Particle Filters

Bryan Minor

November 1, 2011

Outline

- Introduction: why particle filters?
- Particle Filter Tutorial
  - Basics
  - Strengths/Weaknesses
- Current Uses of Particle Filters
- Tracking Using a Detector Particle Filter
  - Analysis
  - Discussion

Why Particle Filters?

- Useful tools for a variety of situations
  - Tracking
  - Image processing
  - Smart environments
  - Etc.
- Allow analysis of complex systems
  - Non-linear
  - Non-Gaussian

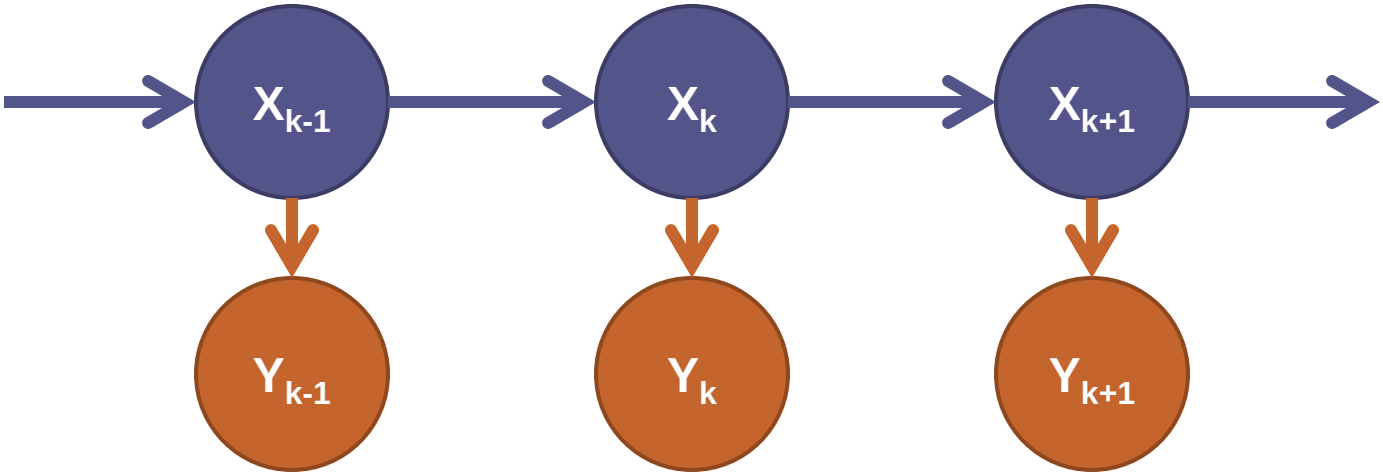
Sources: [1], [2], [9]

Outline

- Introduction: why particle filters?
- Particle Filter Tutorial
  - Basics
  - Strengths/Weaknesses
- Current Uses of Particle Filters
- Tracking Using a Detector Particle Filter
  - Analysis
  - Discussion

Tutorial - HMMs

- Consider hidden Markov model (HMM)
  - Can observe outputs Y_i
  - Cannot observe states X_i

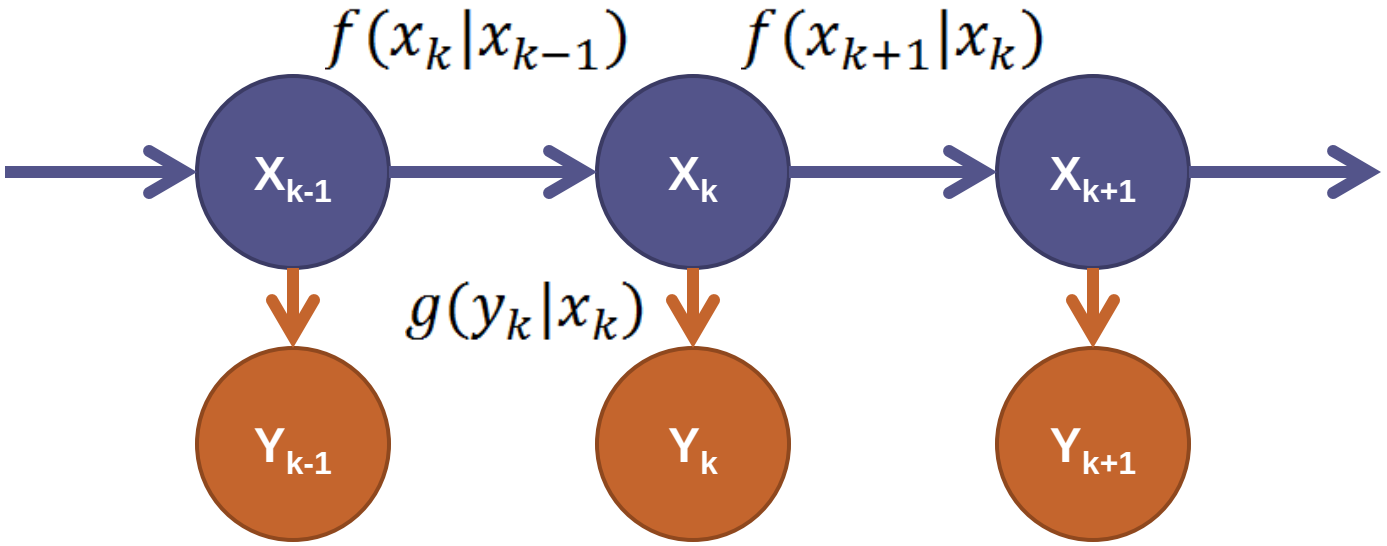


Tutorial Source: [2]

Tutorial - HMMs

- Simple model of:
  - State equation (transition):
    $X_1 \sim \mu(x_1)$ and $f(x_k|x_{k-1})$ (for k > 1)
  - Observation equation:
    $g(y_k|x_k)$
  - Assume homogeneous case
    - Probability of transitions and observations independent of time

Tutorial - HMMs



Tutorial - HMMs

- Goal: estimate the state x_n, given all of the observations up to that point ($y_{1:n}$)
- Alternatively, want to find the posterior distribution:

$$p(x_{1:n}|y_{1:n})$$

Tutorial - Bayesian Inference

- Using Bayes:

$$p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})}$$

- The joint probability can be written

$$p(x_{1:n}, y_{1:n}) =$$
$$p(x_{1:n-1}, y_{1:n-1}) f(x_n|x_{n-1}) g(y_n|x_n)$$

Tutorial - Bayesian Inference

- Thus:

$$p(x_{1:n}|y_{1:n}) = p(x_{1:n-1}, y_{1:n-1}) \frac{f(x_n|x_{n-1})g(y_n|x_n)}{p(y_n|y_{1:n-1})}$$

Tutorial - Bayesian Inference

- Ultimately we end up with two steps:
  - Update step:

$$p(x_n|y_{1:n}) = \frac{g(y_n|x_n)p(x_n|y_{1:n-1})}{p(y_n|y_{1:n-1})}$$

  - Prediction step:

$$p(x_n|y_{1:n-1}) = \int f(x_n|x_{n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1}$$

Tutorial - Bayesian Inference

- Problem:
  - Often, these distributions are intractable in closed-form
  - This is especially the case in non-linear and non-Gaussian models

Tutorial - Sequential Monte Carlo Methods (SMC Methods)

- Solution:
  - Approximate the distributions using a large number of samples (particles)
  - As the number of particles N increases toward ∞, this converges to the actual distribution

Tutorial - SMC Methods

- Sample sequentially from sequence of probability densities $\{\pi_n(x_{1:n})\}$
- We sample N independent random variables $X_{1:n}^i$ from each probability distribution and estimate the distribution as

$$\hat{\pi}_n(x_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X_{1:n}^i}(x_{1:n})$$

where $\delta_{X_{1:n}^i}(x_{1:n})$ is the delta at each sample

Tutorial - SMC Methods

- Two Problems:
  1. It is difficult to sample from complex distributions
  2. Sampling is computationally complex (at least linear with n)

Tutorial - SMC Methods

- Solution 1: Importance Sampling
  - Use an “importance density” and weighting to model density

$$\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n}$$

$$\pi_n(x_{1:n}) = \frac{w_n(x_{1:n})q_n(1_{1:n})}{Z_n}$$

where $w_n(x_{1:n})$ are weights, $q_n(1_{1:n})$ is the importance density, and Z_n a normalization factor

Tutorial - SMC Methods

- The density can then be estimated as

$$\hat{\pi}_n(x_{1:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_{1:n}^i}(x_{1:n})$$

where $W_n^i = \frac{w_n(X_{1:n}^i)}{\sum_{j=1}^{N} w_n(X_{1:n}^j)}$

- Note: we want to pick q so that variance is minimized – pick something close to π

Tutorial - SMC Methods

- Solution 2: Sequential Importance Sampling
  - Select importance distribution so that $q_n(x_{1:n}) = q_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})$
  - At first time step, sample from original distribution initial probability $q_1(x_1)$
  - For subsequent steps, pick from the conditional probabilities $q_k(x_k|X^i_{1:k-1})$
  - The variance of these estimations often increases with n, so further refinement is needed

Tutorial - SMC Methods

- Resampling
  - To reduce variance, sample again from the newly created approximation distributions
  - Each particle is associated with a number of “offspring” samples to estimate the estimate distribution

Tutorial - SMC Algorithm

- At time n = 1:
  1. Sample $X_1^i \sim q_1(x_1)$
  2. Compute weights $w_1(X_1^i)$ and W_1^i
  3. Resample $\{W_1^i, X_1^i\}$ to obtain N particles $\{\frac{1}{N}, \bar{X}_1^i\}$
- At time n ≥ 2:
  1. Sample $X_n^i \sim q_n(x_n | \bar{X}_{1:n-1}^i)$ and set $X_{1:n}^i \leftarrow (\bar{X}_{1:n-1}^i, X_n^i)$
  2. Compute weights $\alpha_n(X_{1:n}^i)$ and W_n^i
  3. If needed, resample $\{W_n^i, X_{1:n}^i\}$ to obtain $\{\frac{1}{N}, \bar{X}_{1:n}^i\}$

Tutorial - Particle Filters

- Let the distribution to be modeled be
  $\pi_n(x_{1:n}) = p(x_{1:n}|y_{1:n})$
  we can find a importance density
  $q_n(x_n|x_{1:n-1}) = q(x_n|y_n, x_{n-1})$
  and weights $\alpha_n(x_{1:n}) = p(y_n|x_{n-1})$
- This allows us to create a particle filter algorithm for estimating the posterior distribution
- Note that q need only depend on the previous state and current observation

Tutorial - Particle Filter Algorithm

- At n = 1:
  1. Sample $X_1^i \sim q(x_1|y_1)$
  2. Compute weights

$$w_1(X_1^i) = \frac{\mu(X_1^i)g(y_1|X_1^i)}{q(X_1^i|y_1)} \qquad W_1^i$$

  3. Resample $\{W_1^i, X_1^i\}$ to obtain N equal-weight particles $\{\frac{1}{N}, \bar{X}_1^i\}$

Tutorial - Particle Filter Algorithm

- For n ≥ 2:
  1. Sample $X_n^i \sim q\left(x_n | y_n, \bar{X}_{1:n-1}^i\right)$ and set $X_{1:n}^i \leftarrow (\bar{X}_{1:n-1}^i, X_n^i)$
  2. Compute weights $\alpha_n\left(X_{n-1:n}^i\right) = \frac{g\left(y_n | X_n^i\right) f\left(X_n^i | X_{n-1}^i\right)}{q\left(X_n^i | y_n, X_{n-1}^i\right)} \qquad W_n^i$
  3. Resample $\{W_n^i, X_{1:n}^i\}$ to find N equal-weight particles $\{\frac{1}{N}, \bar{X}_{1:n}^i\}$

Tutorial - Particle Filter Algorithm

- At each time step, we obtain estimate distributions:

$$\hat{p}(x_{1:n}|y_{1:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_{1:n}^i}(x_{1:n})$$

$$\hat{p}(y_n|y_{1:n-1}) = \sum_{i=1}^{N} W_{n-1}^i \alpha_n(X_{n-1:n}^i)$$

Tutorial - Particle Filters

- Problems:
  - Even if optimal importance distribution is used, the model may not be efficient
  - Only most recent particles are sampled at time n

Tutorial - Particle Filters

- This basic algorithm can be enhanced:
  - Resample before computing weights (from independence) – auxiliary particle filtering
  - Resample-move algorithm using Markov Chain Monte Carlo (MCMC)
    - Introduces diversity, but same number of resampling steps
  - Block sampling resamples previous components in blocks

Outline

- Introduction: why particle filters?
- Particle Filter Tutorial
  - Basics
  - Strengths/Weaknesses
- Current Uses of Particle Filters
- Tracking Using a Detector Particle Filter
  - Analysis
  - Discussion

Current Particle Filter Uses

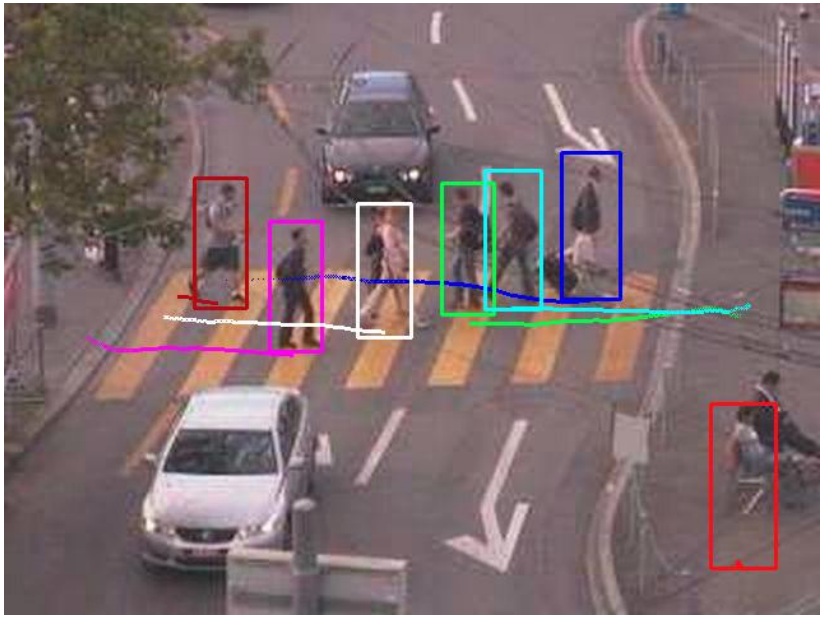
- Localizing acoustic signals [3]
- Geosciences data assimilation [4]
- Video surveillance [5]
- Automated vehicles [6]
- Tracking
  - In smart environments [7], [8]
  - Visual people tracking [9]

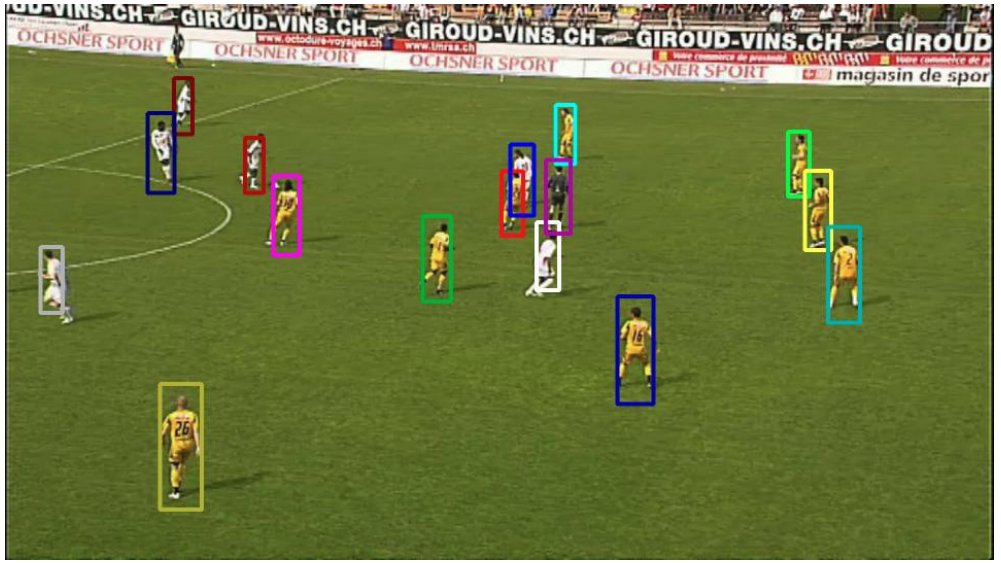
Outline

- Introduction: why particle filters?
- Particle Filter Tutorial
  - Basics
  - Strengths/Weaknesses
- Current Uses of Particle Filters
- Tracking Using a Detector Particle Filter
  - Analysis
  - Discussion

Paper Review

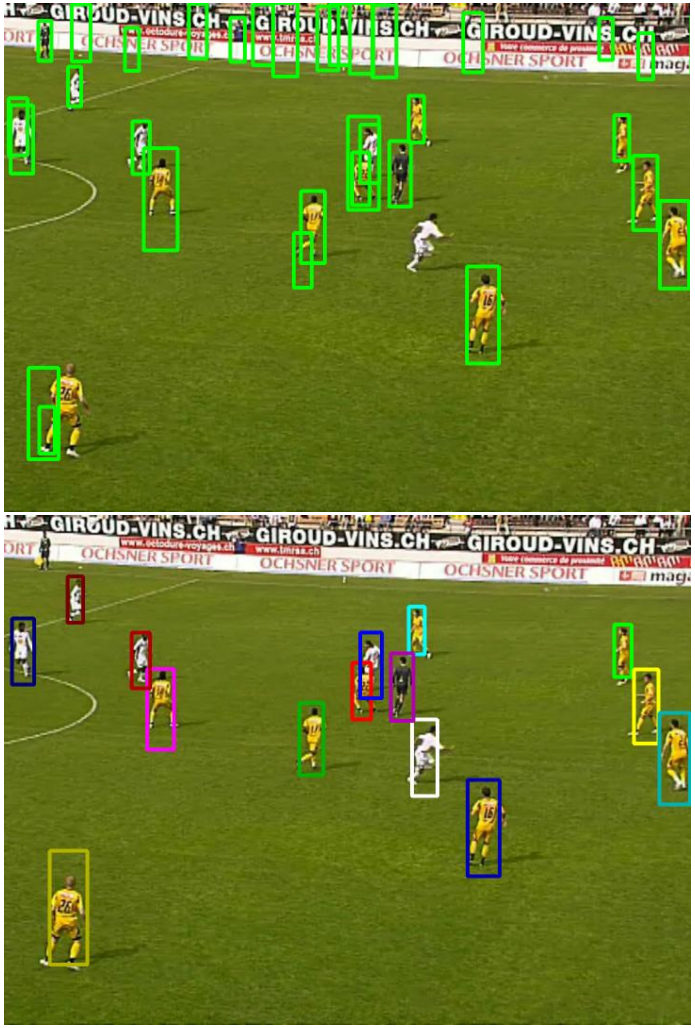
- “Robust Tracking-by-Detection using a Detector Confidence Particle Filter” [9]





Motivation

- Current approaches using detection algorithms are sparse and often erroneous (false positives/negatives)
- Many attempts rely on future data, preventing real-time analysis
  - Online applications require the Markovian properties, and thus can use a particle filter
- Most approaches rely on detector output, without any confidence information



Normal detector output (top) misses some objects, the authors' algorithm (bottom) is more accurate.

Contribution

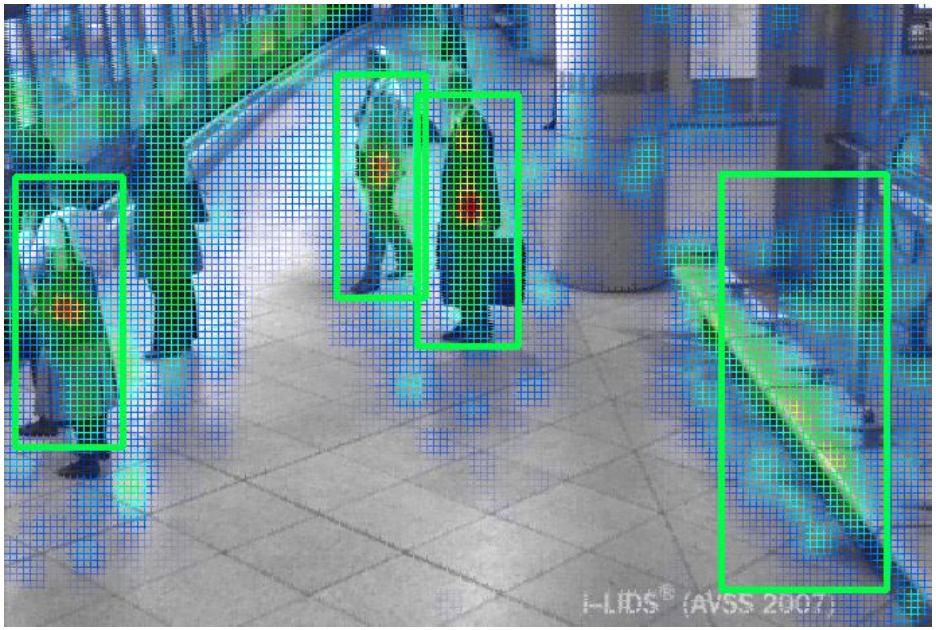
- Provide online tracker algorithm
- Utilize person-specific classifiers for each detected person
  - Particle filters
- Utilize confidence of the detection algorithms

Detection Confidence

- Detection algorithms create a confidence density
  - Sliding-window approaches (e.g. HOG) use sampling from 3D grid
  - Feature-based approaches (e.g. ISM) use a vote of matching features
  - Both approaches find local maxima and suppress non-maxima
- Most simply output a high-confidence detection, but lose the information found in densities

Detection Confidence

- Authors' algorithm utilizes confidence densities from the intermediate steps



Detector confidence density for ISM; green boxes show high-confidence detections.

Particle Filter Trackers

- Separate particle filter tracker initialized for each detection
- State $\boldsymbol{x} = \{x, y, u, v\}$ maps 2D position (x,y) and velocity (u,v)
- Utilize a bootstrap filter – importance distribution is the state transition density:

$$(x, y)_t = (x, y)_{t-1} + (u, v)_{t-1}\Delta t + \varepsilon_{(x,y)}$$

$$(u, v)_t = (u, v)_{t-1} + \varepsilon_{(u,v)}$$

Particle Filter Trackers

- Particle filter weights for each particle *i* are formed as
  $w_t^{(i)} \propto p(y_t|\boldsymbol{x}_t^{(i)})$
  since the previous weights are resampled and normalized at each step
- Trackers initialized for overlapping detections in subsequent frames
  - Only for objects appearing at the edge
- Position and size are estimated from the particles and iterative detections

Data Association

- Each tracker and detection pair is scored using:

$$s(tr,d) = g(tr,d)\left(c_{tr}(d) + \alpha \sum_{p \in tr}^{N} p_{\mathcal{N}(d-p)}\right)$$

  tr is the tracker of interest, d is the detection, p are the particles, and $p_{\mathcal{N}(d-p)}$ normal based on the distance from d to p
- $g(tr,d)$ is a gating function that assesses the direction the person may move
- $c_{tr}(d)$ is a classifier trained for the tracker

Data Association

- The highest-scored tracker and detection pair is associated and removed from the grid, along with its corresponding tracker and detector rows/columns
- This is repeated until all scores fall below a threshold
  - Some trackers may not be paired to a detector

Weight Calculation

- Particle weights are calculated using three criteria:
  - Detection term: strong if a detection is associated with the tracker, weak otherwise
  - Detector confidence term: uses intermediate detector confidence (verified by comparison with existing detections)
  - Classifier term: uses output of classifiers (color and texture) associated with particle

Experimental Results

- Algorithm used for a variety of image sequences
- Authors determined that red-green-intensity and local binary pattern features provide the best accuracy/complexity ratio
- CLEAR MOT metrics used to evaluate
  - Scored on precision and accuracy (misidentifications and ID switches)

Experimental Results

- Accuracy ranges around 73% - 86%, with approximately 65% precision
- Most errors caused by close proximity of other people and obscured views
- Authors compare to original algorithms for the sequences
  - State that their algorithm performs better
  - Generally appears to be the case based on table

Dataset	Prec.	Accur.	F. Neg.	F. Pos.	ID Sw.
ETH Centr.	70.0%	**72.9%**	26.8%	0.3%	0
ETH Centr. [16]	66.0%	33.8%	51.3%	14.7%	5
UBC Hockey	57.0%	**76.5%**	22.3%	1.2%	0
UBC Hockey [20]	51.0%	67.8%	31.3%	0.0%	11
i-LIDS easy	67.0%	**78.1%**	16.4%	5.3%	18
i-LIDS med*	66.0%	**76.0%**	22.0%	2.0%	2
i-LIDS [12]	-	68.4%	29.0%	13.7%	-
i-LIDS [26]	-	55.3%	37.0%	22.8%	-
TUD Cross.	71.0%	**84.3%**	14.1%	1.4%	2
Soccer	67.0%	**85.7%**	7.9%	6.2%	4

Table 1: CLEAR MOT results on 5 datasets demonstrate the performance of our algorithm compared to state-of-the-art methods.

Conclusions

- Algorithm helps to improve accuracy and complexity by using
  - Particle filter models
  - Detection confidence densities
  - Tracker and classifier associations
- Results seem strong
- Future work should look at using information about the scene in the model

Discussion

- Strengths:
  - Markov model reduces complexity of calculations
  - Relatively high accuracy
  - Tested on a wide variety of datasets
  - Good description of methods

Discussion

- Weaknesses:
  - No analysis of computational complexity
  - No theoretical analysis or discussion of parameter determination in model
  - Slow frame rate (at 0.4-2 fps)
  - No statistical tests performed for thorough comparison to other methods

Discussion

- Questions:
  - What other state-of-the-art methods are used?
    - Fuzzy logic
    - Auxiliary particle filters
    - Iterated likelihood weighting filter
  - What other applications can this be applied to?
    - Found some examples:
      - Power amplifiers
      - Medical diagnosis
      - Neural systems

Discussion

- Questions:
  - Would this algorithm be improved by incorporating 3D or scene information?
  - Is this work only an extension of specific knowledge to others' methods? Is it really a new approach?
  - What if scenes have more people in them?
  - How does data quality affect performance?
  - Could this model be extended to track non-human objects?

References

[1] M. Bolic, “Theory and Implementation of Particle Filters,” University of Ottawa, Nov. 2004

[2] M. Sanjeev Arulampalam, S. Maskell, and N. Gordon, “A Tutorial on Particle Filtering and Smoothing: Fifteen years later,” Version 1.1, Dec. 2008

[3] F. Xavier, G. Matz, P. Gerstoft, and C. Mecklenbräuker, “Localization of acoustic sources using a decnetralized particle filter,” *EURASIP Journal on Wireless Coummunications and Networking 2011*, Sept. 2011

[4] P. J. van Leeuwen, “Nonlinear data assimilation in geosciences: an extremely efficient particle filter,” *Quarterly Journal of the Royal Meteorological Society*, vol. 136, issue 653, pp. 1991-1999, Oct. 2010

References

[5] V. Thomas and A. K. Ray, "Fuzzy Particle Filter for Video Surveillance," *Fuzzy Systems, IEEE Transactions on* , vol.19, no.5, pp.937-945, Oct. 2011

[6] H. Loose, U. Franke, and C. Stiller, “Kalman Particle Filter for lane recognition on rural roads,” *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 60-65, June 2009

[7] A. Crandal, “Behaviometrics for Multiple Residents in a Smart Environment”, Ph.D. dissertation, Dept. Elect. Eng. And Comp. Sci., Washington State Univ., Pullman, WA, 2011

[8] K. Bernardin, T. Gehrig, and R. Stiefelhagen, “Multi-level Particle Filter Fusion of Features and Cues for Audio-Visual Person Tracking,” *Lecture Notes on Computer Science*, vol. 4625, pp. 70-81, 2008

References

[9] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, “Robust Tracking-by-Detection using a Detector Confidence Particle Filter,” *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1515-1522, Oct. 2009