefore the bilinear method takes longer than nearest neighbour

interpolation, and the bicubic method takes longer than bilinear. However, the greater the number of pixels considered, the more accurate the effect is, so there is a trade-off between processing time and quality.

Notice there was no distinction whether the sampling referred to link trackers or tip trackers. This was intentional because the sampling process itself does not depend on knowing if it is going to be used for link or tip tracking purposes.

4.3 Searching for Edges

This step must analyze the intensity profiles to determine edge points as accurately and robustly as possible. There are two types of edge points, one regarding the sides of the phalanges and the other the finger tips. Points of the first kind are divided into groups which will later be used to determine the axial lines. Points of the second kind indicate the position of the finger tip.

Figure 4.7 - Feature search is the block responsible for analysing the image profiles and finding the location of the edge points.

The way of retrieving edge points for the links or tips are not necessarily the same. In fact, apart from the initial motivation, the investigation methods head different ways. In this case however, finding tip edges is always dependent of the axial line determination.

The importance of extracting features from an image is only equalled by the pose estimation algorithm. Good pose estimation is achieved in part by good and robust feature extraction and in part by an appropriately convergent pose estimation algorithm.

Feature search is the process responsible for getting information from an image and returning data ready to use in pose estimation. Appendix B shows how to simulate this data when feature search routines are not implemented.

This is the most critical step of the image analysis process. Initially it was a simple method but was progressively enhanced for performance and robustness until the

method described in this section was reached. Prior developments of the method and their alternatives can be found in Appendix D. The first of those methods is the most basic and is based on [1]. It suffices for a very basic implementation of feature extraction.

The present method evolved from that same method, so it still preserves the same goal: using each image profile sample to determine the location of the edges used in line fitting.

However, this method is very different from the preceding ones in one aspect: it is the only one designed to handle finger occlusion. When considering fingers there can be mutual or self occlusion, respectively if a finger occludes another or itself.

The proposed method is a compromise between the perpendicular sampling idea in [1] and the templates and windowing functions concepts in [2]. A consequence of this compromise is no longer using the derivatives of the image profiles and using the profiles instead. The profiles are downsamples of the finger phalanges, so they still contain information about the aspect of the fingers on image. Derivatives aren't used because if an image profile samples a region where a finger is occluded, the derivative of that image profile will present peaks in undesired regions.

4.3.1 Template Matching

Another concept borrowed from [2] is template matching using the standard sum of squared differences (SSD). Template matching using correlation tends to privilege regions with high intensity, as those resulting from a more intense illumination. To avoid such discrimination, and since it is reasonable to assume there are no changes in illumination along the image sequence, the sum of square differences (SSD) is used and is defined as

$$SSD_{\mathbf{s}_1\mathbf{s}_2}(k)=\sum_j\left[\mathbf{s}_1(j)-\mathbf{s}_2(j+1-k)\right]^2$$

with a best match given by

$$\hat{k}=arg\ \min_k\ SSD_{\mathbf{s}_1\mathbf{s}_2}(k)\ .$$

For performance reasons the SSD is computed as a function of correlations, since the correlation of two signals is already implemented as an optimized function. When written as a function of correlations, SSD becomes

$$SSD_{\mathbf{s}_1\mathbf{s}_2}(k) = R_{\mathbf{s}_1\mathbf{i}}(k) - 2R_{\mathbf{s}_1\mathbf{s}_2}(k) + \mathbf{s}_2^T\mathbf{s}_2 ,$$

where $\mathbf{i}$ is a vector the same size as $\mathbf{s}_2$ filled with ones and $R_{\mathbf{x}\mathbf{y}}(k)$ is the correlation of signals $\mathbf{x}$ and $\mathbf{y}$.

In order for the SSD to be robust to occlusions it is necessary to restrict its scope of calculation to the image regions where the phalanx in question is visible. This is where another concept is borrowed from [2]: windowing functions. These functions present ones where the phalanx is visible and zeros outside. In the present adaptation of that concept each window function masks an image profile template originating windowed image profile templates (see Figure 4.8).

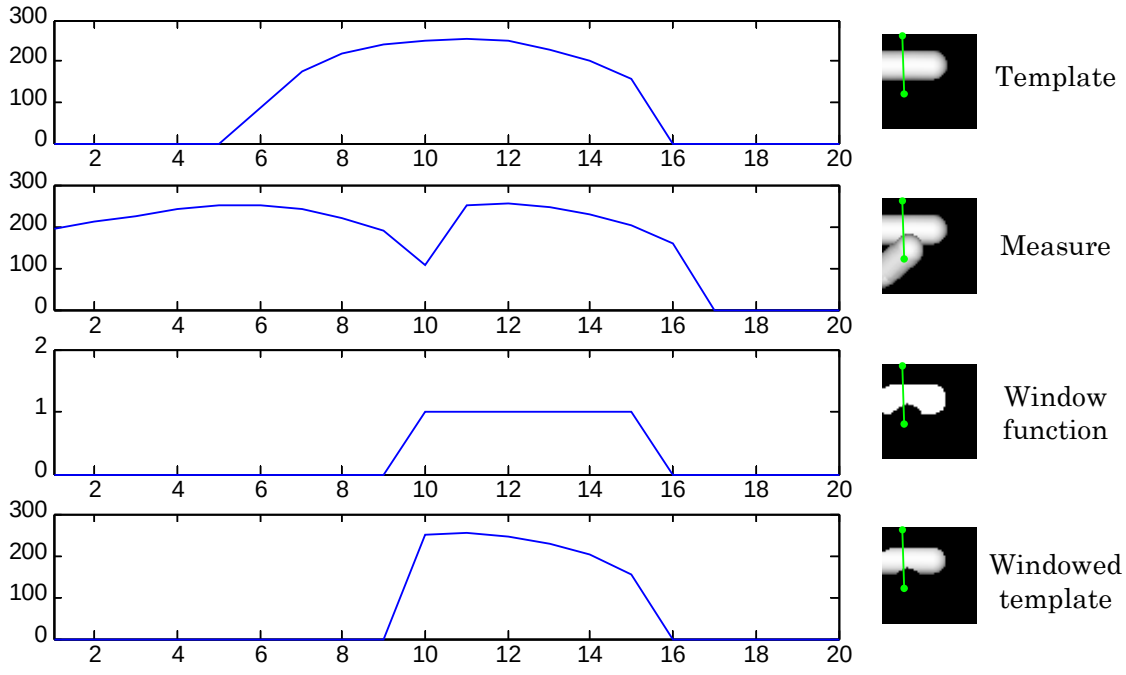


Figure 4.8 - The entities involved in windowed template matching (starting from top): a template of an unoccluded finger is taken and stored for future reference; a measure of the link profile is taken in the current frame; entity M(q) informs where the link is still visible and helps define a window function; a windowed template results from multiplying the stored template by the window function. Plots on the left represent the image profile along the green paths on the right; the x-axis is the pixel number and the y-axis the intensity value, defined as an integer between 0 and 255 with the exception of the window function which is binary.

Imagine an image profile sample $\mathbf{I}$ has an associated visibility function $\mathbf{v}$ the same size as it but with ones where the sample is visible and zeros outside. The visibility function is used to mask a profile template $\mathbf{t}$, which represents the aspect of an image profile sample when there is no occlusion. The masked template $\mathbf{m}$ is computed with $m(k)=v(k)\,t(k)$. This masked template is used on a windowed version of the SSD, only computed where $\mathbf{m}>0$ and given by

$$WSSD_{\mathbf{I},\,\mathbf{t}|\mathbf{v}}(k)=WSSD_{\mathbf{I},\,\mathbf{m}}(k)=\sum_{j\,:\,v(j)>0}\left[\mathbf{I}(j)-\mathbf{m}(j+1-k)\right]^2 .$$

The WSSD curves of a link can be grouped side by side to originate a WSSD surface.

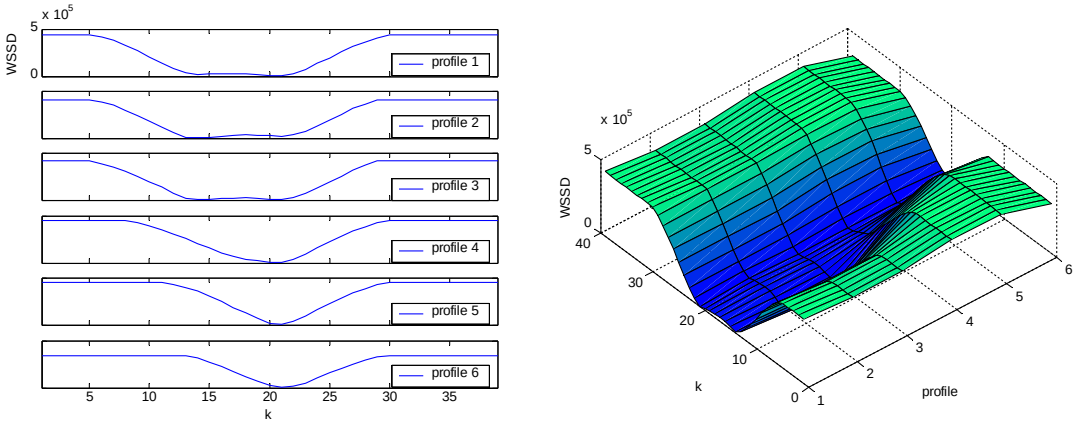


Figure 4.9 - The WSSD curves of the link profiles (left) can be combined into a single WSSD surface (right). The sample x-axis coordinate is represented by k, the units of the WSSD are squared intensity values.

4.3.2 Edge Estimation

The proposed edge estimation method resorts to *a priori* knowledge of phalanx width and to the WSSD surface. After combining the WSSD curves to form a surface (see Figure 4.9), a line is fit to the valley crossing all curves. The valley corresponds to minima of the surface which are searched individually for each curve using a gradient descent strategy. These minima are then used to estimate the line. The line indicates the offset between each template and its match, which is a measure of the deviation between model $\mathrm{M}(\mathbf{q})$ and the object in the image. These offsets are marked perpendicularly to the model to produce estimates of the whereabouts of the object,

and will be referred to as central points. The location of each pair of edge points is estimated using a central point and the width of the phalanx, so to position the edges at the same distance of the central point, along the same perpendicular used before.

Let $\mathbf{S}$ be the WSSD surface composed by a set of curves $\mathbf{s}_i$. Let $\hat{\mathbf{y}}$ be the estimated coordinates of the regression line on the y-axis (see Figure 4.10). The purpose is to find the direction of the minimum at each curve on a specified point y_i. This is done by differentiating each of the $\mathbf{s}_i$, or similarly, by computing the gradient of $\mathbf{S}$ along each column (the $\mathbf{s}_i$ are column vectors). The result is a surface $\mathbf{G}$ the same size as $\mathbf{S}$, composed of several $\mathbf{g}_i$ the same size as the $\mathbf{s}_i$. At each point these gradient curves $\mathbf{g}_i$ specify the direction of growth of the respective $\mathbf{s}_i$. To reach a minimum along any $\mathbf{s}_i$, each y_i is updated so the corresponding value on $\mathbf{S}$ has less magnitude than before. This is achieved with the update equation

$$\mathbf{y}_{k+1} = \mathbf{y}_k - sign(\tilde{\mathbf{g}}),$$

where $\tilde{\mathbf{g}}$ is a vector with the gradients of each $\mathbf{s}_i$ on a specified y_i. This is the same principle as gradient descent but with the update step restricted to the unit.

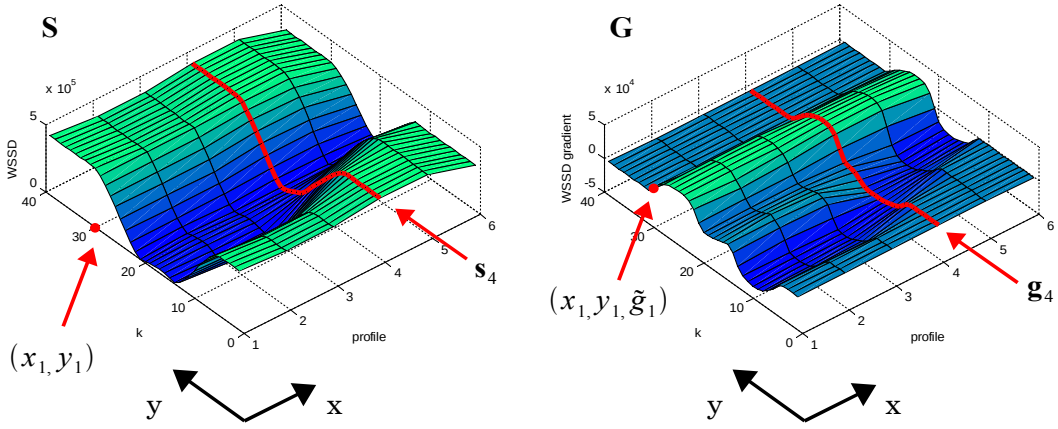


Figure 4.10 - The WSSD surface (left) and its gradient surface (right). Also featuring: the fourth WSSD and gradient curves (red curves), the first point used for regression (left red dot) and its respective gradient (right red dot).

The several y_i updates do not necessarily evolve in the same direction (see Figure 4.11), so a line is adjusted to the updates in order to estimate a collinear set of points.

This is done using a partial linear least squares (PLLS) regression, explained in detail in Appendix D, where model $y=c_0+c_1 x$ is adjusted to the (x_i, y_i). The x_i in the model are given by the column number of $\mathbf{S}$ and the y_i in the model are the updated y_i. The PLLS method determines the line which fits the data. However, the coordinates of the line at each point must be integers, so it is necessary to round the $\hat{y}_i$ towards the nearest integer.

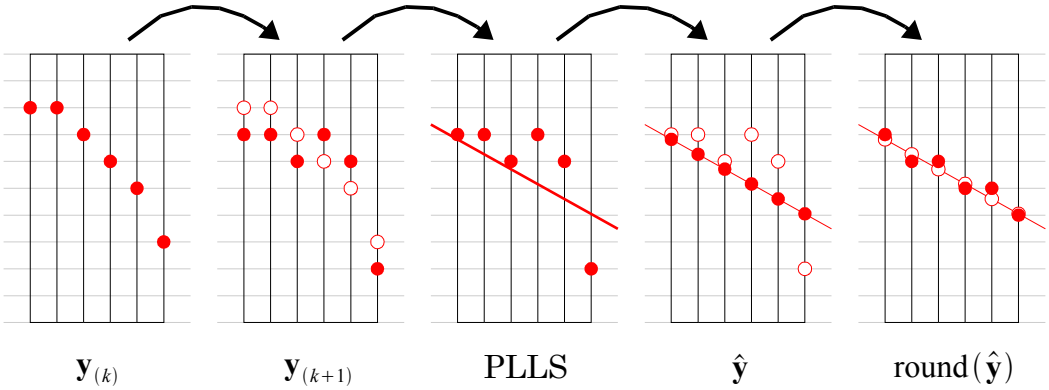


Figure 4.11 - The steps of the algorithm used to fit a line to the valley of the WSSD surface. Each grid is a top view of the WSSD surface. The initial data is updated by an unit considering the gradient of the WSSD curves; a line is adjusted to the updated data; the estimated data is placed over the line; the estimated data is rounded toward the nearest integer. Red dots indicate the result of using the current operation on red circles (data carried from last step).

The process is then repeated until the rounded $\hat{y}_i$ converge or a certain number of iterations is reached.

The orientation and position of the valley are defined by the $(x_i, \hat{y}_i)$. These points are then used to estimate the offset of each template. The offset corresponds to the deviation of $\hat{y}_i$ relatively to the centre of $\mathbf{s}_i$. This offset is marked relatively to the centre of the profile template and the width of the phalanx is used to extrapolate the locations of the edges. Those locations are then converted back to image coordinates so they can be used by the line fitting algorithm.

4.3.3 Visibility Ordering

The third component of the edge estimation process is the computation of the windowing functions used in the WSSD. In order to determine these functions it is necessary to know how each phalanx template is layered towards the camera.

Visibility ordering is the process responsible for determining that layering. There is more than one way of doing this. Some methods are more parametric and accurate than others. In [2], for instance, this problem is extensively studied and solved in a parametric way. However, the visibility ordering method now introduced is based on a very simple concept: it needs information about the 3D shape of the phalanges and a state vector to produce an estimate of the region each phalanx occupies in the image.

Each of these estimates is ordered based on the distance their 3D counterparts are from the camera. After ordering, the polygonal regions regarding occluded phalanges are submitted to Boolean operations where the occluding polygonal regions are subtracted to it. This originates truncated polygons which manifest the occlusion taking place and can be used to define windowing functions, so the SSD is only computed with the visible portion of a template. The windowing function for an image profile sample $\mathbf{I}$ is easily computed determining which sample points of $\mathbf{I}$ fall inside the truncated polygon.

Picture a situation where one finger occludes another. Each finger has three phalanges which are modelled by a convex 3D shape, like a cylinder toped with hemispheres.

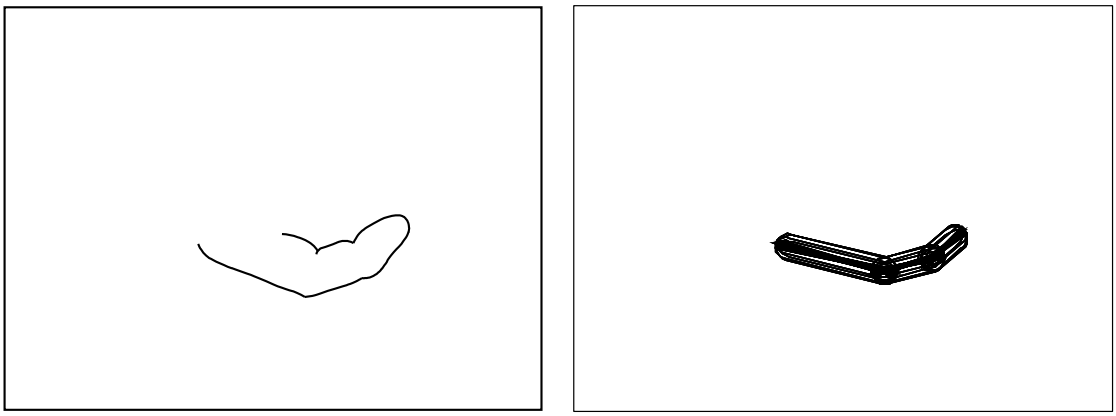


Figure 4.12 - Side view of a finger (left) and corresponding model formed by 3D pill primitives (right).

The image coordinates of each vertex of these 3D shapes is found with a process described in Appendix C. This process also preserves information about the distance of each 3D vertex to the camera, which will be used in depth sorting.

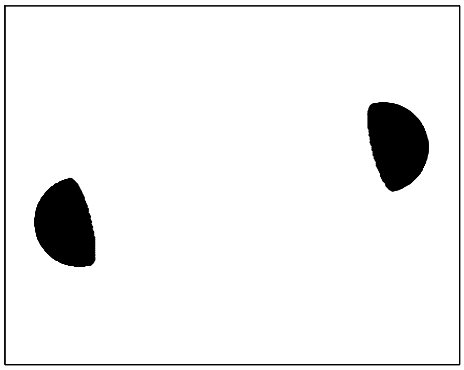


Figure 4.13 - Cloud of points formed by vertices of a 3D pill primitive projected on the image frame (only the vertices of the hemispheres are needed).

After each shape is projected on the image its convex hull is computed as described in Appendix C. This indicates the outer points of each projected 3D shape, thus enabling it being described by a convex polygon. Redundant points in each convex polygon are eliminated by a method also described in Appendix C, leading to identical polygons defined with less vertices.

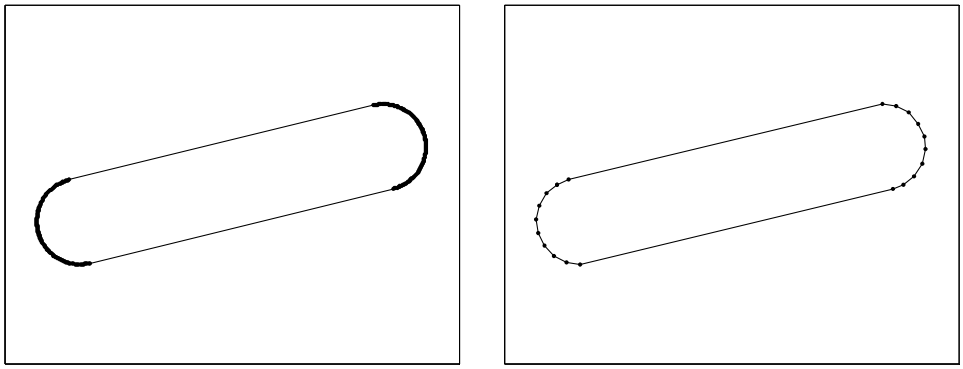


Figure 4.14 - Convex hull of projected 3D pill primitive (left) and its reduced version (right).

Since depth information is preserved it is possible to know the distance of each peripheral point to the camera. This enables determining which polygons are in front of the others.

Depth sorting is the process responsible for ordering polygons in layers according to their sequence towards the camera. If a polygon is in front of another, from a camera point of view, then it will be placed in a layer closer to the viewer.

The attribution of layers to each polygon is based on a coarse depth sorting strategy, which is nonetheless more than enough for most situations. Each polygon has an associated depth computed as the average depth of its vertices.

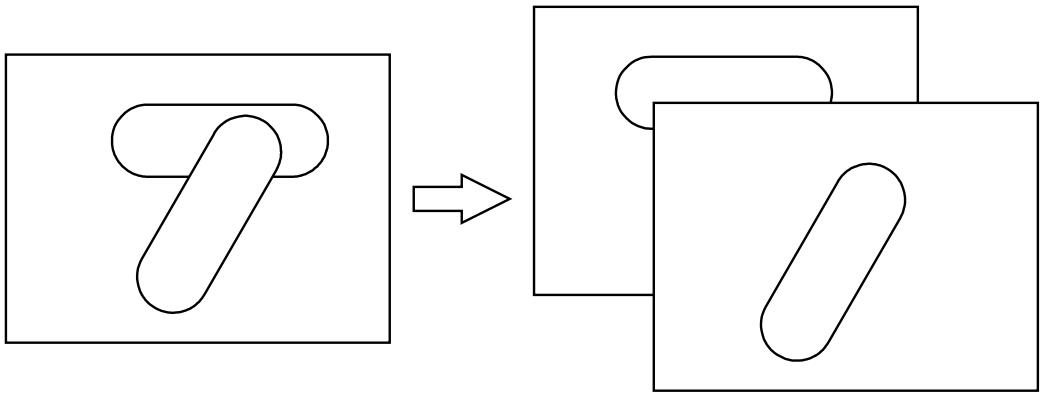


Figure 4.15 - A depth sorting strategy is used to attribute a layer to each polygon, ordering them from the farthest away to the closest one.

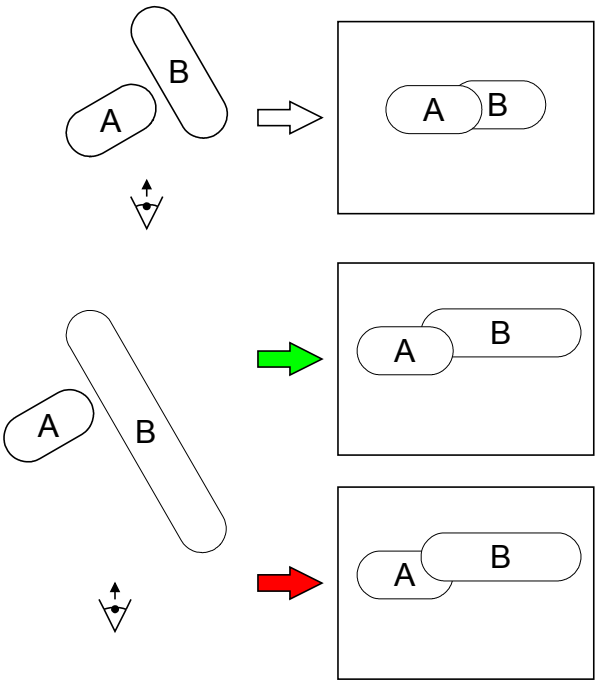


Figure 4.16 - The viewer sees shape B occluded by shape A. Depth sorting works fine (white arrow situation) if the average depth of the occluded shape B is greater than that of the occluding shape A. However, if the average depth of B is less than that of A, B will be incorrectly considered the occluding shape (red arrow situation) when it should be the occluded one (green arrow situation).

The polygons can then be ordered based on their average depth. This will work just fine as long as the depth of the occluded polygon does not indicate it should be in front of the occluding polygon. Of course this method works better if the main direction of

the 3D shapes is on a plain parallel to the image plain.

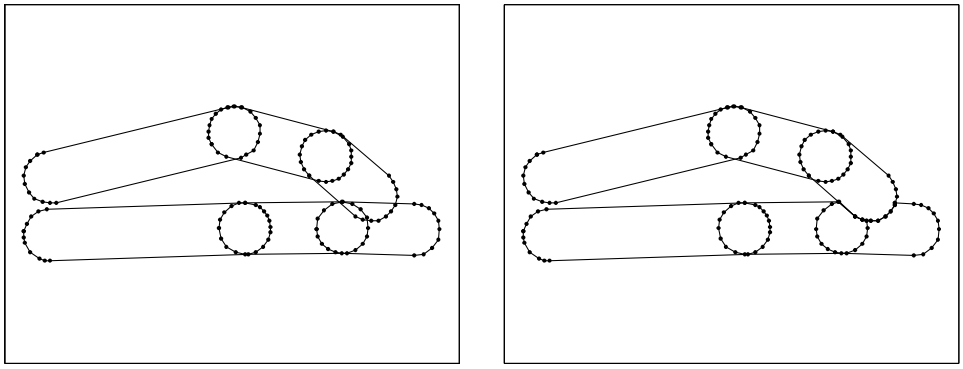


Figure 4.17 - The polygons on the left are depth sorted and subject to Boolean subtractions originating the truncated polygons on the right, which will be used to define the visibility functions. The top finger occludes the bottom one, hence the bottom finger polygons are truncated. Phalanges of the same finger not considered for occlusion purposes.

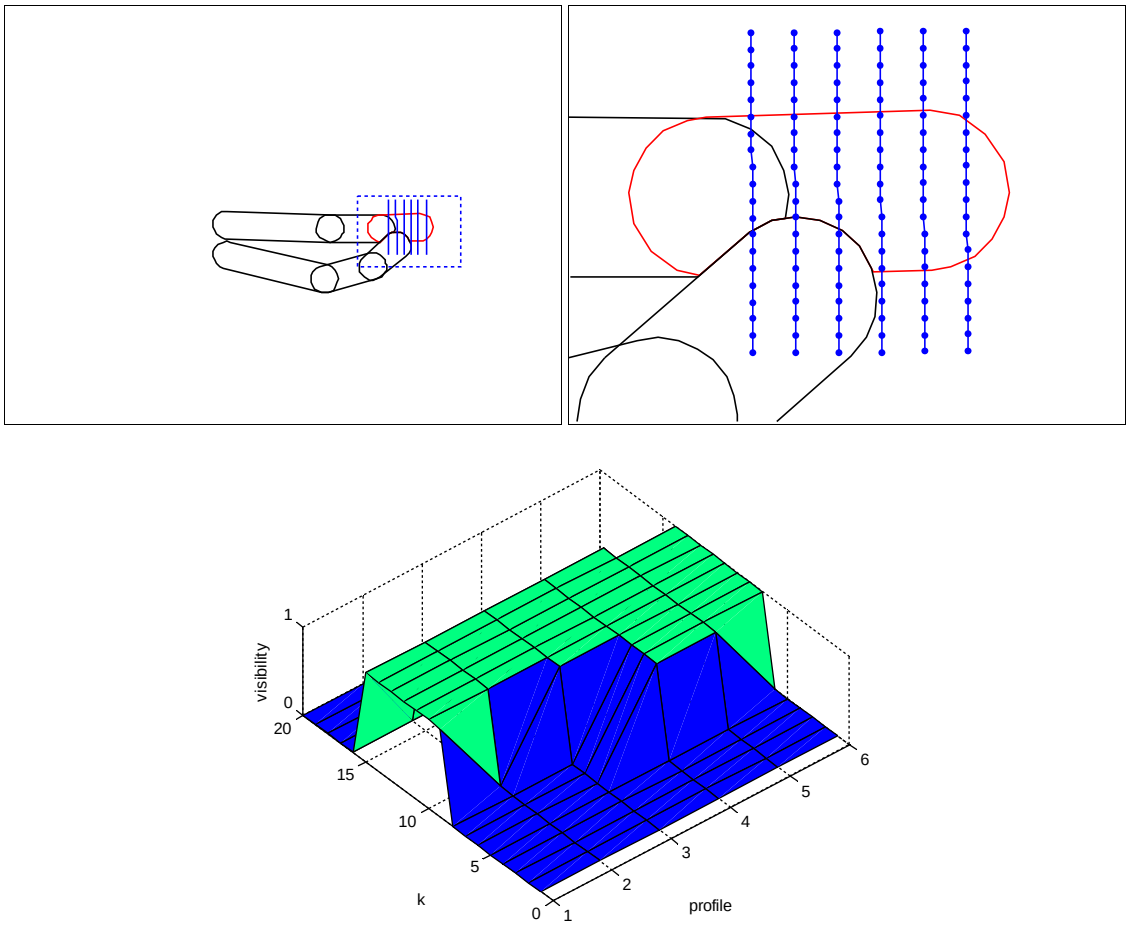


Figure 4.18 - Defining window functions with the help of truncated polygons. Left: truncated polygons and link samplers (blue segments). Right: detail of region on left, presenting the sampling points. Bottom: window function surface has ones where sampling points fall inside the truncated red polygon and zeros outside.

Once each polygon is depth sorted it is necessary to eliminate the occluded regions of the occluded polygons. This produces other polygons which could be flattened into the same layer and still convey where occlusions take place. Starting with the farthest away polygon, each polygon is cut where the ones in front occlude it. This is done with a Boolean subtraction of polygons.

The visibility functions result from finding which sampling points of the image profile samples fall within the truncated polygons. Remember each link has an associated truncated polygon and associated image profile samples. The visibility functions present ones inside the truncated polygon and zeros outside.

Usually the determination of visibility functions is performed at the end of each frame, after the pose is estimated and the pose prediction for the next frame is made. These visibility functions are then used in the beginning of the next frame, where the image profiles are sampled.

4.4 Line Fitting

The fourth and last step in the image analysis process is to fit lines to each set of extracted boundary points in order to estimate the axial lines which describe the orientation of the phalanges in the image. This line fitting process is something completely different from the line fitting process used in edge estimation. While in edge estimation the goal was to parameterize the valley in the WSSD surface, here the goal is to parameterize the sides of the phalanges to later parameterize the orientation of the phalanges.

After extracting contour points from the image it is necessary to fit lines to the sets of points regarding the sides of the phalanges. The line fitting process assumes these contour points are correctly divided into groups of points sharing the same characteristics, such as being on the same side of a phalanx.

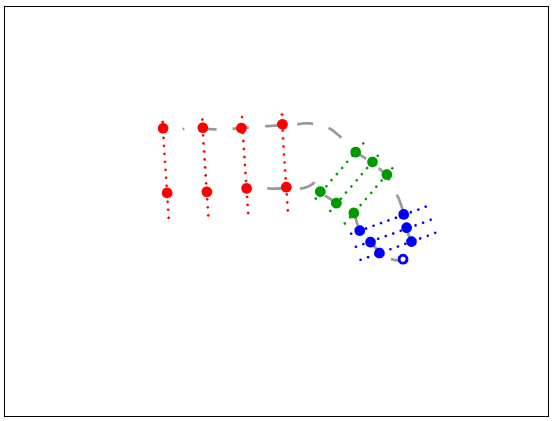


Figure 4.19 - The contour points used in the line fitting algorithm are divided into groups, sharing same link (same colour) and same side of a phalanx. The dotted segments represent the link samplers and the blue circle the finger tip.

The line fitting routine fits a line to each of these groups of points using a deterministic variation of the RANSAC (Random Sampling Consensus [22]) algorithm. Whereas RANSAC (Figure 4.20) randomly selects pairs of points, fits a line to the pair, and refines the best configuration considering the points which lie within a certain distance from the line (the inliers), this variation deterministically goes over every possible pair of points to determine which one leads to more inliers. This is considered an acceptable procedure since the number of points to fit is very small, usually not more than six per group. In the present case, the Euclidean distance is used to measure the distance between a line and a point.

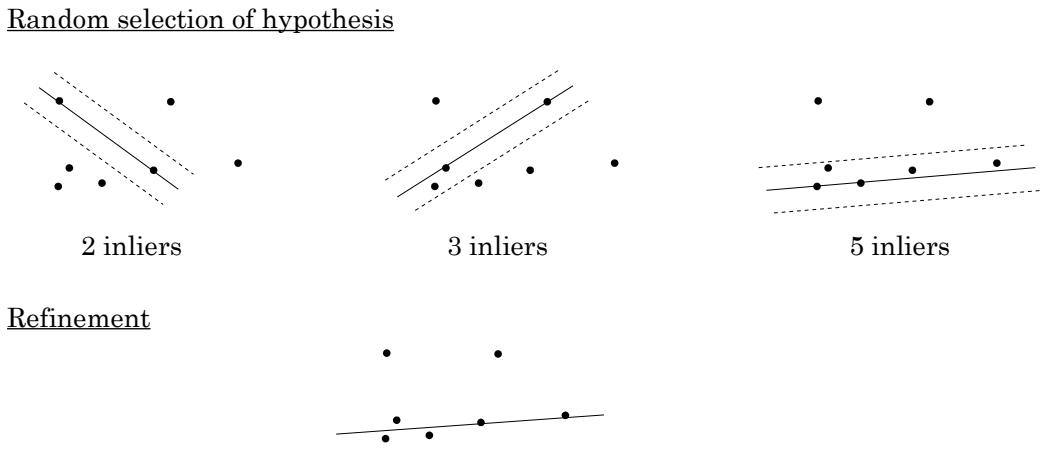


Figure 4.20 - The RANSAC (Random Sampling Consensus) algorithm: random selection of pairs of points searching for the configuration with more inliers (top); refinement of the best configuration (bottom).

The line fitting algorithm takes as argument a group **S** of n similar points,

arranged in a matrix as

$$\mathbf{S}=\begin{bmatrix} S_1^x & \cdots & S_n^x \\ S_1^y & \cdots & S_n^y \end{bmatrix},$$

where each column indicates the image coordinates of a point. This structure makes it easier to determine the line.

Then, it does an exhaustive search fitting a line to each pair of points and determining the inliers. A point is considered an inlier if it is closer than a distance τ from the line.

After determining which set of points leads to more inliers, it is necessary to refine that choice. This is achieved with a minimization of a total least squares problem, or similarly, performing a principal component analysis (PCA) [23] of the inliers scattering. First, the m inliers are stored in a $(2\times m)$ matrix arranged as

$$\mathbf{Y}=\begin{bmatrix} Y_1^x & \cdots & Y_m^x \\ Y_1^y & \cdots & Y_m^y \end{bmatrix},$$

so that its (2×2) covariance matrix $\mathbf{R}_{\mathbf{YY}}$ can be computed. By definition

$$\mathbf{R}_{\mathbf{YY}}=\mathrm{E}\left[\left(\mathbf{Y}-\boldsymbol{\mu}_{\mathbf{Y}}\right)\left(\mathbf{Y}-\boldsymbol{\mu}_{\mathbf{Y}}\right)^T\right]\propto\left(\mathbf{Y}-\boldsymbol{\mu}_{\mathbf{Y}}\right)\left(\mathbf{Y}-\boldsymbol{\mu}_{\mathbf{Y}}\right)^T,$$

where the proportionality factor can be $1/(m-1)$ or $1/m$ depending on the definition used. Either way, the refinement will stand no matter the factor used. Afterwards, the eigenvalues and eigenvectors of $\mathbf{R}_{\mathbf{YY}}$ are computed, since the main direction corresponds to the eigenvector whose eigenvalue has the largest magnitude.

Once the eigenvalues and eigenvectors are determined, the refined fitting line is defined as passing by point $\mathbf{p}=\mathrm{E}\left[\mathbf{Y}_\mathbf{i}\right]$, the centre of mass of the inliers, with an orientation parallel to $\mathbf{v}$, the eigenvector with the largest eigenvalue.

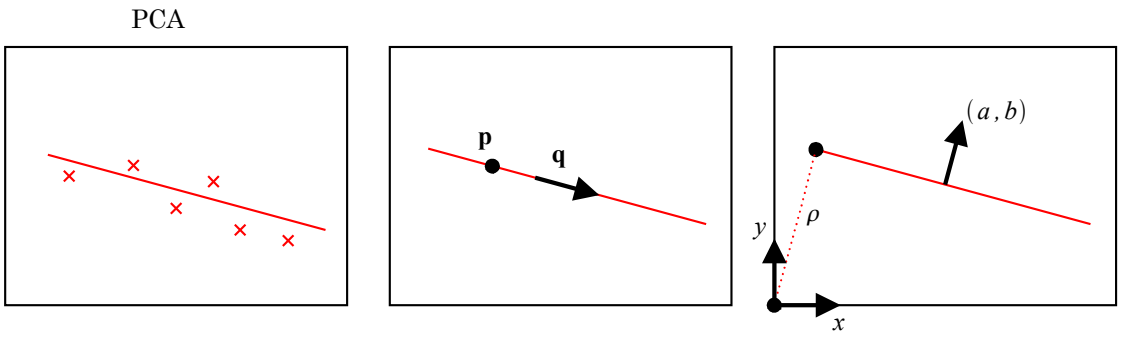


Figure 4.21 - After the PCA fits a line to a set of points of the same group (left), that line can be described by a point and a line (middle) or by three line parameters (right).

Remember these lines are adjusted to boundary points of the finger phalanges. However the pose estimation algorithm requires axial lines describing the orientation of the finger phalanges. Thus, it is necessary to convert each pair of boundary lines to a link feature representing the link axial line.

In practice, the axial line can be described by parameters $[a_{ax} \quad b_{ax} \quad \rho_{ax}]$ or by a point $\mathbf{p}_{ax}$ and a vector $\mathbf{v}_{ax}$ parallel to the line. It was concluded to be easier to determine $\mathbf{p}_{ax}$ and $\mathbf{v}_{ax}$ first and then convert the result to $[a_{ax} \quad b_{ax} \quad \rho_{ax}]$.

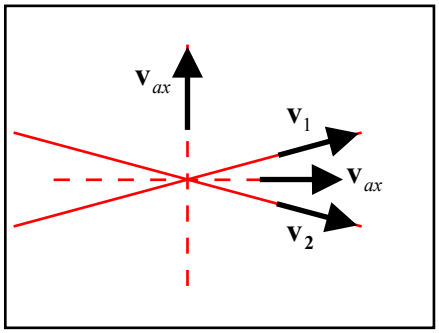


Figure 4.22 - The direction of the axial line is chosen as the one making the least of two angles with the peripheral lines. In the depicted case, the horizontal axial line would be chosen since it makes a smaller angle than the vertical axial line.

Given any two boundary lines with unitary direction vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, generally not parallel, there are two possible solutions for the direction of $\mathbf{v}_{ax}$, always perpendicular to each other. The goal is having the axial line making the least of two possible angles with the boundary lines. The conclusion is made based upon the following criterion: if $\|\mathbf{v}_1+\mathbf{v}_2\| \geq \|\mathbf{v}_1-\mathbf{v}_2\|$ then $\mathbf{v}_{ax}=(\mathbf{v}_1+\mathbf{v}_2)/2$, else $\mathbf{v}_{ax}=(\mathbf{v}_1-\mathbf{v}_2)/2$. The resulting $\mathbf{v}_{ax}$ is generally not unitary, but computing it this way (with unitary vectors)

is easier and faster than resorting to trigonometric functions to determine each of the angles of the boundary vectors, more algebra to determine the correct angle for the axial line and yet more trigonometry to compute the components of $\mathbf{v}_{ax}$.

As for point $\mathbf{p}_{ax}$, a point in the axial line, its computation is slightly more complex and the computational precision must be taken in account to avoid wrong results, especially when the boundary lines are close to being parallel (see Figure 4.23). Given two boundary lines defined by points $\mathbf{p}_1$ and $\mathbf{p}_2$ and with direction vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, generally not parallel, the goal is to determine the point of intersection $\mathbf{p}_{ax}$ which satisfies

$$\mathbf{p}_1 + k_1 \mathbf{v}_1 = \mathbf{p}_2 + k_2 \mathbf{v}_2 = \mathbf{p}_{ax}, \tag{4.2}$$

where k_1 and k_2 are multiplicative factors to be determined. The solution is easier to compute if the first equality of (4.2) is rewritten as

$$k_1 \mathbf{v}_1 - k_2 \mathbf{v}_2 = (\mathbf{p}_2 - \mathbf{p}_1) \Leftrightarrow \begin{bmatrix} v_1^x & -v_2^x \\ v_1^y & -v_2^y \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = (\mathbf{p}_2 - \mathbf{p}_1)$$

in order to form a matricial system like

$$\mathbf{k} = \mathbf{V}^{-1}(\mathbf{p}_2 - \mathbf{p}_1), \tag{4.3}$$

where $\mathbf{k}$ is the vector of multiplicative factors and $\mathbf{V}$ is the square matrix formed by $\mathbf{v}_1$ and $-\mathbf{v}_2$. After this, point $\mathbf{p}_{ax}$ can be found substituting either k_1 or k_2 in (4.2).

However, when the boundary lines are almost parallel (4.3) cannot be used. From the geometric point of view, it means an intersection on an infinite point. From the computational point of view, the inversion of $\mathbf{V}$ may result inaccurate if the matrix is nearly singular. To avoid such results, (4.3) is only carried out if the magnitude of the determinant is over 10^{-9}. Also, the determinant of $\mathbf{V}$ has the same expression as the dot product of $\mathbf{v}_1$ by a vector perpendicular to $\mathbf{v}_2$, so when the boundary lines are close to parallel this dot product is very close to zero. When the boundary lines are

almost parallel, $\mathbf{p}_{ax}$ is alternatively computed with

$$\mathbf{p}_{ax}=(\mathbf{p}_1+\mathbf{p}_2)/2\ , \tag{4.4}$$

which is fairly reasonable since when two lines are parallel any $\mathbf{p}_1$ and $\mathbf{p}_2$ have an equally distant point $\mathbf{p}_{ax}$ over the axial line.

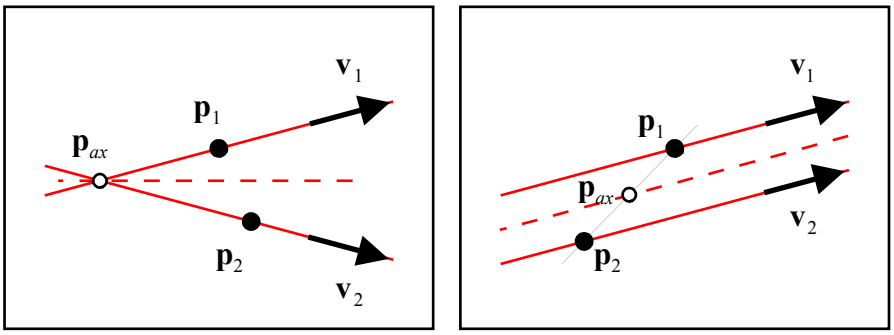


Figure 4.23 - Determination of a point belonging to the axial line depends on the parallelism of the peripheral lines. When peripheral lines are not parallel (left) the point of the axial line is the point of intersection of the two lines. When the lines are parallel (right) the point of the axial line is the geometric mean of the points of the peripheral lines.

To convert the axial line parameterization from point-vector notation $(\mathbf{p}_{ax}, \mathbf{v}_{ax})$ to the three element notation $[a_{ax} \quad b_{ax} \quad \rho_{ax}]$, it's only necessary to understand that in the general line equation $a x+b y-\rho=0$, (a, b) is a vector normal to the line and ρ is the distance from the line to the origin of the reference frame.

4.5 Chapter Outcome

By now, the three fundamental parts of the proposed pose estimation method have been explained. This chapter has explained how to extract data from an image to use with the pose estimation algorithm. This data corresponds to lines aligned with the finger phalanges and points indicating the position of the finger tips.

5 Temporal Integration

The measurement and estimation of a quantity generally has an associated uncertainty. The source of uncertainty lies in many aspects [24][25]:

- No mathematical model is perfect. Any such model depicts only those characteristics of direct interest to the problem in question. Even effects which are modelled are necessarily approximated by a mathematical model;
- Dynamic systems are driven not only by control inputs but also by disturbances which can neither be controlled nor modelled deterministically;
- Sensors do not provide perfect and complete data about a system. First, they generally do not provide all the information one would like to know. In other situations, a number of different devices yield functionally related signals, and one must then ask how to generate a best estimate of the variables of interest based on partially redundant data. Sensors do not provide exact readings of desired quantities, but introduce their own system dynamics and distortions as well. Furthermore, these devices are always noise corrupted.

In the present situation image noise cannot be ignored neither the fact both the image domain and intensity are discrete, thus introducing some quantization errors, even if neglectable. On top of this, there is the process of retrieving information from an image, which is achieved by subsampling an image region. Then there are sources of uncertainty, mainly due to estimation routines such as hand pose estimation and the existence of roundings towards integers such as in edge detection and line fitting.

All these factors contribute to an overall uncertainty and error on the hand pose estimation problem.

This uncertainty is often visible when the temporal evolution of any estimated parameter is plotted, usually showing in the form of a curve with high frequency jitter of low amplitude.

If the way this parameter is allowed to evolve with time is parameterizable, and the typical measurement noise is quantifiable, then it may be possible to design a filter which takes these assumptions and the measurements to produce estimates with less associated error.

One such filter is the Kalman filter, which will be used to refine the estimated state vector of the present frame and to predict the state vector of the next frame. Appendix E shows an additional situation where the Kalman filter is used to determine the phalanx width.

5.1 Kalman Filter

The Kalman filter provides a computationally efficient recursive solution of the least squares state estimation. The filter is very powerful in several aspects: it supports estimations of past, present and even future states, and it can do so even when the precise nature of the modelled system is unknown.

The filter has many different variations depending on the desired application, however, they all have a common background. First, the basic Kalman filter is presented as an introduction to the estimation of a stationary process. After that, a more elaborate Kalman filter capable of dealing with position, velocity and acceleration is described.

5.1.1 The Discrete Kalman Filter

This section describes the filter in its original formulation [26], where the measurements occur and the state is estimated at discrete points in time. For a more direct introduction to the basic concepts of the Kalman filter check [24] or [25] for a hands-on approach with some examples and practical applications.

The Process to be Estimated

The Kalman filter addresses the problem of estimating the state $\mathbf{x} \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\mathbf{x}_k = \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{B}\,\mathbf{u}_k + \mathbf{w}_{k-1} ,$$

where the $(n \times n)$ matrix $\mathbf{A}$ relates the state at the previous time step $k-1$ to the state at the current step k in the absence of either a driving function or process noise; the $(n \times l)$ matrix $\mathbf{B}$ relates the optional control input $\mathbf{u}_k \in \mathbb{R}^l$ to the state $\mathbf{x}$; the random variable $\mathbf{w}_k$ represents the process noise.

This process is observed with a measurement $\mathbf{y} \in \mathbb{R}^m$ described by

$$\mathbf{y}_k = \mathbf{H}\,\mathbf{x}_k + \mathbf{v}_k ,$$

where the $(m \times n)$ matrix $\mathbf{H}$ relates the state to measurement $\mathbf{y}_k$ and $\mathbf{v}_k$ is a random variable representing measurement noise.

In general, matrices $\mathbf{A}$ and $\mathbf{H}$ may change over time, but here are assumed to be constant.

The process and measurement noises are assumed to be independent of each other, white and with normal probability distributions

$$\mathrm{p}(\mathbf{w}) \sim \mathrm{N}(0, \mathbf{Q})$$
$$\mathrm{p}(\mathbf{v}) \sim \mathrm{N}(0, \mathbf{R})$$

where the noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ might change with each time step but are also assumed to be constant, for convenience.

The Discrete Kalman Filter Algorithm

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the Kalman filter fall into two groups:

time update equations and measurement update equations. The time update equations are responsible for projecting forward in time the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback: incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

After algebraic manipulation [25], the discrete Kalman filter prediction equations are given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}\,\hat{\mathbf{x}}_{k-1} + \mathbf{B}\,\mathbf{u}_k$$
$$\mathbf{P}_{k|k-1} = \mathbf{A}\,\mathbf{P}_{k-1}\,\mathbf{A}^T + \mathbf{Q}$$

where $\hat{\mathbf{x}}_{k|k-1}$ is the prediction for the next state; $\mathbf{P}_{k|k-1}$ is the *a priori* and $\mathbf{P}_{k-1}$ the *a posteriori* estimate error covariances, with

$$\mathbf{P}_k = \mathrm{E}\left[\mathbf{e}_k\,\mathbf{e}_k^T\right]^T,$$

where

$$\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k .$$

The filter starts off from this point using initial conditions for $\hat{\mathbf{x}}_{k-1}$ and $\mathbf{P}_{k-1}$.

On the other hand, the discrete Kalman filter measurement update equations are given by

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\,\mathbf{H}^T\left(\mathbf{H}\,\mathbf{P}_{k|k-1}\,\mathbf{H}^T + \mathbf{R}\right)^{-1}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}\,\hat{\mathbf{x}}_{k|k-1}\right)$$
$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\,\mathbf{H}\right)\mathbf{P}_{k|k-1}$$

where the first task is to compute the Kalman gain $\mathbf{K}_k$. The next step is to actually measure the process to obtain $\mathbf{y}_k$, and then to generate an *a posteriori* state estimate $\hat{\mathbf{x}}_k$ by incorporating the measurement. The final step is to obtain an *a posteriori* error covariance $\mathbf{P}_k$.

After each time (prediction) and measurement (correction) update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter.

5.1.2 The g-h-k Kalman Filter

In the previous Kalman filter, the Kalman gain decreases with time. This induces the filter to ignore the measurements at a later time step and to rely mostly on its history. Besides this, it also requires a large number of parameters to be tuned.

The g-h-k Kalman filter [27] is a way of accounting for first and second order aspects while estimating the process, which makes it suitable for tracking a target having a constant acceleration. In such a case, the target equations of motion become

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \dot{\mathbf{x}}_{k-1} T + \ddot{\mathbf{x}}_{k-1} \frac{T^2}{2}$$

$$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}_{k-1} + \ddot{\mathbf{x}}_{k-1} T$$

$$\ddot{\mathbf{x}}_k = \ddot{\mathbf{x}}_{k-1} .$$

But the novelty in this new filter formulation, is the fact that the gains are constant over time, so the filter always makes the same compromise between new measurements and history.

To avoid a 3-fold increase in the complexity of the filter, due to inclusion of velocity and acceleration, the formulation of the equations follows a more heuristical rather than probabilistical approach. The tracking equations needed for updating the prediction estimates of position, velocity and acceleration for the constant-accelerating target model become

$$\hat{\ddot{\mathbf{x}}}_k = \hat{\ddot{\mathbf{x}}}_{k|k-1} + \frac{2k}{T^2}\left(\mathbf{y}_k - \hat{\mathbf{x}}_{k|k-1}\right) \tag{5.1}$$

$$\hat{\dot{\mathbf{x}}}_k = \hat{\dot{\mathbf{x}}}_{k|k-1} + \frac{h}{T}\left(\mathbf{y}_k - \hat{\mathbf{x}}_{k|k-1}\right) \tag{5.2}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + g\left(\mathbf{y}_k - \hat{\mathbf{x}}_{k|k-1}\right) \tag{5.3}$$

where the form of the position update clearly resembles that of the previous filter.

The g-h-k prediction equations then become

$$\hat{\ddot{\mathbf{x}}}_{k|k-1} = \hat{\ddot{\mathbf{x}}}_{k-1} \tag{5.4}$$

$$\hat{\dot{\mathbf{x}}}_{k|k-1} = \hat{\dot{\mathbf{x}}}_{k-1} + \hat{\ddot{\mathbf{x}}}_{k-1} T \tag{5.5}$$

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1} + \hat{\dot{\mathbf{x}}}_{k-1} T + \hat{\ddot{\mathbf{x}}}_{k-1} \frac{T^2}{2} \tag{5.6}$$

This filter relates with the $\alpha-\beta-\gamma$ filter when g, h and k are respectively replaced by α, β and $\gamma/2$. This filter has the advantage that it can track a constant-accelerating target with zero lag error in steady-state. It will have a constant lag error for a target having a constant jerk (derivative of the acceleration). It is a three state filter, tracking position, velocity and acceleration.

In the previous filter it was necessary to tune up two noise covariances and set two initial conditions per parameter in the state vector. In the present filter, due to the heuristical formulation, it is necessary to tune up three gains and three initial conditions per parameter, instead of two noise covariances and four initial conditions.

Critically Damped Filter

However, it is possible to define g, h and k as functions of a single parameter θ. The critically damped g-h-k filter represents the filter minimizing the discounted least-squares error for a constantly-accelerating target, with gains defined by

$$g = 1-\theta^3$$

$$h = 1.5(1-\theta^2)(1-\theta)$$

$$k = 0.5(1-\theta)^3$$

where θ is the discounting factor given by

$$0 \leq \theta \leq 1 .$$

The discounted least-squares fit is a line fit which minimizes the sum of the weighted errors with the weighting decreasing as the data get older; that is, the older the error, the more it is discounted.

This way it is only necessary to tune up one discounting parameter and set three initial conditions per parameter in the state vector.

5.2 Pose Estimate Refinement and Prediction

The state vector estimates determined with the pose estimation algorithm have an associated uncertainty. When the temporal evolution of the pose estimates is overlaid on the analysed image sequence this uncertainty is visible as a sort of trembling, which relates directly to the jitter in the estimates. If this uncertainty was not susceptible to corrupting future estimates, the jitter could be attenuated applying a post processing low-pass filter on the estimates. However, each estimate starts from an initial condition related with the previous estimate, so if an estimate is corrupted (which is to say it has an intolerable uncertainty), future estimates are bound to be corrupted too.

The Kalman filter offers the possibility of refining the estimates, and thus reducing the uncertainty, in real-time with the determination of estimates rather than on a post-processing level. Furthermore, it offers the possibility of including the dynamics of the process in the filtering process, which will in part account for disregarding estimates which disrespect those dynamics. By reducing the uncertainty at each frame there is a stronger probability of obtaining a more accurate and jitter-free estimate.

To avoid confusing entities, when dealing with the Kalman filter the estimates of the pose estimation algorithm will be the measurements of the filter, while the refined estimates are the outputs of the filter. The filter merely takes the present measurements and the recent predictions and refines the measurements producing refined estimates. This is done by simply replacing $\mathbf{y}_n$ in the correction equations

(5.1), (5.2) and (5.3) with the estimated state vector $\mathbf{q}$, yielding

$$\hat{\ddot{\mathbf{x}}}_k = \hat{\ddot{\mathbf{x}}}_{k|k-1} + \frac{2k}{T^2}\left(\mathbf{q}_k - \hat{\mathbf{x}}_{k|k-1}\right)$$

$$\hat{\dot{\mathbf{x}}}_k = \hat{\dot{\mathbf{x}}}_{k|k-1} + \frac{h}{T}\left(\mathbf{q}_k - \hat{\mathbf{x}}_{k|k-1}\right)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + g\left(\mathbf{q}_k - \hat{\mathbf{x}}_{k|k-1}\right).$$

Besides the possibility of refining the estimates, a Kalman filter can also be used to predict the next state vector, since it uses a model of the dynamics. A prediction based on dynamics is likely to be closer to the pose in the next frame than the state vector estimated with the pose estimation algorithm. This is important because the closer an initial estimate of the algorithm is to the real pose in the image the better, since the extracted data is more likely to refer exclusively to the regions of interest and not to neighbour regions.

5.3 Chapter Outcome

This chapter shows a way of reducing uncertainty in the estimates provided by the pose estimation algorithm. It also shows how to predict the pose in the next frame using history. Both these aspects are achieved using the g-h-k Kalman filter, which is a constant gain variation of the standard Kalman filter.

There are no graphics comparing the performance using and not using the Kalman filter, basically because the estimates in one frame are related with the history of estimates. It was considered not to be coherent to compare estimates of the same time instant when the previous estimates are not the same, which inevitably affects the initial conditions in the current frame. Besides, the Kalman filter is used more as a safety measure against sudden errors or changes in the estimates, which can be caused by a bad convergence in the pose estimation algorithm or a bad extraction of features.

6 Results

To understand the capabilities of the tracking method, tests were performed in several stages of development until the proposed method was finished. Three of those tests were selected to integrate the results chapter. They portray tracking with synthetic and real images, using the proposed method and an alternative one, with articulated models of different complexities.

6.1 Proposed Method

i	α_{i-1}	a_{i-1}	d_i	θ_i	***tip***	***link***	***point***	***next***	***prev.***	***Notes***		
										frame	***link***	***finger***
0	-	-	-	-	-	-	-	1, 6	-	Base		
1	π	a_0	0	$-\pi/2$	-	0	-	2	0	Aux.		
2	0	a_1	0	0	-	0	-	3	1	1st	1st	middle
3	$-\pi/2$	0	0	θ_3	-	1	x	4	2	2nd		
4	0	a_3	0	θ_4	-	1	x	5	3	3rd	2nd	
5	0	a_4	0	θ_5	A_5	1	x	-	4	4th	3rd	
6	π	a_5	0	$-\pi/2$	-	0	-	7	0	Aux.		
7	0	a_6	0	0	-	0	-	8	6	1st	1st	index
8	$-\pi/2$	0	0	θ_8	-	1	x	9	7	2nd		
9	0	a_8	0	θ_9	-	1	x	10	8	3rd	2nd	
10	0	a_9	0	θ_{10}	A_{10}	1	x	-	9	4th	3rd	

Table 6.1 - DH parameters and extra notes for an articulated model of a hand with two fingers (middle and index), three DOF each. “tip” indicates the distance to the tip, “link” if there is a rigid body (1) or not (0) attached to the frame, “point” the joint frame axis with which to align the body, “next” and “prev.” the surrounding frames.

These results represent the performance of tracking a human hand using the proposed method on a sequence of synthetic images. The choice of using synthetic images

instead of real images is simple: in synthetic images the real value of the joints is known, so the performance of the method can be better evaluated. Also, it was concluded that synthetic images portray reasonably enough what would happen with real images, provided that the choices and assumptions made in the introduction chapter are still considered.

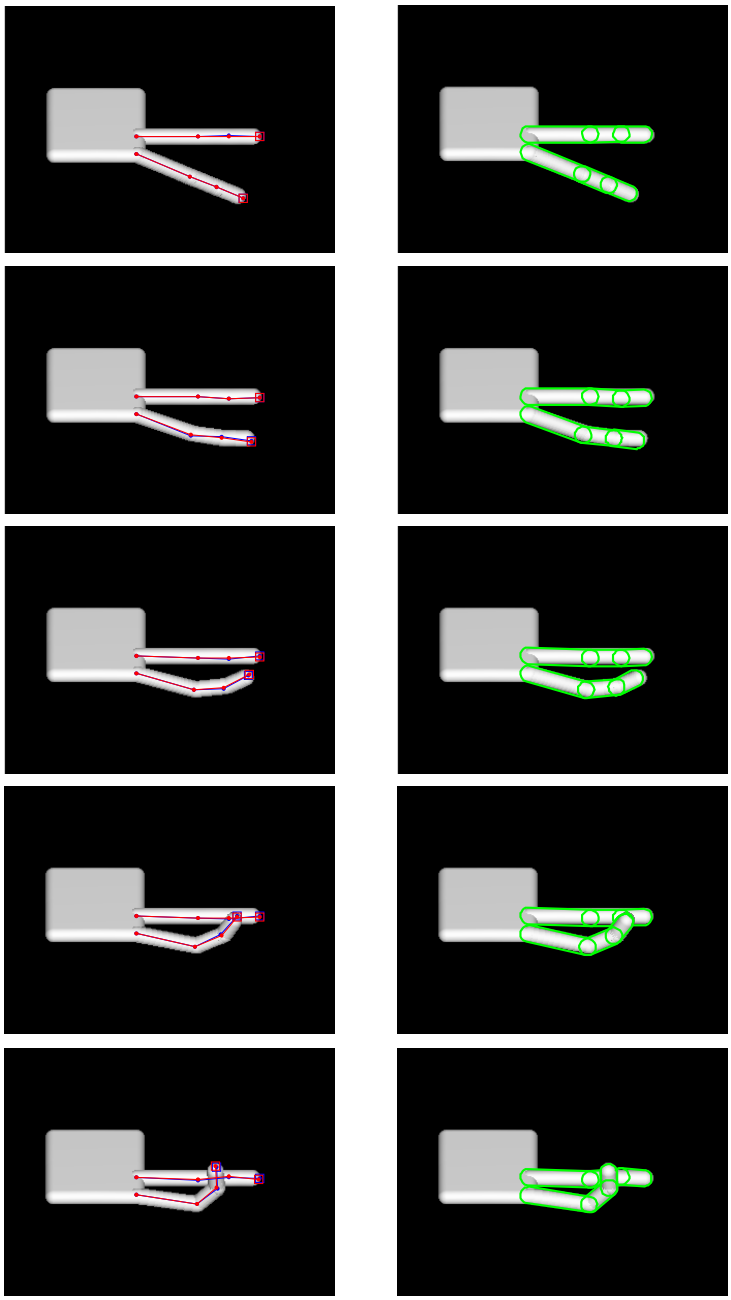


Figure 6.1 - Result of using the proposed method on a sequence of synthetic images (grey model on black background). The pose as estimated by the pose estimation algorithm is represented by the blue lines (left column of images), which generally are covered by the filtered estimates determined by the Kalman filter (red lines). The right column of images features the outlines of the windowing functions, drawn as green polygons. The five lines of images correspond to frames 1, 10, 20, 30 and 40 of an image sequence.

This example shows a hand with index and middle fingers, each with the three

DOF which allow the fingers to bend (flexion/extension) but without the additional DOF which allows the fingers to perform the adduction/abduction movement. This choice of removing that DOF is related with observability issues: from the point of view of the camera, if that DOF exists, a certain image of the fingers can have more than one solution. These multiple solutions are originated by the observability problem itself, but also by uncertainty in the estimation process, which always makes some minor mistakes over time. The hand model in this case is given by Table 6.1.

The movement of the fingers is performed on a plain parallel to the image plain, but that has been tilted 45 degrees towards the camera (around a horizontal axis) to show that tracking is still possible in such a case. The g-h-k filter has $\theta = 0.85$.

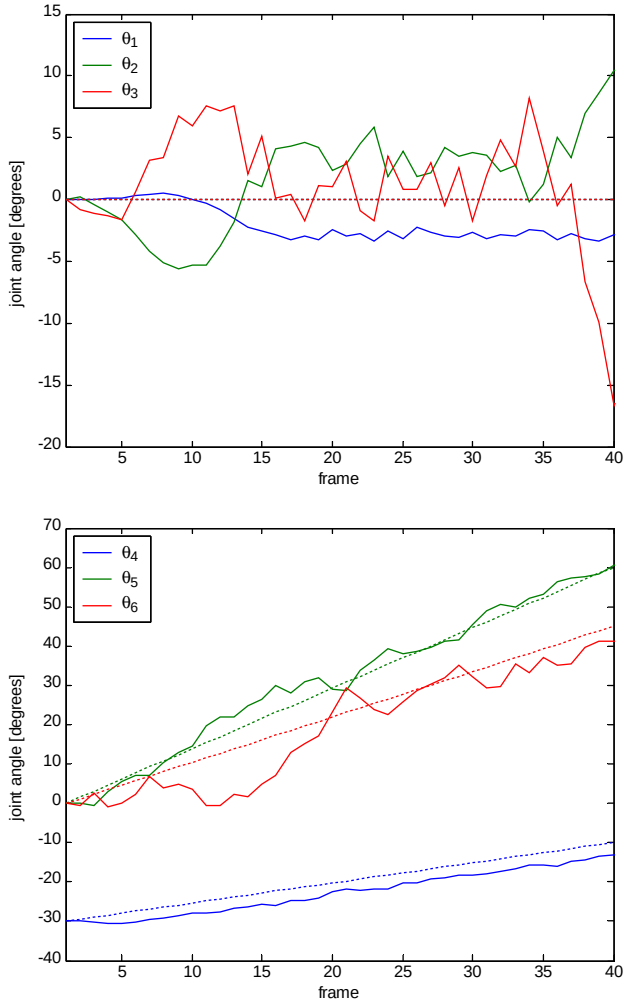


Figure 6.2 - Plots of the filtered pose estimates (continuous lines) versus the true values (dotted lines). The middle finger is represented by the top plot and the index finger by the bottom plot. The higher the subscript number of the angle variable, the closer to the tip of the finger it is.

In the analysed image sequence, the index finger occludes the middle finger. Particular instants of this image sequence can be seen in Figure 6.1, which also shows the pose estimates and filter refinements, as well as the windowing functions. The plots of the filtered estimates versus the ground-truth can be seen in Figure 6.2.

6.2 Alternative Method

These results represent the performance of tracking a human hand using the most complex of the alternative methods (see Appendix D). This alternative method uses:

- Differentiation by convolution with Gaussian derivative;
- Compensated search space restriction;
- Rectilinearity restriction using iterative MWPLLS.

6.2.1 Real Images

This example shows a hand with index finger and thumb, each with three DOF. The index finger has the three DOF which allow it to bend (flexion/extension) but doesn't have the additional DOF which allows it to perform the adduction/abduction movement. The thumb has three DOF instead of five, chosen in a way to only allow it to perform the palmar adduction/abduction movement. The choice of removing some DOF is related with observability issues, as discussed in the first set of results. Another factor that contributed for using this kinematic model was the intent to focus only on the study of grasping movements. The hand model in this case is the same as in the second kinematic model example.

The movement of the fingers is performed on a plain parallel to the image plain. The hand is static during the whole sequence and only the mentioned fingers move. The hand is covered to allow a better tracking of the index and thumb. The g-h-k filter has $\theta = 0.85$.

In the analysed image sequence, the index finger and thumb start by performing spreading and grasping movements and then perform independently of each other. Particular instants of this image sequence can be seen in Figure 6.3, which also shows

the pose estimates and filter refinements. The plots of the filtered estimates can be seen in Figure 6.4.

Figure 6.3 - Result of using the most complex of the alternative methods on a sequence of real images. The hand is covered to allow a better tracking of the index and thumb. The pose as estimated by the pose estimation algorithm is represented by the blue lines, which generally are covered by the filtered estimates determined by the Kalman filter (red lines).

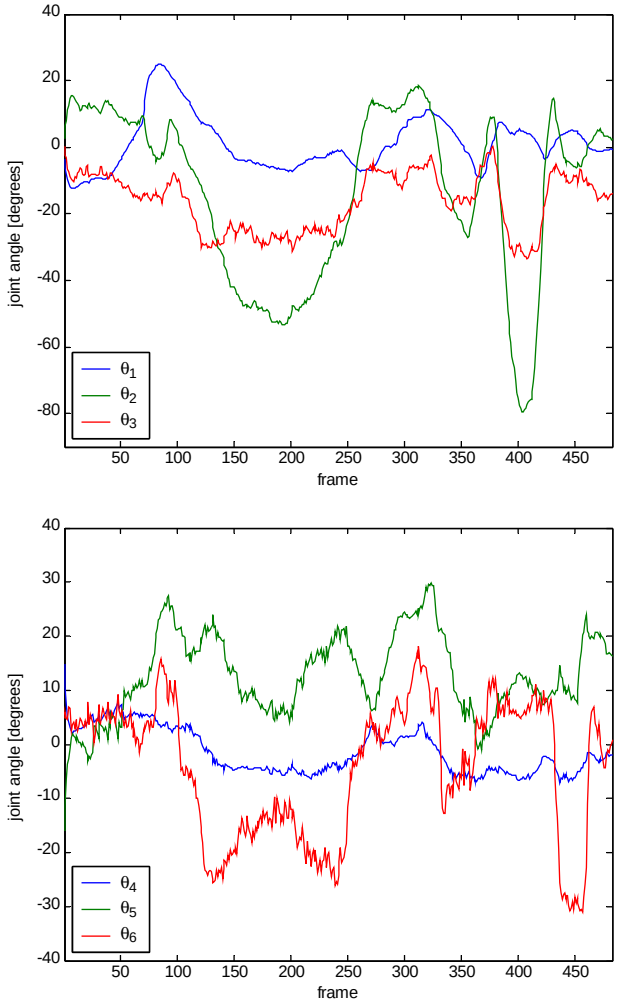


Figure 6.4 - Plots of the filtered pose estimates. The index finger is represented by the top plot and the thumb by the bottom plot. The higher the subscript number of the angle variable, the closer to the tip of the finger it is.

6.2.2 Synthetic Images

This example shows a complete model of a hand. The kinematic model admits a static wrist but considers four DOF per finger and five DOF for the thumb. This choice of DOF allows the fingers and thumb in the model to describe all of the movements described in Figure 2.2. The hand model in this case is given by Table 6.2. The table has been shortened because the parameters for the ring finger are similar to those of the small finger and because the parameters for the index finger are similar to those of the middle finger. The parameters are chosen so that all movements which convey the idea of opening the hand or spreading the fingers are related with angles evolving on a positive direction.

The movement of the fingers is performed on a plain parallel to the image plain, but that has been tilted 45 degrees towards the camera (around a horizontal axis) to show that tracking is still possible in such a case. The g-h-k filter has $\theta = 0.85$.

i	α_{i-1}	a_{i-1}	d_i	θ_i	***tip***	***link***	***point***	***next***	***prev.***	***Notes***		
										frame	***link***	***finger***
0	-	-	-	-	-	-	-	1, 6, 11, 16, 21	-	Base		
1	0	$-\lvert a_0 \rvert$	0	$\pi/2$	-	0	-	2	0	Aux.		
2	0	a_1	0	θ_2	-	0	-	3	1	1st	1st	little
3	$\pi/2$	0	0	θ_3	-	1	x	4	2	2nd		
4	0	a_3	0	θ_4	-	1	x	5	3	3rd	2nd	
5	0	a_4	0	θ_5	A_5	1	x	-	4	4th	3rd	
...												ring
11	π	$+\lvert a_{10} \rvert$	0	$-\pi/2$	-	0	-	12	0	Aux.		
12	0	a_{11}	0	θ_{12}	-	0	-	13	11	1st	1st	middle
13	$-\pi/2$	0	0	θ_{13}	-	1	x	14	12	2nd		
14	0	a_{13}	0	θ_{14}	-	1	x	15	13	3rd	2nd	
15	0	a_{14}	0	θ_{15}	A_{15}	1	x	-	14	4th	3rd	
...												index
21	0	a_{20}	0	0	-	0	-	22	0	Aux.		
22	$\pi/2$	0	0	θ_{22}	-	0	-	23	21	1st	1st	thumb
23	$\pi/2$	0	0	θ_{23}	-	1	x	24	22	2nd		
24	$-\pi/2$	a_{23}	0	θ_{24}	-	0	-	25	23	3rd	2nd	
25	$\pi/2$	0	0	θ_{25}	-	1	x	26	24	4th		
26	0	a_{25}	0	θ_{26}	A_{26}	1	x	-	25	5th	3rd	

Table 6.2 - DH parameters and extra notes for an articulated model of a hand with four fingers and a thumb, with all DOF in a total of 21. “tip” indicates the distance to the tip, “link” if there is a rigid body (1) or not (0) attached to the frame, “point” the joint frame axis with which to align the body, “next” and “prev.” the surrounding frames.

In the analysed image sequence, the thumb remains static. The remaining fingers start by spreading away of each other. Then, while the little finger evolves to a hyper-extended position, the index finger evolves to a flexed position, with the

remaining two fingers evolving to positions gradually between. Particular instants of this image sequence can be seen in Figure 6.5, which also shows the pose estimates and filter refinements. The plots of the filtered estimates versus the ground-truth can be seen in Figure 6.6.

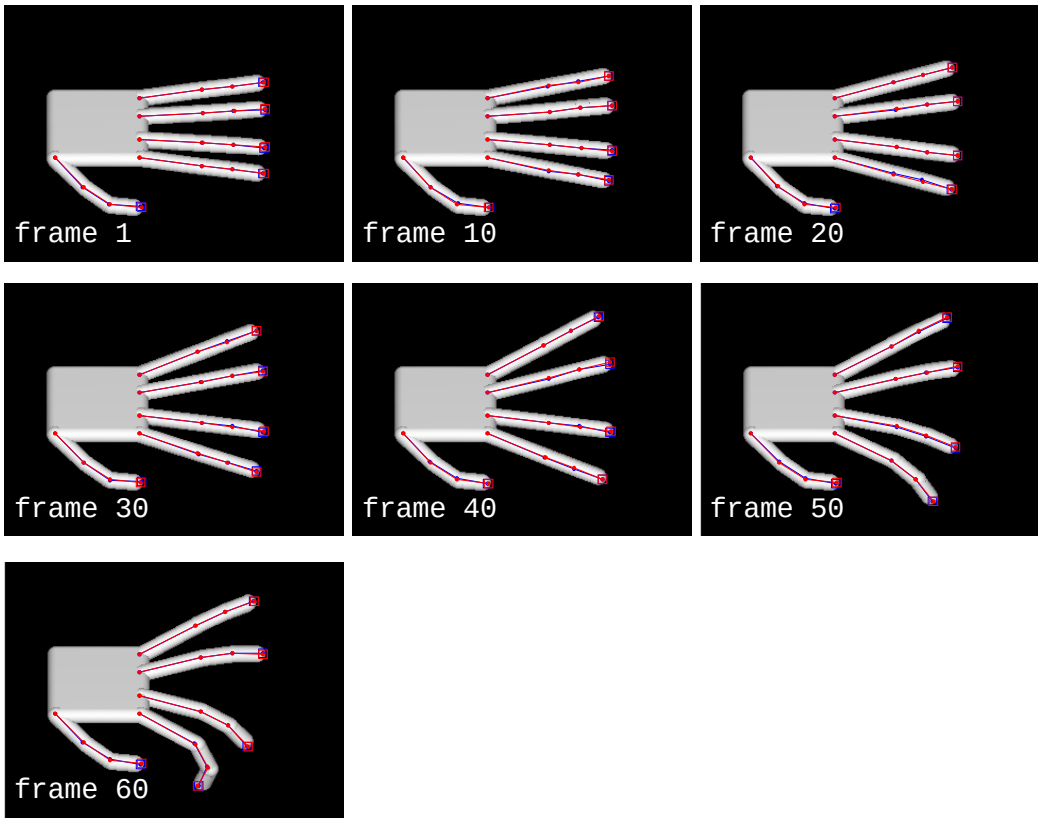


Figure 6.5 - Result of using the most complex of the alternative methods on a sequence of synthetic images (grey model on black background). The pose as estimated by the pose estimation algorithm is represented by the blue lines, which generally are covered by the filtered estimates determined by the Kalman filter (red lines). The model used in tracking has 21 DOF: five for the thumb and four for each finger.

The Gauss-Newton algorithm introduced earlier cannot be used in this case due to the complexity of the cost function and due to the extra DOF which have been neglected in the previous examples. Instead, a more robust Gauss-Newton method is used. This algorithm is a subspace trust region method and is based on the interior-reflective Newton method described in [28] and [29]. The major advantage for the user is a faster and more accurate convergence towards the minimum; other advantage is the user only needs to provide this algorithm with the cost function and the Jacobian. For more details on this algorithm check [30].

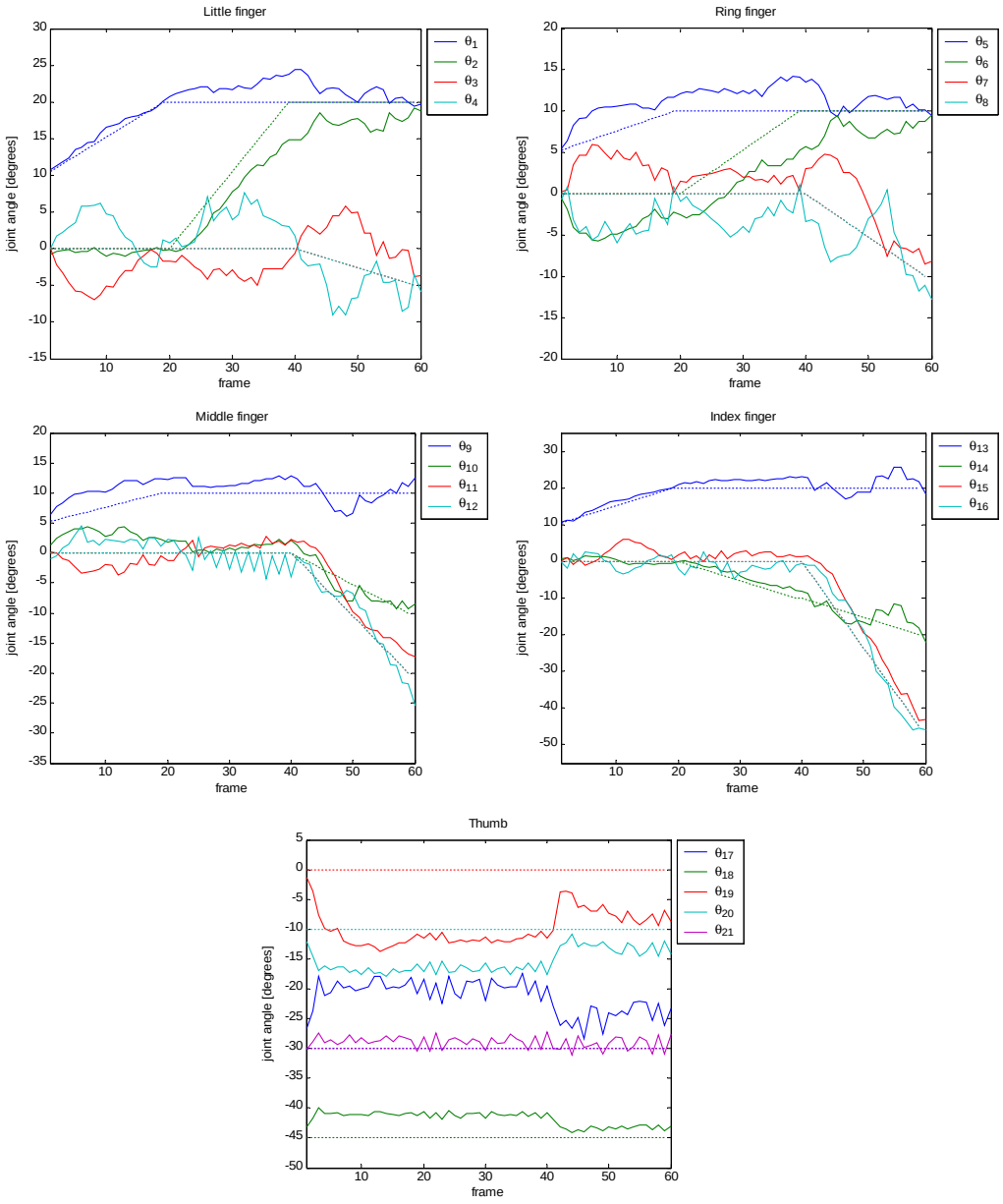


Figure 6.6 - Plots of the filtered pose estimates (continuous lines) versus the true values (dotted lines). On the first four plots, the true value of the fourth DOF overlaps the true value of the third DOF. On the bottom plot, the true value of the fifth DOF overlaps the true value of the first DOF. The higher the subscript number of the angle variable, the closer to the tip of the finger it is.

7 Conclusions and Future Work

The results show that the fingers are successfully tracked, although the estimates present some noise over time. The discrepancies between the estimated angles and the true angles are neglectable when the estimated model is displayed on the image. This happens because the differences between real and estimated angles are compensated along the finger, with the angle in the last articulation compensating the deviations in the previous ones. However, there is still room for improvement.

The results obtained with the proposed method show it deals successfully with occlusion, since both fingers are still tracked when one finger occludes the other. However, care must be taken not to have the occluded and occluding fingers in similar configurations, especially not near total occlusion. This interferes with the ability of correctly matching the profile templates with the correct finger, since in these situations it is harder to solve the matching ambiguity. The result bound to happen when a finger totally or almost totally occludes another, is loss of tracking of the occluded finger.

It is also known, due to the structure of the Jacobian, that there is some redundancy when determining the angles. Generally, the articulations near the base have more redundancy than those near the tips, due to the direct kinematics in the model. This means that generally, the estimates of the joints near the base have a higher confidence than those near the tips. This is perceptible in the plots since the estimates of the joints near the base have less jitter and generally tend to deviate less from the ground-truth.

Since the Jacobian introduces some redundancy in the determination of some joint angles, this could be used to improve the estimation process. Specifically speaking, each element in the vector of residuals could have an associated weight (or contribution) to the total magnitude of the residual. This weight could be determined

by the level of occlusion of a link profile, where a more occluded profile would weight less than a visible profile. Doing so would attribute more confidence to the data for which more information is available. If there is not much data to retrieve from a certain region, then it should not have so much importance in the state estimation.

Besides this weighting factors, it could also prove useful to compromise between the angle estimates provided by the pose estimation algorithm and the angle predictions provided by the Kalman filter. This compromise could be balanced by the confidence a certain angle has in a certain time instant: if to determine the angle a large portion of the finger is visible, then more confidence should be placed in the estimate; on the other hand, if the angle was determined mainly due to redundancy, then more importance should be given to the prediction.

The edge estimation process is very critical, since it provides information which is used by the pose estimation algorithm. As such, it still needs to be improved and have more robustness. Usually, edge estimation may start to loose accuracy when a phalange is more than 50% occluded. The problem is in the lack of robustness of the regression method, used to estimate the valley in the WSSD surface, when there are occlusions.

The results obtained with the alternative method on a complete model of the hand show that observation ambiguity is a problem. This problem tends to aggravate when the joints and links configuration enables several poses to look the same on the image. See for example the thumb estimates: the fact that the first two links have two DOF each is enough to make the thumb estimates converge towards a situation which doesn't correspond at all to the ground-truth. However, from the camera point of view no problem is visible with the thumb estimates. Nevertheless, this may prove fatal in other situations, since the estimation error is carried over time.

To deal with the observability problems it could prove useful to analyse additional points of view of the object. For example, if there were two image sequences of the object, where the additional sequence was filmed from a side view (aligned with the horizontal axis of first camera, for example), that information could be enough to solve the ambiguities caused by a single view. The image analysis would be carried out independently in each image belonging to a different point of view. However, the information provided by each analysis would be combined into a single vector of

residuals. The pose estimation algorithm would then focus on this vector.

The presented tracking methods are not fail proof. There can always be situations where the estimated pose does not correspond to the one in the image, either because tracking was already lost, because an estimate was misdetermined or because there were observability errors carried over time which lead to an incompatible situation. For this reason, it could prove useful to establish a measure of confidence for the estimates, as a way of detecting or predicting situations where tracking is lost. The loss of tracking is a problem because a local evaluation of the image is being made. A possible way of detecting loss of tracking would be widening the scope of evaluation and look at the image in a more general way.

In a general way, tracking works well on synthetic images. However, tracking could be more robust. One possibility is to improve the image analysis algorithms, such as the edge estimation ones, to be more robust in presence of occlusions. Also, it still is necessary to test the proposed method with real images. The proposed method is easier to use with synthetic images because the phalanges are easily modeled by pill primitives. In real images however, this is not the case, and the polygons of the windowing functions are much harder to model.

Temporal integration of the estimates is also an aspect to improve. In the presence of occlusions, a bank of filters could be used to provide a series of predictions, originating a multiple hypothesis tracking problem. Although this is more complex than the single hypothesis method used, it could be useful to add robustness to the overall method and help surpass the occlusion problem.

Some other aspects which could be explored include:

- Automatic initialization of the state vector. This could be done by asking the user to place his/her hand in a certain region of the image with a certain pose. If necessary, as sequence of images could be used to add more robustness to this initialization;
- Estimation and update of the image profile templates over time. This could be used to add robustness to the method when the phalanges are seen from different points of view over time;
- Image segmentation and background extraction. To allow tracking in

the presence of complex backgrounds;

- Optimization of the code and real-time performance. To allow online tracking instead of resorting to pre-recorded image sequences.

References

[1] J. M. Rehg, T. Kanade, "DigitEyes: Vision-Based Human Hand Tracking", Technical Report CMU-CS-93-220, CMU, 1993. http://citeseer.ist.psu.edu/rehg93digiteyes.html

[2] J. M. Rehg, T. Kanade, "Visual Tracking of Self-Occluding Articulated Objects", In Proc. ICCV, Boston, MA., 1995. http://citeseer.ist.psu.edu/rehg95visual.html

[3] T. Cham, J. M. Rehg, "A Multiple Hypothesis Approach to Figure Tracking", In Proc. Perceptual User Interfaces, pages 19-24, November 1998. http://citeseer.ist.psu.edu/cham98multiple.html

[4] J. M. Rehg, "Motion Capture from Movies", In Proc. of Asian Conf. on Computer Vision (ACCV '00), vol. II, pages 1125-1131, Taipei, Taiwan, January 2000. citeseer.ist.psu.edu/478848.html

[5] C. Sminchisescu, B. Triggs, "A Robust Multiple Hypothesis Approach to Monocular Human Motion Tracking", Technical Report RR-4208, INRIA, 2001. http://citeseer.ist.psu.edu/sminchisescu01robust.html

[6] Q. Delamarre, "Modelisation de la Main pour sa Localisation dans une Sequence d'Images", Technical Report 0198, I.N.R.I.A., December 1996. http://citeseer.ist.psu.edu/delamarre96modelisation.html

[7] Q. Delamarre, O. Faugeras, "3D Articulated Models and Multi-view Tracking with Physical Forces", In Computer Vision and Image Understanding, Vol. 81, No. 3, pp. 328-357, March 2001.

[8] V. Athitsos, S. Sclaroff, "3D Hand Pose Estimation by Finding Appearance-Based Matches in a Large Database of Training Views", Technical Report BU-CS-TR-2001-021, Boston University, 2001. http://citeseer.ist.psu.edu/athitsos01hand.html

[9] V. Athitsos, S. Sclaroff, "An Appearance-Based Framework for 3D Hand Shape Classification and Camera Viewpoint Estimation", In Proceedings of the lnt. Conf. on Automatic Face & Gesture Recognition, pages 45-50, Washington, DC, 2002. http://citeseer.ist.psu.edu/athitsos01appearancebased.html

[10] H. Sidenbladh, M. J. Black, "Learning Image Statistics for Bayesian Tracking", In Intl. Conf. on Computer Vision, 2001. http://citeseer.ist.psu.edu/sidenbladh01learning.html

[11] S. Li, W. Hsu, Pung H.K., "Twins: A Practical Vision-Based 3D Mouse", In

Journal of Real-time Imaging, 1998. http://citeseer.ist.psu.edu/450424.html

[12] S. Li, W. Hsu, Pung H. K., "A Real-time Monocular Vision-based 3D Mouse System", In 7th International Conference of Computer Analysis of Images and Patterns (CAIP) , Sept 10-12, 1997, Kiel, Germany, pp. 448-455. http://citeseer.ist.psu.edu/455518.html

[13] H. Ouhaddi, P. Horain, "Conception, ajustement et recalage d'un modele 3D articule pour le suivi de la main dans des sequences d'images", Research Report #980101-SIM, INT, Evry, France, 1998. http://www-eph.int-evry.fr/~horain/Publications/RR98001-ouhaddi.pdf

[14] H. Ouhaddi, P. Horain, "Hand tracking by 3D model registration", In Proceedings of the International Scientific Workshop on Virtual Reality and Prototyping, June 3-4, 1999, Laval, France, pp. 51-59. http://citeseer.ist.psu.edu/ouhaddi99hand.html

[15] N. Shimada, Y. Shirai, "3-D Hand Pose Estimation and Shape Model Refinement from a Monocular Image Sequence", In Proc. of VSMM'96 in GIFU, pp.423-428, 1996. http://citeseer.ist.psu.edu/82627.html

[16] K. Nirei, H. Saito, M. Mochimaru, S. Ozawa, "Human Hand Tracking from Binocular Image Sequences", In 22th Int'l Conf. on Industrial Electronics, Control, and Instrumentation, pages 297-302, Taipei, August 1996. http://citeseer.ist.psu.edu/nirei96human.html

[17] P. Fua, A. Gruen, N. D'Apuzzo, R. Plankers, "Markerless Full Body Shape and Motion Capture from Video Sequences", In International Archives of Photogrammetry and Remote Sensing, Vol. XXXIV, part 5, pp. 256-261. September 2002, Corfu, Greece. http://www.photogrammetry.ethz.ch/general/persons/AG_pub/corfu_isprs5_body.pdf

[18] R. Plankers, P. Fua, "Articulated Soft Objects for Multiview Shape and Motion Capture", In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 9, pp. 1182-1187, September 2003. http://ieeexplore.ieee.org/iel5/34/27551/01227995.pdf

[19] John J. Craig, Introduction to Robotics, 2nd Edition, Addison Wesley, USA, 1989.

[20] C. Eaton, Joint range of motion, in E-Hand.com The Electronic Textbook of Hand Surgery. http://www.e-hand.com/nor/nor002.htm

[21] M. W. Kadous, "GRASP: Recognising Australian Sign Language Using Instrumented Gloves", Thesis, Univ. New South Wales, October 1995. http://waleed.web.cse.unsw.edu.au/thesis.pdf , http://www.cse.unsw.edu.au/~waleed/thesis/node17.html

[22] M.A. Fischler, R.C. Bolles, "RANSAC random sampling concensus: A paradigm for model fitting with applications to image analysis and automated cartography", Communications of ACM, 26, pp. 381-395, 1981.

[23] J. E. Jackson, A User's Guide to Principal Components, John Wiley and Sons,

Inc. 1991, pp. 1-25.

[24] P. S. Maybeck, Stochastic models, estimation, and control - Volume 1, Chapter 1, Academic Press, USA, 1979. Reproduced in "An Introduction to the Kalman Filter", SIGGRAPH 2001, Course 8. http://www.menem.com/~ilya/digital_library/control/welch-bishop-01.pdf

[25] G. Welch, G. Bishop, "An Introduction to the Kalman Filter", SIGGRAPH 2001, Course 8. http://www.menem.com/~ilya/digital_library/control/welch-bishop-01.pdf

[26] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", In Transactions of the ASME - Journal of Basic Engineering, pp. 35-45, March 1960. Reproduced in "An Introduction to the Kalman Filter", SIGGRAPH 2001, Course 8. http://www.menem.com/~ilya/digital_library/control/welch-bishop-01.pdf

[27] E. Brookner, Tracking and Kalman Filtering Made Easy, Wiley-Interscience, 1998.

[28] T.F. Coleman, Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds", in SIAM Journal on Optimization, Vol. 6, pp. 418-445, 1996.

[29] T.F. Coleman, Y. Li, "On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds", in Mathematical Programming, Vol. 67, Number 2, pp. 189-224, 1994.

[30] MATLAB help, lsqnonlin, The MathWorks, Inc. http://www.mathworks.com/access/helpdesk/help/toolbox/optim/lsqnonlin.html

[31] C. Eaton, Hand Facts and Trivia, in E-Hand.com The Electronic Textbook of Hand Surgery. http://www.e-hand.com/hw/facts.htm

[32] National Science and Technology Research Center for Computation and Visualization of Geometric Structures (The Geometry Center), Qhull, University of Minnesota. 1993. . http://www.geom.umn.edu/software/qhull/ , http://www.qhull.org/

[33] MATLAB help, reducem, The MathWorks, Inc. http://www.mathworks.com/access/helpdesk/help/toolbox/map/reducem.html

[34] MATLAB help, Selecting a Renderer, The MathWorks, Inc. http://www.mathworks.com/access/helpdesk/help/techdoc/creating_plots/chprin22.html

[35] MATLAB help, Selecting a Renderer, The MathWorks, Inc. http://www.mathworks.com/access/helpdesk/help/techdoc/creating_plots/figure10.html

8 Appendix A: Human Hand Facts

To understand how complete a human hand is from the conceptional point of view it is worthy to point some relevant facts [31]. A human hand is a highly complex mechanism containing:

- 29 major and minor bones (many people have a few more);
- 29 major joints;
- at least 123 named ligaments;
- 34 muscles which move the fingers and thumb:
  - 17 in the palm of the hand;
  - 18 in the forearm.
- 48 named nerves:
  - 3 major nerves;
  - 24 named sensory branches;
  - 21 named sensory branches.
- 30 named arteries and nearly as many smaller named branches.

Fingers are never perfectly straight. Usually, the index, ring and small finger each curve sideways slightly toward the middle finger, and the middle finger may curve toward either side.

The thumb is controlled by 9 individual muscles, which are controlled by all 3 major hand nerves, and moves in such a complex fashion that there are 6 separate descriptive terms just for particular directions of movement of one thumb joint, the basal joint, at the base of the thumb.

Contrary to popular opinion, humans (*homo sapiens*) are not the only primates possessing opposable thumbs. Chimpanzees and monkeys can oppose the thumb to the

index digit. What makes the human hand unique in the animal kingdom is the ability of the small and ring fingers to rotate across the palm to meet the thumb, owing to a unique flexibility of the carpometacarpal joints of these fingers, down in the middle of the palm. This is referred to as "ulnar opposition" and adds unparalleled grip, grasp, and torque capability to the human hand. This feature developed after the time of Lucy, a direct human ancestor, who lived about 3.2 million years ago.

When one curls the fingers into a fist, the fingertips naturally group together side by side. If the fingertips all bend together, they continue into the palm side by side. However, the natural tendency is for each fingertip to aim for the same point at the base of the thumb, which is obvious when touching each finger down to the base of the palm. For this reason, if a hand problem (stiffness, swelling, etc.) prevents a finger from meeting the side of the adjacent fingertip midway into making a fist, that finger will tend to cross over and overlap the adjacent finger when making a fist.

The finger bones are straight on the back side, but curved on the palm side. When one bends the fingers into a fist, the finger bones produce a shape similar to a circle in a square, round on the inside, square on the outside.

When one makes a fist, the fingertips curve through a spiral, not a circle. This is because the lengths of the finger bones are related in a way seen often in naturally occurring spirals. These spirals in turn relate to a mathematical series of numbers discovered by Fibonacci in 1202. In this series, each number is the sum of the previous two numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21 and so on. The lengths of the finger bones approximate the ratio of the Fibonacci numbers 2, 3, 5, and 8.

The muscles which power the fingers are strong enough for some people to climb vertical surfaces supporting their entire weight at times by a few fingertips. The muscles which accomplish this feat are stronger than one might imagine, for the biomechanics of the hand require that the force generated by the muscles which bend the fingertips must be at least four times the pressure which is produced at the fingertips.

The fingers work by remote control. Of course, in one sense, all moving body parts work by remote control, the control centre being the brain. However, the fingers are special, because there are no muscles inside them. The muscles which bend the finger joints are located in the palm and up in the mid forearm, and are connected to the

finger bones by tendons, which pull on and move the fingers like the strings of a marionette.

9 Appendix B: Simulating Edge Points

The edge points had to be simulated in an early stage because the image processing routines were not yet implemented. There are two types of edge points, one related to the sides of the finger links and the other to the finger tips. The edge points for the links are divided into groups of similar individuals, sharing properties like same link and the same side of that link. These points are generated from a geometric model with the wished configuration. When a silhouette point regarding the tip is needed, the tip of the geometric model with the wished configuration is projected on camera and used as the extracted tip feature.

This simulation follows a very simple procedure in order to generate the silhouette points for the links. For ease of understanding and implementation the procedure is described in Figure 9.1 (steps 1-4). This simulation also serves a second purpose, since it is possible to test different scattering configurations for the points, which can then be used to evaluate the performance of the line fitting algorithm.

The algebra behind this simulation is carried out as follows. A link ranging from point ${}^0\mathbf{p}_1$ to point ${}^0\mathbf{p}_2$ is projected on the image frame using (2.3) to transform each of these points. Hence, a point ${}^0\mathbf{p}_i$ on the base frame of the kinematic chain is projected on point ${}^C\mathbf{p}_i$ of the image frame with

$$ {}^C\mathbf{p}_i = {}^C_0\mathbf{T}\ {}^0\mathbf{p}_i , \tag{9.1} $$

completing step 2 of the procedure.

The edge points are generated perpendicularly to the link in specific locations. The intersections of the link with the perpendiculars are marks given by

$$ \mathbf{M} = \Delta\mathbf{p} \cdot \mathbf{s}^T + \mathbf{P}_1 , \tag{9.2} $$

where $\mathbf{M}$ is a $(2\times n)$ matrix with n vectors of size (2×1) representing the location of the marks in camera units, and is arranged as

$$\mathbf{M}=\begin{bmatrix} M_1^x & \cdots & M_n^x \\ M_1^y & \cdots & M_n^y \end{bmatrix}; \tag{9.3}$$

vector $\Delta\mathbf{p}$ is a (2×1) vector describing the orientation of the projected link and is computed with

$$\Delta\mathbf{p}=\begin{bmatrix} p_2^x-p_1^x \\ p_2^y-p_1^y \end{bmatrix},$$

where the left superscript informs the points are in the image frame {C};

vector $\mathbf{s}^T$ is a $(1\times n)$ vector of normalized scaling factors describing the spread of the marks within the line segment. In the case of an uniform spread of marks

$$\mathbf{s}^T=\frac{1}{n+1}\begin{bmatrix}1 & 2 & \cdots & n\end{bmatrix};$$

finally, $\mathbf{P}_1$ is an offset matrix of size $(2\times n)$ with n repetitions of the coordinates of ${}^C\mathbf{p}_1$, hence being arranged as

$$\mathbf{P}_1=\begin{bmatrix} p_1^x & \cdots & p_1^x \\ p_1^y & \cdots & p_1^y \end{bmatrix}.$$

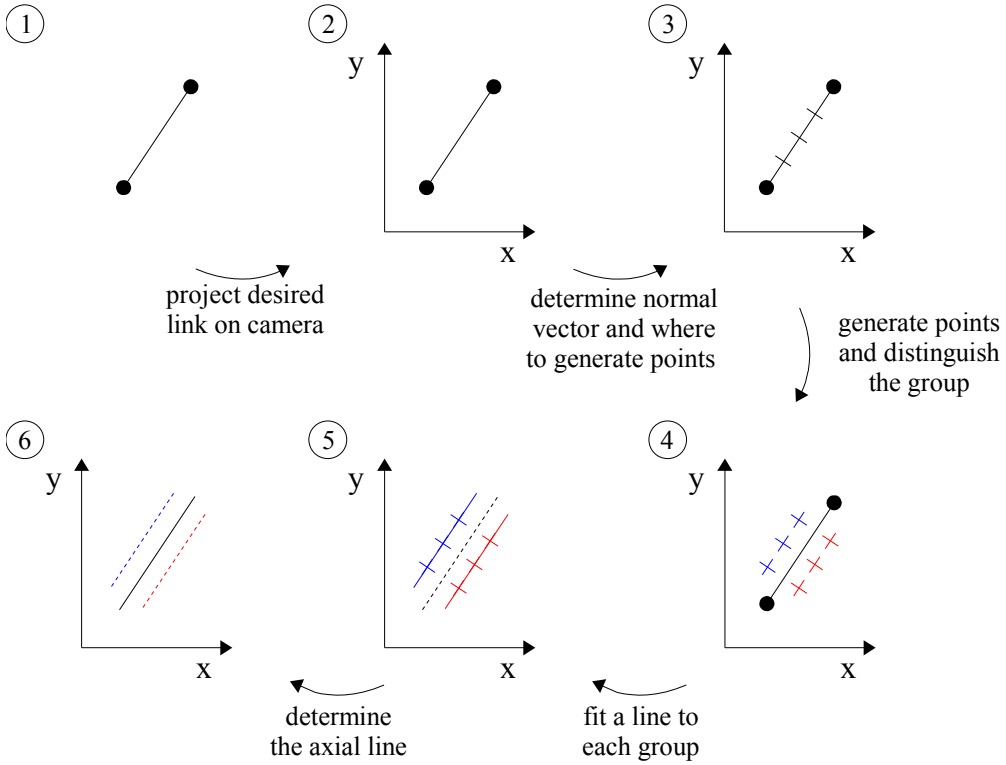


Figure 9.1 - A six-step procedure describing how to generate silhouette points for the links in the simulation of extracted data (1-4), how to determine the boundary lines (5) and the axial line (6).

The normal vector describing the direction of the perpendiculars has unitary

magnitude and is computed with

$$\mathbf{v}=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\frac{\Delta\mathbf{p}}{\|\Delta\mathbf{p}\|}, \tag{9.4}$$

thus concluding step 3 of the procedure.

Points are generated to each side of the line segment with

$$\mathbf{S}_{\pm}=\mathbf{M}\pm\mathbf{B}, \tag{9.5}$$

where the sum generates points on one side and the subtraction points on the other, thus distinguishing the groups of points; the $(2\times n)$ matrix $\mathbf{B}$ contains the offsets from the marks to the simulated boundary points and is composed of n vectors of size (2×1), each computed with $\sigma\mathbf{v}$, where σ is a scalar representing the distance to the segment. This parameter is constant because all boundary points of a group are generated at the same distance of the segment. If each point had its deviation there would be a σ for each of them. Both $\mathbf{S}_{\pm}$ and $\mathbf{B}$ are arranged similarly to $\mathbf{M}$ in (9.3). In the end there will be as many pairs of $\mathbf{S}$ matrices as the number of finger links.

The generation of tip points was already described in the opening paragraph and is easily derived from (9.1) leading to as many tip points as the amount of fingers.

10 Appendix C: Creation of Synthetic Images

Synthetic images are used to study the performance of the feature extraction and state estimation routines under a controlled scenario. In their simplest formulation the synthetic images are binary images presenting a black background and a white silhouette of the 3D model of the articulated object (the foreground). In a more elaborate conception the 3D model is drawn with a range of shades of grey, conferring it a more realistic look. The 3D model in the synthetic image should have some morphologic resemblance with the articulated object obtained from a real case.

The forthcoming description grows around the idea of creating synthetic images of one finger. The concept is easily expanded to more. A finger can be modelled by a chain of cylinders toped with hemispheres, forming a shape resembling a pill.

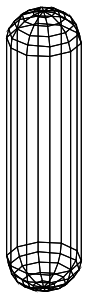


Figure 10.1 - Wireframe representation of a 3D primitive resembling a pill.

The model which results from chaining these pills has a smoother appearance than a model resulting from chaining cylinders.

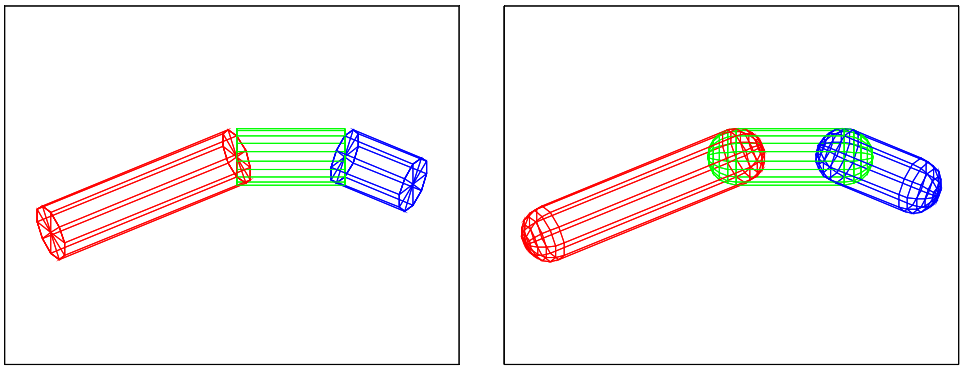


Figure 10.2 - A model built with pill (right) has much smoother transitions than one built with cylinders (left).

Building a pill is rather simple since it is only necessary to parameterize the cylinder and hemispheres and connect them correctly. Each of these geometric primitives is achieved with a patch graphic object, which is composed of one or more polygons that may or may not be connected, originating the desired 3D configuration. These primitives can then be rotated and displaced with the same spatial transformations derived in the kinematic definition applied to the vertices of the polygons.

The synthetic images are created projecting each of the polygons on the image frame, which is easy since both polygons and camera projection matrix are known. In practice patch objects are composed by an index of vertices and an index of faces. The index of vertices has information about the 3D coordinates of all points in the patch, while the index of faces defines which vertices compose each polygon. Again, when projecting the polygons on the image only the index of vertices is affected by the transformations, the index of faces remains unchanged.

Situations as addition of image noise and presence of an elaborate background are not exclusive of neither type of image and can be used in any of them to further determine the potential of the mentioned routines.

10.1 Binary Images

Binary images are the most basic way of representing a 3D object on camera. Apart from each pixel assuming only one of two discrete values, zero for black and one for white, binary images allow some simplifications and considerations to be made when

projecting an object on the image frame. They also provide strong contrast between background and foreground.

Each of the three phalanges of the model is represented by a pill, which is a convex object in 3D space. A convex object in 3D space when projected in a 2D space originates a convex surface. If all vertices are projected on camera there will be a cloud of points, some outlining the boundary of the projection and some filling it. To avoid the time consuming task of painting each projected polygon on the image, and the redundancy of painting over and over again the pixels which happen to match more than one polygon, a simplification must be made.

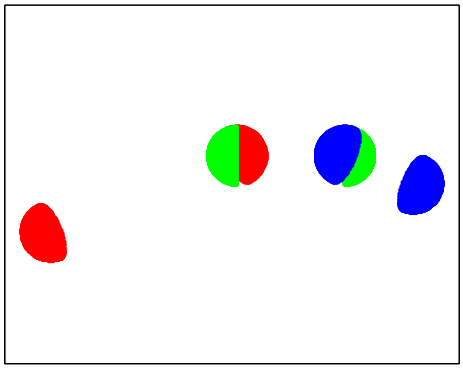


Figure 10.3 - Cloud of points of a finger with three phalanges modelled by pill primitives.

Even though it has an excess of points the projected surface is convex, hence the smallest polygon which can contain the region is still convex, and is called a convex hull. So if the convex hull of the cloud of points is computed, the resulting polygon is formed by all peripheral points of that shape, and still preserves the same silhouette. However, the number of vertices to use is significantly reduced and there is no more need to use the index of faces since there is only one polygon. The convex hull routine used is based on [32].

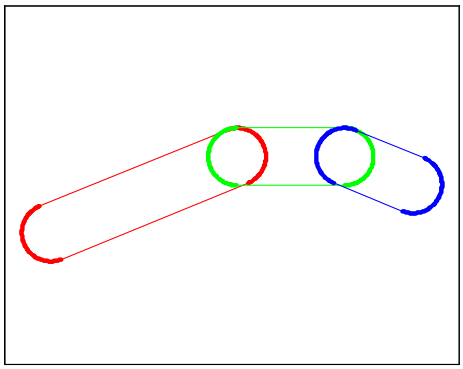


Figure 10.4 - Convex hulls of each of the projected cloud of points.

By now there is a huge benefit in using the convex hull points to paint the polygon on the image. However, it is likely there are still some redundant points in the polygon, so it could prove a good thing getting rid of them too. This is achieved by a method used in land mapping known as Douglas-Peucker line simplification algorithm. This method recursively subdivides a polygon until a run of points can be replaced by a straight line segment, with no point in that run deviating from the straight line by more than the tolerance. The tolerance is computed automatically by the algorithm with a value corresponding to $A/(250\times N)$, where A is the sum of arclengths of the polygon and N the number of points used to describe the polygon. For more details check [33].

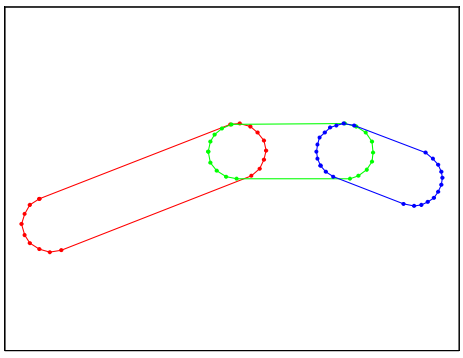


Figure 10.5 - Reduced versions of the convex hulls.

This tolerance is usually small enough to make the resulting reduced polygon produce the same image counterpart as the longer polygon would. However, the image will not be painted just yet since the reduced polygon also allows faster polygon Boolean operations, because there are less points involved. If all three polygons of a

finger model are projected and have their convex hull computed and reduced, the three reduced polygons can be combined in a single polygon, generally not convex, leading to painting the wished pixels only once, instead of painting them twice where the pills projections overlap. This may not seem a great improvement, but when there is more than a finger and an image sequence to create, all time reductions count.

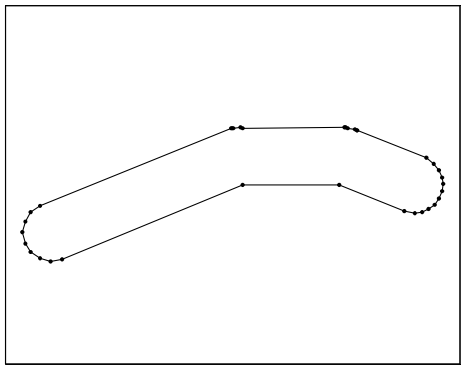


Figure 10.6 - Polygon resulting from merging three reduced convex hulls representing the finger phalanges.

The function used for registering the polygon to the image is designed to select a polygonal region of interest within an image and return a binary image that can be used as a mask for masked filtering. Since the goal is to create a binary image presenting zeros outside the region of interest and ones inside, it suffices to supply this function with a dummy image (the same size as the intended image) and the coordinates of the polygon points. The resulting mask is the painted binary image.

Figure 10.7 - Image resulting from painting a white merged polygon on a black background. The y-axis of the image frame is oriented downwards instead of upwards as with the plots, leading to a vertically flipped version of those plots.

10.2 Shaded Images

Binary images are useful for the high contrast they provide between foreground (the object) and background, and can be used to represent one or more fingers as to test the aforementioned performances. In spite of this, their binary nature represents a limitation and a problem whenever the different fingers overlap each other or themselves. This phenomenon is called occlusion, and is divided into mutual and self occlusion, respectively. When occlusion occurs in binary images the boundaries of the fingers in the occluded regions are hard to identify.

Figure 10.8 - In binary images (left) it is hard to locate the edges of the overlapping fingers whereas in shaded images (right) this is much easier and the occluding finger is easily identified.

Other limitation which arises with binary images is the fact sometimes they do not embody what would happen in a real case, where contrast is generally much lower and there is a range of intensity values for the foreground rather than just one.

These two situations, along with the fact it is hard to determine what is going wrong when the analysis of real images has problems, trigger the need of having more complex images than binary ones, as a means of emulating a real case in a controlled environment. Thus, the concept of image and 3D model simulator is brought forth with the creation of shaded images.

Creation of images where the model is represented with more fidelity, namely shape relief and illumination, are not so easy to obtain. The representation of 3D objects on a 2D image with preservation of relief and illumination are delegated on an entity called renderer. A renderer [19][34][35] is software and/or hardware that processes graphics data (such as vertex coordinates) to display, print, or export a

figure. Several modes of displaying (or rendering) faceted surface models exist, differing in complexity, accuracy, represented features, and overall look. To understand the choice made it is important to know the most commonly available techniques and their pros and cons.

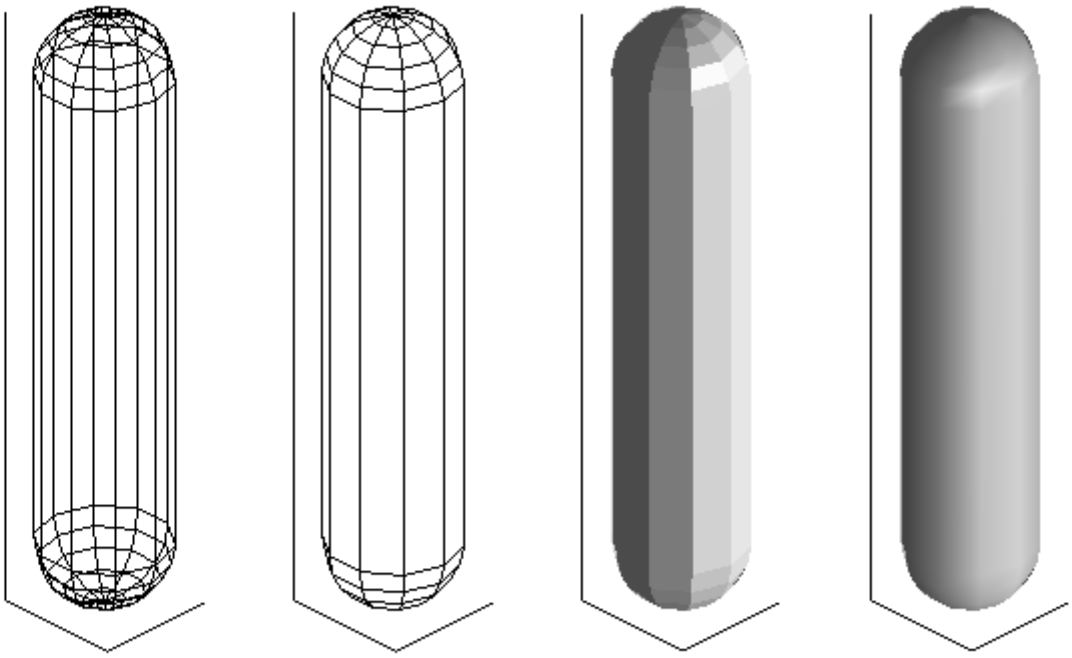


Figure 10.9 - Different rendering and shading techniques (from left to right): wireframe, back face elimination, Z-buffer with flat shading, Z-buffer with Gouraud shading.

The simplest rendering technique is wireframe, in which each edge of the object is drawn using a perspective transformation, such as the camera projection matrix. This representation can be improved using a technique called back face elimination, which removes the hidden lines of individual convex shapes, and yields an image which is somewhat more easily understood by the viewer. Back face elimination is much simpler than the complete hidden line elimination problem. One method of achieving hidden line elimination is to fill the facets to create a shaded image. By drawing facets farthest from the viewer first, near facets tend to cover far facets, producing a hidden line eliminated image. Perfect rendering of shaded images requires a Z-buffer technique implemented either in software or hardware to do depth sorting on a pixel by pixel basis. This method can be further refined using a technique such as Gouraud

shading to smooth over the edges that exist between facets to make the rendered surface appear smooth.

To control the shading intensity of the facets it is necessary to resort to a technique called lighting (or shading) to add realism to the graphical scene. It does this by simulating the highlights and dark areas that occur on objects under natural lighting (e.g. the directional light that comes from the sun). It basically computes how the rays of light from a light source are reflected by the object surface towards the camera.

In short, to represent the object in a realistic manner it is necessary to use rendering techniques to display it and include lighting techniques to control the shading intensity of the facets. This narrows down the choice to back face elimination renderers such as Z-buffer or OpenGL and shading methods such as flat, Gouraud or Phong.

Regarding the rendering methods, Z-buffering is the process of determining how to render each pixel by drawing only the front-most object, as opposed to drawing all objects back to front, redrawing objects that obscure those behind. The pixel data is buffered and then blitted to the screen all at once. OpenGL is available on many computer systems. It is generally faster than Z-buffer and in some cases enables the use of the system's graphics hardware (which results in significant speed increase).

As for the shading methods, flat lighting produces uniform colour across each of the faces of the object and should be used to view faceted objects. Gouraud lighting calculates the colours at the vertices and then interpolates colours across the faces and should be used to view curved surfaces. Phong lighting interpolates the vertex normals across each face and calculates the reflectance at each pixel. It should also be used to view curved surfaces since Phong lighting generally produces better results than Gouraud lighting, although it takes longer to render.

The ideal thing would be to have access to the rendering and shading routines and directly create the desired image. Unfortunately, in this case, these routines are not directly accessible by the user and do not provide that output, so it is necessary to find a way around the problem. Prior to presenting the solution it is important to understand how graphic objects are displayed in this case.

Figure objects are the individual windows on the screen where graphical output is

supplied. These figure objects have a hierarchy of other objects which can be docked to them, namely the axes object, which is used to dock and display graphic objects such as images, lights, lines, patches, rectangles, surfaces and text. Displaying patches on top of an image does not affect the image data, but the fact it is possible to do this display is the first step to solve the problem.

The solution takes shape as follows: the background image is displayed on the image object attached to the axes object, then vertices of the model are transformed with the camera projection matrix and the transformed patch objects are displayed and docked to the same axes object. This information is visible on screen as the desired image, but it still is a group of layered objects: background image behind and patch graphics on top. The only step left is to create a new image with this information merged together. This is done with a supplied screen capture application, which allows the screen region of interest to be converted to a discrete image. This is actually suggested as the only way of converting data on a figure object to a discrete image. Although not as refined a solution as working directly with the rendering and shading routines, it still is considered a valid alternative since these images are created offline (before being used) and not online (at the same time they're needed).

Patch objects are automatically displayed, the user only has to supply the index of vertices, the index of faces and optionally choose the rendering and shading methods. To confer more realism to the scene it is possible to set the reflectance properties of the surface material and place the light sources, the renderer and shader take care of displaying it properly.

As for the projection matrix, it cannot be a (3×4) matrix, it has to be (4×4) to preserve the depth coordinate so the renderer can perform the depth sort. Remember this is only a problem because in the present case it is not possible to render a 3D object in a 3D space directly to an image. This matrix is achieved expanding the first one with an extra row. If the (3×4) matrix is given by

$$\mathbf{P}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

then the (4×4) matrix is given by

$$\tilde{\mathbf{P}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ \tilde{r}_{31} & \tilde{r}_{32} & \tilde{r}_{33} & \tilde{p}_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For example, if the y-axis of the image frame and frame {0} are symmetric, with coincident x and z axes, since frame {0} respects $\hat{X} \times \hat{Y} = \hat{Z}$, the image frame would have the anti-symmetric definition $\hat{X} \times \hat{Y} = -\hat{Z}$, and thus

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Even in a simple example like this, it is important to mind the scaling factors. Those along the x and y axes can be set by camera calibration, while the one along the z-axis can be extrapolated with the help of the other two avoiding as much as possible to skew the resulting projected model along the image z-axis.

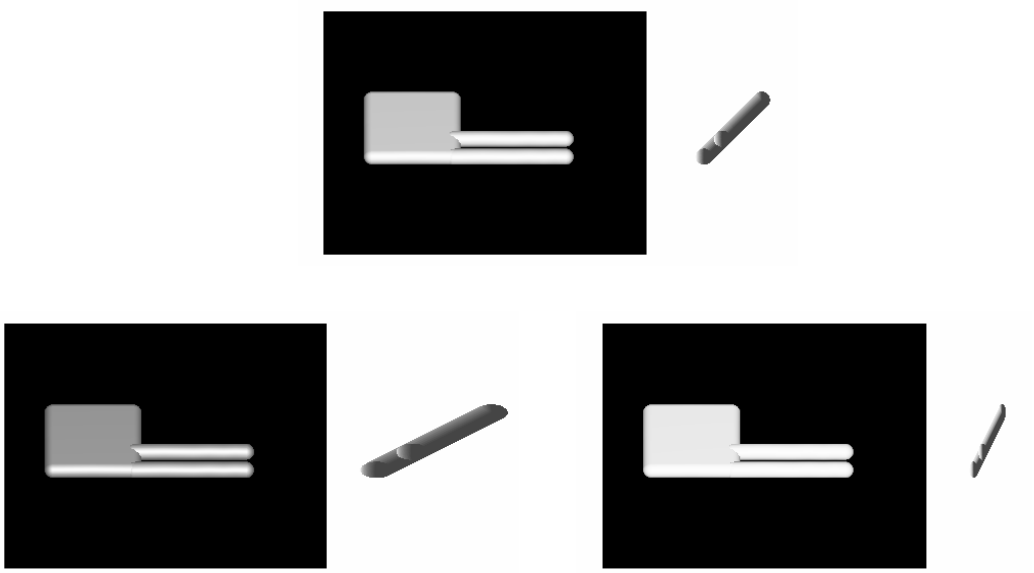


Figure 10.10 - Image views and respective side views of a 3D object transformed to the camera frame. When the depth scaling coefficient is correctly chosen (top) the fingers section is circular. On the other hand, if the depth coefficient is greater (left) or smaller (right) than it should, the object appears skewed along the camera z-axis. Notice how the brightness of the object in the image view is affected when the scaling factor is not appropriate.

The scaling factors are important for computing light reflection on the faces of the projected model. The shading process confers different intensities to the faces if the object is too thin or too large along the z-axis.

In short, a shaded image is currently produced by capturing a region of interest on screen which displays the background image and a rendered patch graphic with transformed coordinates.

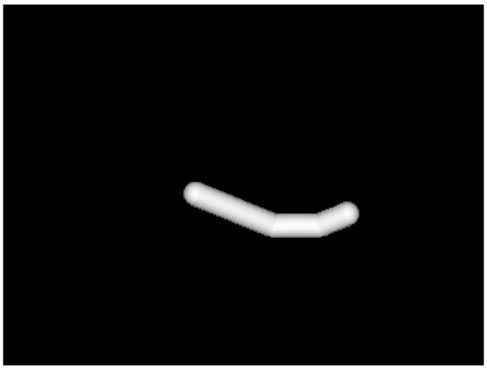


Figure 10.11 - Image resulting from rendering a model of a finger over a black background.

This kind of images is also useful for animation purposes, for example, to overlay a rendered 3D model to an analysed sequence of real images, in order to stress the coincidence of estimated pose and real pose.

11 Appendix D: Feature Search Alternatives

The following sections are a set of image processing alternatives to the one presented in section 4.3. One thing in common to all methods is they all use image profiles sampled perpendicularly to a predicted model pose. However, the following methods explore image profile derivatives rather than profiles themselves.

The first method is the most basic and follows the principles of [1], while the second one improves it resorting to knowledge of width in the form of search restriction and template matching masks. The third method improves the second one exploring properties in the template matching measures of several profiles of the same link. Hence, each method includes the previous one.

11.1 Differentiation of Image Profiles

The extraction of features presented at this stage assumes it is possible to correctly retrieve the necessary data. Hence, it is assumed this is a situation where there is no self or mutual occlusion of fingers, the illumination is done in a way not to cast shadows on the fingers and the background is homogeneous and provides strong contrast with the object.

This method is motivated by the idea that differentiation of an image profile potentially leads to peaks near the edges of the finger phalanges. This means if the derivative of an image profile is computed, then it is expected to present a positive and a negative peak where the edges are located. Identification of these peaks leads to finding the link edges location for that profile.

The are two variants to the process of differentiation which are presented below.

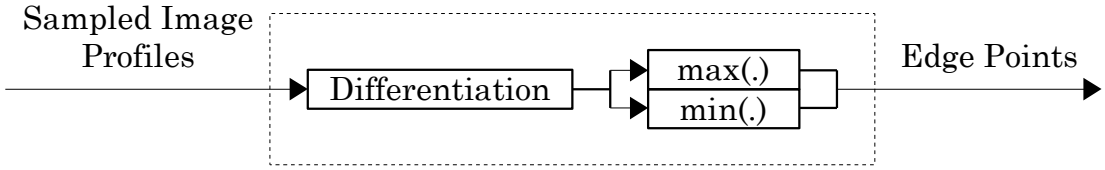


Figure 11.1 - The first alternative method for feature search finds edge points by locating intensity peaks in the derivatives of the image profiles.

11.1.1Simple Discrete Differentiation

This way of computing the derivative is based on a particularly simple concept: after retrieving a discrete image slice $\mathbf{I}(k)$ of size m, the derivative $\mathbf{D}_\mathbf{I}(k)$ of the profile is computed using

$$\mathbf{D}_\mathbf{I}(k)=\mathbf{I}(k+1)-\mathbf{I}(k), \tag{11.1}$$

where $k\in\{1, 2,\ldots, m-1\}$ is an element in the array. A derivative computed this way has size $m-1$ and is not robust to image noise. It suffices, however, for the purpose of demonstration.

In the case of a link slice the peaks of the derivative are given by $\max(\mathbf{D}_\mathbf{I})$ and $\min(\mathbf{D}_\mathbf{I})$, which correspond respectively to positions

$$k_{\max}=\underset{k}{arg\,max}\ \mathbf{D}_\mathbf{I}(k) \text{ and } k_{\min}=\underset{k}{arg\,min}\ \mathbf{D}_\mathbf{I}(k)$$

in the image derivative. Due to (11.1) the corresponding positions in the image slice are dubious because position k must be an integer. The best would be to relate $k_{\max}$ and $k_{\min}$ in the derivative with

$$\hat{k}_{\max}=\left\lfloor k_{\max}+(k_{\max}+1)\right\rfloor/2 \text{ and } \hat{k}_{\min}=\left\lfloor k_{\min}+(k_{\min}+1)\right\rfloor/2$$

in the slice, however this would lead to a position in image coordinates which had not been sampled. Hence, the positions in the image slice are respectively set to $\hat{k}_{\max}=k_{\max}$ and $\hat{k}_{\min}=k_{\min}$ and this shall be assumed as the default procedure.

The coordinates of the corresponding points in the image frame are stored in a slot of the respective group, materialized by a column of matrix $\mathbf{S}$ which has size $(2\times n)$ as in (9.5).

In the case of a tip slice, the peak of the derivative is found with $\max(|\mathbf{D}_\mathbf{I}|)$, where $k_{\max}=\underset{k}{arg\,max}\ |\mathbf{D}_\mathbf{I}(k)|$, and the coordinates of the corresponding point in the image frame are stored to form the tip feature, which is stored in a vector containing the x and y image coordinates.

11.1.2Differentiation by Convolution with Gaussian Derivative

This way of computing the derivative resorts to a concept which is widely known in image processing and has the purpose of yielding derivatives without amplifying image noise.

The derivative of the image profile obtained with the differentiation method described in equation (11.1) amplifies image noise which is present in images obtained with a camera. In a situation where an image profile $\mathbf{I}$ is affected by noise $\mathbf{n}$, the measured value is a noisy version of $\mathbf{I}$, given by

$$\mathbf{I}_{\mathbf{n}}(k)=\mathbf{I}(k)+\mathbf{n}(k), \tag{11.2}$$

where k represents an element in the array.

If $\mathbf{n}(k)$ is a random variable with normal distribution, then it is described by a mean and a variance, which are respectively

$$\mathrm{E}[\mathbf{n}(k)]=\mu_n$$
$$\mathrm{VAR}[\mathbf{n}(k)]=\mathrm{E}[\mathbf{n}(k)^2]-\mu_n^2=\sigma_n^2.$$

As for the derivative $\mathbf{D}_{\mathbf{n}}$ of the noisy image profile, its expression is obtained from (11.1), resulting in

$$\begin{aligned}\mathbf{D}_{\mathbf{n}}(k)&=\mathbf{I}_{\mathbf{n}}(k+1)-\mathbf{I}_{\mathbf{n}}(k)=\big(\mathbf{I}(k+1)+\mathbf{n}(k+1)\big)-\big(\mathbf{I}(k)+\mathbf{n}(k)\big)\\&=\mathbf{D}_{\mathbf{I}}(k)+\big(\mathbf{n}(k+1)-\mathbf{n}(k)\big)\end{aligned}.$$

The noise level $\mathbf{n}_{\mathbf{D}}(k)$ of $\mathbf{D}_{\mathbf{n}}(k)$ is characterized by two adjacent image noise terms, yielding

$$\mathbf{n}_{\mathbf{D}}(k)=\mathbf{n}(k+1)-\mathbf{n}(k).$$

This new random variable is characterized by a normal distribution with parameters

$$\mathrm{E}[\mathbf{n}_{\mathbf{D}}(k)]=\mu_{n_D}$$
$$\mathrm{VAR}[\mathbf{n}_{\mathbf{D}}(k)]=\sigma_{n_D}^2.$$

For convenience of manipulation, the random variables $\mathbf{n}_{\mathbf{d}}(k)$, $\mathbf{n}(k+1)$ and $-\mathbf{n}(k)$ are respectively replaced by Z, X and Y, leading to $Z=X+Y$. This more general representation of variables is useful for algebraic manipulation. Since X and Y are not necessarily independent:

$$
\begin{aligned}
&\mathrm{E}[Z]=\mathrm{E}[X+Y]=\mu_x+\mu_y=\mu_z \\
&\mathrm{VAR}[Z]=\mathrm{E}\left[Z^2\right]-\mathrm{E}[Z]^2=\mathrm{E}\left[X^2+2XY+Y^2\right]-\left(\mu_x^2+2\,\mu_x\mu_y+\mu_y^2\right) \\
&\qquad=\left(\mathrm{E}\left[X^2-\mu_x^2\right]\right)+\left(\mathrm{E}\left[Y^2-\mu_y^2\right]\right)+2\left(\mathrm{E}\left[XY-\mu_x\mu_y\right]\right) \\
&\qquad=\sigma_x^2+\sigma_y^2+2\left(\mathrm{E}\left[XY-\mu_x\mu_y\right]\right)
\end{aligned}
.
$$

If X and Y are independent then $\mathrm{E}[XY]=\mathrm{E}[X]\mathrm{E}[Y]=\mu_x\mu_y$, so the last term of the variance disappears. Returning to the old set of random variables, this means noise in the derivative is characterized by

$$
\begin{aligned}
&\mathrm{E}\left[\mathbf{n_D}(k)\right]=\mu_n-\mu_n=0 \\
&\mathrm{VAR}\left[\mathbf{n_D}(k)\right]=\sigma_n^2+\sigma_n^2=2\,\sigma_n^2\,,
\end{aligned}
$$

assuming noise in a pixel is independent from noise in other.

So it is clear the variance of the derivative of the noisy image profile is greater than the variance of the noisy image profile, hence it is demonstrated that the derivative amplifies noise (the bigger the variance, the bigger the noise amplitude).

Computing the derivative with (11.1) is equivalent to perform the convolution of the image profile by a step mask, as results from

$$
\mathbf{D_I}(k)=\mathbf{I}(k)*\mathbf{U}(k)\,,
$$

where $\mathbf{U}=\begin{bmatrix}1 & -1\end{bmatrix}^T$ is the step mask and the convolution of two discrete 1D signals $\mathbf{s}_1$ and $\mathbf{s}_2$ is defined by

$$
\mathbf{s}_1(k)*\mathbf{s}_2(k)=\mathbf{w}(k)=\sum_j \mathbf{s}_1(j)\mathbf{s}_2(k+1-j)\,.
$$

Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of $\mathbf{s}_1$ and $\mathbf{s}_2$. Conceptually it can be thought of as if the flipped version of $\mathbf{s}_2$ slided across $\mathbf{s}_1$ and their windowed product was computed each time. The sum is over all values of j which lead to legal subscripts for $\mathbf{s}_1(j)$ and $\mathbf{s}_2(k+1-j)$. If m is the length of $\mathbf{s}_1$ and n the length of $\mathbf{s}_2$, then the sum is carried out for $j\in\{\max(1,k+1-n),\cdots,\min(k,m)\}$.

It is now clear how the simple process of differentiation increases the noise level and how it can be represented by a convolution. This parallelism with convolution helps understanding the upcoming solution to the noise increase problem.

Thinking of the mask as a sliding signal it is quite immediate to think it would help to have a mask which accounted for a set of adjacent pixels instead of just two of them. This way each derivative point is computed as a smooth differentiation, or in

other words, a weighted differentiation of a set of adjacent pixels. Usually the derivative using this variant can be expressed by

$$\hat{\mathbf{D}}_{\mathbf{n}}(k)=c_1\left(\mathbf{I}_{\mathbf{n}}(k+1)-\mathbf{I}_{\mathbf{n}}(k-1)\right)+c_2\left(\mathbf{I}_{\mathbf{n}}(k+2)-\mathbf{I}_{\mathbf{n}}(k-2)\right)+\cdots,$$

which means the noise component is given by

$$\hat{\mathbf{n}}_{\mathbf{D}}(k)=c_1\left(\mathbf{n}(k+1)-\mathbf{n}(k-1)\right)+c_2\left(\mathbf{n}(k+2)-\mathbf{n}(k-2)\right)+\cdots,$$

hence its variance is

$$\mathrm{VAR}\left[\hat{\mathbf{n}}_{\mathbf{D}}(k)\right]=c_1^2 2\sigma_n^2+c_2^2 2\sigma_n^2+\cdots=\left(c_1^2+c_2^2+\cdots\right)2\sigma_n^2.$$

The c_i are multiplicative factors ruled by

$$\sum_i c_i=1$$
$$0<c_i<1, c_i\subset\mathbb{R}, \forall i.$$

The convolution needs to be computed strictly within the scope of $\mathbf{s}_1$, so when a part of the sliding mask $\mathbf{s}_2$ falls outside these boundaries, the missing values of $\mathbf{s}_1$ are replaced with the value $\mathbf{s}_1$ has at the end closest to that boundary.

In image processing it is known such a mask can be the derivative of a Gaussian function, where the c_i are small near and far from the centre of the convolution mask and assume greater values in between. These coefficients are obtained with

$$c_i=\frac{d}{dk}\mathrm{N}(k=i|0,\sigma),$$

where $\mathrm{N}(k|0,\sigma)$ is the normal function given by

$$\mathrm{N}(k|0,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{k^2}{2\sigma^2}}.$$

The result of convolving a signal with the derivative of a Gaussian is a smooth derivative of the signal: the slower evolving character of the derivative is preserved while the faster is attenuated.

11.2 Width Dependent Search Restriction and Template Matching

The process of extracting boundary points from the image is adequate but rather weak in terms of robustness to image noise, misplacement of trackers, low amplitude of the peaks in the derivative of the image profiles and low curvature in the vicinity of those peaks. This lack of robustness leads to errors determining the edge points, which then

leads to loss of tracking, generally revealed in the form of an estimated pose which does not correspond either partially or globally to the analysed image.

From the previously introduced differentiation methods the one which is more robust to image noise is differentiation by convolution with a Gaussian derivative, thus from this point on this will be the method used when referring to differentiation and derivatives of image profiles.

One way to improve the previously presented edge extraction method is to realize the knowledge of length, position and orientation of the links of the articulated model is being used as a restriction for local search in the image. However, the retrieval of contour points does not use any knowledge of the geometric model, which is why adjacent slices of the same link are still allowed to present contour points which seem to be independent or unaligned with each other, whereas they should present some kind of continuity along the edges and should be all similarly distant from their counterparts.

Considering the width of the finger links to be known, which makes sense since their length is already assumed to be known, it can be integrated in the algorithm in two ways, both of which assume the width of a finger along a slice is likely to remain the same between image frames:

1. Each finger slice has its own associated width;
2. Finger slices belonging to the same link have the same associated width.

The first option is quite basic and deals with each slice independently, while the second one intends to add more consistency between results of the same frame. The difference between these two options is more evident when the width of the links is being estimated rather than being provided beforehand.

This knowledge of width can be used in two ways: one which merely restricts the search space, regardless of trackers misplacement, and other which also restricts the search space but tries to compensate for misplacements first.

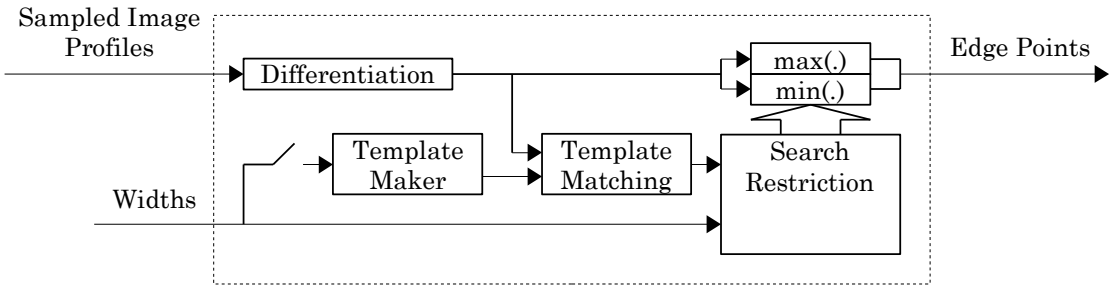


Figure 11.2 - The second alternative method for feature search improves the first one by resorting to the phalanx width to help restrict the peak search space. The usage of template matching is optional.

11.2.1Basic Search Space Restriction

In this case the search space for peak search in an image sample is restricted to the vicinity of where the edge of the link is thought to be, thus avoiding to choose points which lay too far from their corresponding edge.

Given an image sample of size n with an associated finger width of d units, the predicted edges location is given by $c \pm d/2$, where $c=(n+1)/2$ is the centre of the sample. If each search space window has width τ, the peak search is performed within the windows defined by $(c \pm d/2) \pm \tau/2$, with the window limits rounded towards the integer nearest to the window centre. The window limits are only rounded in the end to avoid propagation of rounding errors in the intermediate computations.

11.2.2Compensated Search Space Restriction

Basic search space restriction tries to improve unrestricted peak determination by limiting the search space. However, it may prove insufficient if the link trackers have been misplaced or if the finger in the image described a larger movement, because once the search space is restricted the allowable amount of movement between movie frames is also seriously affected. This is why it is of the uttermost importance to find a solution which tagged with this one will not compromise the range of movement so seriously.

The proposed complementary solution also uses the knowledge of width of the finger links, but this time the purpose is to compensate the misdetermination of the restricted search space so it no longer lies where the edges are thought to be but instead where they are more likely to be located.

This is achieved by a two part process where the first part compensates the misplacement of the trackers by adjusting the centre of the image sample via correlation, and the second part restricts the search space as in basic search space restriction.

The operation called correlation is closely related to convolution. In correlation, the value of an output pixel is also computed as a weighted sum of neighbouring pixels. The difference is that the matrix of weights, in this case called the correlation kernel, is not flipped during the computation, hence the correlation of two discrete 1D signals $\mathbf{s}_1$ and $\mathbf{s}_2$ is defined by

$$R_{\mathbf{s}_1\mathbf{s}_2}(k)=\sum_j \mathbf{s}_1(j)\mathbf{s}_2(j+1-k) .$$

Correlation can be used to locate features within an image by a concept generally known as template matching, where the occurrences of a small signal (the template) are located in a larger signal (the image). This concept also applies to 1D signals: in this case the measured signal is the derivative of the image sample and the template is a synthetic signal, the same size as the derivative, with a positive and a negative peak equally distant of the centre and separated by the associated finger width. The result of this operation is a correlation curve which is stored in a vector, as happened with convolution.

This method is also useful when one of the peaks in the derivative is well defined and narrow but the other is not. By correlating the derivative with an ideal template, there is a stronger probability for the strong peak in the derivative to be aligned with one of the peaks of the template. When the search space is restricted the strong peak is clearly defined, while the determination of the weaker and wider peak is bound to be more correct due to the use of correlation and search restriction.

The fact that one peak in the derivative stands out more than the other is usually related with an illumination which casts shadows under the fingers and lights the upper parts.

11.3 Rectilinearity Restriction in Correlation Surface

Using width in the preceding reinforcement is a way of avoiding misdetermining the locations of the edge points. The use of template matching in the derivative also

compensates for minor misplacement of link trackers and adds some robustness when one of the peaks in the derivative is not well defined as a consequence of a non frontal illumination of the object in the image.

As a consequence of the advantages of using knowledge of width, from this point on when referring to use of knowledge of width or search space restriction the method used is compensated search space restriction using one associated width per link.

This new improvement builds from the need of having aligned edge points of the same link, rather than scattered ones as resulting from determination of peaks in the image profile derivatives. It is also motivated by realizing that the surface formed by joining correlation curves of the same link side-by-side generally presents a rectilineal crest crossing them, which may be explored to yield aligned edge points.

The process called rectilinearity restriction has this name for more than a reason: a finger link can generally be considered a rectilineal segment, but the correlation surface also presents a rectilineal crest, not to mention the fact the edge points should be aligned. However, this method could be modified and still be used if the tracked segment was curved rather than rectilineal. For that matter it would suffice to attend to the correlation surface, for it would still present the rectilineal crest, because it is related with the offset of the template centre perpendicularly to the projected model and not to the curvature of the object.

Hence, the purpose of this method is to evaluate the correlation surface and determine the rectilineal crest. Then, through a parameterization of the curvature of the object, which in this case is a simple line segment, the estimated edge points are estimated compliantly with the collinearity constraint.

There were devised three ways of fitting a line to the crest of the correlation surface, each evolving chronologically as the limitations of the previous one were perceived, with names based on the underlying algorithm:

- Partial linear least squares (PLLS) of correlation maxima;
- Modified weighted PLLS (MWPLLS);
- Iterative MWPLLS (iMWPLLS);

The common part of these methods, the correlation surface, is achieved by flanking

adjacent correlation curves of a link. This means in a link with m associated correlation curves, each described by a vector of size n, the resulting correlation surface is described by a matrix of size $(n \times m)$, where the vectors appear in the same order as their link samples. All these methods were devised before the iterative method in 4.3.2.

After a line is adjusted to the surface, the phalanx width is used to estimate the location of the contour points as happens in 4.3.2.

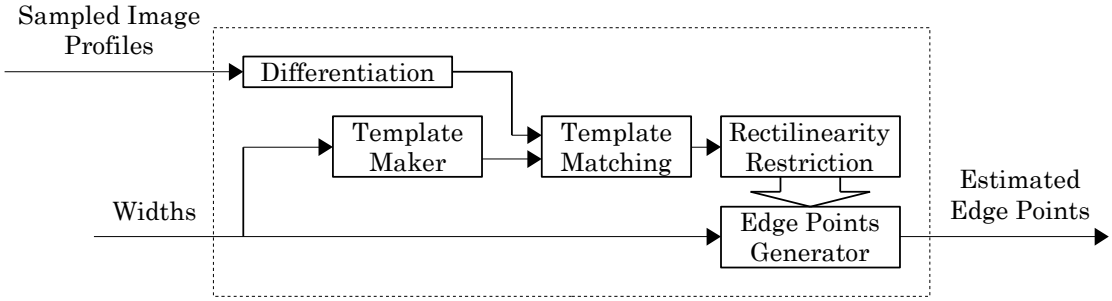


Figure 11.3 - The third alternative method for feature search improves the second one by estimating the edge points instead of directly measuring them. Peak search is no longer done, template matching is mandatory and rectilinearity restriction in the correlation surface is used together with the phalanx width to generate the positions of the estimated edge points.

11.3.1Partial Linear Least Squares (PLLS)

A simple way of determining the correlation surface crest is to find the maximum of each correlation curve and then use them in a regression model.

Given a set of m correlation curves $\mathbf{C}_i(k)$ of size n each, their maxima are computed with $\max(\mathbf{C}_i)$, which corresponds to positions given by

$$m_i = \underset{k}{arg\,max}\, \mathbf{C}_i(k),$$

where $i \in \{1, \cdots, m\}$ and $k \in \{1, \cdots, n\}$.

The fact each maxima is computed in a 1D space and the correlation surface is a 3D space should not cause confusion. The regression is computed in a 2D space, which can be thought as if the position of each maxima was transformed in a set of two coordinates, one given by position itself and other by the position of the correlation curve in the correlation surface. This means curve i with maximum on m_i originates coordinates $(x_i, y_i) = (i, m_i)$ which will be used in regression.

The regression is given by the solution to a partial linear least squares problem,

which is linear because the regression model is a line and partial because the error to minimize is only computed along the y-axis. In the regression model, object x_i will have a measured and a fitted occurrence, respectively represented by y_i and $\hat{y}_i$. The partial errors are measured with $e_i = y_i - \hat{y}_i$, which represents the vertical distance between measured data and the estimated regression model. The purpose of regression is to minimize the squared sum of all these distances, given by

$$E = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} \left(y_i - \hat{y}_i \right)^2 .$$

On the other hand, the linear model is a parameterization of the line, and is written assuming the measured data can be modelled by

$$y = c_0 + c_1 x .$$

Substituting the model equation in the sum equation yields

$$E = \sum_{i=1}^{m} \left(y_i - \left(c_0 + c_1 x_i \right) \right)^2 ,$$

where c_1 and c_0 are the coefficients to be found, respectively representing the slope and the offset along the y-axis of the adjusted line.

To minimize the fitting error, E must be differentiated with respect to each coefficient and the result set to equal zero, leading to

$$\frac{\partial E}{\partial c_1} = -2 \sum_{i=1}^{m} x_i \left(y_i - \left(c_0 + c_1 x_i \right) \right) = 0 \text{ and } \frac{\partial E}{\partial c_0} = -2 \sum_{i=1}^{m} \left(y_i - \left(c_0 + c_1 x_i \right) \right) = 0 ,$$

which can then be rewritten as

$$c_0 \sum x_i + c_1 \sum x_i^2 = \sum x_i y_i \text{ and } m c_0 + c_1 \sum x_i = \sum y_i .$$

Further rewriting these two equations, now in matrix-vector notation, results in

$$\underbrace{\begin{bmatrix} m & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}}_{\mathbf{R}} \underbrace{\begin{bmatrix} c_0 \\ c_1 \end{bmatrix}}_{\mathbf{c}} = \underbrace{\begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}}_{\mathbf{r}} .$$

The coefficients can then be determined solving $\mathbf{c} = \mathbf{R}^{-1}\mathbf{r}$. The estimated data is thus given by $\hat{y}_i = c_1 x_i + c_0$, with an associated error of $e_i = y_i - \hat{y}_i$.

This method depends of a good determination of the maxima in order to properly adjust a line to the crest. If there are spurious maxima points along the correlation curves they are chosen over others if they have greater magnitude. It is known that PLLS is highly influenced by all data, including points which clearly do not belong to

the set, the outliers. The farther away an outlier is located the more it affects regression, inducing changes in offset and skew of the fitted line.

This regression was not used by itself due to these limitations, but is presented as a basis for better understanding the forecoming methods and the rectilinearity restriction used in the SSD surface of the proposed method.

11.3.2Modified Weighted PLLS (MWPLLS)

In order to avoid dependency on correct maxima determination and ponder the fit with more than a candidate per correlation curve, it is necessary to devise a variation of the PLLS method. This variation considers more important for regression those peaks which are well defined in the vicinity of their maximum, and at the same time neglects the influence of sporadic peaks (with high magnitude and narrow base).

The motivation is that when facing two peaks with the same amplitude but where one is narrower than the other, the narrow peak should be considered a spurious influence while the wider peak should provide the bulk of contributions for the regression. Furthermore, if the narrow peak is in fact spurious, it is unlikely to be present in the adjacent correlation curves, hence the regression is more influenced by the wider peaks.

To do this, all points of all correlation curves give their contribution. However, this contribution must have some criterion responsible for giving more importance to some points than to others. The formalization occurs by establishing a cost function where the error (in each curve) is a sum of errors between each point and the estimate, with a cost associated to each error as a way of materializing the criterion.

The value of the costs is a critical aspect to consider and was chosen based on a regional evaluation of the correlation curve rather than a local one: a region with more interest should be associated with a greater cost, so a mistake in the regression is more penalized.

The definition of the MWPLLS follows a deduction analogous to the PLLS case, still considering a cost function defined by

$$E=\sum_{i=1}^{m} e_i^2$$

but with the novelty of each partial error being given by

$$e_i=\sum_{j=1}^{m} w_{ji}\left(y_{ji}-\hat{y}_i\right)=\sum w_{ji}\left(y_{ji}-\left(c_0+c_1 x_i\right)\right)=\sum w_{ji}y_{ji}-\sum w_{ji}\left(c_0+c_1 x_i\right),$$

where w_{ji} is the cost of an error associated with element j of curve i, and y_{ji} is the corresponding measured value.

Differentiating E with respect to the coefficients and setting the result equal to zero yields

$$\frac{\partial E}{\partial c}=2\sum e_i\frac{\partial e_i}{\partial c}=0,$$

where the partial derivatives of the partial error are given by

$$\frac{\partial e_i}{\partial c_0}=-\sum w_{ji} \text{ and } \frac{\partial e_i}{\partial c_1}=-\sum w_{ji}x_i.$$

For ease of representation and further deduction these results are condensed as

$$\frac{\partial e_i}{\partial \mathbf{c}}=-\sum w_{ji}\begin{bmatrix}1\\ x_i\end{bmatrix},$$

which means the partial derivatives of the total error are given by

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{c}}&=-2\sum_i\left(e_i\sum_j w_{ji}\begin{bmatrix}1\\ x_i\end{bmatrix}\right)\\ &=-2\sum_i\left\{\left(\sum_j w_{ji}y_{ji}-\sum_j w_{ji}\left(c_0+c_1x_i\right)\right)\sum_j w_{ji}\begin{bmatrix}1\\ x_i\end{bmatrix}\right\}=0\end{aligned}.$$

Manipulation of the bottom expression then leads to

$$\sum_i\left(\sum_j w_{ji}y_{ji}\sum_j w_{ji}\begin{bmatrix}1\\ x_i\end{bmatrix}\right)=\sum_i\left(\sum_j w_{ji}\left(c_0+c_1x_i\right)\sum_j w_{ji}\begin{bmatrix}1\\ x_i\end{bmatrix}\right),$$

which can be rewritten as

$$\sum_i\left(W_i\begin{bmatrix}1\\ x_i\end{bmatrix}\right)=\sum_i\left(\left(\sum_j w_{ji}\right)^2\begin{bmatrix}1 & x_i\\ x_i & x_i^2\end{bmatrix}\begin{bmatrix}c_0\\ c_1\end{bmatrix}\right),$$

considering

$$W_i=\sum_j w_{ji}y_{ji}\sum_j w_{ji}$$

and noticing

$$\left(c_0+c_1x_i\right)=\begin{bmatrix}1 & x_i\end{bmatrix}\begin{bmatrix}c_0\\ c_1\end{bmatrix}.$$

It could prove relevant to mention there are scalar terms involved, namely

$$\sum_j w_{ji}y_{ji} \text{ and } \sum_j w_{ji},$$

which then originate other scalar terms, which are

$$\sum_j w_{ji} y_{ji} \sum_j w_{ji} \text{ and } \left(\sum_j w_{ji}\right)^2 .$$

The coefficients are then determined with

$$\underbrace{\begin{bmatrix} c_0 \\ c_1 \end{bmatrix}}_{\mathbf{c}} = \underbrace{\left\{ \sum_i \left(\sum_j w_{ji}\right)^2 \begin{bmatrix} 1 & x_i \\ x_i & x_i^2 \end{bmatrix} \right\}^{-1}}_{\mathbf{R}^{-1}} \underbrace{\sum_i W_i \begin{bmatrix} 1 \\ x_i \end{bmatrix}}_{\mathbf{r}} .$$

Baring in mind the mentioned choices, it was verified the curve itself already provides information about cost quantification, each cost being related with the amplitude of the curve in the affected point. The weighting is considered a modified version because it relates to the amplitude of a surface in a 3D space rather than a distance measured in a 2D plain, so it is included in the name of the method to remember that particularity.

The presence of a ponderation is visible in the first sum of W_i. As a consequence of this term the regression will be strongly conditioned to the centre of mass of each correlation curve, since there are no other terms to cancel this effect. This may represent an advantage in some cases but can also be a problem if the correlation surface presents concurrent crests, even if with different amplitudes. In this case the regression is likely to be on the valley between crests and biased towards the crest with a larger volume underneath (the "heaviest" one). The result will also be biased if there is only a crest but that crest is biased to one of the sides.

11.3.3Iterative MWPLLS (iMWPLLS)

This method is basically an improvement of the previous one. Since MWPLLS may get too conditioned to the centres of mass, this option tries to iteratively compensate that by only using points in the vicinity of a regression. Hence, starting with the regression yielded by MWPLLS, the new regression is still computed with MWPLLS but only points close to the regression are considered. Each new regression is used in the next regression iteration. The process is repeated until the regression has converged or until a certain number of iterations is reached.

This method is still conditioned to the centres of mass, although in a much more

compact way because only close points are used. This effect will be more evident if the concurrent peaks fall within the scope of considered points, or if a single crest is biased within the scope.

12 Appendix E: Phalanx Width Estimation

Estimating phalanx width is a way of producing more accurate templates of the image profile derivatives. Normally, the width would be a constant value set by the user, but in this case the goal is to determine that constant with the help of the Kalman filter.

Since the unknown in question is a constant along time, the only part of the Kalman filter needed to be used is the one referring to position. Hence, first and second order aspects such as velocity and acceleration will not be considered. It is important to notice however, that this constant can be determined in different ways, depending if the width is associated with one profile or all profiles of a link.

In the first case the associated width of each profile is estimated with direct application of the position equations of the filter. Replacing position with width should make no confusion.

On the other hand, if all profiles of a link are being used to estimate the width of the phalanx, the filter will have to consider the information supplied by each of them in order to determine the single width.

Hence, if each i-th width is considered independently, the prediction equation is

$$p^{i}_{(k|k-1)} = \hat{p}^{i}_{(k-1)}$$

and the update is

$$\hat{p}^{i}_{(k)} = p^{i}_{(k|k-1)} + g\, e^{i},$$

with $e^{i} = p^{i}_{(k)} - p^{i}_{(k|k-1)}$. But if the N widths are considered as redundant measures of the same single width, then the prediction is set by

$$p_{(k|k-1)} = \hat{p}_{(k-1)}$$

and the update by

$$\hat{p}_{(k)} = p_{(k|k-1)} + g\, \mathrm{E}\left[e^{i}\right],$$

where $e^i = p^i_{(k)} - p_{(k|k-1)}$ and $\mathrm{E}[e^i] = \frac{1}{N}\sum_{i=1}^{N}\left(p^i_{(k)} - p_{(k|k-1)}\right)$. Hence, the update equation can be rewritten as

$$\hat{p}_{(k)} = (1-g)\,p_{(k|k-1)} + \frac{g}{N}\sum_{i=1}^{N} p^i_{(k)}\,.$$