KTH Computer Science
and Communication

Probabilistic Tracking of Multiple Rodent Whiskers in Monocular Video Sequences <SECOND DRAFT>

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

JIM HOLMSTRÖM
JIMHO@KTH.SE
EMIL LUNDBERG
EMLUN@KTH.SE

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

TRITA xxx yyyy-nn

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 extremely 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 pretrained 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 på 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 extremt dyra, sätter begräsningar på experimentuppställningen eller misslyckas i stökiga miljöer eller när överlappning förekommer.

Denna avhandling bidrar med en enkel implementation av ett probabilistiskt följningssystem. 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 NumpPy och SQLite3.

Testresultaten indikerar att metoden är rimlig. Med endast en grovt snarlik databas lyckades algoritmen tracka 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, Modelevaluering, Proof-of-Concept

Contents

Chapter 1

Introduction

1.1 Background

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. [4] Such restrictions may cause systematic errors and not give a representative view of how the animal naturally uses its whiskers.

1.2 Difficulties

In general, the main difficulty in tracking and localization is to separate the tracked object from clutter and occlusion. In the case of whisker tracking, the dominant problems are that whiskers often overlap, and the relatively low spatial and temporal resolution in highspeed video results in subpixel whiskers and motion blur. [3]

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 apperent problems in whisker tracking is occlusion, 3D to 2D projection inambiguities and motion blur. Other, more subtle problems include that the

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

whisker root is constantly occluded by facial hairs. This gives us the problem of not knowing where the whiskers are rooted, making the whiskers more difficult to model.

1.3 Approaching the difficulties

The problems above will be addressed as they appear in the thesis.

1.4 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 a probabilistic solution is indeed feasible.

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

Chapter 2

Related Work

2.1 Examples of whisker tracking systems

2.1.1 Unsupervised Whisker Tracking in Unrestrained Behaving Animals[8]

Image flow, gabur

It works well under temporary occlusion but needs clipping of most whiskers and thereby introduce possible effects to the experimental results.

2.1.2 High-Precision, Three-Dimensional Tracking of Mouse Whisker Movements with Optical Motion Capture Technology[4]

Uses a 3D tracking system consisting of two high-speed cameras and markers mounted on the whiskers of the head-fixed mouse. The head-fixed is for the markers must be visible at all times to make the tracking work.

2.2 Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences[6]

Chapter 3

Definitions

3.1 Images and videos

3.1.1 Grayscale image

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

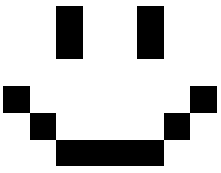
$$\begin{array}{rcl} I : \mathbb{N}^2 & \rightarrow & \mathbb{R}^+ \\ I : \text{position} & \rightarrow & \text{intensity}. \end{array} \tag{3.1}$$

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]

Example 1.

$$\begin{pmatrix} 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 0 & 255 & 255 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 0 & 255 & 255 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 0 & 255 & 255 & 255 & 255 & 255 & 255 & 0 & 255 \\ 255 & 255 & 0 & 255 & 255 & 255 & 255 & 0 & 255 & 255 \\ 255 & 255 & 255 & 0 & 0 & 0 & 0 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{pmatrix} \Rightarrow$$



3.1.2 Video

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

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

[1] Integers in the range $[0, 255]$.

3.2 Image Processing

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.

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

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

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

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

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

Definition 6. The image transformation ϕ takes an image I and returns a transformed image.

$$\begin{aligned} \phi : \mathcal{I} &\to \mathcal{I} \\ I &\mapsto \phi(I) \end{aligned} \tag{3.6}$$

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

Definition 7. We will use subtraction loosely[2] as elementwise subtraction and then subtract the smallest element on all elements to make the operation closed.

$$\begin{aligned} - : \mathcal{I} \times \mathcal{I} &\to \mathcal{I} \\ (I_a, I_b) &\mapsto I_a - I_b \end{aligned} \tag{3.7}$$

3.3 States, hypotheses and estimates

A system is said to have a *state*. The state is some quantity that defines the qualities of the system.

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, $Z \in \mathcal{Z}$.

[2]Meaning, no analysis on the structure is done.

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.

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

Chapter 4

Theory

4.1 Introduction

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) [7]. It builds upon the theory of *Markov processes* and the *hidden Markov model*.

4.2 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

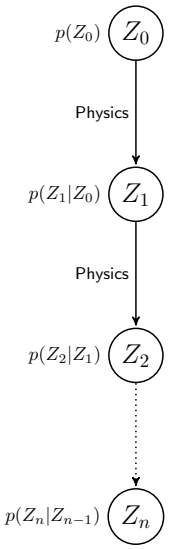


Figure 4.1. Schematic image of a Markov process.

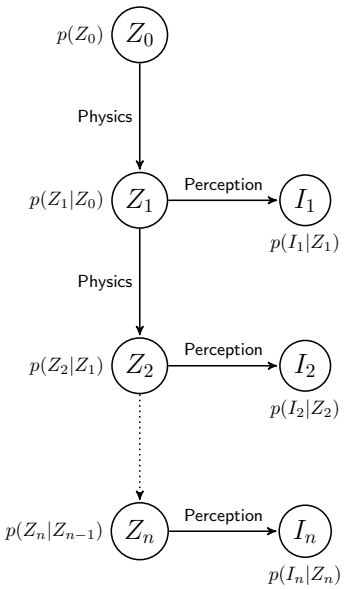


Figure 4.2. Schematic image of a hidden Markov model.

reason, a Markov process is often said to be "forgetful".

In mathematical terms, a Markov process satisfies the following:

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

$p\left(Z_t|Z_{t-1} \wedge Z_{t-2} \wedge \cdots \wedge Z_0\right)$ is the probability that the system will have state Z_t at time t, given that the previous states where $Z_{t-1}, Z_{t-2}, \ldots, Z_0$.

4.3 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.

4.4 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.

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

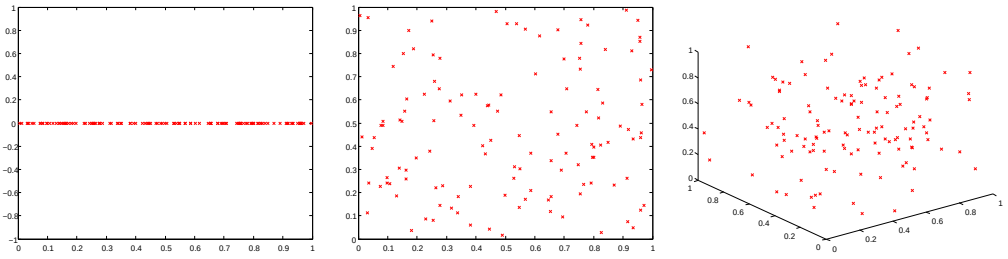


Figure 4.3. Plots of 128 scattered samples in 1, 2 and 3 dimensions, respectively.

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 with high dimensional space is the large "borders" of the sample-set compared to lower dimensional space which results in orders of magnitude higher chance for an point one want to approximate to fall outside the sample-set and needs to be extrapolated instead of the better alternative of interpolation.

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

Example 3. For a 16 DOF model one needs $10^{16} = 10$ quadrillion datapoints to acquire a density of 10 samples per unit volume. Millions of gigabytes would be needed just to store the samples.

Example 4. 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.

4.4.1 Overcoming the Curse

One way to overcome the curse in the context of tracking is to perform a directed search. Let the search be in an n dimensional space with a grid of g grid lines in each direction.

1. Use the information about the most recent[2] 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]In the Bayesian case, the most recent estimate

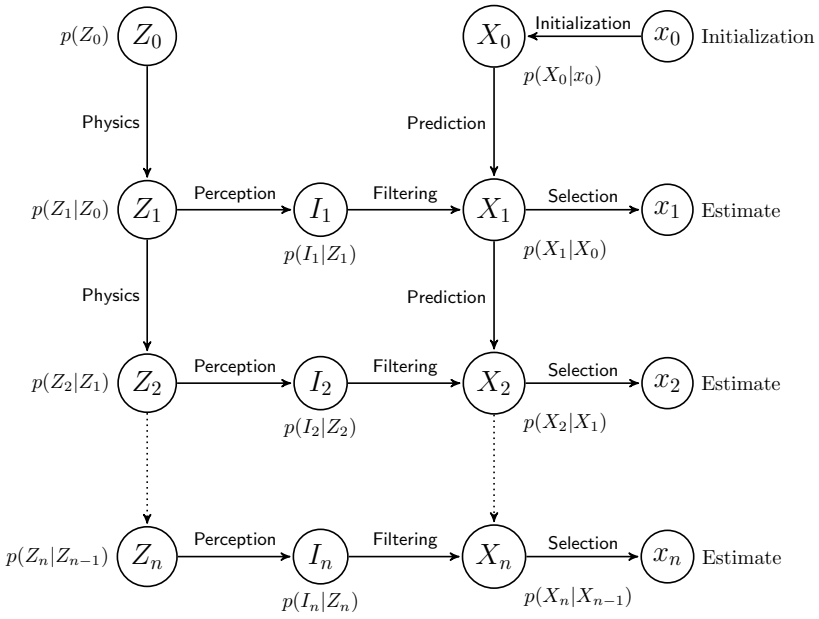


Figure 4.4. Schematic image of the particle filter alongside a HMM.

2. With prior knowledge of how the tracked objects move [3] 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.

4.5 The Particle Filter

One naive way to compute the state Z_t would be to perform an exhaustive search in the state space, and select the state for which $p(\, I_t|Z_t\,)$ is maximised.<ref:COD>

The particle filter is a technique for reducing the search space<ref:COD>. 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 refered 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 increses computational cost. Therefore the number of particles is an important trade-off. The particle filter employs a

[3] Such as the state transition probabilites $p(\, Z_t|Z_{t-1}\,)$ in a HMM

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 $\bar{X}_t$ of $p(Z_t|Z_{t-1})$. This is done by drawing new samples $p(x_t|x_{t-1})$, 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.

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 result will be a surjection, where X_t will be a subset of $\bar{X}_t$ where more probable hypotheses appear multiple times.<ref:problem without sigma> For this reason, this is also known as the *resampling* step. The set X_t is the *belief*, our approximation of $p(Z_t|Z_{t-1})$.

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 means 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 $p(x_{t-1}|I_{t-1} \wedge \cdots \wedge I_0)$, the PDF $p(x_t|I_t \wedge \cdots \wedge I_0)$ can be expressed as

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

where κ is a normalization constant

For the proof of this, see [6].

Let's sketch out the connection between this and what we've done so far. «THIS WILL BE EDITED.» For a more rigorous derivation, see chapter 4 of [7].

In this thesis, $p(x_t|I_t \wedge \cdots \wedge I_0)$ is approximated as the set X_t of hypotheses. Substituting this into (4.2) and replacing the integral with a sum yields

$$X_t = \kappa p(I_t|x_t) \sum p(x_t|x_{t-1}) X_{t-1}. \quad (4.3)$$

Note that this is horrible abuse of notation, but from here we can take the step to

$$X_t \sim p(I_t|x_t) p(x_t|x_{t-1}) \quad (4.4)$$

```
PARTICLE-FILTER(X_{t-1}, I_t)
 1  X̄_t ← ∅
 2  for each x_{t-1} ∈ X_{t-1}
 3       do
 4          x_t ← PREDICT(x_{t-1})
 5          w ← IMPORTANCE(x_t, I_t)
 6          Append ⟨x_t, w⟩ to X̄_t
 7
 8  X_t ← ∅
 9  while |X_t| < |X̄_t|
10       do
11          Take ⟨x_t, w⟩ from X̄_t with probability ∝ w
12          Append x_t to X_t
13  return X_t
```

Table 4.1. The particle filter algorithm.

4.5.1 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.

4.6 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.

A *visual cue* is an image transformation ϕ that extracts some property of the image, such as intensity [4], edges, ridges[6] or different refinements as in section 5.4.

4.7 Response

«intro»

[4] In our case this is possible since we have an homogenous object in the form of a backlit rodent

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

$$\underset{\bar{e}}{argmax} \left(\int_{\Omega} f(\bar{x}) f(\bar{x} - \bar{e}) \right) = 0 \tag{4.5}$$

Proof. Firstly define the window function as

$$\begin{aligned}
W_a^b(x) &= \begin{cases} 0, & x < a \\ 1, & a \le x \le b \\ 0, & x > b \end{cases} \\
\text{Multiplication:} \quad (W_a^b W_c^d)(x) &= W_{\max(a,c)}^{\min(b,d)}(x) \\
\text{Translation:} \quad W_a^b(x-e) &= W_{a+e}^{b+e}(x) \\
\text{Integration:} \quad \int_{\mathbb{R}} W_a^b(x)dx &= \Theta(b-a).
\end{aligned}$$

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

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. Additionaly, this also holds in higher finite dimensions since we can just repeat the process one dimension at a time.

All riemann functions can be written as a superposition of windows like this

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

and therefore (4.5) holds for any Riemann function f with finite support.

□

4.8 Model

A whisker can be modeled as a function:

Definition 8. Let

$$\begin{aligned}
\text{whisker} : \mathbb{R}^+ &\to \mathbb{R}^2 \\
\omega &\mapsto \text{whisker}(\omega)
\end{aligned} \tag{4.6}$$

where ω is a coordinate along the length of the whisker and whisker(ω) 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 5. 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 follows.

4.8.1 Simplifications

One simplification one might make is to assume that the root of the whisker is fixed in some point on the cheek. 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

$$\text{whisker}^*(0) = \bar{0}. \tag{4.7}$$

Manual inspection of whisker videos suggests that this is not the case for real whiskers. However, there seems to be some point within the cheek 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

$$d(\omega) = \begin{cases} D - \frac{D\omega}{L}, & \omega < L \\ 0, & \omega \geq L \end{cases} \tag{4.8}$$

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

4.8.2 Difference measure

The standard way to quantify distance between functions defined on an interval $[a, b]$ is to use the norm in the $\mathrm{L}^2(a, b)$ 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 $||\text{whisker} - \text{whisker}^*||_{\mathrm{L}^p}$ can be made arbitrarily small.

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

The first and simplest candidate is the polynomials. 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

$$(\omega, a_3\omega^3 + a_2\omega^2 + a_1\omega) \tag{4.9}$$

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.

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

Another promising candidate is the Fourier series. Considering (4.7), 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.

These define parameterized curves in the xy plane as

$$(\omega, \sum a_n \sin(\frac{2\pi n}{L}\omega)) \tag{4.10}$$

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. A classic separation of variables solution to the equation would be a sine series for the spatial part.

4.8.5 Theoretical evaluation

It is hard to analytically 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.

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 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.

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 program 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 $\langle a_1, a_2, a_3 \rangle$. 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 $\langle x_{\text{from}}, x_{\text{to}} \rangle$ 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 $w \leftarrow (||x_{\text{from}} - x_t||_{\text{L}^p})^{-a}$
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, with 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 $||x_{\text{from}} - x_{t-1}||_{\text{L}^p}$. A high a means closer transitions get a much higher weight than ones far away, see figure 5.1.

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 term[1] v to make the prediction nondeterministic.

[1]The 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.

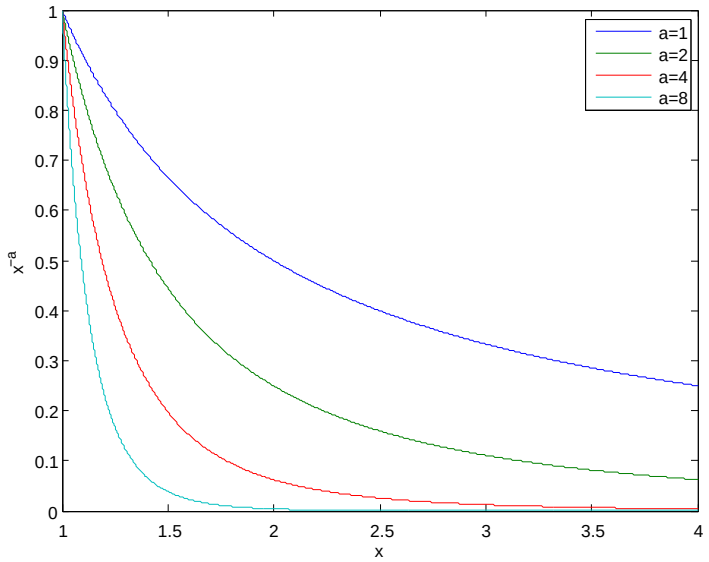


Figure 5.1. Transitions closer to x_{t-1} recieve a much greater weight if a is large.

5.3 Tracker

The tracker uses the whisker model described in chapter 4 and internally represents whiskers with the tuple $\langle a_3, a_2, a_1 \rangle$ of polynomial coefficients.

5.3.1 Filtering $p(I_t|x_t)$

IMPORTANCE(x_t, I_t)
1 $J \leftarrow \phi(R(x_t)) * \phi(I_t)$
2 **return** sum(J)

Table 5.2. Pseudocode for the importance function. p.

The IMPORTANCE function in table 4.1 is implemented as follows. Given a hypothesis x_t, the tracker renders an image $R(x_t)$ depicting the whisker shape represented by x_t. An example can be seen in figure 5.2. The image $R(x_t)$ is then multiplied with I_t. This has the result of "masking" the image I_t, extracting the pixels that the whisker would occupy if its shape were given by x_t. The importance

is then calculated as the sum of the pixels of $R(x_t) * I_t$, meaning the importance is high if the image pixels along x_t are bright.

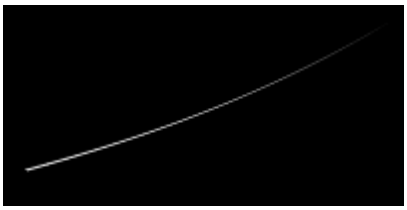


Figure 5.2. Example rendered image $R(x_t)$ for some hypothesis x_t.

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. The filtering function described in the previous section expects whisker pixels to be bright and the body has the same pixel values as the whiskers, so the images had to be inverted.

5.4.1 Subtracting background

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

Assuming we always have a static camera setup foreach sequence and that we have a adiquit sequence without the rodent. The background was captured a priori by taking a mean4 of the 100^2 first frames BG

$$I_{BG} = \frac{\sum_{bg \in BG} I_{bg}}{\#\{BG\}}. \tag{5.1}$$

«image showing the mean background»

Then we can simply subtract7 the background I_{BG} from the orignal image I [3]

$$I_{FG} = I - I_{BG} \tag{5.2}$$

«example with the 3 images involved above alongside eachother»

5.4.2 Finding the body p(body)

Finding the body has two purposes, firstly to find the head-coordinate system and secondly to subtract the body from the image to find whiskers.

[2] Just a sufficient number

[3] Worth noting is that more sofisticated methods exists to subtract the background, but this solution works relatively well considering the simplicity.

Definition 9.

$$p(\text{body}) \in \mathcal{I} \tag{5.3}$$

Is an estimation[4] on the probablity of the pixel to be a householding a rodent body.

Assuming that the only non-static objects in the image is the whiskers and body. One simple feature that easily classifies $p(\text{whisker})$ from $p(\text{body})$ is size. Bluring the image removes fine structures and retains larger ones. This was performed by convoluting the image with a gaussian with $\sigma = 9\text{px}$[5]

$$p(\text{body}) = I_{blur} = I_{FG} \star \text{gauss}(0, \sigma) \tag{5.4}$$

«example of the blur showing that the whiskers has almost disapeared» Which then was made to a mask by

$$\text{body} = p(\text{body}) > 0.6 \in \mathcal{I}.^{6} \tag{5.5}$$

representing the location of the body. «example showing p(body) och body alongside each other»

5.4.3 Estimating $p(whisker)$

Getting an indicator on the location of whiskers is needed for the filtering.5.3.1

Definition 10.

$$p(whisker) \in \mathcal{I} \tag{5.6}$$

Is an estimation[7] on the probablity of the pixel to be a householding a rodent whisker.

This is naïvly performed by

$$p(\text{whisker}) = I_{FG} * (1 - \text{body}) \tag{5.7}$$

«example showing the above»

5.4.4 Estimate snout translation

Firstly we hand pick the first frame where the snout is fully visible and use this frame as reference to estimate the translation. The mask is then again blured out with a $\sigma = 5\text{px}$[8] and filtered through a prewittfilter

$$\text{snout}_{ref} = \sqrt{(\text{snout} \star \text{prewitt}_x)^2 + (\text{snout} \star \text{prewitt}_y)^2} \in \mathcal{I} \tag{5.8}$$

[4] Only needs to be an indication on wether it is a body
[5] The value for σ was obtained by manual testing.
[6] 0.6 found to be a value making the body stable between frames
[7] Only needs to be an indication on wether it is a whisker
[8] Obtained by manual testing.

to smooth out the response of the edges.

Then a local search within 5px[9] from the last location was performed foreach image in the sequence

$$\text{translation} = \underset{(x,y)\in\text{close}}{argmax}\,(\sum \text{translate}(\text{snout}_{ref}, -(x,y)) * \text{snout}) \in \mathbb{Z}^2 \tag{5.9}$$

where snout is the current image under the same filter as snout_{ref}

5.4.5 Fixate the head programmatically

As a last touch we transform the video to fixate the head coordinates to simplify the tracking analysis, this will not effect the final result one could simply pass the moving coordinate system directly to the tracker.

[9]This could easily be extended to include rotation.

Chapter 6

Results

<choose the word:(performance vs fitness)>
The bake-off <ref> setup for the experimental parameter evalution<ref>:
Evaluated parameters:

p The norm (where)

a Penelty for the norm

σ Resample blur (?)

ϕ Transformation

N Number of transitions

n Particle count

will be tested on a small set of benchmarks consisting of 4 disperse(d?)/scatter generated whisker videos.
The evaluation tensor:

$$\Phi_{p,a,N,...}(benchmark) \tag{6.1}$$

TODO (check the comments here)
The measures that will be used as fitness of the parameters:

$\int ||\epsilon(t)||_{L^p} dt$ integrating over time the the difference with the ground truth (do this for L1:10 and see if it correlates with the p choosed (to see how much it deviates from the ground truth)

$\int \sum \phi R(x_t) \phi I_t dt$ integrating over time the response for the choosed hypothesis (to see how the different image transformation affects the results, that is if it only follows what it thinks is best (phi))

Subjective 4 image samples

And all this are done for all 4 benchmark videos.

"There are many metrics by which a model may be assessed." - Encyclopedia

The fitness test was runned on different machines but this wont effect the result since we initially wont consider the running time.

The runtime for the algorithm is handled separately on one machine setup <...>

Tillvägagångsätt:

1. Since we have prior knowledge about the effect off varying the parameters n,N they will firstly be set to a sufficently large value. 2. A partion of the test-matrix will then be evaluated by ... parameter group

Chapter 7

Analysis and Discussion

One must be carefull doing this type of experimental analysis on generate whiskers and then applying the conclusions on real whiskers, but we could still use the best parameters as a starting point for further studies on real whiskers.

" However, much machine learning research includes experimental studies in which algo- rithms 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 experi- mental evaluation has become known disparagingly as a bake-off . " - Encyclopedia (>algorithm evaluation)

"A learning algorithm must interpolate appropriate predictions for regions of the instance space that are not included in the training data." - Encyclopedia (>model evaluation)

Can a particlefilter overfit?(should this perhaps be in theory?)

Bibliography

[1] Rafael C. Gonzlez and Richard E. Woods. *Digital Image Processing, Third Edition*. Pearson Education, 2008.

[2] Hans Lundh. *Grundläggande Hållfasthetslära*. Instutionen 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 Hogskolan, 2001.

[7] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2006.

[8] Jakob Voigts, Bert Sakmann, and Tansu Celikel. Unsupervised whisker tracking in unrestrained behaving animals. *Journal of Neurophysiology*, 100:504–515, 2008.