



**KTH Computer Science
and Communication**

Probabilistic Tracking of Multiple Rodent Whiskers in Video Sequences

A small step towards cheap, reliable, and non-intrusive automatic tracking of whiskers.

JIM HOLMSTRÖM, EMIL LUNDBERG
jimho@kth.se, emlun@kth.se

SA104X Examensarbete inom teknisk fysik, grundnivå
Bachelor's Thesis at CSC, KTH
Supervisor: Prof. Örjan Ekeberg
Examiner: Prof. Mårten Olsson

Abstract

The interest in studying rodent whiskers has recently seen a significant increase, particularly in the field of neurophysiology. As a result, there is a need for automatic tracking of whisker movements. Currently available commercial solutions either are expensive, restrict the experiment setup, or fail in the presence of clutter or occlusion.

This thesis proposes a proof-of-concept implementation of a *probabilistic* tracking system. This solution uses a technique known as the *particle filter* to propagate a whisker model between frames of high speed video. In each frame, the next state of the model is predicted by querying a pre-trained *database* and filtering the results through the particle filter. The implementation is written in Python 2.6 using NumPy and SQLite3.

Testing results indicate that the approach is feasible. Even using a rather crude database, the tracker manages to track multiple real whiskers at once, though only for short sequences at a time. Better training data, such as hand-labeled real data, might vastly improve the result.

Keywords: Tracking, Multiple, Whisker, Particle Filter, Transition Database, Model Evaluation, Proof-of-Concept

Referat

Statistisk Följning av Multipla Morrhår hos Gnagare i Video.

Intresset för att studera morrhår hos gnagare har på senare tid ökat, speciellt inom neurofysiologin. Som ett resultat av detta finns det ett behov av automatisk följning av morrhår. Nuvarande kommersiella lösningar är antingen dyra, sätter begränsningar på experimentupställningen eller misslyckas i stökiga miljöer eller när överlappning förekommer.

Denna avhandling bidrar med en enkel implementation av ett *probabilistiskt* följsystem. Lösningen använder sig av en teknik som kallas *partikelfilter* för att propagera en morrhårmodell mellan bildrutor i höghastighetsvideo. För varje bildruta förutspås nästa tillstånd genom att fråga en förtränad *databas* och filtera svaret genom partikelfiltret. Implementationen är skriven i Python 2.6 och använder sig av programbiblioteken NumPy och SQLite3.

Testresultaten indikerar att metoden är rimlig. Med endast en grov, genererad databas lyckades algoritmen följa multipla morrhår, dock endast under korta sekvenser i taget. Bättre träningsdata, såsom handmarkerad riktig data, skulle kunna förbättra resultatet väsentligt.

Nyckelord: Följning, Multipla, Morrhår, Partikelfilter, Övergångsdatabas, Modellevaluering, Proof-of-Concept

Contents

1	Introduction	1
1.1	Difficulties	1
1.2	Contributions of this thesis	2
2	Related Work	3
3	Definitions	5
3.1	Images and Image Processing	5
3.2	States, hypotheses and estimates	6
4	Theory	7
4.1	Markov processes	7
4.2	The Hidden Markov Model	7
4.3	The Curse of Dimensionality	9
4.3.1	Overcoming the Curse	10
4.4	The Particle Filter	10
4.4.1	The Particle Filter algorithm	12
4.5	Visual Cues	12
4.6	Response	13
4.7	Model	14
4.7.1	Simplifications	15
4.7.2	Difference measure	15
4.7.3	Polynomial $a_3\omega^3 + a_2\omega^2 + a_1\omega$	15
4.7.4	Sine series $\sum a_k \sin\left(\frac{k\pi\omega}{L}\right)$	16
4.7.5	Theoretical evaluation	16
5	Algorithms and Implementations	17
5.1	The particle filter	17
5.1.1	Initilization x_0	18
5.2	The state transition database	18
5.2.1	Data format	18
5.2.2	Prediction $p(x_t x_{t-1})$	18
5.3	Tracker	19

5.3.1	Filtering $p(I_t x_t)$	20
5.4	Preprocessing of real images	20
5.4.1	Background subtraction	22
5.4.2	Extract the body $p(\text{body})$	23
5.4.3	Extract the whiskers $p(\text{whisker})$	23
5.4.4	Find the snout coordinate system	24
6	Results	27
6.1	Parameter Benchmark	27
6.1.1	Test data	27
6.1.2	Evaluated parameters	28
6.1.3	Benchmark procedure	29
6.1.4	Results	30
6.2	Run on real data	33
7	Analysis and Discussion	37
7.1	Discussion of benchmark results	37
7.2	Possible improvements	39
7.3	Conclusions	40
	Bibliography	41

Chapter 1

Introduction

With the ever increasing power and mobility of computers, computer vision has recently seen a large increase in interest. Encompassing problems such as classification, recognition, perception and tracking, application of the discipline could make many tasks easier and more efficient.

In particular, biology and neuroscience researchers are interested in tracking the movements and posture [3] of animals or parts of animals. One such field aims to study the movements of rodent whiskers. However, most whisker tracking software available today suffers a few fundamental flaws. They are either so expensive that not even well funded laboratories feel they can afford them or have problems tracking multiple whiskers at once, often requiring removal of almost all whiskers. Some higher precision systems impose other restrictions on the experiment, such as restraining the animal or attaching motion capture markers to the whiskers. The latter does not seem to be a significant problem [4], but keeping the animal restrained eliminates the possibility of studying whisker use in exploration.

1.1 Difficulties

In general, the main difficulty in tracking and localization is to separate the tracked object from clutter and occlusion. In 2001, Hedvig Sidenbladh investigated probabilistic methods for tracking three-dimensional human motion in monocular video. [6] Many of the problems inherent in computer vision were regarded, and the thesis shows that powerful conclusions can be drawn by combining multiple visual cues.¹

The most apparent problems in whisker tracking is occlusion and motion blur. Other, more subtle, problems include 3D to 2D projection ambiguities and that the whisker root is constantly occluded by facial hairs. The latter gives us the problem of not knowing where the whiskers are rooted, making the whiskers more difficult to model. The relatively low spatial and temporal resolution in high-speed video may also result in sub-pixel whiskers and motion blur.[3]

¹The solution in this thesis employs only a single visual cue.

1.2 Contributions of this thesis

This thesis provides three main contributions:

Functional model Use of functions as the model, and the L^p norm to better represent "distance" between two hypotheses, for the prediction part of the particle filter.

Proof of concept An example implementation showing that probabilistic whisker tracking is indeed feasible.

Parameter investigation Parameter analysis on the algorithm and how the tracking parameters affect the quality of tracking.

This mainly addresses the problem of occlusion. Using a model makes the tracking more robust against crossing whiskers, as opposed to pure frame-by-frame image analysis where this is often a great problem. Similarly, the problem of overlapping whiskers becomes less apparent, but is still an issue.

Chapter 2

Related Work

The following is a summary of some of the work that has been done in the field of automatic whisker tracking, as well as a Ph.D thesis on probabilistic tracking of human motion. The work in this thesis will be based on the latter since the whisker tracking problem is similiar, though in many regards simpler.

In 2011, Roy et al. [4] describe a whisker tracking system that uses motion capture markers and two tracking cameras to track whisker movements in 3D. A spatial resolution of < 0.5 mm in all dimensions and a temporal resolution of 5 ms is reported. The impact of the markers on whisker movements were investigated by comparison with a light beam detection system, and no significant difference was detected. The system requires head fixation since the markers need to be visible to both cameras at all times.

In 2008, Voigts et al. [9] developed a system that uses frame-by-frame image analysis on off-line monocular images, and does not impose other restrictions on the setup. The solution is based on creating vector fields using anisotropy in the image and tracks movements along the full length of the whiskers. It is fully automatic, and successfully tracks up to 8 whiskers on each side of the snout simultaneously, though it suffers some difficulties when applied on full whisker arrays.

In 2001, Hedvig Sidenbladh [6] investigated the general problem of tracking 3D human motion in monocular video without making assumptions about the appearance of the human or environment. The thesis sets up a probabilistic framework and combines multiple visual cues, and achieves good accuracy in tests. It has been used throughout the work on this thesis as inspiration and a reference on probabilistic tracking.

Chapter 3

Definitions

3.1 Images and Image Processing

Definition 1. A grayscale *image* can be defined as a function

$$\begin{aligned} I : \mathbb{N}^2 &\rightarrow \mathbb{R}^+ \\ I : \text{position} &\rightarrow \text{intensity}. \end{aligned} \tag{3.1}$$

It can also be identified with $\mathbb{N}^2 \times \mathbb{R}^+$ as the tuple $\langle \text{position}, \text{intensity} \rangle$. The *image space* is denoted as \mathcal{I} in this thesis.

In a computer an image is represented as an integer matrix, often 8 bit integers.¹

Definition 2. A *video* is a function mapping an integer to an image:

$$\text{video} : \mathbb{N} \rightarrow \mathcal{I}. \tag{3.2}$$

Definition 3. The rendering function R takes a hypothesis x and renders an image with a resemblance of how a real whisker would have looked like having the same underlying model and parameters as x .

$$\begin{aligned} R : \mathcal{X} &\rightarrow \mathcal{I} \\ x &\mapsto R(x) \end{aligned} \tag{3.3}$$

Definition 4. The addition operation on images is performed by element-wise addition.

$$\begin{aligned} + : \mathcal{I} \times \mathcal{I} &\rightarrow \mathcal{I} \\ (I_a, I_b) &\mapsto I_a + I_b \end{aligned} \tag{3.4}$$

The structure $\langle \mathcal{I}, +, * \rangle$ inherits the properties from $\langle \mathbb{R}^+, +, * \rangle$ by just being a vectorized version.

¹Integers in the range $[0, 255]$.

Definition 5. The multiplication operation on images is element-wise multiplication.

$$\begin{aligned} * : \mathcal{I} \times \mathcal{I} &\rightarrow \mathcal{I} \\ (I_a, I_b) &\mapsto I_a * I_b \end{aligned} \tag{3.5}$$

Definition 6. We will use subtraction loosely² as element-wise subtraction and then subtract the smallest element on all elements to make the operation closed.

$$\begin{aligned} - : \mathcal{I} \times \mathcal{I} &\rightarrow \mathcal{I} \\ (I_a, I_b) &\mapsto I_a - I_b \end{aligned} \tag{3.6}$$

Definition 7. The image transformation ϕ takes an image I and returns a transformed image.

$$\begin{aligned} \phi : \mathcal{I} &\rightarrow \mathcal{I} \\ I &\mapsto \phi(I) \end{aligned} \tag{3.7}$$

3.2 States, hypotheses and estimates

A system is said to have a *state*. The state is some quantity that defines the qualities of the system. Below follows definitions of key terms and quantities used throughout the thesis.

State The state of a system is denoted Z . When time is relevant, the state at time t is denoted Z_t .

State space The set \mathcal{Z} of all possible states, $Z \in \mathcal{Z}$.

Hypothesis A guess x at the state Z of a system.

Hypothesis space The set \mathcal{X} of all possible hypotheses, $x \in \mathcal{X}$. In general, $\mathcal{X} \neq \mathcal{Z}$ since most models are simplifications of the system.

Estimate The hypothesis x^* we believe approximates Z best.

Observation In general, it is not possible to directly record the state Z of a system.³ We instead get an *observation* I of the state.

Degrees of Freedom The number of adjustable parameters in a model, often abbreviated DOF.

Note that all of the above depend on the model used.

²Meaning, no analysis on the structure is done.

³If it were, there would be no need for tracking.

Chapter 4

Theory

The core of the tracking engine is a technique known as the *particle filter*. The particle filter is a kind of Bayesian filtering where one uses discrete hypotheses, also known as *particles*, to approximate continuous probability density functions (PDFs) [8]. It builds upon the theory of *Markov processes* and the *hidden Markov model*.

4.1 Markov processes

A Markov process is a special case of a stochastic process. For a Markov process, the next state depends only on the present state and not on past states. For this reason, a Markov process is often said to be “forgetful”.

In mathematical terms, a Markov process satisfies the following:

$$p(Z_t | Z_{t-1} \wedge Z_{t-2} \wedge \cdots \wedge Z_0) = p(Z_t | Z_{t-1}), \quad (4.1)$$

$p(Z_t | Z_{t-1} \wedge Z_{t-2} \wedge \cdots \wedge Z_0)$ is the probability that the system will have state Z_t at time t , given that the previous states were $Z_{t-1}, Z_{t-2}, \dots, Z_0$. Figure 4.1 shows how the state Z changes between time steps, depending on the previous state.

4.2 The Hidden Markov Model

The working principle of the particle filter is based on the *hidden Markov model* (HMM). A HMM describes a Markov process where we cannot measure the state directly - it is “hidden”[5]. Instead we obtain an *observation* I^1 of the state. This *perception* is generally non-deterministic, so we need to denote it as $p(I_t | Z_t)$ which is the probability that we will observe I_t if the state is Z_t . Figure 4.2 shows how the observation I_t fits into the underlying Markov process.

¹In this thesis, the observation is always a grayscale *image*, therefore the observation is denoted I .

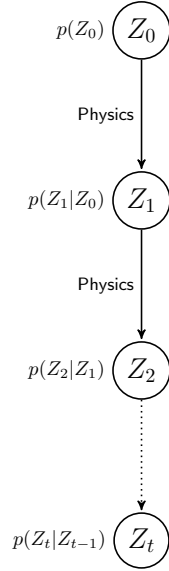


Figure 4.1. Schematic image of a Markov process.

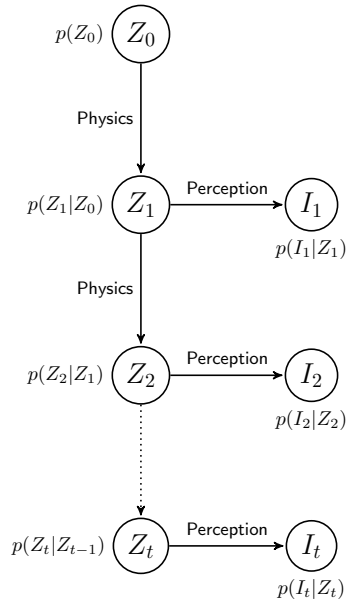


Figure 4.2. Schematic image of a hidden Markov model.

4.3. THE CURSE OF DIMENSIONALITY

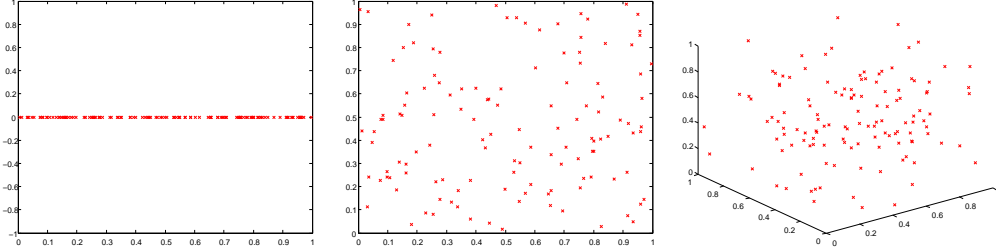


Figure 4.3. Plots of 128 scattered samples in 1, 2 and 3 dimensions, respectively.

4.3 The Curse of Dimensionality

A phenomenon that becomes apparent in high-dimensional spaces is the so-called “Curse of dimensionality” [5]. The problem is that the search volume grows exponentially with the number of dimensions. It originates from the fact that we need $\mathcal{O}(C^n)$ samples to obtain a sample density of C in a n -dimensional space.

The first consequence of this is that in order to approximate a high-dimensional function one needs orders of magnitude more samples.

The other drawback of high dimensional spaces is the “borders” of the sample set compared to lower dimensional spaces. The result is that the chance for a point one wants to approximate to fall outside the sample set is orders of magnitude greater, and the point then needs to be extrapolated instead of the better alternative of interpolation.

The following examples illustrate some of the difficulties with high dimensional spaces, as well as a case when an exhaustive is feasible.

Example 1. Figure 4.3 shows 128 randomly scattered points in 1, 2 and 3 dimensions. Notice how the density decreases with increasing dimension.

Example 2. For a 16 DOF model one needs $10^{16} = 10$ quadrillion data points to acquire a density of 10 samples per unit volume. Millions of gigabytes would be needed just to store the samples.

Example 3. In 2 dimensions it is sometimes feasible to use an exhaustive search. An example of this is the Hough transform [1], where the search is done through the $\rho\theta$ space of line responses on images.

4.3.1 Overcoming the Curse

One way to overcome the curse in the context of tracking is to perform a directed search, which could be done in one of the following ways:

1. Let the search be in an n dimensional space with a grid of g grid lines in each direction. Use the information about the most recent² location and assume that the tracked object cannot travel more than $R < g$ grid steps in one time step. This reduces the volume of the (discrete) search space from $\mathcal{O}(g^n)$ to $\mathcal{O}(R^n)$.
2. With prior knowledge of how the tracked objects move³ we can direct our search to specific regions in the state space, depending on how probable it is for the tracked object to be located there. This reduces the size of the search space depending on how sure we are of the previous state.

The next section will describe a technique for reducing the size of the search space using the second of these methods.

4.4 The Particle Filter

The *particle filter* is a technique for reducing the size of the search space. It uses a finite set X_t of hypotheses to approximate the PDF $p(Z_t|Z_{t-1})$ of a HMM. The hypotheses X_t are also referred to as *particles*, thereby the term “particle filter”.

Figure 4.4 shows the principle of the particle filter working alongside a hidden Markov model. The following is the core function of the particle filter:

The particle filter attempts to approximate the PDF $p(Z_t|Z_{t-1})$ as a set X_t of discrete hypotheses.

More particles mean greater accuracy, since the PDF can then be approximated more closely. However, using many particles increases computational cost. The particle filter employs a few tricks to *filter* the hypotheses, tending to keep probable ones and throwing improbable ones away, in order to intelligently reduce the number of particles needed for a good approximation. The filter works in four steps:

Prediction The hypotheses X_{t-1} are updated in the *prediction* step to an approximation \tilde{X}_t of $p(Z_t|Z_{t-1})$. This is done by drawing new samples from $p(x_t|x_{t-1})$, for each x_{t-1} in X_{t-1} .

Perception By measuring the state of the system, we gain an *observation* $I_t \sim p(I_t|Z_t)$ of the state Z_t .

²In the Bayesian case, the most recent estimate

³Such as the state transition probabilities $p(Z_t|Z_{t-1})$ in a HMM

4.4. THE PARTICLE FILTER

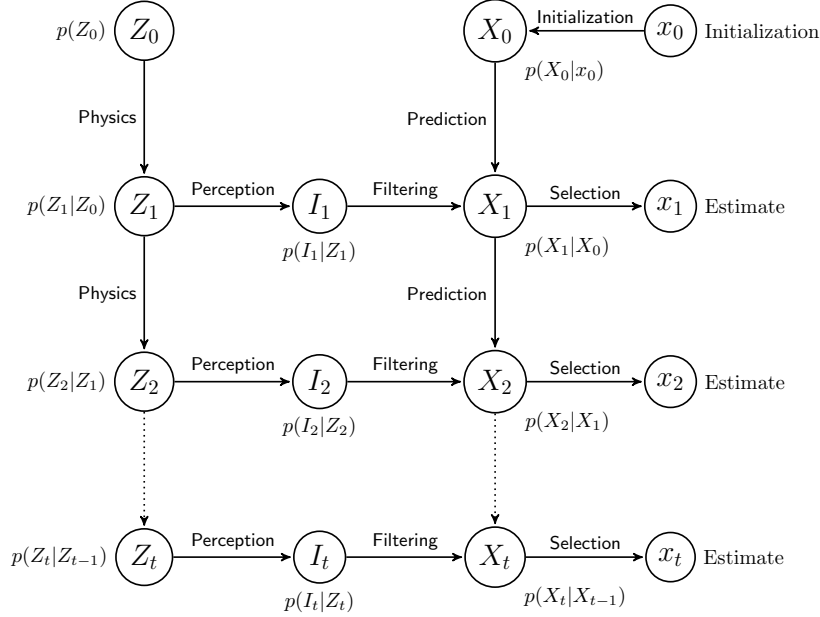


Figure 4.4. Schematic image of the particle filter alongside a HMM.

Filtering The observation I_t of the system is then used for filtering bad hypotheses out of \bar{X}_t . We draw samples X_t from \bar{X}_t with probabilities given by $p(I_t|\bar{X}_t)$. The resulting X_t will be a subset of \bar{X}_t where more probable hypotheses appear multiple times. For this reason, this is also known as the *resampling* step. The set X_t is the *belief*, our approximation of $p(Z_t|Z_{t-1})$.

The property $\bar{X}_t \subset X_t$ can cause problems if the prediction step is deterministic. This is because hypotheses “condense” into the most probable ones during filtering, meaning that all hypotheses will eventually be the same if the prediction step is deterministic.

Selection Finally, we produce a single hypothesis x_t from X_t as our *estimate* of the state Z_t . Assuming X_t is a good approximation of $p(Z_t|Z_{t-1})$, and that $p(Z_t|Z_{t-1})$ is unimodal, the mean value of X_t is a good estimate since it approximates the expectation of $p(Z_t|Z_{t-1})$. If $p(Z_t|Z_{t-1})$ is multimodal, however, the mean could be a bad estimate since the expectation may be very improbable.

The reason why this works is the following theorem, which is Theorem 3.1 in [6] with some modifications to notation:

Theorem 1. Given the PDFs $p(x_t|x_{t-1})$, $p(I_t|x_t)$ and $p(x_{t-1}|I_{t-1} \wedge \dots \wedge I_0)$, the PDF $p(x_t|I_t \wedge \dots \wedge I_0)$ can be expressed as

$$p(x_t|I_t \wedge \dots \wedge I_0) = \kappa p(I_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|I_{t-1} \wedge \dots \wedge I_0)dx_{t-1}, \quad (4.2)$$

where κ is a normalization constant[6].

The proof is beyond the scope of this thesis.

Remembering that we approximate $p(x_{t-1}|I_{t-1} \wedge \dots \wedge I_0)$ with the set X_{t-1} , one can see the connection to the following algorithm.

4.4.1 The Particle Filter algorithm

```

PARTICLE-FILTER( $X_{t-1}, I_t$ )
1   $\bar{X}_t \leftarrow \emptyset$ 
2  for each  $x_{t-1} \in X_{t-1}$ 
3      do
4           $x_t \leftarrow \text{PREDICT}(x_{t-1})$ 
5           $w \leftarrow \text{IMPORTANCE}(x_t, I_t)$ 
6          Append  $\langle x_t, w \rangle$  to  $\bar{X}_t$ 
7
8   $X_t \leftarrow \emptyset$ 
9  while  $|X_t| < |\bar{X}_t|$ 
10     do
11         Take  $\langle x_t, w \rangle$  from  $\bar{X}_t$  with probability  $\propto w$ 
12         Append  $x_t$  to  $X_t$ 
13 return  $X_t$ 

```

Table 4.1. The particle filter algorithm.

Table 4.1 shows the particle filter algorithm. Note that the functions PREDICT and IMPORTANCE are unspecified - they are problem specific. They correspond to the PDF $p(x_t|x_{t-1})$ and $p(I_t|x_t)$, respectively.

In this thesis, the real x_{t-1} is not known. Rather x_{t-1} is estimated with a set X_{t-1} of N particles.

4.5 Visual Cues

The biggest problem with computer vision is that computers do not have vision, only a data input device in the form of a camera.

4.6. RESPONSE

A *visual cue* is an image transformation ϕ that extracts some property of the image, such as intensity⁴, edges, ridges[6] or different refinements as in section 5.4.

4.6 Response

Definition 8. A *response* is how much a hypothesis matches an image. The similarity measure for hypothesis response in this thesis is defined by

$$\langle x_t, I_t \rangle_\phi = \sum \phi(R(x_t)) * \phi(I_t) \quad (4.3)$$

where $\langle x_t, I_t \rangle_\phi$ will denote the response.⁵

Assuming R renders x_t perfectly and that I_t does not have clutter, the maximum response will uniquely⁶, be when the two underlying models coincide, by theorem 2. How nice the response peak is is highly dependent on all the terms in (4.3).

Theorem 2. Let f be a positive Riemann function with finite support, defined on a set Ω . Then

$$\operatorname{argmax}_{\bar{e}} \left(\int_{\Omega} f(\bar{x}) f(\bar{x} - \bar{e}) \right) = 0 \quad (4.4)$$

Proof. We first define the window function

$$W_a^b(x) = \begin{cases} 0, & x < a \\ 1, & a \leq x \leq b \\ 0, & x > b \end{cases}$$

Multiplication: $(W_a^b W_c^d)(x) = W_{\max(a,c)}^{\min(b,d)}(x)$

Translation: $W_a^b(x - e) = W_{a+e}^{b+e}(x)$

Integration: $\int_{\mathbb{R}} W_a^b(x) dx = \Theta(b - a).$

$$\begin{aligned} \operatorname{argmax}_e \left(\int W_a^b(x) W_a^b(x - e) \right) &= \\ \operatorname{argmax}_e \left(\int W_a^b(x) W_{a+e}^{b+e}(x) \right) &= \\ \operatorname{argmax}_e \left(\int W_{\max(a,a+e)}^{\min(b,b+e)}(x) \right) &= \\ \operatorname{argmax}_e (\Theta(\min(b, b+e) - \max(a, a+e))) &= 0. \end{aligned}$$

⁴In our case this is possible since we have a homogeneous object in the form of a backlit rodent

⁵The notation $\langle \cdot, \cdot \rangle_\phi$ for response was chosen to show the similarity with inner product.

⁶Disregarding the phenomenon of having $\exists x_a \neq x_b : R(x_a) = R(x_b)$

This trivially holds for superposition of windows, since all windows will scale and translate the same way. With a finite support, $e = 0$ is the only solution. Additionally, this also holds in higher finite dimensions since we can just repeat the process one dimension at a time.

All Riemann functions with compact support can be written as a superposition of finite number of finite windows like this

$$f(x) = \sum c_i W_{a_i}^{b_i}(x),$$

and therefore (4.4) holds for any Riemann function f .

□

4.7 Model

A whisker can be modeled as a function:

Definition 9. Let

$$\begin{aligned} \text{whisker} : \mathbb{R}^+ &\rightarrow \mathbb{R}^2 \\ \omega &\mapsto \text{whisker}(\omega) \end{aligned} \tag{4.5}$$

where ω is a coordinate along the length of the whisker and $\text{whisker}(\omega)$ is a point in the x - y plane.

The tracking problem is then to find a function whisker^* that approximates whisker. The function class used in a model must satisfy the following conditions:

1. It must be able to approximate the whisker sufficiently well.

Example 4. A straight line will not suffice since the whisker is generally curved and straight lines can not fill that partition of the space.

2. The functions must be C_1 and be possible to represent with a finite number of parameters.

With this in mind, two classes of functions immediately appear as candidates:

- Polynomials $\sum_{i=0}^n a_i \omega^i$
- Fourier series $\sum_{k=0}^n a_k \sin(\frac{k\pi\omega}{L}) + b_k \cos(\frac{k\pi\omega}{L})$

A brief analysis of both will follow, after a discussion of simplifications and difference measures..

4.7. MODEL

4.7.1 Simplifications

One simplification one might make is to assume that the root of the whisker is fixed in some point on the snout. This means that we can let the whisker function be defined in a head-fixed coordinate system for each whisker, with the root of the whisker at the origin. This gives us the boundary condition

$$\text{whisker}^*(0) = \bar{0}. \quad (4.6)$$

Manual inspection of whisker videos suggests that this is not the case for real whiskers. However, there seems to be some point within the snouts that stays approximately still and can be regarded as the root of the whisker. A better model could take this into account, but that will not be covered in this thesis.

The thickness of a whisker is not constant, but decreases as the distance from the head increases. A simple model for this is to define the whisker thickness as

$$d(\omega) = \begin{cases} D - \frac{D\omega}{L}, & \omega < L \\ 0, & \omega \geq L \end{cases} \quad (4.7)$$

where $D > 0$ is the thickness at the root and $L > 0$ is the total length of the whisker.

4.7.2 Difference measure

The standard way to quantify distance between functions defined on an interval $[a, b]$ is to use the norm in the $L^2([a, b], 1)$ Hilbert space. In this thesis, the norms for $p = 2, 4, 8$ will be used, and the impact of the choice of p will be investigated. This means that condition 1 above says the model must be such that the L^p distance $\|\text{whisker} - \text{whisker}^*\|_{L^p}$ can be made sufficiently small.

4.7.3 Polynomial $a_3\omega^3 + a_2\omega^2 + a_1\omega$

The first and simplest candidate is the polynomials. Manual tests in MATLAB indicate that three terms are enough to approximate a whisker well enough that the difference is not visible to the naked eye. A whisker function can therefore be modeled as a third degree polynomial $a_3\omega^3 + a_2\omega^2 + a_1\omega$, also known as a *spline*.

These define parameterized curves in the xy plane as

$$(\omega, a_3\omega^3 + a_2\omega^2 + a_1\omega) \quad (4.8)$$

This choice can be justified by comparing with the theory of beams under small deformations in solid mechanics. After all, a whisker is not too different from a beam. The two main assumptions for this to hold is that deformations are small and that the effects of motion on whisker deformation are negligible. [2] This may not quite be the case, but a spline is still capable of approximating the momentaneous shape of a whisker well enough.

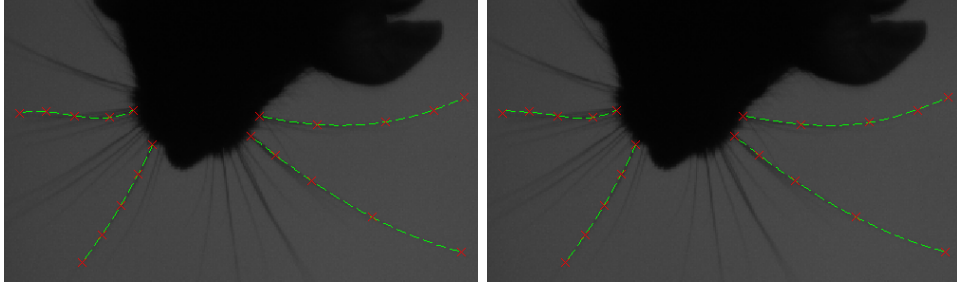


Figure 4.5. Comparison of whisker models. Left: Splines, Right: Sine series

4.7.4 Sine series $\sum a_k \sin\left(\frac{k\pi\omega}{L}\right)$

Another promising candidate is the Fourier series. Considering (4.6), the Fourier series is reduced to a sine series $\sum a_k \sin\left(\frac{k\pi\omega}{L}\right)$. However, such a series will always be zero at $\omega = L$, and is therefore not a reasonable model for whiskers. To address this, one could instead use quarter-periods: $\sum a_k \sin\left(\frac{k\pi\omega}{2L}\right)$, but at the cost of losing the orthogonality property of the sine basis. Manual tests in MATLAB indicate that such a series to third order performs approximately the same as a spline.

These define parameterized curves in the xy plane as

$$(\omega, \sum a_n \sin(\frac{2\pi n}{L}\omega)) \quad (4.9)$$

The choice of a sine series can be justified by comparing with the theory of stiff strings in analytical mechanics. The movement equation for a stiff string is a partial differential equation containing the second and fourth spatial derivatives.⁷ A classic separation of variables solution to the equation would be a sine series for the spatial part.

4.7.5 Theoretical evaluation

It is hard to theoretically justify the choice of whisker* model since we do not have an analytic model for the whisker.

For this thesis, the spline model was used in the tracking engine. The main reason for this is that it is slightly simpler than the sine model. It also is rather easy to get an intuitive grasp of how the parameters affect the curve shape - the ω^3 term mostly affects the tip of the whisker while the ω term affects the overall orientation.

⁷Source: Solving exercise 7.10 of [7] using variational calculus.

Chapter 5

Algorithms and Implementations

The testing implementation was developed with high modularity in mind, since it is meant to be a proof-of-concept implementation and not a production grade system. High modularity also makes development easier and the system more robust against changes, two very important qualities during this project.

The implementation is written in Python using NumPy and SQLite3, and consists of three main parts:

Particle Filter A direct implementation of the procedure in table 4.1.

Database A database with functions for extracting transition hypotheses. Provides the prediction PDF $p(x_t|x_{t-1})$ to the particle filter.

Tracker Manages the model and performs matching between hypotheses and images. Provides the filtering PDF $p(I_n|x_n)$ to the particle filter.

The implementation was developed in two steps. It was first tested on tracking white squares against a black background. The purpose of this was to begin with very simple test cases until the general parts of the algorithm were in place. After that, whisker tracking modules were developed and tested on artificially generated whisker videos.

5.1 The particle filter

The particle filter implementation is a direct implementation of the procedure in table 4.1. It is implemented as a function that takes the parameters X_{t-1} , I_t , `importance_function` and `sampling_function`. The parameters are the hypotheses from the last time step, the current video frame and the functions to use as PREDICT and IMPORTANCE in 4.1, respectively. This means that the particle filter function is general and independent of the model used. The implementations of PREDICT and IMPORTANCE are provided by the database and tracker, respectively.

5.1.1 Initilization x_0

The test implementation needs to be manually initialized. When tracking generated whiskers, the states were always known and the initialization could therefore be programmatically inserted. When testing on real whiskers, the start states were calculated by manually selecting five or six pixels along each whisker and using a MATLAB script to find the least squares solution for the coefficients (a_3, a_2, a_1) . The problem of automatic initialization is a difficult one [6], and is not covered in this thesis.

5.2 The state transition database

5.2.1 Data format

A *state transition* is a pair $(x_{\text{from}}, x_{\text{to}})$ that denotes we have observed a system go from state x_{from} to state x_{to} in one time step. Technically, the state transition database is implemented as an SQLite3 database. One transition is represented in the database as a row with the state parameters of the model before and after the transition. The set of transitions in the database will be denoted \mathcal{T} .

5.2.2 Prediction $p(x_t|x_{t-1})$

```

DB-PREDICT( $x_{t-1}$ )
1   $x_t \leftarrow 0$ 
2   $W \leftarrow 0$ 
3  for each  $(x_{\text{from}}, x_{\text{to}}) \in \mathcal{T}$ 
4      do
5           $w \leftarrow (\|x_{\text{from}} - x_t\|_{L^p})^{-a}$ 
6           $x_t \leftarrow x_t + w \cdot x_{\text{to}}$ 
7           $W \leftarrow W + w$ 
8  Take  $v \sim \mathcal{N}(0, \Sigma)$ 
9  return  $x_t/W + v$ 
    
```

Table 5.1. Pseudocode for the prediction function. Notice the parameters a and p .

The PREDICT function in table 4.1 is implemented as a weighted mean of the state transitions in the database. The function is stated in table 5.1. Notice the parameters a and p . p is a positive integer that determines which L^p space to compute the norm in. a is a positive number, and determines how fast the weight w declines with the distance $\|x_{\text{from}} - x_{t-1}\|_{L^p}$. A high a means closer transitions get a much higher weight than ones far away, see figure 5.1.

5.3. TRACKER

At this point, however, the prediction is still deterministic - the result for any given input x_{t-1} is completely determined by the parameters a and p and the content of the database. A deterministic prediction function is not desirable, since the filtering step only removes improbable hypotheses and replaces them with duplicates of probable ones. This means that having a deterministic prediction effectively reduces the number of hypotheses with each filtering step. For this reason, the result is offset by a small normal distributed term¹ v to make the prediction nondeterministic.

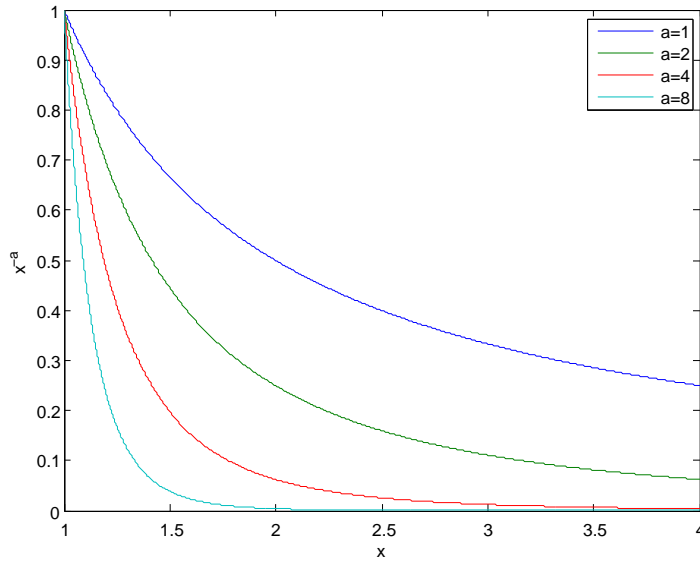


Figure 5.1. Transitions closer to x_{t-1} receive a much greater weight if a is large.

5.3 Tracker

The tracker uses the whisker model described in chapter 4 and internally represents whiskers with the tuple (a_3, a_2, a_1) of polynomial coefficients. When multiple whiskers are to be tracked, they are tracked independently in sequence as individual tracking assignments. This means that the whisker models cannot account for each other's positions.

¹The offset v is a polynomial $b_3\omega^3 + b_2\omega^2 + b_1\omega$ where coefficient $b_i \sim \mathcal{N}(0, \sigma_i)$ and σ_i is different for the different i .

5.3.1 Filtering $p(I_t|x_t)$

The IMPORTANCE function in table 4.1 is implemented simply as the response of x_t on I_t , raised to a power g . A high g means that the peak in the response is further amplified. Figure 5.2 shows an example of a rendered hypothesis, as used for computing the response. See section 4.6 for details on the response.

```

IMPORTANCE( $x_t, I_t$ )
1  return  $\langle x_t, I_t \rangle_\phi^g$ 
    
```

Table 5.2. Pseudocode for the importance function. Notice the parameter g .

The IMPORTANCE function in table 4.1 as being the response for x_t on I_t .



Figure 5.2. Example rendered image $R(x_t)$ for some hypothesis x_t .

To indicate how the response works for generated and real data we look at the following figures. Figure 5.3 shows the response curve for a generated image with parameters $(0, 0, 0)$ when varying the different parameters in the polynomial model. Figure 5.4 shows the response curve for real images which has approximately the parameters $(0, 0, 0)$, we can clearly see that the curve is far from perfect. The ground truth peak is still around $(0, 0, 0)$ but much less apparent. The faulty peaks are generated by clutter from the other whiskers and we can see a tendency of high responses for positive values. This is explained by the fact that there was more clutter in the positive region in the tested image.

5.4 Preprocessing of real images

In the images of real rodents used for testing, the rodent was illuminated from below, meaning it and its whiskers were dark against a bright background, as shown in figure 5.5. The filtering function described in the previous section expects whisker pixels to be bright, so the images had to be inverted. However, the body had the same pixel values as the whiskers, meaning no conclusions can be drawn simply by inspecting the value of a pixel. Therefore the following steps also had to be taken.

5.4. PREPROCESSING OF REAL IMAGES

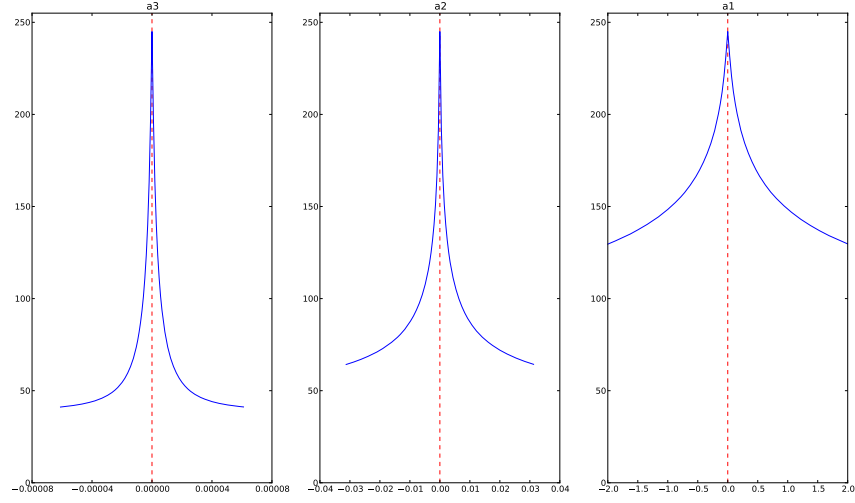


Figure 5.3. Response curves over the parameters (a_3, a_2, a_1) of the polynomial model on a generated whiskers image with the parameters $(0, 0, 0)$. With $\phi = 1$.

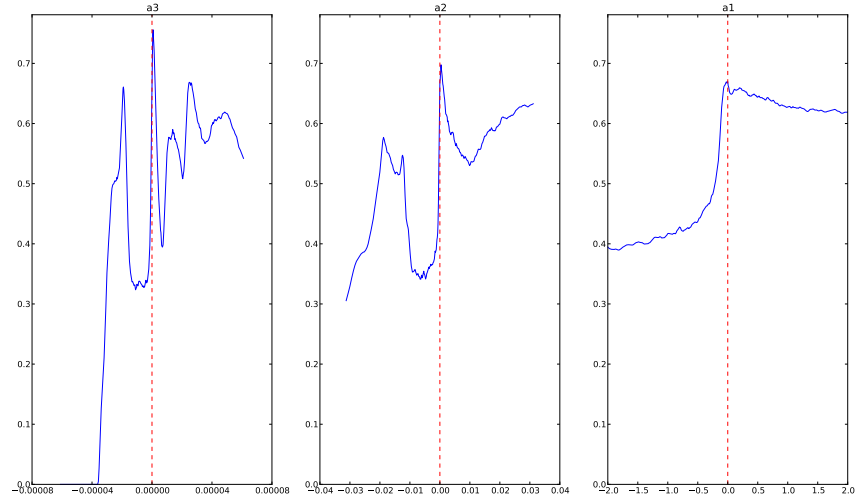


Figure 5.4. Response curves over the parameters (a_3, a_2, a_1) of the polynomial model on a real whisker image, $p(\text{whisker})$ from 5.4.3, with approximately the parameters $(0, 0, 0)$. With $\phi = 1$.

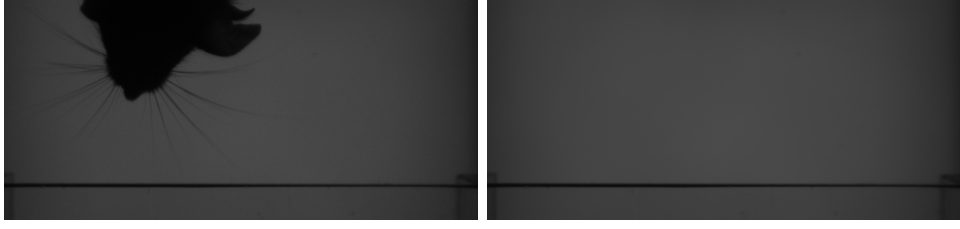


Figure 5.5. Left: Original image, Right: Mean static background of the first 100 frames.

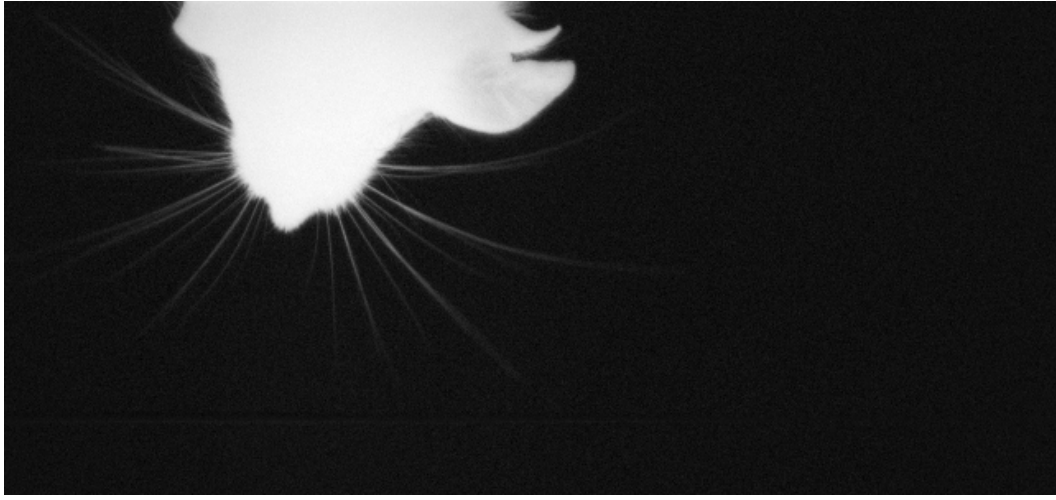


Figure 5.6. Image after background subtraction and inversion.

5.4.1 Background subtraction

Removing effects like difference in background illumination and static objects is preferred in order to minimize faulty responses.

Assuming we always have a static camera setup for each sequence and that we have an adequate sequence without the rodent, the background BG was captured a priori by taking the mean I_{BG} of the first 100 frames as seen to the right in figure 5.5.² Then we can simply subtract the background I_{BG} from the original image I and getting the result shown in figure 5.6:³

$$I_{FG} = I - I_{BG}. \quad (5.1)$$

²A sufficient number

³Worth noting is that more sophisticated background subtraction methods exist, but this solution works relatively well considering the simplicity.

5.4. PREPROCESSING OF REAL IMAGES



Figure 5.7. The image to the left shows the blurred image with $\sigma = 9\text{px}$, to the right we have the snout mask.

5.4.2 Extract the body $p(\text{body})$

Extracting the body has two purposes. First, to find the head-fixed coordinate system. Second, to subtract the body from the image in order to highlight the whiskers.

Definition 10. The image

$$p(\text{body}) \in \mathcal{I} \quad (5.2)$$

is an estimation of the probability of the pixel being a body pixel.⁴

We will assume that the only non-static objects in the image are the whiskers and body. One simple feature that easily classifies $p(\text{whisker})$ from $p(\text{body})$ is size. Blurring the image removes fine structures and retains larger which can be seen to the left in figure 5.7. This was performed by convoluting the image with a Gaussian with $\sigma = 9\text{px}$:⁵

$$p(\text{body}) = I_{\text{blur}} = I_{FG} \star \mathcal{N}(0, \sigma). \quad (5.3)$$

Note that $\mathcal{N}(\mu, \sigma)$ traditionally denotes a set of normal distributed stochastic variables. In this thesis, we will also use this to denote the PDF of the normal distribution, and let the context provide the distinction.

$p(\text{body})$ was then used to create a *mask* I_{body} :

$$I_{\text{body}} = p(\text{body}) > 0.6 \in \mathcal{I}, \quad (5.4)$$

meaning I_{body} is white where $p(\text{body})$ is greater than 0.6 and black everywhere else, the result is displayed to the right in figure 5.7.⁶

5.4.3 Extract the whiskers $p(\text{whisker})$

The filtering described in section 5.3.1 needs an indicator for the locations of whiskers.

⁴The estimate only needs to be an indication of whether it is a body

⁵The value of σ was obtained by manual testing and inspection.

⁶The 0.6 threshold was found, through manual testing, to make the body stable between frames.

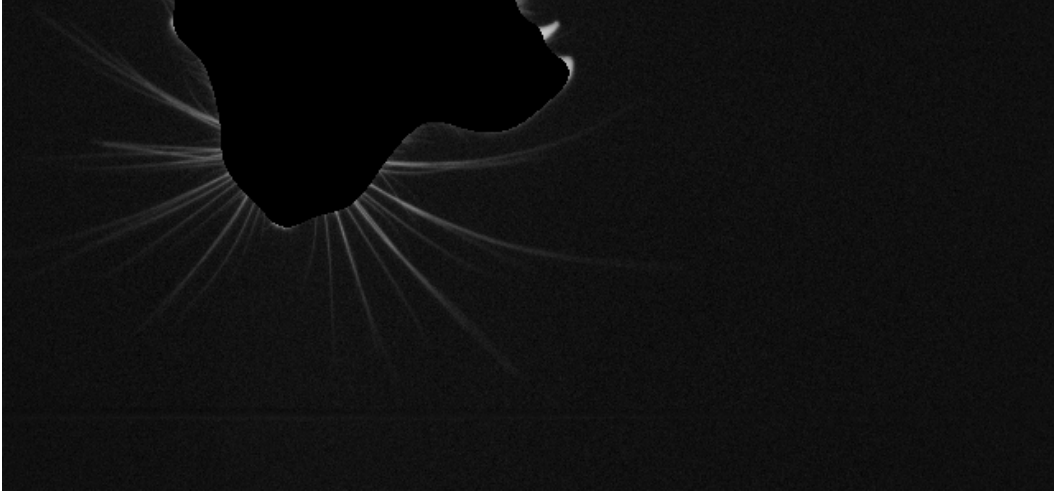


Figure 5.8. The finished preprocessed image.

Definition 11. The image

$$p(\text{whisker}) \in \mathcal{I} \quad (5.5)$$

is an estimation of the probability of a pixel being a whisker pixel.⁷

This is naively performed by masking with the inverse \bar{I}_{body} of I_{body} :

$$p(\text{whisker}) = I_{FG} * \bar{I}_{\text{body}} \quad (5.6)$$

which finally gives the results shown in figure 5.8.

5.4.4 Find the snout coordinate system

The whisker model does not take head movements into account, and therefore expects whisker roots to be stationary throughout the sequence. Therefore, the video is translated such that the head stays approximately fixed throughout the image sequence.

For this, we use a transformation ϕ that extracts the shape of the body. ϕ first blurs the image⁸, to smooth out the response, then applies a Prewitt filter:

$$\begin{aligned} \phi_{\text{blur}}(I) &= I * \mathcal{N}(0, 5\text{px}) \\ \phi(I) &= \sqrt{(\phi_{\text{blur}}(I) * \text{Prewitt}_x)^2 + (\phi_{\text{blur}}(I) * \text{Prewitt}_y)^2}. \end{aligned}$$

that extracts the shape of the body.

⁷The estimate only needs to be an indication of whether it is a whisker.

⁸The value $\sigma = 5$ pixels was obtained through manual testing.

5.4. PREPROCESSING OF REAL IMAGES

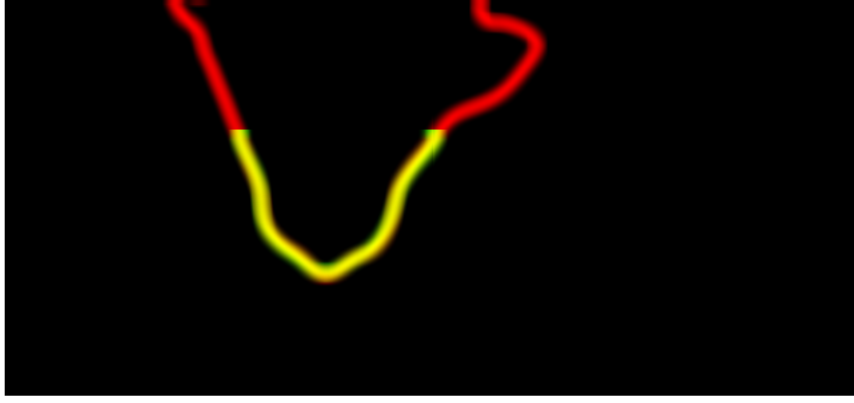


Figure 5.9. The snout coordinate system tracking in progress, here the best match is shown. Reference image is green and the tracked image is red, meaning the overlap becomes yellow.

The first frame where the snout is fully visible is hand picked and used to create an image $I_{\text{ref}} = \phi(I_{\text{body}})$, where I_{body} is created through the steps detailed in section 5.4.2.

A local search within 5px from the last location is then performed⁹ for each body image I_{body} in the sequence, which can be seen in figure 5.9. The translation $(\Delta x, \Delta y)$ from I_{ref} is extracted:¹⁰

$$(\Delta x, \Delta y) = \underset{(x,y) \text{ close}}{\operatorname{argmax}} \left(\sum \text{translate}(I_{\text{ref}}, -(x, y)) * \phi(I_{\text{body}}) \right) \in \mathbb{Z}^2, \quad (5.7)$$

and the frame is then translated accordingly.

⁹This could easily be extended to include rotation as well.

¹⁰In the same way as $\langle \cdot, \cdot \rangle_{\phi}$ but without the hypothesis.

Chapter 6

Results

The results consist of two parts:

- A benchmark of the impact of different parameters on the tracking performance, performed on artificial videos and evaluated programmatically
- A test run on real data with the best performing parameters from the benchmark, evaluated manually

6.1 Parameter Benchmark

A benchmark was performed in order to identify how tracking performance is affected by the different parameters. The benchmark was performed on generated videos of artificial whiskers, which enabled programmatic evaluation of the results since the correct shapes of the whiskers, the *ground truth*, were known.

The benchmarks were run on about 16 computers¹ in the computer halls Grå and Karmosin at KTH for two nights. The computers were all equipped with 2.83 GHz quad-core Intel® Core™ 2 Quad processors and 3.8 GiB of RAM, and were running Ubuntu Linux 10.04.

6.1.1 Test data

The test data consisted of a single generated video of artificial whiskers. The video contained 6 whiskers and was 64 frames long. Each whisker had length 200 and a random base shape $a_3\omega^3 + a_2\omega^2 + a_1\omega$. Each a_i was in the range $[0, \sigma_i)$, where $\sigma_3 = 1.6 \cdot 10^{-5}$, $\sigma_2 = 4 \cdot 10^{-3}$, $\sigma_1 = 1$. Each whisker was then assigned a random phase $d \in [0, 2\pi)$, and the shape at time step t was $(a_3\omega^3 + a_2\omega^2 + a_1\omega) \sin(\frac{2\pi t}{30} + d)$.

The shape and the frequency $\frac{1}{30}$ were selected through manual inspection of a video of real whiskers in order to make the generated whiskers roughly realistic. The resulting whiskers were roughly reminiscent of real whiskers, and 6 sample frames can be seen in figure 6.1.

¹This varied in time as other students also wanted access to the computers.

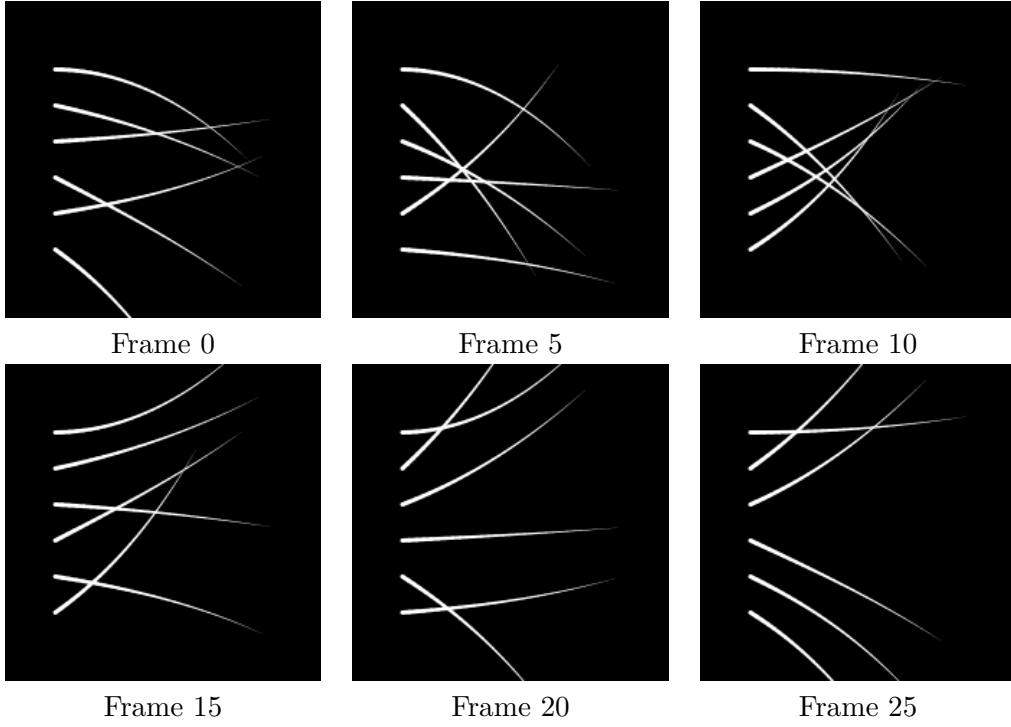


Figure 6.1. Sample frames from the testing video.

The database was generated with the same settings as the video, and contained $2^{14} = 16384$ transitions. Each transition consisted of a “from” part x_{from} and a “to” part x_{to} . x_{from} was created by generating a base state and phase in the same way as for the video, and setting $t = 0$. x_{to} was created by taking x_{from} and increasing the phase by $\frac{2\pi}{30}$.

6.1.2 Evaluated parameters

The following parameters were evaluated:

n Number of particles

p In which L^p space to compute $\|x_{\text{from}} - x_{t-1}\|_{L^p}$ in the prediction step

a The exponent for the weights in the prediction step, $w = \|x_{\text{from}} - x_{t-1}\|_{L^p}^{-a}$

σ Standard deviation modifier for the offset in the prediction step. Standard deviations for the ω^3, ω^2 and ω terms are $\sigma\sigma_3, \sigma\sigma_2$ and $\sigma\sigma_1$, respectively.²

g The exponent for the importance in the filtering step

²See section 6.1.1 for the σ_i values.

6.1. PARAMETER BENCHMARK

Parameter	Values
n	64, 128, 256, 512
p	2, 4, 8
a	1, 2, 4, 8
σ	0.025, 0.05, 0.1, 0.2
g	1, 2, 4, 8

Table 6.1. Specification of test cases.

Table 6.1 shows the tested values. The values to test were chosen for the following reasons:

- n** 512 particles was the most the computer systems could handle in reasonable time. This was then scaled down to 64 as lower particle counts are, subjectively, quite uninteresting.
- p** The L^p norm computation implementation used only handled even values of p . Computation time increased rapidly with p , so $p = 8$ was the highest value for which the computations could be done in time.
- a** The range was chosen to start from 1 because lower values would counteract the intended purpose of the parameter. The end 8 was chosen since higher values were believed to restrict the prediction too much.
- σ** The max value 0.2 was selected so that each parameter could vary roughly a fifth of the parameter space in each direction. The other three were selected as lower, exponentially spaced points.
- g** The range was selected in the same way as that of a .

All value series were spaced exponentially since orders of magnitude are most interesting in this benchmark.

6.1.3 Benchmark procedure

The benchmark was run on the test video for each combination of parameters in table 6.1. The output from the benchmark is an error matrix $\epsilon_{i,t}$, where index i, t contains $\|Z_t - x_t\|_{L^2}$, the L^2 distance between the ground truth and estimated shape for whisker i at time t . From this, a list ϵ_i was computed as the root mean square of $\epsilon_{i,t}$ in the t direction. Finally, the maximum value in ϵ_i was selected as the *error* ϵ of that run. This gives us the error tensor

$$E_{n,p,a,\sigma,g} = \max_i \sqrt{\frac{\sum_t \epsilon_{i,t}^2}{64}} \quad \forall (n, p, a, \sigma, g). \quad (6.1)$$

However, some tests took a very long time to run, and therefore some did not finish in time. This included all tests for $n = 512$ and $p = 8$, some of whose running times were observed to exceed 12 hours. These test cases and the ones specified in table 6.2 were not included in the analysis.

n	p	a	σ	g
64	8	4	0.05	8
64	8	8	0.05	1
64	8	8	0.1	2
64	8	8	0.025	4
512	2	4	0.025	4
512	2	4	0.05	4
512	2	4	0.1	4
512	4	1	0.2	8

Table 6.2. Unfinished test runs. Omitted are $(512, 8, a, \sigma, g) \forall a, \sigma, g$.

Note that the absolute magnitude of the error ϵ is irrelevant - it is the ratios between these errors that is interesting. One could argue for instead using the relative error $\|Z_t - x_t\|_{L^2} / \|Z_t\|_{L^2}$, but the relative error is misleading here. The reason is that a deviation from a very bent whisker would then be considered better than the same deviation from a very straight whisker. Therefore the absolute error is used for the analysis.

Since the error tensor is five-dimensional, displaying all of its contents is not feasible. Instead the index $(n_0, p_0, a_0, \sigma_0, g_0)$ for which $E_{n_0, p_0, a_0, \sigma_0, g_0}$ is minimized was used as the common point of the following plots. Using this index, the the cross sections

$$E_{p,a} = E_{n_0, p, a, \sigma_0, g_0} \quad \forall p, a \quad (6.2)$$

$$E_{\sigma,g} = E_{n_0, p_0, a_0, \sigma, g} \quad \forall \sigma, g \quad (6.3)$$

$$E_n = E_{n, p_0, a_0, \sigma_0, g_0} \quad \forall n \quad (6.4)$$

were extracted and plotted.

Also investigated was the effect of computing E using the L^4 and L^8 norms, in order to see if this had some correlation to the choice of p . However, the resulting $E_{p,a}$ and $E_{\sigma,g}$ for L^2, L^4 and L^8 all had the same qualitative behavior, so the L^4 and L^8 versions of E are omitted from this report.

6.1.4 Results

The minimum of $E_{n,p,a,\sigma,g}$ was found to be 38.18 for

$$n = 512, p = 2, a = 2, g = 4, s = 0.2.$$

Figures 6.2 and 6.4 show $E_{p,a}$ and $E_{\sigma,g}$, respectively. Figures 6.3 and 6.5 show the same graphs, but with the dominant points removed. Figure 6.6 shows E_n .

6.1. PARAMETER BENCHMARK

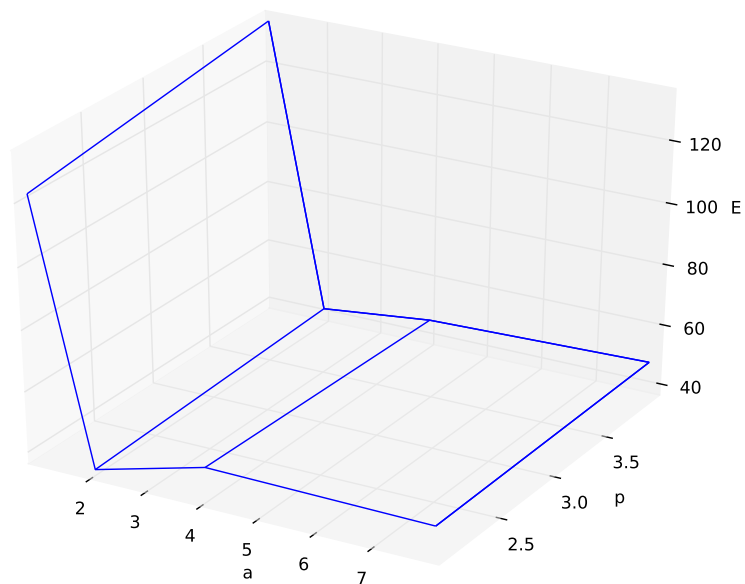


Figure 6.2. $E_{p,a}$.

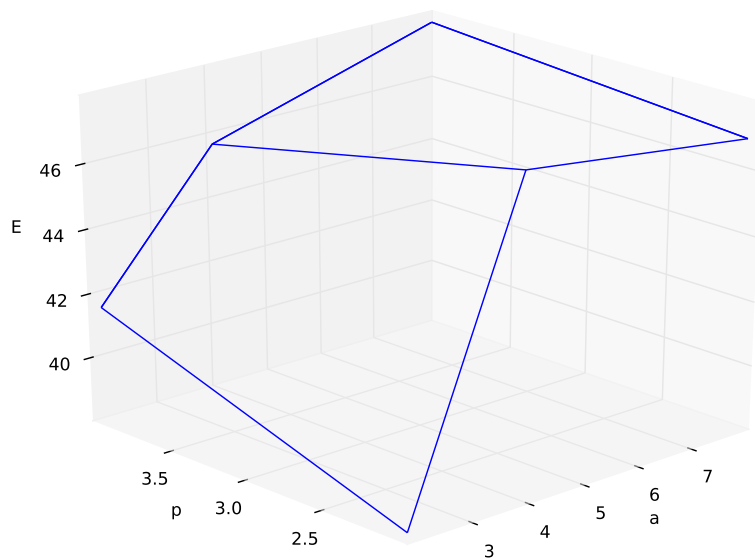


Figure 6.3. $E_{p,a}$ with dominant points removed. Notice that the plot has been rotated so that the p axis now extends to the left.

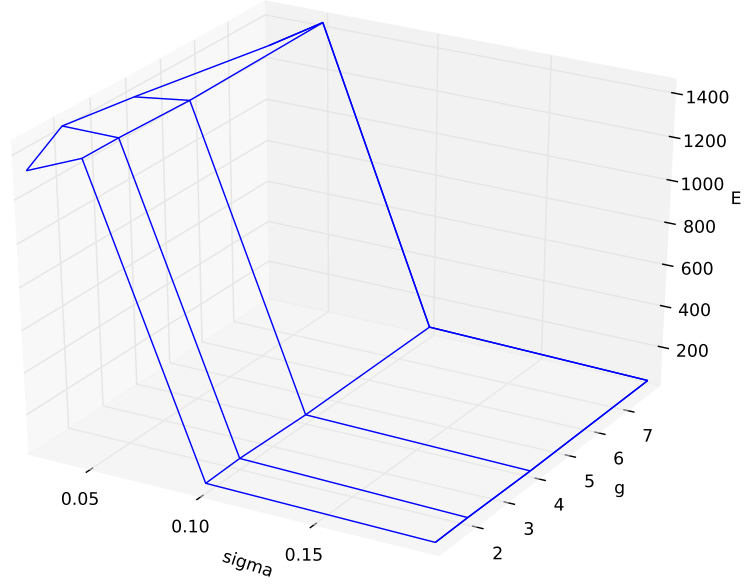


Figure 6.4. $E_{\sigma,g}$.

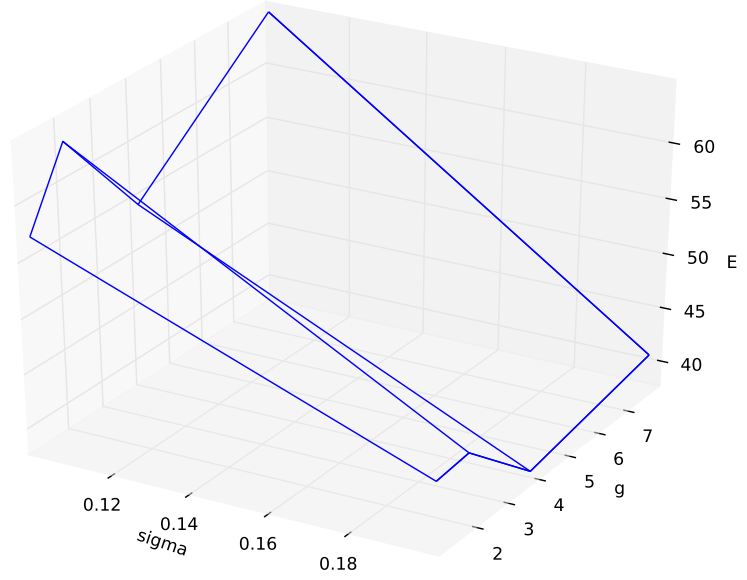


Figure 6.5. $E_{\sigma,g}$ with dominant points removed.

6.2. RUN ON REAL DATA

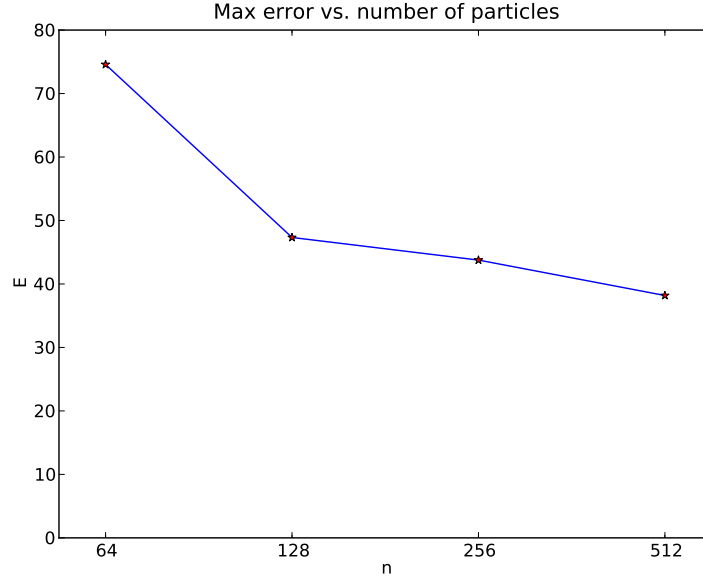


Figure 6.6. E_n . Note the logarithmic scale of n .

6.2 Run on real data

The parameters p_0, a_0, σ_0, g_0 from the parameter benchmark were used for running the tracker on a real whisker video of 131 frames. The particle count n , however, was set to 128 to reduce computation time.³ The performance was evaluated manually.

Figure 6.7 shows the first 18 frames, including the manually initialized start frame. Green lines are the estimated shapes of the whiskers.. The following figures show a selection of images from the tracking sequence. Figure 6.8 shows how the model of the lower whisker “jumps” to another whisker further down the array. Figure 6.9 shows another jump, this time resulting in both models coinciding. Figure 6.10 shows how both models simultaneously jump to other whiskers, the top being a smaller whisker.

³This was performed late in the project, and there was no time to run the test with 512 particles. The error should roughly follow the behaviour in figure 6.6, perhaps steeper since real data poses more difficulties.

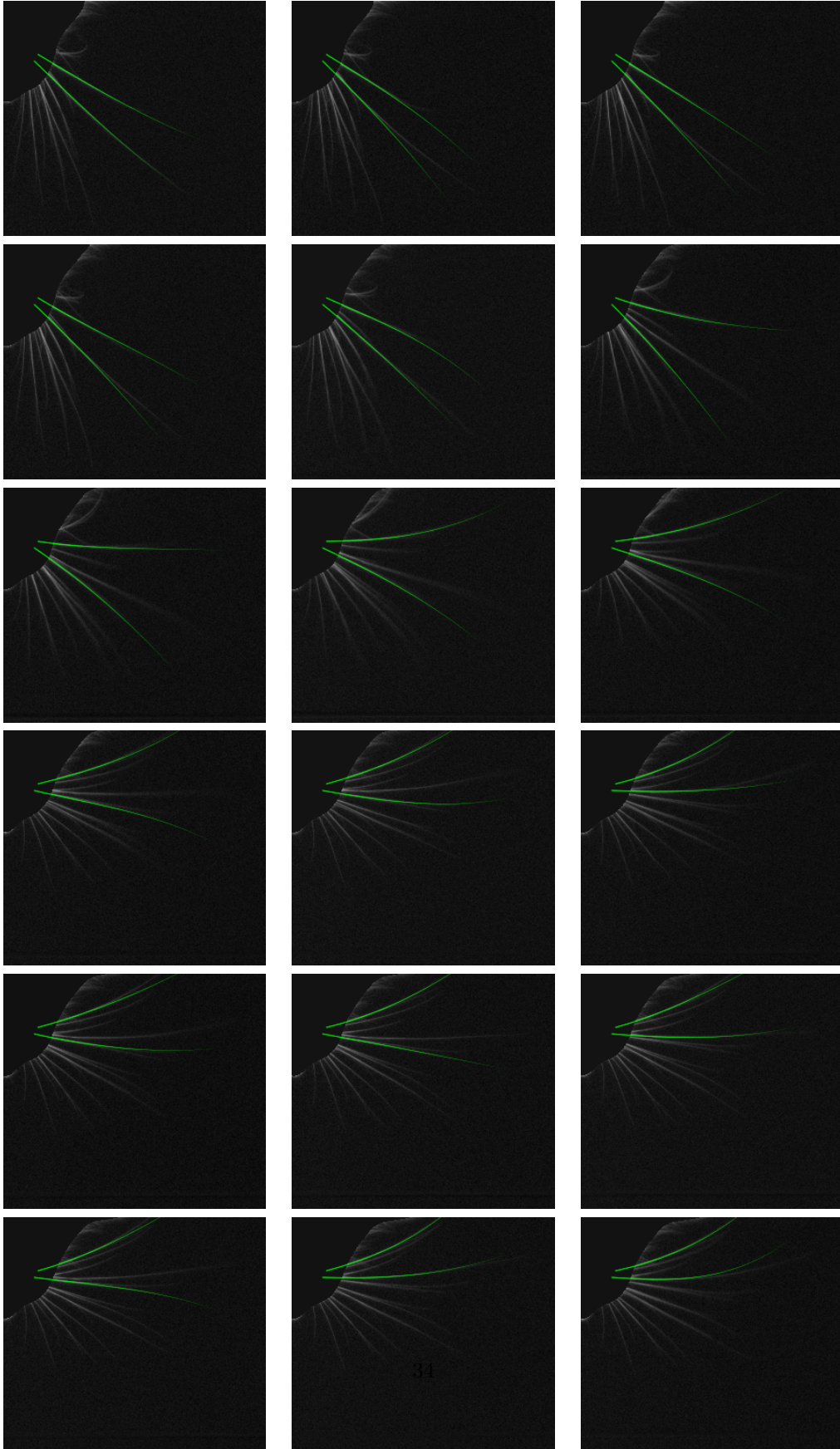


Figure 6.7. The 18 first frames from tracking results with 128 particles on real data, ordered left-to-right, top-to-bottom. The first frame shows the initialization.

6.2. RUN ON REAL DATA

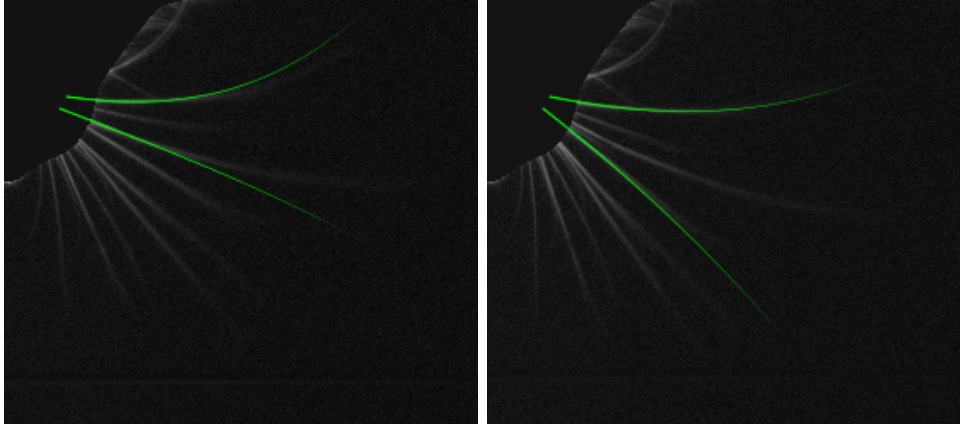


Figure 6.8. Frames 22 and 23. The model of the lower whisker “jumps” to another whisker.

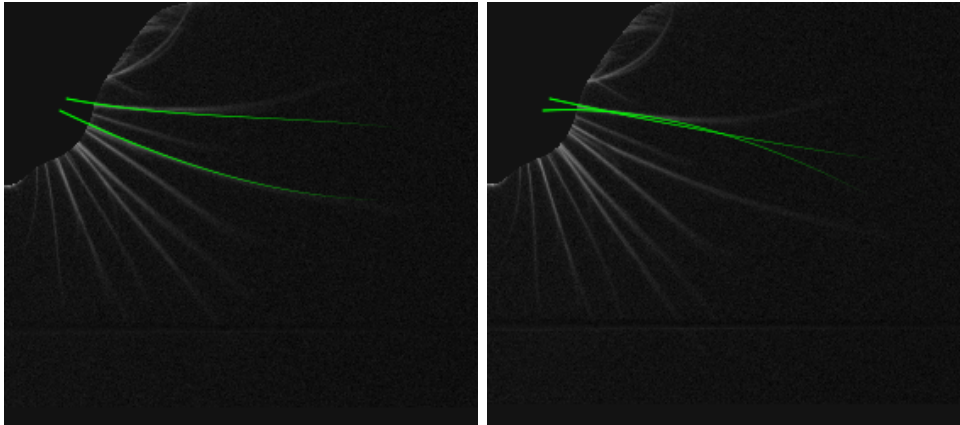


Figure 6.9. Frames 45 and 46. The model of the lower whisker jumps to another whisker, this time coinciding with the top model.

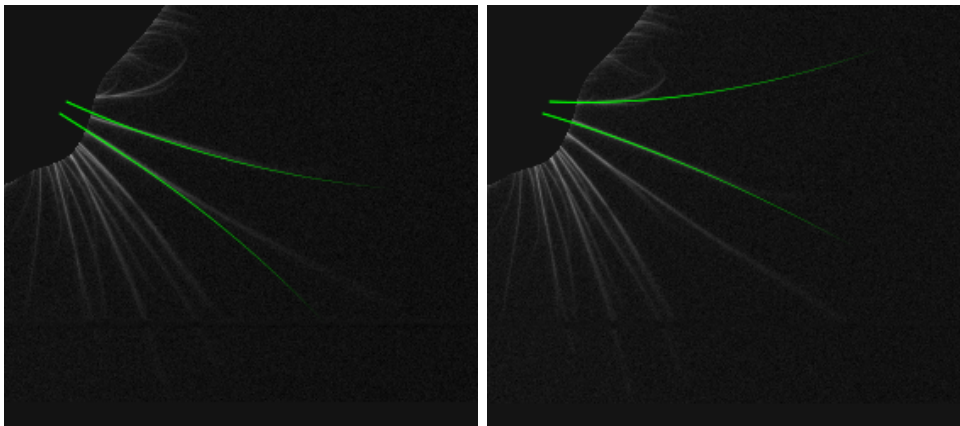


Figure 6.10. Frames 122 and 123. Both models jump simultaneously.

Chapter 7

Analysis and Discussion

Care must be taken when doing this type of experimental analysis on artificial data, and then applying the conclusions on real data. Quoting Encyclopedia of Machine Learning [5], section “Algorithm Evaluation”:

“However, much machine learning research includes experimental studies in which algorithms are compared using a set of data sets with little or no consideration given to what class of applications those data sets might represent. It is dangerous to draw general conclusions about relative performance on any application from relative performance on this sample of some unknown class of applications. Such experimental evaluation has become known disparagingly as a bake-off.”

However, the results should still be able to give some direction for the parameters for further studies on real whiskers. A configuration that fails even under ideal circumstances is not very likely to succeed on real data.

7.1 Discussion of benchmark results

As expected, the highest number of particles yielded the best result. This is not surprising, since more particles mean the PDFs can be more closely approximated. However, as figure 6.6 shows, the decrease in error per increase in particle count gets smaller and smaller, meaning even a significant increase in particle count does not guarantee a measurable decrease in error. The model used in this thesis has only 3 DOF, which can at most barely be considered high-dimensional, meaning more particles may not be necessary for similar whisker models.

The value of p does not seem to notably affect tracking performance, as seen in figure 6.2 which is almost flat in the p direction. Apparently, computing the distances in any of the tested L^p spaces seems to suffice. It should be noted, however, that many tests for $p = 8$ were left out because of the long computation time in the testing implementation. Considering the increased computation time associated

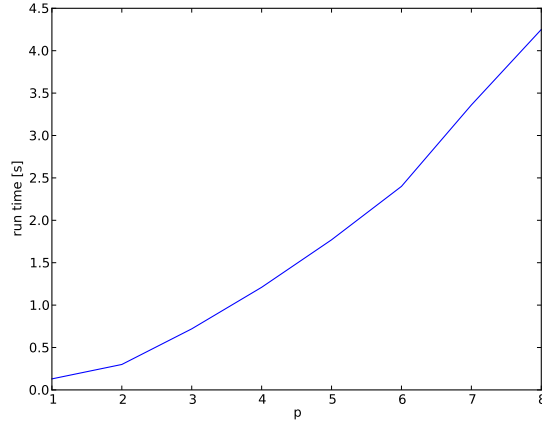


Figure 7.1. Computation times for computing $\|10^{-7}\omega^3 + 10^{-3}\omega^2 + 150\omega\|_{L^p}$ 16384 times for $p = 2, 4, 8$.

with higher p , as illustrated in figure 7.1, it is probably best to settle for $p = 2$ and instead increase the particle count, for the same computational cost or less.

The value of a that yielded the best result was $a = 2$. The error increased by more than 20% when a was increased to 4 or 8, and more than tripled for $a = 1$. This is reasonable, since a we want close transitions to receive higher weights, but a too high a would make the weighted mean converge to the argmax function instead, taking only a single training transition into account.

The modifier σ for the standard deviation of the offset to the sampled particles had a large impact on the error. The best value was 0.2, the maximum value, meaning the magnitude of the offset could be roughly a fifth of the width of the populated region of the database in each direction. This may be a bit surprising at first thought, but one has to remember that the database used was artificially generated. Even real data samples suffer selection bias if the training data is not sampled from the model one intends to learn.¹ In this case, the training data is not sampled from real data, and furthermore was generated with settings guessed after inspection of whisker videos. This means that the training data is probably a bad representative of whisker dynamics, and a large σ therefore helps “blur out” the deficiencies of the training data.

Another note on the bias effect of the database is the size of the time step. In the first runs performed on real data, the time scale of the database was approximately two thirds that of the video. The result was that the posture estimations were left behind when the whiskers moved away, and the model instances frequently jumped between whiskers in the video. In conclusion, the time scale is perhaps one of the most important properties of the training data.

Finally, the value of the parameter g that minimized E was $g = 4$. The depen-

¹As mentioned in the section “Data Preparation” of Encyclopedia of Machine Learning. [5]

7.2. POSSIBLE IMPROVEMENTS

dence of E on g , however, is unclear and no real conclusions can be drawn from the results. No qualitative behavior can be seen in the g direction of figure 6.4. It is also difficult to predict what values of g should give a good performance. A low g would make clutter too important. A high g amplifies high responses, but a high response does not necessarily mean that the posture coincides well with the whisker it is intended to track. For instance, in the test run on real video there was a small, bright whisker above the top tracked whisker, which sometimes gave a higher response than the latter.

7.2 Possible improvements

Judging by the results of the run on real data, the following are the greatest weaknesses of the testing implementation:²

- The whisker model expects whisker roots to be stationary.
- The response function³ is too simple.

A combination of these manifested clearly in the run on real data. Because of the first, the whisker models had to be rooted inside the snout, meaning no response at all could be collected in a radius of a few pixels from the root. This means that the estimated posture could emerge from the snout at multiple locations and get comparable responses, which in turn means the model could sometimes find a better match by switching to another whisker. This combined with the low temporal resolution of the video - which sometimes makes it difficult even for a human to notice a “swap” between whiskers - makes it difficult for the tracker to keep the model tracking the same whisker. We therefore draw the following conclusion.

A good whisker model must account for movements of the whisker root along the surface of the snout.

Here the “root” of the whisker refers to the point where the whisker shaft emerges from the snout.

Other possible improvements include:

- Developing a statistically rigorous response function. The one used in this thesis is very simple, and only gives a rough estimate of the probability that a hypothesis matches a whisker in the image, as indicated by figure 5.4. A better solution might be to create a classifier that estimates the probability of a pixel being part of a whisker as opposed to the probability of being a non-whisker pixel, and combine multiple visual cues.⁴ This might also include motion and

²Excluding the fact that the database is probably not representative of whisker dynamics

³Including the chosen transformation ϕ

⁴As is common in computer vision, and also used by Sidenbladh. [6]

direction cues, and not only pure pixel cues. A direction cue could consider the directions of whiskers, making it easier to cancel noise caused when whiskers cross.

One could also investigate if it is possible to combine the particle filter with the analysis developed by Voigts et al. [9] as it proved to be a powerful classifier even on its own.

- Having the tracker manage all whiskers simultaneously, as opposed to independently in sequence as is used in this thesis. The tracker could then avoid mapping multiple whisker models to the same whisker in the image, and attempt a perfect bipartite matching between time steps. This could make the jumps between whiskers less frequent, or perhaps eliminate them altogether.

7.3 Conclusions

We conclude that probabilistic whisker tracking is feasible, and our results on tracking real whiskers are even quite promising. Our simple implementation manages to track real whiskers, though the tracking is very unreliable and the posture estimations frequently jump between whiskers. With more work, including better image analysis and a database of real data, the performance could probably be vastly increased, and eventually compete with the best solutions available today.

Bibliography

- [1] Rafael C. Gonzlez and Richard E. Woods. *Digital Image Processing, Third Edition*. Pearson Education, 2008.
- [2] Hans Lundh. *Grundläggande Hållfasthetslära*. Institutionen för hållfasthetslära, KTH, 2004.
- [3] Jason T. Ritt. High-speed videography of embodied active sensing in the rodent whisker system. In Tommaso Fellin and Michael Halassa, editors, *Neuronal Network Analysis*, volume 67 of *Neuromethods*, pages 283–302. Humana Press, 2012.
- [4] Snigdha Roy, Jerí L. Bryant, Ying Cao, and Detlef H. Heck. High-precision, three-dimensional tracking of mouse whisker movements with optical motion capture technology. *Frontiers in Behavioral Neuroscience*, 5(00027), 2011.
- [5] Claude Sammut and Geoffrey Webb. *Encyclopedia of Machine Learning*. Springer, 2010 (First edition).
- [6] Hedvig Sidenbladh. *Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences*. PhD thesis, Kungliga Tekniska Högskolan, 2001.
- [7] KTH Teoretisk Fysik. Exempelsamling fysikens matematiska metoder. http://courses.theophys.kth.se/SI1142/FMM_old_examples.pdf. Accessed 2012-05-20.
- [8] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2006.
- [9] Jakob Voigts, Bert Sakmann, and Tansu Celikel. Unsupervised whisker tracking in unrestrained behaving animals. *Journal of Neurophysiology*, 100:504–515, 2008.