CHASING THE COLOUR GLOVE: VISUAL HAND TRACKING

by

Brigitte Dorner

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
June 1994

APPROVAL

Name:	Brigitte Dorner
Degree:	Master of Science
Title of thesis:	Chasing the Colour Glove: Visual Hand Tracking

Examining Committee:

Dr. William Havens
Chair

Dr. Brian V. Funt
Senior Supervisor

Dr. Ze-Nian Li
Supervisor

Dr. David Lowe
Department of Computer Science
University of British Columbia
External Examiner

Date Approved: April 6, 1994

Abstract

I present a visual hand tracking system that can recover 3D hand shape and motion from a stream of 2D input images. The hand tracker was originally intended as part of a computer interface for (American) sign language signers, but the system may also serve as a general purpose hand tracking tool.

In contrast to some previous 2D-to-sign approaches, I am taking the 3-dimensional nature of the signing process into account. My main objective was to create a versatile hand model and to design an algorithm that uses this model in an effective way to recover the 3D motion of the hand and fingers from 2D clues.

The 2D clues are provided by colour-coded markers on the finger joints. The system then finds the 3D shape and motion of the hand by fitting a simple skeleton-like model to the joint locations found in the image. This fitting is done using a nonlinear, continuous optimization approach that gradually adjusts the pose of the model until correspondence with the image is reached.

My present implementation of the tracker does not work in real time. However, it should be possible to achieve at least slow real-time tracking with appropriate hardware (a board for real-time image-capturing and colour-marker detection) and some code optimization. Such an 'upgraded' version of the tracker might serve as a prototype for a 'colour glove' package providing a cheap and comfortable—though maybe less powerful—alternative to the data glove.

Acknowledgements

My sincerest thanks go to my supervisor Dr. Brian V. Funt, for encouraging and supporting me even when I insisted on following my own ideas rather than to listen to his advise. I greatly appreciate his attitude towards supervision, to lead his students to develop their own ideas and interests and be an advisor and partner for discussion rather than a 'boss'.

I am also deeply indebted to Dr. David Lowe for his help in a critical stage of my work. He provided me with may helpful technical hints, but most of all his support and friendly advise gave me the courage to go through with my plans.

Further, I would like to mention Dr. Alex P. Pentland here, for his encouragement and friendly suggestions about possible improvements of my hand-model formalism, and Dr. William H. Edmondson, for helpful suggestions on sign language interpretation.

Special thanks go to Dr. Edwin Langmann and Eli Hagen. Edwin provided unfailing moral support and many helpful hints on mathematical details. Eli got me involved with American Sign Language understanding and saved me a lot of work by providing me with the necessary background in this area.

Last, but not least, I would like to thank my friends and fellow students—especially Diana Cukierman, Alicja Pierzynska, Sergio Sloseris, David Mitchell, Petr Kubon, Allan James Bennett-Brown, and Jörg Überla—without whom this work might have been finished much sooner, but certainly not with as much fun.

Contents

Abstract iii

Acknowledgements iv

1 Introduction 1

1.1 The ideal system: requirements 3

1.1.1 Sign language characteristics 3

1.1.2 Difficulties waiting around the corner 4

1.2 Previous work 6

1.2.1 General: motion tracking 6

1.2.2 Special: human motion and sign language 7

1.2.3 The rest of the body 9

1.3 Towards a hand tracking system 10

1.3.1 Decision ♯1: a 3D approach 10

1.3.2 Looking for a hand model 11

1.3.3 Decision ♯2: a simple model 13

1.3.4 Decision ♯3: colour-coded markers 14

1.3.5 How to get from 2D to 3D 15

1.4 Finally: The complete hand tracking system 17

1.4.1 Overview in pictures and words 17

1.4.2 The bigger picture: ASL interface 20

2 From Image Sequence to Marker Positions **23**

2.1 The gloves 23

2.1.1 Hardware 23

2.1.2 Designer markers 24

2.2 Marker detection algorithm 27

2.2.1 Searching for 3-ring markers 30

3 The Fitting Algorithm **32**

3.1 Newton in 1D 32

3.2 Applied to our least-squares problem 33

3.2.1 Basic Gauss-Newton algorithm 34

3.2.2 Quasi-Newton extension 36

3.2.3 Steplength 36

4 Modeling a Human Hand **37**

4.1 Anatomy of the human hand 37

4.2 A mathematical model 39

4.2.1 Coordinate systems 41

4.2.2 What we have so far 42

4.3 How to marry Newton and the hand model 42

4.3.1 Prerequisite: projection operator 42

4.4 Derivatives 45

4.5 Adding some ad hoc physiology 47

4.5.1 Range limits 47

4.5.2 Interdependences 50

4.6 Motion constraints 52

4.6.1 Estimating speed (and acceleration) 53

4.6.2 Digression: the 2D alternative 54

5 The Correspondence Problem Revisited **56**

5.1 Measure ♯1: using predicted position as a criterion 57

5.2 Measure ♯2: asking the hand model 58

6 The System in Action **61**

6.1 Quantitative evaluation with simulated input data 61

6.1.1 Tracker versus original 63

6.1.2 Do we need constraints? 64

6.1.3 Loosing track and finding it again 65

6.2 Tracking real image sequences 78

6.2.1 Using the glove with uniquely encoded markers 78

6.2.2 Using the glove with simple markers 79

6.3 Getting started 84

6.4 Speed 85

6.4.1 Marker detection 85

6.4.2 Model fitting 85

7 Conclusion **87**

7.1 Summary: what has been accomplished so far 87

7.2 Outlook: what remains to be done 88

List of Figures

1.1 The hand tracker (overview) 18

1.2 Outline of an ASL interface 21

2.1 Marker design 24

2.2 Encoding scheme for unique markers 26

2.3 Difficult cases for marker identification 27

2.4 Glove with simple markers 28

2.5 Glove with uniquely encoded markers 28

4.1 The human hand 38

4.2 Local coordinate systems for one finger 39

4.3 Summary of hand parameters 40

4.4 A simple camera model 43

4.5 Calculation schema for the hand model 48

4.6 Constraint function to enforce range limits on the hand parameters . 49

4.7 Constraint function to model the interdependence between the last two finger segments 50

6.1 Graphical visualization of tracking results 62

6.2 tracking 'A' (from model-generated input) frame 0 to 4 66

6.3 tracking 'A' (from model-generated input): frame 5 to 9 67

6.4 tracking 'A': errors in recovered position 68

6.5 tracking 'A': errors in palm orientation 68

6.6 tracking 'A': errors in joint angles for thumb 69

6.7 tracking 'A': errors in joint angles for finger 2 69
6.8 tracking 'A': errors in joint angles for finger 3 70
6.9 tracking 'A': errors in joint angles for finger 4 70
6.10 tracking 'A': errors in joint angles for finger 5 71
6.11 tracking 'A' without constraints: errors in recovered position 72
6.12 tracking 'A' without constraints: errors in palm orientation 72
6.13 tracking 'A' without constraints: errors in joint angles for thumb . . . 73
6.14 tracking 'A' without constraints: errors in joint angles for finger 2 . . 73
6.15 tracking 'A' without constraints: errors in joint angles for finger 3 . . 74
6.16 tracking 'A' without constraints: errors in joint angles for finger 4 . . 74
6.17 tracking 'A' without constraints: errors in joint angles for finger 5 . . 75
6.18 tracking 'A' (from model-generated input): frame 0 to 4 76
6.19 tracking 'A' (from model-generated input): frame 5 to 9 77
6.20 tracking 'A' (from real images): glove with uniquely encoded markers 81
6.21 tracking 'D–E' (from real images): glove with uniquely encoded markers 82
6.22 tracking 'time' (from real images): glove with uniquely encoded markers 83
6.23 tracking 'A' (from real images): glove with simple markers 84
6.24 tracking 'C-D' (from real images): glove with simple markers 84

Chapter 1

Introduction

Eye movement tracking experiments show that the objects we pay most attention to are our fellow human beings, especially their faces and upper body parts. This is not surprising, considering that as members of a social species our success as individuals and as a group largely depends on how well we can communicate. The first step towards communication is to spot another individual as soon as possible and correctly infer the other's intentions. Our vision system, accordingly, is very well tailored to detect, identify, and analyze humans and their motion. In contrast, computer vision, until recently, has shown comparatively little interest in human targets. The only area that has received major attention is face location and recognition for identification purposes. The so called 'inverse problem'—extracting 3D spatial information from an image (a '2D projection')—that a computer vision system, and in fact any vision system, has to solve, is a very hard, underconstrained problem. Naturally researchers attacked easier instances of the problem first. The choice was—and largely is—objects and scenes that can be well approximated by simple geometrical shapes. Humans do not fit very well in this category.

Also most practical robot vision systems were developed to perform tasks that humans would like to delegate, such as sorting and handling industrial parts or navigating through living rooms for cleaning purposes. Such tasks usually involve very little, if any, interaction with humans.

Of course, humans always have had to communicate with robots and computers.

For some time a keyboard and a screen were thought to be sufficient for this purpose, but as information processing capabilities increased and computers were used for a wider range of tasks by a more diverse group of users the need for more task-specific and user-friendly interfaces became apparent. On one hand this meant presentation of information in a style that humans would find easier to handle, usually some form of graphic representation reflecting the fact that vision is our most important sense. On the input side progress has not been that dramatic; although typing is slow and bothersome, the keyboard is still largely in place. This bottleneck is the main culprit causing communication between humans still to be much faster than between humans and computers.

In this work I will describe a system that can track a human hand and recognize its 3D shape and position. (I have presented preliminary results on this system in [13]). The major motivation was, and is, to make a first step towards a more user-friendly interface for the deaf or hearing impaired. These people can talk to each other in sign language just as freely and easily as others in spoken language, but problems arise in the communication with people not able to understand sign language. Even though most deaf people learn to write English (or the native spoken language of their country), it is a second language for them, and communicating through writing would be tedious and slow in any case, even in the written form of the spoken language. A computerized sign language interface would help in making communication easier at least with the computer; it could also serve as a front end for a sign-language-to-spoken-language translation system.

Before a successful sign language interface can actually be built there are several other hard problems to be solved: much like in oral speech understanding, the stream of hand shape information has to be segmented into sign language signs, and signs and sign combinations actually have to be interpreted in context. Thus, a sign language interface is a rather long term goal, but the hand tracker in itself has several other, more immediate applications as well: it could be used as a maybe less powerful, but cheaper and more comfortable version of a data glove. As such it could be used, for example, as an input device in virtual reality or to teach a computer realistic hand-motion sequences for animation or robotics. Hand tracking could also be helpful in videophoning: the transmission of images requires much higher capacities than the transmission of sound, and current technologies cannot provide sufficient bandwidth to allow videophoning on a larger scale. An encoding of hand shapes in terms of joint angles would be very efficient to transmit and relatively easy to reconstruct using a graphical hand model.

1.1 The ideal system: requirements

1.1.1 Sign language characteristics

It is not my goal here to actually build a full sign language interface, or even to interpret hand motion in terms of sign language signs. However, the hand tracker *is* designed with mainly such an application in view. Sign language offers a well structured set of meaningful human gestures. If nothing else, this constitutes an ideal pool of test examples for a hand tracking system. The goal of sign language understanding also provides specifications about what kind of output the tracker has to deliver and how much precision is necessary. To clarify this, I will give a short overview of *A*merican *S*ign *L*anguage (ASL)[1].

In a sign language the signer has two different way of expressing him- or herself: by *fingerspelling* words in English or some other spoken language and by signing. Sign languages are genuine languages and can be very different from the spoken language of the country in which the signer lives. There is not necessarily a word-to-sign correspondence, and even if there is, the word might not cover all the meanings of the sign and vice versa.

Even though many signs mimic or exaggerate characteristic features and actions, it is possible to describe at least most of the signs in terms of three parameters. According to Stokoe et al. [45], the action of an 'active' hand in an ordinary[2] ASL

[1]I am mainly concerned with American Sign Language, but most of what is said here about the structure of ASL is valid for other sign languages as well.

[2]The main category of signs that do not fit that pattern is signs that involve people or places. When, for example, a person not present in the room is introduced into a signers sign flow, the signer selects a location in signer space where s/he 'establishes' the person (by spelling the name or making a nickname sign). This person can then be referred to as to a person present in the room. If, for

sign can be uniquely described by:

- **tab** (tabula), the position of the hand at the beginning of the sign,
- **dez** (designator), the hand shape of the hand at the beginning of the sign, and
- **sig** (signation), the action of the hand(s) in the dynamic phase of the sign.

The *tab* can be anywhere in the signer space, which extends from the hips upwards, roughly speaking as far as the signer can conveniently reach with the hands. The signer space is most differentiated around the face, distinguishing about 5 regions there (depending on the classification system), but partitions the remaining regions rather crudely. A list of 12 possible locations for *tab* is given in [45] (p. *x*). In addition, the *tab* can also be somewhere on or near the other (non-active) hand. In this case the *tab* stands for the shape the non-active hand is in and is described by a *dez* position.

The *dez* gives a hand shape, i.e., the position of the fingers and the orientation of the hand in space. Stokoe lists 19 different designators (p. *xi*), most of them similar or identical to characters in fingerspelling (described on p. *xxii* of the same book). The *dez* is often the first letter of the English word for what the sign represents.

The *sig* describes the direction of the hand movement (e.g. up and to the right), a possible rotation of the hand, and the change the hand shape undergoes during this movement. Luckily, the ways in which the hand shape can change during a sign are very limited. There are only two main types, *open* and *close*, along with some variations, plus special-purpose *sigs* such as *wriggling fingers*. Stokoe lists 24 different *sigs* (p. *xii*), two or more of which can be combined to describe fully the dynamic part of a sign, if necessary. Seven of the *sigs* describe the interaction of the two hands. In general, either one or both hands can be 'active' during a sign.

1.1.2 Difficulties waiting around the corner

As mentioned above, sign language recognition can be viewed as speech recognition with visual input data. As such, it has to face most of the problems of traditional

example, the signer wants to express that A gives something to B s/he establishes locations for A and B and then makes the sign for *give*, starting at A's location and terminating at B's location.

speech recognition. One such problem is segmentation: the input is a continuous stream of signs, and the system has to decide when one sign ends and a new sign begins. Another typical problem is the influence of context on the execution of a sign, analogous to differing pronunciation of a word depending on its neighbours and factors such as emphasis, position in the sentence, type of sentence etc.. Signs also vary from person to person and from dialect to dialect.

While all these problems do not directly affect the hand tracking system as such, they do imply requirements for its design: for one thing, we cannot assume that the hand is always in the process of making a regular sign; there will be transitions and irregularities, and the tracker must be able to follow these since it will be impossible to say a priori when exactly the next sign begins.

The large variation in sign execution makes it difficult to describe the characteristics of a sign in terms of 2D images of the signer, a warning that can be found in virtually any book or dictionary on sign language. The essence of a sign is conveyed in terms of the 3+1D[3] properties hand shape, position, and orientation and their change during the execution of the sign. Thus, signing is essentially a 3+1D process, and a sign interpreter will most likely need this kind of 3+1D input (which, of course, implies that the hand tracker has to provide it somehow).

The analysis of the visual input data itself, i.e., the computer vision aspect of the tracker poses a different set of problems and requirements. The human hand is not an easy object to track. It does not have a simple geometrical shape. Moreover, it can move fast and change its shape very quickly. A short experiment shows that it is possible to open or close a hand in 3-4 frames, assuming a frame rate of 60 Hz[4]. In the same time the hand itself can make a 30cm sweep through the air. This is an extreme example, though. Humans cannot perceive any details of hand motion at this speed and sign language signs are (usually) executed at a slower pace. Moreover, the hand shape tends to change only slowly, if at all, during a fast, sweeping motion of the hand as a whole, while a rapid change in hand shape is usually performed in place. This alleviates the problem of tracking fast motion, but we still have to expect

[3]3D space + time component
[4]i.e., 60 frames per second.

a displacement of more than a few pixels between consecutive frames. On top of that, hand motion is sufficiently intricate that there is almost always occlusion of some part of the hand or other. And human motion is complicated in the sense that it is muscle controlled and thus neither describable by a simple physical model with a few parameters nor predictable over more than a few frames.

All these unpleasant qualities make visual hand tracking a hard problem and also exclude many general motion-tracking-and-pose-recognition algorithms that can be found in the literature. In fact, it seems to be even beyond the capability of the human vision system to keep *always* track of exact hand pose and motion. Thus, it is probably more realistic to go for general robustness and a good ability to recover from failures instead of aiming for a perfect system. Preferably, this robustness should take the form of *graceful degradation* behaviour when the system has to deal with difficult, fast shape changes, massive, prolonged occlusion, and/or poor image quality (low resolution of the hand itself as part of an upper body image, motion blur, etc.).

1.2 Previous work

1.2.1 General: motion tracking

There is a vast amount of literature on motion in computer vision. Most work seems to concentrate on analyzing motion between two or three frames rather than tracking objects over long sequences of frames. I will not attempt to give a complete overview, but only describe some of the approaches that seem particularly relevant for my project.

The most relevant work to be mentioned here is an iterative, nonlinear optimization approach to model based object tracking presented by Lowe [32] (see also [31, 49]). In this approach a 3D model of the object to be tracked is generated. Relevant primitives (lines), are extracted from the image frames. Tracking is then performed by an optimization loop that involves calculating the model's current 2D projection to obtain the expected contour and comparing this contour to the detected primitives. The result of this comparison is used to calculate a correction of the model's 3D pose, and the loop is repeated. My system follows Lowe in using an optimization approach and, at least partially, in his choice of the particular mathematical algorithm, an extension of Newton's algorithm for finding the zero crossings of a function to several dimensions.

The other area of special interest that should be mentioned here is tracking feature points of known or unknown objects. Sethi and Jain [44] develop an algorithm that recovers the trajectories for a set of feature points over a sequence of frames using smoothness constraints on direction and speed of the motion. Rangarajan and Shah [40] use a similar approach. Other authors have described extensions including predictions about the expected locations of tracked feature points in subsequent frames (see, for example, [50, 20, 19] and the references given there).

1.2.2 Special: human motion and sign language

There are some systems that track and recover human pose and/or motion, using various techniques (see, e.g., [15, 35, 30, 12]), but they concentrate on motion of the body as a whole, especially limb motion, and are not designed to track the intricate motion of human hands.

Existing approaches to hand tracking and sign language interpretation in particular can be roughly divided into 2D-'direct' (or trajectory-based) and 3D model-based. Direct approaches as in [47, 4, 11, 9] try to identify signs directly from 2D trajectories and silhouettes of hand shapes.

Tamura and Kawasaki [47] develop a classification scheme for signs based on location and/or motion of the hand and the hand shape. Tracking is accomplished by segmenting the skin area from the background. Only one hand and the face are assumed to be in the image, and the background is assumed to be dark to allow easy segmentation and identification of segmented regions. A polygon approximation of the hand contour is calculated for hand shape classification. The system is tested on standard sign language videos, in which the signer starts each sign from a neutral start position and returns to the neutral position after finishing the sign. Sign identification is accomplished using a dynamic programming approach. The system can recognize twenty words of Japanese sign language, though in about half the cases it is only able to extract a set of best candidates.

Charayaphan and Marble [4] use a similar approach, but with more sophisticated techniques. Signs are executed from a neutral start position with two stop positions on the way before the signer returns into the neutral position. The hand is tracked in real time on a PC with special image processing hardware by subtracting successive frames and calculating the center of motion. Signs are then classified by hand location in the two stop positions and shape of trajectory, if necessary. If the system cannot make a definite decision, the hand shape in stop positions is used as an additional means of classification, using the general Hough transform. This system was tested on 31 different signs.

Darell and Pentland [9, 10] use a set of view templates to model the hand. Correlation indices for these templates are calculated for each frame, and gestures are represented as correlation profiles for a (time warped) signing sequence. The system works in real time with special-purpose correlation hardware. The authors show recognition rates for two gestures that are performed interleaved with five other unfamiliar gestures.

Fletcher, Warwick, and Mitchell [20] describe a hybrid algorithm for feature point tracking and demonstrate its performance by recovering the trajectories of a set of light markers on a hand during a simple waving motion. The authors also announce their intention to use their system for ASL understanding.

The gesture recognition system of Davis and Shah [11], another real time system, uses light markers in combination with Rangarajan's and Shah's feature tracking algorithm [40] to find the trajectories of the finger tips. These trajectories are compared to stored gesture models by means of a finite-state automaton. The system's performance is demonstrated on a set of 7 gestures corresponding to ASL signs.

In contrast, model-based approaches try to recover some 3D description of the signer's actions from the sequence of frames. Lowe [32] hints at the possibility of using his system for tracking hand movements, but does not develop this idea further.

Downton and Drouet [15] use a 3D model-based approach to track human upper body motion, with the goal of developing a system for sign language analysis. They use a cylindrical body model and line segments as primitives to be matched (the matching process itself is not described in detail). The system can find a static pose of a signer when the model is initially placed closely enough, but it fails to track a signer in motion "due to propagation of errors".

Cipolla, Okamoto, and Kuno [6] describe a robust algorithm for structure from motion. Their algorithm uses motion parallax to recover structure and motion parameters for a rigid object from a set of two views. The authors show a real-time application in which hand motion is used to steer a computer graphics model of a plane. Four color markers are used as reference points for the algorithm. Since the tracked object is assumed to be rigid, however, the system is of limited use as a hand tracking tool.

Tracking with markers

General human-motion tracking systems often use some kind of marker attached to the major joints of the tracked person (see, e.g., [16, 12, 30]). These systems also sometimes employ more than one camera to eliminate occlusion and ambiguities in identifying markers as much as possible. Trajectories of markers are then used for applications such as medical diagnosis and improvement of athletic performance. This marker method is mainly used for whole body motion tracking where markers are far apart and follow simple trajectories. Two of the systems mentioned above use reflective markers to track hand motion [20, 11], but the examples given only show gestures without marker occlusion and with simple, easy-to-disentangle trajectories.

1.2.3 The rest of the body

All the systems mentioned above concentrate mainly on the hands (or rather, one hand). The face is considered more as a reference point and only given marginal attention. There is evidence (see [48]) that facial expressions might carry mostly redundant information in ASL, but this need not be the case for other sign languages, e.g. German Sign Language (see [36]).

As a prerequisite to face recognition there has been considerable effort to find faces and facial parts in an image (see, e.g., [8, 25, 42, 43]), and the recent interest of the vision community in human gestures has also lead to increased interest in facial expressions as something more than a mere nuisance for face identification. Pentland et al. [37] and Ralescu and Iwamoto [39] describe two different approaches for recognizing human facial expressions—eigenfaces and the analysis of expressive facial regions using fuzzy logic. Both systems would be able to provide enough information about facial expressions for ASL understanding.

1.3 Towards a hand tracking system

1.3.1 Decision ♯1: a 3D approach

Reconstructing the 3D structure of an object from its 2D projection is not easy, especially when the object is as complicated as the human hand. The 2+1D-to-gesture approaches show promising results, more promising anyway than attempts at 3+1D, so why try to recover the 3D structure at all?

As suggested above, the reason is that signs and gestures are performed in a 3+1D world, for human recipients who understand space as 3-dimensional. We perceive signs and gestures in this 3D space and we classify them intuitively according to a natural metric that is easily described in terms of 3+1D properties (though it is *not* identical to the Euclidean metric in 3D or in 3+1D).

Gestures may be called similar because they differ only in palm orientation, handedness, duration, etc.. The exact form of this natural metric depends on the set of gestures or the sign language in question. The metric tells us what—in which context—is considered as relevant information. If we want to build a system for sign language understanding we need to recognize a huge number of signs, from noisy and incomplete input data, and the execution of these signs may vary considerably from our prototype examples. The ideal metric for this task provides a search space for gesture matching in which prototype samples are optimally distributed, and the intuitive metric that we use as humans is probably as close to this ideal as we can get. Hence it would be wise to use it as a base for sign recognition.

Since the signer is observed with a camera the projection of the 3+1D signer space into 2+1D can hardly be avoided. Some information is irretrievably lost in this process (due to occlusion etc.), and the rest is crammed into a much smaller space. Searching and matching in this cluttered space is more sensitive to errors, and the intuitive metric we would like to use does not translate very well into this lower dimensional space: for example, the similarities between a fist facing the viewer and a fist held parallel to the viewer are easily recognized in 3D, but not in 2D. Thus, a system that identifies gestures in 2+1D has to work in a cluttered feature space and without the benefit of a good metric.

Of course, reconstruction of 3+1D information cannot give the lost information back, but, after all, most of this information is also lost for human observers, after all, who always recognize gestures and signs without having a full 3D view. Though I have to start out from 2+1D space and hence cannot completely avoid the cluttering effect and its implications either, I can employ knowledge about the hand's properties and projective geometry to disentangle noise from relevant information. Once back in 3+1D, it is possible to choose a good metric for sign identification. The 3+1D approach also has the advantage that the tracking system can be used for any desired sign language or set of gestures/hand motion, with or without a high-level interpretation process.

1.3.2 Looking for a hand model

My system follows the traditional model-based-object-tracking-and-pose-recovery cycle of

$$\begin{tikzcd}
\text{project model} \arrow[rr] & & \text{compare to image data} \\
& \text{update model} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

but unlike most other approaches of this kind my system does not include a model that can be directly matched to the image.

The human hand is not a simple geometrical object. Even just to model its surface adequately is difficult: fingers are cylinder-like and the palm can be approximated by some sort of box or polyhedron, but such a representation would hardly be good enough to produce contours that can be matched directly to the image without further processing and abstraction of image data such as connecting edge pixels into lines. This preprocessing of image data would be a difficult task as well since the outlines of finger segments do not always fit a description as a line or another simple mathematical object. The matching problem is further aggravated by additional features of the hand, such as finger nails, creases and patterns on the skin, variations in skin colour etc.. They all create spurious edges that have to be accounted for somehow.

Of course there are more sophisticated building blocks available on both sides of the fence: there are methods that allow better description of arbitrary contours and may even be able to follow these contours from frame to frame (e.g., snakes, see, e.g., [29]), and geometrical objects such as geons or generalized cylinders allow much more sophisticated models without significantly increasing model complexity. Still, it is hard to imagine that any of these one-primitive-object-per-rigid-segment, low-complexity approaches to modeling could provide flexible enough basics for an object like the human hand.

An alternative approach to modeling, as it is proposed by Lowe [32], would be to give up a simple, globalized representation in favour of a computer-graphics-like approach featuring a localized surface representation. This sure is (or can be made) sufficiently flexible to model a human hand well enough that its projection could be compared to the image data even on a pixel by pixel base. Unfortunately it is costly to compute the projection of such a local representation, and the task of creating such a model in the first place is daunting.

Keeping the model simple means one can do fast and easy projection, a rather important feature: the size of the search space for possible candidate poses grows exponentially in the number of interdependent free parameters[5]. To reduce complexity one could consider fitting in a hierarchical way, but with a crude hand model it is advisable to include as much correspondence information as possible right from the start, to reduce the influence of false matches and to avoid getting trapped in false

[5]For a human hand one would typically assume 5 chains of 10-11 interdependent parameters each (see chapter 4), though these parameters have a limited range of possible values

minima.

The question is whether or not it is really necessary and/or desirable to include any detail of an object that is likely to leave significant traces in the image, even though the detail is irrelevant to the kinematic structure of the object. Models get more unwieldy with every detail that is included, they use up more storage space, and they get more user-specific. If even warts cannot be left out one has to adapt much more than a few simple parameters, such as finger length, to switch to a different user.

1.3.3 Decision ♯2: a simple model

A simple model is hard to compare to the image data, while a realistic model is expensive to build and use. My modeling approach was originally motivated by the hope of finding a way out of this dilemma without having to give up the advantages of a 3D model-based approach.

Additional inspiration came from psychological experiments on human motion perception, especially so-called *biological motion* (rotation of limbs around joints) pioneered by Johansson [27, 28]. These experiments show that humans can recognize the motion of a test person in three dimensional space just from the motion of light markers, commonly known as MLDs (*M*oving *L*ight *D*isplays).

Tartter and Knowlton [48] showed that such joint markers on the hand and fingers even provide enough information for humans to understand hand motion: two test subjects, fluent ASL signers, were provided with gloves with protruding light markers. Although each of them only saw a live video image of the light spots of the other's hand (plus a light marker on the tip of the nose) they were able to conduct a natural conversation after a short period of accommodation.[6]

MLD experiments may well lead to interesting insights into psychological aspects of motion perception and possible internal abstractions we make somewhere in our

[6]The main objective of the experiment was to investigate possibilities for data compression to reduce the bandwidth required for videophone transmissions. Hence the transmission of the data via video; if the subjects had been allowed to actually see each other they might have used 3D stereo clues that cannot be transmitted over a videophone.

cortex to represent the characteristics of a particular motion. Independent of such speculations we can at least conclude, though, that biological motion is well described by the motion of two points on the limb—the point where the limb connects with the joint around which it is moving, and another fixed point further out on the limb, preferably the end point.

From a mathematical point of view this is no big surprise, of course, but it suggests that a hand model need not necessarily contain more than joints and finger tip locations to represent the hand adequately for motion tracking purposes. Of course, there is the difficulty of comparing this kind of model to the image data as joint locations and finger tips are not a priori identifiable in an image. If they are not specially emphasized, as in the MLD experiments, they may not even be recognizable as primary features of one single image; a joint may reveal itself only as a fixed point of motion over a sequence of frames. Thus a joints-only hand model demands considerable preprocessing and data abstraction before the actual fitting process can be started. There is a crucial detail, though: the preprocessing can all be done in 2D. In particular, it should be possible to identify joints and their 2D motion by an analysis of image flow or related 2D motion descriptions.

1.3.4 Decision $\sharp$3: colour-coded markers

This said, I still decided to start out with a system that uses markers to provide the locations of joints and finger tips. Unfortunately, existing MLD systems may not be adequate for something like hand and finger tracking, though. From the viewpoint of computer vision MLDs provide enhancement of relevant feature points and hence eliminate detection problems, but the correspondence problem—which marker belongs to which joint—is still very much there.

In fact, even humans sometimes have difficulties keeping track of individual markers in MLD experiments. In a walking sequence, the marker attached to the wrist of a walking person seems periodically to turn into that person's knee, giving the overall impression of an arm-swinging chimpanzee, and in the sign language experiment described above the signers had considerable difficulties with the more intricate hand movements used in fingerspelling. Although the authors do not offer an explanation for these difficulties, they may well be due to marker confusion.

The MLD systems for hand tracking mentioned in section 1.2 do not attempt to track any signs that involve intricate marker trajectories or occlusion. Even though the feature-point-tracking algorithm of Fletcher and Mitchell [20]) can handle some limited occlusion, it would probably fail when whole parts of the hand are hidden for a longer time.

I suggest a simple method of disambiguating markers: colour coding. Special hardware makes it possible to detect a set of colour coded markers in real time (see [6]), but colour identification does have its problems (lighting conditions, cast shadows, etc., see 2.1). Since the vision system can be placed in a well controlled environment for this application, however, such difficulties should be manageable.

I do not attach my markers directly to the skin (as it is sometimes done with MLD markers), but use painted cotton gloves. Since the straight forward encoding—one colour per marker—requires too many distinct colours I experimented with two versions of colour coding that use fewer colours: a simple one-colour-per-finger scheme that allows easy marker detection, but does not completely eliminate the correspondence problem, and a scheme in which a marker is encoded by a combination of a joint- and a finger-colour. These colour markers are described in more detail in section 2.1.2. Here it should be just noted that even colour coding can only alleviate, but not eliminate, the correspondence problem unless one can achieve a 100% correct marker identification rate. Thus I had to include means to deal with correspondence ambiguities into my system, even for the experiments with the unique encoding scheme.

1.3.5 How to get from 2D to 3D

Back to the fitting cycle at the beginning of this discussion: reconstruction of a 3D (or even a $2 + \frac{1}{2}$d) scene from a 2D image is known to be a hard and usually underconstrained problem. Even if we have a model of the object in the image, together with a set of correspondence pairs between points in the model and in the image, this does not necessarily remove ambiguities; there might still be more than one possible 3D pose of the object that projects onto the same 2D description.

In any case it is far more difficult to write down a set of equations that render the 3D pose of an object from a set of projected points than it is to calculate the projection of a 3D object. Mathematically speaking, projection is a nonlinear, non-invertible operation (though it is usually possible to get at least partial information about the original pose). Even to extract the six pose parameters for a simple rigid object from a set of projected points requires some inventiveness (see [26, 6]). Although it would be nice to have such an explicit set of formulas allowing reconstruction of the hand shape from the marker correspondence pairs, this is a daunting task.[7]

The next best (or next worst) possibility is to reformulate the pose recovery problem as a search or optimization problem: if we cannot directly infer the 3D pose of the model from its 2D image, we can at least calculate what the projection of any pose of the model would look like. Thus it is possible to find a hand shape that comes at least close to the image by simply trying out a number of different shapes. One could do an exhaustive search over all possible hand shapes (after a proper discretization of the search space) to find the correct solution, or any set of alternative solutions in case the problem does not have an unique solution. Unfortunately, the size of the search space grows exponentially in the number of interdependent parameters, though. With only two values for each parameter that makes $2^4 = 16$ possibilities just for the shape of one finger and $2^3(2^4 \times 5) = 640$ possible hand shapes.

Even if such a coarse description of hand shape should prove sufficient for sign language understanding—not that it is likely that it would—it might well happen that the closest hand shape in terms of marker proximity on this rough search grid would have little to do with what one would like to get: something that is is at least close to the shape one would get by taking the correct hand shape and mapping each parameter onto a binary space.

I am attacking the problem in a different way, using a gradient descent method that moves the model closer to the pose in the image through a series of iterative steps.

[7]Pose recovery requires at least *three* linearly independent points per rigid object (assuming the position of the points on the object is known). Hence we cannot use the marker scheme to simply reconstruct the 3D hand shape segment by segment.

This kind of local optimization approach sifts through the search space of possible hand shapes in an intelligent way, and it only searches a small part of this search space, but its result depends on the initial shape of the model and it may not find the correct solution. My optimization algorithm is based on an extension of Newton's method for finding the zero crossing of a function to several dimensions (see [22]) that is especially suited for least squares problems (the model fitting problem can easily be formulated as a least squares problem, see chapter 3).

1.4 Finally: The complete hand tracking system

1.4.1 Overview in pictures and words

The hand tracking system can be roughly divided up into two parts: marker extraction from the image and fitting of the model according to the marker locations.

The first module (box A) extracts 2D marker, or rather, colour blob positions from the raw image. As mentioned, the correspondence problem can hardly be eliminated completely. Thus, box B, a module that resolves ambiguities in marker correspondence, is necessary even for the version with uniquely encoded markers. As far as positioning of box B inside the system is concerned one has to consider a tradeoff between efficiency and reliability: on one hand, ambiguous marker assignments should be detected and resolved as early as possible, preferably already in box A, to avoid additional work for covering each alternative in the fitting routine. Unfortunately, in difficult cases knowledge of the physiology of the human hand is needed to find the correct assignment. This kind of knowledge is embedded in the hand model and can only be accessed by way of the fitting routine (box C). Thus box B belongs to both stage 1 (marker detection) and stage 2 (model fitting), and it may be needed more than once for each image. The feedback loop $B \rightarrow C \rightarrow D \rightarrow B$ is used to put 'questions' to the hand model: "which marker assignment looks better?" (see chapter 5 for details).

Once the correspondence problem is solved the actual fitting can begin, and the output, a description of the current shape and position of the hand, can be passed

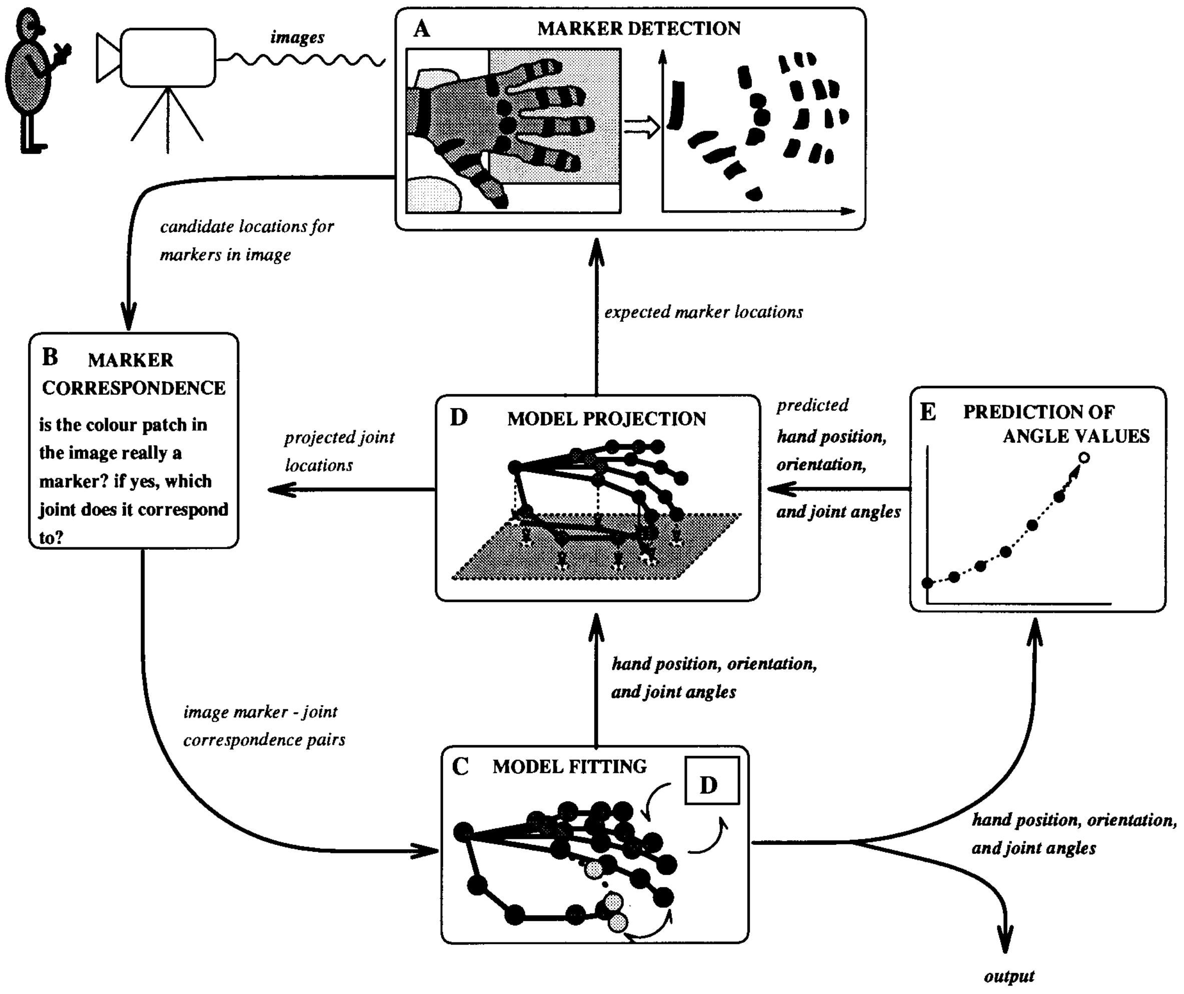
images
A MARKER DETECTION
candidate locations for markers in image
expected marker locations
B MARKER CORRESPONDENCE
is the colour patch in the image really a marker? if yes, which joint does it correspond to?
projected joint locations
D MODEL PROJECTION
predicted hand position, orientation, and joint angles
E PREDICTION OF ANGLE VALUES
hand position, orientation, and joint angles
image marker - joint correspondence pairs
C MODEL FITTING
D
hand position, orientation, and joint angles
output

Figure 1.1: The Hand Traker

on for further processing. Before the next image is tackled, however, box E, the prediction module, becomes active. Its purpose is to predict the hand motion in the time interval to the next frame and, accordingly, what hand shape to expect next.

Box D projects the hand model onto an imaginary image plane and thus translates information about marker positions in the hand model into 2D image coordinates. This projection is used at many points in the system, whenever the hand model has to be compared to 2D image data.

In the following chapters I will provide more detailed descriptions of these modules and their interactions:

- **Chapter 2** deals with the issues relevant for stage 1: glove design and marker detection [box A (and, to some extend, box B)].

- **Chapter 3** provides insights into the mathematical wheels and springs that make the fitting algorithm tick [box C].

- **Chapter 4** traces the design of the hand model, from the properties of the life original down to the details of mathematical formulation as it is needed for the fitting algorithm [box C, D, and E].

- **Chapter 5** is all about the uglier instances of the correspondence problem that can not be tackled at an earlier stage and have to be handed on to the fitting algorithm. It describes two versions of the $B \rightarrow C \rightarrow D \rightarrow B$ feedback loop, one for the simple-marker version, the other for the uniquely encoded marker version [box B].

- In **chapter 6** the fruit is reaped and inspected: test runs on both simulated and real image data show successes and limitations.

- Last but not least, **chapter 7** sums it all up and provides some further outlook.

1.4.2 The bigger picture: ASL interface

Figure 1.2 shows the hand tracker in the context of a, so far, hypothetical ASL interface. (A preliminary account of such an ASL interface, joint work with Eli Hagen, will appear in [14].)

The visual interface has to provide low-level interpretation and abstraction of the visual input data. Information contained in the image that is relevant to ASL understanding—hand shape and motion and facial expression—has to be extracted and cast into a form suitable for further processing. The design of at least part of this visual interface is the topic of this thesis.

The information extracted from the visual input is then handed on to a sign segmentation (and interpretation) system. The purpose of this module is similar to speech segmentation and phoneme identification in the processing of spoken language: the module has to find out when a new sign begins and it has to recognize the particular sign. With sufficient information from the visual interface this last part is relatively easy, given reasonable input and a good 'dictionary' (see sections 1.1.1 and 1.3.1). As in speech recognition the trickiest part is sign segmentation. It is tempting and, for a start, reasonable, to avoid the issue and deal only with isolated signs, but this should not be taken as an excuse to assume the 'isolated-signs' hypothesis for the hand tracker as well.[8] Ultimately, the goal is to understand fluent signing, and a visual interface that requires artificial isolation of signs, such as start and/or stop positions, maneuvers itself into a dead end. There are several interesting prototype systems based on input from a data glove that recognize human gestures and/or sign language signs [18, 17, 2, 46]. Since the information provided by the data glove—hand position, palm orientation, and two flex angles for each finger—is very similar to the output provided by the hand tracker the data glove could be easily replaced with the more comfortable colour glove in any of these systems.

The ASL interpreter, finally, has to 'understand' what the signer wants to express and translate this information into some internal representation suitable for further processing. Depending on the range of possible context or complexity of expected

[8]Of the gesture and sign recognition systems described in section 1.2, only the systems of Davis and Shah [11] and Darell and Pentland [9] work on continuous sign streams.

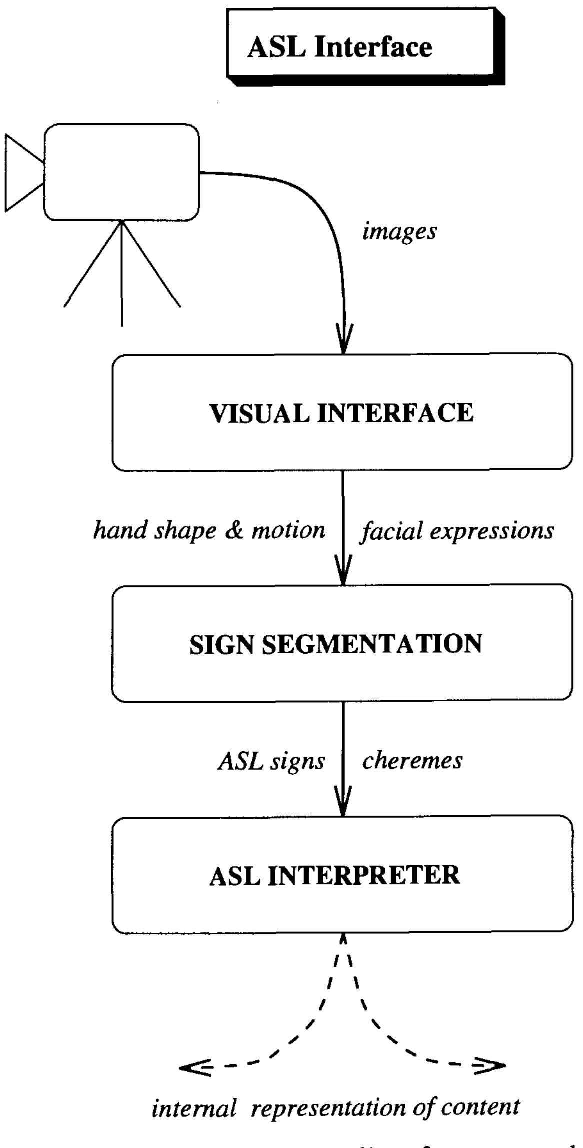

Figure 1.2: Tentative outline for a complete ASL interface: the hand tracker would roughly fill up the black box labeled 'visual interface', though a complete system would also have to have some rudimentary understanding of facial expressions.

expressions this can be a straight forward or a very difficult endeavor. Again there is a strong parallel to understanding more conventional spoken or written languages, though ASL expressions can have a number of peculiarities, such as quasi-parallel execution of signs, that are not found in any spoken language (see [24]).

Chapter 2

From Image Sequence to Marker Positions

2.1 The gloves

2.1.1 Hardware

Even though the painted glove may be seen as a main characteristic of the hand tracker, it is *not* an integral part of the system in itself—only the markers are. In fact, the one-size-fits-all cotton lab gloves I am using in my experiments are far from optimal since the markers hardly ever lie where they are intended to be. The major recommendation of cotton gloves is that they can be easily painted with fabric paint. This fit problem has a nice side effect though: the system is tested under rather adverse conditions. That it still works shows some robustness. The switch to a different user should only require minor adjustments, if any.

Still, it would definitely help to use tight-fitting rubber gloves (surgeons, painters?) if there was a feasible way to colour them. Markers directly attached to (or painted onto) the fingers would be even better, of course, though maybe not from the viewpoint

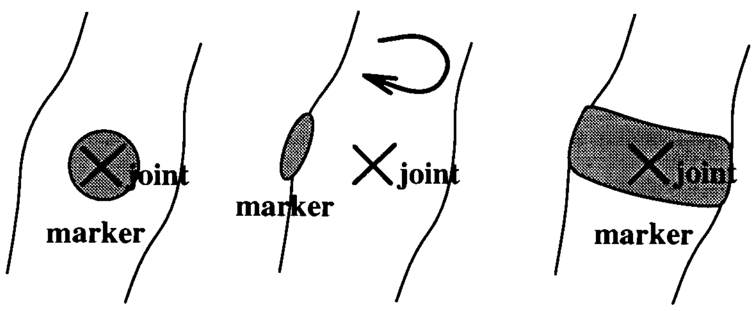

Figure 2.1: Marker design: round or square markers are sensitive to orientation of the joint. Markers of this type will correctly indicate the actual joint location only as long as their surface is turned towards the camera. Ring-shaped markers always give a correct joint location, no matter which way the joint is turned.

of the user.[1]

2.1.2 Designer markers

Traditional MLD markers are typically round or square reflectors. This makes them very much features of the *surface*, which is fine, as long as the markers face the camera. In most of the standard MLD applications (whole body tracking, gait analysis), where the target is moving in a line more or less parallel to the image plane, this is not a problem. The actual joint, could it be seen, would appear pretty much at the same position in the image as its marker.

In ASL signing hand and fingers are constantly rotated in space, however, and markers will often turn away from the camera. Thus, a round or square blip attached to the surface would be occluded (though the joint is perfectly visible), or at least it would not indicate the real position of the joint any more (see figure 2.1).

My suggestion is to use (the center of) ring-shaped markers instead, wherever possible[2]. No matter which way the joint is turned, the markers will be visible and its center still roughly correspond to the joint position. With this trick it is possible

[1]Tartter and Knowlton [48] used gloves with dice-shaped, protruding markers sown on for their ASL experiments and Davis and Shah [11] mark their gloves with something that looks like tape, but in other light marker systems the markers actually are stuck right onto the subject (e.g. [19]).

[2]The markers for CMC, MCP2, and MCP5 (see figure 4.1) can at least be made semi-ring-shaped. The only joints for which I had to resort to traditional round patches are the inner MCPs (MCP3 and MCP4).

to keep the hand model free from any kind of surface representation and concentrate on the essentials, the joints.

As mentioned in the introduction (see page 15) the second idea is to use colour coding to identify markers. Potentially this provides a 3D space to distribute the markers in (instead of the 1D intensity space). Unfortunately, colour is not only a property of the object's surface, but also of the spectral distribution and intensity of the light illuminating the surface. The dependency on spectral distribution only would create problems if the light source changes during the experiments, though. Variations in intensity can be neutralized by using *chromaticity values*, i.e., $\frac{r}{r+g+b}$, $\frac{g}{r+g+b}$, instead of raw r–g–b values, but this means that we have to sacrifice one dimension of the colour space[3].

How many, and which colours are actually distinguishable in chromaticity space ultimately depends on the quality of the equipment: the camera sensors, but also (uniformity of) glove colouring and lighting conditions. I found it impossible to use a distinct colour for each joint on the hand (the 16 joints + 5 finger tips, to be more precise), but even with much better equipment it would be hard to implement this simplest of encoding schemes and accommodate yet another 21 distinct markers for the other hand.

Thus, there are two alternatives: either allow more than one marker with the same colour or use more than one colour per marker to encode markers. Alternative one allows easy marker detection without any further processing on the detection level, but reintroduces the correspondence problem. Alternative two keeps the correspondence problem to a minimum, but requires a more sophisticated marker detection algorithm and some thought about a good encoding scheme.

Such an encoding scheme is harder to find than one might think: since the hand is very flexible and can appear from virtually any perspective it is surprisingly difficult to prevent unexpected popping up of 'impostor markers'. Colour patches on any two parts of the hand can easily become neighbours in the image to create the unexpected and undesired illusion of a marker; and there is no way—short of restricting hand movement—to avoid this. And such incidental markers are not the only problem;

[3] $\frac{r}{r+g+b} + \frac{g}{r+g+b} + \frac{b}{r+g+b} = 1!$

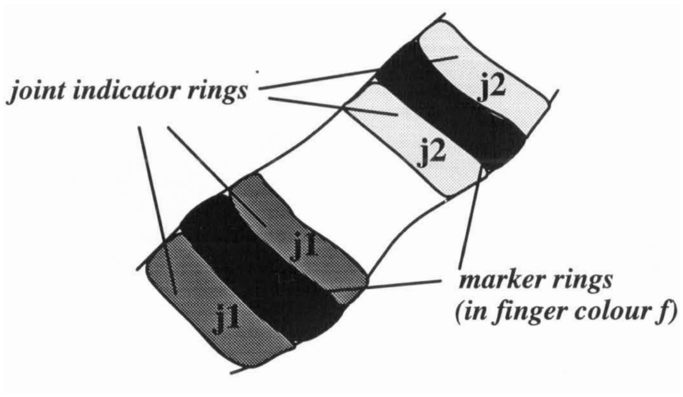

Figure 2.2: Encoding scheme for unique markers: the scheme requires 10 distinct colours, 5 to indicate the finger (+1 for the wrist), another 4 to determine joint type or indicate a finger tip. A marker consists of three rings: the actual marker in finger colour in the center surrounded by two joint-indicator rings.

sophisticated patterns tend to disappear, courtesy of motion blur or lack of resolution, and an encoding that requires visibility of the whole marker is of doubtful value as one may be left with little information to fit the model to.

After some experimenting I found an encoding scheme that, even though it is far from perfect, reduces marker detection problems to a tolerable amount[4]. It developed out of the scheme of simple one-colour markers that I tested as an alternative—one colour for each finger plus one colour for the wrist. These are still used as the actual joint markers, and their center is taken as the location of the joint. An additional pair of rings—one ring on each side of the ring with the main colour—indicates the joint type. With three joints plus tip for each finger this requires four additional colours. Thus the encoding requires ten colours altogether (for one hand), not including the background colour for the glove (black) (see figure 2.2 and 2.5).

Rings can be made wide enough to resist even considerable motion blur and low resolution. The two joint-indicator rings ensure good marker recognition rates even for partially occluded markers, and the matrix-like scheme of colouring minimizes

[4]'Tolerable' in the sense that it is possible to build a reasonable tracking system on top of the marker detection module. The error rate for the sequences presented in chapter 6 lies at about 1 to 2 minor errors per frame, i.e., errors, that can be detected and corrected by the system. Major errors that lead to serious 'misinterpretations' of hand shape occurred in 2 out of 10 of the sequences with the unique encoding scheme.

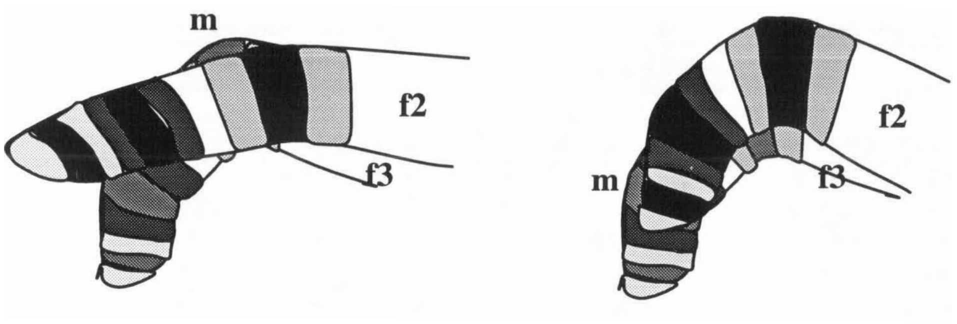

Figure 2.3: Difficult cases for the marker identification algorithm: in both cases the main ring of marker **m** would probably be paired with joint-indicator rings from the occluding finger (f2), but in the first case—the more common one—this still leads to correct marker labeling.

creation of spurious markers or incorrect marker labeling. Problems of this kind are likely to occur when the hand is seen from the side and a marker is partially occluded. If the finger-coloured ring is still visible it may be paired with the joint-indicator of a marker on the occluding finger (see figure 2.3). For most hand shapes this joint will be the joint of the same type and hence with the same joint colour, though, and the marker is still labeled correctly.

2.2 Marker detection algorithm

In order to save time and eventually build a real time system the marker detection algorithm should be as simple as possible.[5]

In fact, for the simple one-colour-per-finger encoding scheme all there is to do is to detect 'blobs' of a specific colour, which could be done in many ways, preferably with special-purpose hardware (see [6]). Failing this (and having therefore to rely on a software solution) I decided for a simple region-growing algorithm: the image is searched for a 'seed pixel' of one of the desired marker colours and the marker around the seed is then expanded into a maximal contiguous area of the seed colour:

[5]Yet another argument against fancy encoding schemes: any kind of shape and/or pattern analysis would be too time consuming.

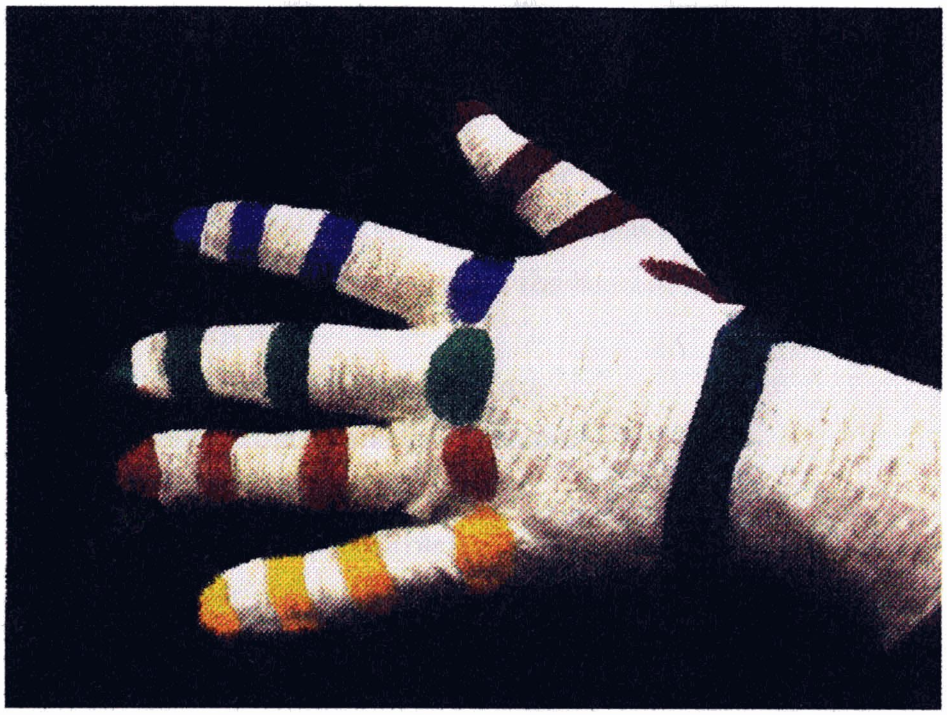

Figure 2.4: Glove with simple markers

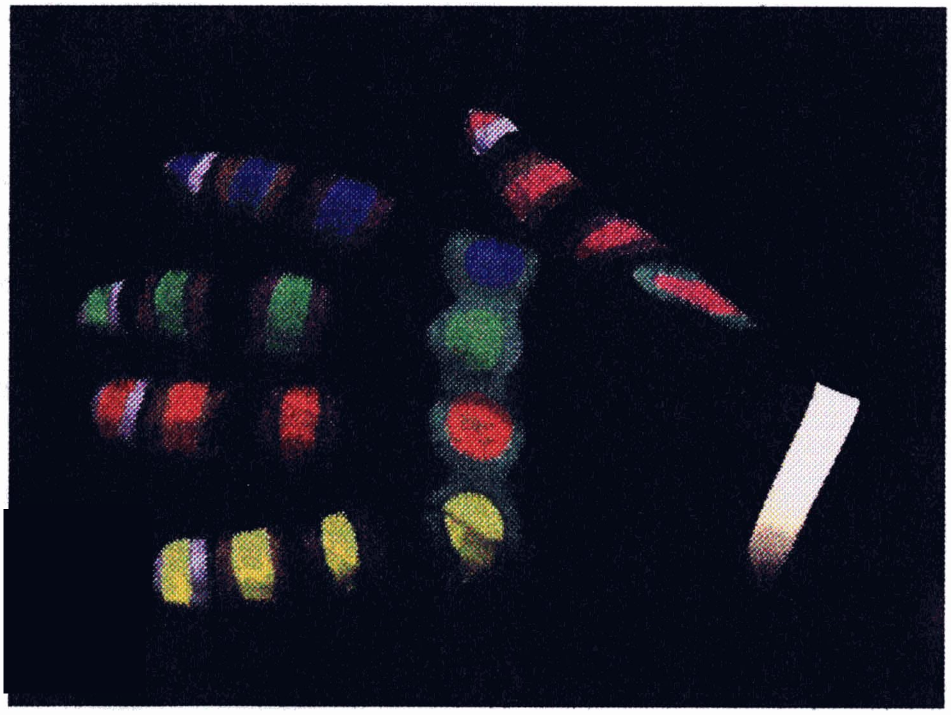

Figure 2.5: Glove with uniquely encoded markers

searching the image for simple colour markers

```
for each pixel $p = (x, y, colour)$ in the image:
    if (pixel has marker colour) and (pixel is not marked)
        \* reset cumulative pixel coordinate counters:
        they will be needed to determine the marker center *\
        xsum = ysum = 0
        current colour = this marker colour
        grow–region–around–pixel(x, y, colour)
        if (region large enough)
        \* [i.e., ♯ of pixels on marker > THRESHOLD VALUE] *\
            marker center = $(\frac{xsum}{\sharp\, of\ pixels\ on\ marker}, \frac{ysum}{\sharp\, of\ pixels\ on\ marker})$
            add marker to candidate markers for current colour
```

recursive region-growing

```
grow–region–around–pixel(x, y, col)
    if (pixel at (x, y) is marked as visited) or
    (pixel at (x, y) does not have marker colour col) return
    xsum = xsum + x
    ysum = ysum + y
    mark pixel at (x, y) as visited
    grow–region–around–pixel(x+1, y, col)
    grow–region–around–pixel(x−1, y, col)
    grow–region–around–pixel(x, y+1, col)
    grow–region–around–pixel(x, y−1, col)
```

The colour of a pixel in the image is determined by its Euclidean Distance in chromaticity space to the nearest marker colour. If this distance is less than a threshold value[6] the pixel is accepted as a marker pixel.

Chromaticity space does not allow any distinction between white and grey—or black, for that matter—so, strictly speaking, I should not have used the white marker for the wrist and the blue-grey for the finger tips. However, the bright white usually reflects enough intensity to saturate all three camera sensors and black will be black, i.e., near zero intensity, no matter what illumination. One does not risk too much by searching for white, black, and grey by thresholding in raw r-g-b space.

2.2.1 Searching for 3-ring markers

To detect and identify the markers used in the unique encoding scheme requires some additional effort: after the finger-indicator ring in the middle is established, which can be done using the simple region-growing algorithm described above, we need to find at least (part of) one of the two joint-indicator rings to identify the marker.

A simple extension of the above algorithm will do the job: "do not stop when the edge of the marker is reached, but search the next n pixels in the direction away from the marker". These pixels then vote for a joint colour, and the marker is finally identified with the joint that gets the most votes.

revised and extended version of the region-growing algorithm:

```
grow-region-around-pixel(x, y, col, direction)
        if (pixel at (x, y) is marked) return
        if (pixel at (x, y) does not have colour col)
                search-vicinity-of(x, y, direction, depth = 0)
                return
        xsum = xsum + x
        ysum = ysum + y
```

[6] 0.05 for seed pixels, 0.15 for region growing.

```
        mark pixel at (x, y) as visited
        grow-region-around-pixel(x+1, y, col, 'right')
        grow-region-around-pixel(x-1, y, col, 'left')
        grow-region-around-pixel(x, y+1, col, 'up')
        grow-region-around-pixel(x, y-1, col, 'down')
```

searching for joint-indicator rings

```
search-vicinity-of(x, y, direction, depth)
    if (depth > MAXDEPTH) return
    if (pixel at (x, y) has any joint colour 'j') add 1 vote for joint 'j'
    case (direction) of
        right: search-vicinity-of(x+1, y, 'right', depth+1)
        left: search-vicinity-of(x-1, y, 'left', depth+1)
        up: search-vicinity-of(x, y+1, 'up', depth+1)
        down: search-vicinity-of(x, y-1, 'down', depth+1)
```

The actual search depth depends on the expected size of the marker, resp. its rings, in the image. The pixels immediately at the edge of two rings usually do not have the pure colour of either joint- or finger-indicator ring and are therefore useless for marker identification. A large search depth, on the other hand, slows down the algorithm and increases the risk that a marker is assigned to the wrong joint because the search region contains part of another marker's joint-indicator ring(s) (see figure 2.3). A MAXDEPTH of around 10 produced good results for the type of images shown in chapter 6, but this might have to be adapted for lower-resolution images.

Chapter 3

The Fitting Algorithm

To find the optimal shape and position of the hand (i.e., the shape and position of then hand that is 'closest' to the image) I use a standard NAG-library routine for nonlinear, continuous optimization known as *Quasi-Newton Algorithm* (see [34], section 3.2). I will shortly describe the basic principle and then show how the method can be applied to solve our fitting problem.

3.1 Newton in 1D

Given a smooth function $f(x)$, we want to find an x^* such that $f(x^*) = 0$. Developing $f(x)$ in a Taylor series around x yields

$$f(x + \Delta x) = f(x) + \frac{df(x)}{dx}\Delta x + \frac{1}{2}\frac{d^2 f(x)}{dx^2}(\Delta x)^2 + \ldots. \tag{3.1}$$

Assuming that $f(x)$ is 'close enough' to $f(x^*)$ and that the function is 'sufficiently' linear in this neighbourhood we can approximate

$$f(x + \Delta x) \approx f(x) + \frac{df(x)}{dx}\Delta x. \tag{3.2}$$

Thus, setting $f(x^*) = f(x + \Delta x) = 0$ implies

$$f(x) \approx -\frac{df(x)}{dx}\Delta x, \tag{3.3}$$

and we have obtained a correction vector Δx such that $x + \Delta x \approx x^*$. If f is really a linear function, these equations are exact, and the method finds the zero crossing in one step[1]. Otherwise we have to repeat the calculation, replacing $x^{(k)}$ by $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$ in each iteration step, until $f(x^{(k)}) \approx 0$.

If the function is reasonably well behaved and we have a good initial guess, Newton's method is extremely efficient (convergence is quadratic), though there are a few pathological cases where the algorithm fails to converge. The routine is less likely to succeed, however, when the initial guess is too far away from the desired result and the search interval includes local maxima and minima. Luckily we need not worry too much about these bad global convergence properties since our goal is model based object *tracking*, which usually guarantees a good initial approximation to the desired solution.

3.2 Applied to our least-squares problem

We want to adjust the pose of our hand model in such a way that the difference between the projected joints and finger tips of the hand model and the actual marker positions in the image becomes minimal. More precisely, let $\vec{\alpha}$ be a vector containing the n parameters that describe shape, orientation, and position of the hand (see chapter 4, for my hand model $n = 26$) and let $\mathcal{P}(\vec{\alpha})$ be a vector describing the projection of the hand model, i.e., a vector containing the projected positions of joints and finger tips. Then the difference between projected marker positions and marker positions in the image can be written as a vector

$$\vec{\mathcal{D}}(\vec{\alpha}) = \vec{\mathcal{P}}(\vec{\alpha}) - \vec{M} \tag{3.4}$$

where $\vec{M}$ is the vector of observed marker positions. The function

$$\mathcal{F}(\vec{\alpha}) = \sum_{i=0}^{m} \mathcal{D}_i(\vec{\alpha})^2 = \vec{\mathcal{D}}(\vec{\alpha})^T \vec{\mathcal{D}}(\vec{\alpha}), \tag{3.5}$$

a scalar, gives the *least-squares error*, or the square of the Euclidean distance between projection and image markers. Thus, $\mathcal{F}$ is a measure for the quality of the recovered

[1]Except for the degenerate case $f(x) = const.$, of course.

shape.

Our goal is to find an $\vec{\alpha}^*$ that minimizes $\mathcal{F}(\vec{\alpha})$,

$$\min_{\vec{\alpha}}\{\mathcal{F}(\vec{\alpha})\} = \vec{\alpha}^*. \tag{3.6}$$

Let $\vec{g}(\vec{\alpha})$ be the *gradient vector* (the vector of first derivatives) of $\mathcal{F}(\vec{\alpha})$, i.e.,

$$\vec{g}(\vec{\alpha}) = [\frac{\partial\mathcal{F}(\vec{\alpha})}{\partial\alpha_0}, \frac{\partial\mathcal{F}(\vec{\alpha})}{\partial\alpha_1}, \dots, \frac{\partial\mathcal{F}(\vec{\alpha})}{\partial\alpha_n}]^T \tag{3.7}$$

and $\mathbf{G}^{\mathcal{F}}(\vec{\alpha})$ be the *Hessian matrix* (matrix of second derivatives) of $\mathcal{F}(\vec{\alpha})$, i.e.,

$$G^{\mathcal{F}}_{ij}(\vec{\alpha}) = \frac{\partial^2\mathcal{F}(\vec{\alpha})}{\partial\alpha_i\alpha_j}. \tag{3.8}$$

Then we can write down the following (sufficient) conditions for $\vec{\alpha}^*$ to be a minimum of $\mathcal{F}(\vec{\alpha})$:

1. $\|\vec{g}(\vec{\alpha}^*)\| = 0$. ($\|\cdot\|$ denotes the Euclidean norm.)

2. $\|\mathbf{G}^{\mathcal{F}}(\vec{\alpha}^*)\|$ is positive definite.

3.2.1 Basic Gauss-Newton algorithm

Following Newton's idea, we want to construct a sequence

$$\vec{\alpha}^{(k+1)} = \vec{\alpha}^{(k)} + s^{(k)}\vec{p}_N^{(k)} \tag{3.9}$$

converging towards $\vec{\alpha}^*$, where $\vec{p}_N^{(k)}$ is a correction vector for $\vec{\alpha}^{(k)}$ that determines the direction of search (or descend) and $s^{(k)}$ is a scalar steplength. Analogous to equation 3.3 we can find $\vec{p}_N^{(k)}$ using

$$\mathbf{G}^{\mathcal{F}}(\vec{\alpha}^{(k)})\vec{p}_N^{(k)} = -\vec{g}(\vec{\alpha}^{(k)}). \tag{3.10}$$

The gradient vector $\vec{g}(\vec{\alpha})$ can be written in terms of $\vec{\mathcal{D}}$ and its derivatives as

$$\vec{g}(\vec{\alpha}) = 2\mathbf{J}(\vec{\alpha})\vec{\mathcal{D}}(\vec{\alpha}) \tag{3.11}$$

where $\mathbf{J}(\vec{\alpha})$ is the *Jacobian matrix* of $\vec{\mathcal{D}}(\vec{\alpha})$, i.e.,

$$J_{ij}(\vec{\alpha}) = \frac{\partial\mathcal{D}_i(\vec{\alpha})}{\partial\alpha_j}, \tag{3.12}$$

and for the Hessian matrix we find

$$\mathbf{G}^{\mathcal{F}}(\vec{\alpha}) = 2[\mathbf{J}(\vec{\alpha})^T\mathbf{J}(\vec{\alpha}) + \mathbf{B}(\vec{\alpha})], \tag{3.13}$$

with

$$\mathbf{B}(\vec{\alpha}) = \sum_{i=0}^{m} \mathcal{D}_i(\vec{\alpha})\mathbf{G}_i(\vec{\alpha}) \tag{3.14}$$

and $\mathbf{G}_i(\vec{\alpha}) \equiv \mathbf{G}^{\mathcal{D}_i}(\vec{\alpha})$ the Hessian matrix of $\mathcal{D}_i(\vec{\alpha})$.

Since we expect $\mathcal{F}(\vec{\alpha})$ to be small around $\vec{\alpha}^*$, $||\vec{\mathcal{D}}(\vec{\alpha})||$ should be small compared to $||\mathbf{J}(\vec{\alpha})^T\mathbf{J}(\vec{\alpha})||$. Hence, we may approximate $\mathbf{G}^{\mathcal{F}}(\vec{\alpha})$ by $2\mathbf{J}(\vec{\alpha})^T\mathbf{J}(\vec{\alpha})$ and get by without actually having to calculate second derivatives of $\vec{\mathcal{D}}(\vec{\alpha})$.

Substituting equations 3.11 and 3.13 into equation 3.10 we get

$$\mathbf{J}(\vec{\alpha}^{(k)})^T\mathbf{J}(\vec{\alpha}^{(k)})\vec{p}_N^{(k)} = -\mathbf{J}(\vec{\alpha}^{(k)})^T\vec{\mathcal{D}}(\vec{\alpha}^{(k)}) \tag{3.15}$$

or

$$\mathbf{J}(\vec{\alpha}^{(k)})\vec{p}_N^{(k)} = -\vec{\mathcal{D}}(\vec{\alpha}^{(k)}), \tag{3.16}$$

a linear equation for $\vec{p}_N^{(k)}$.

If the rank of $\mathbf{J}$ is < 26 (the length of $\vec{\alpha}$: the number of parameters in the hand model) the problem is underconstrained and there is no unique solution, a situation one would naturally like to avoid. The set of constraints added to enforce a solution close to the shape predicted for this frame —one constraint per parameter (see section 4.6)—also has the virtue of preventing such disasters, even when there are not enough markers visible to guarantee 26 independent entries into $\vec{\mathcal{D}}$.

Hence the rank of $\mathbf{J}$ can be assumed to be > 26, which means the problem is overconstrained and we have to settle for a least-squares approximation of $\vec{p}_N$. Thus, we have to solve

$$\min_{\vec{p}_N^{(k)}} ||\mathbf{J}(\vec{\alpha}^{(k)})\vec{p}_N^{(k)} + \vec{\mathcal{D}}(\vec{\alpha}^{(k)})||. \tag{3.17}$$

There seems to be a consensus in the numerics community that the proper way to do this is to use *S*ingular *V*alue *D*ecomposition of $\mathbf{J}$: to find $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ such that

$$\mathbf{J} = \mathbf{U}\begin{bmatrix} \mathbf{S} \\ 0 \end{bmatrix}\mathbf{V}^T \tag{3.18}$$

where $\mathbf{S}$ is a diagonal matrix containing the singular values ('pseudo eigenvalues') of $\mathbf{J}$, $\mathbf{U}$ is an $m \times m$ orthonormal matrix, and $\mathbf{V}$ is an $n \times n$ orthonormal matrix. Further details and an alternative method of solving this linear least-squares problem can be found in [22] resp. [32].

3.2.2 Quasi-Newton extension

Unfortunately, our approximation of $\mathbf{G}^{\mathcal{F}}(\vec{\alpha})$ as $2\mathbf{J}(\vec{\alpha})^T\mathbf{J}(\vec{\alpha})$ may not be valid any more when $\vec{\alpha}^{(k)}$ is too far away from $\vec{\alpha}^*$ or when $\mathcal{F}(\vec{\alpha})$ is too 'flat' around $\vec{\alpha}^*$. The Quasi-Newton algorithm takes this into consideration by calculating a Newton-style approximation to $\mathbf{B}(\vec{\alpha})$. The idea is to avoid costly numerical estimation of $\mathbf{G}^{\mathcal{F}}(\vec{\alpha})$, but nevertheless include some information about the curvature of $\mathcal{F}(\vec{\alpha})$ along the latest search direction when slow convergence of the basic routine indicates that $2\mathbf{J}(\vec{\alpha})^T\mathbf{J}(\vec{\alpha})$ does not provide a good approximation of $\mathbf{G}^{\mathcal{F}}(\vec{\alpha})$. In a Quasi-Newton scheme the $k+1^{st}$ approximation to $\mathbf{B}$, $\mathbf{H}^{(k+1)}$, is given by

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{C}^{(k)}. \tag{3.19}$$

Gill [22] describes several such Quasi-Newton schemes that make use of the current gradient vector to compute a correction matrix $\mathbf{C}^{(k)}$ of rank one or two. For details, see also [23].

3.2.3 Steplength

All that remains to be done now is to select a proper steplength $s^{(k)}$. We have to solve

$$\min_{s^{(k)}}\{\mathcal{F}(\vec{\alpha}^{(k)} + s^{(k)}\vec{p}_N^{(k)})\}, \tag{3.20}$$

a one-dimensional optimization problem. NAG's Quasi-Newton algorithm does this by fitting a cubic polynomial to $\mathcal{F}$ (as a function of $s^{(k)}$; for details, see [34], section 3.2.4 and 2.4.1).

Chapter 4

Modeling a Human Hand

4.1 Anatomy of the human hand

My hand model describes a simplified version of the anatomy of a human hand in mathematical terms.

A human hand (normally) consists of 15 joints (including the wrist) and some 20+ bone segments in between, constituting palm and fingers. The joints are named according to their location on the hand as M*eta*C*arpo*P*halangeal* (joining fingers to the palm) or I*nter*P*halangeal* (joining finger segments) (see figure 4.1).

A short self-inspection confirms that the nine interphalangeal joints can be accurately described as having only one degree of freedom, *flexion–extension.* The situation is more complicated for the metacarpophalangeals, though: all five of them are described in the literature (see, e.g., [3, 1]) as saddle joints with two degrees of freedom, with the capability of *abduction–adduction* or *yaw* in the plane defined by the palm, in addition (and perpendicular) to flexion–extension as in the interphalangeals. This is clearly the case for the four 'proper' fingers, but the abduction–adduction capability of the thumb at the metacarpophalangeal seems to be very restricted.

Most of the thumbs flexibility originates in the C*arpo*M*eta*C*arpal* joint. Buchholz and Amstrong [3] confirm this to be another saddle joint with the two degrees of freedom described above, but also claim that "because of incongruity between the trapezium and the metacarpal base and laxity of the ligaments in the area" it would

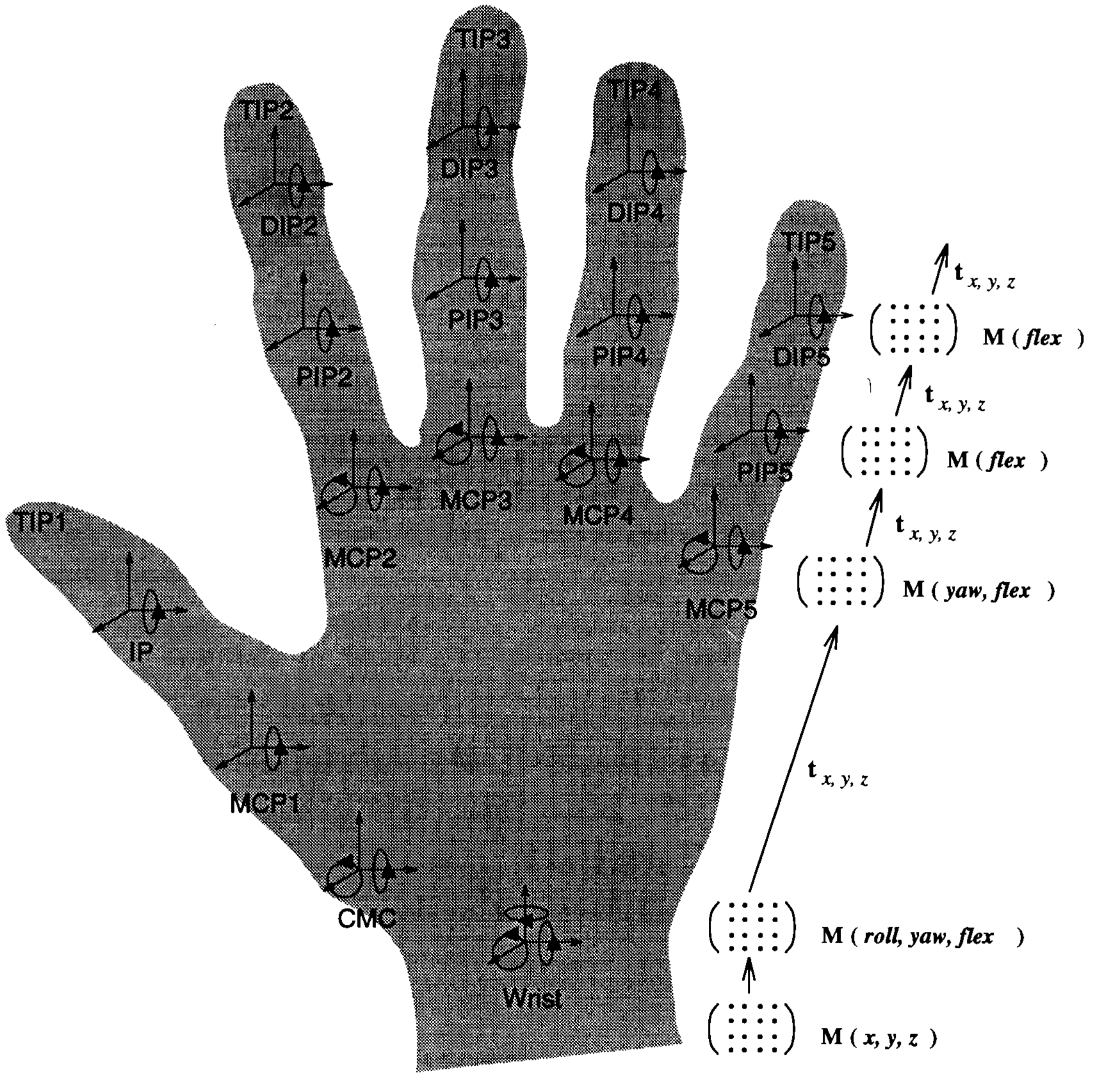

Figure 4.1: View of a human hand (back of right hand). The icon coordinate systems represent the joints and their respective degrees of freedom. The joints are named after their position on the hand: *Interphalangeal* joints—**P***roximal* **I***nter***P***halangeals* and **D***istal* **I***nter***P***alangeals*—join finger segments, **M***eta***C***arpo***P***halangeal* joints connect fingers to the palm; the **C***arpo***M***eta***C***arpal* joint of the thumb is optically part of the palm. In the mathematical hand model, the joints are represented as ideal, i.e., all rotation takes place around a common origin, the idealized center of the joint. Each hand segment is assigned its own local coordinate system, located in the center of the joint that connects it to the previous segment. The relationship between two adjoining segments (i.e., the transformation from one local coordinate system to the other) is expressed in terms of a unified matrix that represents the current rotation at the connecting joint and the translation necessary to shift the origin of one local system into the next.

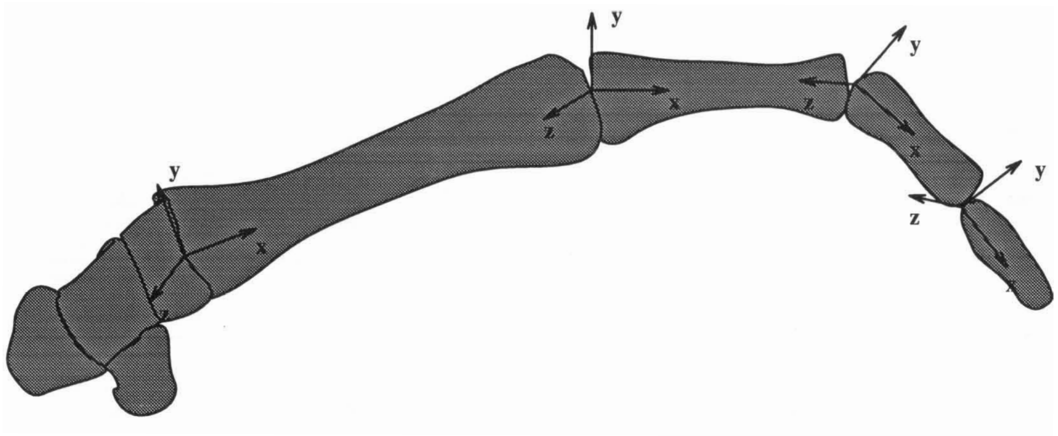

Figure 4.2: Local coordinate systems for one finger: the x-axis corresponds to the main axis through the bone of the finger segment. The coordinate system is defined in such a way that a flexion-extension movement at the joint corresponds to rotation around the z-axis.

be more appropriate to assign a third degree of freedom—rotation around the main axis of the thumb—to this joint.

Both this pseudo-rotation at the CMC and the abduction–adduction at the MCP of the thumb are ignored in my hand model. The variation in hand shape effected by any of the two parameters seems to be negligible for sign language interpretation purposes. Including them would only increase the complexity of the hand model, and thus of the fitting algorithm.

4.2 A mathematical model

For the mathematical description of the hand skeleton an approach commonly used in computer graphics comes handy: the hand is seen as consisting of various rigid subparts—the palm and finger segments—living in their own local coordinate systems. These subparts are connected via coordinate transformations relating their coordinate systems. In this scheme the hand is a hierarchical tree of skeleton segments, starting with the palm as the root and ending in the finger tips.

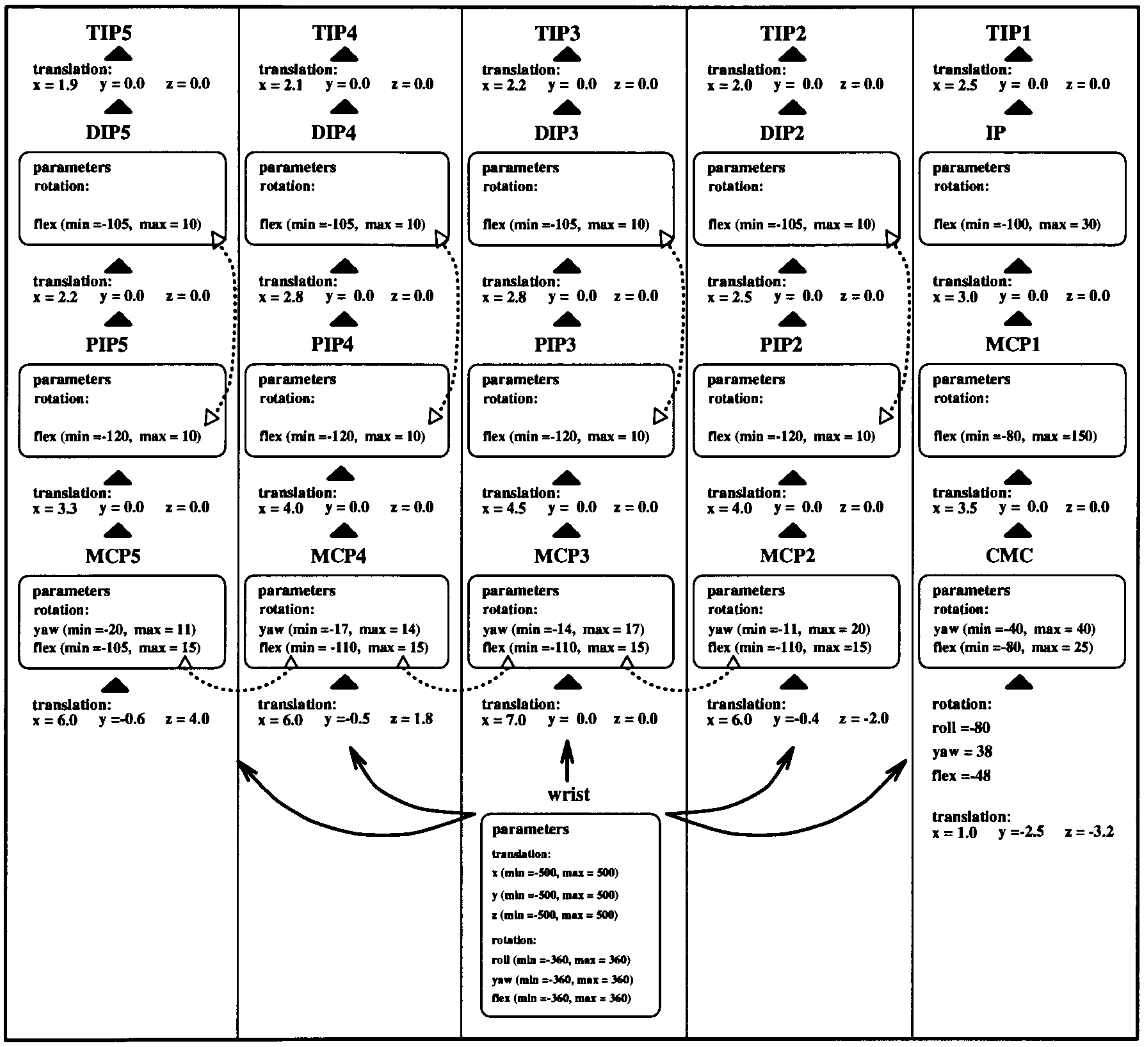

Figure 4.3: Summary of relevant hand parameters: the chart gives the fixed relations between the local coordinate systems of adjoining segments (translation of origin and, for the CMC, rotation into opponent position). The number and specification of parameters attached to each joint (its degrees of freedom), together with range limits on these parameters, can be found in the *parameter* box attached to each joint. The dotted arrows indicate the most important interdependences between parameters incorporated into the hand model.

The position of features such as joints (and markers) can be given in the local coordinate system of the segment, independent of the current hand shape, since the information about the hand shape is contained in the coordinate transformations. These coordinate transformations are best formulated in terms of *homogeneous* coordinates[1] describing rotations of the main axes followed by a translation of the origin.

The reference coordinate system for the hand model is located in the idealized center of the wrist joint. This is yet another complicated saddle joint with three degrees of freedom, but the wrist is not actually included into the hand model. Rather, the hand is viewed as a disconnected object moving freely in space, and the six degrees of freedom (3 for translation + 3 for rotation) assigned to the wrist actually express the relation between object-centered and viewer-centered coordinate systems (or, in other words, the position and orientation of the hand in space).

4.2.1 Coordinate systems

As mentioned above, the origin of the hand's reference coordinate system is set in the wrist. The x-axis points from the wrist through the fully extended middle finger, the y-axis gives the direction perpendicular to the palmar plane, pointing out of the back of the hand, and the z-axis lies in the palmar plane, pointing away from the thumb. Thus, a right handed coordinate system describes a right hand, and vice versa.

The local coordinate systems of the various finger segments are located in the center of the (idealized) joints connecting the segment to the palm or to the next segment towards the palm. The x-axis always corresponds to the main axis or the segment's bone. The remaining two axes are positioned in such a way that flexion of the joint corresponds to a rotation around the z-axis (see figure 4.2; the local coordinate system inherits the handedness of the main system of the hand.) According to Buchholz and Amstrong ([3]) abduction–adduction can be described as rotation around the y-axis in this local system, provided it is considered *before* flexion–extension in the composition

[1]In a world of 3×3 matrix operations translation would have to be represented by a vector addition. For the sake of a simpler formalism we can add an additional row (and column) to our matrices instead, which also allows us to express translation by a matrix multiplication and thus get a *homogeneous* representation for both rotation and translation (for details, see [21]).

of the transformation matrix.

The 'reference hand shape' relative to which all transformations are given looks much like figure 4.1: fingers and thumb are extended in the palmar plane, but all four fingers are aligned with the x-axis of the main coordinate system, not abducted as in the picture. For this reference shape the local coordinate systems for all fingers, except the thumb, are simply translated versions of the main system (see table 4.3 for details). An additional (fixed) rotation at the CMC brings the thumb into its opponent position; Buchholz and Amstrong [3] give $flex = 48°$, $yaw = 38°$, and $roll^2 = 80°$.

4.2.2 What we have so far

We now have a mathematical description of the hand skeleton in terms of idealized joints and rigid segments. The joints constitute the origins of local coordinate systems for the adjoining finger segments. The *shape* of the hand is defined by the set of transformation matrices that relate the local coordinate systems of the segments; or, more precisely, by the respective amount of flexion–extension and abduction–adduction at the various joints, which enters the transformation matrices in form of rotation angle values. The hand has 15 joints, each capable of flexion–extension, and 5 of these are also capable of abduction–adduction. Thus, a particular hand shape is determined by $15 + 5 = 20$ parameters, plus $3 + 3$ parameters for position and orientation of the hand in signer space.

4.3 How to marry Newton and the hand model

4.3.1 Prerequisite: projection operator

In order to compare a given 3D pose of the hand model to the evidence in the image, the hand model has to be projected onto the image plane of a virtual camera, i.e., we have to simulate the process of taking a picture or video image of the hand model. A

[2] rotation around the x-axis; should be considered last in composing the transformation matrix.

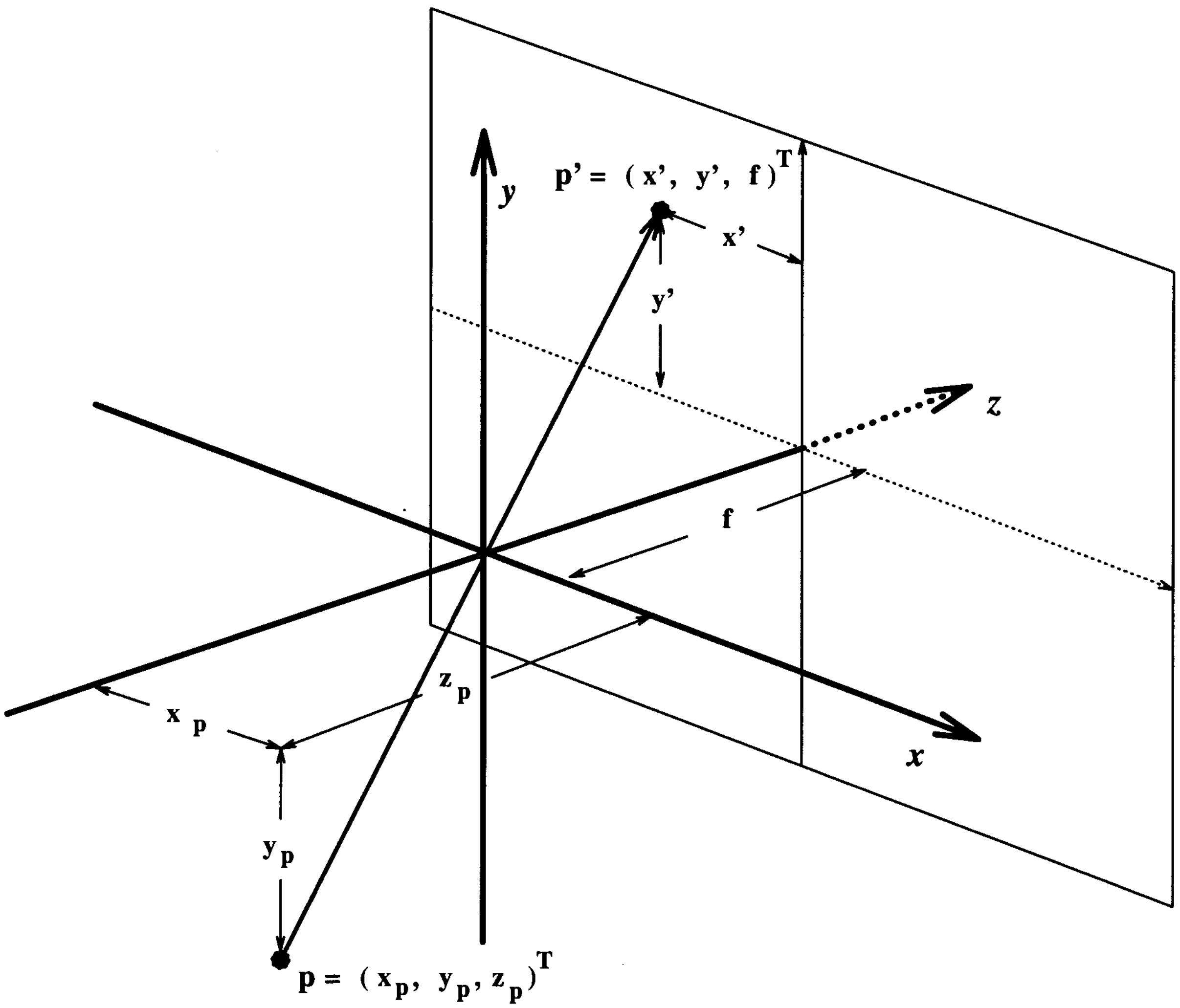

Figure 4.4: A simple camera model: the point p, located at $[x_p, y_p, z_p]^T$ is projected along a ray through the center of an imaginary lens or *pinhole* (in the origin of the coordinate system) onto the image plane located at $z = f$ (the focal length of the camera). p', the projection of p has coordinates $[x', y', f]^T$.

simple 'pinhole' camera model will do just fine for this purpose: in figure 4.4, imagine a camera with its lens (the pinhole) sitting at the origin and its image plane parallel to the x-y-plane at z-value f (the focal length of the camera). A point p with coordinate vector $[x, y, z]^T$ is projected along a ray of light through the pinhole at the origin onto the image plane at $[x', y', f]^T$. Solving the resulting equation

$$\nu[x, y, z]^T = [x', y', f] \tag{4.1}$$

for x' and y' respectively yields

$$x' = \nu x = \frac{fx}{z} \text{ and } y' = \nu y = \frac{fy}{z}. \tag{4.2}$$

To get the projection of a point on any segment of the hand one first has to calculate its position in terms of the viewer coordinate system (the coordinate system of the camera) using the ladder of transformation matrices. Then equations 4.2 can be used to get the point's position in the image plane.

Thus, the function that describes the projection of the hand model—or, more exactly, of a joint marker p with local coordinates $[x, y, z]^T$ in the hand model—onto the image plane is defined as

$$\mathcal{P}(\vec{\alpha}, [x, y, z]^T) = \begin{bmatrix} \frac{x'f}{z'} \\ \frac{y'f}{z'} \end{bmatrix} \tag{4.3}$$

where

$$[x', y', z', 1]^T = \mathbf{M}_\Lambda(\vec{\alpha})[x, y, z, 1]^T \tag{4.4}$$

and $\mathbf{M}_\Lambda(\vec{\alpha})$ is the product of transformation matrices relating the local coordinate system of the finger segment on which p lies to the viewer coordinate system (Λ can be viewed as an index selecting the appropriate $\mathbf{M}$ for the finger segment containing p).

As it is, this function describes the projection of one point, respectively marker, only. To get the projection function $\mathcal{P}$ needed for the Quasi-Newton algorithm as described in section 3.2, we simply have to cram the projections of all markers into one long vector. Since the marker positions on the hand (the $[x, y, z]$s) depend only on the user's hand measures and are fixed during the whole tracking process we can write $\mathcal{P}(\vec{\alpha})$ instead of $\mathcal{P}(\vec{\alpha}, [x, y, z]^T)$.

4.4 Derivatives

Unfortunately, the Quasi-Newton algorithm does not only call for $\vec{\mathcal{P}}(\vec{\alpha})$, but also for its derivatives with respect to the components of $\vec{\alpha}$.[3] With some additional computational effort (evaluating $\vec{\mathcal{D}}$ twice, at $\vec{\alpha}$ and at $\vec{\alpha} + \delta\alpha_i$, for each column of the Jacobi matrix) the partial derivatives of $\frac{\delta \mathcal{D}_)(\vec{\alpha})}{\delta \alpha_j}$ can be evaluated numerically, but deriving an analytical formula for these partial derivatives is not as forbidding as it may look at the first glance:

The partial derivatives for the x component are given by

$$\frac{\partial(\frac{x'(\vec{\alpha})f}{z'(\vec{\alpha})})}{\partial\alpha_i} = f\,\frac{\frac{\partial x'(\vec{\alpha})}{\partial\alpha_i} z'(\vec{\alpha}) - x'(\vec{\alpha})\frac{\partial z'(\vec{\alpha})}{\partial\alpha_i}}{[z'(\vec{\alpha})]^2} \tag{4.5}$$

Analogous for the y component.

Since, by equation 4.2,

$$x'(\vec{\alpha}) = m_{11}(\vec{\alpha})x + m_{12}(\vec{\alpha})y + m_{13}(\vec{\alpha})z + m_{14}(\vec{\alpha}) \tag{4.6}$$

$$z'(\vec{\alpha}) = m_{31}(\vec{\alpha})x + m_{32}(\vec{\alpha})y + m_{33}(\vec{\alpha})z + m_{34}(\vec{\alpha}) \tag{4.7}$$

with $m_{ij}(\vec{\alpha})$ the ijth element of $\mathbf{M}_\Lambda(\vec{\alpha})$, we have to calculate $\frac{\partial m_{ij}(\vec{\alpha})}{\partial\alpha_k}$ or, equivalently, $\frac{\partial \mathbf{M}_\Lambda(\vec{\alpha})}{\partial\alpha_k}$.

Now each $\mathbf{M}_\Lambda(\vec{\alpha})$ is a product of rotation and translation matrices and can be decomposed into basic components M_j that are either constant or depend on one parameter α_i only. For example, such a decomposition for the MCP2–PIP2 segment would look like

$$\begin{aligned}\mathbf{M}_{MCP2-PIP2}(\vec{\alpha}) &= \underbrace{\mathrm{M}_{trans_x}(\alpha_0) \times \mathrm{M}_{trans_y}(\alpha_1) \times \mathrm{M}_{trans_z}(\alpha_2)}_{\text{position in space}} \times \\ &\quad \underbrace{\mathrm{M}_{yaw}(\alpha_4) \times \mathrm{M}_{flex}(\alpha_5) \times \mathrm{M}_{rot}(\alpha_3)}_{\text{orientation}} \times \\ &\quad \underbrace{\mathrm{M}_{trans_{wrist-MCP2}}}_{\text{distance wrist–MCP2}} \times \underbrace{\mathrm{M}_{yaw}(\alpha_{10}) \times \mathrm{M}_{flex}(\alpha_{11})}_{\text{pose of MCP2–PIP2 segment}}\end{aligned}$$

[3]Actually, what is required is the derivatives of $\vec{\mathcal{D}}(\vec{\alpha}) = \vec{\mathcal{P}}(\vec{\alpha}) - \vec{m}$, of course; but since $\vec{m}$ is a constant these are identical to the derivatives of $\vec{\mathcal{P}}(\vec{\alpha})$.

In forming $\frac{\partial \mathbf{M}_\Lambda(\vec{\alpha})}{\partial \alpha_i}$ only the one component matrix actually depending on α_i has to be considered; the rest can be treated as a constant with respect to α_i.

The derivatives $\frac{d\mathrm{M}_j(\alpha_k)}{d\alpha_k}$ are easily obtained. E.g., for a flexion matrix (rotation around the z-axis)

$$\mathrm{M}_{flex}(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.8}$$

and

$$\frac{d\mathrm{M}_{flex}(\alpha)}{d\alpha} = \begin{pmatrix} sin(\alpha) & -\cos(\alpha) & 0 & 0 \\ \cos(\alpha) & sin(\alpha) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.9}$$

To get $\frac{\partial \mathbf{M}_\Lambda(\vec{\alpha})}{\partial \alpha_k} = \prod \mathrm{M}_j(\vec{\alpha})$, simply replace the M depending on α_k, by its derivative.

Using this kind of decomposition, computation of $\mathbf{M}_\Lambda(\vec{\alpha})$ and its derivatives $\frac{\partial \mathbf{M}_\Lambda(\vec{\alpha})}{\partial \alpha_i}$ can be done in parallel. Let M_k be the component matrix in $\mathbf{M}_\Lambda$ that depends on α_k, i.e., $\mathrm{M}_k(\vec{\alpha}) = \mathrm{M}_k(\alpha_k)$. To get a transformation matrix $\mathbf{M}_\Lambda$, and its derivative $\frac{\partial \mathbf{M}_\Lambda}{\partial \alpha_k}$, for each parameter α_k $(k \in \Lambda)$

1. compute

$$\mathbf{M}_{<k}(\vec{\alpha}) = \prod_{\text{j before k in } \mathbf{M}_\Lambda} \mathrm{M}_j(\vec{\alpha}) \tag{4.10}$$

   and the derivatives $\frac{\partial \mathbf{M}_{<k}(\vec{\alpha})}{\partial \alpha_i}$ for all α_i that occur in $\mathbf{M}_{<k}$.

2. compute $\mathrm{M}_k(\alpha_k)$ and $\frac{d\mathrm{M}_k(\alpha_k)}{d\alpha_k}$.

3. compute

$$\mathbf{M}_{>k}(\vec{\alpha}) = \prod_{\text{j after k in } \mathbf{M}_\Lambda} \mathrm{M}_j(\vec{\alpha}) \tag{4.11}$$

   and the derivatives $\frac{\partial \mathbf{M}_{>k}(\vec{\alpha})}{\partial \alpha_i}$ for all α_i that occur in $\mathbf{M}_{>k}$.

4. multiply together:

$$\mathbf{M}_\Lambda(\vec{\alpha}) = \mathbf{M}_{<k}(\vec{\alpha})\mathbf{M}_k(\alpha_k)\mathbf{M}_{>k}(\vec{\alpha}) \tag{4.12}$$

and

$$\frac{\partial \mathbf{M}_\Lambda(\vec{\alpha})}{\partial \alpha_i} = \begin{cases} \frac{\partial \mathbf{M}_{<k}(\vec{\alpha})}{\partial \alpha_i}\mathbf{M}_k(\alpha_k)\mathbf{M}_{>k}(\vec{\alpha}) & i \text{ before } k \\ \mathbf{M}_{<k}(\vec{\alpha})\frac{d\mathbf{M}_k(\alpha_k)}{d\alpha_k}\mathbf{M}_{>k}(\vec{\alpha}) & i = k \\ \mathbf{M}_{<k}(\vec{\alpha})\mathbf{M}_k(\alpha_k)\frac{\partial \mathbf{M}_{>k}(\vec{\alpha})}{\partial \alpha_i} & i \text{ after } k \end{cases} \tag{4.13}$$

As an illustration, figure 4.5 shows the calculation tree for $\mathbf{M}_{MCP2-PIP2}$, i.e., for the MCP2–PIP2 segment.

4.5 Adding some ad hoc physiology

In its present form the hand model has a serious flaw: it does not distinguish between reasonable and absurd hand shapes. This provides the Quasi-Newton algorithm with many tempting false minima. What constitutes a reasonable hand shape is determined by the hand's physiology, the functionality of the joints and the interaction of muscles and tendons that impose restrictions on possible hand motion and the flexibility of the various joints. A set of constraint functions that model the most significant aspects of this physiology make the fitting process more stable and results more reliable.

These constraint functions all work by penalizing hand configurations that involve undesirable angle values with large residual errors. They must, of course, correspond to the requirements listed in chapter 3: they must be smooth and should be at least locally linear. Constraint functions are weighted according to their relative importance in comparison to each other and the residuals generated by the marker correspondence pairs.

4.5.1 Range limits

One of the most obvious ways to introduce physiological aspects into the hand model is to impose range limits on the joint angles. This can be done by adding a series of

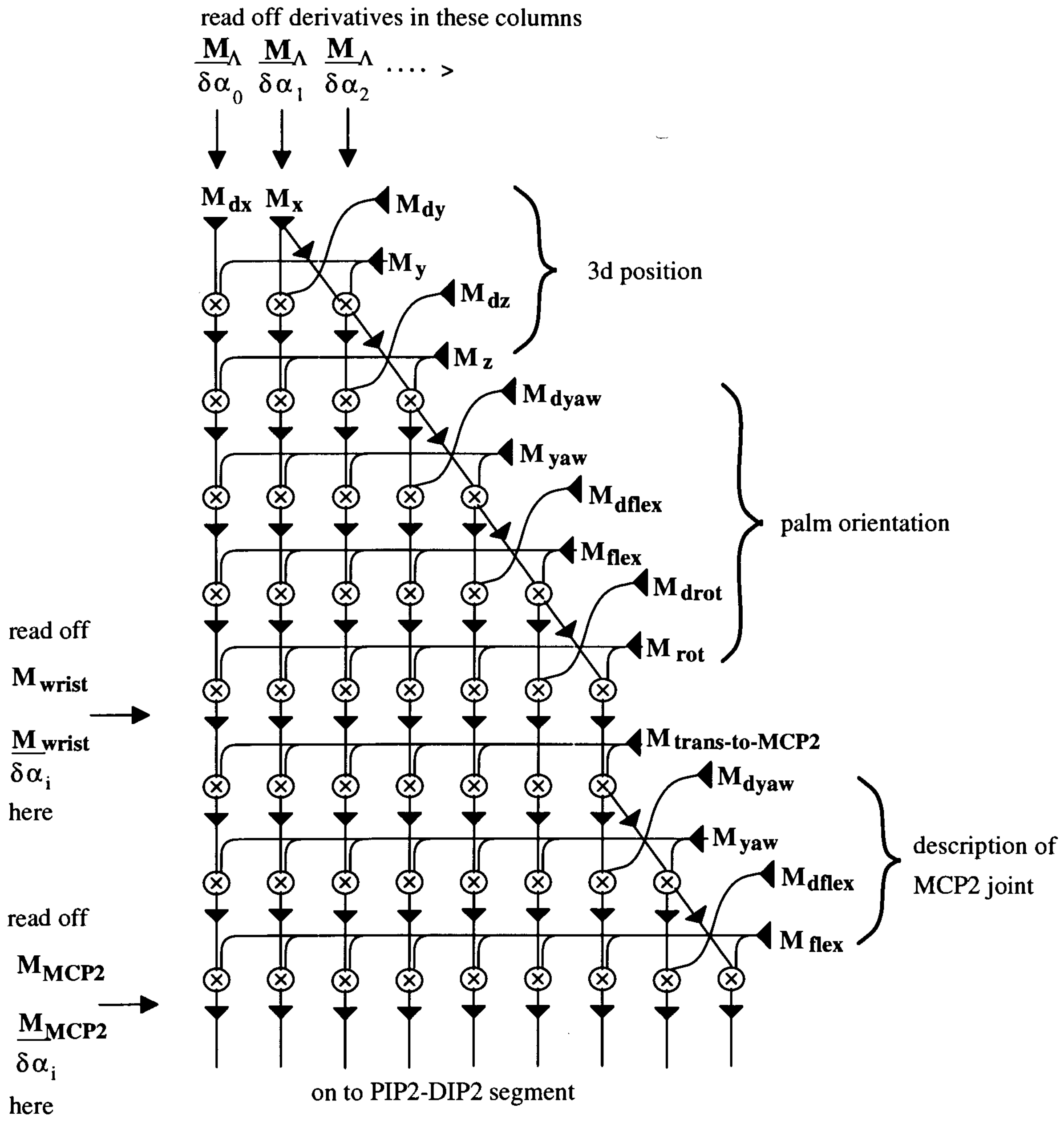
read off derivatives in these columns
$\frac{M_\Lambda}{\delta\alpha_0}$ $\frac{M_\Lambda}{\delta\alpha_1}$ $\frac{M_\Lambda}{\delta\alpha_2}$ .... >
M_{dx}
M_x
M_{dy}
M_y
M_{dz}
M_z
3d position
M_{dyaw}
M_{yaw}
M_{dflex}
M_{flex}
M_{drot}
M_{rot}
palm orientation
$M_{trans\text{-}to\text{-}MCP2}$
M_{dyaw}
M_{yaw}
M_{dflex}
M_{flex}
description of MCP2 joint
read off M_{wrist} $\frac{M_{wrist}}{\delta\alpha_i}$ here
read off M_{MCP2} $\frac{M_{MCP2}}{\delta\alpha_i}$ here
on to PIP2-DIP2 segment

Figure 4.5: Calculation tree for the transformation matrix that describes the MCP2–PIP2 segment: the calculation proceeds from the root (translation in x: M_x and $M_{dx} = \frac{dM_0(\alpha_0)}{\alpha_0}$) and spreads out towards the fingers as more parameters have to be considered. Partial derivatives of the α_i are handed down through the columns. The transformation matrices and their partial derivatives can be read off in their respective rows, with the actual transformation matrix in the last element of the row. Thus, the first element in row 6 is $\frac{\mathbf{M}_{wrist}}{\partial\alpha_0}$, the second is $\frac{\mathbf{M}_{wrist}}{\partial\alpha_1}$ and so on, and the last element of row 6 is $\mathbf{M}_{wrist}$ itself.

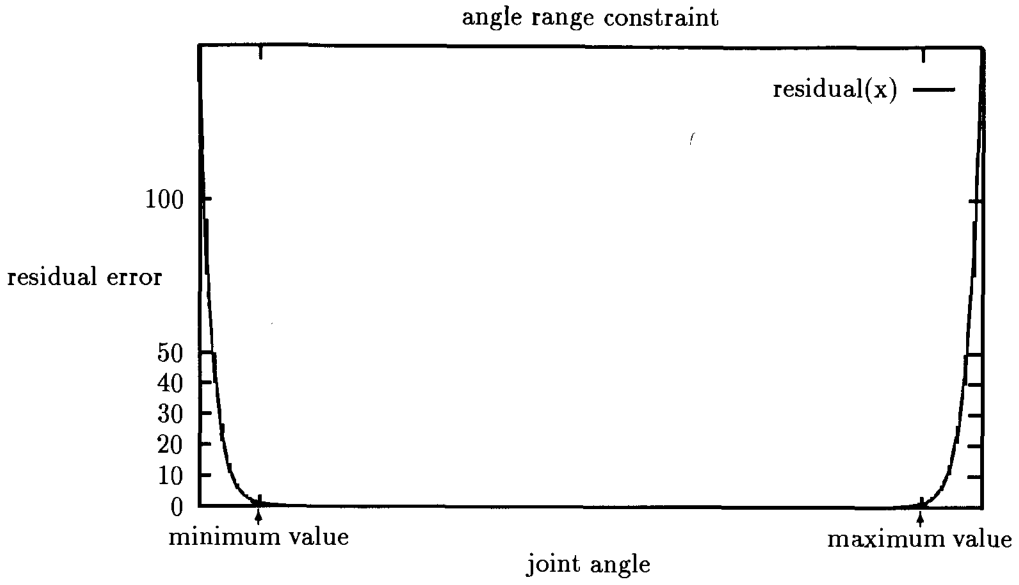

Figure 4.6: Residual error generated by the constraint function that enforces range limits on joint angle values. The function increases the residual error for any hand shape that involves out-of-bounds angle values and thus prevents that such a shape from being accepted as a minimum.

pot-shaped functions

$$f(\alpha_i) = [2\frac{\alpha_i - min_{\alpha_i}}{max_{\alpha_i} - min_{\alpha_i}} - \frac{1}{2}]^{N[4]} \tag{4.14}$$

to the hand model, one for each parameter (see figure 4.6). The bottom area of the pot corresponds to the range of admissible angle values. The residual error generated by the pot function increases steeply for out-of-bound angle values. Hand shapes that involve out-of-bound angle values therefore produce large residual errors and are not attractive for the fitting routine.

[4]N determines the steepness of the pot's wall. If N is too small the function imposes an undesirable preference for angle values in the middle of the permissible range. A large N, on the other hand, creates very sharp edges at the sides of the pot and thus generates strong nonlinearities (which have a negative effect on the performance of the Quasi-Newton algorithm (see chapter 3)). $N = 15$ provides a good compromise.

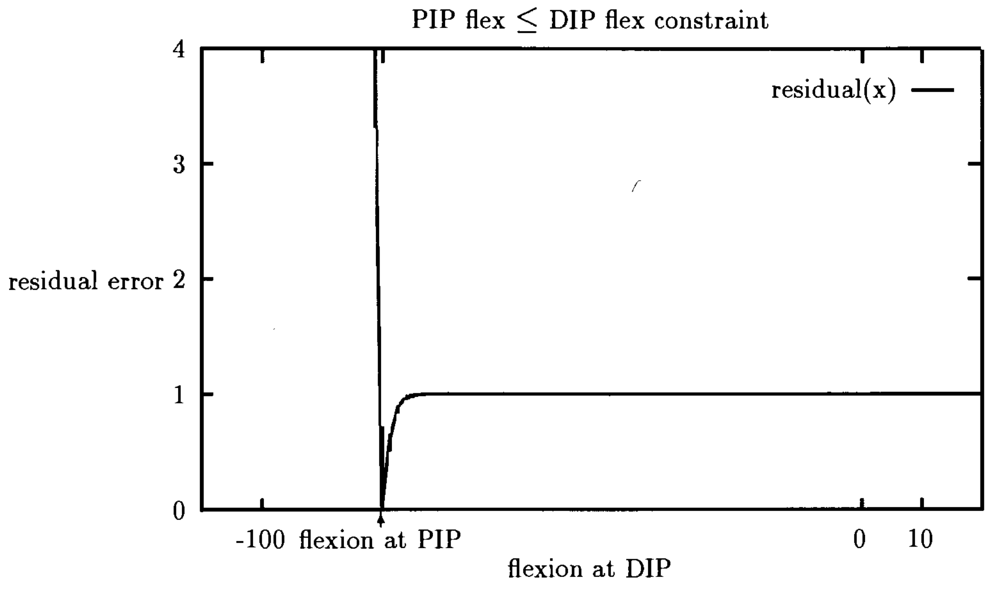

Figure 4.7: The last two segments of each finger do not move independently; to flex the finger stronger at the DIP than at the PIP is almost impossible. The constraint function described in equation 4.15 (plotted here as a function of DIP flexion for a fixed DIP flexion) reflects this fact.

4.5.2 Interdependences

Closer observation of hand and finger motion suggests further constraints: fingers and finger segments do not move independently of each other. This is most conspicuous for the last two segments of each finger. The flexion at PIP and DIP of the same finger tends to be the same and it is virtually impossible to bend the last segment without bending the middle one. Thus we may design a constraint function that favours hand shapes where PIP=DIP and punishes any attempt to flex the DIP more than the DIP:

$$f(\alpha_{PIP_{flex}}, \alpha_{DIP_{flex}}) = [e^{\alpha_{PIP_{flex}} - \alpha_{DIP_{flex}}} - 1]^2 \tag{4.15}$$

This constraint does not add any additional residual error for hand configurations where PIP=DIP and only adds a very moderate punishment for configurations where the flexion at the PIP is stronger than at the DIP. It increases rapidly, however, for flex at DIP < flex at PIP (flexion of the fingers towards the palm results in *negative* flex angles) (see figure 4.7).

Another interdependence concerns the flexion at neighbouring MCPs. The effect is not as marked as in the case of PIP and DIP, but a finger bending at the MCP does tend to drag its neighbours along. A 'MCP≈MCP' rule should not be used as a hard constraint, but it is useful as a weak, probabilistic rule: "if marker locations do not prompt otherwise fingers tend to move together." A simple way of implementing this is to add a constraint function

$$f(\alpha_{MCPi_{flex}}, \alpha_{MCPj_{flex}}) = [\alpha_{MCPi_{flex}} - \alpha_{MCPj_{flex}}]^2, \tag{4.16}$$

but to give it a low enough weight that it does not interfere with the fitting process unless the markers do not provide sufficient constraints.

There are other useful interdependences that are just as weak as the MCP–MCP relation, but can be utilized as stabilizing, probabilistic rules. Whether any of them should actually be included and how much weight they should get depends on the difficulty of the input sequence, on factors such as frame density, reliability of marker detection, and on how many markers are available. For easy-to-track, model-generated test sequences these additional constraints actually increase the overall error in recovered pose because they introduce artifacts when the shape would otherwise be recovered perfectly. However, for real-image sequences with, among other things, very imprecise marker locations these constraints have a stabilizing effect.

- *fingers do not cross* rule:
  this can be implemented by adding a constraint function similar to the PIP≤DIP constraint to keep the yaw values of adjacent MCPs in line. Since there are signs in ASL where fingers do cross we have another lenient condition. The rule can be very helpful, however, since the Quasi-Newton algorithm tends to make heavy use of the degrees of freedom provided through the yaw at the MCPs to adapt to the marker positions in the image when it follows a false minimum.[5]

- $MCP_{flex} \approx PIP_{flex}$ rule:
  just as the MCP≈MCP rule, this is a heuristic rule that does not reflect any real physiological constraint. It is perfectly possible to adopt a hand shape

[5]The resulting shape is reminiscent of a dead spider, crushed legs sticking out at impossible angles.

that contradicts this rule (e.g., the *E*-hand in ASL). The rule rather reflects a tendency of hand motion: when the hand is closed into a fist this is usually not done by flexing first the DIP, then the PIP, and then the MCP (or vice versa), but rather by flexing all three joints at the same time.

- $PIP_i \approx PIP_j$ rule:
  another stabilizing rule that does not correspond to an actual physiological constraint.

4.6 Motion constraints

So far, everything described in this chapter was intended to capture as much knowledge as possible about the physiology of the human hand and put it into a mathematical form for the Quasi-Newton algorithm. It is also helpful, however, to consider the motion-over-time aspect of the hand tracking problem. Knowledge of position and shape of the hand at time (or frame) t, together with the knowledge that the hand may only move with a certain maximum speed drastically reduces the search space for frame $t+1$. And we can do even better. As mentioned in the introductory chapter (see section 1.1.2), hand motion is controlled by a complicated interaction of muscles and tendons, and it is impossible to write down equations that fully describe the dynamics of hand motion in terms of only a few parameters. Still, Newton's first law guarantees us at least short-range motion continuity for any massive object. This is reflected in short-range continuity in joint angle motion and can be described, or at least approximated, in terms of speed, and maybe to second order in terms of acceleration.

An estimate for the current angular speed is easily computed from the angle values of the preceding frames. We can use such estimates to obtain a prediction for the angle values in the current frame:

$$\alpha(t+1) = \alpha(t) + speed_\alpha(t)\Delta t \quad \{+\frac{1}{2}acceleration_\alpha(t)(\Delta t)^2\}. \tag{4.17}$$

(Δt stands for the time elapsed between frame t and frame $t+1$. Given a constant frame rate it is easiest to use a time unit of 'time-to-next-frame', rather than, say, seconds to make the Δts cancel out.)

Such a prediction of the new hand shape and position provides a better initial point for the fitting routine than just starting with the current shape. It also provides a good clue as to where to look for the various markers in the image and thus can reduce search time and resolve at least part of the ambiguity arising when there are multiple marker candidates. But most importantly, favouring a hand shape and position for the current image that is close to the predicted one effectively means enforcing a motion-continuity constraint. Formulated for the fitting algorithm, this constraint translates into another residual

$$[\alpha_{predicted}(t) - \alpha_{current}(t)]^2 \tag{4.18}$$

for each joint angle α.

The motion-continuity constraint also serves another, although related purpose: it provides default values for joint angles. This is crucial whenever there are not enough markers visible to constrain the optimization problem properly (see section 3.2). Thus, there is a guarantee that the Quasi-Newton algorithm returns a predictable and plausible solution even for occluded fingers or finger segments.

4.6.1 Estimating speed (and acceleration)

The interpolated speed for the time interval $t-1$ to t is

$$speed_\alpha(t) = \alpha(t) - \alpha(t-1), \tag{4.19}$$

and the acceleration is given as

$$acceleration_\alpha(t) = (\alpha(t)-\alpha(t-1))-(\alpha(t-1)-\alpha(t-2)) = (speed_\alpha(t)-speed_\alpha(t-1)). \tag{4.20}$$

Hence, it takes at least one previous frame to estimate speed and at least two to estimate acceleration.

Substituting into equation 4.17 yields

$$\alpha(t+1) = \alpha(t) + \frac{3}{2}speed_\alpha(t) - \frac{1}{2}speed_\alpha(t-1) \tag{4.21}$$

or

$$\alpha(t+1) = \frac{5}{2}\alpha(t) - 2\alpha(t-1) + \frac{1}{2}\alpha(t-2). \tag{4.22}$$

Unfortunately, this kind of estimate is very sensitive to noise. It registers and amplifies small tracking errors and it would throw the system completely off track after a major tracking error. The standard solution in such a case—taking the average over a number of frames—is to be used with caution here: with hand and finger motion the emphasis is rather on *short* term continuity, especially when motion may become very fast or the frame density is relatively low.

Since the test sequences in chapter 6 are not derived from actual video sequences, but made up of a pseudo-motion sequence of consecutive stills, the system had to deal with low effective frame density, and there was little room for extended averaging. I used a weighted[6] average of the last two speed estimates, without any attempt to include acceleration. If one can rely on higher frame density it would probably be a good idea to use more sophisticated means of prediction such as Kalman filtering (see [5]) to get a better estimate and maybe even adjust the recovered hand shape on the fly to reduce the noise in the output.

4.6.2 Digression: the 2D alternative

The motion continuity constraint is employed in most visual tracking systems in one form or another. For tracking feature points in two dimensions it takes the form of a 'smoothness-of-trajectory' constraint. This is justified by the observation that the projection of a smooth 3D trajectory is again a smooth 2D trajectory. Would there be any advantage in using one of the 2D feature-tracking algorithms described in the literature to find marker correspondences (see, e.g., [41, 19]) in addition to, or maybe instead of, joint angle prediction?

Obviously, a 2D feature-tracking algorithm would not have the stabilizing properties of joint angle prediction; motion continuity as a constraint would only be visible to the marker search and identification algorithm, but not to the fitting routine. Also, a prediction algorithm based on joint angles implicitly takes into account that markers

[6]Weights are 0.8 for $speed_\alpha(t)$ and 0.2 for $speed_\alpha(t-1)$.

can only move in a coordinated way as points on one object. A 2D tracking algorithm cannot use this kind of knowledge to correct its results.

Another problem is occlusion of markers and thus interrupted trajectories. Occlusion can be handled reasonably well in 2D as long as it does not occur too often or too long (see, e.g., [20]), but cases like the disappearance of a whole finger with all its markers for several frames would create serious problems.

On the other hand, 2D-trajectory-based methods have the advantage that they do not depend on the correctness of shape recovery in previous frames. The link between shape recovery and joint angle prediction is especially problematic when the system gets stuck in a false minimum, in which case predicted marker locations are not much use any more and on top of that the prediction enforcement tries to hold the system in the false minimum. 2D tracking of image markers could help find the correct correspondences in such a case. 2D tracking of markers could also be used as a go-between in a slightly modified system that does not do model fitting for every single frame.

Chapter 5

The Correspondence Problem Revisited

We have seen before (see section 1.3.4) that it is not possible to eliminate ambiguities in image-marker correspondence completely. When all the markers on a finger are of the same colour the correspondence problem has to be solved for each frame and each finger, but even when the markers are uniquely encoded in the 3-ring scheme one has to account for false markers (spots in the image that are identified as markers because they resemble markers, but are actually background noise, or misinterpretation of ring colours under bad lighting condition) and errors in marker identification (correct identification of the main marker-ring and hence the finger it belongs to, but incorrect joint assignment, see figure 2.3). The result of either of these errors are two (or more) markers with the same label—one of which is hopefully the correct one—or a marker with the wrong label, posing for the (occluded) rightful owner of that label.

In the latter case—and in general, when the correct marker is not among the candidates found for the particular joint—there is little one can do, unless the impostor marker(s) are actually far enough away from where one would expect the marker in the image to exclude them from consideration. This maximum-distance constraint is a dangerous measure, however. After all, there is a slim chance that it was actually the marker in the frame before that was not assigned correctly and/or the system has lost track of the hand shape. In this case the predicted position of a marker may not even be near the actual position of this marker in the image, and rejecting the correct marker as too far away will effectively prevent the system from getting back on track. Thus, the main protection against rogue markers is to be conservative in selecting candidate markers in the image: better reject a few correctly identified markers because they are too small or do not get unambiguous votes than allow false candidates to creep in.

If, on the other hand, the true marker is actually among the set of candidates it is possible to clear up the situation by choosing the candidate that fits best into the general picture. This should (hopefully) be the correct marker.

5.1 Measure ♯1: using predicted position as a criterion

The first, and cheapest, measure for solving the correspondence problem is to make use of the predicted hand shape for the current frame. Since this prediction should be very near the actual hand shape the 2D locations of the joints should be equally near to the actual marker locations when projected onto the image.

The correct assignment for a set of candidate image-markers for one or more joints should be the one with the minimal net distance between candidate marker-joint pairs, i.e.,

$$\min\{\sum(marker\,candidate_i - projected\,joint_i)^2\}. \tag{5.1}$$

For the unique encoding scheme this means comparing a few distances and choosing the candidate nearest to the projected location. In the simple one-colour-per-finger encoding scheme with m markers per finger there are $m!$ possible assignments for each finger, and hence just as many net distances to compare. For a glove with 4 markers per finger this amounts to 24 possible assignments per finger, and maybe a few hundred if more than 4 candidates have been found in the image. Such a check is still cheap since exhaustive search does not cost much on this level.

Unfortunately, the shortest-net-distance criterion only works well for hand shapes where the markers are far apart (more or less extended fingers) or when the prediction is very accurate. In these cases there is a fairly large difference between the net distance for the correct assignment and the net distance for the next-best assignment. When we have an open shape and the prediction is not very accurate this tends to increase the overall error in net distance, but the correct assignment will still look much better than the next best. For fist shapes, however, the differences in net distance are never as marked, and a slight over- or under-estimation of angular speed in MCP or PIP flexion can easily lead to confusion and wrong marker assignments.

It is not easy to make the prediction more accurate: consider the typical case of having to estimate the speed with which fingers that may just have started to curl will be closing into a fist. The whole movement may take as much as 20–30 or no more than 3–4 frames. Especially for very fast motion it is impossible to get more than a crude prediction under these circumstances. But even if we had a good enough prediction for most cases, there is still the possibility that the shape in the last frame was not correct and the system has to be dragged out of a wrong minimum. If one wants to rely solely on prediction to solve the correspondence problem it would therefore be advisable to also consider 2D prediction based on 2D trajectory smoothness (see section 4.6.2) that would not be affected by tracking difficulties. Then there would be the question which prediction to believe in the particular case, of course, and especially in cases with partial prolonged occlusion neither prediction is likely to be very reliable.

It should be possible in most cases to refute a wrong assignment that happens to have the shortest net distance on grounds that it would enforce an impossible finger shape, though. Thus, it would help to employ the knowledge about the hand's physiology that is embedded in the hand model for difficult marker assignments—both for the uniquely encoded markers and the one-colour-per-finger case.

5.2 Measure ♯2: asking the hand model

The knowledge about the hand that is contained in the hand model is only accessible through the fitting routine. Hence, if we want to utilize this knowledge to solve the correspondence problem we have to do this via the fitting routine, one tentative fit for each possible assignment. This amounts to asking the hand model how it 'likes' the assignment in question, i.e., how well it can adapt its shape to it. The answer comes back in the form of the residual error of the fit, a composite of residual marker–joint distance and strain in the model (the residual error generated by the constraint functions described in chapter 4). Since the motion-continuity-constraint pulls the model towards the predicted shape a test score based on residual error also subsumes the net distance score from the last section.

It is not surprising that the success rate for the residual-error measure is better than for the net distance score. However, there is a price: a whole model-fitting cycle for each assignment to be tested. This may be acceptable in the case of the uniquely-encoded 3-ring markers as a last measure to resolve the occasional difficult ambiguity, but it is far too costly to use it for a regular exhaustive search on all 24×5 possible assignments in the case of one-colour-per-finger markers—let alone try all the $(5 \times 4)!$ possible net combinations. Hence the reduction to three, instead of four markers per finger that is mentioned in chapter 2. This leaves $3! = 6$ possible assingments per finger, $6 \times 5 = 30$ for the whole frame; which is still way too much.

To go further in that direction and reduce the number of markers per finger to 2 unfortunately endangers the stability of the model and the hand shape recovery. Experiments with model-generated test data quickly show that even under ideal conditions the system does very poorly when it is only given two markers on each finger.[1]

There is another option though for cutting down on complexity for the one-colour-per-finger glove: since the possible assignments we have to test and compare come finger by finger and the shape of other fingers has at most a mild influence on the shape of the finger currently looked at we might just as well do the test fitting for that one finger only, fixing the joint angles of the remaining fingers at their predicted values. This reduces the number of internal parameters from 26 to 10 per fit[2] and

[1]The MCP markers cannot be left out without creating havoc. Putting the second marker on the PIP may give reasonably stable tracking results, but it does not convey essential information about the shape of the fingers from the PIP onwards. A marker only on the DIP or the TIP, on the other hand, means too much freedom—and hence instability—in the recovery of finger shape.

[2]4 for the finger plus 6 for hand position and orientation. The latter may be left out, but there is some danger that holding them fixed may not allow the model to adapt enough to the marker positions to provide useful test scores, especially if the prediction was not very accurate.

increases the speed of the fit accordingly.

Still, even such a minimalist version of the fitting process is not something that one would want to use for each possible assignment and each single frame; it is more an emergency measure for cases where the predicted-position criterion does not provide a clear decision. The system reverts to tentative fitting only when each of the possible assignments gets bad scores from the distance measure or when the best assignment in terms of distance measure is not clearly better than the next best assignment.

Chapter 6

The System in Action

6.1 Quantitative evaluation with simulated input data

In order to test various aspects of the system during development, I designed an interface for graphical visualization of hand-shape descriptors (see figure 6.1)—in other words, I implemented a crude version of a computer-animated hand.

It would not be difficult to render this hand more realistic. In its present form, though, it reflects the system's view of a human hand. This makes it easier to understand why the system has difficulties with certain gestures and to see, whether a particular problem can be corrected by revising physiological constraints or whether it is due to fundamental limitations of the hand model.[1]

The interface in figure 6.1 also has facilities that allow the user to manipulate the current hand shape interactively and to 'hide' and 'uncover' joints and finger tips at will. The user can then print out the current joint locations after they are projected through a camera model that is set up to reflect a typical setting of a real camera-input-scene[2]. Thus it is possible to generate input for the fitting routine that

[1]The latter seems to be the case only when the signer's hand is facing the camera, palm forward. Letting the hand drop backwards from this position, dropping the hand slightly forward while curling the fingers into a fist, and dropping the hand to point towards the viewer with extended fingers all generate very similar joint location patterns for the visible joints.

[2]focal length: 6.0cm,; initial distance wrist–camera: 1m. The dimensions of the hand are rough

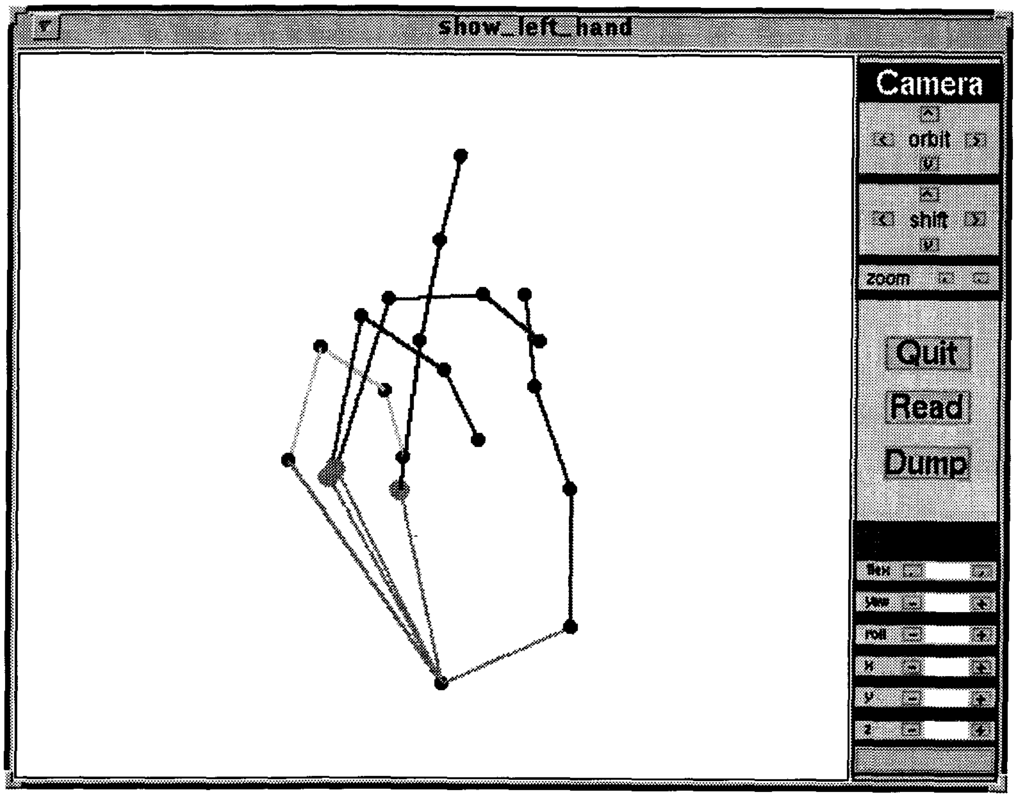

Figure 6.1: Display program to visualize hand-shape descriptors: the user can select a file to animate the stick model of the hand with a sequence of hand-shape descriptors. S/he can also manipulate the hand by changing joint angle values or position of the hand and thus generate input sequences to test the fitting routine.

would normally have to come from the marker-detection module (see figure 1.1).[3] The advantage of this procedure—apart from saving time and storage space for images—is that the original hand shape is available not only as a visual 2D impression to the human viewer but as a hand-shape descriptor that can be directly compared to the descriptor obtained by the fitting routine. This allows a quantitative evaluation of the tracker's performance that would be impossible, or at least very difficult, for input from real image sequences.

6.1.1 Tracker versus original

Figures 6.2 and 6.3 show the tracker in action on a 10 frame hand-motion sequence, ASL fingerspelling of the letter 'A' (see [7], page 13). The original sequence is shown in the first column, the tracking result in the second column. (The third column contains tracking results for a test run without using any of the physiological constraint functions. This experiment will be discussed in the next section.)

The images shown are enlarged snapshots taken from the hand animator in figure 6.1. Black dots represent 'visible' markers. The 2D location of these markers has been available to the tracker. The slightly bigger grey dots indicate occluded markers invisible to the tracker. For this and all further examples the observer's viewpoint of the hand is chosen to correspond to that of the tracker, i.e., the reader has the same view of original and tracking result as the tracker (except that the images are enlarged). This way it is easiest to compare the original and the tracking result, though a 'walk' around the result can sometimes be useful to show tracking errors that are not clearly visible from the camera perspective. The hand shape shown on the display in figure 6.1 is in fact a different view of the tracking result for the first frame of the image sequence in figure 6.21 (ASL fingerspelling of 'D' and 'E').

measures of my own hand. The length from wrist to the tip of the middle finger is assumed to be 16.5cm (for more details see figure 4.3).

[3]This simulates an ideal marker-detection system for a glove with uniquely encoded markers: the fitting routine is provided with exactly one candidate location for each joint and finger tip, unless the joint or tip is occluded in which case there are no candidates reported for that particular joint marker.

The quantitative evaluation of tracking results for the model-generated spelling-'A' sequence in figures 6.2 and 6.3 can be found in figures 6.4 to 6.10. The plots, one for each parameter, show the absolute tracking error in cm resp. degrees for each frame.

6.1.2 Do we need constraints?

It should be noted that most of the tracking error in the spelling-'A' sequence is actually due to the influence of the constraint functions. For example, the relatively high initial error for MCP, DIP, and PIP flexion in the first frame reflects a strain of the hand model against a hand shape that is close to the range limits for flexion in the interphalangeals. The tracker was run on the model-generated sequences with the same setup as for the real image sequences shown at a later point in this chapter. One could easily get better results for these model-generated sequences by lowering the relative weight of the constraint functions.

Since the 2D joint and tip positions are exact, the fitting algorithm converges to the exact position and constraint functions are only needed in this case as a guard against false minima. This is not the case when working with real image sequences, though, where the marker centers only approximately indicate joint and tip locations. There the optimal solution according to proximity of joint positions is not always correct, and constraint functions help to stabilize the result.

The rightmost column in figures 6.2 and 6.3 and the respective error plots (figures 6.11 to 6.17) show what happens when the same sequence is tracked without using any constraint functions, i.e., neither prediction nor physiological or 'pseudo-physiological' constraints. The results exhibit a typical pattern: tracking is still good most of the time—often better, in fact, than with strong constraint functions. The tracker becomes more unstable, however, more likely to drift off into a false minimum. For example, watch the thumb in frames 3 to 6 and the index finger in frames 6 to 8.[4] This instability has much more serious consequences for tracking real image sequences, though, where errors in joint locations flatten out the correct minimum and introduce

[4]Note the different scaling for the error graphs here! The maximum angle error is now 50°.

additional false minima.

6.1.3 Loosing track and finding it again

Even when it is allowed to use all it's knowledge about constraints the tracker is not infallible. Figure 6.18 and 6.19 show the first 10 frames of another gesture: the signer starts with her hand in front of the body and sweeps it outwards/upwards into the position used for fingerspelling. (This is again followed by fingerspelling 'A' (not shown), for which tracking performance is similar to the example above.) Again, the original sequence is leftmost, the middle column shows the tracking result for 'normal' tracking, and the rightmost column shows tracking without constraints.

Because of its preference for slightly bent fingers (see the discussion above), the tracker consistently misunderstands this sweep with extended fingers as a hand-closing motion. When the constraint functions are left out the tracker does not make this mistake, but comes up with a series of impossible hand shapes.

As mentioned above (see footnote on page 61) pointing gestures are difficult for the tracker, and it may not be possible with this kind of system to get reliable tracking results in such a situation. The tracker still provides useful information, however—the position and orientation of the hand—that signal that tracking results may be unreliable, at least as far as hand shape is concerned. Maybe more importantly, the system recovers without any additional measures as soon as the gesture becomes unambiguous again.[5] The abrupt change in hand shape when the system snaps back into the correct shape can be used as further indication that the tracker had been following a false interpretation and is now back on track again.

[5]Results for tracking the rest of this sequence without constraints have the usual wobbly look.

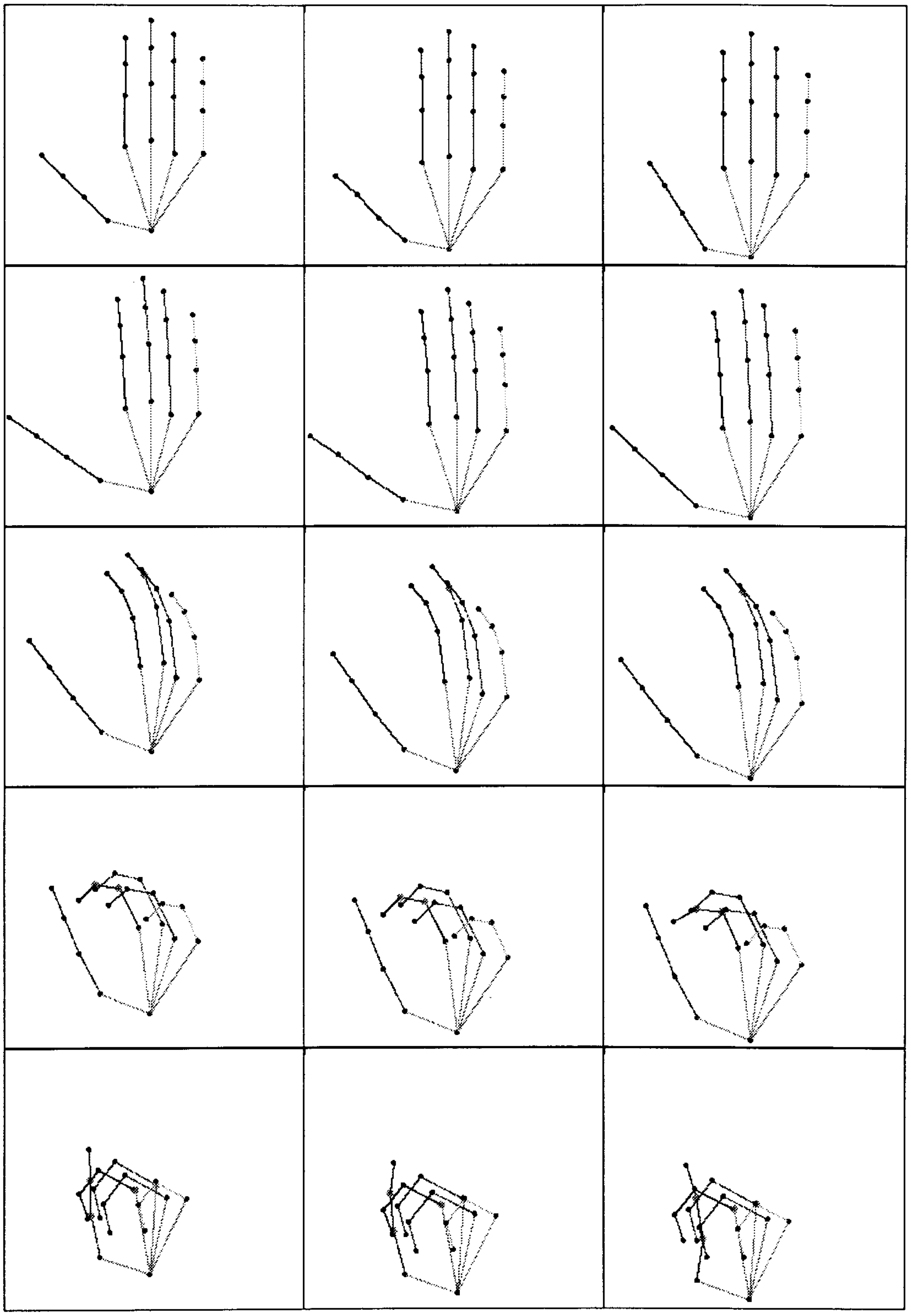

Figure 6.2: tracking 'A' from model-generated input, frame 0 to 4

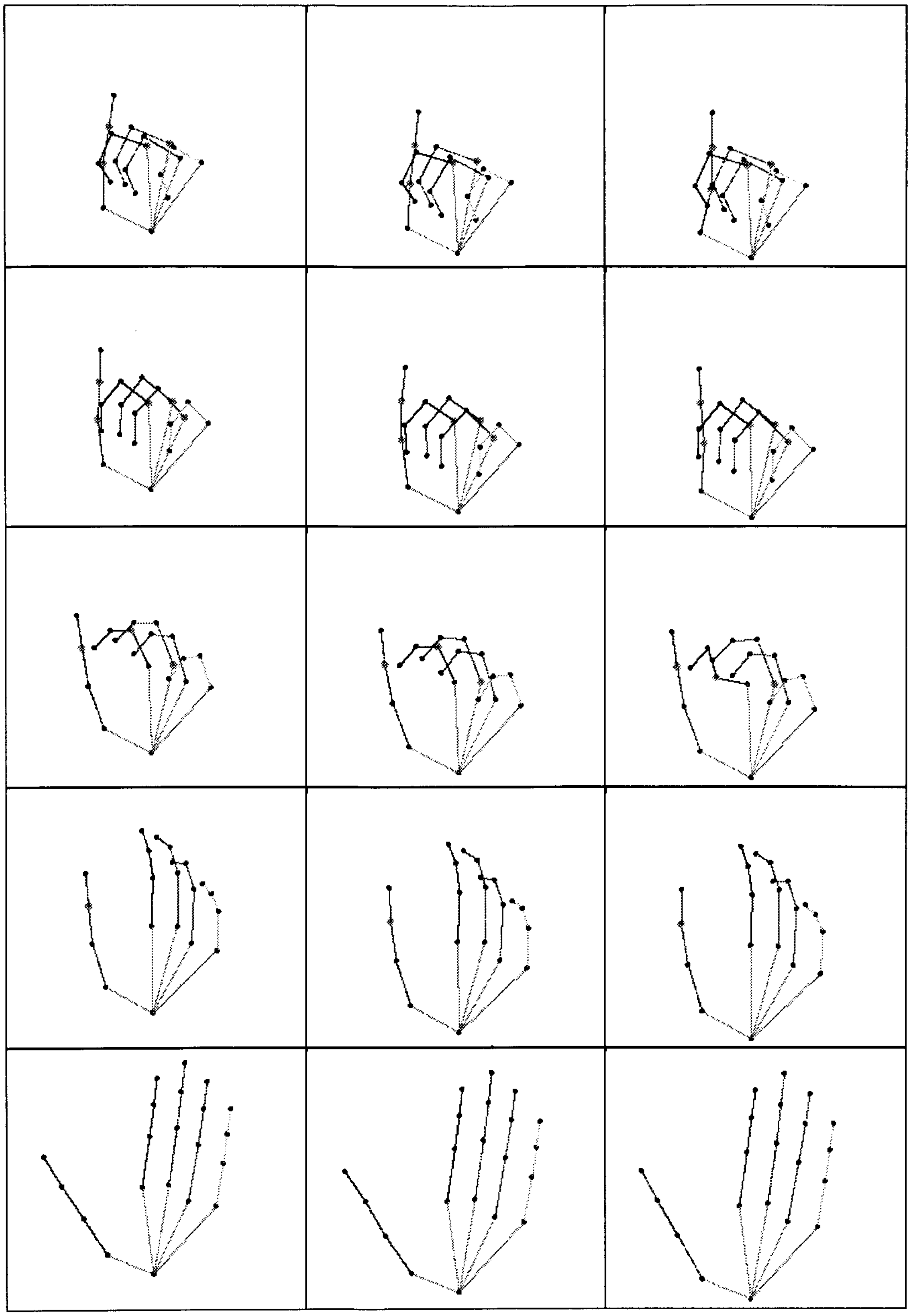

Figure 6.3: tracking 'A' from model-generated input, frame 5 to 9

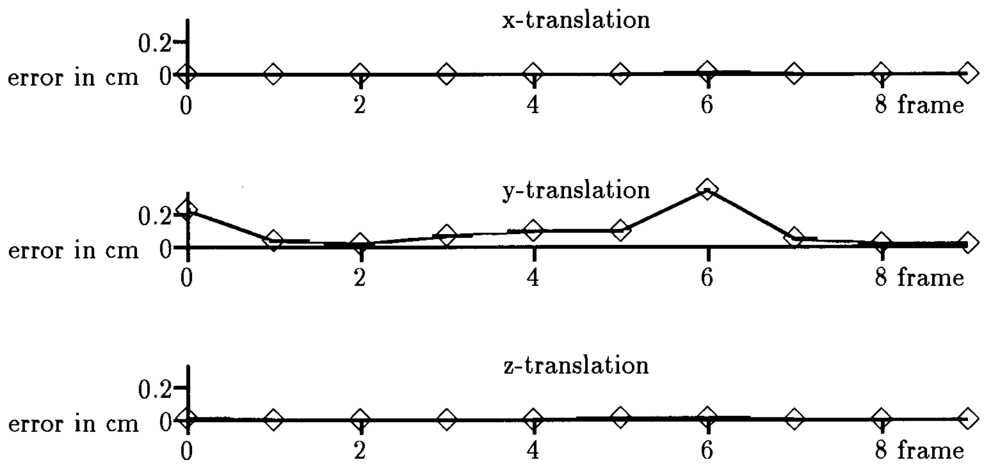

Figure 6.4: tracking 'A': errors in recovered position

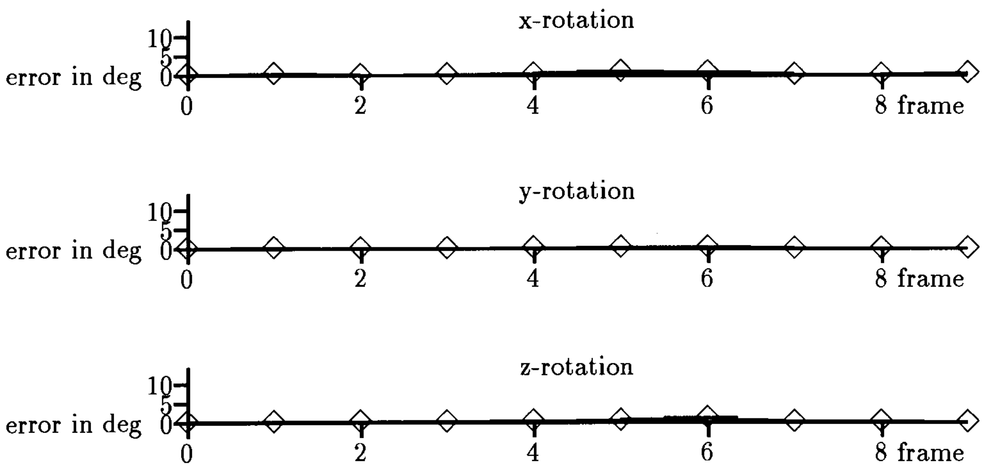

Figure 6.5: tracking 'A': errors in palm orientation

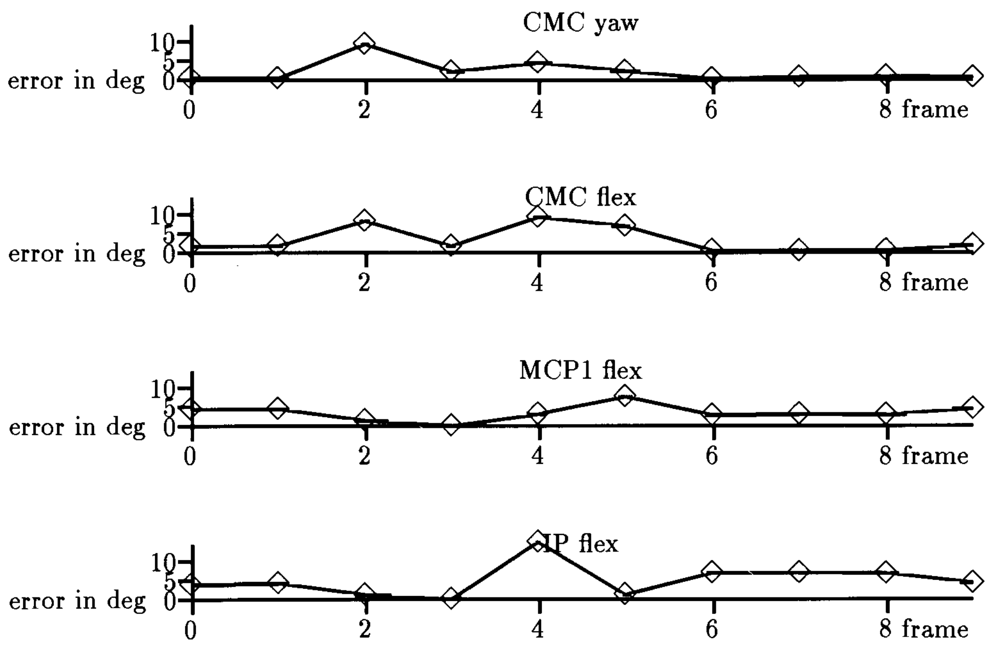

Figure 6.6: tracking ‘A’: errors in joint angles for thumb

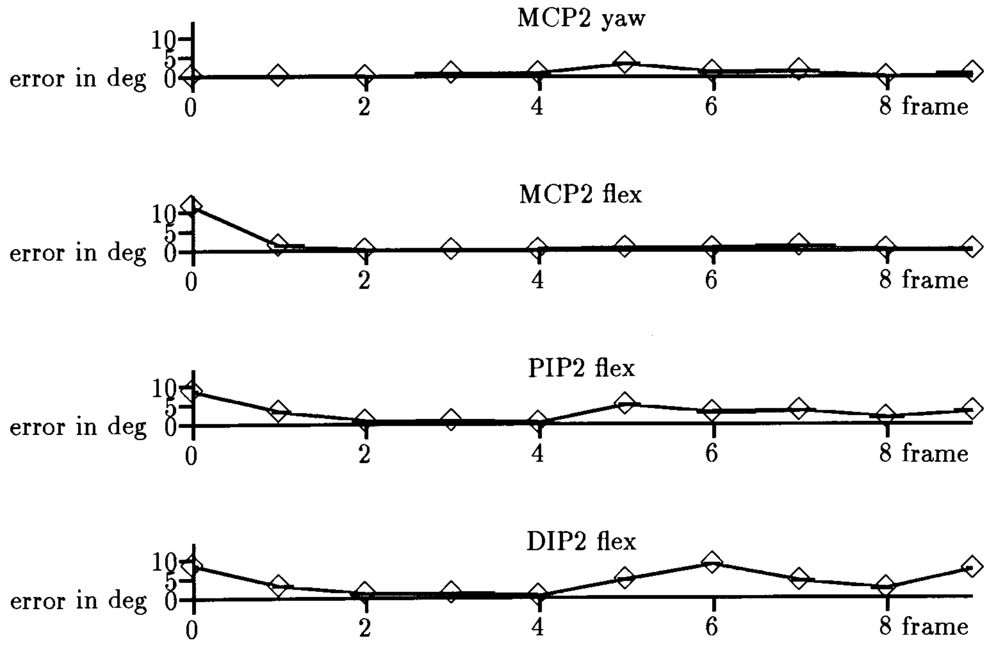

Figure 6.7: tracking ‘A’: errors in joint angles for finger 2

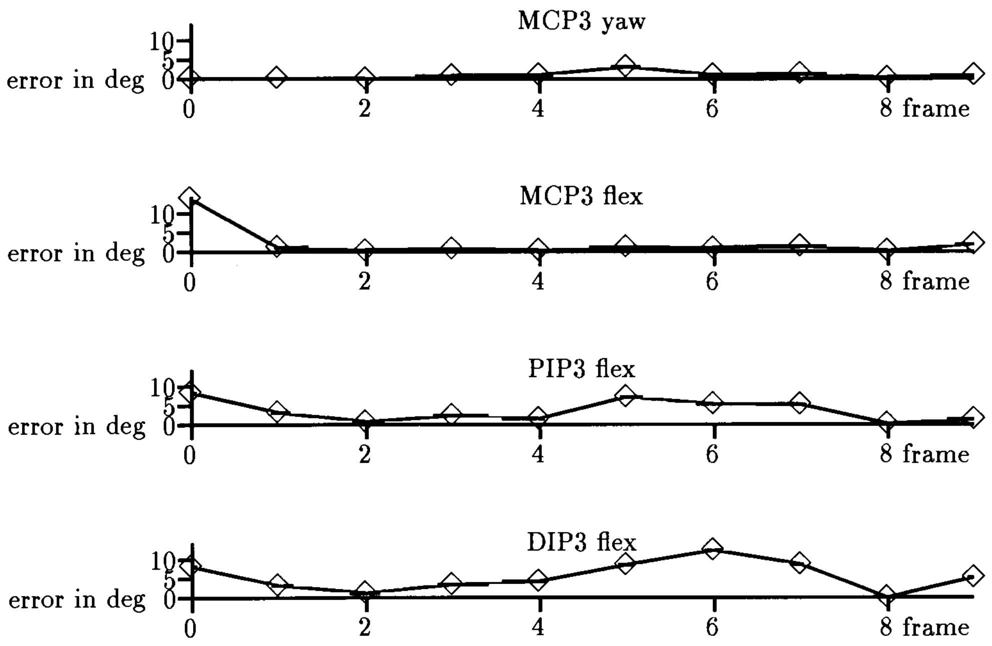

Figure 6.8: tracking 'A': errors in joint angles for finger 3

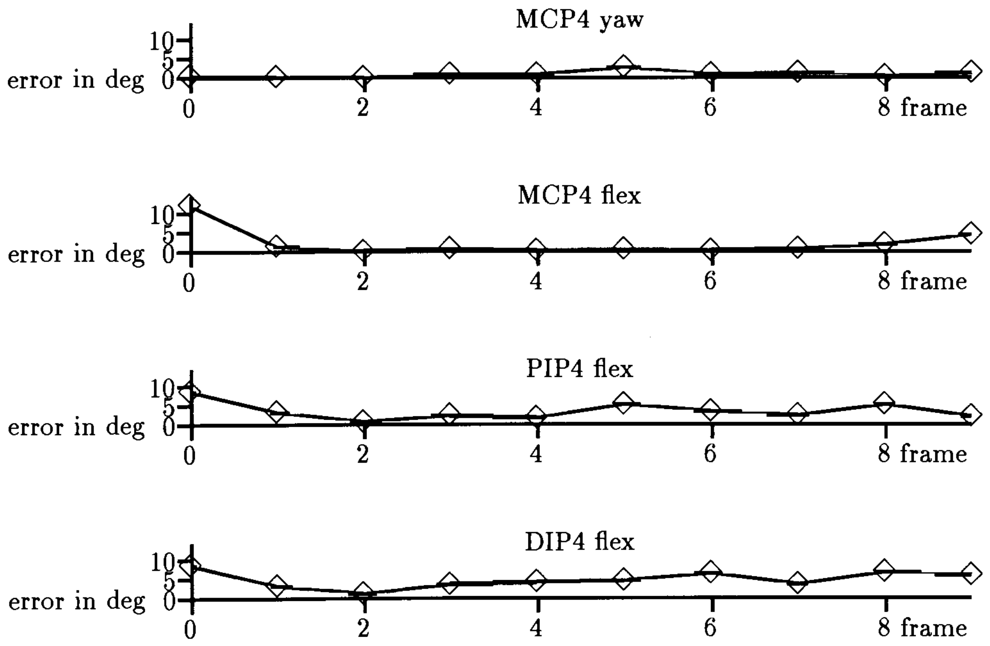

Figure 6.9: tracking 'A': errors in joint angles for finger 4

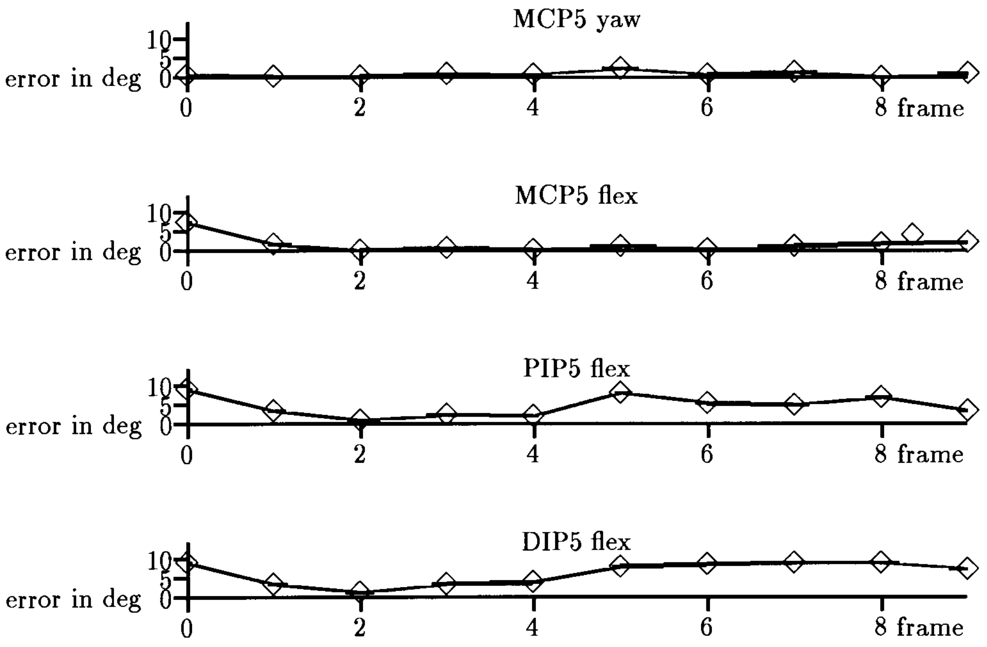

Figure 6.10: tracking 'A': errors in joint angles for finger 5

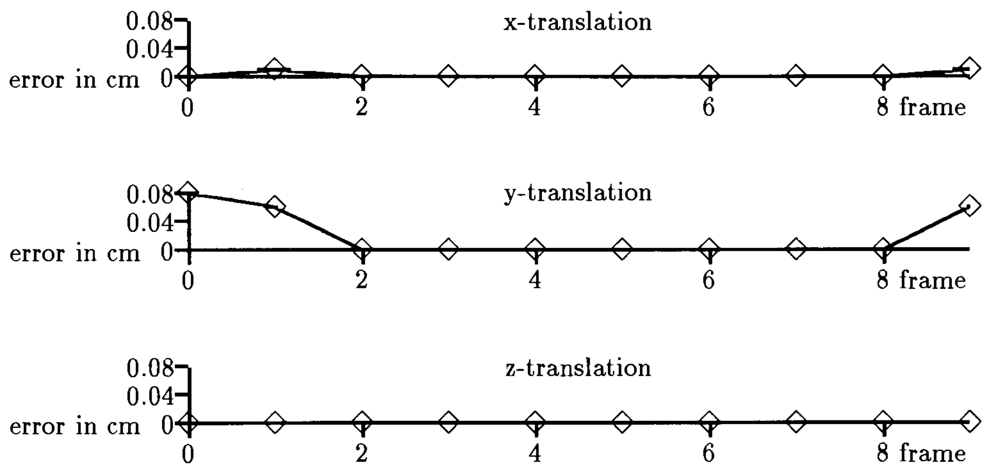

Figure 6.11: tracking 'A' without constraints: errors in recovered position

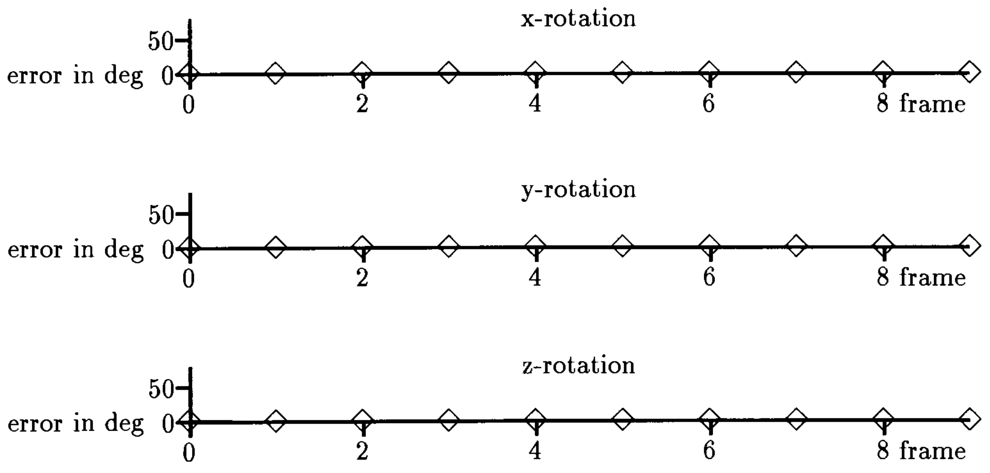

Figure 6.12: tracking 'A' without constraints: errors in palm orientation

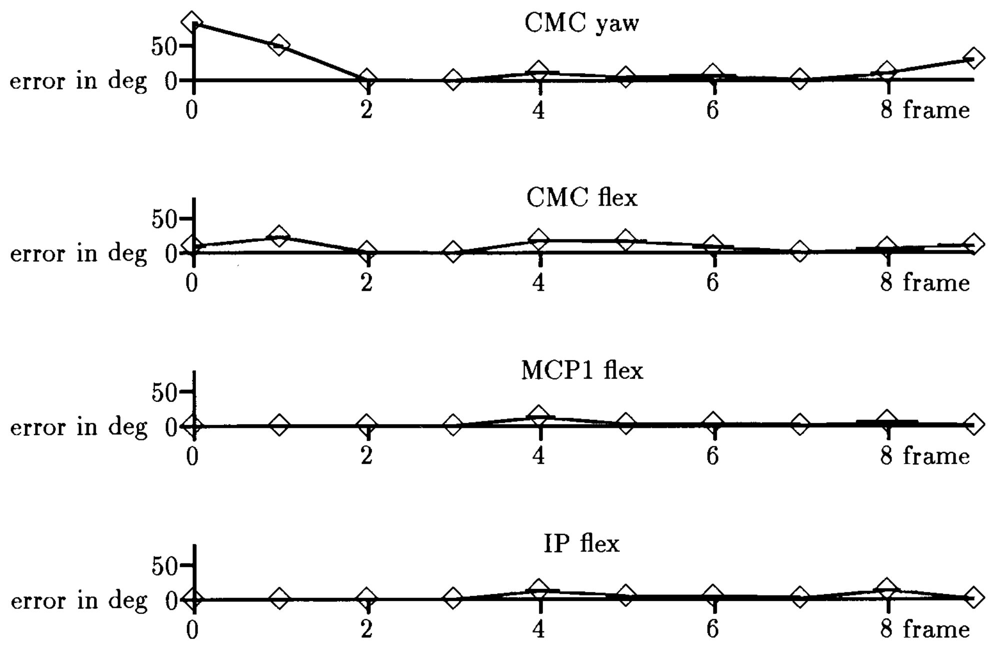

Figure 6.13: tracking 'A' without constraints: errors in joint angles for thumb

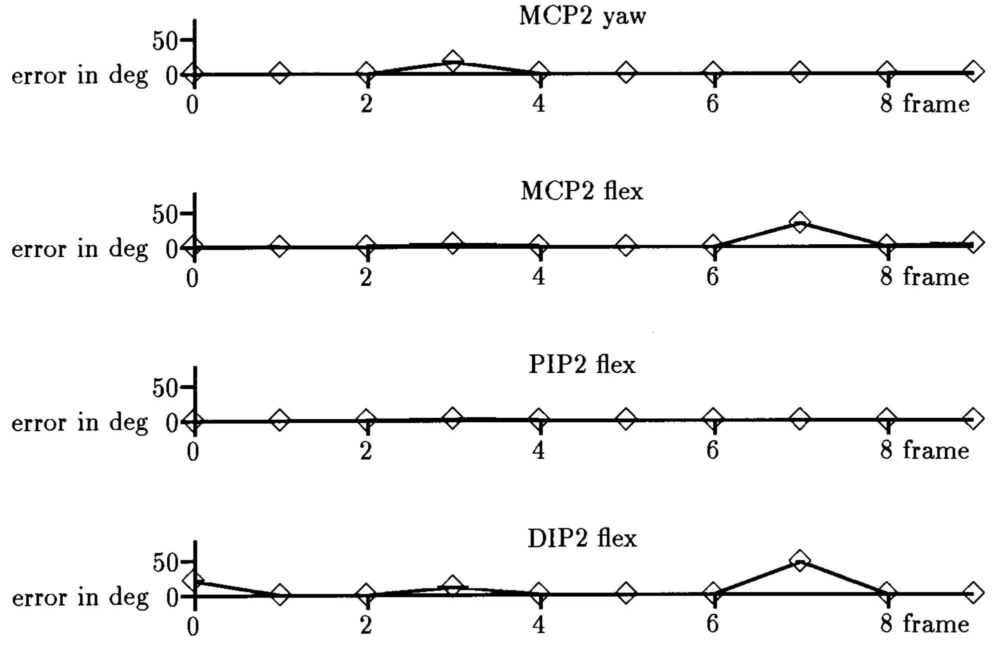

Figure 6.14: tracking 'A' without constraints: errors in joint angles for finger 2

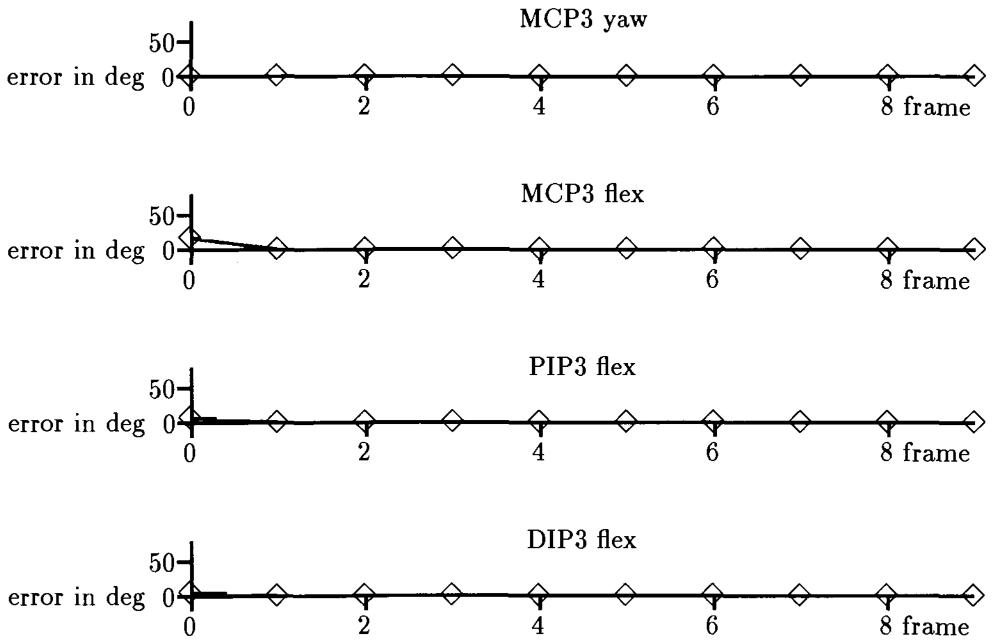

Figure 6.15: tracking 'A' without constraints: errors in joint angles for finger 3

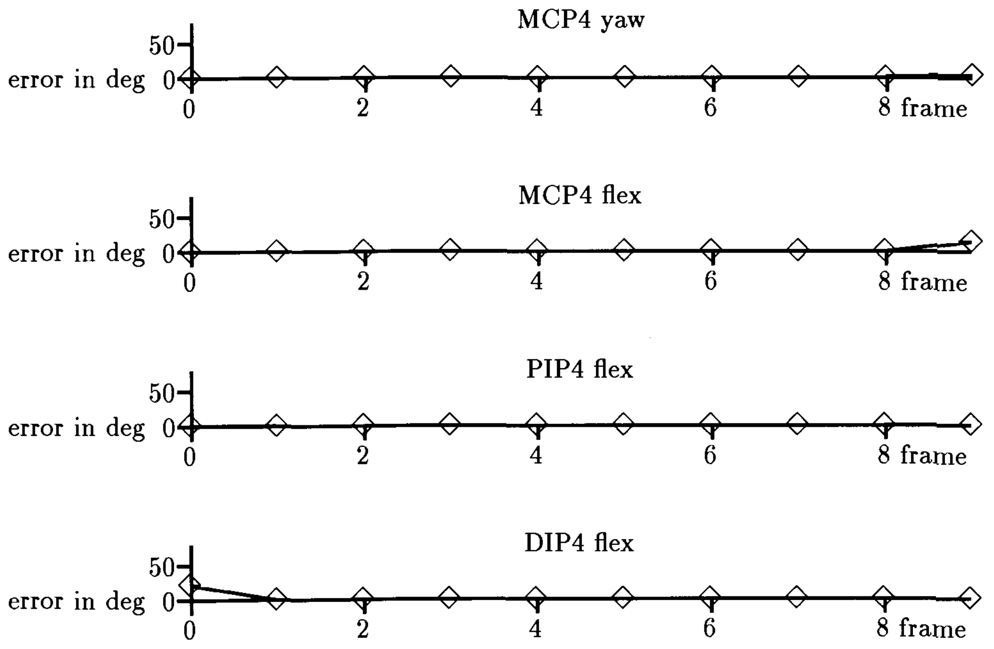

Figure 6.16: tracking 'A' without constraints: errors in joint angles for finger 4

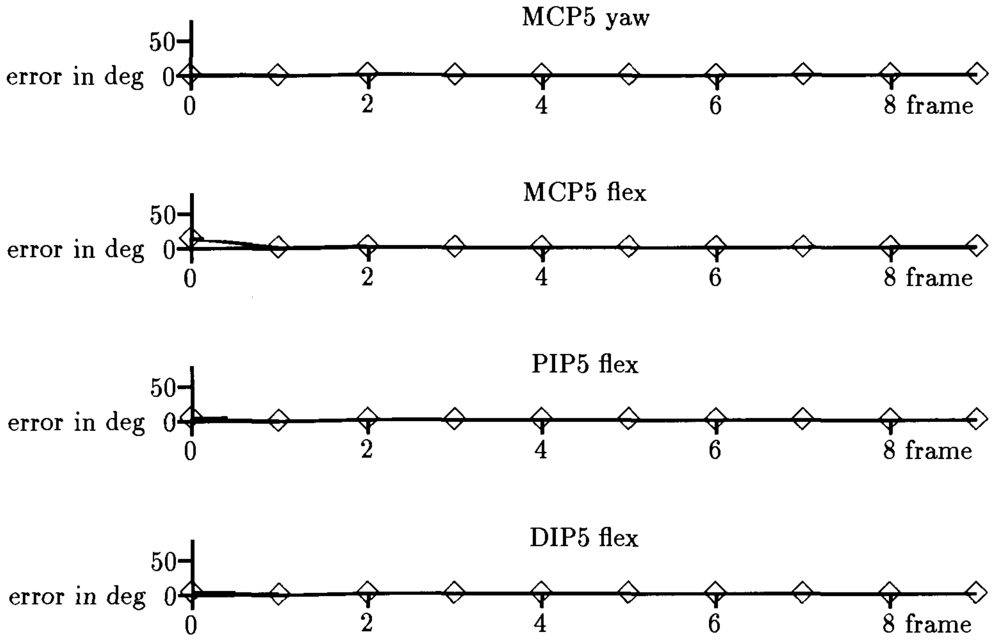

Figure 6.17: tracking ‘A’ without constraints: errors in joint angles for finger 5

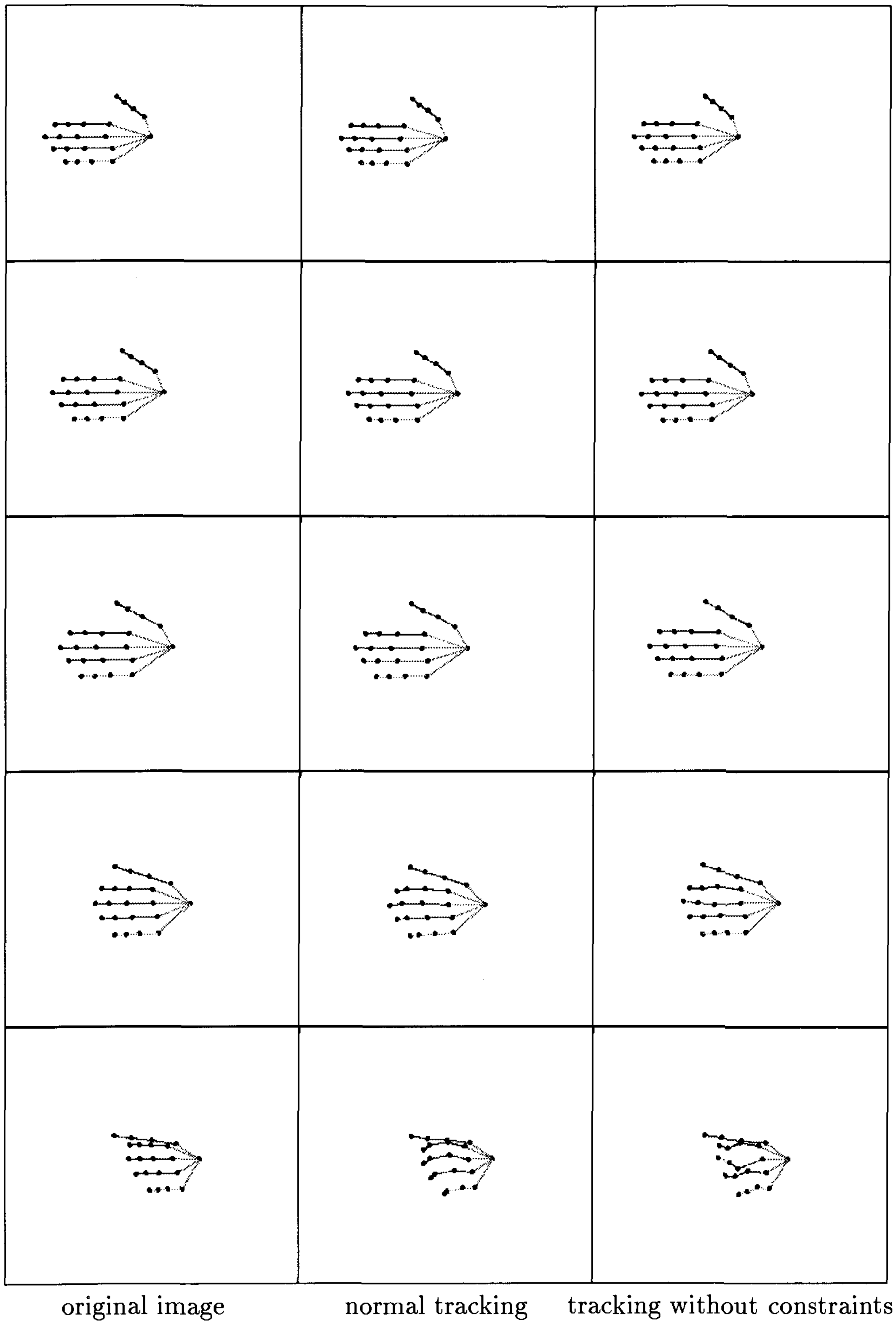

Figure 6.18: tracking 'A' from model generated input, frame 0 to 4

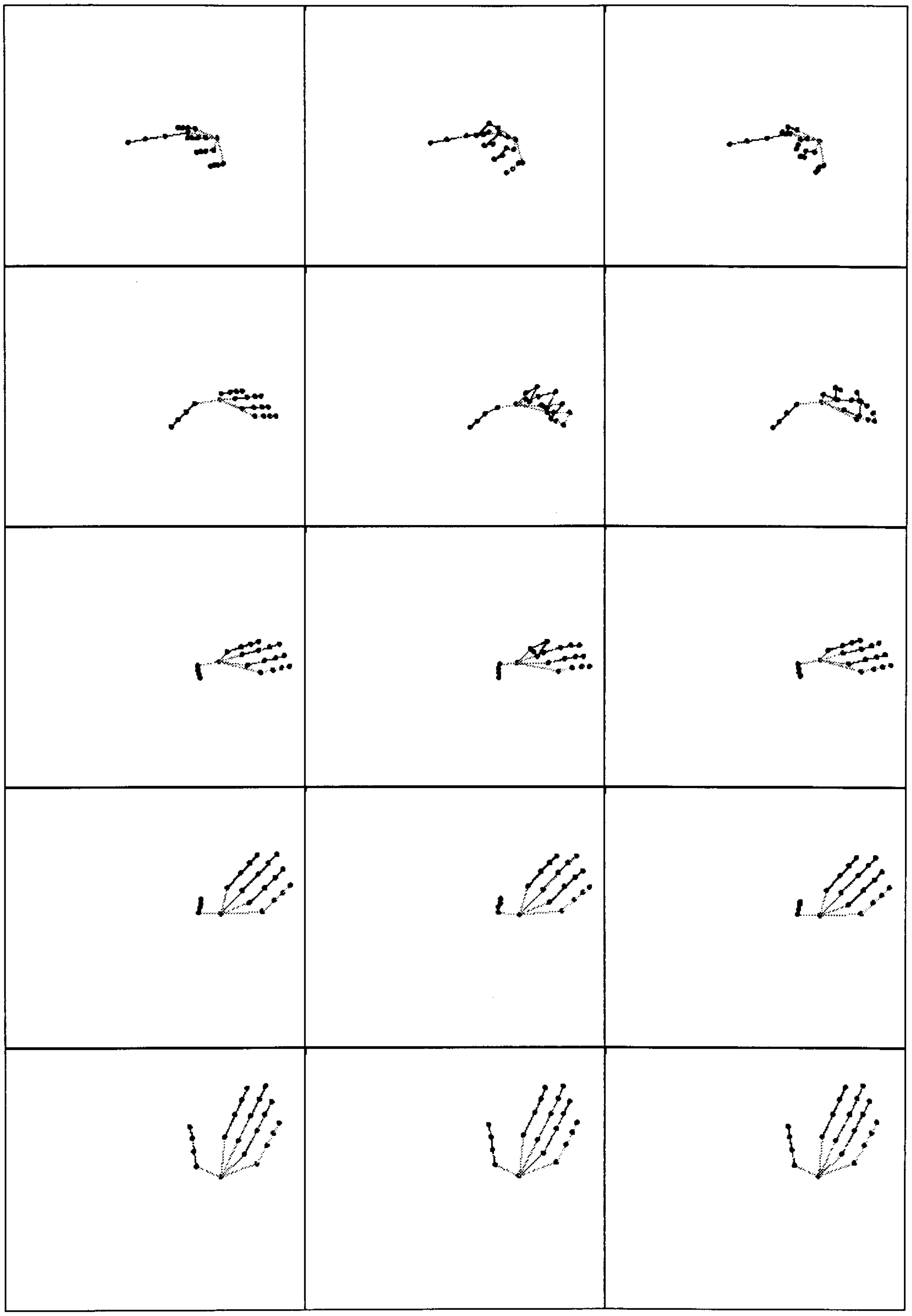

Figure 6.19: tracking 'A' from model-generated input, frame 5 to 9

6.2 Tracking real image sequences

6.2.1 Using the glove with uniquely encoded markers

Figures 6.20, 6.21, and 6.22 show the tracker in action on input derived from digital color images. The first example (figure 6.20) is again (part of[6]) a sequence that shows ASL fingerspelling of the letter 'A'. Figure 6.21 shows an excerpt[7] of another spelling sequence, the transition from 'D' to 'E' (see [7], page 13), and figure 6.22 shows part of[8] the ASL sign for *'time'* or *'what's the time?'* (see [7], page 109).

Altogether, the tracker was tested on more than 20 sequences ranging in length from 6 to 10 frames. As there was no real-time image-capturing equipment available, these series are actually series of stills made to resemble frames of a movie sequence as closely as possible. With this technique it would have been very difficult to generate test sequences that contain one or more complete signs at high frame density. Hence, the tracker was effectively tested on what corresponds to low-density image sequences.

It is difficult to give quantitative statistics for the tracker's performance on a real image series. In general, the behaviour is qualitatively the same as for model-generated input data: the recovered shape is close to the original shape, but a complicated motion or an error in marker assignment may throw the system off track.

When an error in marker assignment causes a problem the system recovers immediately as soon as the assignment is correct again (usually in the next frame). This is indicated by two sudden jumps in hand shape, when the tracker tries to adapt to the wrong marker assignment and then again when the system 'snaps back' into correct tracking.

Recovery after a difficult motion exhibits the same sudden change in shape, though the point of divergence from the correct shape is not always as marked. The same is true for errors due to occlusion (see, e.g., the position of the finger tips during the time sequence).

In any case, a sudden change of shape or, more correctly, a strong deviation from

[6]frames 0 to 5 of a 9 frame sequence
[7]frames 4 to 9
[8]frames 2 to 7

the predicted shape is a useful trouble indicator.

Performance of the marker-detection algorithm

In 151 frames (15 sequences) the marker detection algorithm made 62 wrong assignments (a marker was labeled incorrectly or noise in the image was labeled as a marker). In 56 cases this resulted in two or more alternative choices of marker position for some joint (one of them being the correct one), i.e., there were only 6 cases where an error in marker detection automatically led to an error in pose recovery. In 12 of the remaining 56 cases the correct choice could be identified without using tentative fitting.

I did not monitor the fitting algorithm for all 151 frames, but due to the motion-continuity-constraint its behaviour shows the expected pattern: whenever tentative fitting *does not* select the correct marker candidate the selected impostor is very close to the correct candidate, and the resulting error in shape recovery is small.

6.2.2 Using the glove with simple markers

The simple-marker version was tested on a 60 frame spelling sequence that was created in the same manner as the test sequences for the glove with uniquely encoded markers. Figure 6.23 shows the first 6 frames of this sequence (turning of the hand into position and, again, spelling 'A'). Figure 6.24 shows another excerpt from this spelling sequence, the transition from 'C' to 'D' (see [7], page 13).[9] To speed up the fitting process the DIP markers were eliminated by hand in this spelling sequence to reduce the number of alternative marker assignments from 24 to 6 per finger (see the discussion in section 5.2). In spite of this reduction of shape information the tracking results are almost as good as for the glove with uniquely encoded markers. Tracking

[9]The reader may have wondered about the selection of examples; I have tried to choose interesting sequences, or parts of sequences. Since many ASL signs start with initial shapes that correspond to the first letter of the English equivalent of the word or phrase being signed (see section 1.1.1 and also [45]) fingerspelling makes a good test example for the tracker. The ASL spelling of 'A', the fist gesture, is generic, in a way, as the purest of closing (and opening) gestures. The signs for 'B' and 'C' are relatively simple and are therefore skipped (see [7], page 13). The sign for time was selected because it shows the hand in a different perspective and gives an example of two-handed signing.

is less stable, however, and average model-fitting time per frame is longer (see section 6.4).

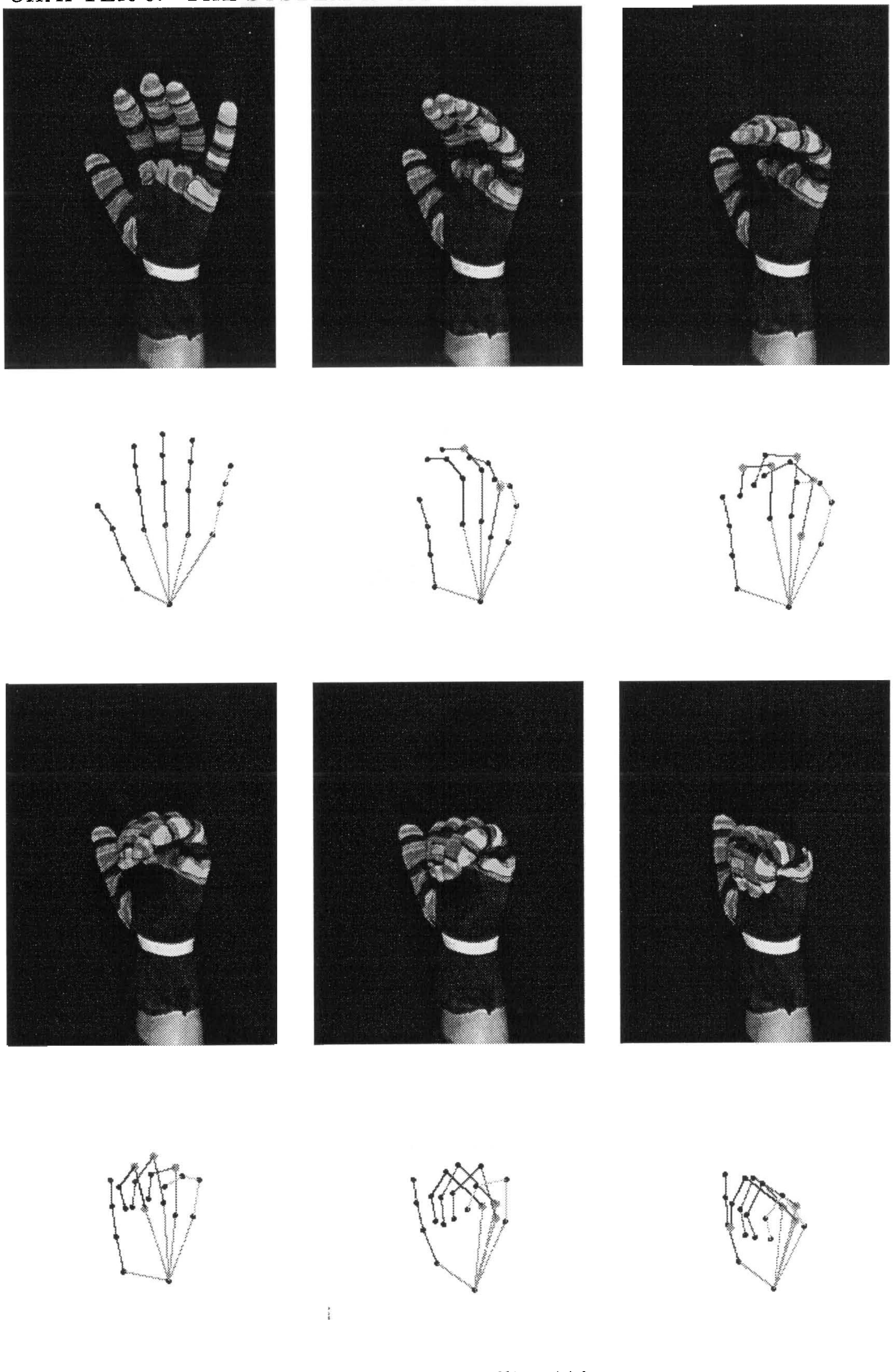

Figure 6.20: spelling 'A'

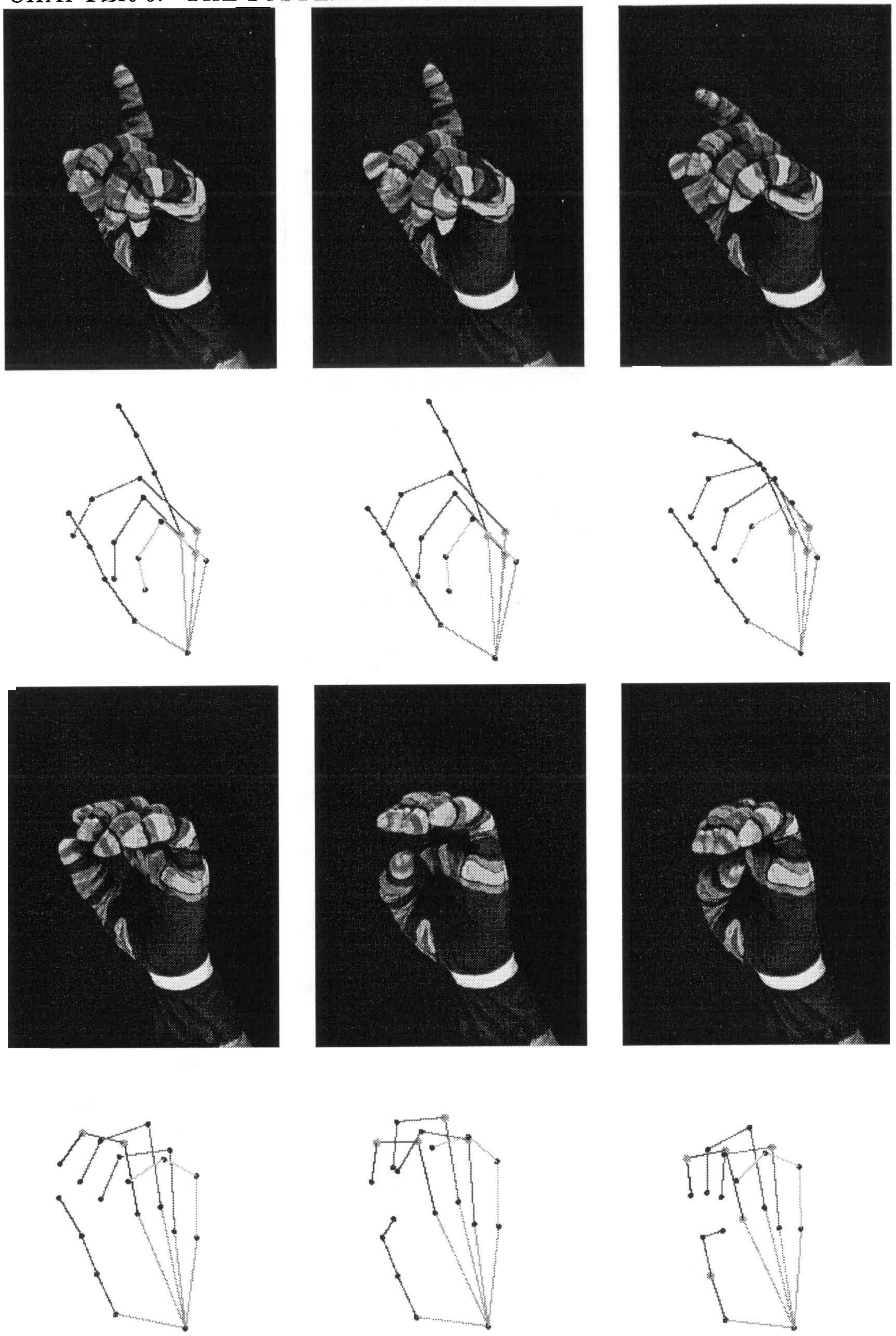

Figure 6.21: spelling 'D–E'

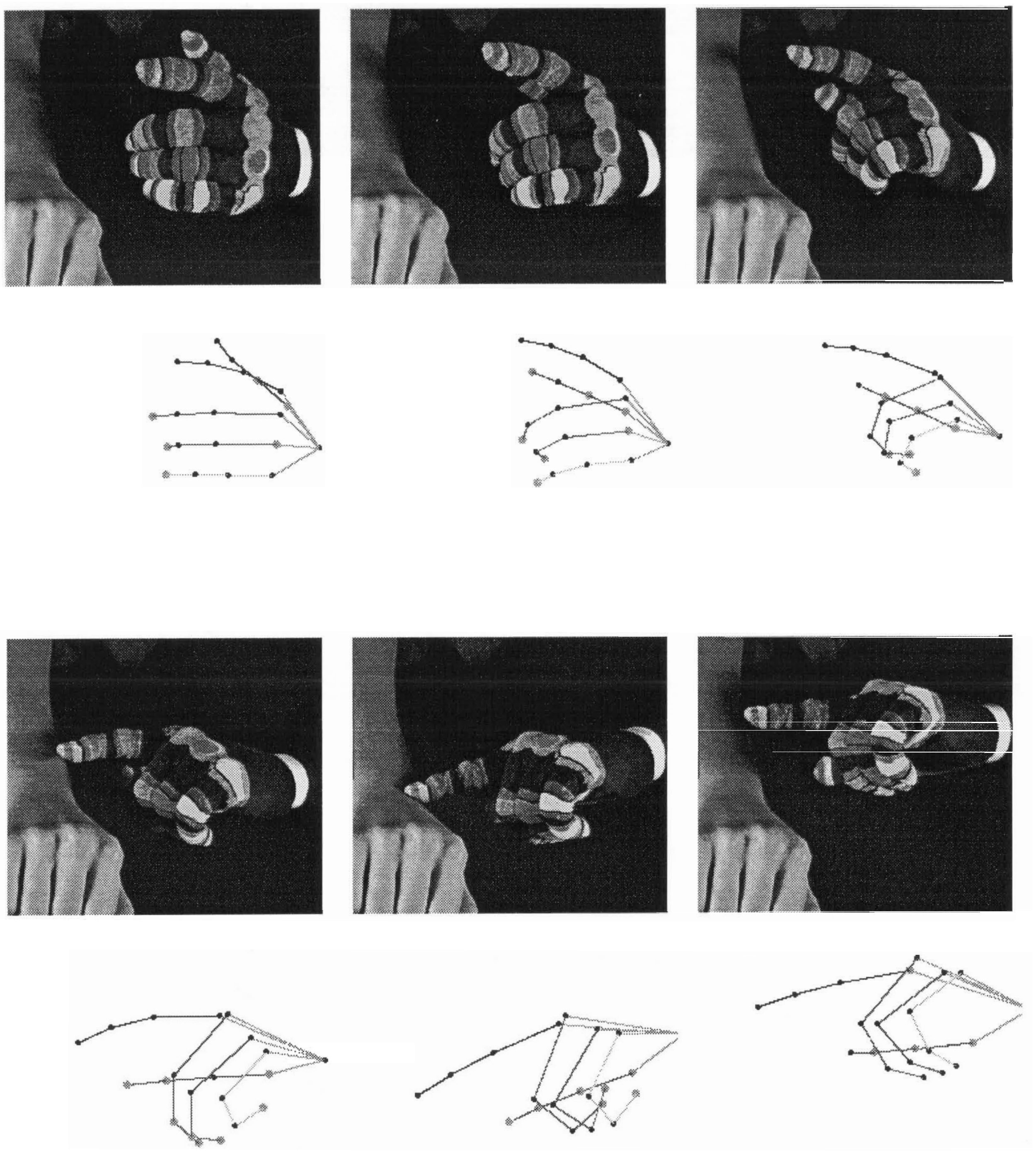
Figure 6.22: signing 'time'

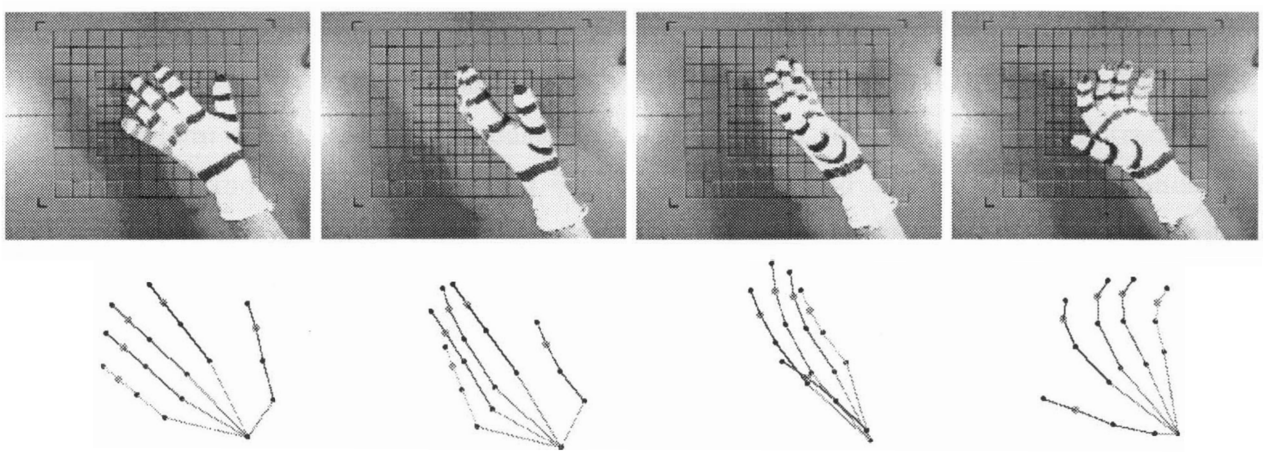

Figure 6.23: spelling ‘A’

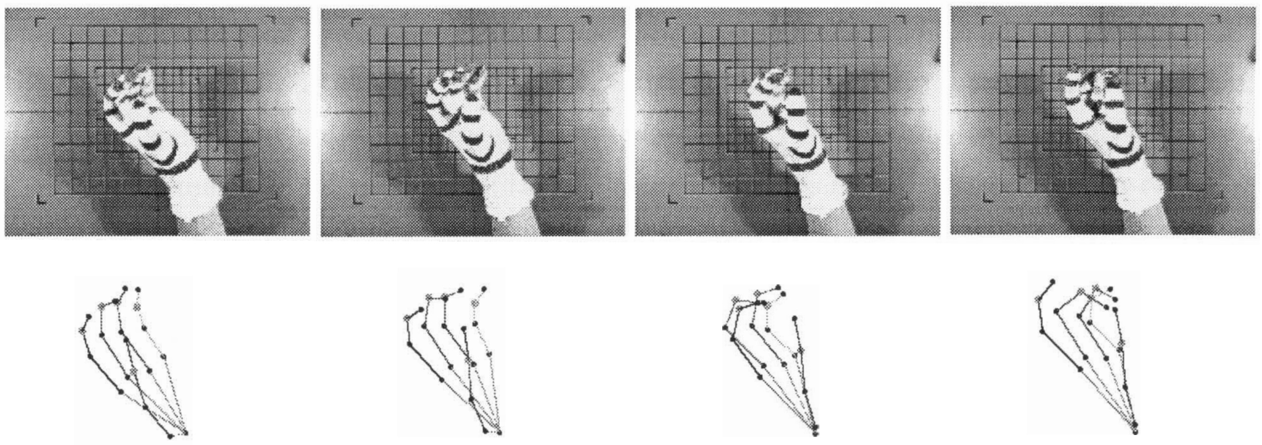

Figure 6.24: spelling ‘C-D’

6.3 Getting started

As indicated by it’s name, the hand tracking system is designed for *tracking* rather than for *finding* a hand, and it makes use of its knowledge about the hand’s previous shape and position to accomplish its task.

The system's behaviour when it is stripped of all its knowledge about the hand's physiology suggests that the fitting problem is a relatively well-behaved optimization problem: even though the constraints are essential to ensure successful tracking of difficult movements (see section 6.1.2), for many hand shapes the system finds the correct minimum even when there is a big jump between frames (and hence the initial position is not very near to the result). Thus, it is not surprising that the system does not require a special initial position or hand shape. To be on the safe side, most of the sequences tested begin with an open shape where most markers are visible, but any unambiguous shape should do.[10] The only special measure required for the startup is to disable the motion-continuity constraint for the first frame.

6.4 Speed

The main goal of this project has been to test the viability of a 3D model-based approach for hand tracking. At this stage there has been little emphasis on optimization for speed and real-time processing. Accordingly, I have made no effort to get exact benchmarks of CPU runtime; the following estimates reflect average system-response time and are intended to give the reader a rough idea of the tracker's performance.

6.4.1 Marker detection

There is special hardware for real-time color-marker detection available on the market (see [6]). Hence, the use of colored markers should be no impediment to building a real-time system based on the prototype presented here, though the marker-labeling algorithm for the unique-encoding scheme will have to be adapted.

6.4.2 Model fitting

In the experiments—all run on an IRIS 4d—the system needed 1/2 to 1 sec. per frame on average for the model-generated input sequences, and 5 to 6 sec. per frame for real

[10]Since the simple-marker version is generally more fragile than the version with uniquely encoded markers it is advisable to be conservative in this respect when using the simple-marker version.

image input for the unique-encoding scheme. For the glove with the simple-marker scheme the system took up to 40 sec. when it had to do tentative fitting for all 5 fingers.

One iteration step of the Quasi-Newton takes approximately constant time and the cost for one such iteration step is largely determined by the number of parameters in the hand model.[11]

The factor mainly responsible for the differences in execution time—apart from the additional tentative fits for the simple-marker version—is the number of iterations needed to reach an acceptable minimum. This took $\sim 10 - 15$ iterations per frame for the model-generated sequences. For the real image sequences the numbers are anywhere between 20 and 120 iterations[12] for a full hand fit (~ 40 iterations for a single-finger fit).

The slower convergence for real image sequences is a direct result of the inexact marker locations that make the fitting process less stable and may force the Quasi-Newton algorithm to estimate the Hessian matrix (see chapter 3). A simple measure to reduce the number of iterations would be to relax threshold values for the acceptance of a minimum. It certainly does not make sense to expect a solution that is more accurate than the input data, and the current thresholds are on the tight side. There should be no harm in setting these thresholds to match the expected accuracy of the tracker, though one should keep in mind that the accuracy of prediction will also be affected and the advantage gained may be lost again due to a bad initial position for the fitting.

In summary, one can say that the system is currently not fast enough for real-time applications, but there is some room for improvement. It should be possible without too much effort to make at least the version with uniquely encoded markers fast enough for real-time tracking on low frame rates, and tracking results indicate that the tracker can perform well on such low-density sequences.

[11]Calculation of constraint functions and derivatives is the main cost factor for each iteration (see [34, 22]). The calculation scheme in section 4.4 was designed to make this calculation more efficient, but I have not further optimized any details of the implementation.

[12]The fitting routine was stopped after 120 iterations even if the conditions for a minimum were not fulfilled.

Chapter 7

Conclusion

7.1 Summary: what has been accomplished so far

I have designed and implemented a system that can track a human hand over extended image sequences. The system recovers the position, orientation, and shape of the hand in 3 dimensions for each frame. It represents this information about the hand in form of a 26-dimensional *(hand) shape descriptor*: 3 dimensions each for position and orientation in signer space and 20 dimensions for joint angles. After shape recovery is finished the resulting shape descriptor can be used for various purposes, though the application that originally motivated this research is American Sign Language understanding.

The system is based on a simple 3D skeleton model of the human hand. This model is fitted to match the hand shape in the image frame with a standard numerical routine for nonlinear optimization. The metric used to measure convergence is (the sum of squares of) the Euclidean distance between joint and finger tip positions in the image and their backprojected counterparts in the model, augmented with functions that describe physical and physiological constraints on possible shape and motion of the hand.

The fitting routine aims to minimize a function that describes the position of the hand model in this metric space in terms of the parameters contained in the shape descriptor. Its output is a shape descriptor that brings the model closest to the shape observed in the image while maintaining all constraints on possible hand shape and motion.

Location of joints and finger tips in the image is facilitated by colour-coded markers painted on a glove that the signer wears during tracking. To minimize errors in joint location the markers are ring shaped wherever possible. I have tested two different marker schemes. The first scheme uses simple colour rings, one distinct colour per finger. The second scheme relies on more sophisticated encoding—3 colour rings per marker—to provide a unique marker for each joint and finger tip.

In both versions the knowledge about the hand, its current hand shape and possible motion, is used to eliminate errors and ambiguities in marker detection that could not be resolved at an earlier stage.

The hand tracker has been successfully tested on both synthetic data and real-image sequences. It can follow most hand gestures and it can recover from tracking failures after a few frames without any external interference or need for special recovery procedures.

7.2 Outlook: what remains to be done

There are several points that may be addressed here. The most important one, perhaps, is speed since most of the possible applications of the hand tracker require real-time performance. As stated above (section 6.4), tracking speed was not a high priority issue in the development of this prototype system, and, although I have tried to make the implementation reasonably efficient, there should be plenty of room for improvement. This includes simple things like code optimization and adaptation of threshold parameters. One might also want to consider more serious restructuring of the system, however.

Since the tracker can handle fairly large jumps between hand shapes it might be a good idea to go through the model-fitting cycle only when the marker locations indicate a sufficiently large change in hand shape and track the markers in the image with a 2D feature-tracking algorithm over intermittent frames (see section 4.6.2).

Another possible candidate for improvement is the fitting routine itself. There is a well known alternative/extension to the Quasi-Newton algorithm suggested by *Levenberg* and *Marquardt* ([33, 38, 32]) that enforces convergence in problematic cases. The idea is to add a linear term that transforms the Newton algorithm into a steepest-gradient-descent algorithm. The weight of this term is adjusted after each iteration, according to the algorithm's convergence rate. Although the fitting routine never actually failed to converge, convergence was sometimes rather slow for the real-image sequences (see section 6.4), and the Levenberg-Marquard algorithm may help to improve this situation.

The second major issue that comes to mind when talking about improvements is the question of tracking failures. Additional and/or better constraint functions could help there, but it is unlikely that tracking failures can be completely prevented. Luckily, the tracker's behaviour in this respect is predictable (see section 6.1.3), and it should be easy to implement a routine that monitors the tracker's behaviour and warns of possible tracking problems. One may even go a step further and try to correct these problems by re-running the fitting routine with a different initial shape or with a modified set of constraints.

Bibliography

[1] K.N. An, E.Y. Chao, W.P. Cooney, and R.L. Linscheid. Normative model of human hand for biomechanical analysis. *Journal of Biomechanics*, 12:775–788, 1979.

[2] T. Baudel and B.M. Charade. Remote control of objects using free-hand gestures. *CACM*, pages 28–35, July 1993.

[3] B. Buchholz and T.J. Amstrong. A kinematic model of the human hand to evaluate its prehensile capabilities. *Journal of Biomechanics*, 25(2):149–162, 1992.

[4] C. Charayphan and A.E. Marble. Image processing system for interpreting motion in ASL. *Journal of Biomedical Engineering*, 14(5):419–425, Sept. 1992.

[5] C.K. Chui and G. Chen. *Kalman Filtering with Real-Time Applications.* Springer Verlag, second edition, 1991.

[6] R. Cipolla, Y. Okamoto, and Y. Kuno. Robust structure from motion using motion parallax. In *International Conference on Computer Vision and Image Processing*, pages 374–382. IEEE, 1993.

[7] E. Costello. *Signing: How to Speak with your Hands.* Bantam Books, 1983.

[8] I. Craw, D. Tock, and A. Bennett. Finding face features. In *Proceedings ECCV-92*, pages 92–96, 1992.

[9] T. Darell and A.P. Pentland. Space-time gestures. In *Workshop 'Looking at People: Recognition and Interpretation of Human Action', IJCAI-93/Chambery, France.*

[10] T. Darrell and A. Pentland. Space-time gestures. In *CVPR*, pages 335–340. IEEE, 1993.

[11] J. Davis and M. Shah. Gesture recognition. Technical Report CS-TR-93-11, University of Central Florida, 1993.

[12] R.B. Davis. Clinical gait analysis. *IEEE Engineering in Medicine and Biology Magazine*, 7(3):35–40, Sept. 1988.

[13] B. Dorner. Hand shape identification and tracking for sign language interpretation. In *Workshop 'Looking at People: Recognition and Interpretation of Human Action', IJCAI-93/Chambery, France.*

[14] B. Dorner and E. Hagen. Towards an american sign language interface. to be published in*: Artificial Intelligence Review.*

[15] A.C. Downton and H. Drouet. Model-based image analysis for unconstrained upper-body motion. In *International Conference on Image Processing and its Applications*, pages 274–277. IEE, 1992.

[16] Kadaba et al. Assessment of human motion with Vicon. In *Proceedings of the Biomechanics Symposium, ASME, New York*, pages 335–338, 1987.

[17] S.S. Fells. Building adaptive interfaces with neural networks: The glove-talk pilot study. Technical Report CRG-TR-90-1, University of Toronto, Toronto, Canada, 1990.

[18] S.S. Fels and G.E. Hinton. Building adaptive interfaces with neural networks: The glove-talk pilot study. In D.Daiper, D.Gilmore, G.Cockton, and B.Shakel, editors, *Proceedings of the third International Conference on Human-Computer Interaction*, pages 683–688. North-Holland.

[19] M.J. Fletcher and R.J. Mitchell. Predicting multiple feature locations for a class of dynamic image sequences. *Image and Vision Computing*, 8(3):193–198, Aug. 1990.

[20] M.J. Fletcher, K. Warwick, and R.J. Mitchell. The application of a hybrid tracking algorithm to motion analysis. *CVPR*, pages 84–89, 1991.

[21] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, second edition, 1990.

[22] P.E. Gill and W. Murray. Algorithms for the solution of nonlinear least-squares problems. *SIAM Journal of Numerical Analysis*, 15:977–992, 1978.

[23] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

[24] E. Hagen. A Flexible American Sign Language Interface to Deductive Databases. Master's thesis, School of Computer Science, Simon Fraser University, Burnaby, Canada, 1993.

[25] P.W. Hallinan. Recognizing human eyes. *SPIE Geometric Methods in Computer Vision*, 1570:214–226, 1991.

[26] R.M. Haralick, H. Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1446, 1989.

[27] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perceptive Psychophysics*, 14(2):201–211, 1973.

[28] G. Johansson, C.Jr. von Hofstein, and G.A. Jansson. Event perception. *Annual Review of Psychology*, pages 27–63, 1980.

[29] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *ICCV*, 1987.

[30] W. Long and Y.-H. Yang. Log-tracker: an attribute-based approach to tracking human body motion. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(3):439–458, 1991.

[31] D.G. Lowe. Integrated treatment of matching and measurement errors for robus model-based motion tracking. In *International Conference on Computer Vision and Image Processing*, pages 436–440. IEEE Computer Society Press, 1990.

[32] D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.

[33] D.W. Marquard. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Applied Math.*, 11(2):431–441, 1963.

[34] The Numerical Algorithms Group Limited 1988. *NAG Fortran Library Manual*, 1988.

[35] J. O'Rourke and N. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):522–536, 1980.

[36] C. Papaspyrou and H. Zienert. The *syncwriter* computer programme. In Prillwitz and Vollhaber, editors, *Sign Language Research and Application: Proceedings of the International Congress on Sign Language Research and Application*, 1990.

[37] A. Pentland, T. Starner, N. Etcoff, A. Masiou, O. Oliyide, and M. Turk. Experiments with eigenfaces. In *Workshop 'Looking at People: Recognition and Interpretation of Human Action', IJCAI-93/Chambery, France.*

[38] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[39] A. Ralescu and H. Iwamoto. Reading faces: A fuzzy logic approach to representation, recognition and description of facial expressions. In *Workshop 'Looking at People: Recognition and Interpretation of Human Action', IJCAI-93/Chambery, France.*

[40] K. Rangarajan and M. Shah. Establishing motion correspondence. *CVGIP: image understanding*, pages 56–73, 1991.

[41] R.F. Rashid. Towards a system for the interpretation of moving light displays. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):574–581, 1980.

[42] G.J.S. Robertson and K.C. Sharman. Object location using proportions of the direction of intensity gradient. In *Proceedings Vision Interface 92*, pages 189–194, 1992.

[43] A. Samal and P.A. Iyengar. Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 25(1):65–77, 1992.

[44] I.K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, Jan. 1987.

[45] W.C. Stokoe, D.C. Casterline, and C.G. Croneberg. *A Dictionary of American Sign Language on Linguistic Principles.* Linstok Press, new edition, 1976.

[46] T. Takahashi and F. Kishino. A hand gesture recognition method and its application. *Systems and Computers in Japan*, 23(3):38–48, 1992.

[47] S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, 1988.

[48] V.C. Tartter and C.K. Knowlton. Perception of sign language from an array of 27 moving spots. *Nature*, 239:676–678, 1981.

[49] G. Verghese, K.L. Gale, and C.R. Dyer. Real-time motion tracking of three-dimensional objects. In *Proceedings 1990 IEEE International Conference on Robotics and Automation*, pages 1998–2003. IEEE Computer Society Press, 1990.

[50] Z. Zhang and O.D. Faugeras. Tracking and motion in a sequence of stereo frames. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 747–752, 1990.