Short course

A vademecum of statistical pattern recognition techniques with applications to image and video analysis

Lecture 6
The Kalman filter. Particle filters

Massimo Piccardi
University of Technology, Sydney, Australia

Agenda

- Recursive Bayesian estimation
- The Kalman filter
- Particle filters
- A mention to probabilistic data association
- Example papers
- References

State space models

- In many cases, we are interested in systems whose state is a continuous multivariate r. v. (as opposed to HMM where it is discrete). Such systems are often referred to as *state(-)space models*
- Their evolution in time is described by two equations, the *state equation*, or *system*, or *process*, *model,* and the *output equation*, or *measurement model*
- Our further assumptions are: i) time-discrete evolution; ii) time-invariant parameters; iii) no control inputs; iv) noise over the state transitions and on the measurements; v) first-order Markov process; vi) independence of measurements given the state (aka Bayesian tracking hypotheses)

State space models

- Under these assumptions, we obtain the following equations:

$$x_k = f\left(x_{k-1}, v_{k-1}\right)$$
$$z_k = h\left(x_k, n_k\right)$$

- x_k is the state r.v. $\in R^n$
  v_k is the process noise r.v. $\in R^l$
  z_k is the measurement r.v. $\in R^p$
  n_k is the measurement noise r.v. $\in R^r$

Recursive Bayesian estimation

- Problem: estimate the state of a system, x_k, from the sequence of received measurements, $Z_k = \{z_i, i=1\ldots K\}$
- Recursive estimation: every new $z_k \rightarrow$ new x_k
- Estimating the state of a system means to estimate its pdf given the received measurements, **$p(x_k|Z_k)$**

  $p(x_k|Z_k)$ is known as the *filtering density* and is a marginal of the *posterior density* $p(X_k|Z_k)$
- Typical application: target tracking

Recursive Bayesian estimation

- How do we recursively estimate $p(x_k|Z_k)$?
  - At time k, $p(x_{k-1}|Z_{k-1})$ is available
  - Repeatedly apply the Bayes theorem and use the independence of observations given the state:

$$p(x_k \mid Z_k) = \frac{p(Z_k \mid x_k)p(x_k)}{p(Z_k)} = \frac{p(z_k, Z_{k-1} \mid x_k)p(x_k)}{p(z_k, Z_{k-1})} =$$

$$= \frac{p(z_k \mid Z_{k-1}, x_k)p(Z_{k-1} \mid x_k)p(x_k)}{p(z_k \mid Z_{k-1})p(Z_{k-1})} = \frac{p(z_k \mid Z_{k-1}, x_k)p(x_k)}{p(z_k \mid Z_{k-1})p(Z_{k-1})}\frac{p(x_k \mid Z_{k-1})p(Z_{k-1})}{p(x_k)} =$$

$$= \frac{p(z_k \mid Z_{k-1}, x_k)p(x_k \mid Z_{k-1})}{p(z_k \mid Z_{k-1})} = \frac{p(z_k \mid x_k)p(x_k \mid Z_{k-1})}{p(z_k \mid Z_{k-1})}$$

Recursive Bayesian estimation

posterior* filtering density** — ***likelihood — ***prior* filtering density** — ***evidence***

$$p(x_k \mid Z_k) = \frac{p(z_k \mid x_k)\, p(x_k \mid Z_{k-1})}{p(z_k \mid Z_{k-1})}$$

- prior: $p(x_k \mid Z_{k-1}) = \int p(x_k \mid x_{k-1})\, p(x_{k-1} \mid Z_{k-1})\, dx_{k-1}$

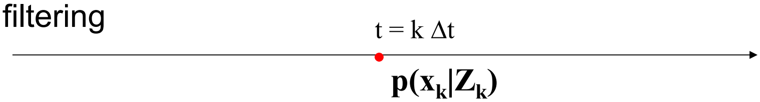
  because $p(x_k \mid x_{k-1}) = p(x_k \mid x_{k-1}, Z_{k-1})$ (Markov)
- likelihood: given by the measurement model
- evidence: $p(z_k \mid Z_{k-1}) = \int p(z_k, x_k \mid Z_{k-1})\, dx_k =$

  $= \int p(z_k \mid x_k, Z_{k-1})\, p(x_k \mid Z_{k-1})\, dx_k =$

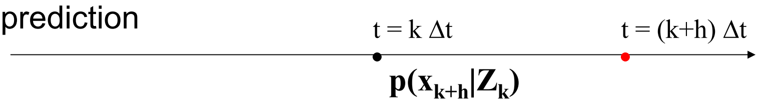
  $= \int p(z_k \mid x_k)\, p(x_k \mid Z_{k-1})\, dx_k$

Other related problems

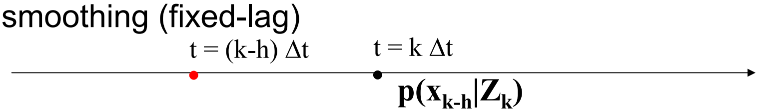
- filtering



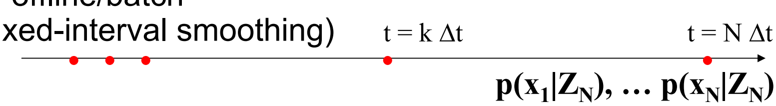
- prediction



- smoothing (fixed-lag)



- offline/batch (fixed-interval smoothing)



The Kalman filter

- "Linear/Gaussian" assumptions: linear system and measurement models, Gaussian distributions for all random variables
- If $p(x_0)$ and noise distributions are assumed Gaussian, subsequent distributions remain Gaussian under linear transformations

  $X \sim N(\mu, \Sigma)$
  $Y = AX + K$
  $\rightarrow Y \sim N(A\mu + K, A\Sigma A^T)$

- An optimal solution exists (R.E. Kalman, 1960)

The Kalman filter

- The assumptions lead to the following equations:

$$x_k = Ax_{k-1} + v_{k-1}$$
$$z_k = Hx_k + n_k$$

- x_k is the state r.v. $\in R^n$
  v_k is the process noise r.v. $\in R^n$, independent, $\sim N(0,Q)$
  A is the state transition matrix (n x n)
  z_k is the measurement r.v. $\in R^p$
  n_k is the measurement noise r.v. $\in R^p$, independent, $\sim N(0,R)$
  H is the state transition matrix (p x n)

Solution

- The target is the filtering distribution's pdf, $p(x_k|Z_k)$; however, since it is Gaussian, mean and covariance suffice to identify it fully
- Usually, the mean at time interval k is noted as $\hat{x}_k$, the covariance as P_k and called the *error covariance*
- The solution is obtained in two steps:
  - the **time update**, i.e. the application of the dynamics (system model); a *prediction*; quantities are noted as $\hat{x}_k^-$, P_k^- and called a-priori estimates
  - the **measurement update**, i.e. the use of the measurement (measurement model); a *correction* to the previous estimates yielding the a-posteriori estimates

Time update equations

- A-priori estimates:

$$\hat{x}_k^- = A\hat{x}_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

- Directly from transformation of Gaussian r.v.
- Optimal estimate in the absence of measurement
- Process noise causes P to increase

Weighting

- A weighting matrix, K_k (n x p), called the *Kalman gain*:

$$K_k = P_k^- H^T \left(H P_k^- H^T + R \right)^{-1}$$

- K_k decides how much the a-priori estimates should be corrected by the k-th observation, z_k: the larger the measurement noise, R (uncertainty on the observation), the smaller the correction

Measurement update equations

- A-posteriori estimates:

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H \hat{x}_k^- \right)$$
$$P_k = \left(I - K_k H \right) P_k^-$$

- $z_k - H\hat{x}_k^-$ is the difference between the actual and predicted observations (aka *innovation* or *measurement residual*)

Drift model

- State vector contains only the position of the target
- Constant position assumption – x changes only due to the effect of noise
- 1-D example (u: position)

$$x = [u] \qquad A = [1]$$

Constant velocity model

- State vector contains position and velocity of the target
- Constant velocity assumption
- 1-D example (u: position; v: velocity)

$$x = \begin{bmatrix} u \\ v \end{bmatrix} \qquad A = \begin{bmatrix} 1 & \Delta_t \\ 0 & 1 \end{bmatrix}$$

Constant acceleration model

- State vector contains position, velocity and acceleration of the target
- Constant acceleration assumption
- 1-D example (u: position; v: velocity; a: acceleration)

$$x = \begin{bmatrix} u \\ v \\ a \end{bmatrix} \quad A = \begin{bmatrix} 1 & \Delta_t & 0 \\ 0 & 1 & \Delta_t \\ 0 & 0 & 1 \end{bmatrix}$$

Periodic motion model

- State vector contains position and velocity of the target
- Periodic motion assumption: $d^2u/dt^2 = -cu$
- 1-D example (u: position; v: velocity)

$$x = \begin{bmatrix} u \\ v \end{bmatrix} \quad A = \begin{bmatrix} 1 & \Delta_t \\ -c\Delta_t & 1 \end{bmatrix}$$

Example

- An example courtesy of Greg Welch and Gary Bishop, An Introduction to the Kalman Filter, ACM SIGGRAPH 2001 tutorial
- 2D data from a PC tablet
- A rich resource page at http://www.cs.unc.edu/~welch/kalman/
- A Java-based 1-D Kalman Filter Learning Tool

Drift model

- Process model:

$$x = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} Q_{xx} & 0 \\ 0 & Q_{yy} \end{bmatrix}$$

- Measurement model (z: measured position):

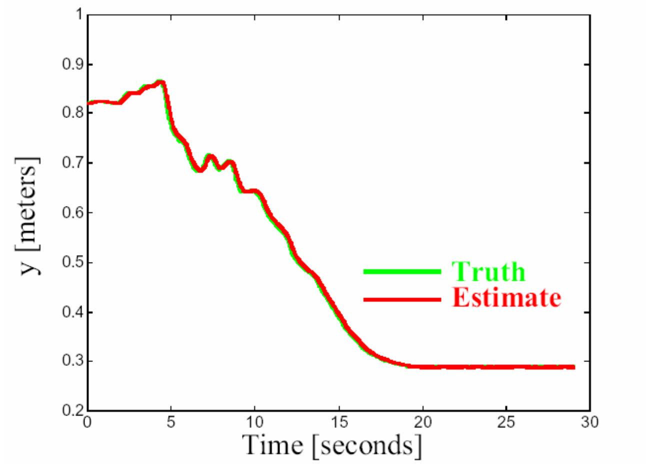
$$z = \begin{bmatrix} z_x \\ z_y \end{bmatrix} \qquad H = \begin{bmatrix} H_x & 0 \\ 0 & H_y \end{bmatrix} \qquad R = \begin{bmatrix} R_{xx} & 0 \\ 0 & R_{yy} \end{bmatrix}$$

- Initialization:

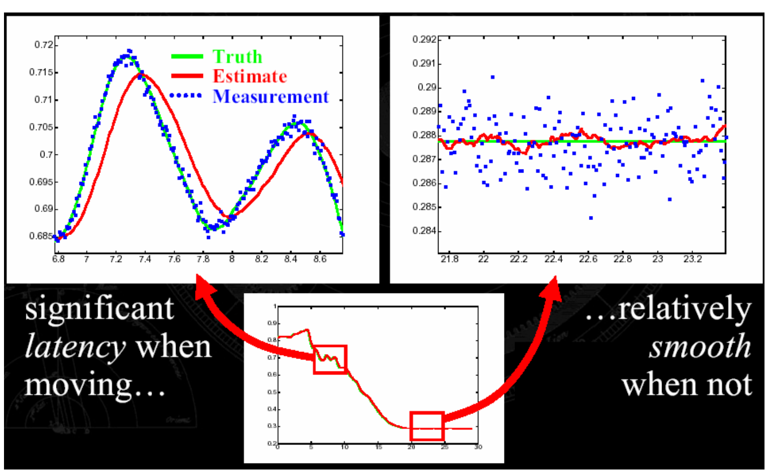
$$\hat{x}_0 = H^{-1} z_0 \qquad P_0 = \begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix}$$

Results

- y co-ordinate: first moving, then still



Results: highlights



Constant velocity model

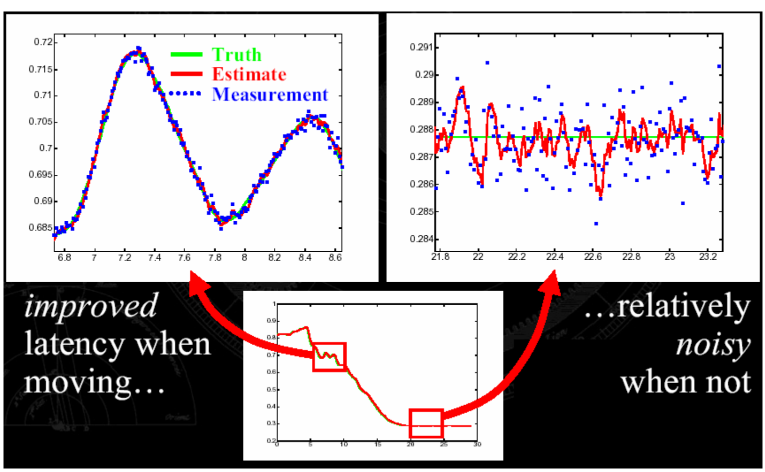
- Process model:

$$x = \begin{bmatrix} u_x \\ u_y \\ v_x \\ v_y \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Measurement model (same as before):

$$z = \begin{bmatrix} z_x \\ z_y \end{bmatrix} \qquad H = \begin{bmatrix} H_x & 0 & 0 & 0 \\ 0 & H_y & 0 & 0 \end{bmatrix}$$

Results: highlights



Beyond linear/Gaussian

- The Gaussian modelling of distributions proper of the Kalman filter is restrictive is many cases
- If transformations are not linear, subsequent distributions would become non-Gaussian in any case
- Many other models have been proposed, the main of which are named here:
  - Extended Kalman filter
  - Grid filter
  - Unscented Kalman filter
  - Particle filters
- A nice punch line: the Kalman filter provides an exact solution for an approximate model; we may prefer an approximate solution for an exact model

Particle filters

- In the following, we will focus on particle filters
- In particle filters, distributions are represented by weighted sets of samples (called *particles*)
- Particle filters are not a single algorithm, rather a family of methods, also known as *sequential Monte Carlo* methods
- The main target is again the filtering density, $p(x_k|Z_k)$
- Distributions are not restricted to be Gaussian
- The model may be linear or not linear

Monte Carlo methods

- In many practical cases, it is hard to evaluate certain target distributions, and expectations based on them
- In such cases, approximation schemes are needed (either deterministic or probabilistic)
- *Monte Carlo methods* are probabilistic methods where a distribution is simply represented by a set of its samples
- These techniques were first adopted by physicists; the name was coined by N. C. Metropolis based on the gambling habits of a colleague's relative

Monte Carlo: expectations

An expectation based on the exact integral:

$$E[f(x)] = \int_X f(x)p(x)dx$$

is estimated by:

$$\frac{1}{L}\sum_{l=1}^{L} f(x^l)$$

where the x^l are L samples from p(x) (requiring that p(x) can be sampled, and possibly easily)

Sequential Monte Carlo methods

- In particle filters, distributions are represented a la Monte Carlo by a set of their samples
- Given that observations become available one by one, the filtering density, $p(x_k|Z_k)$, is, as usual, estimated recursively
- This has paved the way for a number of sequential (i.e. recursive) Monte Carlo methods; here, we will see:
  - Sequential Importance Sampling
  - Sequential Importance Sampling with Resampling
  - Sampling Importance Resampling

Importance sampling

- Before we can jump the gun on particle filters, we need to cover two introductory topics:
  - Importance sampling
  - Resampling
- *Importance sampling* is a sampling technique based on the following assumptions:
  - we want to sample p(x); yet, it is hard
  - we can easily sample another distribution, q(x) (known as the *proposal distribution* or *importance density*)
  - we can easily evaluate both p(x) and q(x) (which means: given x, we can easily compute p(x) and q(x), at least up to proportionality)

Importance sampling

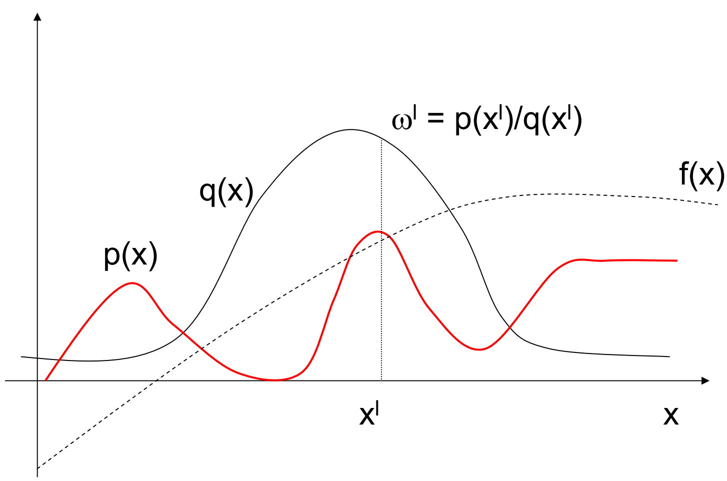
$$E[f(x)] = \int_X f(x)p(x)dx =$$

$$= \int_X f(x)\frac{p(x)}{q(x)}q(x)dx \cong \frac{1}{L}\sum_{l=1}^{L} f(x^l)\frac{p(x^l)}{q(x^l)}$$

where the x^l are sampled from q(x)

- Quantities $\omega^l = p(x^l)/q(x^l)$ are called the *importance weights*
- The weighted set of samples $\{x^l, \omega^l\}$ is a representation for p(x)
- For efficiency, q(x) should be large where f(x)p(x) is

Example



Resampling

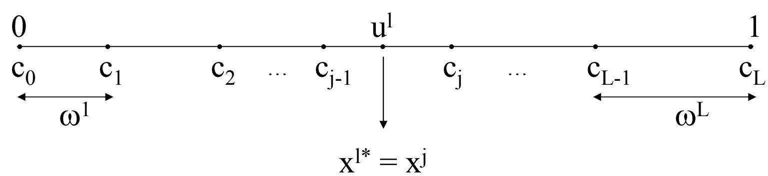
- Given a distribution represented by a weighted set of its samples, $\{x^l, \omega^l\}$, l:1…L, we sample it to obtain a replacement set of L samples, x^{l*}, all of equal weight, $\omega^{l*} = 1/L$
- The x^{l*} samples take value in the $\{x^l\}$ set
- Auxiliary assumption: samples can be generated out of a uniform distribution between 0 and 1, U[0,1]

Resampling

- Construct the cdf of the weighted set:

$$c_i = c_{i-1} + \omega^i \qquad c_0 = 0,\ i = 1\ldots L$$

- Sample U[0,1] L times to obtain the u^l samples
- u^l falls in the j-th interval $\rightarrow x^{l*} = x^j$

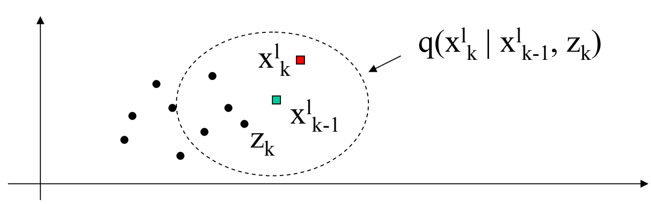


SIS particle filter

- *Sequential importance sampling* (SIS) is the sequential extension of importance sampling
- It can be used to recursively estimate the filtering density, $p(x_k|Z_k)$
- L samples, x^l_k, are drawn out of a proposal distribution, $q(x^l_k \mid x^l_{k-1}, z_k)$, at every time k
- The advantage of the sequential approach is that new weights ω^l_k need only be adjusted from previous weights, ω^l_{k-1}, through the dynamics and measurement models, and then normalised to add up to 1

SIS particle filter

- Sampling:



- Weight update:

$$\omega^l_k = \omega^l_{k-1} \frac{p\left(z_k \mid x^l_k\right) p\left(x^l_k \mid x^l_{k-1}\right)}{q\left(x^l_k \mid x^l_{k-1}, z_k\right)}$$

- Filtering density:

$$p\left(x_k \mid Z_k\right) \cong \sum_{l=1}^{L} \omega^l_k \, \delta\left(x_k - x^l_k\right)$$

SIS particle filter

- SIS particle filter:

  At every k,

  - draw the x^l_k from the current proposal distribution
  - update weights ω^l_k based on formula
  - normalise weights

  the new set of samples and weights is an approximation for the desired filtering density, $p(x_k|Z_k)$

SIS particle filter: weight degeneracy

- Degeneracy of the weights: it was shown that the variance of the ω^l_k weights can only increase over time: after a few iterations, all but one particle will have weight $\omega^l_k = 0$
- Useful measure of degeneracy:

$$\hat{N}_{eff}^{-1} = \sum_{l=1}^{L} \left(\omega_k^l\right)^2$$

- There are two practicable countermeasures:
  1. choice of a good proposal function
  2. resampling

Choice of proposal function

- It was shown that the optimal proposal function to sample each x^l_k is:

$$p\left(x_k \mid x^l_{k-1}, z_k\right)$$

since it minimises the degeneracy; yet, its use is not possible in most cases

- Often, the proposal function is taken as:

$$p\left(x_k \mid x^l_{k-1}\right)$$

this simplifies the weight update formula greatly; yet, it ignores z_k in the sampling of x^l_k: it may sample away from useful directions

Resampling

- Degeneracy can be monitored at every iteration; if > threshold, resampling is applied
- SIS particle filter *with resampling*:

  At every k,

  – Draw the x^l_k from the current proposal distribution
  – update weights ω^l_k based on formula
  – normalise weights
  – measure degeneracy: if > threshold, *resample*

SIR particle filter

- The *sampling importance resampling* (SIR) particle filter is a particle filter that uses $p(x_k \mid x^l_{k-1})$ as proposal and resamples at *every* time k
- SIR particle filter:

  At every k,
  - draw the x^l_k from $p(x_k \mid x^l_{k-1})$
  - compute weights ω^l_k; formula simplifies because of proposal and previous weights being all equal to 1/L
  - normalise weights
  - resample

SIR particle filter: issues

- The SIR particle filter uses a simple proposal distribution and avoids weight degeneracy by frequent resampling
- Yet, the proposal distribution ignores z_k in the sampling of x^l_k: it may sample away from useful directions
- Moreover, a very frequent resampling tends to create *sample impoverishment*: the samples with highest weights tend to be sampled more often and create x^l_k that are all equal; the degeneracy of the ω^l_k weights is mollified at the cost of a degeneracy in the x^l_k samples

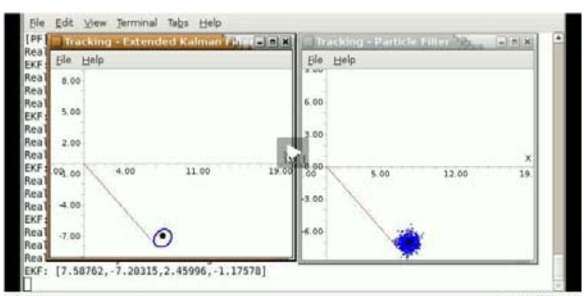
Other particle filters

- From Wikipedia (Jan 09):
  - Auxiliary particle filter
  - Gaussian particle filter
  - Unscented particle filter
  - Monte Carlo particle filter
  - Gauss-Hermite particle filter
  - Cost Reference particle filter
  - Rao-Blackwellized particle filter

Demo

Particle filter (PF) classes in the Mobile Robot Programming Toolkit (MRPT)

http://babel.isa.uma.es/mrpt/index.php/Particle_Filters



Probabilistic data association

- At least a mention must go to *probabilistic data association* (PDA)
- Correspondence between observations and tracked targets may be challenging and outstrip the predictive capability of the filter
- Single target, multiple observations: PDA filter (PDAF)
- Multiple targets: joint PDAF (JPDAF)
- Other algorithms: multiple hypothesis tracker (MHT), interacting multiple model (IMM, IMMJPDAF), integrated PDA (IPDA), …

Example papers

- Application: *tracking of visual objects*

  Tracking of multiple targets in videos.
- **Tracking with particle filters**:

  M. Isard, A. Blake, "CONDENSATION -- conditional density propagation for visual tracking," Int. J. Computer Vision, vol. 29, no. 1, pp. 5-28, 1998.

  2051 cites on Google Scholar, 18 Nov 2008
- **Tracking and data association with EM**:

  H. Tao, H. S. Sawhney, R. Kumar, "Object Tracking with Bayesian Estimation of Dynamic Layer Representations," IEEE Trans. on Pattern Anal. and Machine Intell., vol. 24, no. 1, pp. 75-89, 2002.

References

Recursive Bayesian estimation:

- M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking,” IEEE Trans. on Signal Processing, 50(2):174-188, February 2002
- S. Dablemont, Université catholique de Louvain, Introduction to particle filters, 2006 (slide set)
- K. Murphy, A tutorial on dynamic Bayesian networks, 2002 (slide set)

Kalman and particle filters:

- Materials from Greg Welch and Gary Bishop, University of North Carolina at Chapel Hill:
  – The Kalman Filter, http://www.cs.unc.edu/~welch/kalman/ (online resource)
  – An Introduction to the Kalman Filter, STC Lecture Series (slides)
  – An Introduction to the Kalman Filter, SIGGRAPH 2001 (slides and paper)
  – An Introduction to the Kalman Filter, TR95-041, University of North Carolina at Chapel Hill (tech rep)

References (2)

- Peter S. Maybeck, Stochastic models, estimation, and control. Volume 1, Academic Press, 1979
- Siome Klein Goldenstein, “A gentle Introduction to predictive filters,” Revista de Informática Teórica e Aplicada - RITA, Volume 11, vol. 11, no. 1, 2004
- M S Arulampalam, S Maskell, N Gordon, and T Clapp, idem

Probabilistic data association:

- Kirubarajan, T, Bar-Shalom, Y., “Probabilistic data association techniques for target tracking in clutter,” Proceedings of the IEEE, vol. 92, no. 3, Mar 2004, pp. 536 - 557
- I. Cox, “A Review of Statistical Data Association Techniques for Motion Correspondence,” Int'l J. Computer Vision, vol. 10, no. 1, pp. 53-65, 1993
- Rasmussen, C., Hager, G.D., “Probabilistic Data Association Methods for Tracking Complex Visual Objects,” PAMI(23), No. 6, June 2001, pp. 560-576

Thank you!

- I hope the topics covered will prove useful for your future research

Massimo Piccardi
iNEXT Research Centre, FEIT,
University of Technology, Sydney
massimo@it.uts.edu.au