Computer Vision 2
WS 2018/19

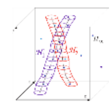
Part 10 – Multi-Object Tracking
27.11.2018

Prof. Dr. Bastian Leibe

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

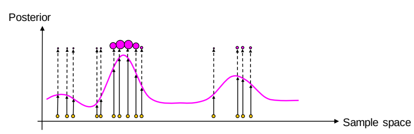
Course Outline

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
  - Introduction
  - MHT, JPDAF
  - Network Flow Optimization
- Visual Odometry
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis



Recap: Particle Filtering

- Many variations, one general concept:
  - *Represent the posterior pdf by a set of randomly chosen weighted samples (particles)*



  - Randomly Chosen = Monte Carlo (MC)
  - As the number of samples become very large – the characterization becomes an equivalent representation of the true pdf.

Recap: Sequential Importance Sampling

function $\left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\quad \mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — Sample from proposal pdf

$\quad w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — Update weights

$\quad \eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$\quad w_t^i = w_t^i / \eta$ — Normalize weights

end

Recap: Sequential Importance Sampling

function $\left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\quad \mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — Sample from proposal pdf

$\quad w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — Update weights

$\quad \eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$\quad w_t^i = w_t^i / \eta$ — Normalize weights

end

For a concrete algorithm, we need to define the importance density $q(.|.)$!

Recap: SIS Algorithm with Transitional Prior

function $\left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\quad \mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$ — Sample from proposal pdf

$\quad w_t^i = w_{t-1}^i p(\mathbf{y}_t|\mathbf{x}_t^i)$ — Update weights

$\quad \eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$\quad w_t^i = w_t^i / \eta$ — Normalize weights

end

Transitional prior
$q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$

Recap: Resampling

- Degeneracy problem with SIS
  - After a few iterations, most particles have negligible weights.
  - Large computational effort for updating particles with very small contribution to $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.
- Idea: Resampling
  - Eliminate particles with low importance weights and increase the number of particles with high importance weight.

$$\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N \rightarrow \left\{\mathbf{x}_t^{i*}, \frac{1}{N}\right\}_{i=1}^N$$

  - The new set is generated by sampling with replacement from the discrete representation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ such that

$$Pr\left\{\mathbf{x}_t^{i*} = \mathbf{x}_t^j\right\} = w_t^j$$

Slide adapted from Michael Rubinstein

Recap: Efficient Resampling Approach

- From Arulampalam paper:

```
Algorithm 2: Resampling Algorithm
[{x_k^{j*}, w_k^j, i^j}_{j=1}^{N_s}] = RESAMPLE [{x_k^i, w_k^i}_{i=1}^{N_s}]
• Initialize the CDF: c_1 = 0
• FOR i = 2: N_s
  – Construct CDF: c_i = c_{i-1} + w_k^i
• END FOR
• Start at the bottom of the CDF: i = 1
• Draw a starting point: u_1 ~ U[0, N_s^{-1}]
• FOR j = 1: N_s
  – Move along the CDF: u_j = u_1 + N_s^{-1}(j - 1)
  – WHILE u_j > c_i
  * i = i + 1
  – END WHILE
  – Assign sample: x_k^{j*} = x_k^i
  – Assign weight: w_k^j = N_s^{-1}
  – Assign parent: i^j = i
• END FOR
```

Basic idea: choose one initial small random number; deterministically sample the rest by "crawling" up the cdf. This is $\mathcal{O}(N)$!

Slide adapted from Robert Collins

Recap: Generic Particle Filter

$$\textbf{function}\ \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = PF\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$$\textit{Apply SIS filtering}\quad \left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$$

$\textit{Calculate } N_{eff}$

if $N_{eff} < N_{thr}$

$$\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = RESAMPLE\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right]$$

end

- We can also apply resampling selectively
  - Only resample when it is needed, i.e., N_{eff} is too low.
  - ⇒ Avoids drift when the tracked state is stationary.

Slide adapted from Michael Rubinstein

Recap: Sampling-Importance-Resampling (SIR)

function $[\mathcal{X}_t] = SIR\,[\mathcal{X}_{t-1}, \mathbf{y}_t]$

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ — Initialize

for $i = 1{:}N$

$\quad Sample\ \mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$ — Generate new samples

$\quad w_t^i = p(\mathbf{y}_t|\mathbf{x}_t^i)$ — Update weights

end

for $i = 1{:}N$

$\quad Draw\ i\ with\ probability \propto w_t^i$

$\quad Add\ \mathbf{x}_t^i\ to\ \mathcal{X}_t$ — Resample

end

Slide adapted from Michael Rubinstein

Recap: Sampling-Importance-Resampling (SIR)

function $[\mathcal{X}_t] = SIR\,[\mathcal{X}_{t-1}, \mathbf{y}_t]$

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

Important property:

for $i = 1{:}N$

$\quad Sample\ \mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$

$\quad w_t^i = p(\mathbf{y}_t|\mathbf{x}_t^i)$

Particles are distributed according to pdf from previous time step.

end

for $i = 1{:}N$

$\quad Draw\ i\ with\ probability \propto w_t^i$

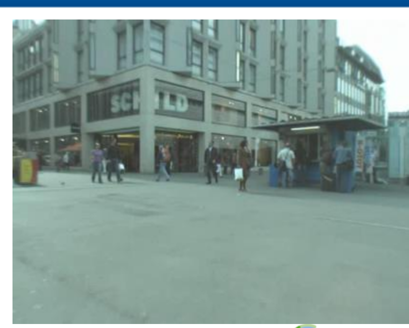
$\quad Add\ \mathbf{x}_t^i\ to\ \mathcal{X}_t$

Particles are distributed according to posterior from this time step.

end

Slide adapted from Michael Rubinstein

Today: Multi-Object Tracking

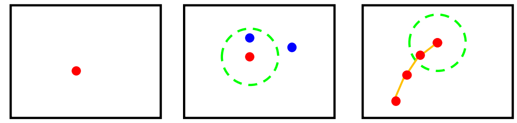


[Ess, Leibe, Schindler, Van Gool, CVPR'08; ICRA'09; PAMI'09]

Topics of This Lecture

- Multi-Object Tracking
  - Motivation
  - Ambiguities
- Simple Approaches
  - Gating
  - Mahalanobis distance
  - Nearest-Neighbor Filter
- Track-Splitting Filter
  - Derivation
  - Properties
- Outlook

Elements of Tracking

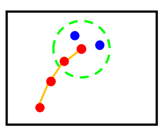


Detection | Data association | Prediction

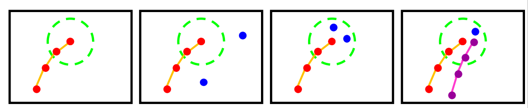
- Detection
  - *Where are candidate objects?* Lectures 2-6
- Data association
  - *Which detection corresponds to which object?* Today's topic
- Prediction
  - *Where will the tracked object be in the next time step?* Lectures 7-9

Motion Correspondence

- Motion correspondence problem
  - Do two measurements at different times originate from the same object?
- Why is it hard?
  - First make predictions for the expected locations of the current set of objects
  - Match predictions to actual measurements
  - This is where ambiguities may arise...



Motion Correspondence Ambiguities



1. Predictions may not be supported by measurements
   - Have the objects ceased to exist, or are they simply occluded?
2. There may be unexpected measurements
   - Newly visible objects, or just noise?
3. More than one measurement may match a prediction
   - Which measurement is the correct one (what about the others)?
4. A measurement may match to multiple predictions
   - Which object shall the measurement be assigned to?

Topics of This Lecture

- Multi-Object Tracking
  - Motivation
  - Ambiguities
- Simple Approaches
  - Gating
  - Mahalanobis distance
  - Nearest-Neighbor Filter
- Track-Splitting Filter
  - Derivation
  - Properties
- Outlook

Let's Formalize This

- Multi-Object Tracking problem
  - We represent a track by a state vector $\mathbf{x}$, e.g.,

$$\mathbf{x} = [x, y, v_x, v_y]^T$$

  - As the track evolves, we denote its state by the time index k:

$$\mathbf{x}^{(k)} = \left[x^{(k)}, y^{(k)}, v_x^{(k)}, v_y^{(k)}\right]^T$$

  - At each time step, we get a set of observations (measurements)

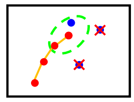
$$\mathbf{Y}^{(k)} = \left\{\mathbf{y}_1^{(k)}, \dots, \mathbf{y}_{M_k}^{(k)}\right\}$$

  - We now need to make the data association between tracks $\left\{\mathbf{x}_1^{(k)}, \dots, \mathbf{x}_{N_k}^{(k)}\right\}$ and observations $\left\{\mathbf{y}_1^{(k)}, \dots, \mathbf{y}_{M_k}^{(k)}\right\}$:

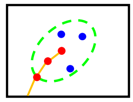
$z_l^{(k)} = j$ iff $\mathbf{y}_j^{(k)}$ is associated with $\mathbf{x}_l^{(k)}$

Reducing Ambiguities: Simple Approaches

- Gating
  - Only consider measurements within a certain area around the predicted location.
  - ⇒ Large gain in efficiency, since only a small region needs to be searched



- Nearest-Neighbor Filter
  - Among the candidates in the gating region, only take the one closest to the prediction $\mathbf{x}_p$



$$z_l^{(k)} = \arg\min_j (\mathbf{x}_{p,l}^{(k)} - \mathbf{y}_j^{(k)})^T (\mathbf{x}_{p,l}^{(k)} - \mathbf{y}_j^{(k)})$$

  - Better: the one most likely under a Gaussian prediction model

$$z_l^{(k)} = \arg\max_j \mathcal{N}(\mathbf{y}_j^{(k)}; \mathbf{x}_{p,l}^{(k)}, \mathbf{\Sigma}_{p,l}^{(k)})$$

which is equivalent to taking the Mahalanobis distance

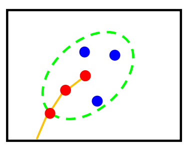
$$z_l = \arg\min_j (\mathbf{x}_{p,l} - \mathbf{y}_j)^T \mathbf{\Sigma}_{p,l}^{-1} (\mathbf{x}_{p,l} - \mathbf{y}_j)$$

Gating with Mahalanobis Distance

- Recall: Kalman filter
  - Provides exactly the quantities necessary to perform this
  - Predicted mean location $\mathbf{x}_p$
  - Prediction covariance Σ_p
  - The Kalman filter prediction covariance also defines a useful gating area.
  - ⇒ E.g., choose the gating area size such that 95% of the probability mass is covered.
- Side note
  - The Mahalanobis distance is χ^2 distributed with the number of degrees of freedom n_z equal to the dimension of $\mathbf{x}$.
  - For a given probability bound, the corresponding threshold on the Mahalanobis distance can be obtained from χ^2 distribution tables.

Mahalanobis Distance

- Additional notation
  - Our KF state of track $\mathbf{x}_l$ is given by the prediction $\hat{\mathbf{x}}_l^{(k)}$ and covariance $\mathbf{\Sigma}_{p,l}^{(k)}$.
  - We define the innovation that measurement $\mathbf{y}_j$ brings to track $\mathbf{x}_l$ at time k as



$$\mathbf{v}_{j,l}^{(k)} = (\mathbf{y}_j^{(k)} - \mathbf{x}_{p,l}^{(k)})$$

  - With this, we can write the observation likelihood shortly as

$$p(\mathbf{y}_j^{(k)}|\mathbf{x}_l^{(k)}) \sim \exp\left\{-\frac{1}{2}\mathbf{v}_{j,l}^{(k)^T}\mathbf{\Sigma}_{p,l}^{(k)^{-1}}\mathbf{v}_{j,l}^{(k)}\right\}$$

  - We define the ellipsoidal gating or validation volume as

$$V^{(k)}(\gamma) = \left\{\mathbf{y} | (\mathbf{y}-\mathbf{x}_{p,l}^{(k)})^T \mathbf{\Sigma}_{p,l}^{(k)^{-1}} (\mathbf{y}-\mathbf{x}_{p,l}^{(k)}) \le \gamma\right\}$$

Problems with NN Assignment

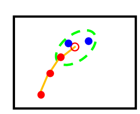
- Limitations
  - For NN assignments, there is always a finite chance that the association is incorrect, which can lead to serious effects.
  - ⇒ If a Kalman filter is used, a misassigned measurement may lead the filter to lose track of its target.
  - The NN filter makes assignment decisions only based on the current frame.
  - More information is available by examining subsequent images.
  - ⇒ Let's make use of this information by postponing the decision process until a future frame will resolve the ambiguity...

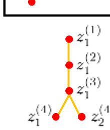
Topics of This Lecture

- Multi-Object Tracking
  - Motivation
  - Ambiguities
- Simple Approaches
  - Gating
  - Mahalanobis distance
  - Nearest-Neighbor Filter
- Track-Splitting Filter
  - Derivation
  - Properties
- Outlook

Track-Splitting Filter

- Idea
  - Problem with NN filter was hard assignment.
  - Rather than arbitrarily assigning the closest measurement, form a tree.
  - Branches denote alternate assignments.
  - No assignment decision is made at this stage!
  - ⇒ Decisions are postponed until additional measurements have been gathered...

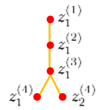




- Potential problems?
  - Track trees can quickly become very large due to combinatorial explosion.
  - ⇒ We need some measure of the likelihood of a track, so that we can prune the tree!

Track Likelihoods

- Expressing track likelihoods
  - Given a track l, denote by $\theta_{k,l}$ the event that the sequence of assignments
$$Z_{k,l} = \left\{ z_{i_1,l}^{(1)}, \ldots, z_{i_k,l}^{(k)} \right\}$$
    from time 1 to k originate from the same object.



  - The likelihood of $\theta_{k,l}$ is the joint probability over all observations in the track
$$L(\theta_{k,l}) = \prod_{j=1}^{k} p(z_{i_j,l}^{(j)} | Z_{(j-1),l}, \theta_{k,l})$$
  - If we assume Gaussian observation likelihoods, this becomes
$$L(\theta_{k,l}) = \prod_{j=1}^{k} \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_l^{(j)}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2} \sum_{j=1}^{k} \mathbf{v}_{i_j,l}^{(j)^T} \Sigma_l^{(j)^{-1}} \mathbf{v}_{i_j,l}^{(j)} \right]$$

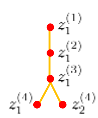
Track Likelihoods (2)

- Starting from the likelihood
$$L(\theta_{k,l}) = \prod_{j=1}^{k} \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_l^{(j)}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2} \sum_{j=1}^{k} \mathbf{v}_{i_j,l}^{(j)^T} \Sigma_l^{(j)^{-1}} \mathbf{v}_{i_j,l}^{(j)} \right]$$
  - Define the modified log-likelihood λ_l for track l as
$$\begin{aligned} \lambda_l(k) &= -2 \log \left[\frac{L(\theta_{k,l})}{\prod_{j=1}^{k} (2\pi)^{-\frac{d}{2}} |\Sigma_l^{(j)}|^{-\frac{1}{2}}} \right] \\ &= \sum_{j=1}^{k} \mathbf{v}_{i_j,l}^{(j)^T} \Sigma_l^{(j)^{-1}} \mathbf{v}_{i_j,l}^{(j)} \\ &= \lambda_l(k-1) + \mathbf{v}_{i_k,l}^{(k)^T} \Sigma_l^{(k)^{-1}} \mathbf{v}_{i_k,l}^{(k)} \end{aligned}$$

$\Rightarrow$ Recursive calculation, sum of Mahalanobis distances of all the measurements assigned to track l.

Track-Splitting Filter

- Effect
  - Instead of assigning the measurement that is currently closest, as in the NN algorithm, we can select the *sequence* of measurements that minimizes the *total* Mahalanobis distance over some interval!



  - Modified log-likelihood provides the merit of a particular node in the track tree.
  - Cost of calculating this is low, since most terms are needed anyway for the Kalman filter.
- Problem
  - The track tree grows exponentially, may generate a very large number of possible tracks that need to be maintained.

Pruning Strategies

- In order to keep this feasible, need to apply pruning
  - Deleting unlikely tracks
    - May be accomplished by comparing the modified log-likelihood $\lambda(k)$, which has a χ^2 distribution with kn_z degrees of freedom, with a threshold α (set according to χ^2 distribution tables).
    - Problem for long tracks: modified log-likelihood gets dominated by old terms and responds very slowly to new ones.

    $\Rightarrow$ Use sliding window or exponential decay term.
  - Merging track nodes
    - If the state estimates of two track nodes are similar, merge them.
    - E.g., if both tracks validate identical subsequent measurements.
  - Only keeping the most likely N tracks
    - Rank tracks based on their modified log-likelihood.

Summary: Track-Splitting Filter

- Properties
  - Very old algorithm
    - P. Smith, G. Buechler, A Branching Algorithm for Discriminating and Tracking Multiple Objects, IEEE Trans. Automatic Control, Vol. 20, pp. 101-104, 1975.
  - Improvement over NN assignment.
  - Assignment decisions are delayed until more information is available.
- Many problems remain
  - Exponential complexity, heuristic pruning needed.
  - Merging of track nodes is necessary, because tracks may share measurements, which is physically unrealistic.

$\Rightarrow$ Would need to add exclusion constraints such that each measurement may only belong to a single track.

$\Rightarrow$ Impossible in this framework...

Topics of This Lecture

- Multi-Object Tracking
  - Motivation
  - Ambiguities
- Simple Approaches
  - Gating
  - Mahalanobis distance
  - Nearest-Neighbor Filter
- Track-Splitting Filter
  - Derivation
  - Properties
- Outlook

Outlook for the Next Lectures

- More powerful approaches
  - Multi-Hypothesis Tracking (MHT)
    - Well-suited for KF, EKF approaches [Reid, 1979]
  - Joint Probabilistic Data Association Filters (JPDAF)
    - Well-suited for Particle Filter based approaches [Fortmann, 1983]
- Data association as convex optimization problem
  - Bipartite Graph Matching (Hungarian algorithm)
  - Network Flow Optimization

  ⇒ Efficient, globally optimal solutions for subclass of problems.

References and Further Reading

- A good tutorial on Data Association
  - I.J. Cox. A Review of Statistical Data Association Techniques for Motion Correspondence. In *International Journal of Computer Vision*, Vol. 10(1), pp. 53-66, 1993.