Probabilistic Tracking of Multiple Rodent Whiskers In Monocular Video Sequences

Jim Holmström, Emil Lundberg
Bachelor's Thesis at CSC, KTH

Background

The Problem

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[1], or fail in cluttered environments or when whiskers occlude each other. A cheap, reliable solution to the tracking problem is needed.

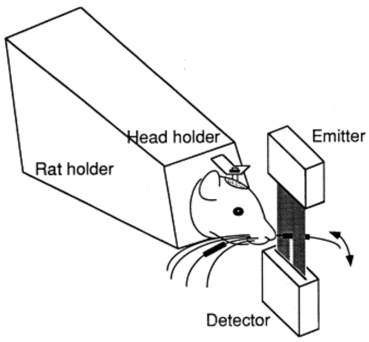


Figure 1: A very restrained experiment setup.

A Probabilistic Approach

We propose solving the problem by a probabilistic approach. We use 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 searching a pre-trained database, and filtering the results through the Particle Filter. The main

[1] A method known as *optoelectronic monitoring* takes this to the extreme by having the rat locked in place by cranium-mounted screws [1]. See figure 1.

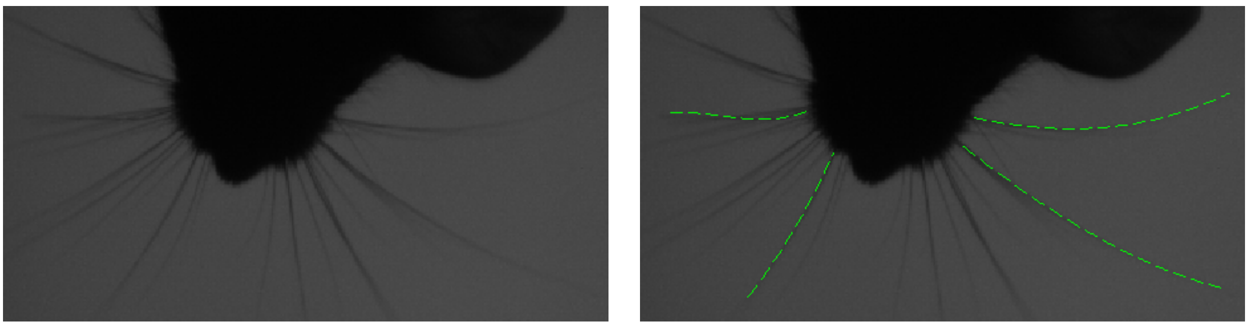


Figure 2: Left: Example image of a rat and its whiskers. Right: Least squares fitted third degree polynomials.

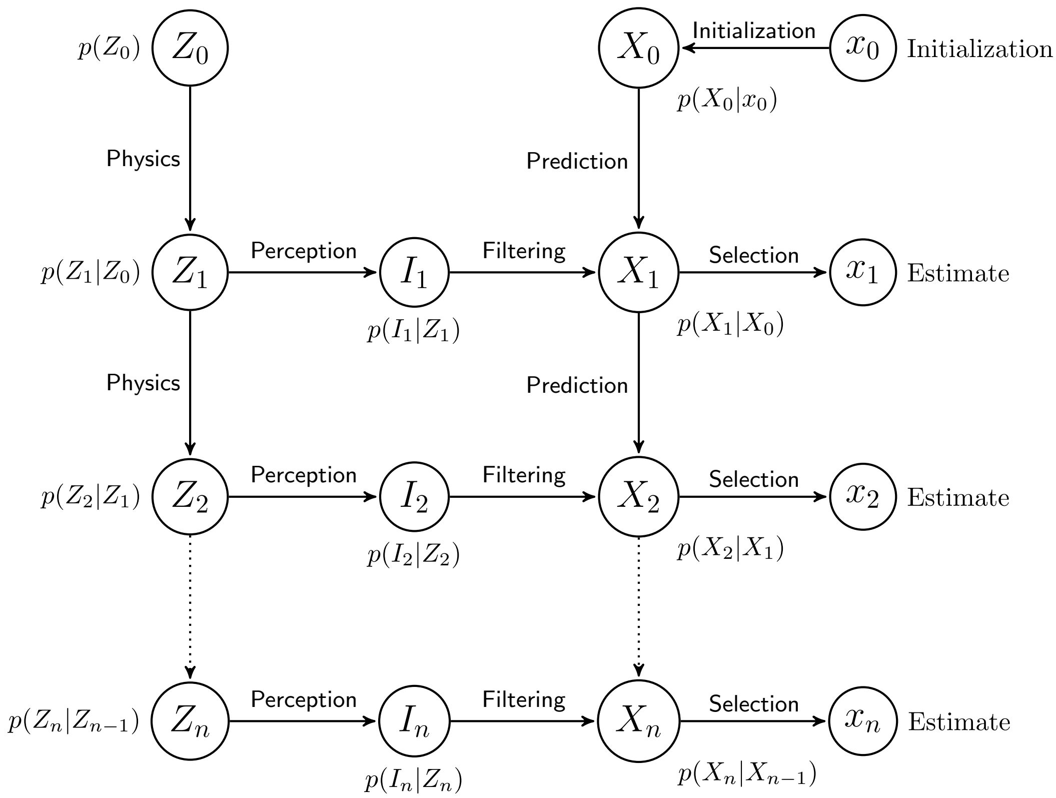


Figure 3: Schematic image of the Particle Filter working alongside a Hidden Markov Model.

difference between this and existing solutions is that it maintains a model of the whiskers. This makes it easier to keep track of them even when they cross or overlap.

The Probabilistic Framework

Our solution is based on discrete *Markov processes*, which are a special case of stochastic processes. For a Markov process, the next state depends only on the present state and not on past states.

An example of a discrete Markov process is that of throwing dice and summing the results: the throws and state space are discrete, and the possible states (sums) after the next throw depends only on the current state.

In mathematical terms, a discrete Markov process satisfies the following:

$$p\left(Z_{n+1}|Z_n \wedge Z_{n-1} \wedge Z_{n-2} \wedge \cdots \wedge Z_0\right) = p\left(Z_{n+1}|Z_n\right), \tag{1}$$

where Z_n is the system's *state* after step n and $p\left(Z_{n+1}|Z_n, Z_{n-1}, Z_{n-2}, \ldots, Z_0\right)$ is the probability that the system will have state Z_{n+1} in the next step, given that the previous states where $Z_n, Z_{n-1}, Z_{n-2}, \ldots, Z_0$.

A *hidden Markov model* (HMM) describes a Markov process where one cannot measure the state Z of the system directly - it is "hidden"[2]. Instead we obtain an *observation* I[2] of the state. This *perception* is generally non-deterministic, so we need to denote it as $p(I_n|Z_n)$ which is the probability that we will observe I_n if the current state of the system is Z_n.

The Particle Filter

The Particle Filter is a technique for simulating a process described by a HMM. It uses a finite set X_{n+1} of hypotheses to approximate the probability function $p\left(Z_{n+1}|Z_n\right)$. The hypotheses X_n are also known as *particles*, thereby the term "particle filter".

Figure 3 shows the working principle of the Particle Filter working alongside a Hidden Markov Model. The following is the core function of the Particle Filter:

[2] In this project, the observation is a grayscale *image*, which is why we use the symbol I.

The particle filter attempts to approximate the probability density function $p(Z_{n+1}|Z_n)$ as a set X_{n+1} of discrete hypotheses.

More particles mean greater accuracy, since the PDF can then be approximated more closely. However, using many particles is also computationally expensive. Therefore the number of particles is an important quantity. The Particle Filter employs a few tricks to *filter* the hypotheses, keeping probable ones and throwing improbable ones away, in order to reduce the number of particles needed for a good approximation. The following is a quick run-down of how the Particle Filter does this. For brevity, we will commit some abuse of notation.

Initialization: Since the algorithm only does tracking, we need to initialize the algorithm with a start guess x_0. Using this we take a number of samples $X_0 \sim \mathcal{N}(x_0, \Sigma)$ and let the set X_0 be an approximation of $p(Z_0)$ [3].

Prediction: The hypotheses X_n are updated in the *prediction* step to an approximation of $p(Z_{n+1}|Z_n)_{n+1}$. This is done by drawing new samples $\bar{X}_{n+1} \sim p(X_{n+1}|X_n)$.

Perception: By measuring the state of the system, we gain an *observation* $I_{n+1} \sim p(I_{n+1}|Z_{n+1})$ of the state Z_{n+1}.

Filtering: The observation I_{n+1} of the system is then used for filtering bad hypotheses out of $\bar{X}_{n+1}$. We draw samples X_{n+1} from $\bar{X}_{n+1}$ with probabilities given by $p(I_{n+1}|\bar{X}_{n+1})$. As a result, X_{n+1} will be a subset of $\bar{X}_{n+1}$ where more probable hypotheses appear multiple times. For this reason, this is also known as the *resampling* step. The set X_{n+1} is the *belief*, our approximation of $p(Z_{n+1}|Z_n)$.

Selection: Finally, we produce a single hypothesis x_{n+1} from X_{n+1} as our *estimate* of the state Z_{n+1}. Supposing X_{n+1} is a good approximation of $p(Z_{n+1}|Z_n)$, and that $p(Z_{n+1}|Z_n)$ is unimodal, the mean value of X_{n+1} is a good hypothesis since it approximates the expectation of $p(Z_{n+1}|Z_n)$.

A Simple Whisker Model

Our simplest model of a whiskers is n-th degree polynomial curve attached at $x = 0$ to a fixed point in space. This means our state parameters are the coefficients $\{a_i\}_{i=0}^{n}$ of the polynomial $\sum_{i=0}^{n} a_i x^i$, and we can represent a whisker's state with the tuple of its coefficients. In this model we implicitly assume that the rat's head movements do not greatly affect the whiskers' movement. Possible improvements to this model may include:

- letting the whisker attach to a fixed point in a moving coordinate system (the "head system"),
- using some other function basis, such as a sine series.

So far we have only investigated the simplest polynomial model. Least squares fitting tests performed using MATLAB show that a third degree polynomial can represent any whisker in figure 2 with an error that is barely visible to the naked eye. Therefore a third degree polynomial was used as a first step. We omit the constant term since this information can instead be included in the position of the whisker base.

Our implementation of the Particle Filter

An implementation of the Particle Filter consists mainly of designing the probability functions $p(X_{n+1}|X_n)$ and $p(I_n|X_n)$, and providing the algorithm with a sensible initialization. This is what this project is all about. The rest just consists of taking samples from these functions.

The prediction step: The Database

We investigate the plausibility of implementging $p(X_{n+1}|X_n)$ as a search through a database of training data. We set up a database of known transitions between whisker shapes. A transition T consists of a "from" state f and a"to" state t. This denotes that "we have observed with 100% accuracy that a whisker went from this shape to that shape in one time step". We then approximate $p(X_{n+1}|X_n)$ as a weighted average of the database, where transitions are weighted by how much their "from" parts differ from the hypotheses in X_n.

What we do in practice when sampling is: for each hypothesis $x_n^i \in X_n$,

[3] The problem of initialization is a tricky one[3], and is not covered in this project. In our testing implementation, we used $\Sigma = 0$

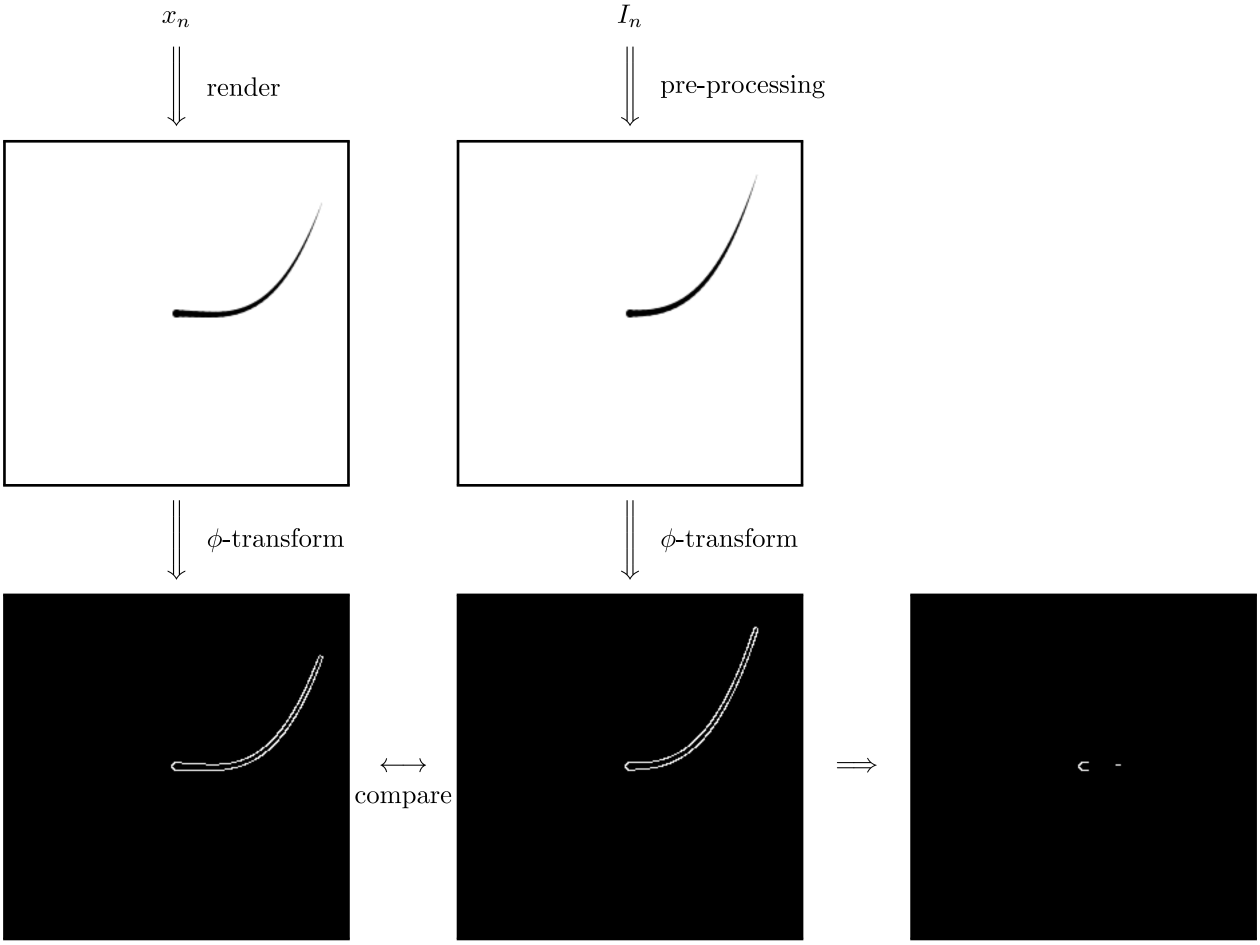


Figure 4: Schematic image of the process to evaluate the importance of a hypothesis.

1. For each transition $T^j = (f^j, t^j)$ in the database, calculate the function d^{ij} that is the difference between the two functions described by x_n^i and f^j. In this case, both are polynomials and thus d^{ij} is the polynomial with coefficients given by the tuple $x_n^i - f^i$.
2. Let $w^{ij} = \left(\frac{1}{\|d^{ij}\|_{L^2}}\right)^a$, the reciprocal of the L^2 norm of d^{ij} raised to a power a.
3. Return $\frac{\sum_j t^j w^{ij}}{\sum_j w^{ij}}$, the weighted average of the "to" states with weights w^{ij}.

Doing this for each hypothesis x_n^i yields the set $\bar{X}_n$. We have not yet thoroughly investigated how the power a qualitatively affects the tracking results, but $a = 4$ seems to be a good value.

The filtering step: Image comparison

We implement the probability function $p(I_n|X_n)$ as simply checking if the image pixels beneath the whiskers X_n look like whisker pixels. The image I_n is grayscale, and so each pixel can be identified with a number between 0 and 255, inclusive. We assume that whisker pixels have a high value and background pixels have a low value.

We first preprocess the image using a transformation ϕ, to make the measurement easier. At the current stage in our testing, the images used are generated synthetic ones that are already easy to process. Therefore we let ϕ be the identity transformation at the moment.

After preprocessing the image, for each hypothesis $\bar{x}_n^i \in \bar{X}_n$ we create a *mask* image I_n^i. This image will be zero everywhere except where the whisker $\bar{x}_n^i$ is. We then let $w^i = \sum_{\text{pixels}} I_n \cdot I_n^i$, where the multiplication is done component-wise. We then let $\left\{(\bar{x}_n^i, w^i)\right\}_{i=1}^N$ define a discrete probability function that returns $\bar{x}_n^i$ with probability w^i, and let this distribution be our approximation of $p(I_N|X_n)$.

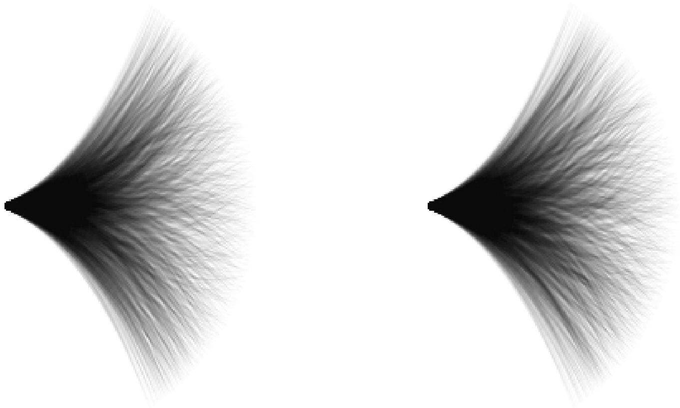


Figure 5: View of all whiskers in transition database. Left: from-states. Right: to-states.

Example tracking image

Figure 6 shows an illustration of the three tracking steps. The blue lines are the hypotheses $\bar{X}_n$ sampled from the database. The red lines are these same hypotheses, but after resampling, X_n. The green line is the estimate x_n, the mean of X_n. One can see how X_n is slightly more concentrated around the tracked whisker (white) than $\bar{X}_n$ is.

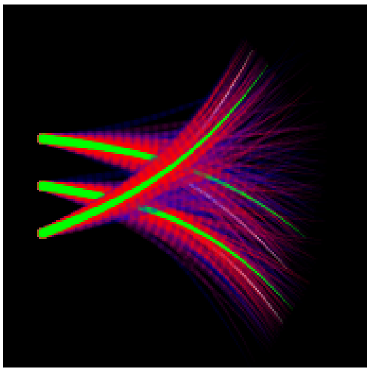


Figure 6: Tracking image with $\bar{X}_n$ (blue) and X_n (blue) drawn along with the estimate x_n (green).

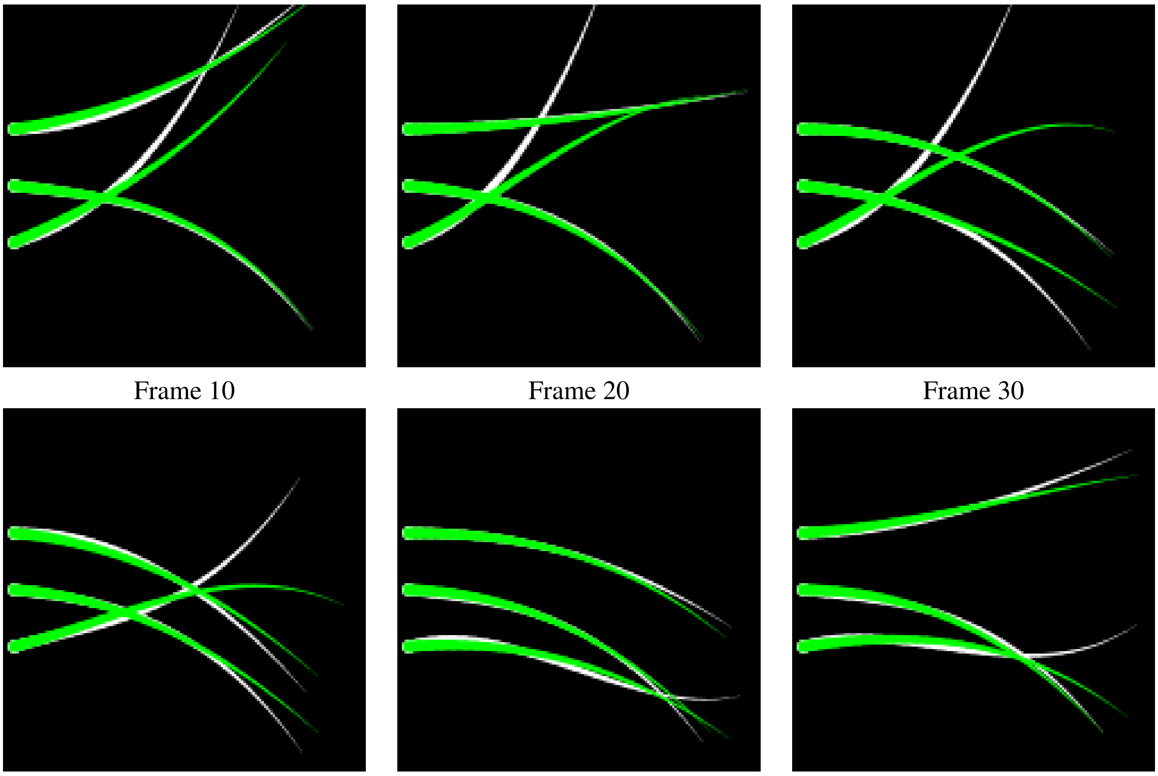
Results

So far, we have run some tests on randomly generated video sequences of whisker-like objects. While the results are far from good enough for practical use, they are still quite promising.

Figure 7 shows the tracking result on a sequence of 64 frames of generated, synthetic whiskers. The estimate of the whiskers' positions are at times close to perfect, but the tracking of the bottom whisker fails most of the time. Still, this illustrates the power of the probabilistic approach since this was run using only 32 particles, which is a very low amount. A 2.83 GHz Intel® Core™ 2 Quad computer system with 3.8 GiB of RAM running Ubuntu 10.04 running the tracker with 128 particles takes a little more than 4 seconds per frame and whisker, running on a single core.

Conclusions

Our results so far lead us to believe that that it is indeed feasible to use this method for tracking whiskers.



Frame 10 Frame 20 Frame 30

Frame 40 Frame 50 Frame 60

Figure 7: Tracking result using 32 particles on 64 frames of generated whiskers. White is the whisker being tracked, green is the estimate of its position.

References

[1] Bermejo R, D Houben, and H.Philip Zeigler. Optoelectronic monitoring of individual whisker movements in rats. *Journal of Neuroscience Methods*, 83, 1998.

[2] Claude Sammut and Geoffrey Webb. *Encyclopedia of Machine Learning*. Springer, 2010 (First edition).

[3] Hedvig Sidenbladh. *Probabilistic tracking of 3D human motion in monocular video sequences*. PhD thesis, Kungliga Tekniska Hogskolan, 2001.