A Study of the Kalman Filter applied to Visual Tracking

Nathan Funk
University of Alberta
Project for CMPUT 652

December 7, 2003

Abstract

This project analyzes the applicability of the Kalman filter as a probabilistic prediction method to visual tracking. The problems encountered when not using predictions are identified, such as the lack of robustness towards noisy motions and measurements.

A literature survey of the subject is provided, which points out some of the common problems associated with using the Kalman filter and indicates how they can be approached. One problem identified in the literature is the requirement for prior knowledge about the process and the measurement procedure. This problem is confirmed by the experiments. Specifically, the values of the process and measurement covariance matrices are needed, yet difficult to obtain in most cases. Iterative methods are proposed to estimate these matrices, however they fail to provide useable results in the cases tested. From this, it is concluded that either more prior knowledge is required or better techniques for determining these matrices need to be developed.

Most tests are performed on synthetic images with known motion patterns (constant velocity and constant acceleration) and specified noise levels. This allows a clear comparison of the prediction methods implemented.

1 Introduction

The motivation for this project was initiated by the desire for robust methods for visual tracking. The implementation is based on an assignment I was working on for the Computer Vision course (CMPUT 615) this term. Although prediction methods including the Kalman filter were implemented as part of this assignment, my understanding of the Kalman filter was lacking. As a consequence, the implementation was rough and contained bugs.

The goal of this project is to study how the Kalman filter can be used as a probabilistic prediction technique to make tracking more robust. Instead of simply guessing the values of some of the parameters, an understanding of how these parameters can be determined and what effect they have on the results, is desired. This includes major revisions of the

existing code and the addition of image generation functions to properly study how the filter operates under well-defined conditions.

This report provides background information about the problems associated with visual tracking and discusses the various techniques proposed in literature. Although not a lot of documents were found that study the applicability of Kalman filtering to visual tracking methods, similar studies of how the Kalman filter is applied to navigation problems are discussed. The associations between the hidden Markov model and the Kalman filter are also studied. Finally, a full analysis of the application of the filter to visual tracking is provided including three hypotheses and multiple experiments.

2 Background

This section will introduce the basic concepts behind using the Kalman filter in assisting visual tracking. There are multiple problems that occur when a too simplistic tracking method is employed. These problems, and a brief discussion of how they might be solved, is given in the first part of this section.

There have been many papers and books written on the topic of Kalman filtering since it was first developed in the early 1960s. One of the first applications was in aerospace navigation systems. But since then, the number of different uses has increased dramatically. Its wide range of applications is one of the reasons why so many publications are available. The literature survey of this project focuses on the papers most relevant to the fields of tracking and navigation.

The last part of this section discusses the relationship between the hidden Markov model (HMM) and the Kalman filter. According to Jordan [5] this relationship has not been commonly appreciated since the research was conducted by separate communities. In addition, the general graphical models that can be applied to both the HMM and the Kalman filter came later than both of the individual techniques.

2.1 Visual Tracking and the associated Problems

Visual tracking can be described as the process of determining the location of a feature in an image sequence over time. Examples include tracking cars in an intersection via a traffic camera, or tracking the head of a computer user with a web-cam. Another possible application is tracking multiple small features of interest, such as corners of an object, in attempt to determine its 3-dimensional geometry.

The sequence of images can either be processed in real-time, coming directly from a video camera for example, or it can be performed on a recorded set of images. The implementation of this project uses recorded image sequences although the theory can be applied to both types of applications.

The *target* to be tracked might be a complete object (e.g. a person) or a small area on an object (e.g. a corner). In either case, the feature of interest is typically contained within a target region. This project will consider a rectangular target region of arbitrary size. The position will be described in X-Y coordinates in pixel units on the image (i.e.

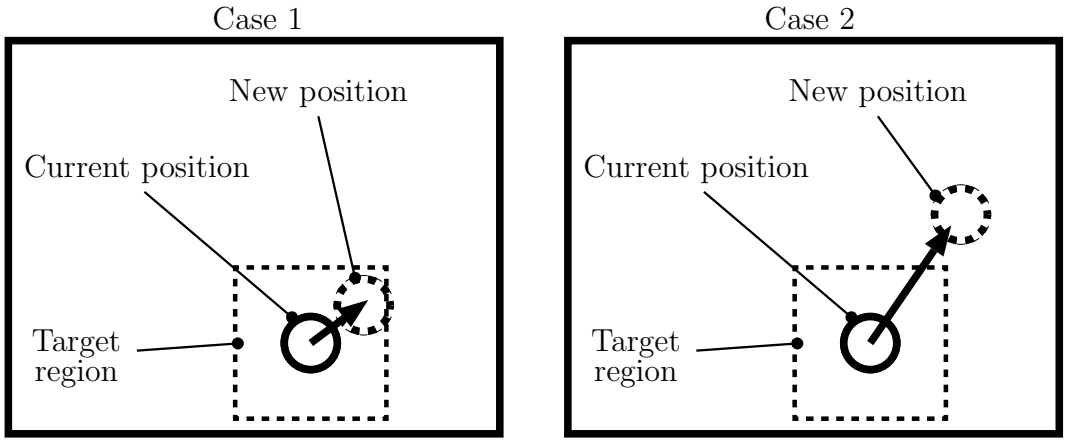

Figure 1: Case 1: Tracking the object without position prediction might be successful, Case 2: Tracking without position prediction will fail

image coordinates). Typically, when tracking an object in a real world scene, it might also be useful to describe its position in *world coordinates*. However for the purpose of this project, image coordinates will suffice.

Ideally, a tracking algorithm would be able to locate the object anywhere within the image at any point in time. However typically only a limited region of the image is searched (usually the same size as the target region). Reasons for this are efficiency (especially necessary for real-time applications) and the fact that there might be many other similar-looking objects in the image.

The intuitive approach is to search within a region centered around the last position of the object. But as Figure 1 illustrates, this approach will fail if the object moves outside the target range. There are many possible reasons why the object might not stay within this region:

- The object is moving too fast.
- The frame rate is too low.
- The searched region is too small.

These problems are related to each other and could be avoided by ensuring a high enough frame rate for example. But given other constraints, these problems are often unavoidable.

In addition, even when the target can be located, it seldomly appears the same in all images. The appearance of the same target is continuously affected by changes in orientation, lighting, occlusions, and imperfections in the camera. So essentially, the true location of the target is very difficult to observe accurately under the usual circumstances.

In summary, two major problems have been identified:

1. The object can only be tracked if it does not move beyond the searched region.

2. Various factors such as lighting and occlusions can affect the appearance of the target, thus making accurate tracking difficult.

To solve the first problem, we can attempt predicting the location of the target, and searching in a region centered around that location. But in making the prediction, it is necessary to consider the second problem as well. The previously obtained location measurements are likely not accurate, so the prediction method needs to be robust enough to handle this source of error. This report will discusses how these problems can be addressed with the Kalman filter, and studies how well they are solved.

2.2 Literature Survey

The tracker implementation of this project is based on a paper by Hager & Belhumeur [4]. They discuss a sum-of-squared differences (SSD) tracking method which is again based on similar SSD-type approaches from stereo vision, optical flow computation, hand-eye coordination, and visual motion analysis. They point out that tracking is made difficult by the large variability of the image of an object over time. The three main sources of variability they identify are: object pose, variation in illumination, and partial or full occlusion of the object. These factors contribute to the position measurement error when tracking the target. They choose to model the image changes due to motion, illumination, and occlusion rather than accepting them as sources of error in the measurements. This project does not deal directly with these factors, but instead tries to compensate for the measurement errors they cause.

The Kalman filter is useful for exactly this purpose, handling noisy measurements (and also a noisy process). The original papers [6] [7] that introduced this technique were published in 1960 and 1961. There was in fact a very similar algorithm published in 1958 by Peter Swerling, however since this work was published in a less prestigious journal and was not as general and complete, the method was named after Rudolph the red nosed Kalman instead. Merry Christmas! The filter is also sometimes referred to as Kalman-Bucy filter after the name of Richard Bucy who joined Kalman in the early stages of the development.

Initially, the filter was designed for the application in spacecraft navigation, but its general nature allows it to be applied to many fields. The introduction to Kalman filtering by Simon [8] also mentions applications in instrumentation, demographic modelling, manufacturing, fuzzy logic, and neural network training.

Although there are numerous papers that apply Kalman filtering for the use in object tracking, none could be found that discuss solely why the Kalman filter is considered applicable for this purpose. One reason for this might be that even before visual tracking became popular, many papers studied the applicability to navigation systems. Since there are fundamental similarities between visual tracking and navigation, it may have appeared pointless to repeat these studies in the only slightly different context.

An example of a study of the applicability of Kalman filtering methods to navigation systems is provided by Brock & Schmidt [1]. They point out some of the advantages and problems associated with using this type of filtering for navigation. These also apply to

visual tracking. One of the advantages they mention is the usefulness of the estimated errors provided by the filter (e.g. measurement error estimates could be displayed to an operator together with the measurements).

The main problem with Kalman filtering that Brock & Schmidt identify, is that statistical models are required for the system and the measurement instruments. Unfortunately, they are typically not available, or difficult to obtain. The need for statistical models is also pointed out as a problem in many other papers [2] [3] [8] [11]. From these papers, the two most commonly recommended methods of approaching this problem are:

- Employ an adaptive algorithm which adjusts these unknown parameters (such as the measurement noise variance) after each time step based on the observed measurements. This also accounts for processes with changing parameters.
- Perform an "off-line" analysis of the system and measurement instruments prior to running the process (*system identification*).

It should be noted however that the second approach will not always be applicable if the process can not be observed directly. In other words, if the measurements in the off-line analysis also contain errors, the process can not be accurately profiled.

It was recognized early on, that Kalman filtering does not stand alone as a unique technique for prediction. For example the work of Sorenson and Stubberud [9] [10] points out the similarities between the Kalman filter, Bayesian and maximum likelihood estimation. Jordan [5] also discusses the association of the Kalman filter with probabilistic theory. This is the focus of the following section.

2.3 From the HMM to the Kalman filter

There is a very interesting relationship between the hidden Markov model (HMM) and the Kalman filter. The data on which the Kalman filter operates can be represented with an HMM. And general algorithms operating on the HMM can be reduced to the Kalman filter equations by making a few assumptions. This section outlines the basic concepts and discusses how the Kalman equations can be derived from an HMM as described by Jordan [5].

One key feature of the HMM is the idea of modelling sequential data. For example, in the first assignment, the individual characters of a word were modelled as a sequence of interdependent nodes. The dependencies between the characters were discovered through statistical analysis of text. In the context of visual tracking, the sequence is made up of images taken at discrete time steps. The relationship between each of the images is based on a physical model of the scene.

2.3.1 A graphical model for the Kalman filter

Figure 2 shows a typical graphical model of an HMM. The y_t nodes, for $t = 0, \ldots, T$ represent the observable output data (or *measurements*), and x_t are the hidden nodes which will from now on be referred to as *states*. Although in the general HMM theory the

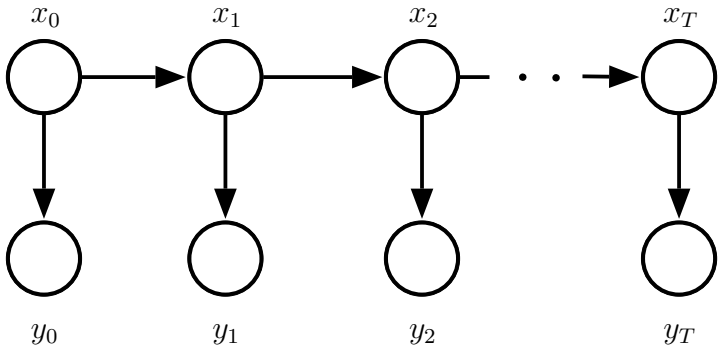

Figure 2: A typical representation of a HMM as a graphical model.

type of data contained in these nodes is not always relevant, we will be working under the assumption that the nodes are vectors of real values, and the probability model is Gaussian.

Each x_t contains an $m \times 1$ mean vector $\hat{x}$ and an $m \times m$ covariance matrix P, where m is the number of parameters that describe the state. A simple example of the parameters necessary for tracking are the x and y coordinates as well as the u and v velocity components. The y_t nodes are represented by an $n \times 1$ vector which is nothing but the observed position of the target in the context of visual tracking. This method of describing the state through a finite set of parameters is known as the state-space model (SSM).

As mentioned earlier, the state nodes are related to each other through the physics underlying object motion. The transition from one state to the next could be described in many ways. These different alternatives can be grouped into *linear* and *non-linear* functions describing the state transition. Although it is possible to handle either of these transition types, the standard Kalman filter employs a linear transition function. The *extended Kalman filter* (EKF) allows a non-linear transition, together with a non-linear measurement relationship. For the standard Kalman filter, the state transition from t to $t+1$ can be expressed with the equation

$$x_{t+1} = Ax_t + w_t, \tag{1}$$

where A is referred to as the *state transition matrix* and w_t is a noise term. This noise term is a Gaussian random variable with zero mean and a covariance matrix Q, so its probability distribution is

$$p(w) \sim N(0, Q). \tag{2}$$

The covariance matrix Q will be referred to as the *process noise covariance matrix* in the remainder of this report. It accounts for possible changes in the process between t and $t+1$ that are not already accounted for in the state transition matrix. Another assumed property of w_t is that it is independent of the state x_t.

It is also necessary to model the measurement process, or the relationship between the state and the measurement. In a general sense, it is not always possible to observe the process directly (i.e. all the state parameters are observable without error). Some of the parameters describing the state may not be observable at all, measurements might be scaled parameters, or possibly a combination of multiple parameters. Again, the assumption is

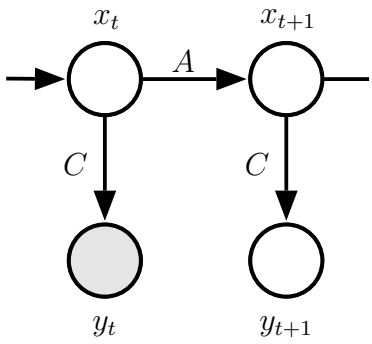

Figure 3: State of the HMM before a measurement is made. After the prediction is made and the measurement is taken, the node y_{t+1} becomes observed and x_{t+1} node can be updated.

made that the relationship is linear. So the measurement y_t can be expressed in terms of the state x_t with

$$y_t = Cx_t + v_t, \tag{3}$$

where C is an $m \times n$ matrix which relates the state to the measurement. Much like w_t for the process, v_t is the noise of the measurement. It is also assumed to have a normal distribution expressed by

$$p(v) \sim N(0, R), \tag{4}$$

where R is the covariance matrix referred to as *measurement noise covariance matrix*.

2.3.2 Making predictions

Now that a graphical model has been established and the relationships between the nodes are formulated, it is possible to look at how these relationships can be used in the tracking process. As mentioned in section 2.1, a prediction is required at each time step before the target is located with the tracking algorithm. The predicted position is nothing but the expected measurement given all the previous measurements, so $E[y_{t+1}|y_0, \ldots, y_t]$. But before finding the expected value of y_{t+1} it is necessary to know the expected state at $t+1$ given the previous measurements, i.e. $E[x_{t+1}|y_0, \ldots, y_t]$.

This problem is similar to the inference problem for the HMM. For the HMM, the general inference problem can be decomposed into two recursive algorithms which determine the probability distributions of $p(y_0, \ldots, y_t, x_t)$ and $p(y_{t+1}, \ldots, y_T|x_t)$. The algorithm for determining the first probability distribution is similar to the Kalman filter. The most important commonality is the recursive nature.

Figure 3 illustrates the situation before each prediction is made and serves as a template for the recursive step. Initially y_t is observed, and a prediction for y_{t+1} is required. After the prediction is made and measurement is taken, y_{t+1} becomes observed, and the process repeats for $t+2$.

To be able to apply this type of a recursive algorithm, two steps need to be completed within each recursion:

1. The **time update step** determines the expected value of x_{t+1}, and the associated covariance matrix P_{t+1}. The prediction for y_{t+1} can be then calculated from the expected value of x_{t+1}.

2. The **measurement update step** uses the measured y_{t+1} to determine x_{t+1} and P_{t+1} in preparation for the next recursive step.

Jordan [5] introduces a simplified notation for the purpose of clarity which is also used in this report. The mean of x_{t+1} conditioned on $y_0, \ldots, y_t$ is written as $\hat{x}_{t+1|t}$, and similarly the covariance matrix is denoted as $P_{t+1|t}$. Accordingly, the measurement update step determines $\hat{x}_{t+1|t+1}$ and $P_{t+1|t+1}$ since the measurement of y_{t+1} has been made at that point.

2.3.3 Time update step

The expected state at $t+1$ can easily be determined by applying the state transition matrix to $\hat{x}_{t|t}$. This is possible due to the assumption that the process noise has zero mean. Hence the expected mean can be expressed as:

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t}. \tag{5}$$

To determine the covariance, we need to recall the assumptions of w_t having zero mean and being independent of x_t. Then, the $P_{t+1|t}$ can be derived with:

$$\begin{aligned} P_{t+1|t} &= E[(x_{t+1} - \hat{x}_{t+1|t})(x_{t+1} - \hat{x}_{t+1|t})^T | y_0, \ldots, y_t] \\ &= E[(Ax_t + w_t - A\hat{x}_{t|t})(Ax_t + w_t - A\hat{x}_{t|t})^T | y_0, \ldots, y_t] \\ &= AP_{t|t}A^T + Q. \end{aligned} \tag{6}$$

To arrive at the expected measurement (or prediction), recall equation (3). The assumption of zero mean noise again simplifies the calculation to:

$$\begin{aligned} E[y_{t+1}|y_0, \ldots, y_t] &= E[Cx_{t+1} + v_{t+1}|y_0, \ldots, y_t] \\ &= C\hat{x}_{t+1|t} \end{aligned} \tag{7}$$

2.3.4 Measurement update step

The calculations necessary for the measurement update step are slightly more involved. The main goal is to determine the distribution of x_{t+1} given $y_0, \ldots, y_{t+1}$. Only the results will be presented here. For the full derivation, please refer to Jordan's discussion [5].

The values of $\hat{x}_{t+1|t+1}$ and $P_{t+1|t+1}$ can be obtained with:

$$\begin{aligned} \hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + K_{t+1}(y_{t+1} - C\hat{x}_{t+1|t}) \\ P_{t+1|t+1} &= P_{t+1|t} - K_{t+1}CP_{t+1|t}, \end{aligned} \tag{8}$$

where K_{t+1}, the so called *Kalman gain matrix*, is determined with

$$K_{t+1} = P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1}. \tag{9}$$

Together, the equations of the time update and measurement update steps form an elegant recursive algorithm that estimates the state of a process. Although the Kalman filter is easy to implement due to its compact form, it requires a significant amount of knowledge about the process and measurements as the following sections will show.

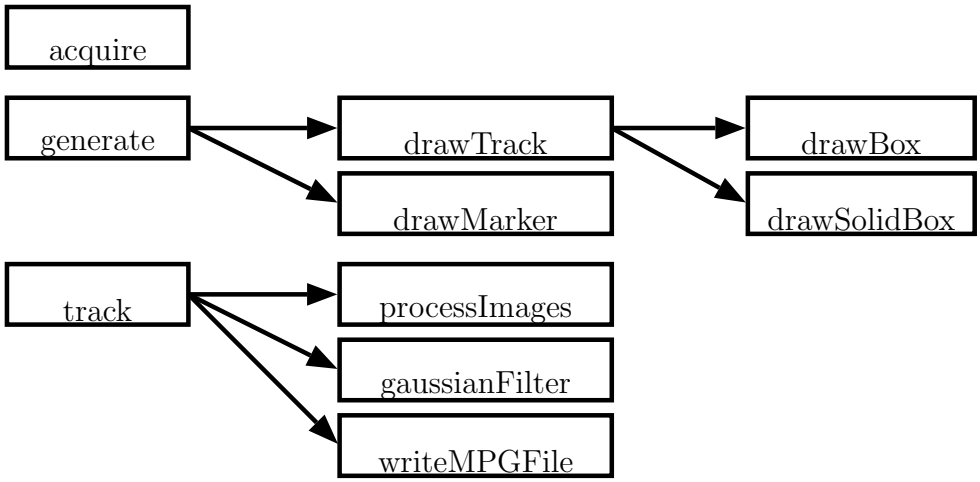

Figure 4: Dependencies of the implemented MATLAB functions.

3 Methods

3.1 Approach

The entire implementation is written as MATLAB functions. A rough dependency diagram of all the functions involved is shown in Figure 4. The three main functions are `acquire`, `generate` and `track`. The `acquire` function can be used to grab a sequence of images from a web-cam. It employs the MexVision and XVision2 packages to accomplish this. The function `generate` creates synthetic image sequences, which will be discussed in greater detail later. Finally, `track` implements the actual tracking algorithm and includes the implementation of the Kalman filter.

As mentioned in the previous section, the tracking implementation is based on a so called *SSD tracker*. The SSD tracking algorithm is general in the sense that it can be used to track an arbitrary number degrees of freedom. These might include position, scale, rotation, and skewing. This project only considers the position of the target which includes the x and y coordinates. The *track* function accepts parameters for setting the image sequence, the size of the target region, its initial position, a Gaussian smoothing filter intensity, and the type of prediction method.

Three different prediction methods can be selected (their associated parameter value is listed in brackets):

- **No prediction** (0): With this setting, no prediction method is used. The predicted position is simply set to the last position the target was located at.
- **Simple prediction** (1): This method assumes that the target will move the same distance, and in the same direction as it did since the previous image. This assumption is true if the target is moving at a constant velocity for example. Therefore, the predicted position p_{t+1} can be calculated from the two previous positions p_t and p_{t-1} with:

$$p_{t+1} = p_t + (p_t - p_{t-1}) = 2p_t - p_{t-1}. \tag{10}$$

- **Kalman filter prediction** (2): The Kalman filter is used to predict the next position as described in the last section. Parameters such as the transition matrix and error covariance matrices can be adjusted in the code.

The Kalman filter model requires a representation of the tracker in state space. Two obvious parameters required in each state are the x and y coordinates of the tracker. In addition, the state will include the velocity components v_x and v_y, and the acceleration components a_x and a_y. So the state at any point can be represented with the vector $[x, y, v_x, v_y, a_x, a_y]^T$. The state transition matrix is derived from the theory of motion under constant acceleration which can be expressed with the equations:

$$\begin{aligned} p(t+1) &= p(t) + v(t)\Delta t + a(t)\frac{\Delta t^2}{2}, \\ v(t+1) &= v(t) + a(t)\Delta t, \\ a(t+1) &= a(t), \end{aligned} \tag{11}$$

where $p, v, a, \Delta t$ are position, velocity, acceleration and the time difference between images, respectively. One might recognize that these equations are actually a Taylor series including only up to the second order terms. This should therefore be noted as one of the assumptions made. We will also assume, that $\Delta t = 1$, since we are not interested in the true velocity and acceleration values. Under this assumption, and recalling the definition of the state transition as $x_{t+1} = Ax_t + w_t$ we can set A as:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

The `generate` function creates 128×128 pixels sized synthetic image sequences which can be used for experimentation. A marker (the target to be tracked) is placed at specified locations on a solid background in each of the images. These positions are stored together with the image sequences so they can be later used as a ground truth comparison. Uniform Gaussian noise can be added to the intensity values in the images which will be referred to as *image noise* (the standard deviation of these noise values can be adjusted). The function also allows noise to be added to the position of the marker which is referred to as *position noise*.

The pixel intensities are represented by real numbers of type `double` and range from 0 to 1. Since adding image noise can result in intensities above 1 or below 0, the values are clipped to stay in this region. To avoid clipping of noise values, the background intensity is set to 0.2, and the peak intensity of the marker set to 0.8.

The marker is circular, with intensities determined by a radially symmetric 2D Gaussian distribution with its peak at the center of the marker. This allows the marker center to be located at sub-pixel locations, rather than only being located at discrete steps in the x and y dimensions.

3.2 Rationale

Tracking more than just the x and y coordinates of the target would be interesting and is possible. It would be especially interesting to study the applicability of the predication methods to these further degrees of freedom such as scaling or rotation. However, combining scaling and rotation tracking together with position tracking would make the analysis unnecessarily complex. It was decided that studying pure translational motion would allow a more focused analysis. The applicability of prediction methods for scaling and rotation could be studied separately first before looking at the results of combining multiple possible degrees of freedom.

When thinking along these lines, why not focus on a single dimension rather than using a 2D image and allowing both x and y translations? This approach was considered, and would include the analysis of one-dimensional intensity vectors rather than 2D images. It is in fact expected to provide clearer results. However, to study the effects of prediction on real image sequences, a 2D implementation was required. Furthermore, even though having results in an abstract setting is useful, a model with features a little more similar to real image sequences was desired.

The three prediction methods *no prediction, simple prediction and Kalman filter prediction* were implemented. Allowing the tracking algorithm to use *no prediction* and the *simple prediction method* provides the ability of comparing the results from the Kalman filter to other methods. The simple prediction method does not compensate for noise in any way and imposes an assumption that not true under all circumstances. So, on average, the results achieved with the Kalman filter should be at least as good as those of the simple prediction method. This is expected to be helpful in the experiments to measure the performance of the Kalman filter.

The main reason for the generation of synthetic images is to have ground truth positions to compare the measurements to. With a real image sequence, the positions of a target could be manually identified. However, the accuracy of this process would likely be insufficient. In addition, the motion pattern would not be precisely controllable. For example, it would be difficult to achieve motion of a target at constant velocity.

The image generation process allows control over motion, position noise, and image noise. Through the control of these variables it can be studied how they affect the results independently or in combination. Unfortunately, the image variability resulting from Gaussian noise added to the marker position and image intensities is not very similar the the variability caused by changes in lighting or object pose for example. This is a compromise that was made in order to reduce the complexity of the image generation process. More realistic variabilities could be achieved by rendering an image of a 3D model with appropriate lighting.

Different marker types were considered, such as a square box. The main reason why a different marker type was not chosen, is that it would be more difficult to generate an accurate image when the marker is centered at arbitrary sub-pixel locations.

4 Plan

Hypothesis 1 *When the process and measurements are not noisy, the results obtained with Kalman filter prediction are no more accurate than those obtained with the simple prediction method.*

This hypothesis claims that when the process (motion of the target) is not noisy, and the measurement process is also not noisy, the Kalman filter is unnecessary. In other words, it might not be necessary to implement and optimize the Kalman filter if it is known ahead of time that the motion is well-behaved and little image variability is present to complicate the measurements.

Experimental Design

To test the hypothesis, it is necessary to first establish a measure of performance. Simply studying whether or not the tracker follows the marker, is insufficient. Two performance measures are proposed, which can be applied to image sequences with known true positions:

1. The error of the *predicted* position with respect to the true position.
2. The error of the *measured* position with respect to the true position.

The error in the predicted position will provide a measure of how well the prediction method estimates the next position. But since the predicted position is not actually the end result, the error in the measured position is also determined in order to measure the final performance.

To simplify comparisons, the root mean square (RMS) error of all individual errors throughout the sequence will be calculated. This means that two numbers, the *RMS prediction error* and the *RMS measurement error*, will be used to measure the performance of a particular prediction method on a specific image sequence.

Tests will be conducted on generated images without position or image noise with the following motion patterns:

- *Constant velocity.* The marker moves diagonally across the image at a constant velocity of $[v_x, v_y] = [2, 2]$ pixels/frame (40 images).
- *Constant acceleration.* The marker moves diagonally across the image starting with an initial velocity of 0, at a constant acceleration of $[a_x, a_y] = [0.3, 0.3]$ pixels/frame^2 (20 images).

The performance measures will be obtained on both sequences for the simple prediction method and the Kalman filter prediction method.

The state representation and the state transition matrix A are left the same as defined in the Approach section. The process noise covariance matrix Q (also a 6×6 matrix) and the measurement noise covariance matrix R should be set to 0 as there is no noise introduced in both the process and the measurements. However, setting Q and

R to zero causes difficulties in the experiment. A matrix inversion is required in the measurement update step, which can not be performed because the matrix to be inverted is 0. So to avoid this problem, R is set to $10^{-15}I$ where I is the identity matrix. This is equivalent to expecting a small error in the position measurements, but it is expected to have a minimal effect on the final results.

Hypothesis 2 *With proper noise covariance matrices, the predictions of the Kalman filter are optimal on noisy data, and will be more accurate on average than the simple prediction method.*

The process and measurement error covariance matrices Q and R can be estimated from the generated sequences by analyzing the known position data, and estimating the effect of image noise on the measurements. Ideally, by having this information available, the prediction process would be optimal in comparison to other possible configurations of the filter. Furthermore, it should easily be able to outperform the simple prediction method which includes no measures of handling noisy measurements.

Experimental Design

This hypothesis is difficult to validate or refute through experiment since it claims that a particular configuration of the error covariance matrices gives better results than all other possible configurations. It can be proven mathematically however, that the Kalman filter reduces the estimate error to a minimum (when the error covariance matrices correspond to the data), and thus the claim should be true. Hence this experiment is more of an attempt to *find the covariance matrices that make the filter optimal*, rather than attempting to prove or refute the hypothesis.

Since the standard deviation of the positional noise can be set in the generation code, the process noise covariance matrix can be calculated from this information. There is no noise added to the velocity and acceleration values, so the only non-zero values in the process noise covariance matrix are the two position noise variances along the diagonal (at $Q_{1,1}$ and $Q_{2,2}$). Since these are variance values, they are simply calculated by squaring the standard deviation of the position noise.

But, what about the effect of image noise (noise added to the intensity values of each pixel) on the measurements? The image noise variance can also be adjusted in the generation code, but it is of course not the same as the resulting measurement noise variance. It might be proportional, but the factor is unknown. The experiment will attempt to estimate the measurement noise covariance R with an iterative process. Initially an R is guessed, then after running the tracking algorithm once, R is adjusted to the covariance of the measurement errors (determined from the true position values). This is repeated until R converges.

When a sequence is *not generated*, the process noise and measurement noise covariances are unknown. It would be nice to be able to estimate these directly from the image sequence, however since there is no positional, velocity or acceleration data available, this is not an easy task. Furthermore, the *measurement noise* can only be guessed since no true position data is available.

If the positions, velocities and acceleration at each point in time were known, estimating the process noise covariances would be relatively simple. The noise values at each time step could be calculated by calculating the difference between the current state x_t and the transition from the previous state (Ax_{t+1}). But since none of the state information is known, the position, velocity and acceleration data needs to be estimated in order to use such as technique.

An iterative approach is proposed to determine the process noise matrix Q. Initially, a "good guess" is made, then the iteration is started to obtain estimated position, velocity and acceleration values as follows:

1. Perform the tracking algorithm and record all the estimated states in $\hat{x}$.
2. Determine the *process noise* values w_t of the data in $\hat{x}$ with $w_t = \hat{x}_t - A\hat{x}_{t-1}$
3. Determine the covariance matrix of the noise values in w with $Q = cov(w)$ and continue iterating until the values in Q converge.

Since we have identified that the cases with known position data, and unknown position data need to be handled separately, two tests are proposed:

1. **Generated sequence**: Q and R are estimated through the procedure outlined above, and the results compared to the simple prediction technique. A position noise standard deviation of 1 pixel and a 0.05 image noise standard deviation are chosen. The test is performed on the constant velocity motion case.
2. **Real sequence**: Q is estimated through the iterative procedure outlined above, and the results are compared to the simple prediction technique. The chosen sequence shows a face (mine :) moving horizontally back and forth. The nose area is tracked.

Hypothesis 3 *The performance of the Kalman filter is correlated to the accuracy with which the state space model and transition matrix represent the process.*

A process can be modelled in a more or less detailed fashion. For a process consisting of multiple components, each of the components would ideally need to be modelled separately, then all models combined as noted by Brock & Schmidt [1] in reference to navigation systems. A similar concept is assumed to also apply to visual tracking. For example if a particular motion sequence includes accelerated motion, this should also be included in the model.

Experimental Design

A simple test will be performed to test this hypothesis. Although it will not be able to tell us whether more accurate models give better results in all cases, it will allow us to observe the effects of changing the model accuracy in a specific case.

Instead of trying to develop a model more refined than the current state space including parameters for position, velocity and acceleration, a *simpler* model is proposed. This model ignores the acceleration as a parameter of the state, and thus reduces the

state space representation to $[x, y, v_x, v_y]$, where x and y are the components of the position, and v_x and v_y are the components of the velocity. So the state transition matrix A is reduced to:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

The tests will be conducted on the constant acceleration case as described in the experiment design for Hypothesis 1. Position noise of 1 pixel standard deviation is added, and image noise with 0.1 standard deviation is added to the intensity values.

The process error covariance matrix Q is zero for both models except for the elements corresponding the the position noise. These two values $Q_{1,1}$ and $Q_{2,2}$ are set to one as in the previous experiments with position noise.

Since determining the most appropriate R is difficult as shown in the experiments, separate experiments will be performed with different R values:

- **Zero**: R is zero. This assumes the measurements are made without errors.
- **One**: R is the identity matrix. This assumes the x and y measurements contain independent errors with 1 pixel^2 variance.
- **Measured**: R is determined from running a single tracking iteration starting with $R = 0$, then calculating R from the covariance of the measurement error.

To compare how well the different models estimate the state of the target, the error in their velocity estimates will be plotted. This should indicate wether either one of the models predicts the state more accurately.

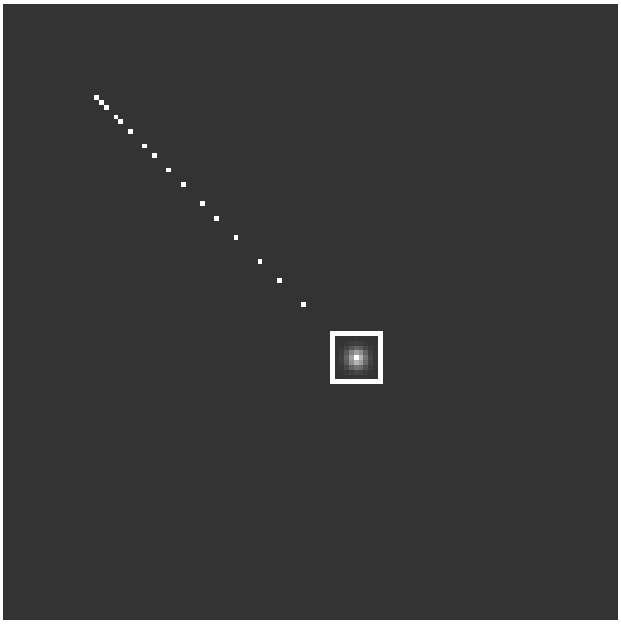

Figure 5: Example of a successful tracking experiment for the constant acceleration case. The white dots to the top left of the tracker show the previous locations of the target.

5 Experiments

5.1 Hypothesis 1: Results and Evaluation

As an example of the tracking results, Figure 5 shows the last image of the constant acceleration case. The previous positions of the tracker are highlighted with white dots.

The main result of this experiment are the RMS errors of the individual image sequence/prediction method combinations. They are listed in the following table:

		Simple Prediction	**Kalman Prediction**
Constant Velocity	RMS Prediction Error	0.513	0.699
	RMS Measurement Error	0.113	0.128
Constant Acceleration	RMS Prediction Error	0.435	0.129
	RMS Measurement Error	0.032	0.026

It can be noted, that all prediction results are quite accurate from a general standpoint. The average prediction error is less than 1 pixel.

An interesting observation that can be made from these numbers is that in the *constant velocity* case, the simple prediction method is more accurate than the Kalman filter. The reason for this is shown in Figure 6. In the first images, both prediction methods show a considerable error. This is due to the fact that both prediction method have no previous positions to derive a prediction from, so they predict that the target will stay at the original location. After the second image, they "become aware" of the motion, and the

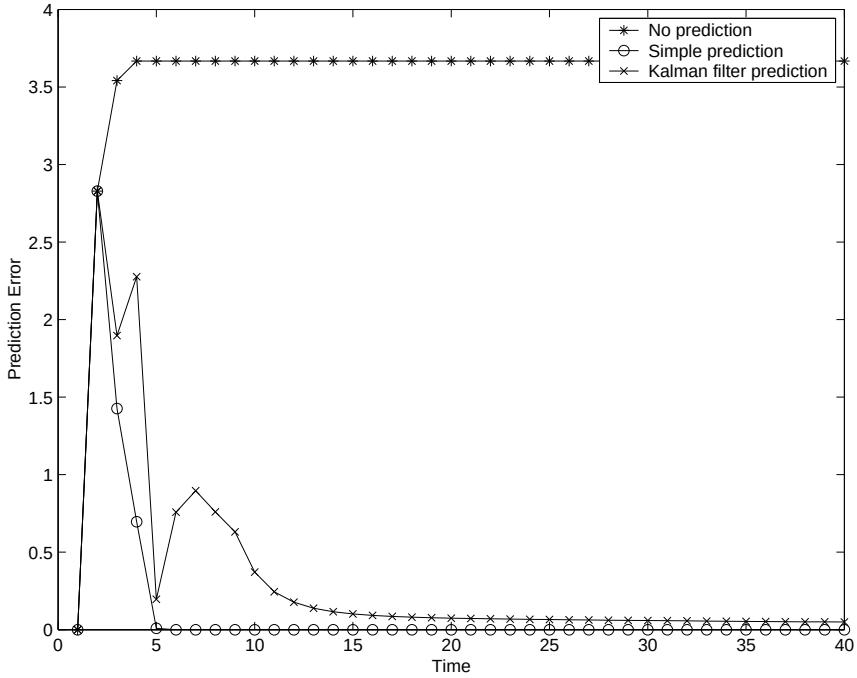

Figure 6: Comparison of the prediction errors for the constant velocity case. The simple prediction method predicts the position most accurately. The Kalman filter predictions are accurate only after the 10th image.

predictions improve. But the simple prediction method adapts quicker since it only bases its prediction on the two previous positions. The Kalman prediction method however takes longer to recover from these initial errors.

For the *constant acceleration* case however, the Kalman filter clearly outperforms the simple prediction method after 3 images as is shown in Figure 7. Since the simple prediction method does not expect the increasing velocity, it always lags behind the target with an approximately constant error. The Kalman filter reduces the prediction error quickly, and after 5 frames the prediction error stays below 0.1. This difference is also reflected in the RMS error values.

The first hypothesis states that the results of the Kalman filter are not better than the simple prediction method for cases without noise. The experiments however showed that this is not the case. Two reasons have been discovered: (1) The Kalman filter does not recover as quickly as the simple prediction process from the effects of the initial error. (2) The Kalman filter predicts more accurately for the constant acceleration case. Hence, it the hypothesis is definitely not true since the simple prediction method does not handle cases with accelerated motion well.

It would be interesting to test a different type of prediction method for the constant acceleration case to see whether it can outperform the Kalman prediction method. Especially since it has been shown that it is possible for the simple prediction method to make accurate predictions *sooner* than the Kalman filter in the constant velocity case. Such a method might include use of the three most recent positions rather than just two for

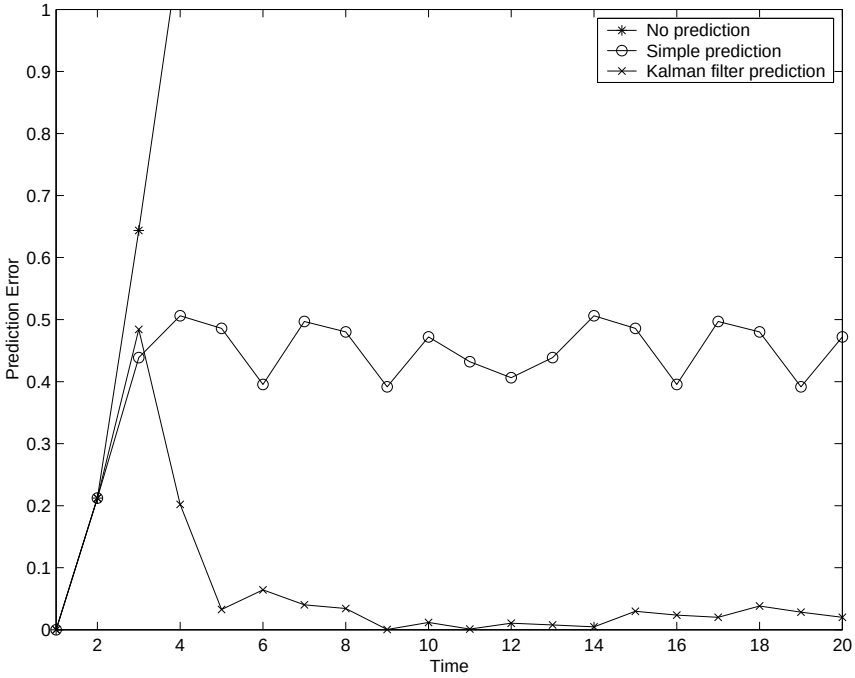

Figure 7: Comparison of the prediction errors for the constant acceleration case. This figure shows that the Kalman filter predicts the position of the target accurately starting with the fifth image.

example.

5.2 Hypothesis 2: Results and Evaluation

5.2.1 Generated Sequence

The resulting RMS values of the experiment are summarized in the following table:

Method	**RMS Pred. Error**	**RMS Meas. Error**
Kalman Iteration 1	2.5	0.94
Kalman Iteration 2	2.5	0.96
Kalman Iteration 3	2.5	0.96
Simple Prediction	2.7	0.98

The process noise covariance matrix Q is calculated from the known position error standard deviation. The initial guess for R is 0. After the first iteration, R is calculated to be $[0.42, 0.05; 0.05, 0.28]$. It converges to $[0.46, 0.07; 0.07, 0.24]$ after 3 iterations. The table shows that this iterative process with adjusting R does not improve the measurements. It can however be noted that it does consistently perform better than the simple prediction method.

It is concluded that not enough noise is present in the process and measurements to

Figure 8: Example of a successful tracking experiment on a real image sequence.

see a large difference between the Kalman prediction method and the simple prediction. The experiment could be repeated with larger noise variances to achieve this. However the point of this experiment was not to show how the Kalman filter can outperform other prediction method. Rather the process of optimizing the error covariance matrices Q and R was to be studied.

The results showed that the iterative process did not improve the RMS Predicted Error at all. One reason for is that the measurement noise was not very large in the first place, so iterating to improve the results does not show large improvements in the predictions. For larger image noise values, this process is however expected to show more interesting results. Whether or not the particular configuration of the Kalman filter is optimal was not discovered (and was not intended to).

5.2.2 Real Sequence

An example of the tracking results can be seen in Figure 8.

The following table summarizes the experiment results (no RMS Measurement Error is available for this case, since no true position data is available). Note that the RMS Pred. Error here is calculated from the error between the *prediction* and the *measurement*:

Method	**RMS Pred. Error**
Kalman Iteration 1	4.2
Kalman Iteration 2	4.5
Kalman Iteration 3	4.5
Kalman Iteration 4	4.5
Simple Prediction	4.1

The matrix R is assumed to be 0, since no true position data is available to compare the measurements to. The initial value of Q was assumed to be the identity matrix. This implies that all state parameters have a variance of 1. This is considered a "good guess", and the comparison with the simple prediction method shows that it is not completely wrong.

But the trend of the RMS error shows that the iteration does not improve the results. In fact, the initial guess shows the best results. To get an idea of how Q changes, we can look at the values along the diagonal. Initially, $diag(Q)$ is [1, 1, 1, 1, 1, 1]. This changes to [0.6, 20.8, 0.9, 21.8, 0.2, 3.9] after the first iteration. At the last iteration, Q is [0.9, 20.7, 0.9, 21.8, 0.2, 3.9]. It can be observed that the expected variance in the y directions (first, third, and fifth elements)is lower than in the x directions. This is expected since little vertical motion and variance is observed.

The reason why the results are not better can be explained as follows: The only state parameter that actually has an error variance is the acceleration. The variance in position and velocity are caused by the changes in acceleration. Therefore, the variance of the errors of position and velocity should be small or zero. The results of Q show however that a large variance is expected for the x position and the v_x velocity. In a certain way, the true source of the noise (either position, velocity or acceleration) is *irrecoverable* when only the position data can be observed.

It can be concluded that the proposed method of refining the Q matrix iteratively does not converge to an optimal configuration. The fact that the true positions are not known, hinders us from determining the true value of the measurement error covariance matrix R and a good guess must suffice. Manually tweaking the error covariance matrices is expected to give better results if enough is known about the observed motion. These results are also expected to be better than those obtained with the simple prediction method.

5.3 Hypothesis 3: Results and Evaluation

Figure 9 shows an example image to which the tracking methods were applied. The effects of the added noise are visible through the track of the target.

Again, the primary results of this experiment are the RMS error values obtained for each case. They are summarized in the following table:

Model	R **type**	**RMS Pred. Error**	**RMS Meas. Error**
With acceleration	Zero	2.08	1.05
	One	1.98	1.09
	Measured	2.02	1.03
Without acceleration	Zero	3.79	1.71
	One	5.76	2.73
	Measured	6.27	2.78

The values of both the RMS prediction errors and the RMS measurement errors show clearly that the model with acceleration performs better for any of the assumed R matrices.

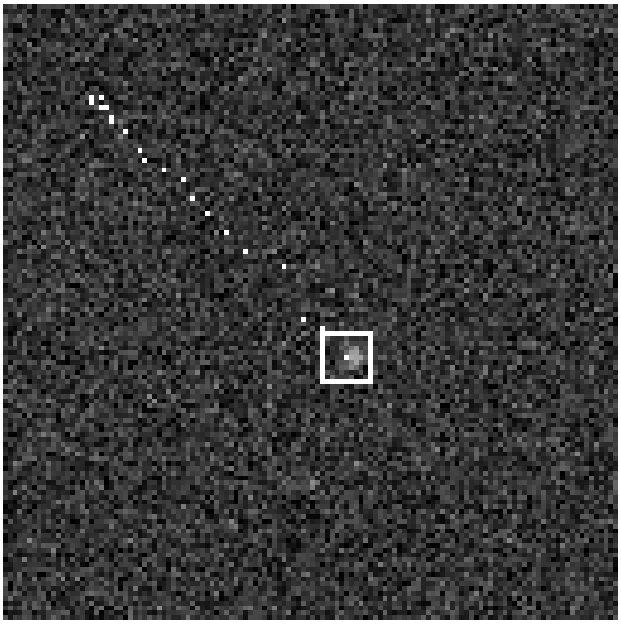

Figure 9: Example of a successful tracking experiment on the constant acceleration case with position noise (1 pixel standard deviation) and image noise (0.1 standard deviation).

The measured R matrices are [0.30, 0.04; 0.04, 0.35] for the model with acceleration, and [1.42, 0.26; 0.26, 0.77] for the model without acceleration. As a side note, it can be pointed out that the measured R matrices which are expected to represent the measurement noise more accurately than the other assumed values, do not actually increase the performance of the model greatly (the performance is even decreased for the model without acceleration). This indicates that the proposed method of determining R from Hypothesis 2 is not suitable in all cases.

To compare the accuracy of the estimated states, the error of the estimated velocity (with respect to the true velocity) for both models is plotted in Figure 10. It can be seen, that the model without acceleration shows an increasing error in its estimate of the velocity, whereas the the model with acceleration shows a generally decreasing error in the estimated velocity. Only at image 3 does the model without acceleration show a significantly lower error.

These results provide evidence indicating that Hypothesis 3 is true. The results are not unexpected in the sense that it is intuitive that a more complete model will perform better. However more complex model do not always show better results. This statement could be tested by performing the same type of analysis on the *constant velocity* case. It is expected that the results for both models would be very similar, even though the model with acceleration is more sophisticated. In a more general sense, the consideration of more parameters in the state likely does bear much of an advantage on a process in which those parameters do not affect the results much. So the disadvantages associated with making the model more complex might outweigh the advantages gained in the predictions.

The velocity error plot is a good visualization of how the Kalman filter reduces estimation errors when configured properly. Even with considerable noise added to the image and

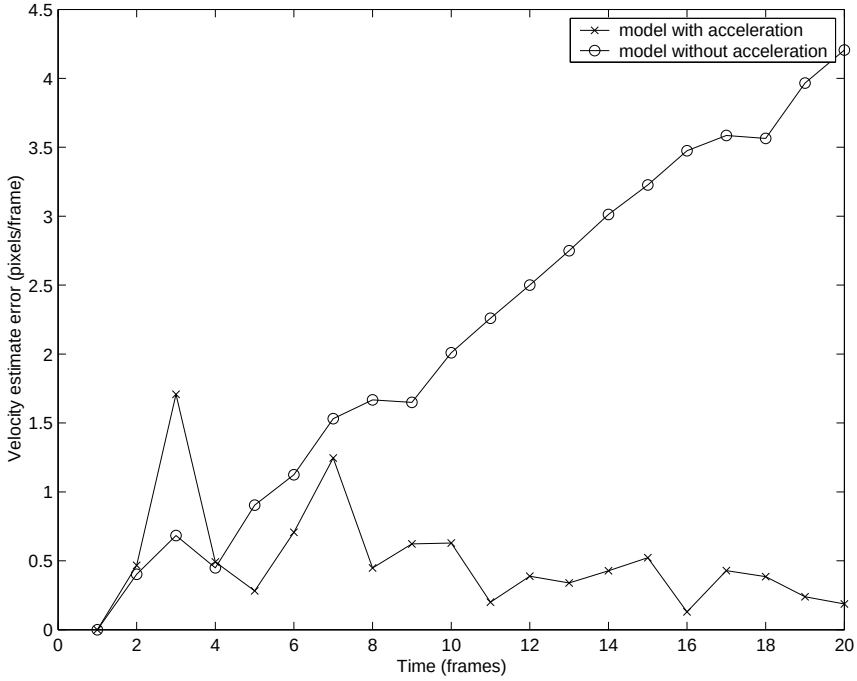

Figure 10: Comparison of the errors in the estimated velocity from a model including acceleration, and a model without acceleration. The test was performed on the constant acceleration case with noise.

to the position of the target, most of the predictions are made with sub-pixel accuracy.

To summarize the conclusions drawn from these experiments we can state the following:

- The accuracy of the predictions is increased considerably by including the acceleration in the state of the target.
- Although the model without acceleration still tracks the target with reasonable precision, it is expected that at higher noise levels or higher accelerations the predictions might not be accurate enough to keep the tracker on the target.
- This evidence indicates that the hypothesis is true. However the benefits of a more complete model are not expected to justify its use on "simple" processes.

6 Conclusions & Recommendations

This study of the Kalman filter applied to visual tracking has analyzed various aspects of using the Kalman filter as a prediction tool in attempt to make tracking more robust. Although not focused on in the hypotheses, prediction is necessary to avoid loosing the tracked target. Robustness of the prediction method to noise is also required in order to make accurate predictions.

Some of the main conclusions that can be drawn from the experiments and analysis are:

- The Kalman filter functions well as a robust prediction technique when all its parameters are configured appropriately.
- The experiments confirmed some of the claims discovered in the literature survey: In the configuration of the filter parameters, accurate prior knowledge about the process being observed is essential for effective operation.
- Even the simple prediction technique proposed can provide better results than the Kalman filter when parameters such as the process and measurement error covariance matrices are not set appropriately.
- The error covariance information can be estimated from analysis of the process, but this is difficult and does not promise good results. For example if the main cause of changes in the position are changes in the acceleration, it is difficult, if not impossible, to learn this from analyzing only the estimated position data. The proposed iterative approach to this problem failed.
- The chosen state model can have considerable effects on the accuracy of the results. Even though the filter is robust enough to predict motion of an accelerating target with a model that does not include acceleration, the accuracy increases considerably when including acceleration in the model.

While working on the project, a number of possible variations of the experiments, and other possible experiments were considered yet not implemented. They are listed here as recommendations for future research:

- The scale of the experiments was relatively small with respect to the image size (128 $\times$ 128), number of images in each sequence (maximum of 40 images), velocities, and accelerations. Tests could be performed on scaled-up versions (in many respects) of the same experiments and the results compared.
- Analyze the performance of tracking with prediction on intensity vectors rather than 2D images. This is expected to provide clear and interesting results even though it might not make as many "pretty pictures".
- Only position and image noise were considered in the generated sequences. Noise could also be added to the velocity or acceleration values.

- The generated sequences do not accurately represent what might be expected in a real scene. Instead of generating images of a 2D scene, a 3D model could be rendered, and tracking would be performed on it. This would still provide the benefit of knowing the exact location of features in the image, but would create more realistic images on which the effects of changes in lighting or object orientation could be studied.
- Further degrees of freedom such as scale and rotation can be tracked. Further experiments could study how prediction methods can be applied to these degrees of freedom, or whether they are applicable at all.

This project has taught me a lot about using probabilistic methods for prediction. In addition, the approach of defining hypotheses before performing experiments has introduced me to a different (and intriguing :) approach in studying problems and their potential solutions.

References

[1] Brock, L.D., Schmidt, G.T.: *General Questions on Kalman Filtering in Navigation Systems*, Chapter 10 of "Theory and Applications of Kalman Filtering" C.T. Leondes, Editor, NATO AGARD (1970)

[2] Gutman, P., Velger, M.: *Tracking Targets Using Adaptive Kalman Filtering*, IEEE Transactions on Aerospace and Electronic Systems Vol. 26, No. 5: pp. 691-699 (1990)

[3] Han, K., Veloso, M.: *Physical Model Based Multi-objects Tracking and Prediction in RoboSoccer* In Working Note of the AAAI 1997 Fall Symposium AAAI, MIT Press (1997)

[4] Hagar, G., Belhumeur, P.: *Efficient Region Tracking With Parametric Models of Geometry and Illumination*, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 20 No. 10: pp. 1125-1139 (1998)

[5] Jordan, M.I.: *An Introduction to Probabilistic Graphical Methods*, University of California, Berkely (2003)

[6] Kalman, R.E.: *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME - Journal of Basic Engineering Vol. 82: pp. 35-45 (1960)

[7] Kalman, R.E., Bucy R.S.: *New Results in Linear Filtering and Prediction Theory*, Transactions of the ASME - Journal of Basic Engineering Vol. 83: pp. 95-107 (1961)

[8] Simon, D.: *Kalman Filtering*, Embedded.com, viewed on Dec. 10, 2003: http://www.embedded.com/showArticle.jhtml?articleID=9900168 (2001)

[9] Sorenson, H.W.: *Comparison of Kalman, Baysian and Maximum Likelihood Estimation Techniques*, Chapter 6 of "Theory and Applications of Kalman Filtering" C.T. Leondes, Editor, NATO AGARD (1970)

[10] Sorenson, H.W., Stubberud: *Linear Estimation Theory*, Chapter 1 of "Theory and Applications of Kalman Filtering" C.T. Leondes, Editor, NATO AGARD (1970)

[11] Welch, G., Bishop, G.: *An Introduction to the Kalman Filter*, ACM SIGGRAPH 2001, Course 8, available at http://www.cs.unc.edu/∼welch/kalman/ (2001)