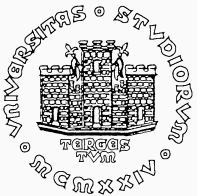
Computer Vision and Pattern Recognition

Course ID: 554SM – Fall 2018

Felice Andrea Pellegrino

University of Trieste
Department of Engineering and Architecture

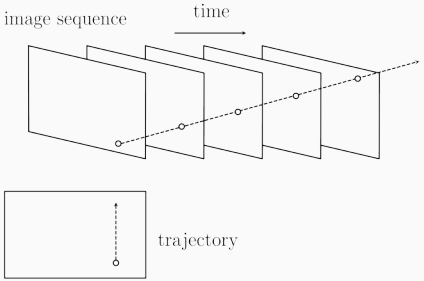


554SM –Fall 2018

Lecture 5: Tracking

Tracking

Tracking is the problem of *inferring the motion of an object given a sequence of images*.



It has various applications:

- motion capture;
- active camera control;
- surveillance;
- behavioral studies;
- ...

Tracking (cont.)

Two important classes of trackers are

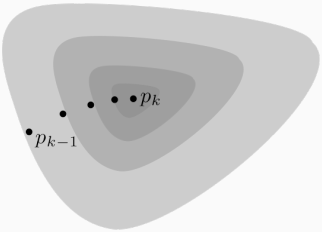
- trackers relying solely on the appearance;
- trackers using a *dynamic model* of the object motion.

In the first case, the detection is performed in the first frame and for each pair of subsequent frames, a displacement vector is computed, such that the local appearance is the same (basically, we search for the same window in the next frame). Since in most of the cases an effective solution is based on the gradient of the image, or of a suitable descriptor, we will refer to these methods as *Gradient-based trackers*.

In the second case, a dynamic model of the motion of the object is employed (for instance 'assuming that the object is moving at a constant velocity). The detection is performed frequently (for instance, at each frame) but the detected location is integrated with the prediction of the dynamic model, in order to improve the estimated location. The *Kalman filter* is a common and powerful approach based on dynamic models.

Gradient-based trackers

Gradient-based trackers



Gradient-based trackers (GBT) in brief:

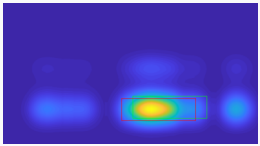
- GBT rely on the assumption that the object to be tracked undergoes a *small* displacement from the previous frame to the current frame; thus
  - the current location is close to the previous one;
  - the appearance is similar in two subsequent frames;
- they are are based on a *score function* that evaluates the quality of the candidate location in the current frame;
- the score function basically compares the previous appearance to the candidate and is assumed to reach a maximum in correspondence of the true new location;
- GBT attempt to *maximize the score function by iteratively refining the estimate of the location, starting from the location of the previous frame.*

Gradient-based trackers (cont.)

current frame and previous location

appearance model

likelihood of object location



mode seeking

current location

The appearance model may take various forms:

- image template;
- color histogram;
- intensity histogram;
- orientation histogram;
- a combination of the above;
- ...

Gradient-based trackers (cont.)

Problem: the tracked object may change its appearance in time. Possible solutions:

- update the appearance model periodically;
- take explicitly into account a parameterized deformation (for instance, affine deformation);
- use an appearance model robust to a class of deformations. Color histograms are often a good choice.

In the following we describe two popular mode seeking techniques:

- Kanade-Lucas-Tomasi (KLT) tracker;
- mean shift.

Kanade-Lucas-Tomasi tracker

The KLT tracker was proposed by (Tomasi and Kanade, 1991), based on the previous work (Lucas and Kanade, 1981).

Consider two subsequent frames $I(x, y, k)$ and $I(x, y, k+1)$. Suppose that in the frame k an interest point has been detected, located in $[x \;\; y]^\top$. Under a small displacement hypothesis, we wish to find the (small) $\Delta = [\Delta_x \;\; \Delta_y]^\top$ such that the appearance of the neighborhood of $[x + \Delta_x \;\; y + \Delta_y]^\top$ in frame $k+1$ is the closest to the appearance of the neighborhood of $[x \;\; y]^\top$ in frame k.

By defining the *residual*, expressing the dissimilarity in the appearance:

$$E(\Delta) = \sum_{d_x,\, d_y \in \mathcal{W}} \left[I(x + d_x + \Delta_x, y + d_y + \Delta_y, k+1) - I(x + d_x, y + d_y, k) \right]^2 ,$$

the displacement that minimizes the dissimilarity is:

$$\Delta^* = \underset{\Delta \in \mathcal{R}}{\operatorname{argmin}}\, E(\Delta),$$

where the set $\mathcal{R} \in \mathbb{R}^2$ is a rectangle centered in the origin, representing the *search range*.

Kanade-Lucas-Tomasi tracker (cont.)

In principle, on could use many different approaches to find the optimal displacement, including exhaustive search.

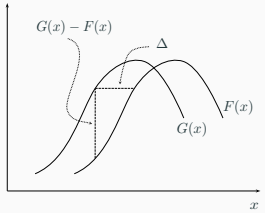
Exhaustive search method:

- avoids the problem of local minima (+);
- results possibly in slow running time (−);
- *does not provide sub-pixel precision* (−);

Kanade-Lucas-Tomasi tracker (cont.)

Alternative to the exhaustive search, or as a refinement of exhaustive search, *differential approaches* exist.

The basic idea of (Lucas and Kanade, 1981) is illustrated in the 1D example below.



Given two functions $F(x)$ and $G(x) = F(x + \Delta)$, we wish to find the displacement Δ. For small Δ:

$$F'(x) \approx \frac{F(x + \Delta) - F(x)}{\Delta} = \frac{G(x) - F(x)}{\Delta},$$

thus

$$\Delta \approx \frac{G(x) - F(x)}{F'(x)},$$

i.e. *the displacement can be estimated given the difference between the two functions and the derivative* $F'(x)$.

As said earlier, we wish to minimize:

$$E(\Delta_x, \Delta_y) = \sum_{d_x, d_y \in \mathcal{W}} \left[I(x + d_x + \Delta_x, y + d_y + \Delta_y, k+1) - I(x + d_x, y + d_y, k) \right]^2 .$$

Redefining $J(x, y) = I(x, y, k+1)$ and $I(x, y) = I(x, y, k)$, by Taylor series expansion we get

$$J(x + d_x + \Delta_x, y + d_y + \Delta_y) \approx J(x + d_x, y + d_y) + \nabla J^\top (x + d_x, y + d_y) \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix},$$

thus

$$E(\Delta_x, \Delta_y) = \sum_{d_x, d_y \in \mathcal{W}} \left[(J - I)(x + d_x, y + d_y) + \nabla J^\top (x + d_x, y + d_y) \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} \right]^2 .$$

Kanade-Lucas-Tomasi tracker (cont.)

Then, rewriting E (and using a light notation)

$$E(\Delta_x, \Delta_y) = \sum (J - I)^2 + 2\left(\sum (J - I)\begin{bmatrix} J_x & J_y \end{bmatrix}\begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}\right) + \begin{bmatrix} \Delta_x & \Delta_y \end{bmatrix} \underbrace{\begin{bmatrix} \sum J_x^2 & \sum J_x J_y \\ \sum J_x J_y & \sum J_y^2 \end{bmatrix}}_{\text{positive semidefinite by construction}} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}.$$

Since E is a convex quadratic function of Δ, the minimum can be found by equating the gradient to zero:

$$\nabla E(\Delta x, \Delta y) = 2\left[\sum (J - I)\begin{bmatrix} J_x \\ J_y \end{bmatrix} + \begin{bmatrix} \sum J_x^2 & \sum J_x J_y \\ \sum J_x J_y & \sum J_y^2 \end{bmatrix}\begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}\right] = 0. \tag{1}$$

Finally, if the second moment matrix is invertible (both eigenvalues are greater than zero):

$$\Delta = -\begin{bmatrix} \sum J_x^2 & \sum J_x J_y \\ \sum J_x J_y & \sum J_y^2 \end{bmatrix}^{-1} \sum (J - I)\begin{bmatrix} J_x \\ J_y \end{bmatrix}. \tag{2}$$

Kanade-Lucas-Tomasi tracker (cont.)

The procedure is applied in an iterative fashion:

1. start with some initial displacement $\boldsymbol{\Delta}_0$ at iteration $i = 0$;
2. obtain J_0 by shifting J by $\boldsymbol{\Delta}_0$;
3. repeat until convergence:
   - find the step s_i from J_{i-1} and I, using (2);
   - obtain J_i by shifting J_{i-1} by $\boldsymbol{\Delta}_i$;
4. the overall displacement is then:

$$\boldsymbol{\Delta}_i = \sum_{j=0}^{i} \Delta_j.$$

Kanade-Lucas-Tomasi tracker (cont.)

From a practical point of view, the following conditions (for suitable positive thresholds $\delta,\ \epsilon,\ \rho,\ i_{\max}$) must be satisfied for the iterations to continue:

1. $\|\boldsymbol{\Delta}_i\| > \delta$ (stepsize above a threshold);
2. $E(\boldsymbol{\Delta}_{i-1}) - E(\boldsymbol{\Delta}_i) > \epsilon$ (decrease of the residual above a threshold);
3. $\|\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_0\| \leq \rho$ (overall displacement not exceeding a threshold);
4. $i \leq i_{\max}$ (maximum number of iterations not exceeded).

Violating either of the first two conditions means *convergence*.

Violating either of the last two conditions means *failure*.

Observations

Observations:

- it is advisable to employ a Gaussian weighting window, to lessen the effects of multiple motions (different displacement) in the same window; thus the second moment matrix becomes

$$\begin{bmatrix} \sum J_x^2 w(d_x, d_y) & \sum J_x J_y w(d_x, d_y) \\ \sum J_x J_y w(d_x, d_y) & \sum J_y^2 w(d_x, d_y) \end{bmatrix}$$

  where

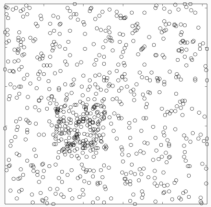
$$w(d_x, d_y) \propto \exp\left(-\frac{d_x^2 + d_y^2}{2\sigma^2}\right).$$

- The cornerness criterion introduced previously, applies to the second moment matrix and defines "good features to track" (Shi and Tomasi, 1994). Indeed, it implies that the linear system (1) is well-behaved.
- The approach can be extended to transformations other than translation: indeed (Shi and Tomasi, 1994) refine the method allowing for affine motions.

Mean shift tracking

The mean shift tracking (Comaniciu et al., 2003) is based on a technique for finding the local maxima of a density function, called *mean shift* (Fukunaga and Hostetler, 1975; Comaniciu and Meer, 2002).

- given a sparse set of samples from an unknown probability density function $x_1, x_2, \ldots, x_N \sim p(x)$;
- goal: find a *mode* (local maximum) of the distribution;



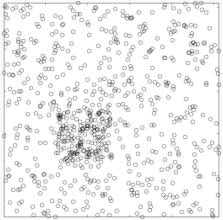
Possible approach:

1. smooth the data by convolving with a kernel $K(\cdot)$ of width h:

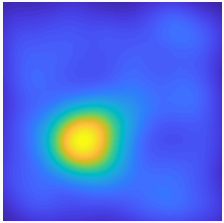
$$f(x) = \sum_{i=1}^{N} K(x - x_i) = \sum_{i=1}^{N} k\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

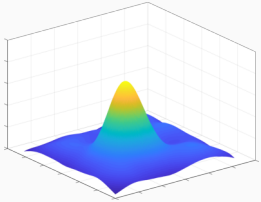
   obtaining an (unnormalized) estimate f of p;
2. find local maxima of f using gradient ascent or other optimization technique.

Example

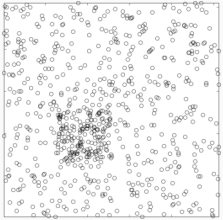


Samples from a distribution.

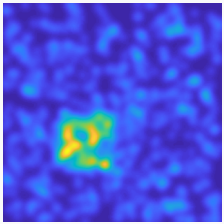


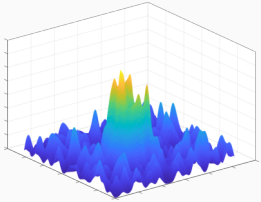


Samples smoothed with Gaussian kernel.



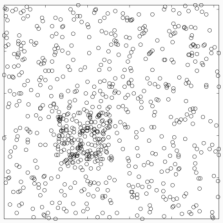
Samples from a distribution.



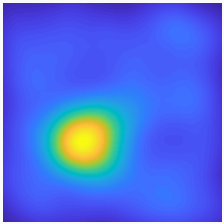


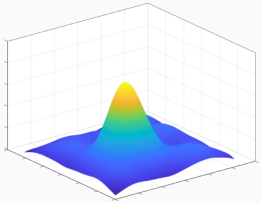
Samples smoothed with smaller Gaussian kernel.

Mean shift



Samples from a distribution.





Samples smoothed with Gaussian kernel.

- Problem: for high dimensions, it becomes computationally prohibitive to evaluate f over the whole search space.
- Observation: to apply gradient ascent we only need to know the *gradient* of f
- Alternative approach: **estimate the gradient in the first place**:
  - guess the local maximum, say $\hat{x}$;
  - estimate the gradient $\nabla f(\hat{x})$ from the samples in the neighborhood of $\hat{x}$;
  - update $\hat{x}$ along the gradient direction;
  - iterate until convergence.

Mean shift (cont.)

Consider again the following:

$$f(x) = \sum_{i=1}^{N} K(x - x_i) = \sum_{i=1}^{N} k\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

where $K(x)$ is a radially symmetric kernel. By defining $k'(r) = \frac{d}{dr}k(r)$ and $g(r) = -k'(r)$, the gradient may be written as

$$\begin{aligned}
\nabla f(x) &= \sum_i \frac{2(x - x_i)}{h^2} k'\left(\frac{\|x - x_i\|^2}{h^2}\right) \\
&= \frac{2}{h^2} \sum_i (x_i - x) g\left(\frac{\|x - x_i\|^2}{h^2}\right) \\
&= \frac{2}{h^2} \left(\sum_i x_i g\left(\frac{\|x - x_i\|^2}{h^2}\right) - x \sum_i g\left(\frac{\|x - x_i\|^2}{h^2}\right)\right) \\
&= \frac{2}{h^2} \left(\frac{\sum_i x_i g\left(\frac{\|x - x_i\|^2}{h^2}\right)}{\sum_i g\left(\frac{\|x - x_i\|^2}{h^2}\right)} - x\right) \sum_i g\left(\frac{\|x - x_i\|^2}{h^2}\right)
\end{aligned}$$

Mean shift (cont.)

Thus, by defining $G(x) = g\left(\frac{\|x\|^2}{h^2}\right)$

$$\nabla f(x) \propto \underbrace{\left(\frac{\sum_i x_i G(x - x_i)}{\sum_i G(x - x_i)} - x\right)}_{\doteq m(x)} \sum_i G(x - x_i) = m(x) \sum_i G(x - x_i)$$

where the vector $m(x)$ is called the *mean shift*, since it is the difference between the weighted mean of the neighbors x_i around x and the current value of x.

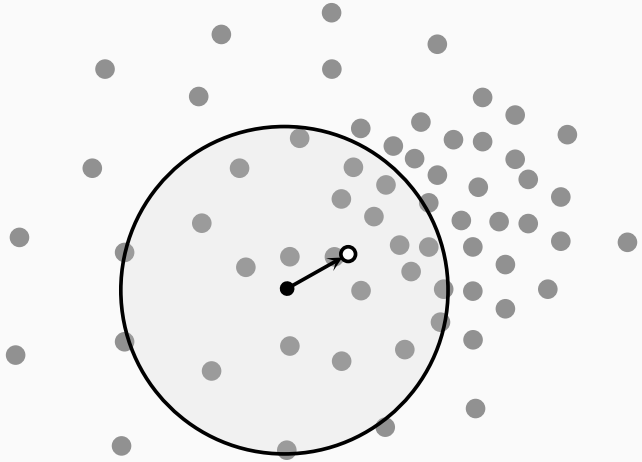
Provided that the term $\sum_i G(x - x_i)$ is positive[1], the mean shift has the same direction as the gradient.

The mean shift algorithm amounts to iteratively applying the following update rule:

$$\hat{x} \leftarrow \hat{x} + m(\hat{x}) = \frac{\sum_i x_i G(\hat{x} - x_i)}{\sum_i G(\hat{x} - x_i)}$$

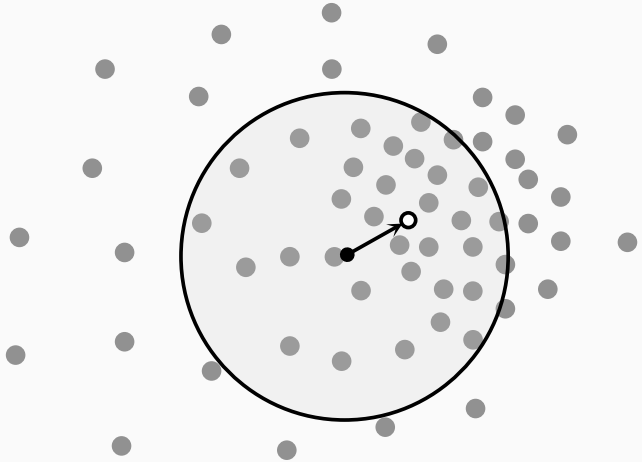
[1] a proper choice of the kernel may guarantee that condition (Comaniciu and Meer, 2002).

Intuition



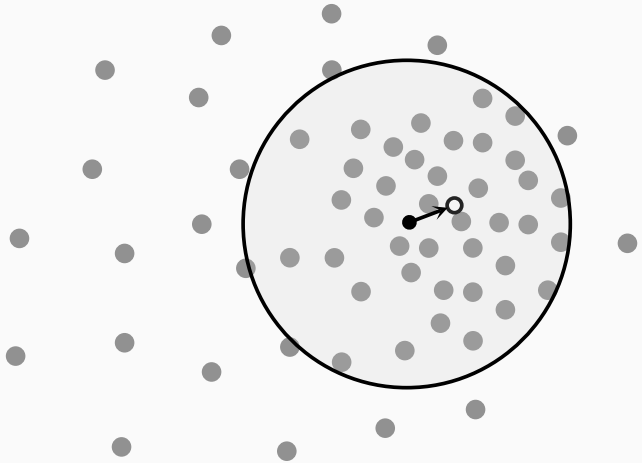
$$\hat{x} \leftarrow \hat{x} + m(\hat{x}) = \frac{\sum_i x_i G(\hat{x} - x_i)}{\sum_i G(\hat{x} - x_i)}$$

Intuition (cont.)



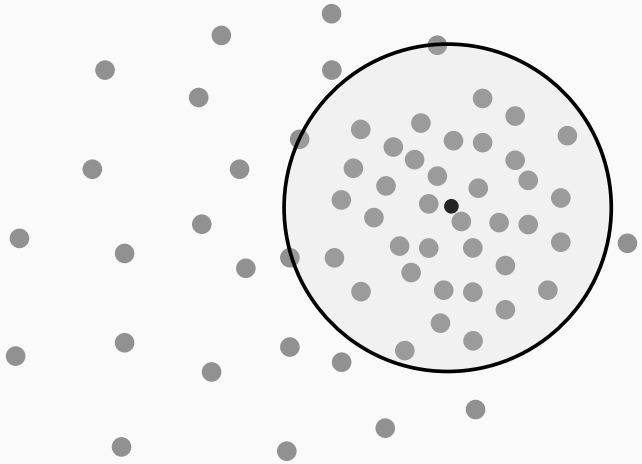
$$\hat{x} \leftarrow \hat{x} + m(\hat{x}) = \frac{\sum_i x_i G(\hat{x} - x_i)}{\sum_i G(\hat{x} - x_i)}$$

Intuition (cont.)



$$\hat{x} \leftarrow \hat{x} + m(\hat{x}) = \frac{\sum_i x_i G(\hat{x} - x_i)}{\sum_i G(\hat{x} - x_i)}$$

Intuition (cont.)



$$\hat{x} \leftarrow \hat{x} + m(\hat{x}) = \frac{\sum_i x_i G(\hat{x} - x_i)}{\sum_i G(\hat{x} - x_i)}$$

Kernels

Several kernels $K(x)$ may be employed, implicitly defined by the function

$$k(r) : \mathbb{R} \to \mathbb{R}, \quad r \geq 0.$$

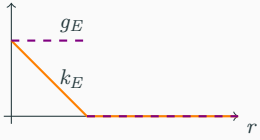
Two common kernels:

- Epanechnikov kernel:

$$k_E(r) = \max\{0, 1 - r\},$$

whose derivative kernel is the indicator function of the unit ball:

$$g_E(r) = \begin{cases} 1 & \text{if } r \leq 1 \\ 0 & \text{otherwise} \end{cases}.$$

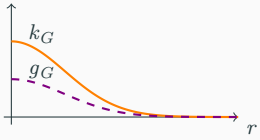


- Gaussian kernel:

$$k_G(r) = \exp\left(-\frac{r}{2}\right),$$

whose derivative kernel is another Gaussian:

$$g_G(r) = \frac{1}{2}\exp\left(-\frac{r}{2}\right).$$



Properties

- Stepsize depends on the gradient itself: *adaptive* gradient ascent.
- Convergence in general is NOT guaranteed.
- Sufficient conditions for convergence exist for special cases, see (Ghassabeh, 2013).
- In practice, it works quite well.
- Faster convergence using Epanechnikov kernel.
- Slower (but better results) with Gaussian.

Properties (cont.)

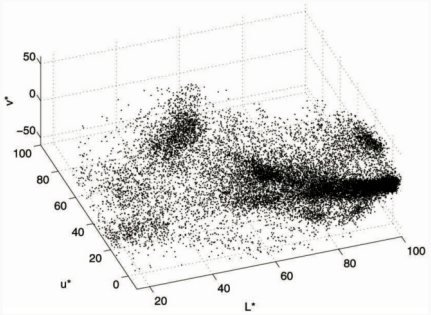
Strengths:

- Application independent tool.
- Does not assume any a priori knowledge on shape of the distribution.
- Only one parameter to choose (the bandwidth h).
- h has a meaning (represents a distance in the feature space).

Weaknesses:

- Results are rather sensitive to the bandwidth.
- Selecting the bandwidth is not trivial.
- Inappropriate bandwidth can cause modes to be merged, or generate additional modes.

Application to clustering



Figures from (Comaniciu and Meer, 2002).

Example: clustering based solely on the pixel color (using L*u*v* color space):

- apply mean shift repeatedly using different initialization;
- assign pixels to a class, based on the *basin of attraction of a mode*, i.e. the data points visited by all the mean shift procedures converging to that mode.

Application to clustering (cont.)

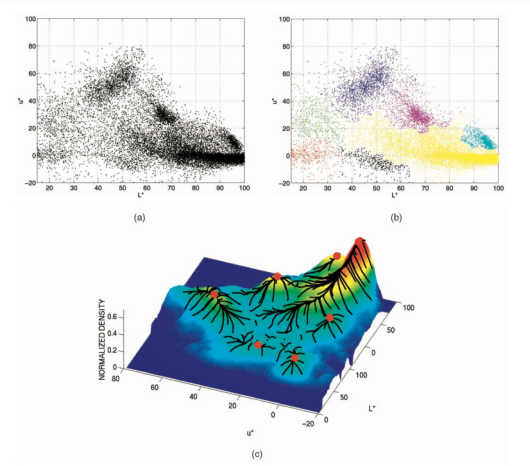


Figure from (Comaniciu and Meer, 2002).

Application to image segmentation

Clustering the pixels as in the previous slide, is itself a form of image segmentation. However, better results can usually be obtained by taking into account the location of the pixels, besides their color. The mean shift procedure can be applied in the *joint domain* of location x_s and color x_r (x_s and x_r are concatenated to form the feature vector x_j.) Location and color may have different scales, thus the kernel is adjusted accordingly:

$$K(x_j) = k\left(\frac{\|x_s\|^2}{h_s^2}\right) k\left(\frac{\|x_r\|^2}{h_r^2}\right).$$

Regions containing less than a given number of pixels must be eliminated.

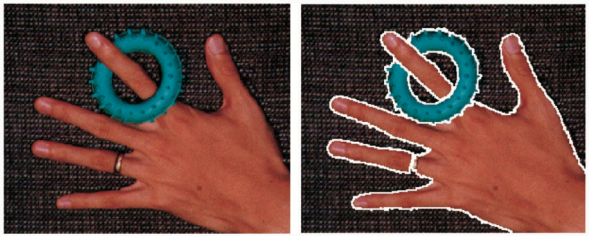


Figure from (Comaniciu and Meer, 2002).

Application to tracking

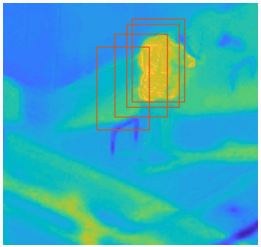
Two approaches:

1. Simplest approach: compute a *weight image* where each pixel has a weight proportional to its similarity to a desired color. The underlying assumption is that the more similar is a pixel's color to the desired color, the more likely that pixel will belong to the object. Apply mean shift to the weight image. The sought mode is the mode of the weighted location.
2. Not so simple, but effective, approach: describe the object as a *histogram of a color distribution within a region*. Use mean shift to find the region that has most similar color distribution.

Mean shift on weight images

previous frame

current frame



weight image

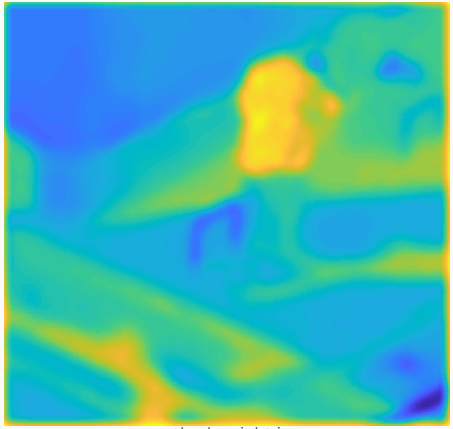
The mean shift is essentially the same procedure described before, except that each pixel location x_i is now weighted according to its weight, i.e. the similarity to the desired color:

$$\hat{x} \leftarrow \frac{\sum_i w(x_i) x_i G(\hat{x} - x_i)}{\sum_i w(x_i) G(\hat{x} - x_i)}.$$

In the example, the weight image is (the opposite of) the distance[2] in the L*a*b* space from each pixel's value to the triplet (89.4,-8.27,-6.12), representative of the shirt color.

[2] It is actually a weighted distance, to reduce the contribution of the luminance channel.

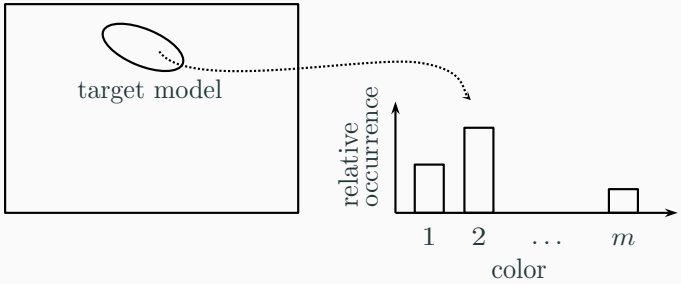
Gradient ascent



smoothed weight image

Performing the mean shift with kernel G on the weight image is equivalent to performing a gradient ascent on the smoothed weight image (precisely, the one obtained by smoothing with the "shadow" kernel K).

Mean shift on histograms



A more sophisticated approach is the one of (Comaniciu et al., 2000) and (Comaniciu et al., 2003):

- choose a reference target model (e.g. an elliptical region in the first frame, containing the object to be tracked);
- choose a feature space (e.g. a quantized color space);
- represent the *target model* as a distribution in the feature space (e.g. a histogram each bin of which contains the number of pixel in the region having a particular color)
- define a *similarity function* between histograms of candidate regions and target region;
- use mean shift to find the local maxima of the similarity function.

Target model

Actually, the histogram is computed by using a radially symmetric kernel of profile $k(r)$ to attribute less importance to peripheral pixels.

Let $\left\{x_i^*\right\}_{i=1\dots n}$ be a region of n pixels centered in zero. Define the *target model* as the histogram

$$q = \{q_1, q_2, \dots, q_m\},$$

where

$$q_u = C \sum_{i=1}^{n} k\left(\|x_i^*\|^2\right) \delta\left[b(x_i^*) - u\right], \quad u = 1 \dots m. \tag{3}$$

The function $b : \mathbb{R}^2 \to \{1 \dots m\}$ associates to the pixel at location x the index of its bin, while δ is the Kronecker delta function and C is a normalization constant such that $\sum_1^m q_u = 1$, i.e.

$$C = \frac{1}{\sum_{i=1}^{n} k\left(\left\|x_i^*\right\|^2\right)}.$$

Notice that each pixel contributes with one and only one term to the sum in (3). **Basically, q_u contains the relative occurrence of pixels of bin u, weighted by their distance from the origin, in a way specified by the function k.**

Target candidates

Let $\{x_i\}_{i=1\dots n_h}$ be a region of n_h pixels centered in $\hat{x}$. Define the *target candidate* as the histogram

$$p(\hat{x}) = \{p_1(\hat{x}), p_2(\hat{x}), \dots, p_m(\hat{x})\},$$

where

$$p_u(\hat{x}) = C_h \sum_{i=1}^{n_h} k\left(\frac{\|x_i\|^2}{h^2}\right) \delta\left[b(x_i) - u\right], \quad u = 1 \dots m, \tag{4}$$

and C_h is a normalization constant:

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\frac{\|x_i\|^2}{h^2}\right)}.$$

Notice that the kernel has now a bandwidth h.

Distance function

We define a distance between the two discrete distributions q and $p(\hat{x})$ as

$$d(\hat{x}) = \sqrt{1 - \rho(\hat{x})} = \sqrt{1 - \sum_{u=1}^{m} \sqrt{p_u(\hat{x}) q_u}}, \tag{5}$$

where $\rho(\hat{x}) = \sum_1^m \sqrt{p_u(\hat{x}) q_u}$ is the sample estimate of the *Bhattacharyya coefficient* between $p(\hat{x})$ and q.

Notice that

$$\rho(\hat{x}) = \underbrace{\left[\sqrt{p_1(\hat{x})} \;\; \sqrt{p_2(\hat{x})} \;\; \dots \sqrt{p_m(\hat{x})}\right]}_{\doteq a^\top} \underbrace{\begin{bmatrix} \sqrt{q_1} \\ \sqrt{q_2} \\ \vdots \\ \sqrt{q_m} \end{bmatrix}}_{\doteq b} \tag{6}$$

thus $\rho(\hat{x}) = a^\top b = \|a\| \, \|b\| \cos\left(\widehat{ab}\right) = \cos\left(\widehat{ab}\right)$, since both a and b are unit vectors due to the normalization. Furthermore, they lie in the first orthant thus their dot product is positive. Hence:

$$0 \leq \rho(\hat{x}) \leq 1.$$

Distance function (cont.)

Clearly, minimizing the distance $d(\hat{x})$ is equivalent to maximizing $\rho(\hat{x})$.

Reasons for choosing (5):

- it imposes a metric structure;
- is relatively invariant to object size (number of pixels);
- it is valid for arbitrary distributions;
- it has a clear geometric meaning.

The aim is to find (in the current frame):

$$\operatorname{argmax}_{\hat{x}} \rho(\hat{x}).$$

Instead of performing an exhaustive search, we can apply the mean shift procedure.

Let $\hat{x}_0$ be the location of the target in the previous frame. The search for the location in the current frame is performed via the mean shift procedure, starting from $\hat{x}_0$. First, the histogram of the target candidate at location $\hat{x}_0$ has to be computed:

$$\{p_u(\hat{x}_0)\}_{u=1\dots m}\,.$$

The Bhattacharyya coefficient $\sum_1^m \sqrt{p_u(\hat{x})q_u}$ may be approximated by its Taylor expansion around the values $p_u(\hat{x}_0)$ as follows:

$$\begin{aligned}
\rho[p(\hat{x}),q] &\approx \sum_{u=1}^{m}\sqrt{p_u(\hat{x}_0)q_u} + \sum_{u=1}^{m}\frac{1}{2}q_u\frac{1}{\sqrt{p_u(\hat{x}_0)q_u}}\left(p_u(\hat{x})-p_u(\hat{x}_0)\right) = \\
&= 2\sum_{u=1}^{m}\frac{1}{2}\sqrt{p_u(\hat{x}_0)q_u} + \sum_{u=1}^{m}\frac{1}{2}q_u\frac{\sqrt{p_u(\hat{x}_0)q_u}}{p_u(\hat{x}_0)q_u}\left(p_u(\hat{x})-p_u(\hat{x}_0)\right) = \\
&= \frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(\hat{x}_0)q_u} + \frac{1}{2}\sum_{u=1}^{m}\left[\sqrt{p_u(\hat{x}_0)q_u} + q_u\frac{\sqrt{p_u(\hat{x}_0)q_u}}{p_u(\hat{x}_0)q_u}\left(p_u(\hat{x})-p_u(\hat{x}_0)\right)\right] = \\
&= \underbrace{\frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(\hat{x}_0)q_u}}_{\text{independent of}\hat{x}} + \underbrace{\frac{1}{2}\sum_{u=1}^{m}p_u(\hat{x})\sqrt{\frac{q_u}{p_u(\hat{x}_0)}}}_{\text{to be maximized}}
\end{aligned} \tag{7}$$

Recalling (3), the term to be maximized can be written as

$$\begin{aligned}\frac{1}{2}\sum_{u=1}^{m} p_u(\hat{x})\sqrt{\frac{q_u}{p_u(\hat{x}_0)}} &= \frac{1}{2}\sum_{u=1}^{m}\left(C_h\sum_{i=1}^{n_h} k\left(\frac{\|x_i\|^2}{h^2}\right)\delta\left[b(x_i)-u\right]\right)\sqrt{\frac{q_u}{p_u(\hat{x}_0)}} = \\ &= \frac{C_h}{2}\sum_{i=1}^{n_h}\underbrace{\left(\sum_{u=1}^{m}\sqrt{\frac{q_u}{p_u(\hat{x}_0)}}\delta\left[b(x_i)-u\right]\right)}_{\doteq w_i} k\left(\frac{\|x_i\|^2}{h^2}\right) = \\ &= \frac{C_h}{2}\sum_{i=1}^{n_h} w_i k\left(\frac{\|x_i\|^2}{h^2}\right)\end{aligned} \tag{8}$$

Observe that the obtained expression represents a density estimate computed with kernel profile $k(x)$ at $\hat{x}$ in the current frame, with the data being weighted by w_i.

As it is reasonable, **a pixel has a heavy weight when its color is frequent in the model but is not frequent in the current candidate**, suggesting that the candidate location should move towards that pixel.

Mean shift

The mean shift iteration is:

$$\hat{x} \leftarrow \frac{\sum_{i=1}^{n_h} x_i w_i g\left(\frac{\|\hat{x} - x_i\|^2}{h^2}\right)}{\sum_{i=1}^{n_h} w_i g\left(\frac{\|\hat{x} - x_i\|^2}{h^2}\right)} \tag{9}$$

where $g(r) = -k'(r)$, and it is applied starting from $\hat{x}_0$, until convergence.

Bhattacharyya coefficient maximization

Algorithm Bhattacharyya coefficient maximization

Input: The target model $\{q_u\}_{u=1\dots m}$ and its location $\hat{x}_0$ in the previous frame.

Output: The location $\hat{x}$ in current frame.

1: Compute $\{p_u(\hat{x}_0)\}_{u=1\dots m}$ and evaluate $\rho[p(\hat{x}_0), q] = \sum_{u=1}^{m} \sqrt{p_u(\hat{x}_0) q_u}$.
2: Compute the weights $\{w_i\}_{i=1\dots n_h}$ according to (8).
3: Find the next location of the target candidate $\hat{x}$ according to (9).
4: Compute $\{p_u(\hat{x})\}_{u=1\dots m}$ and evaluate $\rho[p(\hat{x}), q] = \sum_{u=1}^{m} \sqrt{p_u(\hat{x}) q_u}$.
5: **while** $\rho[p(\hat{x}), q] < \rho[p(\hat{x}_0), q]$ **do**
   $\hat{x} \leftarrow \frac{1}{2}(\hat{x}_0 + \hat{x})$
   evaluate $\rho[p(\hat{x}), q] = \sum_{u=1}^{m} \sqrt{p_u(\hat{x}) q_u}$.
6: **end while**
7: **if** $\|\hat{x} - \hat{x}_0\| < \epsilon$ **then**
8: Stop.
9: **else**
10: $\hat{x}_0 \leftarrow \hat{x}$ and go to step 2.
11: **end if**

The while cycle of steps 5-6 is rarely entered (in (Comaniciu et al., 2000) it is reported that less than 0.1% of the performed maximizations yielded cases where the iterations 5-6 were necessary)

Example

Example (cont.)

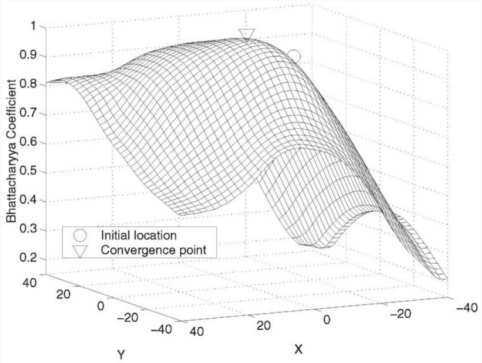


Figure from (Comaniciu et al., 2003). The similarity surface (values of the Bhattacharyya coefficient) corresponding to the rectangle marked in the rightmost frame of the above sequence.

Example (cont.)

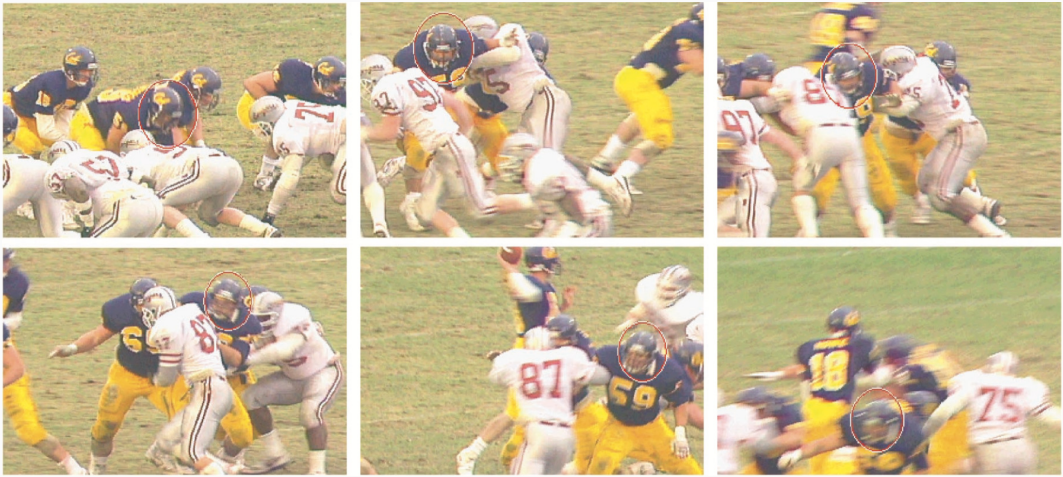


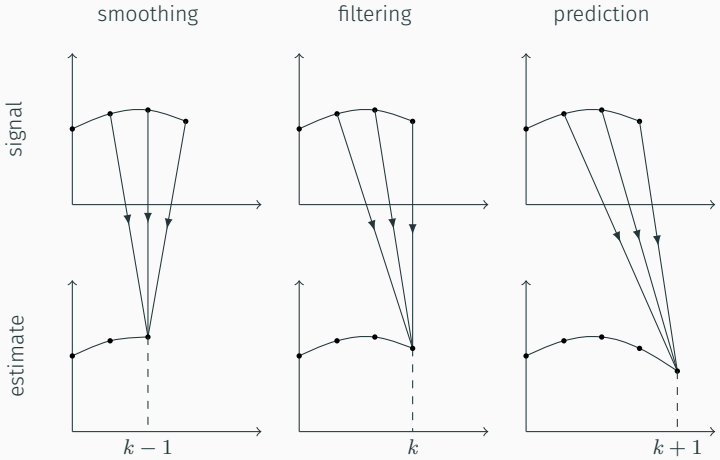
Figure from (Comaniciu et al., 2003).

Kalman Filter

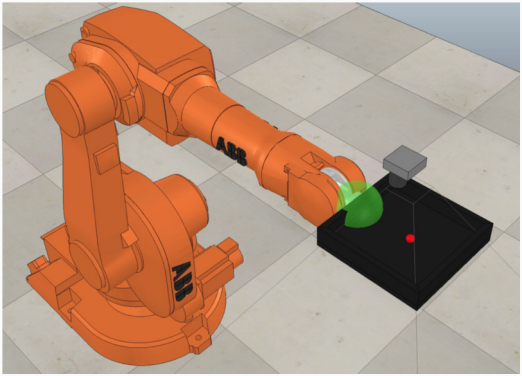
Kalman filter

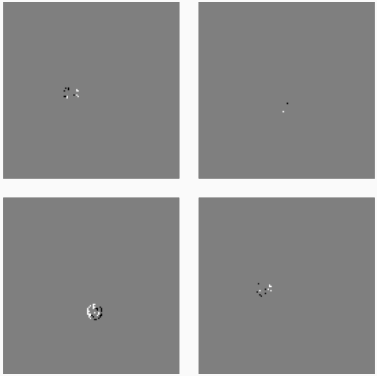
- The Kalman filter (Kalman, 1960) is a tool for estimating the state of a dynamic system from noisy observation.
- The Kalman filter is ubiquitous in science and engineering with applications such as:
  - navigation;
  - fault detection;
  - time series analysis;
  - econometrics;
  - seismology;
  - biology;
  - **tracking**.
- The Kalman filter relies on an **underlying dynamics** (expressed as a differential of difference equation) which is known and linear (extensions to the non-linear case exist).
- In the following we first introduce the Kalman filter from a general, system theoretic, point of view. Then we describe the application the problem of tracking.

Smoothing, filtering and prediction



A motivating example

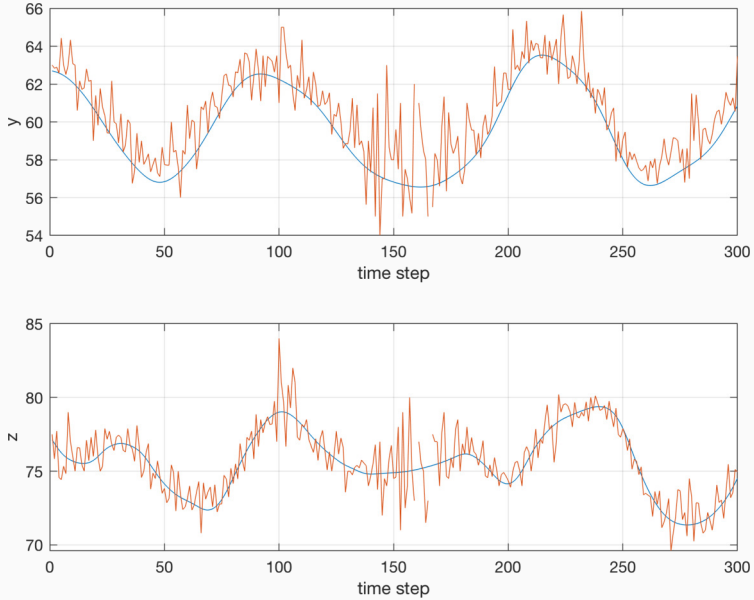




- Ball-and-plate system, equipped with event-based camera (it outputs a stream of events, each consisting of a pixel location, a polarity bit for positive or negative change in log intensity and a timestamp).
- Feedback based on visually estimated position and velocity of the ball.
- Very noisy measurement, especially for low velocity (too few pixels).

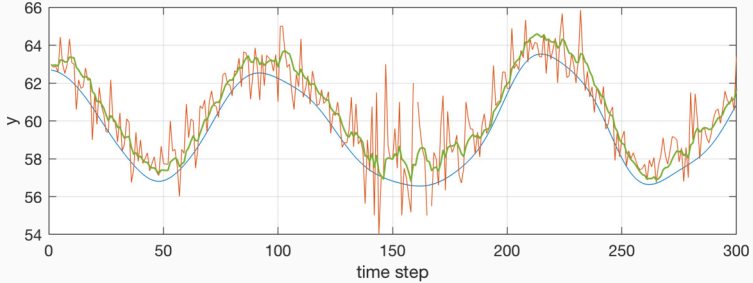
A motivating example (cont.)

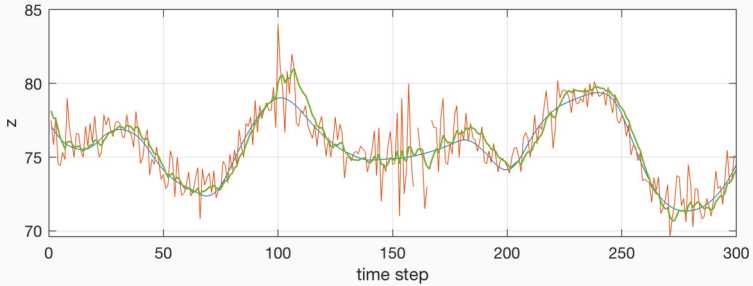
- Blue line: actual coordinates of the ball as a function of time;
- Orange line: detected coordinates based on the event-camera images;
- Detecting the center of mass of black and white pixels results in a rather noisy signal (unsuitable for feedback).



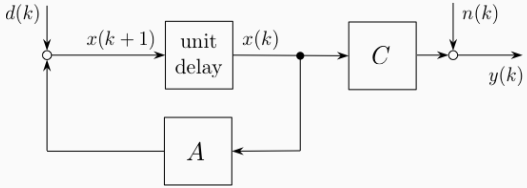
A motivating example (cont.)

- Blue lines: actual coordinates of the ball as a function of time;
- Orange lines: detected coordinates based on the event-camera images;
- Detecting the center of mass of black and white pixels results in a rather noisy signal (unsuitable for feedback).
- Green lines: result obtained with Kalman filter (based on a constant velocity dynamic model).





Assumptions



Consider a discrete-time linear system:

$$\begin{cases} x(k+1) = Ax(k) + d(k) \\ y(k) = Cx(k) + n(k) \end{cases} \tag{10}$$

- $x \in \mathbb{R}^n$ is the state;
- $A \in \mathbb{R}^{n \times n}$ is the state transition matrix;
- $y \in \mathbb{R}^p$ is the measured output;
- $C \in \mathbb{R}^{p \times n}$ is the output matrix;
- $d \in \mathbb{R}^n$ is the process disturbance;
- $n \in \mathbb{R}^p$ is the measurement disturbance (noise).

Assumptions (cont.)

We assume that $d(\cdot)$ and $n(\cdot)$ are zero-mean, discrete-time, mutually uncorrelated, white stochastic processes:

$$\mathbb{E}[d(k)] = 0$$
$$\mathsf{cov}(d(k), d(h)) = \mathbb{E}[d(k)d^\top(h)] = Q\delta(k-h), \quad Q \succeq 0$$
$$\mathbb{E}[n(k)] = 0$$
$$\mathsf{cov}(n(k), n(h)) = \mathbb{E}[n(k)n^\top(h)] = R\delta(k-h), \quad R \succ 0$$
$$\mathsf{cov}(n(k), d(h)) = \mathbb{E}[n(k)d^\top(h)] = 0 \quad \forall h, k$$

The initial state $x(0)$ is a random variable whose expectation and variance are known:

$$\mathbb{E}[x(0)] = x_0$$
$$\mathsf{var}(x(0)) = \mathbb{E}[(x(0) - x_0)(x(0) - x_0)^\top] = P_0$$

Moreover, d and n are uncorrelated to $x(0)$:

$$\mathbb{E}[x(0)n^\top(k)] = 0 \quad \forall k$$
$$\mathbb{E}[x(0)d^\top(k)] = 0 \quad \forall k$$

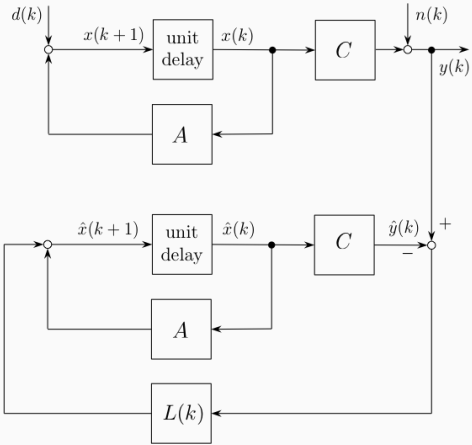
Observer

The aim is to design a dynamic system, called *observer*, in order to compute an estimate $\hat{x}$ of the state x from the noisy measurement y.

We take an observer of the form

$$\hat{x}(k+1) = A\hat{x}(k) + L(k)(y(k) - C\hat{x}(k)). \tag{11}$$

Observer (cont.)



The time-varying matrix $L(k) \in \mathbb{R}^{n \times p}$ has to be designed in order to get a good estimate $\hat{x}$.

Estimation error

We define the *estimation error* $e(k) = x(k) - \hat{x}(k)$.

By subtracting (10) and (11) we get the estimation error dynamics:

$$\begin{aligned} e(k+1) &= x(k+1) - \hat{x}(k+1) \\ &= (A - L(k)C)\, e(k) + d(k) - L(k)n(k). \end{aligned} \tag{12}$$

Criterion

$L(k)$ is called the *gain* of the observer.

We want to choose $L(k)$ optimally with respect to some performance index. A reasonable index is *the expected value of the squared estimation error*, called the *Mean Squared Error (MSE)*

$$\mathbb{E}[\|e(k)\|^2].$$

Notice that

$$\mathbb{E}[\|e(k)\|^2] = \mathbb{E}[\mathsf{tr}[e(k)e^\top(k)]] = \mathsf{tr}\,\mathbb{E}[e(k)e^\top(k)],$$

thus by defining

$$P(k) = \mathbb{E}[(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^\top] = \mathbb{E}[e(k)e^\top(k)],$$

the aim is to minimize the trace of $P(k)$.

Dynamics of the expected value of the error

By taking expectation of both sides of (12)

$$\begin{aligned}\mathbb{E}[e(k+1)] &= \mathbb{E}[(A - L(k)C)\, e(k) + d(k) - L(k)n(k)] \\ &= (A - L(k)C)\, \mathbb{E}[e(k)] + \mathbb{E}[d(k)] - L(k)\mathbb{E}[n(k)] \\ &= (A - L(k)C)\, \mathbb{E}[e(k)].\end{aligned} \tag{13}$$

- The expectation will propagate through time in the same way as the unperturbed system;
- At time $k = 0$, since the statistics of $x(0)$ is known, we can take

  $$\hat{x}(0) = x_0$$

  which implies

  $$\mathbb{E}[e(0)] = \mathbb{E}[x(0) - \hat{x}(0)] = \mathbb{E}[x(0) - x_0] = 0.$$

- if the initial state of the observer is taken as the expected value of the initial state, the estimation error $x(k) - \hat{x}(k)$ is zero-mean, *independent of the choice of* $L(k)$.

Dynamics of the variance of the error

The choice of $L(k)$ affects however the variance $P(k)$ of the estimation error.

Assuming that $\hat{x}(0) = x_0$, the variance of the estimation error at time $k = 0$ is

$$P(0) = \mathbb{E}[(x(0) - x_0)(x(0) - x_0)^\top] = P_0.$$

How does $P(k)$ propagates through time?

$$\begin{aligned} P(k+1) = \mathbb{E}[e(k+1)e^\top(k+1)] = (A - LC)\mathbb{E}[ee^\top](A - LC)^\top + \\ +(A - LC)\mathbb{E}[ed^\top] + (A - LC)\mathbb{E}[en^\top]L^\top + \mathbb{E}[de^\top](A - LC)^\top + \\ \mathbb{E}[dd^\top] + \mathbb{E}[dn^\top]L^\top + L\mathbb{E}[ne^\top](A - LC)^\top + L\mathbb{E}[nd^\top] + \\ L\mathbb{E}[nn^\top]L^\top \end{aligned}$$

where the dependence on k of L, e, d, n has been omitted for simplicity.

Observe that $d(k)$, $n(k)$ and $e(k)$ are uncorrelated. In particular, according to (12), $e(k)$ can be written as a linear combination of $x(0)$ and d and n up to time $k - 1$ (see for instance (Antsaklis and Michel, 2006), Section 1.15). However, $d(k)$ and $n(k)$ are uncorrelated to the past disturbances and to $x(0)$ by assumption.

Dynamics of the variance of the error (cont.)

Then the expectations of the mixed terms are zero, leading to :

$$P(k+1) = (A - LC)\mathbb{E}[ee^\top](A - LC)^\top + \mathbb{E}[dd^\top] + L\mathbb{E}[nn^\top]L^\top. \tag{14}$$

Thus the dynamics of the variance of the estimation error is governed by:

$$P(k+1) = (A - LC)P(A - LC)^\top + Q + LRL^\top. \tag{15}$$

Finding the optimal gain

We first consider the optimal prediction problem.

For each $k > 0$ the optimal gain is the matrix $L_o(k) \in \mathbb{R}^{n \times p}$ that minimizes the trace of $P(k+1)$ (i.e. the expected squared error of the predicted state)

$$L_o(k) = \underset{L(k)}{\text{argmin}}\ \text{tr}\, P(k+1).$$

Useful properties: denoting by $\nabla_X f(X)$ the gradient of a scalar function f of the matrix X, i.e.

$$[\nabla_X f(X)]_{ij} = \frac{\partial f(X)}{\partial X_{ij}},$$

the following identities hold

$$\nabla_X \,\text{tr}(AX) = A^\top, \qquad \nabla_X \,\text{tr}(AX^\top) = A.$$

Finding the optimal gain (cont.)

- The trace of $P(k+1)$ may be rewritten as:

$$\begin{aligned}\text{tr}\, P(k+1) = \text{tr}[(A-LC)P(A-LC)^\top + Q + LRL^\top] = \text{tr}[APA^\top] \\ -2\,\text{tr}[APC^\top L^\top] + \text{tr}[LCPC^\top L^\top] + \text{tr}\, Q + \text{tr}[LRL^\top]\end{aligned}$$

- To find the minimizing L, we differentiate w.r.t. L and equate to zero.

$$\nabla_L \,\text{tr}\, P(k+1) = 2L(CPC^\top + R) - 2APC^\top = 0 \tag{16}$$

- By solving (16) we get the optimal gain at time k:

$$L_o(k) = AP(k)C^\top (CP(k)C^\top + R)^{-1}. \tag{17}$$

Notice that, being $CPC^\top + R$ positive definite, the local extremum is actually a global minimum.

Riccati difference equation

By substituting (17) in (15) we get the *Riccati difference equation*:

$$P(k+1) = AP(k)A^\top - AP(k)C^\top (CP(k)C^\top + R)^{-1} CP(k)A^\top + Q. \tag{18}$$

- The Riccati difference equation tells how the variance P propagates, provided that the optimal gain $L_o(k)$ is applied.
- By solving iteratively the Riccati difference equation from the initial condition $P(0) = P_0$, and by using

$$L_o(k) = AP(k)C^\top (CP(k)C^\top + R)^{-1},$$

  the sequence of optimal gains can be found.
- $L_o(k)$ *does not depend on the actual realization of the processes* $d(\cdot)$ *and* $n(\cdot)$, but on their variances Q and R only.
- Thus , the optimal sequence can be computed off-line.

Optimal one-step-ahead predictor

The results achieved so far can be summarized in the following

Theorem (Kalman Filter - One-step-ahead predictor)

For the system (10), under the stated assumptions on the process disturbance and measurement noise, and assuming that the observer takes the form (11) and is initialized at $\hat{x}(0) = x_0$, the gain $L(k)$ as expressed by (17), where $P(k)$ is the solution of the Riccati equation (18) with initial condition $P(0) = P_0$, is optimal, in the sense that it minimizes the expected value of the squared estimation error for each k.

Optimal filter

- The predictor obtained so far produces an estimate $\hat{x}(k+1)$ of $x(k+1)$ based on the measurements up to the time k.
- Equivalently, it produces an estimate $\hat{x}(k)$ of $x(k)$ based on the measurements up to the time $k-1$.
- Why not exploit the measurement $y(k)$ (which is available at time k) to compute the estimate of the state at time k?

Some notation:

- Call $\hat{x}(i|j)$ the estimated state at time i *given the information up to time* j.
- Call $P(i|j)$ the variance of the error $\hat{x}(i|j) - x(i|j)$.

In the following we build on the predictor in order to be able compute, in an optimal way, $\hat{x}(k|k)$ along with the corresponding variance $P(k|k)$.

Optimal filter (cont.)

Consider the predictor

$$\hat{x}(k+1) = A\hat{x}(k) + L(k)(y(k) - C\hat{x}(k)), \tag{19}$$

and choose $L(k)$ optimally, as in the previous section. Thus:

- at time k, the prediction $\hat{x}(k|k-1)$ is available;
- such prediction is optimal because it minimizes $\mathbf{tr}\, P(k|k-1)$.

To exploit the newly available measurement $y(k)$ we define a *correction equation*:

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)(y(k) - C\hat{x}(k|k-1)). \tag{20}$$

- We try to improve the estimate by exploiting the newly available measurement;
- the correction term is proportional to the difference between the measured and the predicted output.

Finding the optimal gain

How can we choose the gain $K(k)$ in an optimal way? We can choose $K(k)$ such that it minimizes $E(\|e(k|k)\|^2)$, i.e. tr $P(k|k)$ where

$$P(k|k) = \mathbb{E}[(\hat{x}(k|k) - x(k))(\hat{x}(k|k) - x(k))^\top]. \tag{21}$$

From (20):

$$\begin{aligned}
\hat{x}(k|k) - x(k) &= \hat{x}(k|k-1) + K(k)(y(k) - C\hat{x}(k|k-1)) - x(k) \\
&= \hat{x}(k|k-1) + K(k)(Cx(k) + d(k) - C\hat{x}(k|k-1)) - x(k) \\
&= (I - K(k)C)(\hat{x}(k|k-1) - x(k)) + K(k)d(k) \\
&= (I - K(k)C)e(k|k-1) + K(k)d(k).
\end{aligned} \tag{22}$$

By substituting in (21) we get

$$P(k|k) = (I - K(k)C)\underbrace{E\left[e(k|k-1)e^\top(k|k-1)\right]}_{P(k|k-1)}(I - K(k)C)^\top + K(k)RK^\top(k). \tag{23}$$

Finding the optimal gain (cont.)

$$P(k|k) = (I - K(k)C)\underbrace{E\left[e(k|k-1)e^\top(k|k-1)\right]}_{P(k|k-1)}(I - K(k)C)^\top + K(k)RK^\top(k).$$

Notice that $P(k|k-1)$ depends only on the one-step-ahead predictor; in particular it does not depend on the gain $K(k)$.

By denoting $P(k|k-1)$ as P, and writing K instead of $K(k)$:

$$\mathsf{tr}\, P(k|k) = \mathsf{tr}\left[P - KCP - PC^\top K^\top + KCPC^\top K^\top + KRK^\top\right].$$

Now we find the minimizing K by differentiating and equating to zero:

$$\nabla_K \,\mathsf{tr}\, P(k|k) = 2K(CPC^\top + R) - 2PC^\top = 0 \tag{24}$$

Thus the optimal $K(k)$ is

$$K(k) = P(k|k-1)C^\top(CP(k|k-1)C^\top + R)^{-1}. \tag{25}$$

Algorithm for the Kalman filter (a first form)

Algorithm Kalman filter (a first form)

1: initialization

- $k \leftarrow 0$
- $\hat{x}(0|-1) \leftarrow x_0$
- $P(0|-1) \leftarrow P_0$

2: correction

- $K(k) \leftarrow P(k|k-1)C^\top (CP(k|k-1)C^\top + R)^{-1}$
- $\hat{x}(k|k) \leftarrow \hat{x}(k|k-1) + K(k)(y(k) - C\hat{x}(k|k-1))$
- $P(k|k) \leftarrow (I - K(k)C)P(k|k-1)(I - K(k)C)^\top + K(k)RK^\top(k)$

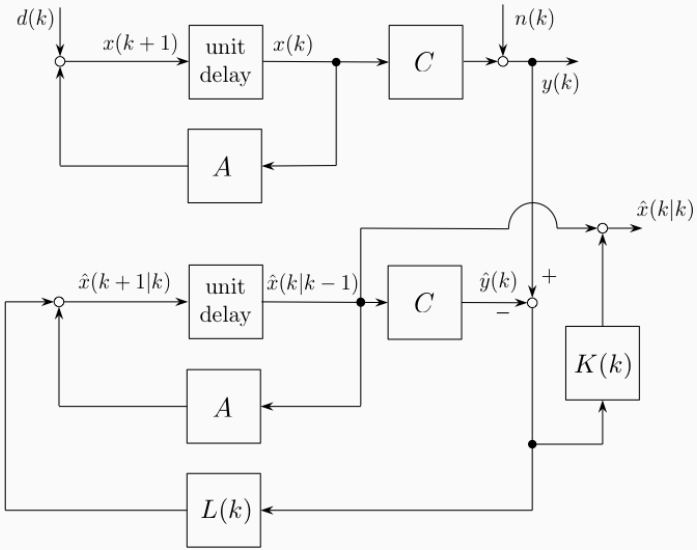
3: prediction

- $L(k) = AP(k|k-1)C^\top (CP(k|k-1)C^\top + R)^{-1}$
- $\hat{x}(k+1|k) \leftarrow A - L(k)C)\hat{x}(k|k-1) + L(k)y(k)$
- $P(k+1|k) \leftarrow AP(k|k-1)A^\top - AP(k|k-1)C^\top (CP(k|k-1)C^\top + R)^{-1} CP(k|k-1)A^\top + Q$

4: iterate

- $k \leftarrow k+1$
- go to step 2

Algorithm for the Kalman filter (a first form) (cont.)



Algorithm for the Kalman filter (compact form)

- The previous form shows that the filter is obtained by combining optimal prediction and correction.
- However, the form in which the Kalman filter is usually presented and implemented is more compact.

In the following we derive the most common and compact form.

By comparing (25) to (17) and recalling that $P(k)$ in (17) is actually $P(k|k-1)$, we find that the optimal gains L and K are related:

$$L(k) = AK(k).$$

Thus, the prediction equation:

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + L(k)(y(k) - C\hat{x}(k|k-1)), \tag{26}$$

may be rewritten in terms of $\hat{x}(k|k)$:

$$\begin{aligned} \hat{x}(k+1|k) &= A\hat{x}(k|k-1) + AK(k)(y(k) - C\hat{x}(k|k-1)) \\ &= A\left[(I - K(k)C)\hat{x}(k|k-1) + K(k)y(k)\right] \\ &= A\hat{x}(k|k) \end{aligned} \tag{27}$$

Similarly, the update of $P(k+1|k)$ can be expressed in terms of $P(k|k)$: first observe that (dropping the dependence on k)

$$P(k+1|k) = A(P - PC^\top (CPC^\top + R)^{-1} CP)A^\top + Q, \tag{28}$$

where P stands for $P(k|k-1)$. By substituting (25) in (23) we get

$$\begin{aligned} P(k|k) &= (I-KC)P(I-KC)^\top + KRK^\top = \\ &= P - KCP - PC^\top K^\top + KCPC^\top K^\top + KRK^\top = \\ &= P - KCP - PC^\top K^\top + K(CPC^\top + R)K^\top = \\ &= P - 2PC^\top (CPC^\top + R)^{-1} CP + PC^\top (CPC^\top + R)^{-1} CP = \\ &= P - PC^\top (CPC^\top + R)^{-1} CP. \end{aligned}$$

Comparing to (28) we get

$$P(k+1|k) = AP(k|k)A^\top + Q. \tag{29}$$

Algorithm for the Kalman filter (compact form) (cont.)

Algorithm Kalman filter (compact form)

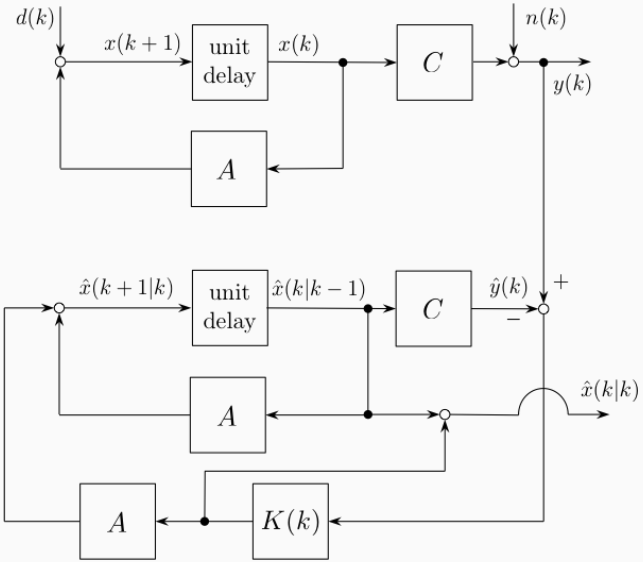
1: initialization
   - $k \leftarrow 0$
   - $\hat{x}(0|-1) \leftarrow x_0$
   - $P(0|-1) \leftarrow P_0$

2: correction
   - $K(k) \leftarrow P(k|k-1)C^\top (CP(k|k-1)C^\top + R)^{-1}$
   - $\hat{x}(k|k) \leftarrow \hat{x}(k|k-1) + K(k)(y(k) - C\hat{x}(k|k-1))$
   - $P(k|k) \leftarrow (I - K(k)C)P(k|k-1)(I - K(k)C)^\top + K(k)RK^\top(k)$

3: prediction
   - $\hat{x}(k+1|k) \leftarrow A\hat{x}(k|k)$
   - $P(k+1|k) \leftarrow AP(k|k)A^\top + Q$

4: iterate
   - $k \leftarrow k+1$
   - go to step 2

Algorithm for the Kalman filter (compact form) (cont.)



Integrating measurements with model predictions

- The Kalman filter puts a dynamic model between the measurement and the estimate.
- We do not just believe the noisy measurements of $p(k)$ of the feature detector: the filter integrates them with model predictions to obtain better estimates.
- The integration is governed by the relative weight of Q and R:
  - if the entries of R are much smaller that those of Q, then the gain of the filter is large (thus the filter relies more on the measurements than on the model prediction);
  - if the entries of Q are much smaller that those of R, then the gain of the filter is small (thus the filter relies more on the model than on the measurements).
- In practice, some trial-and-error adjustments of R and Q are necessary to obtain a proper balance.

Quantifying the state uncertainty

- The Kalman filter **quantifies the uncertainty on the state estimate**, through the state covariance matrix $P(k)$.
- The feature detector can choose the size of the search area in the next frame based on that uncertainty. The uncertainty region in the state space is represented by the ellipsoid:

$$\mathcal{E} = \left\{ x : (x - \hat{x}(k))^\top P^{-1}(k)(x - \hat{x}(k)) \leq \alpha^2 \right\},$$

  where α is a design parameter. Thus, the search area $\mathcal{A}$ can be chosen as the *projection* of that ellipsoid in the output (measurement) space:

$$\mathcal{A} = \{ y = Cx : x \in \mathcal{E} \}.$$

- The entries of P_0 can be initialized to large values as in a well-designed filter they decrease and reach a steady state rapidly, thereby restricting the search region.

On the optimality of the Kalman filter

- The estimator described so far is optimal (meaning that it minimizes the Mean Squared Error) among the estimators of the form (11).
- Other estimators of different form may exist that achieve a smaller MSE.
- However, if we make the further assumption that the disturbance and the noise are **Gaussian** processes, then the estimator can be shown to be **optimal among all the possible estimators** (Moore and Anderson, 1979).
- In that case, the Kalman filter is the Minimum Mean Squared Error (MMSE) estimator. The MMSE estimator is well-known to be the expectation of the variable to be estimated given the observation, thus it provides the conditional mean of the state at time $k+1$ given the observations up to time k:

$$\hat{x}(k+1|k) = \mathbb{E}[x(k+1)|y(0), y(1), \ldots, y(k)].$$

Application to tracking (random walk)

Given a sequence of frames $I(k)$, $k = 0, 1, \ldots$, acquired every Δt time units, consider a feature point $p(k) = [u(k) \;\; v(k)]^\top$ and assume that the point is moving under random walk. Then, its new position is its old position plus some disturbance term. By defining the state vector as

$$x(k) = p(k),$$

the system model can be described as

$$x(k+1) = x(k) + d(k)$$

where $d(k)$ is the system disturbance (assumed to be a zero-mean white stochastic process of covariance $Q \succeq 0$).

Application to tracking (random walk) (cont.)

Assuming that a feature detector detects ("measures") the location $p(k)$ of the feature point at each frame, the measurement model becomes:

$$y(k) = x(k) + n(k)$$

where $n(k)$ is the measurement noise (assumed to be a zero-mean white stochastic process of covariance $R \succ 0$).

To complete the model, the estimate x_0 of the initial state (position of the feature point in the first frame) and the state covariance matrix P_0 are needed.

Application to tracking (constant velocity)

Given a sequence of frames $I(k)$, $k = 0, 1, \ldots$, acquired every Δt time units, consider a feature point $p(k) = [u(k) \;\; v(k)]^\top$ moving at constant velocity in the image plane. Denote as $\dot{p}(k) = [\dot{u}(k) \;\; \dot{v}(k)]^\top$ its velocity. By defining the state vector as

$$x(k) = [u(k) \;\; v(k) \;\; \dot{u}(k) \;\; \dot{v}(k)]^\top,$$

the system model can be described by

$$x(k+1) = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(k) + d(k)$$

where $d(k)$ is the system disturbance (assumed to be a zero-mean white stochastic process of covariance $Q \succeq 0$).

Application to tracking (constant velocity) (cont.)

Assuming that a feature detector detects ("measures") the location $p(k)$ of the feature point at each frame, the measurement model becomes:

$$y(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(k) + n(k)$$

where $n(k)$ is the measurement noise (assumed to be a zero-mean white stochastic process of covariance $R \succ 0$).

To complete the model, the estimate x_0 of the initial state (position and velocity of the feature point in the first frame) and the state covariance matrix P_0 are needed.

Application to tracking (constant acceleration)

Given a sequence of frames $I(k)$, $k = 0, 1, \ldots$, acquired every Δt time units, consider a feature point $p(k) = [u(k) \;\; v(k)]^\top$ moving at constant acceleration in the image plane. Denote as $\dot{p}(k) = [\dot{u}(k) \;\; \dot{v}(k)]^\top$ and $\ddot{p}(k) = [\ddot{u}(k) \;\; \ddot{v}(k)]^\top$ its velocity and acceleration, respectively. By defining the state vector as

$$x(k) = [u(k) \;\; v(k) \;\; \dot{u}(k) \;\; \dot{v}(k) \;\; \ddot{u}(k) \;\; \ddot{v}(k)]^\top,$$

the system model can be described by

$$x(k+1) = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x(k) + d(k)$$

where $d(k)$ is the system disturbance (assumed to be a zero-mean white stochastic process of covariance $Q \succeq 0$).

Application to tracking (constant acceleration) (cont.)

Assuming that a feature detector detects ("measures") the location $p(k)$ of the feature point at each frame, the measurement model becomes:

$$y(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} x(k) + n(k)$$

where $n(k)$ is the measurement noise (assumed to be a zero-mean white stochastic process of covariance $R \succ 0$).

To complete the model, the estimate x_0 of the initial state (position, velocity and acceleration of the feature point in the first frame) and the state covariance matrix P_0 are needed.

References

References

Antsaklis, P. J. and Michel, A. N. (2006). *Linear Systems*. Springer Science & Business Media.

Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, 2(8):142–149.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 25(5):564–577.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40.

Ghassabeh, Y. A. (2013). On the convergence of the mean shift algorithm in the one-dimensional space. *Pattern Recognition Letters*, 34(12):1423–1427.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, pages 674–679, Vancouver (BC).

Moore, J. B. and Anderson, B. D. (1979). *Optimal filtering*. Prentice-Hall.

References (cont.)

Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, number June, pages 593–600. IEEE.

Tomasi, C. and Kanade, T. (1991). Detection and Tracking of Point Features Technical Report CMU-CS-91-132. Technical report, School of Computer Science, Carnegie Mellon Univ. Pittsburgh.

554SM –Fall 2018

Lecture 5
Tracking

END