Computer Vision
A Modern Approach

David A. Forsyth
University of California at Berkeley

Jean Ponce
University of Illinois at Urbana-Champaign

An Alan R. Apt Book

Prentice Hall
Upper Saddle River, New Jersey 07458

17

Tracking with Linear Dynamic Models

Tracking is the problem of generating an inference about the motion of an object given a sequence of images. Good solutions to this problem have a variety of applications:

- **Motion capture:** If we can track a moving person accurately, then we can make an accurate record of their motions. Once we have this record, we can use it to drive a rendering process; for example, we might control a cartoon character, thousands of virtual extras in a crowd scene, or a virtual stunt avatar. Furthermore, we could modify the motion record to obtain slightly different motions. This means that a single performer can produce sequences they wouldn't want to do in person.
- **Recognition from motion:** The motion of objects is quite characteristic. We may be able to determine the identity of the object from its motion; we should be able to tell what it's doing.
- **Surveillance:** Knowing what objects are doing can be very useful. For example, different kinds of trucks should move in different, fixed patterns in an airport; if they do not, then something is going wrong. Similarly, there are combinations of places and patterns of motions that should never occur (e.g., no truck should ever stop on an active runway). It could be helpful to have a computer system that can monitor activities and give a warning if it detects a problem case.
- **Targeting:** A significant fraction of the tracking literature is oriented toward (a) deciding what to shoot, and (b) hitting it. Typically, this literature describes tracking using radar or infrared signals (rather than vision), but the basic issues are the same—what do we

infer about an object's future position from a sequence of measurements? Where should we aim?

In typical tracking problems, we have a model for the object's motion and some set of measurements from a sequence of images. These measurements could be the position of some image points, the position and moments of some image regions, or pretty much anything else. They are not guaranteed to be relevant, in the sense that some could come from the object of interest and some might come from other objects or from noise.

Tracking is properly thought of as an inference problem. The moving object has some form of internal state, which is measured at each frame. We need to combine our measurements as effectively as possible to estimate the object's state. There are two important cases. Either both **dynamics and measurement are linear**, in which case the inference problem is straightforward and has a standard solution. If we are faced with **nonlinear dynamics**, even slight nonlinearities in system dynamics have tremendous effects. As a result, inference can be difficult and appears to be impossible in general. If the dimension of the state space is low, there is a useful algorithm that often works. As tracking through nonlinear dynamics is a somewhat technical activity, we have confined it to its own chapter which appears on the book website In this chapter, we concentrate on the formulation of tracking through linear dynamics. Section 17.1 sketches the overall view of tracking as an inference problem. In Section 17.2, we deal with linear dynamics and the Kalman filter. We then sketch out some examples of tracking applications in Section 17.5. The most interesting current tracking application—tracking people—requires a discussion of nonlinear dynamics and is discussed in the chapter on the book website.

17.1 TRACKING AS AN ABSTRACT INFERENCE PROBLEM

Much of this chapter deals with the algorithmics of tracking. In particular, we see tracking as a probabilistic inference problem. The key technical difficulty is maintaining an accurate representation of the posterior on object position given measurements and doing so efficiently. We model the object as having some internal state; the state of the object at the ith frame is typically written as $\boldsymbol{X}_i$. The capital letters indicate that this is a random variable—when we want to talk about a particular value that this variable takes, we use small letters. The measurements obtained in the ith frame are values of a random variable $\boldsymbol{Y}_i$; we write $\boldsymbol{y}_i$ for the value of a measurement, and, on occasion, we write $\boldsymbol{Y}_i = \boldsymbol{y}_i$ for emphasis. There are three main problems:

- **Prediction:** We have seen $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}$—what state does this set of measurements predict for the ith frame? To solve this problem, we need to obtain a representation of $P(\boldsymbol{X}_i \mid \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$.
- **Data association:** Some of the measurements obtained from the i-th frame may tell us about the object's state. Typically, we use $P(\boldsymbol{X}_i \mid \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$ to identify these measurements.
- **Correction:** Now that we have $\boldsymbol{y}_i$—the relevant measurements—we need to compute a representation of $P(\boldsymbol{X}_i \mid \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_i = \boldsymbol{y}_i)$.

17.1.1 Independence Assumptions

Tracking is difficult without the following assumptions:

- **Only the immediate past matters:** Formally, we require that

$$P(\boldsymbol{X}_i \mid \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}) = P(\boldsymbol{X}_i \mid \boldsymbol{X}_{i-1}).$$

This assumption hugely simplifies the design of algorithms as we shall see. Furthermore, it isn't terribly restrictive if we're clever about interpreting X_i, as we shall show in the next section.

- **Measurements depend only on the current state:** We assume that Y_i is conditionally independent of all other measurements given X_i. This means that

$$P(Y_i, Y_j, \ldots Y_k \mid X_i) = P(Y_i \mid X_i) P(Y_j, \ldots, Y_k \mid X_i).$$

Again, this isn't a particularly restrictive or controversial assumption, but it yields important simplifications.

These assumptions mean that a tracking problem has the structure of inference on a hidden Markov model (where both state and measurements may be on a continuous domain). You should compare this chapter with Section 23.4, which describes the use of hidden Markov models in recognition.

17.1.2 Tracking as Inference

We proceed inductively. First, we assume that we have $P(X_0)$, which is our prediction in the absence of any evidence. Now correcting this is easy: When we obtain the value of Y_0—which is y_0—we have

$$\begin{aligned} P(X_0 \mid Y_0 = y_0) &= \frac{P(y_0 \mid X_0) P(X_0)}{P(y_0)} \\ &= \frac{P(y_0 \mid X_0) P(X_0)}{\int P(y_0 \mid X_0) P(X_0) dX_0} \\ &\propto P(y_0 \mid X_0) P(X_0). \end{aligned}$$

All this is just Bayes rule, and we either compute or ignore the constant of proportionality depending on what we need. Now assume we have a representation of $P(X_{i-1} \mid y_0, \ldots, y_{i-1})$.

Prediction Prediction involves representing

$$P(X_i \mid y_0, \ldots, y_{i-1}).$$

Our independence assumptions make it possible to write

$$\begin{aligned} P(X_i \mid y_0, \ldots, y_{i-1}) &= \int P(X_i, X_{i-1} \mid y_0, \ldots, y_{i-1}) dX_{i-1} \\ &= \int P(X_i \mid X_{i-1}, y_0, \ldots, y_{i-1}) P(X_{i-1} \mid y_0, \ldots, y_{i-1}) dX_{i-1} \\ &= \int P(X_i \mid X_{i-1}) P(X_{i-1} \mid y_0, \ldots, y_{i-1}) dX_{i-1}. \end{aligned}$$

Correction Correction involves obtaining a representation of

$$P(X_i \mid y_0, \ldots, y_i).$$

Our independence assumptions make it possible to write

$$\begin{aligned}
P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_i) &= \frac{P(\boldsymbol{X}_i, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)}{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)} \\
&= \frac{P(\boldsymbol{y}_i \mid \boldsymbol{X}_i, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)} \\
&= P(\boldsymbol{y}_i \mid \boldsymbol{X}_i) P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) \frac{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)} \\
&= \frac{P(\boldsymbol{y}_i \mid \boldsymbol{X}_i) P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})}{\int P(\boldsymbol{y}_i \mid \boldsymbol{X}_i) P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_i}.
\end{aligned}$$

17.1.3 Overview

The key algorithmic issue involves finding a representation of the relevant probability densities that (a) is sufficiently accurate for our purposes, and (b) allows these two crucial sums to be done quickly and easily. The simplest case occurs when the dynamics are linear, the measurement model is linear, and the noise models are Gaussian (Section 17.2). We discuss data association in Section 17.4, and show some examples of tracking systems in action in Section 17.5. Nonlinearities introduce a host of unpleasant problems; we discuss some current methods for handling them in a chapter that appears on the website.

17.2 LINEAR DYNAMIC MODELS

There are good relations between linear transformations and Gaussian probability densities. The practical consequence is that if we restrict attention to linear dynamic models and linear measurement models, both with additive Gaussian noise, all the densities we are interested in will be Gaussians. Furthermore, the question of solving the various integrals we encounter can usually be avoided by tricks that allow us to determine directly *which* Gaussian we are dealing with.

In the simplest possible dynamic model, the state is advanced by multiplying it by some known matrix (which may depend on the frame) and then adding a normal random variable of zero mean and known covariance. Similarly, the measurement is obtained by multiplying the state by some matrix (which may depend on the frame) and then adding a normal random variable of zero mean and known covariance. We use the notation

$$\boldsymbol{x} \sim N(\mu, \Sigma)$$

to mean that $\boldsymbol{x}$ is the value of a random variable with a normal probability distribution with mean μ and covariance Σ; notice that this means that, if $\boldsymbol{x}$ is one-dimensional—we'd write $x \sim N(\mu, v)$—that its standard deviation is $\sqrt{v}$. We can write our dynamic model as

$$\begin{aligned}
\boldsymbol{x}_i &\sim N(\mathcal{D}_i \boldsymbol{x}_{i-1}; \Sigma_{d_i}); \\
\boldsymbol{y}_i &\sim N(\mathcal{M}_i \boldsymbol{x}_i; \Sigma_{m_i}).
\end{aligned}$$

Notice that the covariances could be different from frame to frame as could the matrices. Although this model appears limited, it is in fact extremely powerful; we show how to model some common situations next.

17.2.1 Drifting Points

Let us assume that $\boldsymbol{x}$ encodes the position of a point. If $\boldsymbol{D}_i = Id$, then the point is moving under random walk—its new position is its old position plus some Gaussian noise term. This form of dynamics isn't obviously useful because it appears that we are tracking stationary objects. It is quite commonly used for objects for which no better dynamic model is known—we assume that the random component is quite large and hope we can get away with it.

This model also illustrates aspects of the **measurement matrix** $\mathcal{M}$. The most important thing to keep in mind is that we don't need to measure every aspect of the state of the point at every step. For example, assume that the point is in 3D: Now if $\mathcal{M}_{3k} = (0, 0, 1)$, $\mathcal{M}_{3k+1} = (0, 1, 0)$, and $\mathcal{M}_{3k+2} = (1, 0, 0)$, then at every third frame we measure, respectively, the z, y, or x position of the point. Notice that we can still expect to track the point, even though we measure only one component of its position at a given frame. If we have sufficient measurements, we can reconstruct the state—the state is **observable**. We explore observability in the exercises.

17.2.2 Constant Velocity

Assume that the vector $\boldsymbol{p}$ gives the position and $\boldsymbol{v}$ the velocity of a point moving with constant velocity. In this case, $\boldsymbol{p}_i = \boldsymbol{p}_{i-1}^{\circ} + (\Delta t)\boldsymbol{v}_{i-1}$ and $\boldsymbol{v}_i = \boldsymbol{v}_{i-1}$. This means that we can stack the position and velocity into a single state vector, and our model applies (Figure 17.1). In particular,

$$\boldsymbol{x} = \begin{Bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{Bmatrix}$$

and

$$\mathcal{D}_i = \begin{Bmatrix} Id & (\Delta t)Id \\ 0 & Id \end{Bmatrix}.$$

Notice that, again, we don't have to observe the whole state vector to make a useful measurement. For example, in many cases, we would expect that

$$\mathcal{M}_i = \begin{Bmatrix} Id & 0 \end{Bmatrix}$$

(i.e., that we see only the position of the point). Because we know that it's moving with constant velocity—that's the model—we expect that we could use these measurements to estimate the whole state vector rather well.

17.2.3 Constant Acceleration

Assume that the vector $\boldsymbol{p}$ gives the position, vector $\boldsymbol{v}$ the velocity, and vector $\boldsymbol{a}$ the acceleration of a point moving with constant acceleration. In this case, $\boldsymbol{p}_i = \boldsymbol{p}_{i-1} + (\Delta t)\boldsymbol{v}_{i-1}$, $\boldsymbol{v}_i = \boldsymbol{v}_{i-1} + (\Delta t)\boldsymbol{a}_{i-1}$, and $\boldsymbol{a}_i = \boldsymbol{a}_{i-1}$. Again, we can stack the position, velocity and acceleration into a single state vector, and our model applies (Figure 17.2). In particular,

$$\boldsymbol{x} = \begin{Bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \\ \boldsymbol{a} \end{Bmatrix}$$

and

$$\mathcal{D}_i = \begin{Bmatrix} Id & (\Delta t)Id & 0 \\ 0 & Id & (\Delta t)Id \\ 0 & 0 & Id \end{Bmatrix}.$$

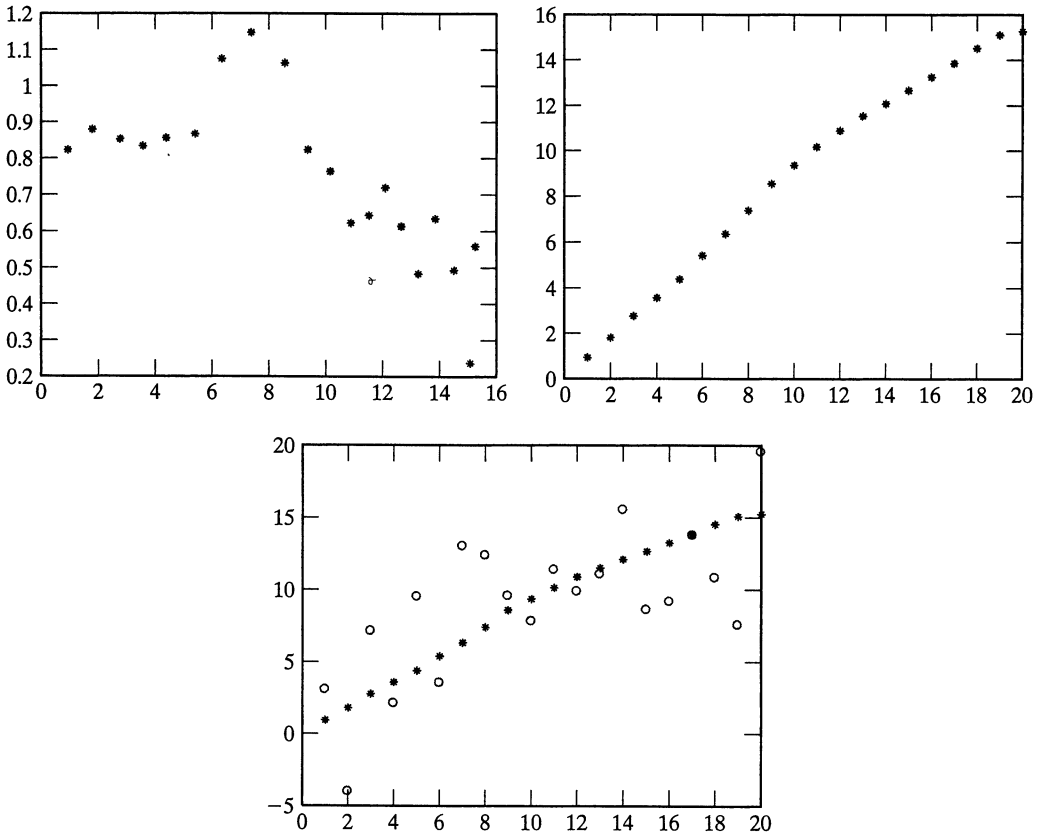

Figure 17.1 A constant velocity dynamic model for a point on the line. In this case, the state space is two dimensional—one coordinate for position, one for velocity. The figure on the **top left** shows a plot of the state; each asterisk is a different state. Notice that the vertical axis (velocity) shows some small change compared with the horizontal axis. This small change is generated only by the random component of the model, so the velocity is constant up to a random change. The figure on the **top right** shows the first component of state (which is position) plotted against the time axis. Notice we have something that is moving with roughly constant velocity. The figure on the **bottom** overlays the measurements (the circles) on this plot. We are assuming that the measurements are of position only, and are quite poor; as we see, this doesn't significantly affect our ability to track.

Notice that, again, we don't have to observe the whole state vector to make a useful measurement. For example, in many cases, we would expect that

$$\mathcal{M}_i = \{ \; Id \quad 0 \quad 0 \; \}$$

(i.e., that we see only the position of the point). Because we know that it's moving with constant acceleration—that's the model—we expect that we could use these measurements to estimate the whole state vector rather well.

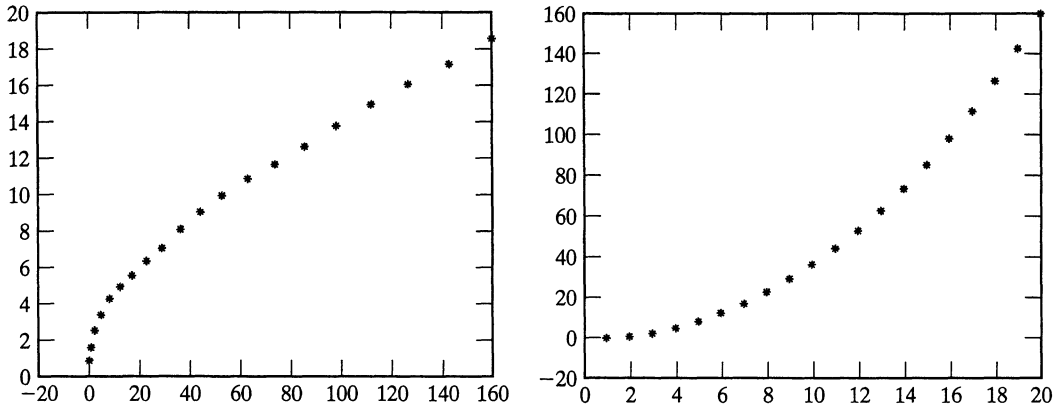

Figure 17.2 This figure illustrates a constant acceleration model for a point moving on the line. On the **left**, we show a plot of the first two components of state—the position on the x-axis and the velocity on the y-axis. In this case, we expect the plot to look like (t^2, t), which it does. On the **right**, we show a plot of the position against time—note that the point is moving away from its start position increasingly quickly.

17.2.4 Periodic Motion

Assume we have a point moving on a line with a periodic movement. Typically, its position p satisfies a differential equation like

$$\frac{d^2p}{dt^2} = -p.$$

This can be turned into a first order linear differential equation by writing the velocity as v and stacking position and velocity into a vector $\boldsymbol{u} = (p, v)$; we then have

$$\frac{d\boldsymbol{u}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{u} = \mathcal{S}\boldsymbol{u}.$$

Now assume we are integrating this equation with a forward Euler method, where the steplength is Δt; we have

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{u}_{i-1} + \Delta t \frac{d\boldsymbol{u}}{dt} \\ &= \boldsymbol{u}_{i-1} + \Delta t \mathcal{S}\boldsymbol{u}_{i-1} \\ &= \begin{pmatrix} 1 & \Delta t \\ -\Delta t & 1 \end{pmatrix} \boldsymbol{u}_{i-1}. \end{aligned}$$

We can either use this as a state equation, or we can use a different integrator. If we used a different integrator, we might have some expression in $\boldsymbol{u}_{i-1}, \ldots, \boldsymbol{u}_{i-n}$—we would need to stack $\boldsymbol{u}_{i-1}, \ldots, \boldsymbol{u}_{i-n}$ into a state vector and arrange the matrix appropriately (see Exercises). This method works for points on the plane, in 3D, and so on, as well (again, see Exercises).

17.2.5 Higher Order Models

Another way to look at a constant velocity model is that we have augmented the state vector to get around the requirement that $P(\boldsymbol{x}_i \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}) = P(\boldsymbol{x}_i \mid \boldsymbol{x}_{i-1})$. We could write a constant

velocity model in terms of point position alone as long as we are willing to use the position of the $i-2$th point as well as that of the $i-1$th point. In particular, writing position as $\boldsymbol{p}$, we would have

$$P(\boldsymbol{p}_i \mid \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{i-1}) = N(\boldsymbol{p}_{i-1} + (\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i-2}), \Sigma_{d_i}).$$

This model assumes that the difference between $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i-1}$ is the same as the difference between $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_{i-2}$—i.e., that the velocity is constant up to the random element. A similar remark applies to the constant acceleration model, which is now in terms of $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_{i-2}$, and $\boldsymbol{p}_{i-3}$.

We augmented the position vector with the velocity vector (which represents $\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i-2}$) to get the state vector for a constant velocity model. Similarly, we augmented the position vector with the velocity vector and the acceleration vector to get a constant acceleration model. In this model, the acceleration vector represents $(\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i-2}) - (\boldsymbol{p}_{i-2} - \boldsymbol{p}_{i-3})$. We might reasonably want the new position of the point to depend on $\boldsymbol{p}_{i-4}$ or other points even further back in the history of the point's track. To represent dynamics like this, all we need to do is augment the state vector to a suitable size. Notice that it can be somewhat difficult to visualize how the model will behave. There are two approaches to determining what $\mathcal{D}_i$ needs to be; in the first, we know something about the dynamics and can write it down, as we have done here; in the second, we need to learn it from data.

17.3 KALMAN FILTERING

An important feature of linear dynamic models is that all the conditional probability distributions we need to deal with are normal distributions. In particular, $P(\boldsymbol{X}_i \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-1})$ is normal, as is $P(\boldsymbol{X}_i \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_i)$. This means that they are relatively easy to represent—all we need to do is maintain representations of the mean and the covariance for the prediction and correction phase. In particular, our model admits a relatively simple process where the representation of the mean and covariance for the prediction and estimation phase are updated.

17.3.1 The Kalman Filter for a 1D State Vector

The dynamic model is now

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i}^2);$$

$$y_i \sim N(m_i x_i, \sigma_{m_i}^2).$$

We need to maintain a representation of $P(X_i \mid y_0, \ldots, y_{i-1})$ and of $P(X_i \mid y_0, \ldots, y_i)$. In each case, we need only represent the mean and the standard deviation because the distributions are normal.

Notation We represent the mean of $P(X_i \mid y_0, \ldots, y_{i-1})$ as $\overline{X}_i^-$ and the mean of $P(X_i \mid y_0, \ldots, y_i)$ as $\overline{X}_i^+$—the superscripts suggest that they represent our belief about X_i immediately before and immediately after the ith measurement arrives. Similarly, we represent the standard deviation of $P(X_i \mid y_0, \ldots, y_{i-1})$ as σ_i^- and of $P(X_i \mid y_0, \ldots, y_i)$ as σ_i^+. In each case, we assume that we know $P(X_{i-1} \mid y_0, \ldots, y_{i-1})$, meaning that we know $\overline{X}_{i-1}^+$ and σ_{i-1}^+.

Tricks with Integrals The main reason we work with normal distributions is that their integrals are quite well behaved. We are going to obtain values for various parameters as integrals usually by change of variable. Our current notation can make appropriate changes a bit difficult to spot, so we write

$$g(x; \mu, v) = \exp\left(-\frac{(x-\mu)^2}{2v}\right).$$

We have dropped the constant and, for convenience, are representing the *variance* (as v), rather than the standard deviation. This expression allows some convenient transformations; in particular, we have

$$\begin{aligned} g(x; \mu, v) &= g(x - \mu; 0, v); \\ g(m; n, v) &= g(n; m, v); \\ g(ax; \mu, v) &= g(x; \mu/a, v/a^2). \end{aligned}$$

We also need the following fact:

$$\int_{-\infty}^{\infty} g(x-u; \mu, v_a) g(u; 0, v_b)\, du \propto g(x; \mu, v_a^2 + v_b^2).$$

There are several ways to confirm that this is true: The easiest is to look it up in tables; more subtle is to think about convolution directly; more subtle still is to think about the sum of two independent random variables. We need a further identity. We have

$$g(x; a, b) g(x; c, d) = g\left(x; \frac{ad+cb}{b+d}, \frac{bd}{b+d}\right) f(a, b, c, d),$$

where the form of f is not significant, but the fact that it is not a function of x is. The exercises show you how to prove this identity.

Prediction We have

$$P(X_i \mid y_0, \ldots, y_{i-1}) = \int P(X_i \mid X_{i-1}) P(X_{i-1} \mid y_0, \ldots, y_{i-1})\, dX_{i-1}.$$

Now

$$\begin{aligned} P(X_i \mid y_0, \ldots, y_{i-1}) &= \int P(X_i \mid X_{i-1}) P(X_{i-1} \mid y_0, \ldots, y_{i-1}) dX_{i-1}) \\ &\propto \int_{-\infty}^{\infty} g(X_i; d_i X_{i-1}, \sigma_{d_i}^2) g(X_{i-1}; \overline{X}_{i-1}^+, (\sigma_{i-1}^+)^2) dX_{i-1} \\ &\propto \int_{-\infty}^{\infty} g((X_i - d_i X_{i-1}); 0, \sigma_{d_i}^2) g((X_{i-1} - \overline{X}_{i-1}^+); 0, (\sigma_{i-1}^+)^2) dX_{i-1} \\ &\propto \int_{-\infty}^{\infty} g((X_i - d_i(u + \overline{X}_{i-1}^+)); 0, (\sigma_{d_i})^2) g(u; 0, (\sigma_{i-1}^+)^2) du \\ &\propto \int_{-\infty}^{\infty} g((X_i - d_i u); d_i \overline{X}_{i-1}^+, \sigma_{d_i}^2) g(u; 0, (\sigma_{i-1}^+)^2) du \\ &\propto \int_{-\infty}^{\infty} g((X_i - v); d_i \overline{X}_{i-1}^+, \sigma_{d_i}^2) g(v; 0, (d_i \sigma_{i-1}^+)^2) dv \\ &\propto g(X_i; d_i \overline{X}_0^+, \sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2), \end{aligned}$$

where we have applied various of the transformations given earlier, and changed variables twice. All this means that

$$\overline{X_i^-} = d_i \overline{X_{i-1}^+};$$
$$(\sigma_i^-)^2 = \sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2.$$

Correction We have

$$P(X_i \mid y_0, \ldots, y_i) = \frac{P(y_i \mid X_i) P(X_i \mid y_0, \ldots, y_{i-1})}{\int P(y_i \mid X_i) P(X_i \mid y_0, \ldots, y_{i-1}) dX_i}$$
$$\propto P(y_i \mid X_i) P(X_i \mid y_0, \ldots, y_{i-1}).$$

We know $\overline{X_i^-}$ and σ_i^-, which represent $P(X_i \mid y_0, \ldots, y_{i-1})$.

Using the notation for Gaussians given earlier, we have

$$P(X_i \mid y_0, \ldots, y_i) \propto g(y_i; m_i X_i, \sigma_{m_i}^2) g(X_i; \overline{X_i^-}, (\sigma_i^-)^2)$$
$$= g(m_i X_i; y_i, \sigma_{m_i}^2) g(X_i; \overline{X_i^-}, (\sigma_i^-)^2)$$
$$= g\left(X_i; \frac{y_i}{m_i}, \frac{\sigma_{m_i}^2}{m_i^2}\right) g(X_i; \overline{X_i^-}, (\sigma_i^-)^2),$$

Algorithm 17.1: The 1D Kalman filter updates estimates of the mean and covariance of the various distributions encountered while tracking a one-dimensional state variable using the given dynamic model.

Dynamic Model:

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$
$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

Start Assumptions: $\overline{x_0^-}$ and σ_0^- are known

Update Equations: Prediction

$$\overline{x_i^-} = d_i \overline{x_{i-1}^+}$$
$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

Update Equations: Correction

$$x_i^+ = \left(\frac{\overline{x_i^-} \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2}\right)$$
$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)}\right)}$$

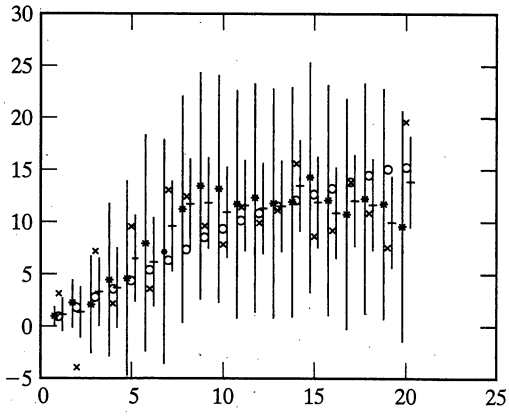

Figure 17.3 The Kalman filter for a point moving on the line under our model of constant velocity (compare with Figure 17.1). The state is plotted with open circles as a function of the step i. The *-s give $\overline{x}_i^-$, which is plotted slightly to the left of the state to indicate that the estimate is made before the measurement. The x-s give the measurements, and the +-s give $\overline{x}_i^+$, which is plotted slightly to the right of the state. The vertical bars around the *-s and the +-s are three standard deviation bars using the estimate of variance obtained before and after the measurement, respectively. When the measurement is noisy, the bars don't contract all that much when a measurement is obtained (compare with Figure 17.4).

and by pattern matching to our identities, we have

$$X_i^+ = \left(\frac{\overline{X_i}^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right);$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}.$$

Figure 17.3 shows a Kalman filter tracking a constant velocity model, and Figure 17.4 shows a Kalman filter tracking a constant acceleration model.

17.3.2 The Kalman Update Equations for a General State Vector

We obtained a 1D tracker without having to do any integration using special properties of normal distributions. This approach works for a state vector of arbitrary dimension, but the process of guessing integrals, and so on, is a good deal more elaborate than that shown in Section 17.3.1. We omit the necessary orgy of notation—it's a tough but straightforward exercise for those who really care (you should figure out the identities first and the rest follows)—and simply give the result in Algorithm 17.2.

17.3.3 Forward–Backward Smoothing

It is important to notice that $P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)$ is not the best available representation of $\boldsymbol{X}_i$; this is because it doesn't take into account the future behavior of the point. In particular, all the measurements *after* $\boldsymbol{y}_i$ could affect our representation of $\boldsymbol{X}_i$. This is because these future

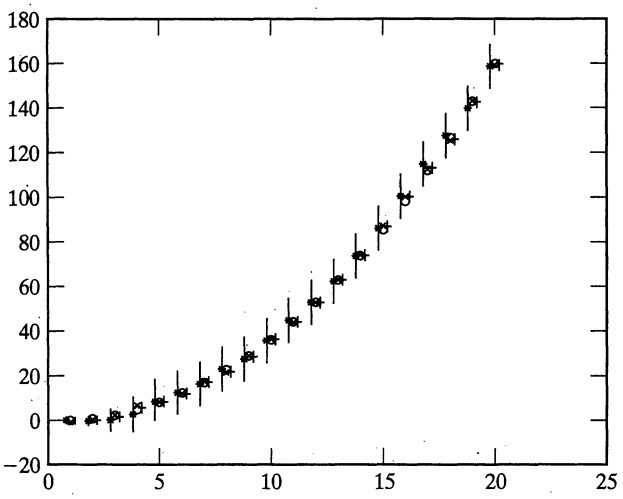

Figure 17.4 The Kalman filter for a point moving on the line under our model of constant acceleration (compare with Figure 17.2). The state is plotted with open circles as a function of the step i. The *-s give $\overline{\boldsymbol{x}}_i^-$, which is plotted slightly to the left of the state to indicate that the estimate is made before the measurement. The x-s give the measurements, and the +-s give $\overline{\boldsymbol{x}}_i^+$, which is plotted slightly to the right of the state. The vertical bars around the *-s and the +-s are three standard deviation bars using the estimate of variance obtained before and after the measurement, respectively. When the measurement is not noisy, the bars contract when a measurement is obtained.

Algorithm 17.2: The Kalman filter updates estimates of the mean and covariance of the various distributions encountered while tracking a state variable of some fixed dimension using the given dynamic model.

Dynamic Model:

$$\begin{aligned}\boldsymbol{x}_i &\sim N(\mathcal{D}_i\boldsymbol{x}_{i-1}, \Sigma_{d_i})\\ \boldsymbol{y}_i &\sim N(\mathcal{M}_i\boldsymbol{x}_i, \Sigma_{m_i})\end{aligned}$$

Start Assumptions: $\overline{\boldsymbol{x}}_0^-$ and Σ_0^- are known

Update Equations: Prediction

$$\begin{aligned}\overline{\boldsymbol{x}}_i^- &= \mathcal{D}_i\overline{\boldsymbol{x}}_{i-1}^+\\ \Sigma_i^- &= \Sigma_{d_i} + \mathcal{D}_i\sigma_{i-1}^+\mathcal{D}_i\end{aligned}$$

Update Equations: Correction

$$\begin{aligned}\mathcal{K}_i &= \Sigma_i^-\mathcal{M}_i^T\left[\mathcal{M}_i\Sigma_i^-\mathcal{M}_i^T + \Sigma_{m_i}\right]^{-1}\\ \overline{\boldsymbol{x}}_i^+ &= \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i\left[\boldsymbol{y}_i - \mathcal{M}_i\overline{\boldsymbol{x}}_i^-\right]\\ \Sigma_i^+ &= \left[Id - \mathcal{K}_i\mathcal{M}_i\right]\Sigma_i^-\end{aligned}$$

measurements might contradict the estimates obtained to date—perhaps the future movements of the point are more in agreement with a slightly different estimate of the position of the point. However, $P(X_i \mid y_0, \ldots, y_i)$ *is* the best estimate available at step i.

What we do with this observation depends on the circumstances. If our application requires an immediate estimate of position—perhaps we are tracking a car in the opposite lane—there isn't much we can do. If we are tracking off-line—perhaps for forensic purposes, we need the best estimate of what an object was doing given a videotape—then we can use all data points, and so we want to represent $P(X_i \mid y_0, \ldots, y_N)$. A common alternative is that we need a rough estimate immediately, and can use an improved estimate that has been time-delayed by a number of steps. This means we want to represent $P(X_i \mid y_0, \ldots, y_{i+k})$—we have to wait until time $i+k$ for this representation, but it should be an improvement on $P(X_i \mid y_0, \ldots, y_i)$.

Introducing a Backward Filter Now we have

$$\begin{aligned}
P(X_i \mid y_0, \ldots, y_N) &= \frac{P(X_i, y_{i+1}, \ldots, y_N \mid y_0, \ldots, y_i) P(y_0, \ldots, y_i)}{P(y_0, \ldots, y_N)} \\
&= \frac{P(y_{i+1}, \ldots, y_N \mid X_i, y_0, \ldots, y_i) P(X_i \mid y_0, \ldots, y_i) P(y_0, \ldots, y_i)}{P(y_0, \ldots, y_N)} \\
&= \frac{P(y_{i+1}, \ldots, y_N \mid X_i) P(X_i \mid y_0, \ldots, y_i) P(y_0, \ldots, y_i)}{P(y_0, \ldots, y_N)} \\
&= P(X_i \mid y_{i+1}, \ldots, y_N) P(X_i \mid y_0, \ldots, y_i) \alpha
\end{aligned}$$

where

$$\alpha = \left(\frac{P(y_{i+1}, \ldots, y_N) P(y_0, \ldots, y_i)}{P(X_i) P(y_0, \ldots, y_N)} \right).$$

This term should look like a potential source of problems to you; in fact, we can avoid tangling with it by a clever trick. What is important about this form is that we are combining $P(X_i \mid y_0, \ldots, y_i)$—which we know how to obtain—with $P(X_i \mid y_{i+1}, \ldots, y_N)$. We actually know how to obtain a representation of $P(X_i \mid y_{i+1}, \ldots, y_N)$, too. We could simply run the Kalman filter *backward* in time, using *backward dynamics* and take the predicted representation of X_i (we leave the details of relabeling the sequence, etc., to the exercises).

Combining Representations Now we have two representations of X_i: one obtained by running a forward filter and incorporating all measurements up to y_i; and one obtained by running a backward filter and incorporating all measurements after y_i. We need to combine these representations. Instead of explicitly determining the value of α (which should look hard), we can get the answer by noting that *this is like having another measurement.* In particular, we have a new measurement generated by X_i—that is, the result of the backward filter—to combine with our estimate from the forward filter. We know how to combine estimates with measurements because that's what the Kalman filter equations are for.

All we need is a little notation. We attach the superscript f to the estimate from the forward filter and the superscript b to the estimate from the backward filter. We write the mean of $P(X_i \mid y_0, \ldots, y_N)$ as $\overline{X}_i^*$ and the covariance of $P(X_i \mid y_0, \ldots, y_N)$ as Σ_i^*. We regard the representation of X_i^b as a measurement of X_i with mean $\overline{X}_i^{b,-}$ and covariance $\Sigma_i^{b,-}$—the minus sign is because the ith measurement cannot be used twice, meaning the backward filter predicts X_i using $y_N \ldots y_{i+1}$. This measurement needs to be combined with $P(X_i \mid y_0, \ldots, y_i)$, which has mean

$\overline{\boldsymbol{X}}_i^{f,+}$ and covariance $\Sigma_i^{f,+}$ (when we substitute into the Kalman equations, these take the role of the representation *before* a measurement because the value of the measurement is now $\overline{\boldsymbol{X}}_i^{b,-}$).

Substituting into the Kalman equations, we find that

$$\begin{aligned}
\mathcal{K}_i^* &= \Sigma_i^{f,+}\left[\Sigma_i^{f,+} + \Sigma_i^{b,-}\right]^{-1};\\
\Sigma_i^* &= [I - \mathcal{K}_i]\,\Sigma_i^{+,f};\\
\overline{\boldsymbol{X}}_i^* &= \overline{\boldsymbol{X}}_i^{f,+} + \mathcal{K}_i^*\left[\overline{\boldsymbol{X}}_i^{b,-} - \overline{\boldsymbol{X}}_i^{f,+}\right].
\end{aligned}$$

It turns out that a little manipulation (exercise!) yields a simpler form, which we give in Algorithm 17.3. Forward–backward estimates can make a substantial difference as Figure 17.5 illustrates.

Algorithm 17.3: The forward–backward algorithm combines forward and backward estimates of state to come up with an improved estimate.

Forward filter: Obtain the mean and variance of $P(\boldsymbol{X}_i \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)$ using the Kalman filter. These are $\overline{\boldsymbol{X}}_i^{f,+}$ and $\Sigma_i^{f,+}$.

Backward filter: Obtain the mean and variance of $P(\boldsymbol{X}_i \mid \boldsymbol{y}_{i+1}, \ldots, \boldsymbol{y}_N)$ using the Kalman filter running backward in time. These are $\overline{\boldsymbol{X}}_i^{b,-}$ and $\Sigma_i^{b,-}$.

Combining forward and backward estimates: Regard the backward estimate as a new measurement for $\boldsymbol{X}_i$, and insert into the Kalman filter equations to obtain

$$\begin{aligned}
\Sigma_i^* &= \left[(\Sigma_i^{f,+})^{-1} + (\Sigma_i^{b,-})^{-1}\right]^{-1};\\
\overline{\boldsymbol{X}}_i^* &= \Sigma_i^*\left[(\Sigma_i^{f,+})^{-1}\overline{\boldsymbol{X}}_i^{f,+} + (\Sigma_i^{b,-})^{-1}\overline{\boldsymbol{X}}_i^{b,-}\right].
\end{aligned}$$

Priors In typical vision applications, we are tracking forward in time. This leads to an inconvenient asymmetry: We may have a good idea of where the object started, but only a poor one of where it stopped (i.e., we are likely to have a fair prior for $P(\boldsymbol{x}_0)$, but may have difficulty supplying a prior for $P(\boldsymbol{x}_N)$ for the forward–backward filter). One option is to use $P(\boldsymbol{x}_N \mid \boldsymbol{y}_0, \ldots, \boldsymbol{y}_N)$ as a prior. This is a dubious act as this probability distribution does not in fact reflect our prior belief about $P(\boldsymbol{x}_N)$—we've used all the measurements to obtain it. The consequences can be that this distribution understates our uncertainty in $\boldsymbol{x}_N$ and so leads to a forward–backward estimate that significantly underestimates the covariance for the later states. An alternative is to use the mean supplied by the forward filter, but enlarge the covariance substantially; the consequences are a forward–backward estimate that overestimates the covariance for the later states (compare Figure 17.5 with Figure 17.6).

Not all applications have this asymmetry. For example, if we are engaged in a forensic study of a videotape, we might be able to start both the forward tracker and the backward tracker by hand and provide a good estimate of the prior in each case. If this is possible, then we have a good deal more information which may be able to help choose correspondences, and so on—the forward tracker should finish rather close to where the backward tracker starts.

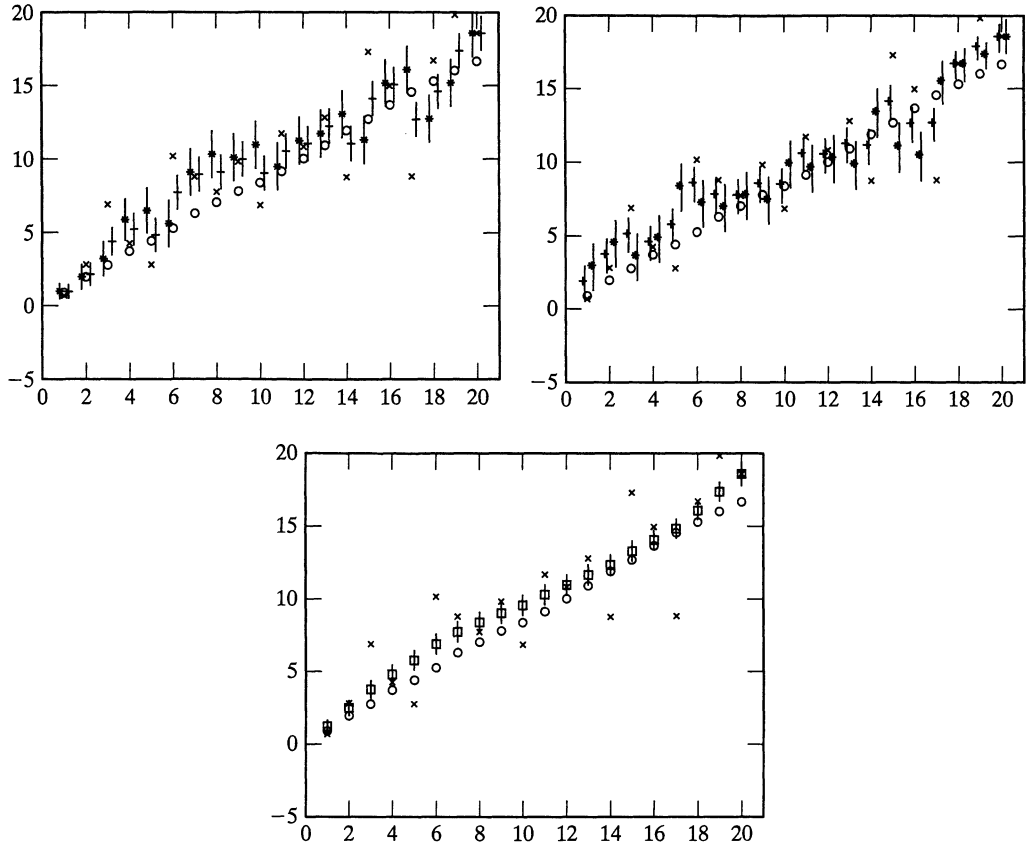

Figure 17.5 Forward–backward estimation for a dynamic model of a point moving on the line with constant velocity. We are plotting the position component of state against time. On the **top left**, we show the forward estimates, again using the convention that the state is shown with circles, the data is shown with an x, the prediction is shown with a *, and the corrected estimate is shown with a +; the bars give one standard deviation in the estimate. The predicted estimate is shown slightly behind the state and the corrected estimate is shown slightly ahead of the state. You should notice that the measurements are noisy. On the **top right** we show the backward estimates. Now time is running backward (although we have plotted both curves on the same axis) so that the prediction is slightly ahead of the measurement and the corrected estimate is slightly behind. We have used the final corrected estimate of the forward filter as a prior. Again, the bars give one standard deviation in each variable. On the **bottom**, we show the combined forward–backward estimate. The squares give the estimates of state. Notice the significant improvement in the estimate.

Smoothing over an Interval Although our formulation of forward–backward smoothing assumed that the backward filter started at the last data point, it is easy to start this filter a fixed number of steps ahead of the forward filter. If we do this, we obtain an estimate of state in real time (essentially immediately after the measurement) and an improved estimate some fixed numbers of measurements later. This is sometimes useful. Furthermore, it is an efficient way to obtain most of the improvement available from a backward filter if we can assume that the effect of the distant future on our estimate is relatively small compared with the effect of the immediate

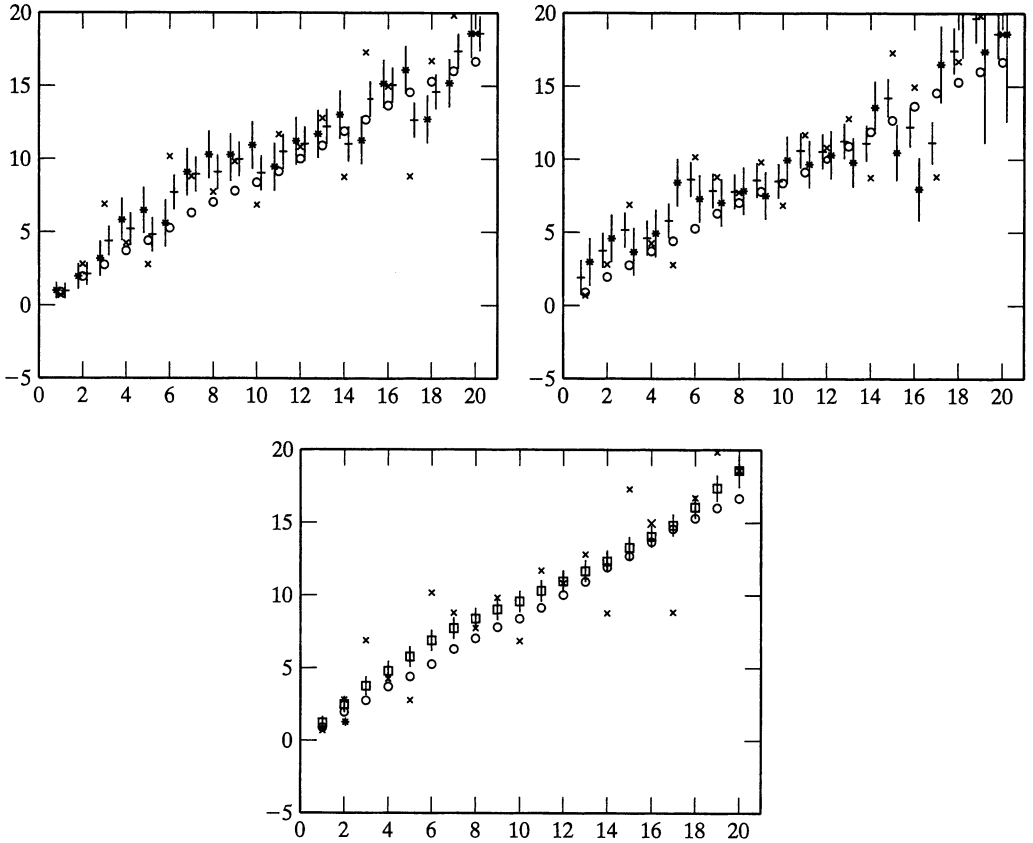

Figure 17.6 We now show the effects of using a diffuse prior for the position of the final point in forward–backward estimation for a dynamic model of a point moving on the line with constant velocity. We are plotting the position component of state against time. On the **top left**, we show the forward estimates, again using the convention that the state is shown with circles, the data is shown with an x, the prediction is shown with a *, and the corrected estimate is shown with a +; the bars give one stanlard deviation in the estimate. The predicted estimate is shown slightly behind the state, and the corrected estimate is shown slightly ahead of the state. You should notice that the measurements are noisy. On the **top right** we show the backward estimates. Now time is running backward (although we have plotted both curves on the same axis) so that the prediction is slightly ahead of the measurement and the corrected estimate is slightly behind. Again, the bars give one standard deviation in each variable. On the **bottom**, we show the combined forward–backward estimate. The squares give the estimates of state. Notice the significant improvement in the estimate.

future. Notice that we need to be careful about priors for the backward filter here; we might take the forward estimate and enlarge its covariance somewhat.

17.4 DATA ASSOCIATION

Not every aspect of every measurement conveys information about the state of the object being tracked. In fact, we have been somewhat disingenuous up to this point and have not really talked

about what is in $\mathbf{y}_i$ at all. Usually, there are measurements that are informative and measurements that are not informative (usually referred to as **clutter**).

Determining which measurements are informative is usually referred to as **data association**. Typically, one wishes to map a series of measurements to a series of tracks, possibly ignoring some—or almost all—of them. The main work in this problem relates to tracking moving objects (aeroplanes, missiles, etc., all conveniently belonging to the bad guys) with radar returns. Typically, there may be many radar returns at any given timestep—we should like to update our representations of the motion of the objects being tracked without necessarily knowing which returns come from which object. As we have seen, tracking algorithms are complicated, but not particularly difficult. Data association is probably the biggest source of difficulties in vision applications, and is not often discussed in the literature. We expect this to change under the impact of practical applications. We will confine our discussion to the case where there is a single moving object. The problem here is that some pixels in the image are very informative about that object, and some are not—which should we use to guide our tracking process?

17.4.1 Choosing the Nearest—Global Nearest Neighbours

In the easiest case, we need to track a single object moving in clutter. For example, we might be tracking a ball moving on a fixed or very slowly varying background. We segment the image into regions, with the reasonable expectation that the ball tends to produce one region, and that the segmentation of the background might change with time. Intuitively, it would be very difficult to confuse the ball with a background region, because we have a strong model of how the ball is moving. This means we would have to be unlucky if there was a new background region that (a) was easily confused with the ball region and (b) confused the dynamic model. This suggests one fairly popular strategy for data association: the rth region offers a measurement $\mathbf{y}_i^r$, and we choose the region with the best value of

$$\begin{aligned} P(\mathbf{Y}_i = \mathbf{y}_i^r \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1}) &= \int P(\mathbf{Y}_i = \mathbf{y}_i^r \mid \mathbf{X}_i, \mathbf{y}_0, \ldots, \mathbf{y}_{i-1}) P(\mathbf{X}_i \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})\, d\mathbf{X}_i \\ &= \int P(\mathbf{Y}_i = \mathbf{y}_i^r \mid \mathbf{X}_i) P(\mathbf{X}_i \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})\, d\mathbf{x}_i \end{aligned}$$

Determining $P(\mathbf{Y}_i = \mathbf{y}_i \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})$ is a particularly easy calculation with the Kalman filter. We know how $\mathbf{Y}_i$ is obtained from $\mathbf{X}_i$—we take a normal random variable with mean $\overline{\mathbf{X}_i^-}$, and covariance Σ_i^-, apply the linear operator $\mathcal{D}_i$ to it, and add some other random variable. The output of the linear operator must have mean $\mathcal{D}_i\overline{\mathbf{X}_i^-}$ and covariance $\mathcal{D}_i\Sigma_i^-\mathcal{D}^T$. To this we are going to add a random variable with zero mean and covariance Σ_{m_i}; the result must have mean

$$\mathcal{D}_i\overline{\mathbf{X}_i^-}$$

and covariance

$$\mathcal{D}_i\Sigma_i^-\mathcal{D}^T + \Sigma_{m_i}.$$

In Figure 17.7, we have plotted bounds on the position of an expected measurement for a Kalman filter following various dynamic models.

Notice that this strategy can be relatively robust depending on the accuracy of the dynamic model. If we are able to use a tight dynamic model, anything that is easily confused with the object being tracked must be more similar to the predicted measurement than the real object does. This means that an occasional misidentification may not create major problems because one is unlikely to find a region that is both similar to the predicted measurement and able to throw off the dynamic model badly. In Figure 17.8, we show a Kalman filter tracking the state of

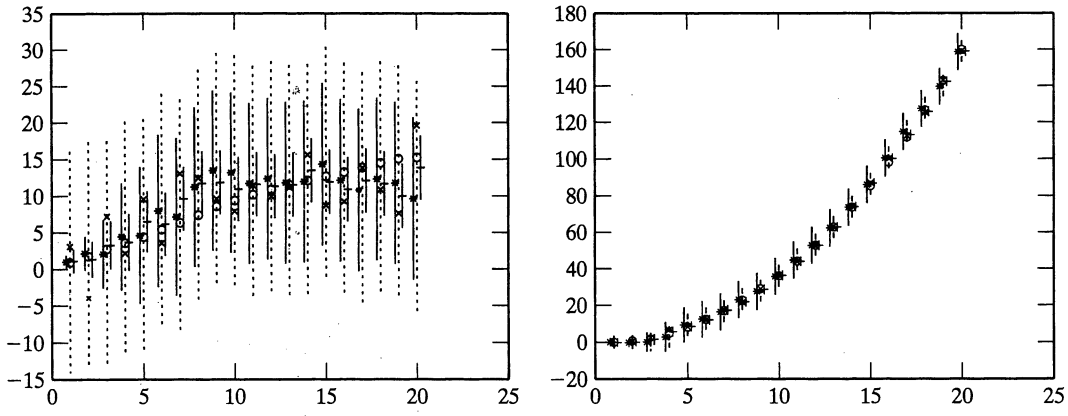

Figure 17.7 Data association for a Kalman filter for a point moving on the line under our model of constant velocity on the **left** and constant acceleration on the **right**. Compare with Figure 17.1 for the constant velocity model and with Figure 17.2 for the constant acceleration model. We have used the conventions of Figure 17.3. We have now overlaid three standard deviation bars for the measurement (the dashed bars passing through the state). These are obtained using the estimate of state before a measurement and our knowledge of the variance of the measurement process. Notice that the measurements lie within these windows.

a point by choosing the best measurement at each step; it does not always correctly identify the point, but its estimate of state is always good.

Notice that what we are doing here is using only measurements that are consistent with our predictions. This may or may not be dangerous: It can be easy to track nonexistent objects this way or to claim to be tracking an object without ever obtaining a measurement from it. If the dynamic model can give only weak predictions (i.e., the object doesn't really behave like that or Σ_{d_i} is consistently large) we may have serious problems because we need to rely on the measurements. These problems occur because the error can accumulate—it is now relatively easy to continue tracking the wrong point for a long time, and the longer we do this the less chance there is of recovering the right point. Figure 17.9 shows a Kalman filter becoming hopelessly confused in this manner.

17.4.2 Gating and Probabilistic Data Association

Again, we assume that we are tracking a single object in clutter and use the example of tracking a ball moving on a fixed or slowly varying background. Instead of choosing the region most like the predicted measurement, we could exclude all regions that are too different and then use all others, weighting them according to their similarity to the prediction.

The first step is called *gating*. We exclude all measurements that are too different from the predicted measurement. What "too different" means rather depends on the application: If we are too aggressive in excluding measurements, we may find nothing left. It is usual to exclude measurements that lie more than some number—commonly, three—of standard deviations from the predicted mean. A more sophisticated strategy is required if the object being tracked has more than one dynamic behavior; for example, military aircraft often engage in high-speed maneuvers. In cases like this, it is common to have several gates and to take all measurements that lie in the tightest gate that contains any measurements.

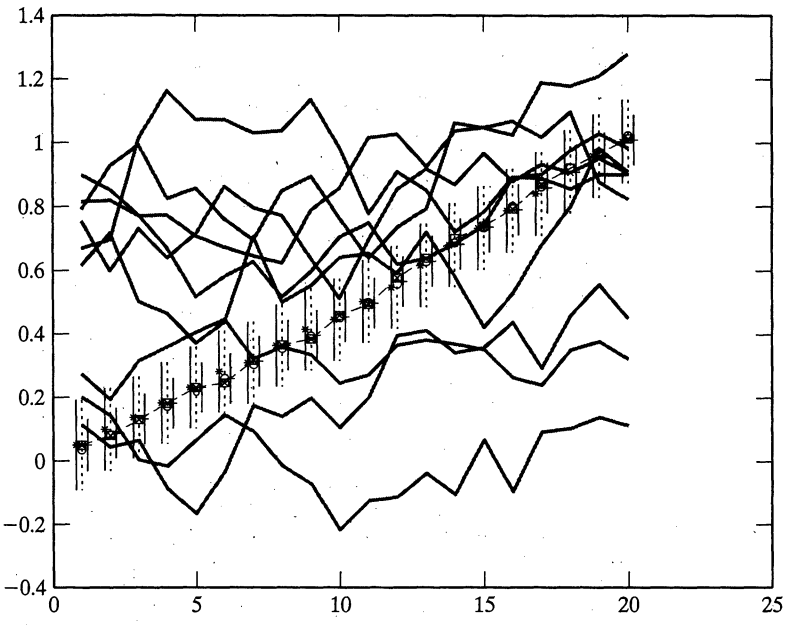

Figure 17.8 Predictions of the point position can identify "good" measurements for a Kalman filter. We are using a Kalman filter to identify a point, moving with constant velocity on a line, and with a small Σ_{d_i} at every stage. There are also 10 drifting points. This plot shows position plotted against time for the drifting points—which are shown with a solid line—and for the point that is being tracked. The trajectory of the point that should be tracked is shown in a dashed line, and each measurement on this trajectory is shown with a square. We have used the conventions of Figure 17.3 (i.e., the state is plotted with open circles, as a function of the step i; * gives $\overline{x_i}^-$, which is plotted slightly to the left of the state to indicate that the estimate is made before the measurement; x gives the measurements, and + gives $\overline{x}_i^+$, which is plotted slightly to the right of the state; the vertical bars around the * and the + are three standard deviation bars using the estimate of variance obtained before and after the measurement, respectively; we have overlaid one standard deviation bars in each case). This filter chooses the measurement at each step by choosing the measurement that maximizes $P(y_i^r | y_0, \ldots, y_{i-1})$; notice that it doesn't choose the right measurement at every step (i.e., the x is not always in the square), but it maintains a good estimate of the state (i.e., the +'s are close to the circles).

The next step is called *probabilistic data association* (PDA). Assume that, in the gate, we have a set of N regions each producing a vector of measurements $\mathbf{y}_i^k$, where the superscript indicates the region. We have a set of possible hypotheses to deal with: Either no region comes from the object, which we call h_0, or region k comes from the object, which we call h_k. The measurement we report is

$$\mathrm{E}_h\left[\mathbf{y}_i\right] = \sum_j P(h_j \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})\mathbf{y}_i^j,$$

where the expectation is taken over the space of hypotheses (which is why we have given it the subscript h). Now the probability that none of the measurements comes from the object depends on the details of our detection process. For some detection processes, this parameter can be calculated; for example, in chapter 4 of Blackman and Popoli (1999), there is a worked example

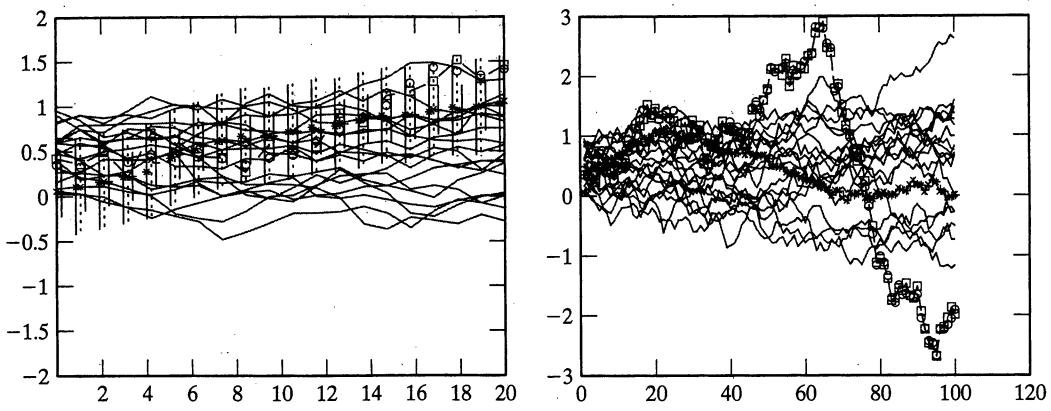

Figure 17.9 If the dynamic model is not sufficiently constrained, then choosing the measurement that gives the best $P(\mathbf{y}_i^r | \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})$ can lead to disaster. On the **left**, 20 steps of a Kalman filter following a point moving periodically on the line with 20 drifting points in the background. We are using the conventions of Figure 17.3 again. Now Σ_{d_i} is relatively large for each step, and so it is easy to follow the wrong measurement for some way. It looks as if the filter is tracking the state well, but in fact as the figure on the **right**—which gives 100 steps—shows, it quickly becomes hopelessly lost.

for a radar system. In other cases, we need to search for a value of the parameter that results in good behavior on a set of training examples. Assume that we have calculated or learned this parameter, which we can write as β. We must also assume that either the object is not detected or only one measurement comes from the object. Now

$$\begin{aligned} P(h_j \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1}) &= \int P(h_j \mid \mathbf{X}_i) P(\mathbf{X}_i \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1}) d\mathbf{X}_i \\ &\doteq P(\mathbf{Y}_i = \mathbf{y}_i^j \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1}) P(\text{object detected}) \\ &= P(\mathbf{Y}_i = \mathbf{y}_i^j \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})(1 - \beta). \end{aligned}$$

In what follows, we write $P(h_j \mid \mathbf{y}_0, \ldots, \mathbf{y}_{i-1})$ as p_j. In practice, this method is usually used with a Kalman filter. To do so, we report the measurement

$$\mathbf{y}_i' = \sum_j p_j \mathbf{y}_i^j$$

to the Kalman update equations. Note that the term for not having a measurement appears here as the factor $(1 - \beta)$ in the expressions for the p_j, but our uncertainty about which measurement should contribute to the update should also appear in the covariance update. We modify the covariance update equations to take the form

$$\begin{aligned} \Sigma_i^+ = (1 - \beta)\left[Id - \mathcal{K}_i \mathcal{M}_i\right] \Sigma_i^- + \beta \Sigma_i^- \\ + \mathcal{K}_i \left[\sum_j p_j (\mathcal{H}_i \overline{\mathbf{x}}_i^- - \mathbf{y}_i^j)(\mathcal{H}_i \overline{\mathbf{x}}_i^- - \mathbf{y}_i^j)^T - \mathbf{y}_i' (\mathbf{y}_i')^T \right]. \end{aligned}$$

Here the first term is the update for the standard Kalman filter weighted by the probability that one observation is good, the second term deals with the prospect that all observations are bad, and the third term contributes uncertainty due to the correspondence uncertainty.

17.5 APPLICATIONS AND EXAMPLES

Tracking is a technology with a number of possible applications. There are three dominant topics.

- **Vehicle tracking** systems could report traffic congestion, accidents, and dangerous or illegal behavior by road users. Traffic congestion reports are useful for potential road users—who might change their travel plans—and to authorities—who might arrange to remove immobilised vehicles blocking lanes, etc. Accident reports can be used to alert emergency services; if the tracking system can read vehicle number plates, it might use reports of dangerous or illegal behavior to send a summons to the vehicle owner.
- **Surveillance** systems report what people are doing, usually with the aim of catching people who are doing things they shouldn't. The police might wish to know which member of a sports audience threw a bottle onto the field, for example, or if the same person visited several different banks just before they were robbed. Customs might wish to know exactly who is loading and unloading aircraft flying to foreign ports.
- **Human–computer interaction** systems use people's actions to drive various devices. For example, the living room might decide for itself, by watching what people are doing, when low lights and soft music are appropriate. The television set might change channels when you wave at it. Your computer might watch what you write on your whiteboard and make a record of the contents when you tell it to.

Currently, the most convincing applications are in vehicle tracking. These systems work reliably under a large range of circumstances. We survey vehicle tracking systems briefly here, and then we discuss human trackers in a chapter that appears on the website.

17.5.1 Vehicle Tracking

Systems that can track cars using video from fixed cameras can be used to predict traffic volume and flow; the ideal is to report on and act to prevent traffic problems as quickly as possible. A number of systems can track vehicles successfully. The crucial issue is initiating a track automatically. In the two systems we describe here, the problem is attacked quite differently. Sullivan, Baker, Worrall, Attwood and Remagnino (1997) construct a set of regions of interest (ROIs) in each frame. Because the camera is fixed, these regions of interest can be chosen to span each lane (Figure 17.10); this means that almost all vehicles must pass directly through a region of interest in a known direction (there are mild issues if a vehicle chooses to change lanes while in the ROI, but these can be ignored). Their system then watches for characteristic edge signatures in the ROI that indicate the presence of a vehicle (Figure 17.10). These signatures can alias slightly—typically, a track is initiated when the front of the vehicle enters the ROI, another is initiated when the vehicle lies in the ROI, and a third is initiated close to the vehicle's leaving—because some of the vehicle's edges are easily mistaken for others.

Each initiated track is tracked for a sequence of frames, during which time it accumulates a quality score—essentially, an estimate of the extent to which predictions of future position were accurate. If this quality score is sufficiently high, the track is accepted as an hypothesis. An exclusion region in space and time is constructed around each hypothesis, such that there can be only one track in this region; if the regions overlap, the track with the highest quality is chosen. The requirement that the exclusion regions do not overlap derives from the fact that two cars can't occupy the same region of space at the same time. Once a track has passed these tests, the position in which and the time at which it will pass through another ROI can be predicted. The track is finally confirmed or rejected by comparing this ROI at the appropriate time with a template that predicts the car's appearance. Typically, relatively few tracks that are initiated reach this stage (Figure 17.11).

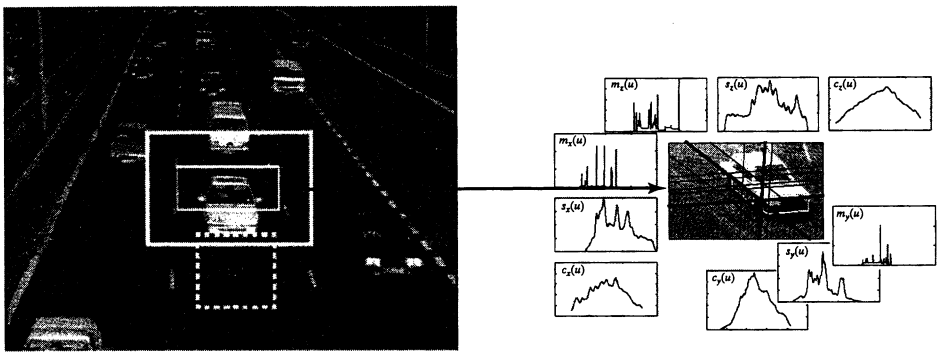

Figure 17.10 Sullivan *et al.* track vehicles in views of the road from a stationary camera. Their tracker uses a series of regions of interest registered to the road, which are shown on the **left**. They initiate tracks by looking for characteristic edge signatures in a particular ROI; these signatures are projected onto three distinct coordinate axes—if the edges projected on these axes have a high enough correlation with the expected form, then a track is initiated (**right**). *Reprinted from "Model-based Vehicle Detection and Classification using Orthographic Approximations," by G D Sullivan, et al., Proc. British Machine Vision Association Conference, 1996, by permission of the K.D. Baker*

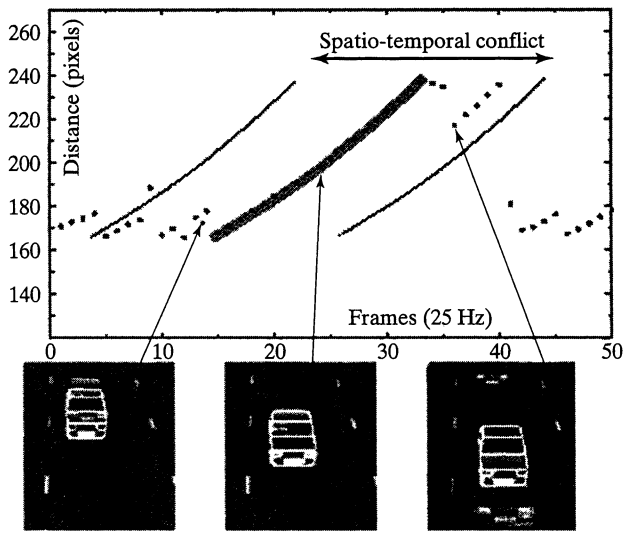

Figure 17.11 In the system of Sullivan *et al.*, tracks are continued if they are of sufficient quality measured by comparing the prediction of the track with measurements. Tracks exclude other tracks: By the time a car reaches the bottom of the view, the system must decide which track to accept. It does so by comparing the track prediction with another ROI. This figure plots a series of tracks (position on the vertical axis and time on the horizontal axis). Notice that the typical alias tracks (that arise because the front of a car and the back of a car both tend to look rather like a registered car to the track initiation process) tend to die out quite quickly; the real track (and its exclusion regions) is indicated. If two tracks attempt to exclude one another, the winner is the track of the highest quality. *Reprinted from "Model-based Vehicle Detection and Classification using Orthographic Approximations," by G D Sullivan, et al., Proc. British Machine Vision Association Conference, 1996, by permission of the K.D. Baker*

An alternative method for initiating car tracks is to track individual features and then group those tracks into possible cars. Beymer, McLauchlan, Coifman and Malik (1997) used this strategy rather successfully. Because the road is plane and the camera is fixed, the homography connecting the road plane and the camera can be determined. This homography can be used to determine the distance between points; and points can lie together on a car only if this distance doesn't change with time. Their system tracks corner points, identified using a second moment matrix (Section 8.3.3), using a Kalman filter. Points are grouped using a simple algorithm using a graph abstraction: Each feature track is a vertex, and edges represent a grouping relationship between the tracks. When a new feature comes into view—and a track is thereby initiated—it is given an edge joining it to every feature track that appears nearby in that frame. If, at some future time, the distance between points in a track changes by too much, the edge is discarded. An exit region is defined near where vehicles leave the frame. When tracks reach this exit region, connected components are defined to be vehicles. This grouper is successful, both in example images (Figure 17.12) and in estimating traffic parameters over long sequences (Figure 17.13).

The ground plane to camera transformation can provide a great deal of information. We have already used this to determine whether points are on rigid objects (by figuring out velocity on the ground plane and comparing velocities). This allowed us to assemble features into objects. Now once an object has been tracked, we can use this transformation to reason about spatial layout and occlusion. Furthermore, we can track cars from moving vehicles. In this case, there are two issues to manage: First, the motion of the camera platform (so-called *ego-motion*); and second, the motion of other vehicles. Ferryman, Maybank and Worrall (2000) estimate the ego-motion by matching views of the road to one another from frame to frame (Figure 17.14). With an estimate of the homography and of the ego-motion, we can now refer tracks of other moving vehicles into the road coordinate system to come up with reconstructions of all vehicles visible on the road from a moving vehicle (Figure 17.14).

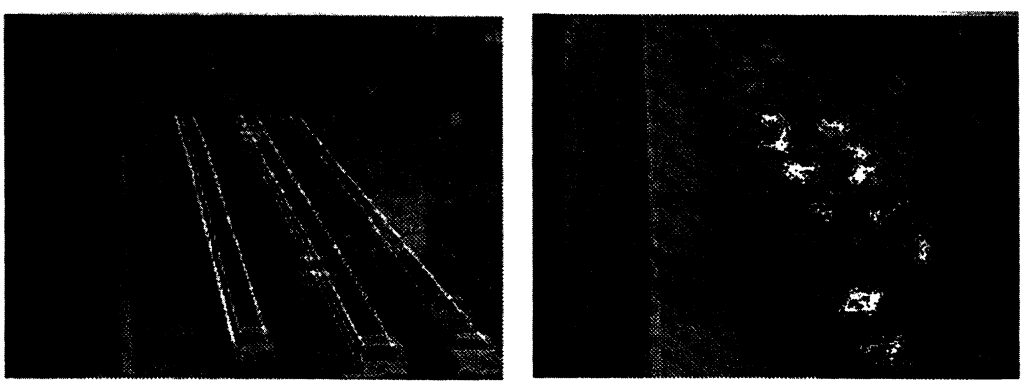

Figure 17.12 The figure on the **left** shows individual tracks for the system of Beymer et al. These tracks are obtained by tracking corner points with a Kalman filter. Because the camera position with respect to the road plane is known, the camera transformation can be inverted for points lying on a plane parallel to the road plane. This means that we can determine pairs of points that remain at a constant distance from one another. The figure on the **right** shows groups of such points. These groups are assumed to represent vehicles. *Reprinted from "A Real-Time Computer Vision System for Measuring Traffic Parameters," by D. Beymer et al., Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1997 © 1997, IEEE*

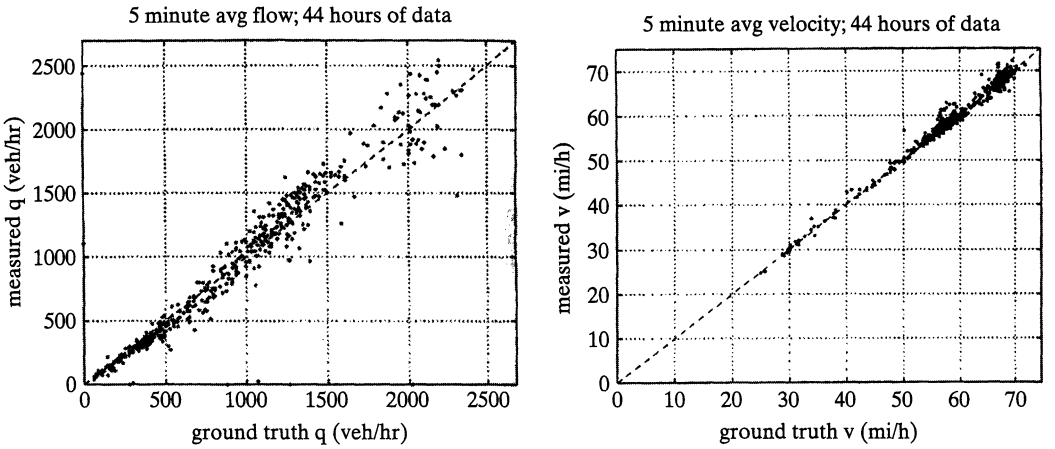

Figure 17.13 The system of Beymer et al. can produce rather accurate estimates of traffic flow and traffic velocity. On the **left**, a scatter plot of estimates of flow versus ground truth; on the **right**, a scatter plot of estimates of velocity vs. ground truth. *Reprinted from "A Real-Time Computer Vision System for Measuring Traffic Parameters," by D. Beymer et al., Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1997 © 1997, IEEE*

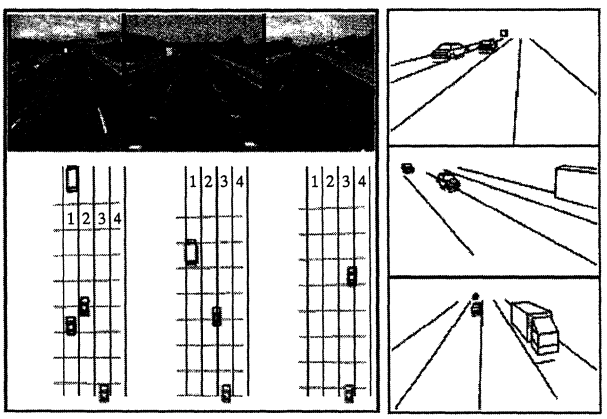

Figure 17.14 Once we know the homography to a ground plane, tracks of other vehicles obtained using the moving camera platform can be referred to the coordinate system relative to the camera platform on this ground plane. This allows detailed reconstructions of traffic geometry illustrated on the **left**. Furthermore, we can use the movement of fixed objects on the ground plane (such as the white marks) to estimate the movement of the camera platform. All this means that we can (a) interpret traffic geometry, for example, predicting impending collision between the camera platform and some other vehicle, and (b) render views of the traffic from some other platform (shown on the **right**). *Reprinted from "Visual Surveillance for Moving Vehicles," by J.M. Ferryman, S.J. Maybank and A.D. Worrall, Proc. 1998 IEEE Workshop on Visual Surveillance, © 1998, IEEE*

17.6 NOTES

The Kalman filter is an extremely useful trick. It is regularly rediscovered, and appears in different guises in different fields. Often dynamics that are not linear can be represented as linear dynamics well enough to fit a Kalman filter. We refer interested readers to Chui (1991), Staff of the Analytical Sciences Corporation (1974) and West and Harrison (1997).

We have not discussed the process of fitting a linear dynamic model. The matter is relatively straightforward *if* one knows the order of the model, a natural state space to use, and a reasonable measurement model. Otherwise, things get tricky—there is an entire field of control theory dedicated to the topic in this case known as *system identification*. We recommend, in the first instance, Ljung (1995).

PROBLEMS

17.1. Assume we have a model $\boldsymbol{x}_i = \mathcal{D}_i\boldsymbol{x}_{i-1}$ and $y_i = \boldsymbol{M}_i^T\boldsymbol{x}_i$. Here the measurement y_i is a one-dimensional vector (i.e., a single number) for each i and $\boldsymbol{x}_i$ is a k-dimensional vector. We say model is *observable* if the state can be reconstructed from any sequence of k measurements.

(a) Show that this requirement is equivalent to the requirement that the matrix

$$\left[\boldsymbol{M}_i\mathcal{D}_i^T\boldsymbol{M}_{i+1}\mathcal{D}_i^T\mathcal{D}_{i+1}^T\boldsymbol{M}_{i+2}\ldots\mathcal{D}_i^T\ldots\mathcal{D}_{i+k-2}^T\boldsymbol{M}_{i+k-1}\right]$$

has full rank.

(b) The point drifting in 3D, where $\mathcal{M}_{3k} = (0, 0, 1)$, $\mathcal{M}_{3k+1} = (0, 1, 0)$, and $\mathcal{M}_{3k+2} = (1, 0, 0)$ is observable.

(c) A point moving with constant velocity in any dimension, with the observation matrix reporting position only, is observable.

(d) A point moving with constant acceleration in any dimension, with the observation matrix reporting position only, is observable.

17.2. A point on the line is moving under the drift dynamic model. In particular, we have $x_i \sim N(x_{i-1}, 1)$. It starts at $x_0 = 0$.

(a) What is its average velocity? (Remember, velocity is *signed*.)

(b) What is its average speed? (Remember, speed is *unsigned*.)

(c) How many steps, on average, before its distance from the start point is greater than two (i.e., what is the expected number of steps, etc.?)

(d) How many steps, on average, before its distance from the start point is greater than ten (i.e., what is the expected number of steps, etc.)?

(e) (This one requires some thought.) Assume we have two nonintersecting intervals, one of length 1 and one of length 2; what is the limit of the ratio (average percentage of time spent in interval one)/ (average percentage of time spent in interval two) as the number of steps becomes infinite?

(f) You probably guessed the ratio in the previous question; now run a simulation and see how long it takes for this ratio to look like the right answer.

17.3. We said that

$$g(x; a, b)g(x; c, d) = g\left(x; \frac{ad + cb}{b + d}, \frac{bd}{b + d}\right) f(a, b, c, d).$$

Show that this is true. The easiest way to do this is to take logs and rearrange the fractions.

17.4. Assume that we have the dynamics

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i}^2);$$

$$y_i \sim N(m_i x_i, \sigma_{m_i}^2).$$

(a) $P(x_i \mid x_{i-1})$ is a normal density with mean $d_i x_{i-1}$ and variance $\sigma_{d_i}^2$. What is $P(x_{i-1} \mid x_i)$?
(b) Now show how we can obtain a representation of $P(\boldsymbol{x}_i \mid y_{i+1}, \ldots, y_N)$ using a Kalman filter.

Programming Assignments

17.5. Implement a 2D Kalman filter tracker to track something in a simple video sequence. We suggest that you use a background subtraction process and track the foreground blob. The state space should probably involve the position of the blob, its velocity, its orientation—which you can get by computing the matrix of second moments—and its angular velocity.

17.6. If one has an estimate of the background, a Kalman filter can improve background subtraction by tracking illumination variations and camera gain changes. Implement a Kalman filter that does this; how substantial an improvement does this offer? Notice that a reasonable model of illumination variation has the background multiplied by a noise term that is near one—you can turn this into linear dynamics by taking logs.