Chapter 10

Motion extraction

Still images are not always appropriate as input to a vision system. For several applications it is crucial to understand about scene dynamics. Examples are robot or car navigation where one wants to monitor ego-motion with respect to the surroundings ot surveillance, where objects need to be tracked.

A first section looks at simply detecting that there is motion. A second section goes a step further and extracts a motion vector at every pixel, i.e. it detects the projection of the 3D motion of objects onto the image plane, the so-called 'optical flow'. A third section deals with tracking, i.e. determining the trajectory of an object throughout a video sequence.

10.1 I - Motion detection

Not available yet

10.2 I - Optical flow

The motion of grey level patterns as observed in the image when a camera is moving relative to the objects in a scene or v.v. is called the *optical flow*. Ideally, the optical flow will correspond to the projection of the three-dimensional velocity vectors on the image plane, but this need not be the case. Consider first a perfectly uniform sphere rotating about an axis through its center. Assume the axis remains in a fixed position. If the sphere is perfectly smooth (no markings or visible texture), the shading on the sphere will not change. So the apparent motion is zero since the whole grey level pattern stands still. Yet the sphere is in motion. On the other hand, now assume the sphere is still but the light source moves. This will result in changes in the shading pattern and, hence, a non-zero optical flow field. These examples make tracking of grey level patterns look questionable as a strategy for velocity extraction. Fortunately, in typical scenes the projected velocities and the motion of grey level patterns are well correlated. The method propounded in the sequel is supposed to work on subsequent frames of image sequences, implying that

normally intermediate changes in ambient lighting are negligible. Hence, tracking identical grey levels over time is a pragmatic solution to what would otherwise be a hard, if not insoluble, problem.

10.2.1 Determination of the optical flow

Suppose two subsequent frames out of a sequence have been taken. The determination of the optical flow amounts to assigning to each pixel in the first image the corresponding pixel *with the same grey level* in the second. The displacement between these pixels yields the local velocity. Denoting the grey level at (x, y) and at time t by $I(x, y, t)$ then yields the following constraint:

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \ .$$

Using the abbreviations $u = dx/dt$, $v = dy/dt$, $I_x = \partial I/\partial x$, $I_y = \partial I/\partial y$, $I_t = \partial I/\partial t$, we obtain

$$I_x u + I_y v + I_t = 0 \ .$$

The derivatives I_x, I_y, I_t are estimated from the image. The above equation is called the *optical flow constraint equation*, since it expresses a constraint on the components u and v of the optical flow. Notice that we have a linear relation between u and v. We can rewrite the constraint as

$$(I_x, I_y).(u, v) = -I_t \ .$$

We therefore may conclude that this linear relation fixes the component of optical flow in the direction of the brightness gradient (I_x, I_y) and the magnitude of this component is

$$\frac{I_t}{\sqrt{I_x^2 + I_y^2}} \ .$$

We cannot, however, determine the component of the optical flow at right angles to this direction. This underdetermined nature of our formulation is referred to as the *aperture problem*. Indeed, suppose a moving pattern is observed through an aperture, as shown in fig. 10.1. If only a straight edge is observed, then only the motion component perpendicular to the edge can be extracted. The motion component parallel to the edge remains unknown, as illustrated in fig. 10.2. Notice that if the edge is curved or has a corner, then both velocity components can be extracted (fig. 10.3). Using the optical flow constraint equation can be considered looking through such an aperture to a straight edge: The neighbourhood in which the derivatives are calculated serves as a mathematical "aperture" through which only a first order approximation to the grey level pattern is observed (taking only first derivatives I_x and I_y into account amounts to approximating this grey level pattern by a plane). Pursuing the analogy further, the aperture problem can sometimes be solved by considering higher derivatives of the grey levels than just the first, i.e. by assessing the curved structure of the local grey level pattern. On the other hand, the aperture problem

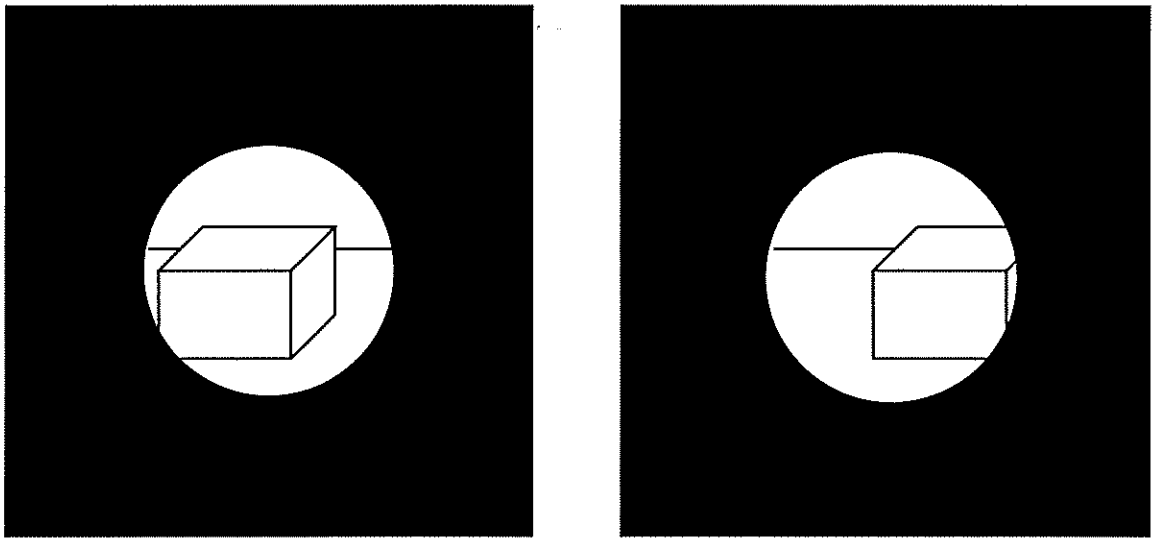


Figure 10.1: *The optical flow problem can be compared to looking at a moving pattern through an aperture and trying to determine the pattern's translational motion.*

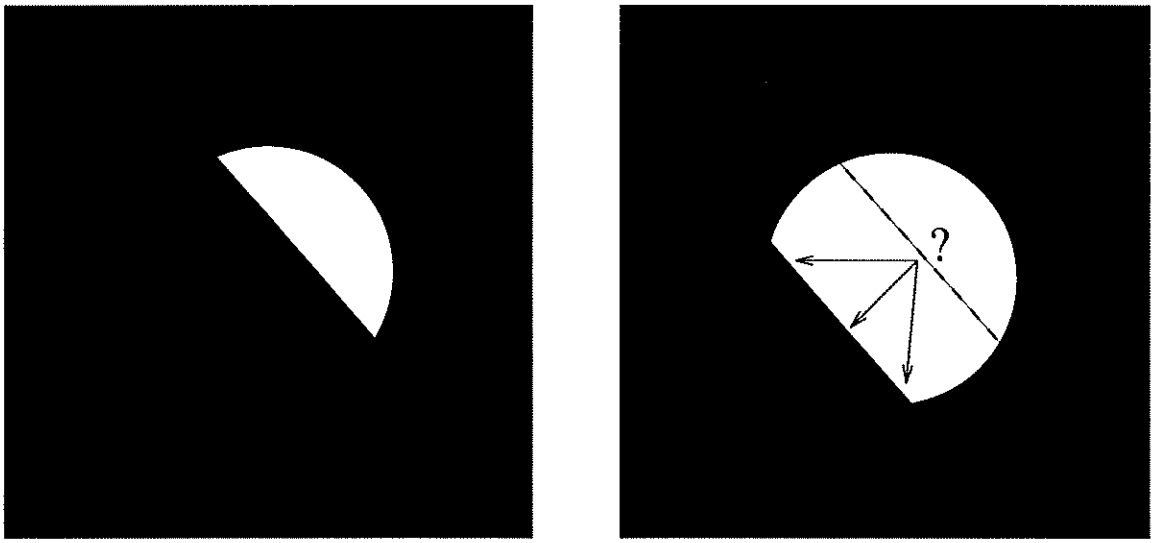


Figure 10.2: *If the pattern observed through the aperture has straight and parallel isophotes (i.e. lines of constant grey level), then only one component of the motion can be retrieved.*

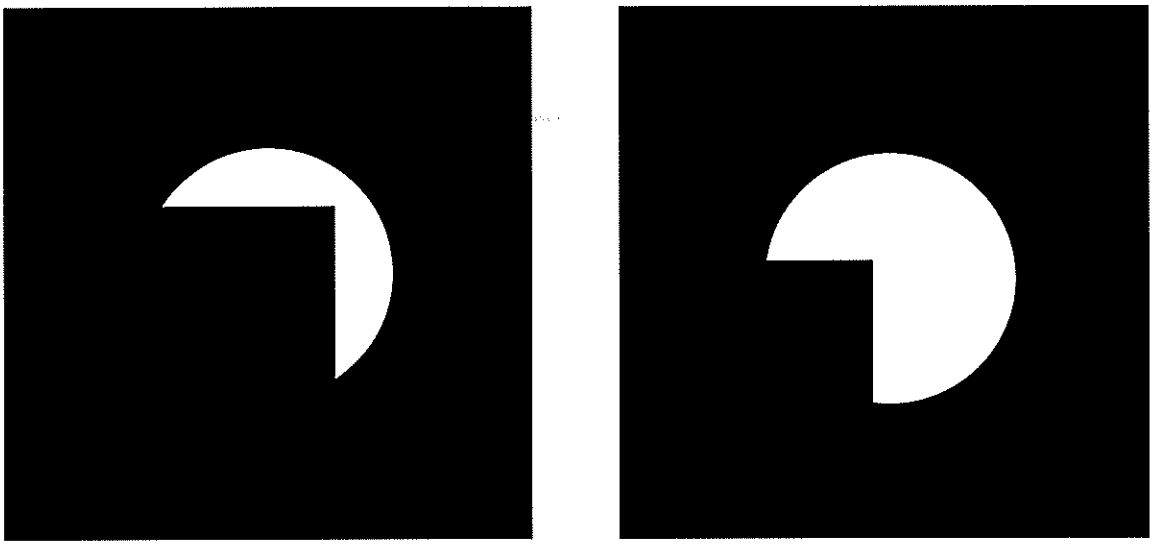


Figure 10.3: *Looking through an aperture, both components of translational motion can in principle be extracted for more complicated grey level patterns.*

is inherent to grey level profiles such as planar patches for which it cannot be solved in this manner.

The optical flow constraint equation yields one velocity component. In order to retrieve the other component, an additional constraint is introduced. Usually motion varies smoothly in most parts of an image. This observation underlies the usual choice for this constraint – referred to as a *smoothness constraint* – which is imposed by minimizing some measure for the changes in the flow field. An example of such measure is

$$e_s = \int\int ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))\ dxdy\ ,$$

with $u_x = \partial u/\partial x$, $u_y = \partial u/\partial y$, $v_x = \partial v/\partial x$, and $v_y = \partial v/\partial y$, i.e. the integral of the square of the magnitude of the gradient of the optical flow. On the other hand, the error in the optical flow constraint equation,

$$e_c = \int\int (I_x u + I_y v + I_t)^2\ dxdy\ ,$$

should also be small. Overall, then, functions $u(x, y)$ and $v(x, y)$ are sought that minimize $e_s + \lambda e_c$, where λ is a parameter that weighs the error in the image motion equation relative to the departure from smoothness. This parameter should be made large if brightness measurements are accurate and small if they are noisy. Minimizing such expression is a problem of variational calculus.

10.2.2 The calculus of variations

Because of its wider relevance, a very concise account of the calculus of variations is given, and more specifically of the derivation of the so-called Euler-Lagrange equations, the solu-

tion of which extremizes the type of integrals we are interested in. Calculus yields extrema of functions of parameters, i.e. finds the values of the parameters extremizing the function. In the calculus of variations, extrema of expressions that depend on functions rather than parameters are found. Such expressions are called *functionals.* As an example, consider the integral

$$I = \int_{x_1}^{x_2} F(x, f, f')dx.$$

Here F depends on the unknown function f and its derivative f'. Assume the curve to be found must pass through the points $f(x_1) = f_1$ and $f(x_2) = f_2$. Suppose that the function $f(x)$ is a solution of the extremum problem. Then we expect that small variations in $f(x)$ should not change the integral significantly. Let $\eta(x)$ be a test function with $\eta(x_1) = 0$ and $\eta(x_2) = 0$. If we add $\epsilon\ \eta(x)$ to $f(x)$, with ϵ some real number, then the resulting function will still obey the boundary conditions. In order for $f(x)$ to be the desired function, we should find

$$\frac{dI}{d\epsilon}|_{\epsilon=0} = 0,$$

and this regardless of the exact nature of $\eta(x)$. Now I becomes

$$I = \int_{x_1}^{x_2} F(x, f + \epsilon\eta, f' + \epsilon\eta')dx.$$

Differentiating with respect to ϵ and setting ϵ equal to zero yields

$$0 = \int_{x_1}^{x_2} \left(\eta(x)\ F_{f|\epsilon=0} + \eta'(x)\ F_{f'|\epsilon=0}\right) dx.$$

Using integration by parts, we see that

$$\int_{x_1}^{x_2} \eta'(x)F_{f'}dx = [\eta(x)F_{f'}]_{x_1}^{x_2} - \int_{x_1}^{x_2} \eta(x)\frac{d}{dx}F_{f'}dx,$$

where the first term on the right hand side is zero because of the boundary conditions. We must therefore have

$$0 = \int_{x_1}^{x_2} \eta(x)\left(F_f - \frac{d}{dx}F_{f'}\right) dx.$$

If this is to be true for all test functions $\eta(x)$, then

$$F_f - \frac{d}{dx}F_{f'} = 0.$$

This is called the *Euler-Lagrange equation* for this problem [1].

[1] Although variational problems had been considered much earlier, it was only until the middle of the 18th century that work by Bernouilli, Euler, and Lagrange put it on a firm mathematical footing. Euler discovered this equation in 1744.

The case in which the integrand depends on several functions $f_1(x), f_2(x), \cdots$ instead of just one has to be dealt with, since there are two functions $u(x,y)$ and $v(x,y)$. Consider

$$I = \int_{x_1}^{x_2} F(x, f_1, f_2, \cdots, f_1', f_2', \cdots)dx.$$

The idea is the same. To each function a weighted test function with zero boundary conditions is added and the zero weight case should yield the optimum. In this case there are as many Euler-Lagrange equations as there are unknown functions:

$$F_{f_i} - \frac{d}{dx}F_{f_i'} = 0.$$

A second modification is the need to deal with several, independent variables – x and y – instead of just one:

$$I = \iint_D F(x, y, f, f_x, f_y)dxdy \ ,$$

where the integral is over some simply-connected closed region D. We introduce a test function $\eta(x,y)$ and add $\epsilon\ \eta(x,y)$ to $f(x,y)$. We are given the values of $f(x,y)$ on the boundary ∂D of the region, so the test function must be zero on the boundary. Minimizing

$$I = \iint_D F(x, y, f + \epsilon\eta, f_x + \epsilon\eta_x, f_y + \epsilon\eta_y)dxdy \ ,$$

differentiating with respect to ϵ and setting ϵ equal to zero yields

$$0 = \iint_D \left(\eta F_f + \eta_x F_{f_x|\epsilon=0} + \eta_y F_{f_y|\epsilon=0}\right)\ dxdy.$$

Now by Gauss's integral theorem,

$$\iint_D \left(\frac{\partial Q}{\partial x} + \frac{\partial P}{\partial y}\right) dxdy = \oint_{\partial D}(Qdy - Pdx),$$

so that

$$\iint_D \left(\frac{\partial}{\partial x}(\eta F_{f_x}) + \frac{\partial}{\partial y}(\eta F_{f_y})\right) dxdy = \oint_{\partial D}(\eta F_{f_x}dy - \eta F_{f_y}dx).$$

Given the boundary conditions, the term on the right must be zero, so that

$$\iint_D (\eta_x F_{f_x} + \eta_y F_{f_y})dxdy = -\iint_D \left(\eta\frac{\partial}{\partial x}F_{f_x} + \eta\frac{\partial}{\partial y}F_{f_y}\right) dxdy.$$

Consequently,

$$0 = \iint_D \eta\left(F_f - \frac{\partial}{\partial x}F_{f_x} - \frac{\partial}{\partial y}F_{f_y}\right) dxdy$$

for all test functions η. We must have, then, that

$$F_f - \frac{\partial}{\partial x}F_{f_x} - \frac{\partial}{\partial y}F_{f_y} = 0.$$

Here the Euler-Lagrange equation is a partial differential equation. The extension to more than two independent variables is also immediate. In fact, solutions to the Euler-Lagrange equation(s) render the "first variation" (terms in ϵ) stationary. There is no guarantee yet that these are extrema, let alone whether it is a maximum or a minimum that is found. In order to make sure about the character of the stationary value, one should consider the second variation (e.g. by performing the "Legendre test"). In general, this discussion is much more involved and it is omitted here.

10.2.3 Euler-Lagrange equations for optical flow

We are now in a position to search the solution that minimizes $e_s + \lambda e_c$. The Euler-Lagrange equations of our problem are

$$\begin{aligned} F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y} &= 0 \ , \\ F_v - \frac{\partial}{\partial x} F_{v_x} - \frac{\partial}{\partial y} F_{v_y} &= 0 \ . \end{aligned}$$

In our case,

$$F = (u_x^2 + u_y^2) + (v_x^2 + v_y^2) + \lambda(I_x u + I_y v + I_t)^2,$$

so the Euler-Lagrange equations are

$$\begin{aligned} \nabla^2 u &= \lambda(I_x u + I_y v + I_t) I_x \ , \\ \nabla^2 v &= \lambda(I_x u + I_y v + I_t) I_y \ , \end{aligned}$$

where as usual

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

is the Laplacian operator. This coupled pair of partial differential equations can be solved by iteratively solving the evolution equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u - \lambda(I_x u + I_y v + I_t) I_x \ , \\ \frac{\partial v}{\partial t} &= \nabla^2 v - \lambda(I_x u + I_y v + I_t) I_y \ . \end{aligned}$$

These equations are iteratively updated at each pixel. The derivatives in these equations can be approximated using finite differences. The diffusion term in these equations is positive if the local value falls below the neighourhood average and negative otherwise. The second term in the equations tries to increase or decrease the velocity so as to satisfy the optical flow constraint equation. The factors I_x and I_y in these terms make sure that the corrective action by the pair of these terms is applied to the velocity component along the intensity gradient. The solution, found at equilibrium $\frac{\partial u}{\partial t} = 0$ and $\frac{\partial v}{\partial t} = 0$, clearly depends on the chosen value of λ, which determines which values will be selected from the range in which the influence of the two terms is opposite.

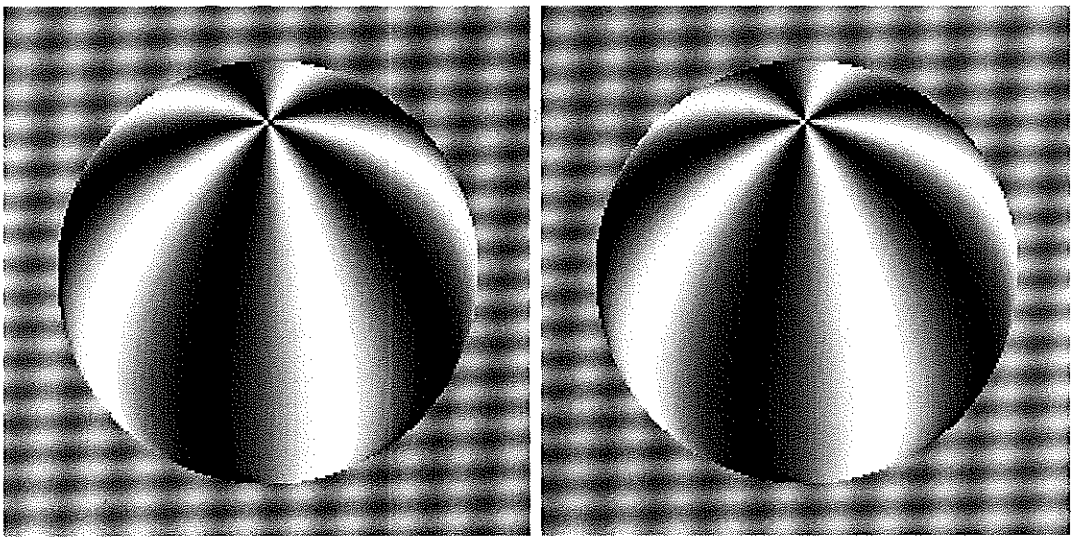


Figure 10.4: *Two subsequent frames from a synthetic image sequence showing a sphere rotating in front of a static background.*

It is interesting to note that having more than two image frames in a sequence allows a better estimation of I_t. Typically, at corners both components of the velocity are available. The above equations result in a spreading of this information to neighbouring regions where the second component cannot be obtained locally. As an example consider the two frames from a motion sequence shown in fig. 10.4. After initialization to zero velocity, the velocity estimations after the first iteration are shown in fig. 10.5. As can be seen, motion is only detected at places with a non-zero intensity gradient in the original image. Figures 10.6 and 10.7 show optical flow estimations after 5 and 50 iterations, respectively. Notice the clearly erroneous flow vectors at some points near the boundary of the sphere. At such positions information in finite difference neighbourhoods is derived from both the sphere and the background. The continuity constraint is inappropriate at these locations.

10.2.4 Regularization

The above process of "filling in" missing information on the basis of general assumptions about the world is not unique to this algorithm. Several problems in early vision have been treated this way. Early vision could be defined as inverse graphics. Whereas in computer graphics the problem is to generate realistically looking images of three-dimensional scenes with known object and illumination geometries and characteristics, computer vision is confronted with the inverse problem of recovering shapes and relative poses from images. Much information is lost during the projection of the three-dimensional world onto the two-dimensional images, and vision therefore must often rely on additional constraints to disambiguate the possible outputs. The identification of appropriate constraints is a recurring theme in computer vision and psychophysics.

The common characteristics of most early vision problems can be formalized: they

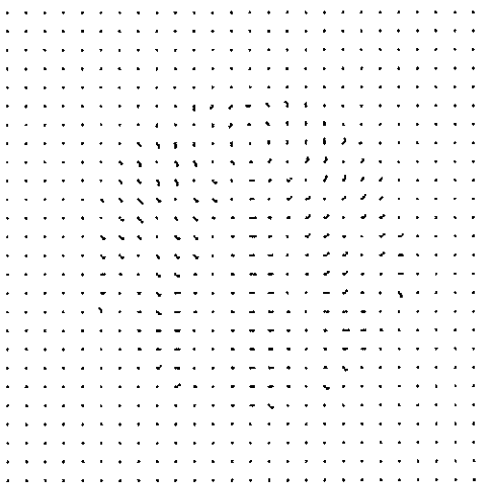


Figure 10.5: *Initial estimates for the velocity vectors (optical flow field). The vectors show both the direction of motion and the speed (longer vectors corresponding to higher speeds). As can be seen these initial results don't correspond with the projections of the true velocity vectors.*

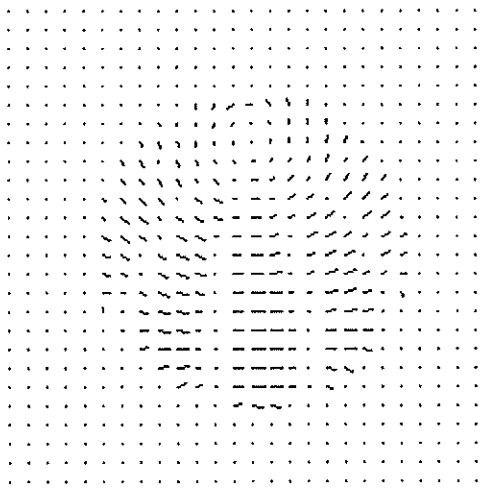


Figure 10.6: *The results after 5 iterations. The improvement is evident.*

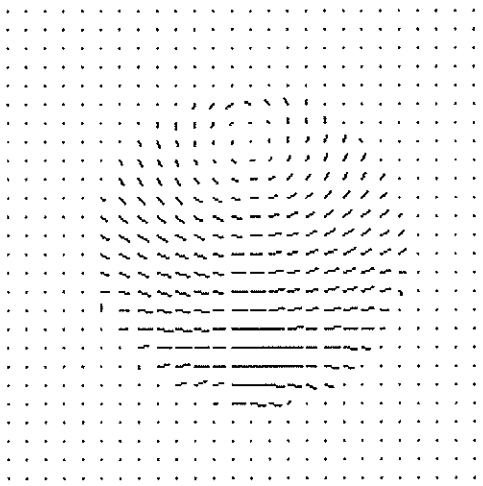


Figure 10.7: *Result after 50 iterations. The result closely resembles the actual optical flow field.*

are ill-posed in the sense defined by Hadamard. According to his definition, a problem is well-posed when its solution exists, is unique and depends continuously on the input data. Ill-posed problems fail to satisfy one or more of these criteria. The introduction of the constraints should turn an ill-posed problem into a well-posed one. In chapter 14 another example will be considered. This methodology is generally referred to as *regularization* and the constraints – such as the smoothness constraint in the optical flow problem – are also referred to as *regularizing constraints.*

10.3 II - Tracking

Not available yet

Chapter 10

Motion Extraction - addendum

1.1 Optical flow

see main text

1.2 Trackers

In contrast to optical flow, where an image motion vector is extracted for almost every pixel in the scene, 'tracking' methods are used to follow a specific object through an image sequence. Such methods will often not only produce the position as output, but also additional information, such as the direction or type of motion, the velocity, etc. With this information, a prediction can be made of the object position in the next frame and, hence, search can be restricted to a small window. For example if we want to follow a tennis ball, it is too inefficient to look for the ball in the whole image all the time. Tracking can predict its possible location, so that only the neighborhood around this position has to be searched. Temporal continuity of this kind also helps to segment out the object from cluttered backgrounds. Image segmentation may detect several possible locations of an object, and tracking can resolve this ambiguity. A tracking process can also increase the robustness, for example when occlusion occurs as information about an object is available even if it can not be detected.

Tracking is one of these applications that have benefited greatly from recent increases in the performance of standard PCs and the dropping prices of cameras. More and more real-time tracking approaches are becoming available. Tracking is relevant for many applications. In surveillance trackers should follow people and analyse their behaviour in order to generate an alarm if something abnormal is going on. Smart vehicles use tracking to support convoy driving and the detection of lane road and lane boundaries. Mobile robots find their locations by tracking landmarks, and so do augmented reality systems. In medicine heart surgery with robots has to compensate for cardiac motion through tracking. The entertainment industry makes increasing use of motion capture where the gestures of an actor are 'transplanted' onto a virtual character.

In the next section we will first explain a general tracking framework and will introduce the necessary notations. Subsequently, we will discuss in Section 1.2.2 a specific continuous tracking approach known as the *recursive Bayesian filter*. This method has only a few analytic solutions as for example the *Kalman filter* which we introduce in Section 1.2.2. As the Kalman filter can only deal with a restricted set of tracking problems we will explain in Section 1.2.2 the CONDENSATION *algorithm* which describes the sample-based version of the recursive Bayesian filter.

1.2.1 The Mean Shift tracker

not available yet

1.2.2 The CONDENSATION tracker

The general tracking framework

The quantities of an object that we want to track are described by an n-dimensional state vector $x \in \mathbb{X}^n$. Depending on the application, points, lines, contours, regions or whole 3D models could be used as state information. The set $\mathbb{X}$ would typically be the set of real numbers $\mathbb{R}$, but it could also be an enumerated set, for example {true, false}.

As we process image sequences, the observed scene changes over time. Accordingly, the state vector x is assumed to evolve according to a *system model*

$$x_t = f_{t-1}(x_{t-1}, w_{t-1}) \tag{1.1}$$

where $f_{t-1} : \mathbb{X}^n \times \mathbb{X}^w \rightarrow \mathbb{X}^n$ is the *system transition function*. As the system model may not be known exactly, $w_{t-1} \in \mathbb{X}^w$ is a noise vector with a known distribution that allows to model uncertainties. Consequently, with the aid of Eq. 1.1 a prediction of the object state for the next time step can be achieved.

At discrete times, measurements $z_t \in \mathbb{X}^m$ based on a segmentation (see chapter 7) become available. These observations are related to the state vector by the *measurement model*

$$z_t = h_t(x_t, v_t) \tag{1.2}$$

where $h_t : \mathbb{X}^n \times \mathbb{X}^v \rightarrow \mathbb{X}^m$ is the *measurement function*. As we assume that the measurements are also inaccurate, $v_t \in \mathbb{X}^v$ is another noise vector. The history of all observations is denoted as $\mathcal{Z}_t = \{z_1, \ldots, z_t\}$.

Generally, we are interested in deriving an estimate for the state vector x_t at time t from the previous state vector x_{t-1} and the measurement z_t. The entire tracking process is therefore described by first calculating a prediction of the state vector for the next time step and then using the measurement to update the state estimation. This results in the iterative scheme of Fig. 1.1.

As an example, let us track an object that moves in only one direction. We consider a 2D state vector

$$x_t = (p_t, \dot{p}_t) \tag{1.3}$$

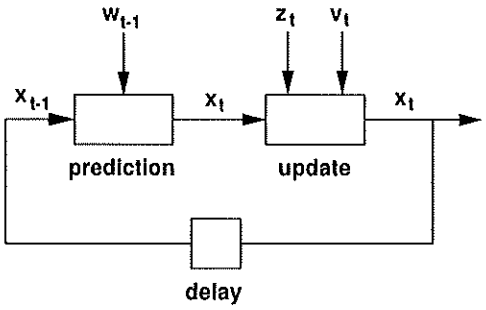


Figure 1.1: *Iterative tracking system.*

which includes the position p_t and the velocity $\dot{p}_t$ of an object. We further assume that the object moves with constant velocity, so the system model is given by

$$\begin{aligned} p_t &= p_{t-1} + \Delta t\, \dot{p}_{t-1} + w_{1,t-1} \\ \dot{p}_t &= \dot{p}_{t-1} + w_{2,t-1}. \end{aligned} \tag{1.4}$$

The vector of random variables $w_{t-1} = (w_{1,t-1}, w_{2,t-1})$ allows the model to adapt itself to small changes in the velocity. This is very important as it may not be possible to initialize the tracker with the correct velocity. For the measurement model, we expect that only the position of the object can be observed, so

$$z_t = p_t + v_t. \tag{1.5}$$

Within the measurement model, the random variable v_t illustrates the inaccuracy of the measurement, which can either be caused by the camera, for example by an insufficient resolution, or by the segmentation. As can be seen, in the state vector x_t we also manage the velocity which can not be measured directly.

Recursive Bayesian Filter

In the following, we consider a statistical tracking approach known as the recursive Bayesian Filter. An object is no longer represented by a single object state but by its probability distribution. As in the framework of Fig. 1.1, we want to calculate the prediction of the probability distribution for the next time step, which is also called the *a priori* distribution $p(x_t|\mathcal{Z}_{t-1})$. The update of this distribution with the given new measurement is then the *a posteriori* distribution $p(x_t|\mathcal{Z}_t)$. Thus, we recursively construct the *a posteriori* probability density of the state conditioned on all measurements.

Prediction The system model is now described by

$$p(x_t|\mathcal{Z}_{t-1}) = \int p(x_t|x_{t-1})\, p(x_{t-1}|\mathcal{Z}_{t-1})\, dx_{t-1} \tag{1.6}$$

where $p(x_t|x_{t-1})$ is the *process density*. This equation is used to estimate the probability distribution for the next time step. An integration must be solved as each x_{t-1} can make a contribution to $p(x_t|\mathcal{Z}_{t-1})$ via $p(x_t|x_{t-1})$. Under the same assumption as for Eq. 1.1 the density function $p(x_t|\mathcal{Z}_{t-1})$ depends only on the immediately preceding distribution $p(x_{t-1}|\mathcal{Z}_{t-1})$, but not on any function prior to $t-1$.

Update For the tracking, we want to update our prediction at each time step by incorporating the new measurement z_t. We can use Bayes' rule to determine the *a posteriori* density $p(x_t|\mathcal{Z}_t)$ from the *a priori* density $p(x_t|\mathcal{Z}_{t-1})$

$$p(x_t|\mathcal{Z}_t) = \frac{p(z_t|x_t)\, p(x_t|\mathcal{Z}_{t-1})}{p(z_t|\mathcal{Z}_{t-1})} \tag{1.7}$$

where $p(z_t|\mathcal{Z}_{t-1})$ is a normalization factor. The *observation density* $p(z_t|x_t)$ defines the likelihood that a state x_t causes the measurement z_t.

At time $t = 0$ the knowledge about the initial state x_0 is assumed to be available in the form of the probability density $p(x_0)$ which is normally Gaussian. The great strength of this tracking method is, that neither restrictions on the system and measurement models nor on the distributions of the error sources have to be considered.

We take another look at the example in Section 1.2.2. As we assume that all distributions are Gaussian, the noise vectors w_{t-1} and v_t of Eq. 1.4 respectively Eq. 1.5 are normally distributed. In the left column of Fig. 1.2 one iteration step of this example is presented. In the top row, we start with the *a posteriori* density $p(x_{t-1}|\mathcal{Z}_{t-1}) = N(0.0, 0.23) \otimes N(0.0, 0.3)$ of time $t-1$. The second row illustrates then the *a priori* density $p(x_t|\mathcal{Z}_{t-1})$ which characterizes the prediction of the distribution for the next time step. Note that the function is more spread due to the uncertainty in the process density $p(x_t|x_{t-1}) = N(p_{t-1} + \dot{p}_{t-1}\Delta t, 0.37) \otimes N(\dot{p}_{t-1}, 0.32)$ which is shown in Fig. 1.3. The segmentation provides measurements which are used to update the *a priori* knowledge. By means of the observation density $p(z_t|x_t) = N(0.2, 0.18)$, the normalized *a posteriori* density $p(x_t|\mathcal{Z}_t)$ is calculated which is finally shown in the last row of Fig. 1.2.

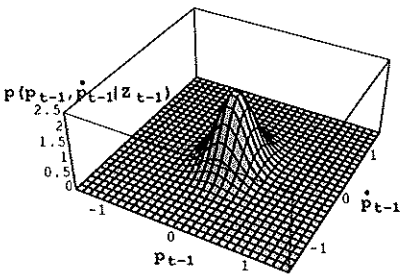
Kalman Filtering

In general, the integral leading to the *a priori* density (Eq. 1.6) is too complex and can not be evaluated in closed form. Analytic solutions to this problem are only available for a relatively small and restrictive choice of system and measurement models, as for example the Kalman filter. Rudolf Emil Kalman published his approach already in 1960 and it is nowadays still used as the standard tracking solution.

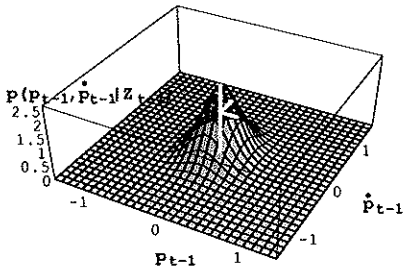
With a Kalman filter the functions f_{t-1} and h_t from Eq. 1.1 and 1.2 have to be linear and their error sources w_{t-1} and v_t are additive Gaussian noises of known variances. The probability functions remain therefore Gaussian at every iteration step, and the Kalman filter propagates and updates the mean and covariance of the probability distribution.

Thus, the model of a linear dynamic process for a Kalman filter is defined by

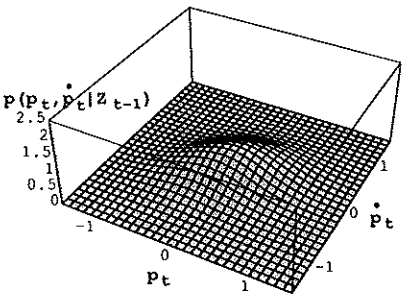
$$x_t = F_{t-1}\, x_{t-1} + w_{t-1}, \quad w_{t-1} \sim N(0, Q_{t-1}) \tag{1.8}$$



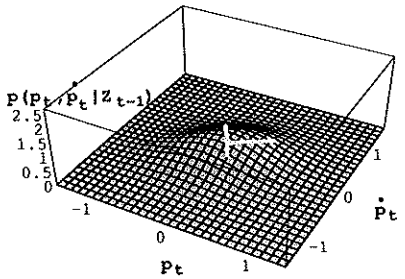
a posteriori density



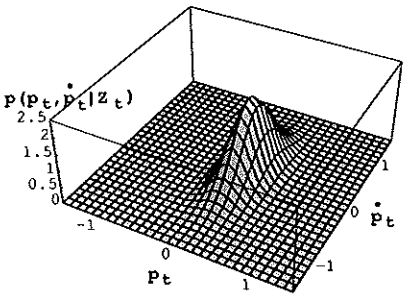
a posteriori knowledge $x_{t-1}(+)$ and $P_{t-1}(+)$



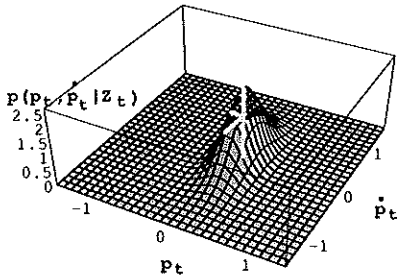
a priori density



a priori knowledge $x_t(-)$ and $P_t(-)$



a posteriori density



a posteriori knowledge $x_t(+)$ and $P_t(+)$

Figure 1.2: *One iteration step of the recursive Bayesian filter and the Kalman filter based on a 2D state vector. In the left column, the continuous solution is illustrated while in the right column, the analytic solution is shown. The results of the Kalman filter are given by the localizations as well as the widths and orientations of the Gaussian and are indicated in light grey.*

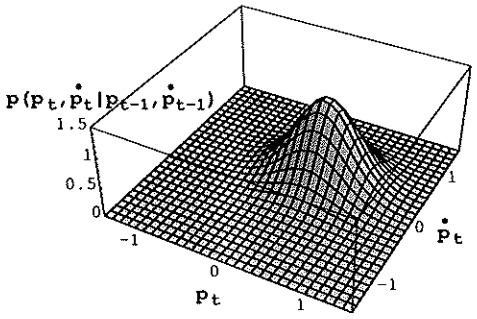


Figure 1.3: *The process density with fixed* $p_{t-1} = 0.0$ *and* $\dot{p}_{t-1} = 0.3$.

where the *system transition matrix* F_{t-1} models the evolution of the state vector x_t. The measurement model

$$z_t = H_t\, x_t + v_t, \quad v_t \sim N(0, R_t) \tag{1.9}$$

determines the measurement z_t as a function of the state x_t where H_t is called the *measurement sensitivity matrix*. The system noise w_{t-1} and the measurement noise v_t are zero-mean Gaussian sequences with given covariance-matrices Q_{t-1} and R_t, respectively.

Prediction The probability distributions are represented by a state vector x and a covariance matrix P within the Kalman filter. The state vector x specifies the localization of the Gaussian. The covariance matrix P describes the shape where the directions of the main axes of the Gaussian are given by the eigenvectors of the matrix and the widths of the Gaussian along the axes by the eigenvalues. The *a posteriori* knowledge $x_{t-1}(+)$ and $P_{t-1}(+)$ from the previous time step is propagated to determine the *a priori* knowledge given by $x_t(-)$ and $P_t(-)$.

Update The *a posteriori* state $x_t(+)$ can be calculated as a linear combination of the *a priori* knowledge and the new measurement. The weighting matrix K_t is called the *Kalman gain matrix*.

The steps of one iteration of a Kalman filter are subsequently indicated.

Kalman Filter

1. *a priori* state estimate prediction:
   $x_t(-) = F_{t-1}\, x_{t-1}(+)$
2. error covariance prediction:
   $P_t(-) = F_{t-1}\, P_{t-1}(+)\, F_{t-1}^T + Q_{t-1}$

3. Kalman gain matrix :
$$K_t = P_t(-)\, H_t^T\, [H_t\, P_t(-)\, H_t^T + R_t]^{-1}$$

4. *a posteriori* state estimate observational update:
$$x_t(+) = [I - K_t\, H_t]\, x_t(-) + K_t\, z_t = x_t(-) + K_t\, [z_t - H_t\, x_t(-)]$$

5. error covariance update:
$$P_t(+) = [I - K_t\, H_t]\, P_t(-).$$

To illustrate the Kalman filter we come back to the example of Section 1.2.2. The five steps of one iteration are subsequently specified in details.

$$\begin{aligned}
x_t(-) &= F_{t-1}\, x_{t-1}(+) \\
&= \begin{pmatrix} 1.0 & \Delta t = 1.0 \\ 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \\
&= \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
P_t(-) &= F_{t-1}\, P_{t-1}(+)\, F_{t-1}^T + Q_{t-1} \\
&= \begin{pmatrix} 1.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} 0.23^2 & 0.0 \\ 0.0 & 0.3^2 \end{pmatrix} \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 1.0 \end{pmatrix} + \begin{pmatrix} 0.37^2 & 0.0 \\ 0.0 & 0.32^2 \end{pmatrix} \\
&= \begin{pmatrix} 0.2798 & 0.09 \\ 0.09 & 0.1924 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
K_t &= P_t(-)\, H_t^T\, [H_t\, P_t(-)\, H_t^T + R_t]^{-1} \\
&= \begin{pmatrix} 0.2798 & 0.09 \\ 0.09 & 0.1924 \end{pmatrix} \begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix} \left[\begin{pmatrix} 1.0 & 0.0 \end{pmatrix} \begin{pmatrix} 0.2798 & 0.09 \\ 0.09 & 0.1924 \end{pmatrix} \begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix} + 0.18^2 \right]^{-1} \\
&= \begin{pmatrix} 0.89622 \\ 0.288277 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
x_t(+) &= x_t(-) + K_t\, [z_t - H_t\, x_t(-)] \\
&= \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} + \begin{pmatrix} 0.89622 \\ 0.288277 \end{pmatrix} \left[0.2 - \begin{pmatrix} 1.0 & 0.0 \end{pmatrix} \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \right] \\
&= \begin{pmatrix} 0.179244 \\ 0.0576553 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
P_t(+) &= [I - K_t\, H_t]\, P_t(-) \\
&= \left[\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} - \begin{pmatrix} 0.89622 \\ 0.288277 \end{pmatrix} \begin{pmatrix} 1.0 & 0.0 \end{pmatrix} \right] \begin{pmatrix} 0.2798 & 0.09 \\ 0.09 & 0.1924 \end{pmatrix} \\
&= \begin{pmatrix} 0.0290375 & 0.00934017 \\ 0.00934017 & 0.166455 \end{pmatrix}
\end{aligned}$$

Condensation Algorithm

The CONDENSATION algorithm — CONditional DENSity propagATION over time — is a sample-based solution of the recursive Bayesian filter. This tracking method was first introduced into the computer vision community by Michael Isard and Andrew Blake from the University of Oxford.

The underlying key idea is to represent the probability distribution by a weighted sample set $S = \{(s^{(n)}, \pi^{(n)}) | n = 1 \dots N\}$. Consequently, the distribution of the samples constitutes a discrete approximation of the probability distribution. Each sample consists of an element s which represents the state vector and a corresponding numerical weighting factor i.e. a discrete sampling probability which is given by π. The size of the sample set corresponds to N.

In analogy with the recursive Bayesian filter the CONDENSATION algorithm proceeds first with a prediction and afterwards with an update in each iteration step.

Prediction We start with a sample set S_{t-1} computed in the previous iteration step, which approximates the *a posteriori* density $p(x_{t-1}|\mathcal{Z}_{t-1})$. The evolution of each sample is described by the application of the system model

$$s_t'^{(n)} = F\, s_{t-1}^{(n)} + B\, w_{t-1}^{(n)} \tag{1.10}$$

where F defines the deterministic and $B\, w_{t-1}$ the stochastic component of this difference equation. The deterministic part of the equation models the system knowledge while the stochastic part allows us to model uncertainties. w_{t-1} is a vector of normal random variables scaled by B so that BB^T is the covariance of the process noise. We have now generated a new sample set S_t' which approximates the *a priori* density $p(x_t|\mathcal{Z}_{t-1})$.

Update As before, the new measurement z_t is used to update the prediction by weighting each element of the sample set S_t' on the basis of

$$\pi^{(n)} = p(z_t|s_t'^{(n)}). \tag{1.11}$$

By resampling the set S_t', we obtain a new sample set S_t which approximates the new *a posteriori* density $p(x_t|\mathcal{Z}_t)$. A particular sample $s_t'^{(j)}$ is namely drawn with replacement, by choosing it with probability $\pi^{(j)}$. Some elements, especially those with high weights, may be chosen several times, leading to identical elements in the new sample set S_t. During the prediction, these elements will be split due to the stochastic component of the system model. Others with relatively low weights may not be chosen at all.

To initialize the CONDENSATION algorithm, we start at $t = 0$ with a random sample set S_0 based on the probability density $p(x_0)$. The programming details for one iteration step are given subsequently.

The Condensation Algorithm

1. Propagate each sample from the set S_{t-1} according to the system model:
   $s_t^{\prime(n)} = f_{t-1}(s_{t-1}^{(n)}, w_{t-1}^{(n)})$ with $n = \{1, \ldots, N\}$
   where $w_{t-1}^{(n)}$ is a vector of random variables with a known distribution.

2. Observe the measurements:
   (a) weight each sample of the set S_t'
   $\pi_t^{(n)} = p(z_t | s_t^{\prime(n)})$
   (b) calculate the normalized cumulative probabilities
   $c_t^{(0)} = 0$
   $c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)} \qquad$ for $n = 1 \ldots N$
   $c_t^{\prime(n)} = \frac{c_t^{(n)}}{c_t^{(N)}} \qquad$ for $n = 1 \ldots N$

3. Select N samples from the set S_t' with probability $\pi_t^{(n)}$:
   (a) generate a uniformly distributed random number $r \in [0, 1]$
   (b) find, by binary search, the smallest j for which $c_t^{\prime(j)} \geq r$
   (c) set $s_t^{(n)} = s_t^{\prime(j)}$

To illustrate the behavior of the CONDENSATION algorithm we have another look at the example of Section 1.2.2. In the left column of Fig. 1.4 the results of the recursive Bayesian filter are shown while in the right column the sample distributions of the CONDENSATION algorithm with $N = 200$ are illustrated. The top row in the right column shows the sample set S_{t-1} which approximates the *a posteriori* density $p(x_{t-1}|\mathcal{Z}_{t-1})$ of time $t-1$. The evolution of each sample is described by

$$\begin{aligned} s_t^{\prime(n)} &= F\, s_{t-1}^{(n)} + B\, w_{t-1}^{(n)} \\ &= \begin{pmatrix} 1.0 & \Delta t = 1.0 \\ 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} p_{t-1}^{(n)} \\ \dot{p}_{t-1}^{(n)} \end{pmatrix} + \begin{pmatrix} 0.37 & 0.0 \\ 0.0 & 0.32 \end{pmatrix} \begin{pmatrix} w_{1,t-1}^{(n)} \\ w_{2,t-1}^{(n)} \end{pmatrix} \end{aligned}$$

where $w_{1,t-1}$ and $w_{2,t-1}$ are random values in the range $[0, 1]$. After the propagation we obtain the sample set S_t' which is shown in the second row. In the third row the weighted sample set S_t' is illustrated. As the observation density is Gaussian in this example the weights are calculated by

$$\begin{aligned} \pi^{(n)} &= p(z_t | s_t^{\prime(n)}) \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(z_t - H_t s_t^{\prime(n)}\right)^2}{2\sigma^2}} \\ &= \frac{1}{\sqrt{2\pi}\, 0.18} e^{-\frac{\left(0.2 - \begin{pmatrix} 1.0 & 0.0 \end{pmatrix} \begin{pmatrix} p_{t-1}^{(n)} \\ \dot{p}_{t-1}^{(n)} \end{pmatrix}\right)^2}{2 \cdot 0.18^2}}. \end{aligned}$$

In all other steps of the CONDENSATION algorithm, the weighting factor is $\pi^{(j)} = \frac{1}{N}$. In the last row the new resampled set S_t which approximates the new *a posteriori* density $p(x_t|\mathcal{Z}_t)$ is shown.

Until now we have only studied linear systems. As the CONDENSATION algorithm can also deal with unrestricted tracking models and error sources we now consider a non-linear problem which can not be solved by a Kalman filter. In this example we regard only a 1D state vector consisting of the object position. We focus on strong non-linear models given by

$$x_t = x_{t-1} + (-0.05\, x_{t-1} + 0.04\, x_{t-1}^2)\Delta t + w_{t-1} \tag{1.12}$$

$$z_t = x_t^2 + x_t^3 + v_t. \tag{1.13}$$

In Fig. 1.5 the exact solution shown in grey is compared to the result of the CONDENSATION tracker illustrated as black line as well as to the result of the Kalman filter plotted as dotted line. The exact solution is calculated by integrating the system model (Eq. 1.12) without noise from the initial value $x_0 = 2.0$. The initial density $p(x_0) = N(2.3, 0.1)$, however, is centered at $x = 2.3$ and Gaussian, so the tracker must be able to converge to the correct state. The noise terms of the system models are specified as $w_{t-1} = N(0.0, 0.4)$ and $v_t = N(0.0, 0.09)$, the step size is chosen as $\Delta t = 0.1$, the sample set has a size of $N = 200$ and the number of iterations is set to 190. As a quantitative interpretation of the CONDENSATION tracker, the mean value of the sample set is calculated

$$E[s] = \sum_{n=1}^{N} s^{(n)} \pi^{(n)}. \tag{1.14}$$

As we can see in this example, the mean state of the CONDENSATION tracker quickly converges close to the exact state and the non-linearity does not present any problem to the tracker. In contrast the Kalman filter lose the reference to the exact solution.

In Fig. 1.6 we consider examples for the interpretation of the sample distributions. Two different scenes of a public room recorded by a surveillance camera are shown. The white points represent the current sample distribution of the likely positions of the person. Thus, the state vector is given by the x- and y-coordinates of the image and the corresponding velocities. The measurement vector is specified by the center of the person which was detected using a region based segmentation. The size of the sample set was $N = 400$ in this example. In black we see the histories of the mean values which were calculated using Eq. 1.14. These trajectories represent the distances that have been covered by the two persons. In the right image we can notice especially that temporal occlusion — when the person passes behind the pillar — is handled correctly.

Comparison between Condensation and Kalman

In contrast to the CONDENSATION tracker the density function used by a Kalman filter is unimodal and evolves as a Gaussian. The great advantage of the CONDENSATION algorithm is therefore, that no restrictions on the system transition and measurement

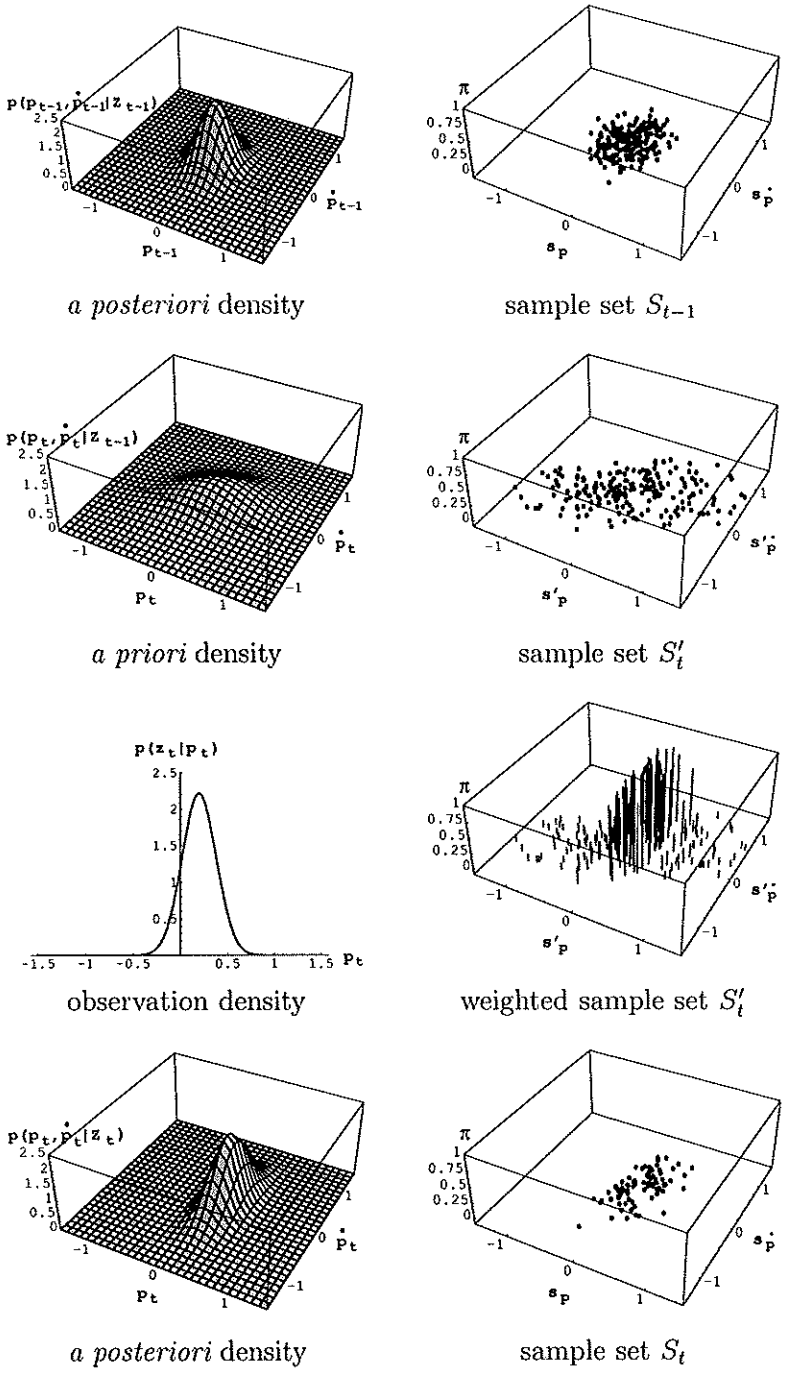


Figure 1.4: *One iteration step of the recursive Bayesian filter and the appropriate* CONDENSATION *tracker based on a 2D state vector. In the left column, the continuous solution is illustrated while in the right column, the sample-based representation is shown.*

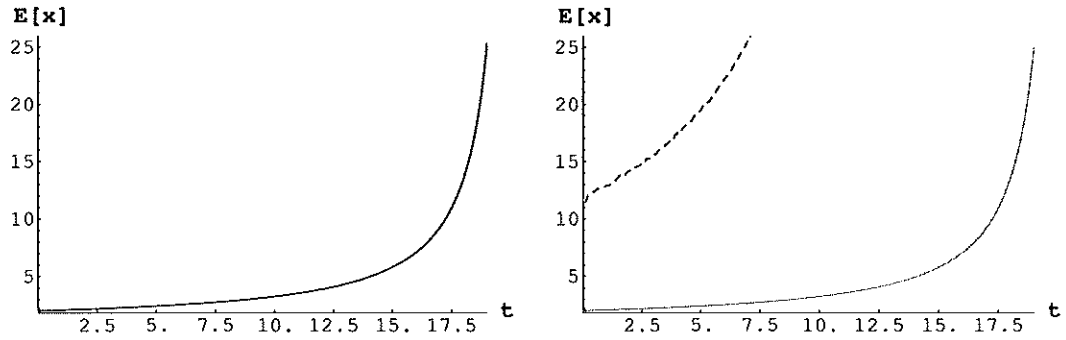


Figure 1.5: *Study of the convergence for non-linear models. The grey line represents the exact solution while the black line shows the result of the* CONDENSATION*tracker (left column) and the dotted line represents the result of the Kalman filter (right column).*



Figure 1.6: *Interpretation of a sample set. The white points represent the sample distributions whereas in black the histories of the mean values are shown.*

tracking methods	advantages	disadvantages
CONDENSATION algorithm	- unrestricted tracking models - unrestricted error sources - multiple hypotheses - simple - easy implementation	- discretization error - interpretation of the sample set
Kalman filter	- linear tracking models - Gaussian error sources - exact solution - easy interpretation	- management system for multiple hypotheses - running time depends on the number of hypotheses

Table 1.1: *Advantages and disadvantages of the different tracking approaches.*

functions, nor on the noise distributions are given. On the other hand if the models *are* linear and the error sources *are* Gaussian, the Kalman filter calculates the exact solution of the recursive Bayesian filter while the solution of the CONDENSATION algorithm has a discretization error as the probability distributions are only approximated by sample sets.

Apart from this restriction the sample-based representation has also many advantages. The calculations for each sample are simple and easy to program. The fixed number of samples additionally leads to a constant running time per iteration step. However, the sample set is not in a form that is always directly usable, but we can calculate for example the mean value of the object state or its moments.

A single Kalman filter is able to track only one object hypothesis. In the case of multiple hypotheses tracking to handle occlusions or unseen objects it is necessary to keep several trackers for each hypothesis. Therefore, Kalman filters can rapidly lead to unwieldy levels of complexity. As the CONDENSATION algorithm has the ability to deal with multi-modal distributions, multiple hypotheses can easily be tracked simultaneously.

An overview of the advantages and disadvantages of these different tracking methods is shown in Table 1.1.

An example of the Condensation approach

The CONDENSATION algorithm has mostly been used for the tracking of contours. These have a number of disadvantages. First, the extraction of the contours tends to take quite a bit of time and it is not straightforward to get such trackers real-time (but some are!). Secondly, contours are rather fragile features and change quite dramatically with viewpoint. Hence, here we describe a simple tracker that tracks elliptical regions, based on their color histograms.

Tracking based on color histograms As explained, in order to achieve robustness against non-rigidity, rotation and partial occlusion we focus on elliptical regions and their

color distributions as target models. The latter are represented by histograms which are produced with the function $h(x_i)$, that assigns the color at location x_i to one of the m bins of the color histogram. The histograms are directly calculated in the RGB space using 8x8x8 bins.

Not all pixels in the region are equally important to describe the objects. For example, pixels that are further away from the region center can be assigned smaller weights by employing a weighting function

$$k(r) = \begin{cases} 1 - r^2 & : \quad r < 1 \\ 0 & : \quad \text{otherwise} \end{cases} \tag{1.15}$$

where r is the distance from the region center. Thus, we increase the reliability of the color distribution when these boundary pixels belong to the background or get occluded. The color distribution $p_y = \{p_y^{(u)}\}_{u=1...m}$ at location y is calculated as

$$p_y^{(u)} = f \sum_{i=1}^{I} k\left(\left\|\frac{y - x_i}{a}\right\|\right) \delta[h(x_i) - u] \tag{1.16}$$

where δ is the Kronecker delta function and the sum is taken over all the pixels in the elliptical region. The variable a is used to adapt the size of the region. This rescaling is explained later. The normalization factor

$$f = \frac{1}{\sum_{i=1}^{I} k\left(\left\|\frac{y - x_i}{a}\right\|\right)} \tag{1.17}$$

ensures that $\sum_{u=1}^{m} p_y^{(u)} = 1$.

In the CONDENSATION tracking approach the estimated state is updated at each time step by incorporating the new observations. Therefore, we need a similarity measure that lets us compare observed and target color distributions. A popular measure between two distributions $p(u)$ and $q(u)$ is the Bhattacharyya coefficient

$$\rho[p, q] = \int \sqrt{p(u)q(u)}\, du. \tag{1.18}$$

Considering discrete densities such as our color histograms $p = \{p^{(u)}\}_{u=1...m}$ and $q = \{q^{(u)}\}_{u=1...m}$ the coefficient is defined as

$$\rho[p, q] = \sum_{u=1}^{m} \sqrt{p^{(u)}q^{(u)}}. \tag{1.19}$$

The larger ρ is, the more similar the distributions are, for two identical histograms we obtain $\rho = 1$, indicating a perfect match. As distance between two distributions we define the measure

$$d = \sqrt{1 - \rho[p, q]}. \tag{1.20}$$

Color-based Condensation The target regions of the proposed CONDENSATION are represented by ellipses, so that a sample is given as

$$s = \{x, y, \dot{x}, \dot{y}, H_x, H_y, \dot{H}_x, \dot{H}_y\} \tag{1.21}$$

where x, y represent the location of (center of) the ellipse, $\dot{x}$, $\dot{y}$ the motion, H_x, H_y the length of the half axes and $\dot{H}_x$, $\dot{H}_y$ the corresponding changes. As we consider a whole sample set the tracker handles multiple hypotheses (ellipses) simultaneously.

The sample set is propagated through the application of a dynamic model

$$s_t \;=\; A\, s_{t-1} + w_{t-1} \tag{1.22}$$

where A defines the deterministic and w_{t-1} the stochastic component, as usual. In our application we currently use a first order model for A describing ellipses moving with constant velocity and with their axis undergoing a constant scale changes from frame to frame. Expanding this model to second order would be straightforward.

To weigh the sample set, the Bhattacharyya coefficient has to be computed between the target distribution and the distribution of the hypotheses. The target histogram is initialised as the histogram of the originally indicated, elliptical region that is to be tracked subsequently. Each hypothetical region is specified by its state vector $s^{(n)}$. Both the target q and the candidate histogram $p_{s^{(n)}}$ are calculated from Eq. 1.16 where the target is centered in the origin. Note that one could gradually update the target distribution, in order to better cope with variations due to e.g. changing illumination conditions or mutual reflections from surrounding objects, or out-of-plane rotation of the target. Here the scheme is kept simple and a fixed target histogram is used.

The tracker should prefer samples (ellipses) with color distributions similar to the target model. The similarity is assessed through the Bhattacharyya distance, which is used to assign the weights $\pi^{(n)}$. The probability of each sample

$$\pi^{(n)} = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{d^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{(1-\rho[p_{s^{(n)}}, q])}{2\sigma^2}} \tag{1.23}$$

is specified by a Gaussian with variance σ. As already pointed out during filtering samples with a high weight may be chosen several times, leading to identical copies, while others with relatively low weights may not be chosen at all. The programming details for one iteration step of this particular CONDENSATION tracker are given in Figure 1.7.

To illustrate the distribution of the sample set generated by the tracker, consider fig. 1.8. The samples are shown in black. It is difficult to discern the individual ellipses, but one gets a reasonable feel of their spatial distribution. The inlet in the figure shows the average ellipse, which represents the most probably hypothesis. Figure 1.9 shows the Bhattacharyya coefficient of a small area around the average ellipse. The samples are located around the maximum of the Bhattacharyya coefficient, which represents the best match to the target model. As can be seen from the white dot, the mean state (average ellipse) of the sample distribution corresponds well to the maximum and consequently the localization of the face is accurate. The different samples are represented by the black dots.

Given the sample set s_{t-1} and the target model
$q = f \sum_{i=1}^{I} k\left(\left\|\frac{x_i}{a}\right\|\right)\ \delta[h(x_i) - u]$,
perform the following steps:

1. **Select** N samples from the set s_{t-1} with probability $\pi_{t-1}^{(n)}$:
   (a) calculate the normalized cumulative probabilities c'_{t-1}
   $$c_{t-1}^{(0)} = 0$$
   $$c_{t-1}^{(n)} = c_{t-1}^{(n-1)} + \pi_{t-1}^{(n)}$$
   $$c_{t-1}^{\prime(n)} = \frac{c_{t-1}^{(n)}}{c_{t-1}^{(N)}}$$
   (b) generate a uniformly distributed random number $r \in [0, 1]$
   (c) find, by binary search, the smallest j for which $c_{t-1}^{\prime(j)} \geq r$
   (d) set $s_{t-1}^{\prime(n)} = s_{t-1}^{(j)}$
2. **Propagate** each sample from the set s'_{t-1} by a linear stochastic differential equation:
   $s_t^{(n)} = A\, s_{t-1}^{\prime(n)} + w_{t-1}^{(n)}$
   where $w_{t-1}^{(n)}$ is the stochastic component.
3. **Observe** the color distributions:
   (a) calculate the color distribution
   $$p_{s_t^{(n)}}^{(u)} = f \sum_{i=1}^{I} k\left(\left\|\frac{s_t^{(n)} - x_i}{a}\right\|\right)\ \delta[h(x_i) - u]$$
   for each sample of the set
   (b) calculate the Bhattacharyya coefficient for each sample of the set s_t
   $$\rho[p_{s_t^{(n)}}, q] = \sum_{u=1}^{m} \sqrt{p_{s_t^{(n)}}^{(u)} q^{(u)}}$$
   (c) weight each sample
   $$\pi_t^{(n)} = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{(1-\rho[p_{s_t^{(n)}}, q])}{2\,\sigma^2}}$$
4. **Estimate** the mean state of the set s_t
   $E[s_t] = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}$

Figure 1.7: *An iteration step of the color-based* CONDENSATION *tracker.*

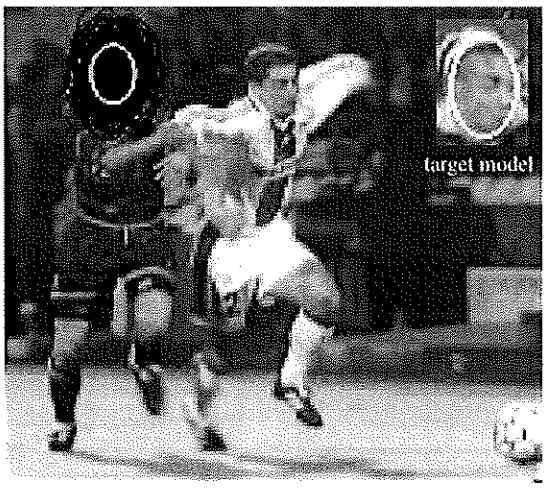


Figure 1.8: *A color-based target model and the different hypotheses (black ellipses) that are generated by the* CONDENSATION *tracker. The white ellipse represents the expected object location.*

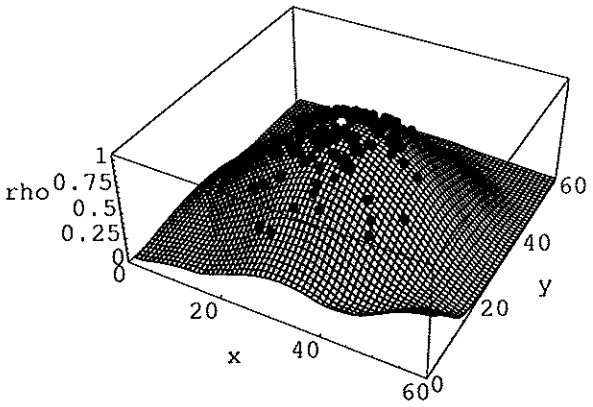


Figure 1.9: *Surface plot of the Bhattacharyya coefficient of a small area around the face of the soccer player shown in Figure 1.9. The black points illustrate the centers of the ellipses of the sample set while the white point represents the mean location.*

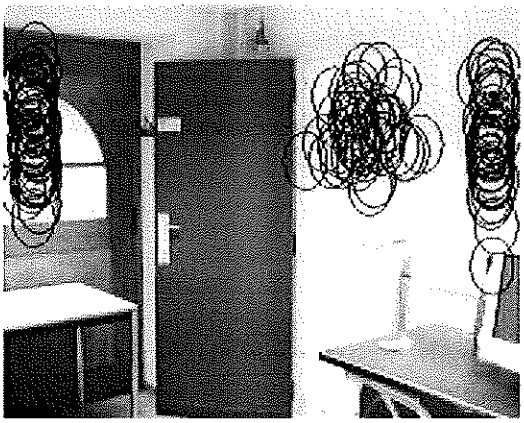


Figure 1.10: *The samples are spread over the image areas, where the object (in this case the person's head) is most likely to appear, like doors and image borders.*

For the initialization of the particle filter, we have to find the initial starting values x, y, H_x and H_y. There are three possibilities depending on the prior knowledge of the target object: manual initialization, automatic initialization using a known histogram as target model or an object recognition algorithm that finds interesting targets, e.g. by the detection of a region that moves over a static background. If the object histogram $q = \{q^{(u)}\}_{u=1...m}$ is known, we can place samples strategically at positions where the target is expected to appear. Fig. 1.10 shows such initialisation for a tracker that is expecting a person to re-enter a room, in order to resume the tracking of his head. The tracker should detect the object when it enters the field of view of the camera. In that case, the Bhattacharyya coefficient in the vicinity of the object position is significantly higher than the average coefficient of the background. This is the sign the tracker is waiting for. To be precise about what 'substantially higher' means, we first calculate the mean value μ and the standard deviation σ of the Bhattacharyya coefficient for every pixel of a background image:

$$\mu = \frac{1}{M}\sum_{i=1}^{M}\rho[p_{x_i}, q] \tag{1.24}$$

$$\sigma^2 = \frac{1}{M}\sum_{i=1}^{M}(\rho[p_{x_i}, q] - \mu)^2 \tag{1.25}$$

and then define an appearance rule as

$$\rho[p_{s_t^{(n)}}, q] > \mu + 2\sigma. \tag{1.26}$$

This indicates a 95% confidence that a sample does not belong to the background. If more than a fraction f of the samples fulfill the appearance rule during initialization, one can consider the object to be found and one can start tracking.

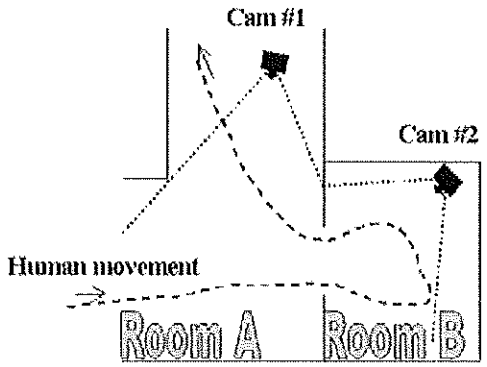


Figure 1.11: *Camera setup and the human movement of the* surveillance *example.*

Likewise, the same rule is used to determine if an object is lost during the tracking. If the number of positive appearances is smaller than f for a couple of frames, the tracker enters the 'initialization' mode and distributes the samples stochastically around the last mean object state. A value of $f = 0.1$ seems to be a reasonable value (as seen from experiments).

Results We first consider a *surveillance* sequence of 450 frames to demonstrate the efficiency of the approach. The system uses two fixed cameras to track a person who is moving inside two connected rooms. The configuration is shown in Figure 1.11. The cameras are static (no pan-tilt or zooming capacity). Their relative exterior orientation is known (externally calibrated). Camera 1 in room A is pointed to a door which leads to a closed room B that is observed by camera 2. Each camera has its own tracker.

Both trackers use the first order kinematic model of eq. 1.22. The new locations and sizes of the object are predicted based on the old values and the corresponding velocities, respectively scale changes. The noise terms are chosen proportional to the size of the initial region. In this experiment we used the initialization method based on a known histogram. Figure 1.12 shows the target histogram of the face/head tracker. Both trackers are put into the 'initialization' mode and will start tracking as soon as a person enters their field of view. When the person later leaves the room, the active tracker will return to the 'initialization' mode. In Figure 1.13 and Figure 1.14 the results of camera 1 respectively camera 2 are shown. The trackers handle the initialization successfully, even when the person is appearing from different sides. Also scale changes and out-of-plane rotations are dealt with. In particular, the rotations are very large as the face is seen from the front as well as from the side.

The method could be further improved by letting the trackers exchange information. For example when a person leaves room B, the exact position, velocity and region size could be handed over to the tracker in room A which can then initialize a sample distribution using this knowledge. Such provisions have not been made in this simple example.

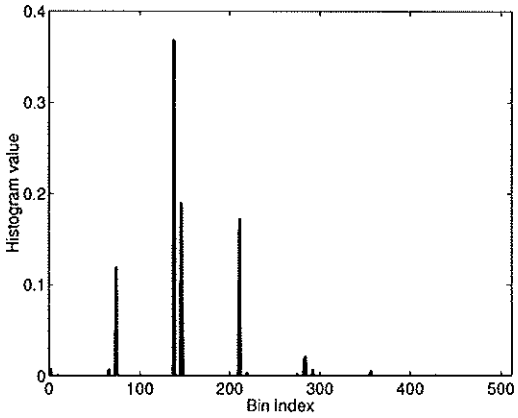


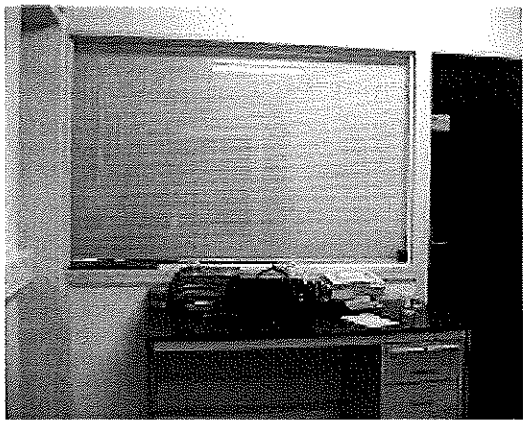
Figure 1.12: *Known face histogram for the surveillance example.*

Switching between the 'initialization' and the 'tracking' mode is done by applying the appearance rule of eq. 1.26. Figure 1.15 shows the number of positive appearances for both cameras. In this experiment we used $N = 100$ samples and the fraction was set to $f = 0.1$.

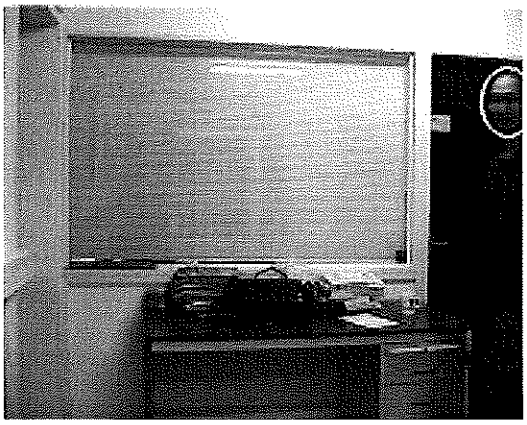
To demonstrate the robustness of the tracker against occlusion and rapid movements, it was applied it to a *soccer* sequence. The tracker had to follow a single player over 438 frames. The results are displayed in Figure 1.16. In this sequence the camera is not fixed but moving. The player is completely occluded by the referee in frame 156 and despite of other the other, surrounding players – which also make good target candidates – the tracker does not get distracted. Short tracking interruptions can occur when the occlusion persists for an extended period. In that case, the mean state is not located accurately for a short time, but due to the coexistence of multiple hypotheses the tracker can recover the player nonetheless. Figure 1.17 shows the sizes and displacements of the object region during this *soccer* sequence. The scaling changes quite smoothly.

1.2.3 Comparison of Means Shift and Condensation

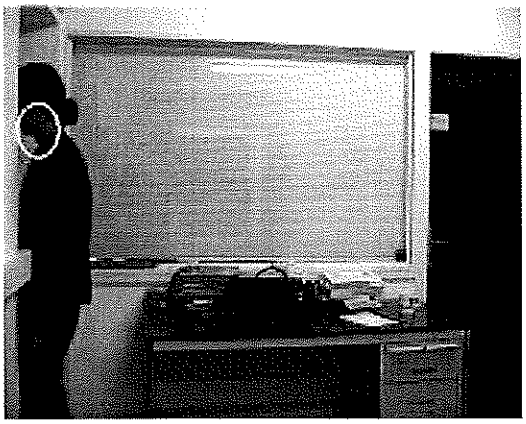
not available yet



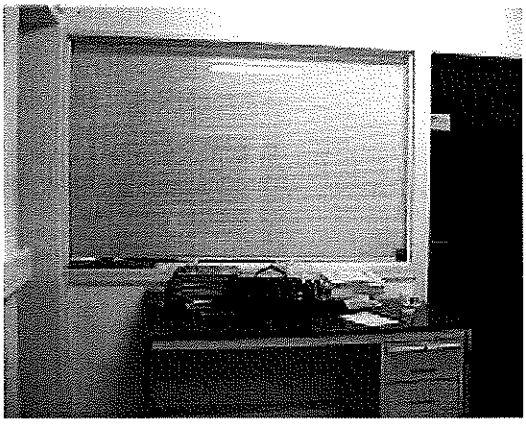
frame 1



frame 44



frame 111



frame 140



frame 387



frame 430

Figure 1.13: *Frame selection from the* surveillance *sequence of camera 1.*

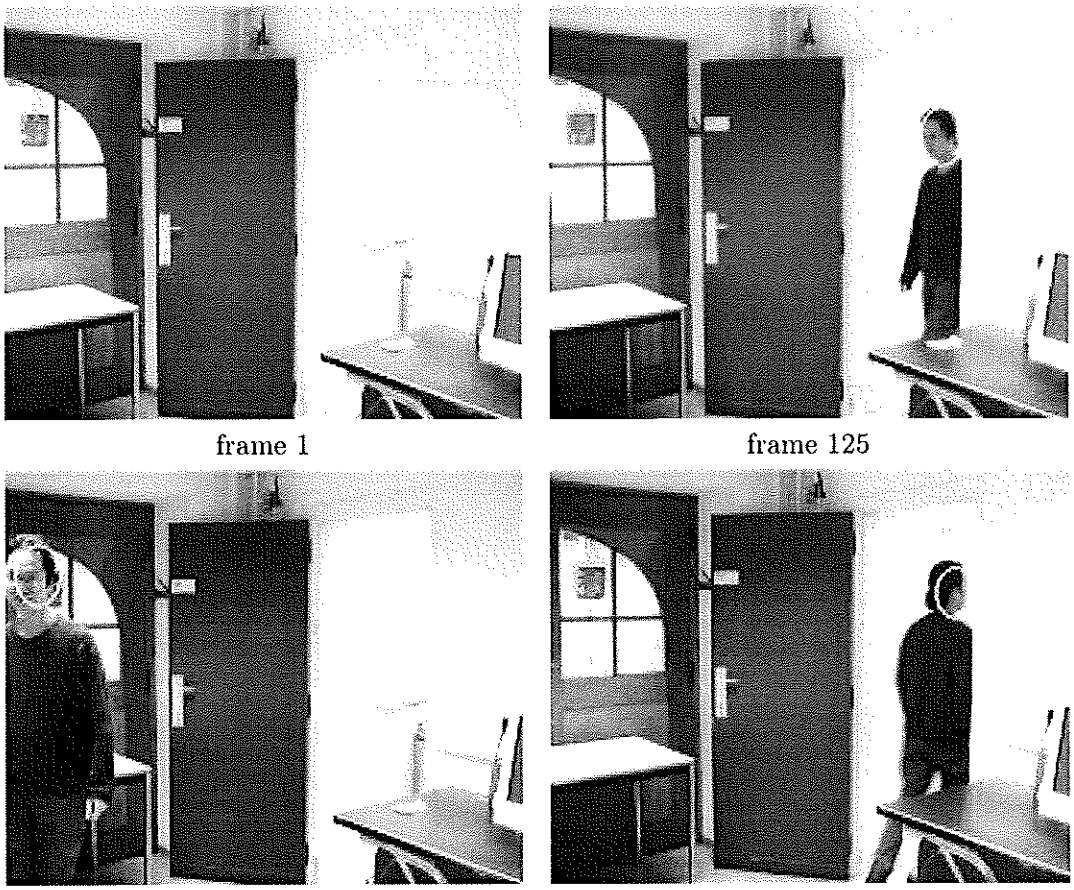


Figure 1.14: *Frame selection from the* surveillance *sequence of camera 2.*

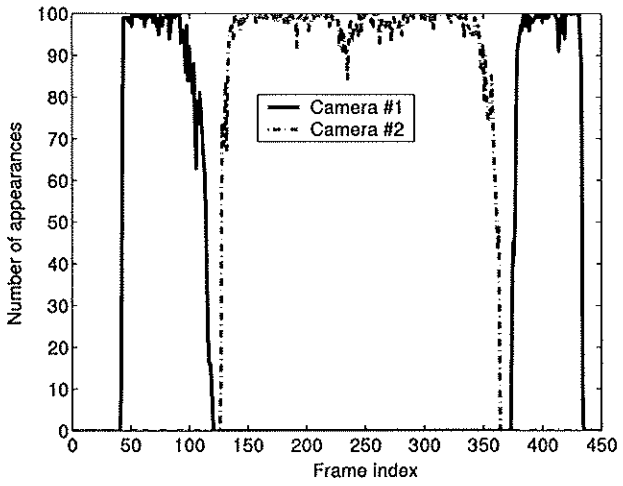
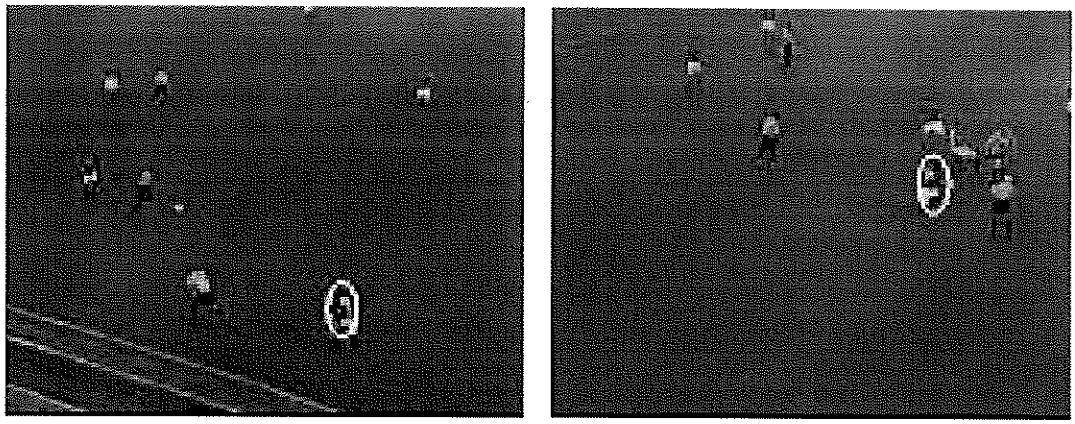
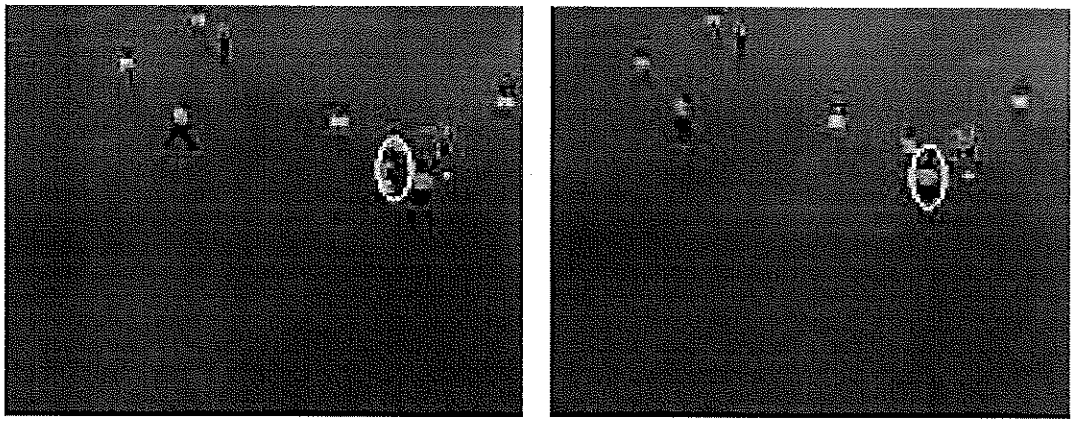


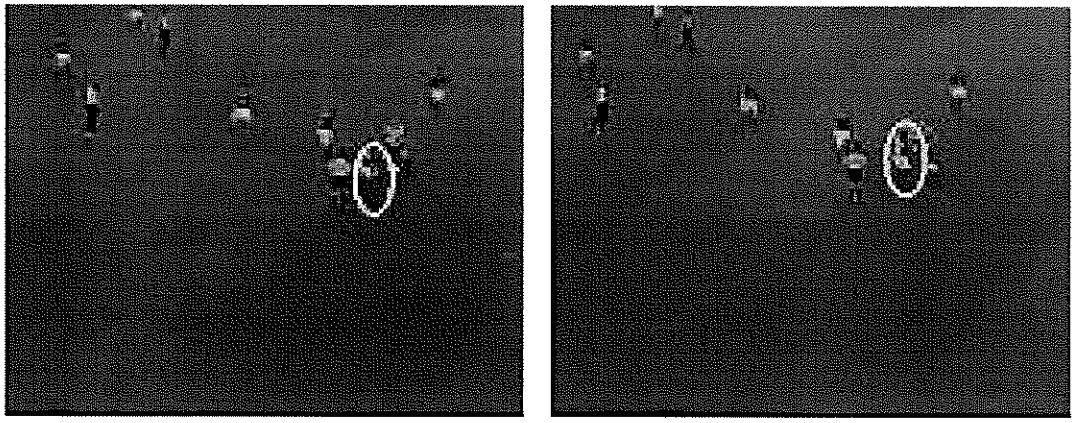
Figure 1.15: *The number of samples in each frame that fulfill the appearance rule for the* surveillance *sequence.*



frame 1 frame 101



frame 151 frame 156



frame 161 frame 166

Figure 1.16: *This* soccer *sequence shows the successful tracking of a player in cases of occlusion and rapid movement.*

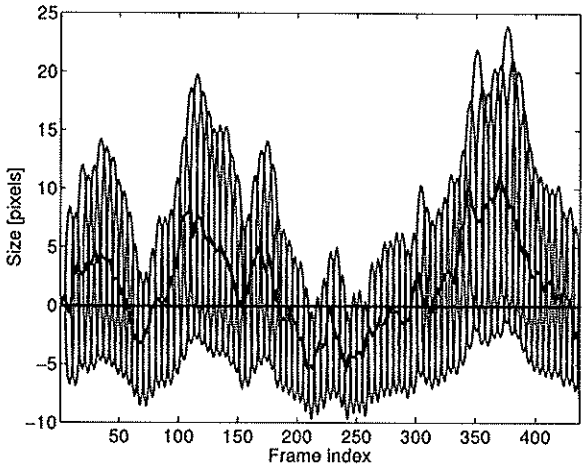


Figure 1.17: *The sizes and displacements along the x-coordinate of the object region during the* soccer *sequence is shown for every fifth frame. The scaling changes relatively smoothly.*