

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, or fail when whiskers cross or overlap. A cheap, reliable solution to the tracking problem is needed.

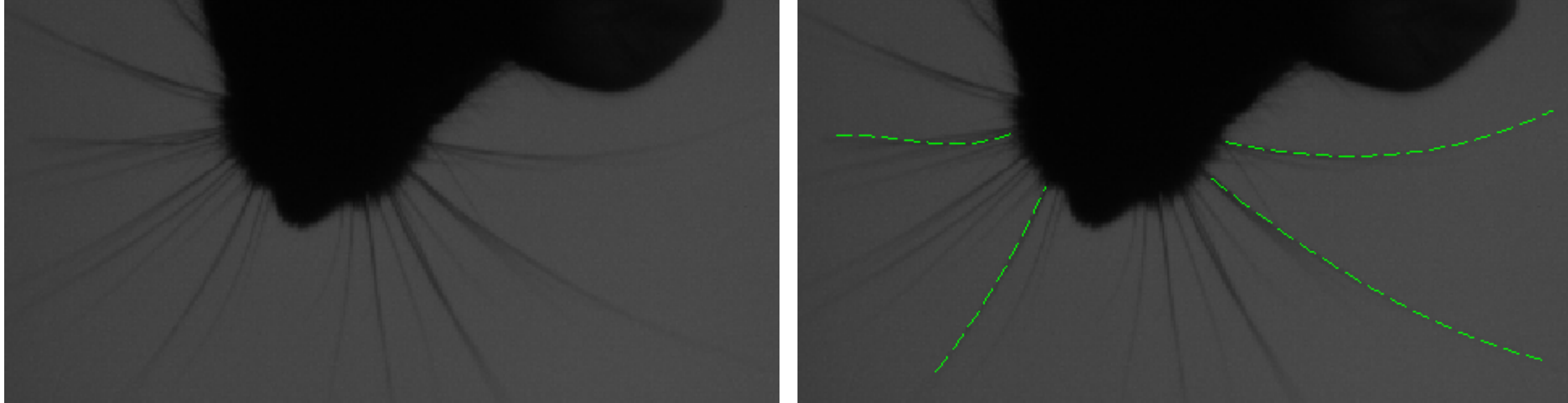


Figure 1:
Left: Example image of a rat and its whiskers.
Right: Least squares fitted third degree polynomials.

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 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:

$$p(Z_{n+1}|Z_n \wedge Z_{n-1} \wedge Z_{n-2} \wedge \dots \wedge Z_0) = p(Z_{n+1}|Z_n), \quad (1)$$

where Z_n is the system's *state* after step n and $p(Z_{n+1}|Z_n, Z_{n-1}, Z_{n-2}, \dots, Z_0)$ is the probability that the system will have state Z_{n+1} in the next step, given that the previous states were $Z_n, Z_{n-1}, Z_{n-2}, \dots, Z_0$. A *hidden Markov model* (HMM) describes a Markov process where one cannot measure the state Z of the system directly - it is "hidden". Instead we obtain an *observation* I^a 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 $p(Z_{n+1}|Z_n)$. The hypotheses X_n are also known as *particles*, thereby the term "particle filter".

^aIn this project, the observation is a grayscale *image*, which is why we use the symbol I .

^b Σ is roughly estimated from the database

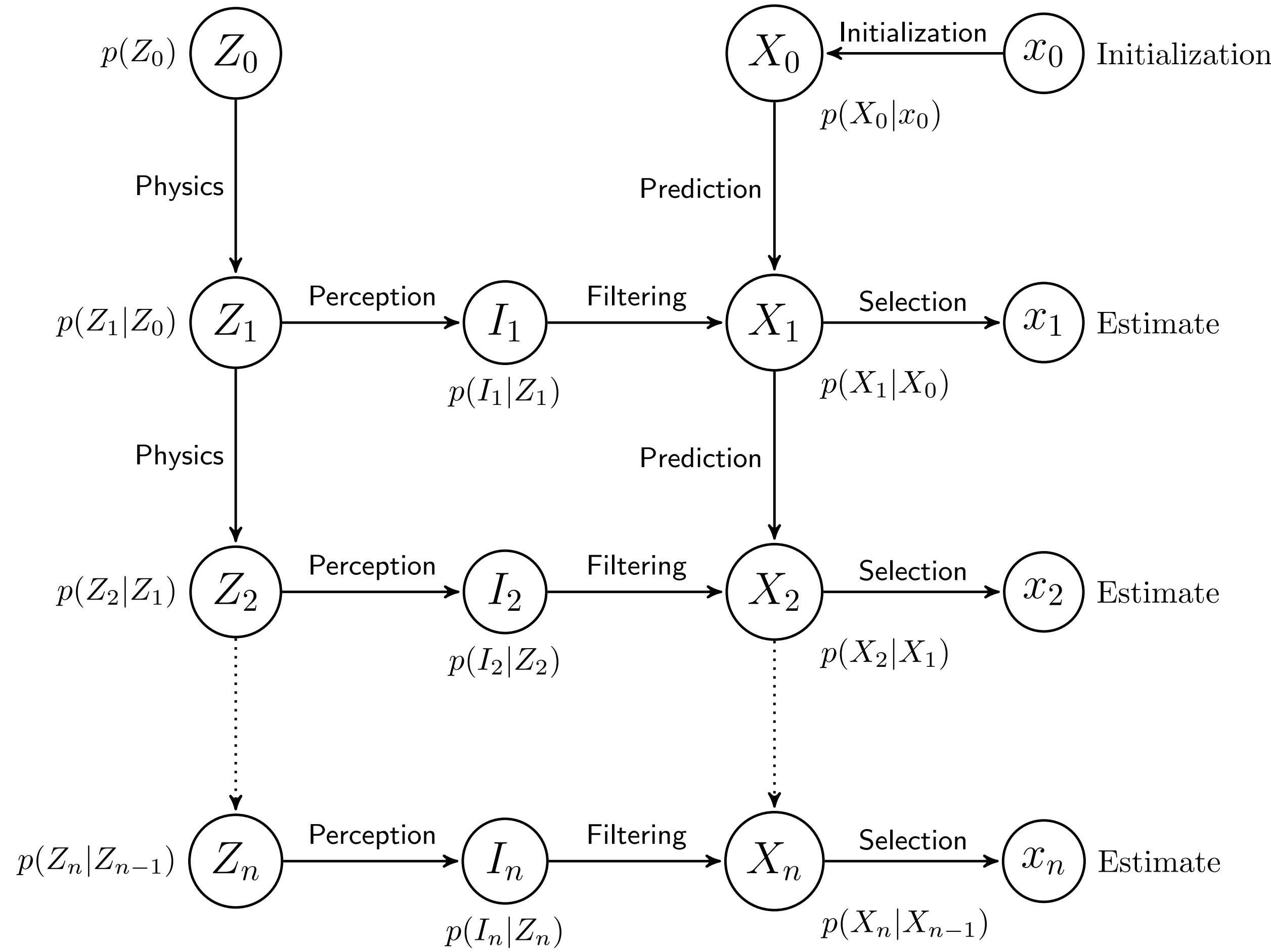


Figure 2: Schematic image of the Particle Filter working alongside a Hidden Markov Model.

Figure 2 shows the working 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 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 $X_0 \sim \mathcal{N}(x_0, \Sigma)^b$ and let the set X_0 be an approximation of $p(Z_0)$.

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 $\tilde{X}_{n+1} \sim p(X_{n+1}|X_n)$.

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 \tilde{X}_{n+1} . We draw samples X_{n+1} from \tilde{X}_{n+1} with probabilities given by $p(I_{n+1}|\tilde{X}_{n+1})$. As a result, X_{n+1} will be a subset of \tilde{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 $p(Z_{n+1}|Z_n)$.

Implementing the Particle Filter

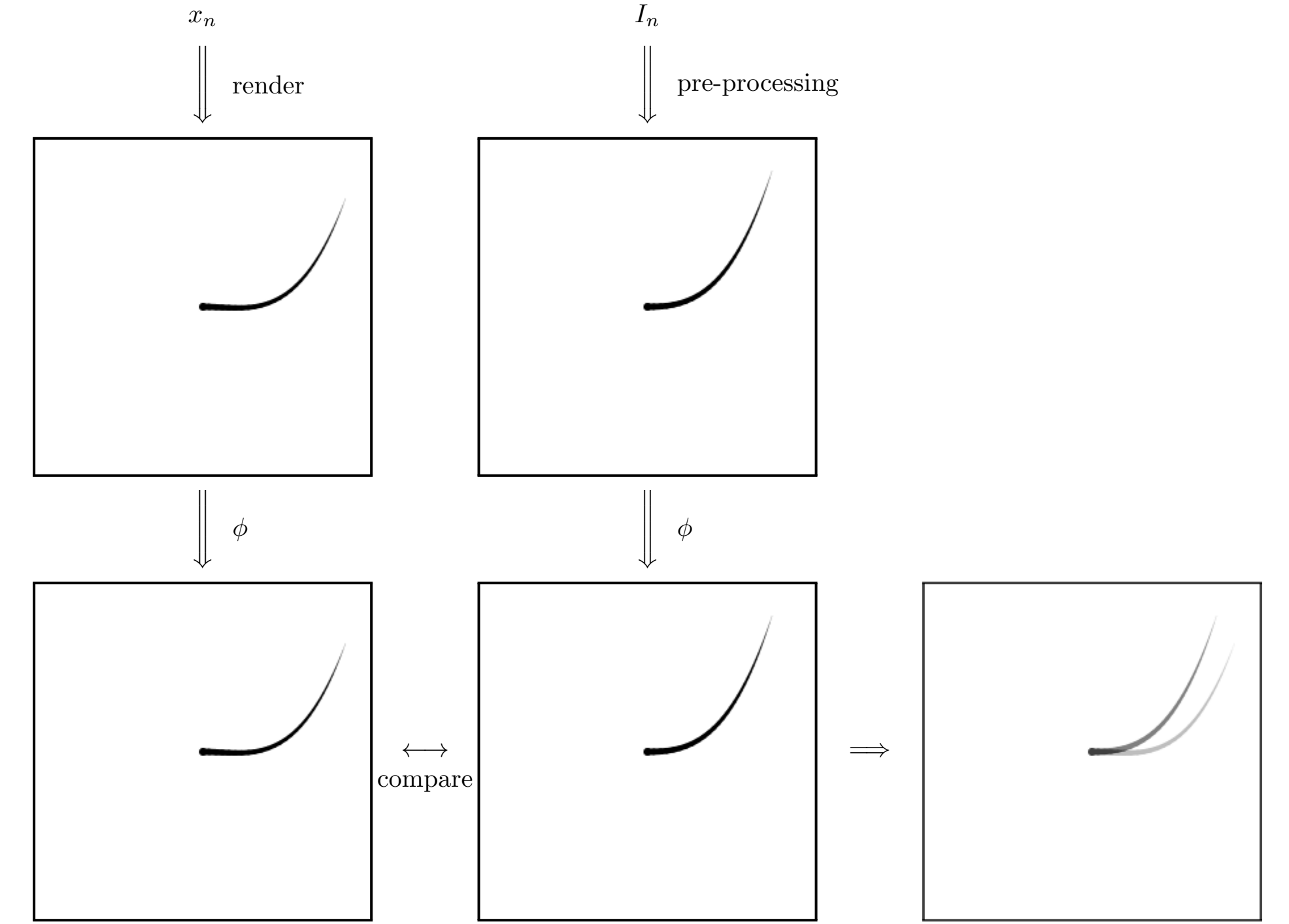
An implementation of the Particle Filter consists mainly of designing the probability functions $p(X_{n+1}|X_n)$ and $p(I_n|Z_n)$, and providing the algorithm with a sensible initialization. This is what this project is all about. The rest just consists of taking samples from these functions.

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 $a_{i=0}^n$ of the polynomial $\sum_{i=0}^n a_i x^i$. 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 1 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.



The Database

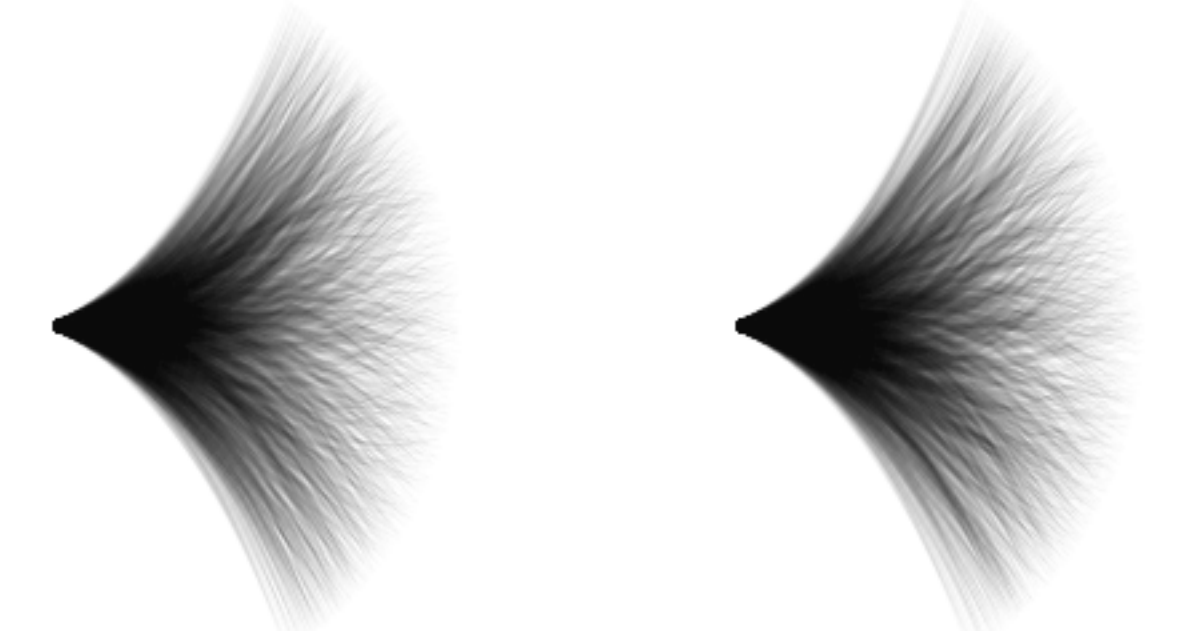


Figure 3: View of all whiskers in transition database, from-states on the left and to-states on the right.

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.

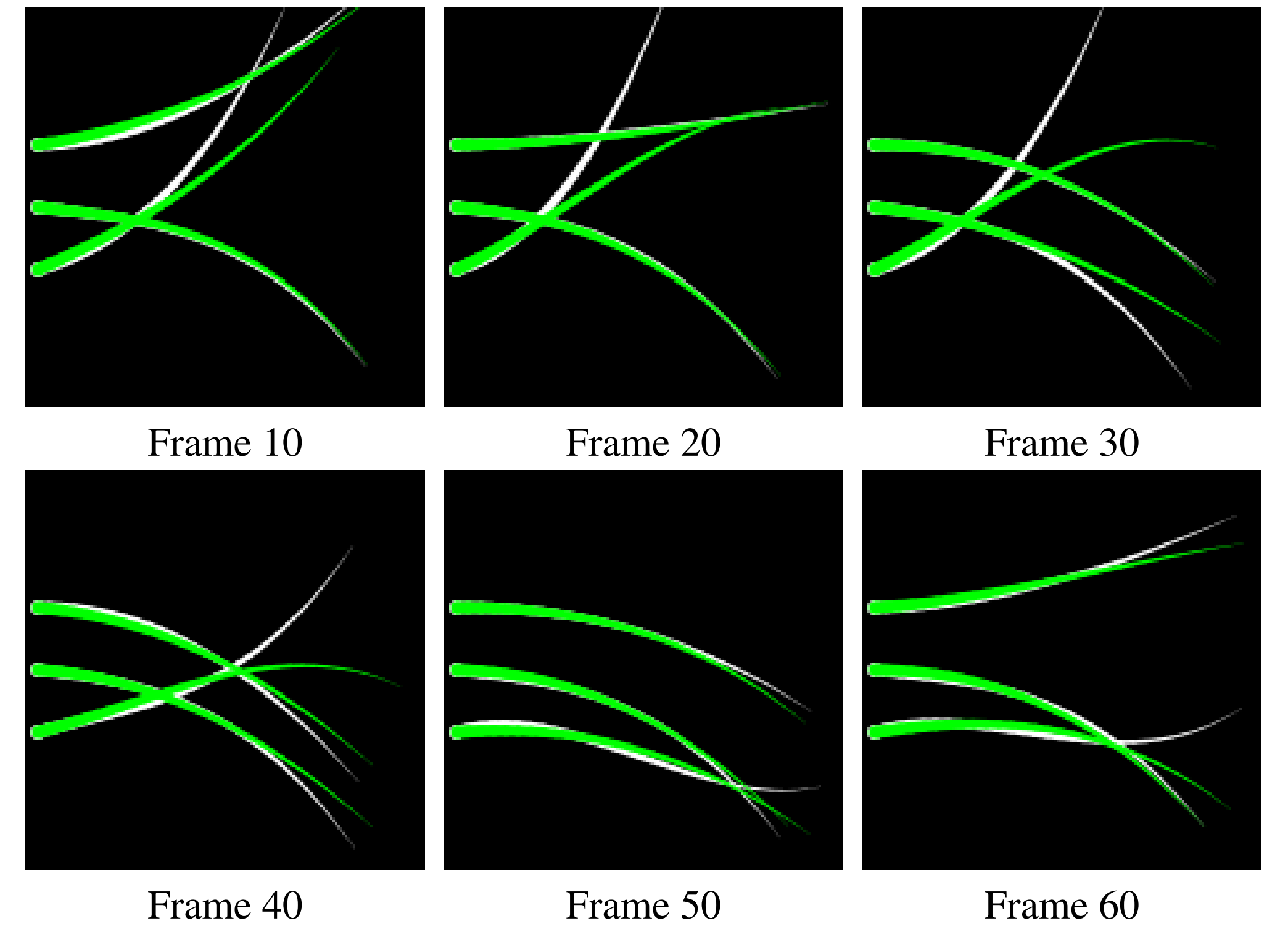


Table 1: Tracking result using 32 particles on 64 frames of generated whiskers. White is the whisker being tracked, green is the estimate of its position.

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