Robust Object Tracking Using Unreliable Object Recognizers

Li Li
Shannon Cognitive Computing Lab
Huawei
Bridgewater, New Jersey, USA
email: li.nj.li@huawei.com

Masood Mortazavi
Shannon Cognitive Computing Lab
Huawei
Santa Clara, California, USA
email: masood.mortazavi@huawei.com

***Abstract*—This paper presents a method for video surveillance systems to correct noisy observations from distributed object recognizers that are unreliable. An unreliable recognizer consists of a hardware sensor (e.g., a camera) and a recognition program (e.g., facial recognition) that may produce random errors including false positives, false negatives, or failures. To address this issue, we use a Bayesian Network (BN) to connect multiple factors that can cause the noisy observations with random errors. We then incorporate the BN into an extended Hidden Markov Model (HMM) to infer optimal object paths from noisy observations. A prototype system is implemented and the simulation tests show that the Forward-Backward algorithm can achieve 77.3% accuracy on average with 47.9% relative improvement over the Viterbi algorithm. Both algorithms are robust to increasing noises and errors in the observations, even when 100% observations have over 66% errors.**

Keywords-video surveillance; object tracking; camera network; baysian network; hidden markov model; forward-backward algorithm; Viterbi algorithm.

I. INTRODUCTION

A video surveillance system consists of distributed cameras that cover the critical areas or even an entire city. These cameras are connected by wireless and wired networks to a regional control center, where the video streams are processed, stored, analyzed and searched by various algorithms to assist the authorities in controlling and preventing hazardous and criminal activities.

A main challenge in video surveillance systems is to automatically track moving objects, such as a person or a vehicle. Traditional approaches focus on monolithic algorithms that integrate three distinct functions: 1) extract features from video frames; 2) recognize the objects across video frames based on the features; and 3) determine object trajectories based on object recognitions and extracted features. While the monolithic approaches allow a tracking system to jointly optimize these functions, they make it difficult to combine different object recognition algorithms and tracking algorithms on the market. To achieve such flexibility, we propose to divide a tracking system into two layers: object recognition and object tracking that can change independently as blackboxes through a well-defined interface. A layered tracking system can easily include additional sensors and recognizers, such as radar, sonar, LIDAR, Infrared, etc. without changing the tracking model. On the other hand, we can easily port a tracking model to different domains with different sensors and recognizers.

Facial recognition can match an unknown human face in image or video to one of the known faces in a database (face identification), or determine if two human faces are the same (face verification). Despite 98% accuracy in some datasets, facial recognition in uncontrolled environment only achieves 74.7% classification accuracy [1]. Gait recognition can identify a person at a distance based on how he or she walks as recorded by camera or motion sensors. A recent survey [2] shows the gait recognition accuracies of different approaches vary from 60% to 90%.

Automatic License Plate Recognition can extract and recognize license plate from images and videos of vehicles. A 2013 survey [3] shows that the accuracies of different methods vary from 80% to 90%. Recent researches in computer vision [4][5] can recognize vehicle make and model, as well as distinct marks, with accuracy in the range of 70% to 84%.

Visual recognizers are unreliable due to constraints imposed by physical environment, such as variations in visibility, reflection, scale, view point, occlusion and motion. An unreliable object recognizer can produce noisy observations with 3 types of random errors: 1) *false positive*: when it recognizes an object that is actually not in its field of view; 3) *false negative:* when it does not recognize an object that is actually in its field of view; and 4) *failure*: when the camera breaks, the network fails or the recognition program crashes. When a tracking system receives an observation from an object recognizer, it should not completely trust the observation. Instead, the tracking system should identify and correct the random errors in the observation.

For this purpose, we introduce three probabilistic models: 1) competence model; 2) intention model; and 3) motion model, to describe the uncertain behaviors of the objects and recognizers. We then use a Bayesian Network (BN) to connect the factors in these models based on how they cause noisy observations. The BN is then incorporated into an extended Hidden Markov Model (HMM) that can correct the noisy observations using the well-known inference algorithms, such as the Forward-Backward and Viterbi algorithms. Simulated tests show that the approach can correct 77.3% noisy observations on average and is robust to increasing noises and errors in the observations, even when 100% observations have over 66.66% errors.

 ISBN: 978-1-61208-559-3

The rest of the paper is organized as follows. Section II reviews the related work. Section III describes the BN and extended HMM for object tracking. Section IV shows the experimental results and we conclude with Section V.

II. RELATED WORK

Taj et al. [6] surveys the two main types of distributed camera network architectures and well-known tracking methods, including Graph matching, HMM, Particle filtering and Kalman filters. The decentralized recognizers send their results to the fusion centers which combine the results into a global trajectory. In contrast, distributed recognizers act as peers that exchange and combine results from their neighbors. In both cases, recognizers in a camera network can form dynamic clusters that move along with the object, in order to reduce search space, save energy and mitigate video traffic.

Within such camera networks, various methods [7]-[10] have been proposed to track objects across different cameras. These methods tightly integrate feature extraction, object recognition and object tracking as each method develops its own set of features.

Our approach is motivated by Fleuret et al. [10] who describes a probabilistic generative model that combines a motion model, an appearance model and a color model to infer the locations of objects from video streams. The method can track up to six people across two to four synchronized video streams taken at eye level and from different angles. In addition, the model can derive accurate trajectories of each individual using a grid map that divides an area into cells, each of which can only be occupied by one person at a time. The motion model calculates the conditional probability that an individual at cell *a* will travel to cell *b* at time *k* based on the distance between the cells, while the appearance and color models estimate the likelihood that a video stream is observed when an individual is at a cell.

However, our approach differs from [10] in some significant ways. First, our approach decouples object recognition from object tracking whereas [10] integrates them tightly. Second, our approach runs parallel HMM inferences for multiple objects whereas [10] is sequential by inferring the most likely object trajectory first and uses it to constrain the next object trajectory. Third, our motion model considers travel modes whereas [10] does not. Fourth, our approach can correct noisy observations, whereas [10] does not.

III. TRACKING SYSTEM BASED ON BN AND HMM

The logic architecture of our tracking system is illustrated in **Figure 1**, where *N* recognizers receive video streams from distributed cameras, perform object recognitions and send the observations to the tracker. The tracker estimates object paths based on extended HMM.

The architecture is independent of camera resolutions and frame rates as well as video compression and transmission technologies. This allows the tracking system to reduce network bandwidth without losing tracking accuracy by matching the camera operations with the speed ranges of the target objects. The tracking system provides sufficient computing power to run the recognizers and trackers in real-time on the video streams.

Each recognizer *R* is treated as a blackbox function $R(C,O)=(t,S_t)$ that maps a video stream from camera *C* and an object *O* to observation S_t at time *t*. For convenience, $S_t=M>0$ denotes that *R* observes *O* at some mode *M*, $S_t=0$ denotes that *R* does not observe *O*, and $S_t=-1$ denotes that *R* fails. Travel mode *M* includes various means of mobility, such as walking, running, bicycle, car, boat, airplanes.

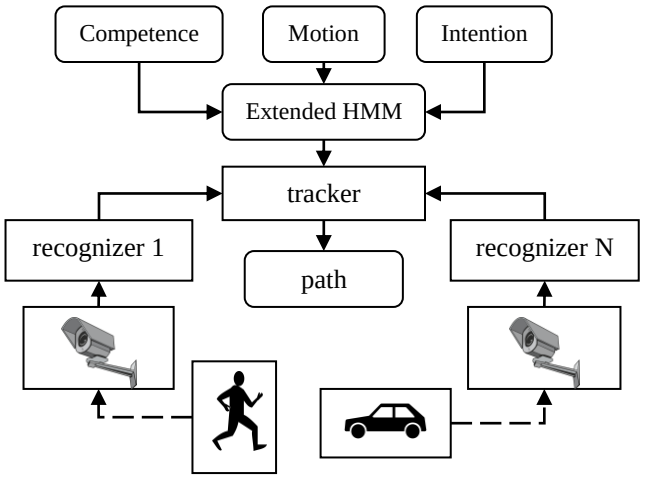

Figure 1. Logic architecture of tracking system

When object *O* moves within a camera network, the cameras it visits form an *object path*. When the recognizers are unreliable, the observed cameras may differ from the actual object path due to random errors:

- Type *0* false negative: $S_t=0$ but the correct values is $S_t>0$ since *O* was at *C*;
- Type *1* failure: $S_t=-1$ but the correct value is $S_t>0$ since *O* was at *C*;
- Type *2* false positive: $S_t>0$, but the correct value is $S_t=0$ since *O* was not at *C*.

To correct these random errors, we model the object path as the hidden states of an extended HMM that generates noisy observations according to some probabilistic distributions. The probabilistic distributions are derived from a BN that connects three probabilistic behavioral models: intention, motion and competence, to define the causes of noisy observations. With these probabilistic distributions, efficient inference algorithms are used to estimate the object paths as the optimal states of the HMM given the noisy observations.

A. Probablistic Behavioral Models

The **intention model** captures the uncertainty of object paths with three conditional probabilities. $P_I(C_i|O)$ is the probability that object *O* will enter the camera network at camera C_i. $P_I(C_j|C_i,O)$ is the probability that object *O* selects a place covered by camera C_j to visit from a place covered by camera C_i. $P_I(M|C_i,C_j,O)$ is the probability that object *O* will travel from camera C_i to C_j in transportation mode *M*. For simplicity, this model assumes that where an object

travels and what mode it uses only depend on the current and next places.

The **motion model** captures the uncertainty of travel time Δt in mode M from camera C_i to camera C_j with conditional probability $P_M(\Delta t|C_i,C_j,M)$. For simplicity, we assume that $P_M(\Delta t|C_i,C_j,M)$ is a Gaussian (Normal) distribution $N(\mu,\sigma)$ with mean μ and standard deviation σ. The motion model is independent of objects and abstracts complex indoor and outdoor physical environments as a 3 dimensional matrix.

The **competence model** captures the uncertainty of recognizers with three probability distributions:

- $true\_pos(C)=P_C(S_t>0|O$ at C at $t)$ is the conditional probability that recognizer R observes object O when it is at C at time t;
- $false\_neg(C)=P_C(S_t=0|O$ at C at $t)$ is the conditional probability that recognizer R misses object O when it is at C at time t, where $false\_neg(C) = 1-true\_pos(C)$;
- $failure(C)=P_C(S_t=-1|O$ at C at $t)$ is the probability that recognizer R fails at time t.

B. Bayesian Network

The BN [11] in **Figure 2** connects the 3 probabilistic models according to the causal processes that produce the random observations when object O transitions between cameras. The BN is represented as a directed factor graph [11], where the circles represent random events and the rectangle, called *factors*, represent the conditional probability distributions between the random events. O is omitted from the factors for clarity.

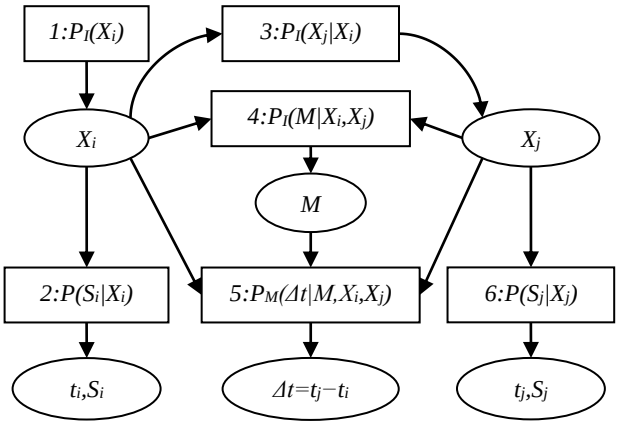

Figure 2: BN represented as a directed factor graph

Object O enters the places covered by the cameras according to factor 1. While at camera X_i at time t_i, object O selects a place to go next according to factor 3, while the place is covered by camera X_j. Then object O chooses mode M according to factor 4. Finally, object O transitions to the selected place during interval Δt according to factor 5. We can combine factors 3, 4 and 5 into one that defines the joint probability distribution of next place, mode and interval given the current place:

$$P(X_j,M,\Delta t|X_i)= P_I(X_j|X_i)P_I(M|X_i,X_j)P_M(\Delta t|X_i,X_j,M) \quad (1)$$

When object O is at camera X_i or X_j, the recognizer will produce observations according to factors 2 and 6, which are defined below:

$$P(S_i|X_i)=(1-falure(X_i))true\_pos(X_i) \text{ if } S_i>0 \quad (2)$$

$$P(S_i|X_i)=(1-falure(X_i))false\_neg(X_i) \text{ if } S_i=0 \quad (3)$$

$$P(S_i|X_i)=falure(X_i) \text{ if } S_i=-1 \quad (4)$$

C. Extended HMM

As object O moves across N cameras in T-1 transitions, the causal processes in the BN (**Figure 2**) are repeated T-1 times to produce T observations. The repetitions of T-1 copies of the BN forms an extended HMM for O in **Figure 3**, which has $N\times T$ states that represent all possible object paths of O that could have produced the T observations.

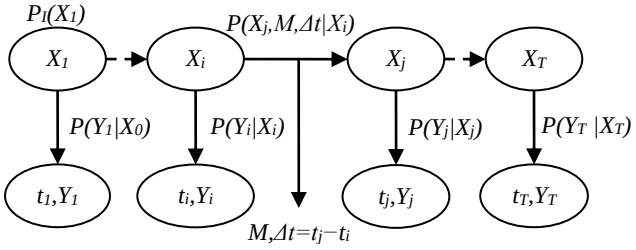

Figure 3: Extended HMM

Since any recognizers can generate false positive observations, concurrent observations of object O at multiple states can occur at the same time, although only one may be true. However, conventional HMM [12] does not permit concurrent observations. To address this problem, we use vector Y_i to represent concurrent observations of N recognizers at time t_i, where $Y_i[j]=S_i$ of the j-th recognizer. For example, $Y_i=[1, 0, 2, -1]$ indicates that at time t_i, recognizer R_1 observes object O in mode 1 and R_3 observes O in mode 2, while R_2 does not observe O and R_4 fails. Furthermore, we assume that each recognizer R observes object O independently at any time. Under this assumption, (5) reduces the probability of concurrent observations to the probability of individual observations.

$$P(Y_i|X_i)=P(Y_i[X_i]|X_i)=P(S_i|X_i) \quad (5)$$

Equations (6)-(8) define the parameters of the extended HMM using the notations in [12], where:

1. a_{ij} is the state transition probability from X_i to X_j;
2. $b_j(k)$ is the probability of observation k at state X_j;
3. π_i is the initial state distribution.

$$a_{ij}=P(X_j,M,\Delta t|X_i) \quad (6)$$

$$b_j(k)=P(Y_j[k]|X_j) \quad (7)$$

$$\pi_i=P_I(X_i) \quad (8)$$

Equation (6) is tied to the intention and motion models by (1), while (7) is tied to the competence model by (2)-(5).

Two well-known algorithms can be used to infer the optimal state sequence with $O(N^2T)$ operations for N states and T observations. The Forward-Backward (FB) algorithm

[12] selects T optimal states X_i that independently maximize $P(X_i|Y_1 \ldots Y_T)$ as follows:

$$q_t = \underset{1 \le i \le N}{\operatorname{argmax}}[\gamma_t(i)], \gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \quad (9)$$

The right-hand side of (9) is computed using the forward pass (10) and the backward pass (11) based on (6)-(8).

$$\begin{aligned} &1)\ \alpha_1(i) = \pi_i b_i(Y_1), 1 \le i \le N \\ &2)\ \alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(Y_{t+1}), 1 \le t < T, 1 \le j \le N \end{aligned} \quad (10)$$

$$\begin{aligned} &1)\ \beta_T(i) = 1, 1 \le i \le N \\ &2)\ \beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(Y_{t+1}) \beta_{t+1}(j), 1 \le t < T, 1 \le i \le N \end{aligned} \quad (11)$$

The Viterbi (VI) algorithm [12] selects T optimal states that jointly maximize $P(X_1 \ldots X_T|Y_1 \ldots Y_T)$ as follows:

$$\begin{aligned} &1)\ \delta_1(i) = \pi_i b_i(Y_1), \psi_1(i) = 0, 1 \le i \le N \\ &2a)\ \delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(j) a_{ij}] b_j(Y_t) \\ &2b)\ \psi_t(i) = \underset{1 \le i \le N}{\operatorname{argmax}} [\delta_{t-1}(j) a_{ij}], 2 \le t \le T, 1 \le j \le N \\ &3)\ p^* = \max_{1 \le i \le N} [\delta_T(i)],\ q_T^* = \underset{1 \le i \le N}{\operatorname{argmax}} [\delta_T(i)] \\ &4)\ q_t^* = \psi_{t+1}(q_{t+1}^*), 1 \le t < T \end{aligned} \quad (12)$$

We extend the FB and VI algorithms to return N-best optimal states at each timestamp, called *N-best path* to be distinguished from object path. A N-best path contains T N-best ranks, and each rank is a sequence of triples (t_i, X_i, P_i) sorted by descending order of P_i, where X_i is the optimal state (camera), and P_i is the probability of the object in state X_i at timestamp t_i given T observations.

IV. Experimental Results

A prototype tracking system was implemented in Python 3.4.3 based on the extended HMM and tested on simulated data. The test procedures are illustrated in **Figure 4**, where the rectangle boxes represent the procedures and the curved boxes represent data. Based on a US city distance map, we bootstrapped the intention, motion and competence models. We then generated object paths and noisy observations from these models. The noisy observations are fed to the extended HMM to infer the N-best paths. Finally, we compare the N-best paths against the object paths to measure the accuracy and performance of the HMM.

The US city map was downloaded from a website [13], which includes distance d_{ij} between 31 cities, where $d_{ij}=d_{ji}$ and $d_{ii}=0$. We simulate a situation where an object (a vehicle) is driving across the cities, while the cameras in the cities may produce noisy observations about the object. The tracker must decide which cities the object has actually visited based on the noisy observations.

A. Probablistic Model Generations

The intention model of the object is derived from the distances by setting $P_I(C_j|C_i,O)=d_{ij}^{-1}/Z_1$ if $d_{ij} \neq 0$, and $P_I(C_j|C_i,O)=0.0$ if $d_{ij}=0$, where $Z_1=\sum_j d_{ij}^{-1}$ normalizes the numbers into a probability distribution. These assignments make the object more likely to travel to closer cities. The initial state distribution is obtained by setting $P_I(C_i|O)=1/Z_2 \sum_j P_I(C_j|C_i,O)$, where Z_2 normalizes the sum into a probability distribution, such that the object is more likely to be in a city which has a more close neighbors.

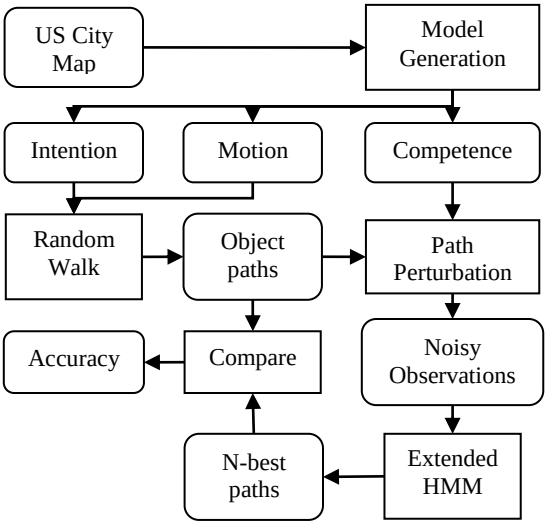

Figure 4: Test procedures

The motion model is derived from distances d_{ij} by setting $\mu=d_{ij}/65$ and $\sigma=d_{ij}/(4\times65)$, based on the average US highway speed limit of 65 miles/hour, and the fact that travel time tends to vary more as the distance increases. As the result, $P_M(\Delta t|C_i,C_j,M) \sim N(d_{ij}/65, d_{ij}/260)$ for driving mode M.

The competence model is derived from a range of random choices. For each camera C_i, we randomly selected $true\_pos(C_i)$ from $\{1.0, 0.90, 0.80, 0.70\}$, $false\_neg(C_i)$ from $\{0.1, 0.01, 0.001\}$, and $failure(C_i)$ from $\{0.01, 0.001, 0.0001\}$, all with uniform distributions, to simulate a tracking system with mixed high-end and low-end recognizers.

B. Object Path and Noisy Observation Generations

We first generated the object paths from the models using a *Random Walk* procedure and then perturbed the paths to obtain the noisy observations using a *Path Perturbation* procedure. An object path is a sequence of (t_i, Z_i, M_i) triples, where t_i is the timestamp, Z_i the camera, and M_i the mode. A noisy observation $Y=[Y_0,..,Y_L]$ is a sequence of observation vectors Y_i as described in Section III, where each Y_i may contain random errors.

The *Random Walk* procedure accepts three arguments: start time t_0, a list of modes ML and step count L and it produces an object path, which is a random sample of the HMM states. The procedure first randomly selects the initial camera Z_0 according to distribution $P_I(C_i|O)$ and then randomly selects a mode M_0 from ML. These selections constitute the first triple (t_0, Z_0, M_0) of the object path. To construct the next triple, the procedure randomly selects the

next camera Z_1 according to distribution $P_I(C_j|Z_0,O)$ and mode M_1 from ML. It then generates a random interval t_1 from Gaussian distribution $P_M(\Delta t|Z_0,Z_1,M_1)$ to construct the second triple (t_1, Z_1, M_1). These steps are repeated L times to produce a path of $L+1$ triples.

The *Path Perturbation* procedure accepts an object path of L+1 triples and produces a noisy observation Y of $L+1$ N-dim vectors Y_i. For each triple (t_i, Z_i, M_i) in the object path, the procedure sets the timestamp of Y_i to t_i, $Y_i[Z_i]=M_i$ and $Y_i[k]=0$ for $k\neq Z_i$. It then randomly selects an error type from set $\{0, 1, 2\}$ to modify Y_i as follows:

- If $type=0$, set $Y_i[Z_i]=0$ with probability $false\_neg(Z_i)$ to simulate false negative errors;
- If $type=1$, set $Y_i[Z_i]=-1$ with probability $failure(Z_i)$ to simulate failures;
- If $type=2$, randomly select two indexes m, n not equal to Z_i and set $Y_i[m]=Y_i[n]=M_i$ to simulate *false positive* errors by cameras C_m and C_n.

For each vector Y_i, there is $false\_neg(Z_i)/3$ chance of 1 type 0 error, $failure(Z_i)/3$ chance of 1 type 1 error, and $1/3$ chance of 2 type 2 errors. Type 0 and 1 errors create observation gaps in the object path, while type 3 errors introduce distractions into the object path. These errors significantly deviate a noisy observation from the true object path at multiple timestamps.

We use two measures, *error count* and *noise ratio*, to quantify the deviation of noisy observations from the object paths. The *error count* of Y is the total number of errors in Y. The *noise ratio* of N observations Y is M/N, where M is the number of observations Y whose error count is greater than 0. Since the error count is proportional to L and the noise ratio is proportional to the error count, both measures will increase as L increases. The plot in **Figure 5** illustrates these relations when L increases from 1 to 20.

C. Accuracy Measurement

After a tracking system produces an N-best path S_b (described in Section III) from noisy observation Y perturbed from an object path S_o, we can measure its accuracy by three metrics: *recall*, *precision* and *F-measure*. Recall measures how many triples in S_o are in S_b, while precision measure how many triples in S_b belong to S_o. If S_b is a perfect match of S_o, then both recall and precision will be 1. If they have no common triple, then both recall and precision will be 0. Higher recall and precision yield higher F-measure. The metrics are calculated by (13), where $NB>0$ decides the size of the N-best ranks in S_b. Since $length(S_b)\geq length(S_o)$, these metrics are in range $[0,1]$.

The *rank* function determines if a triple $R_i=(t_i, Z_i, M_i)$ in S_o can be found in S_b which consists of ranked triples (t_i, X_i, P_i). A match is found if both triples have the same t_i and $Z_i=X_i$. When a match is found, the rank position $0\leq k<NB$ of the N-best triple is returned. The *hit* function weights and accumulates the results of rank functions, such that a higher rank receives more weight up to 1. Since S_0 and S_b have the same length in our case, the recall, precision and F-measure metrics become equal. For this reason, we only include F-measure in our test results.

$$rank(R_i)=\begin{cases} NB \text{ if } R_i \notin S_b \\ k \text{ if } R_i \in S_b \end{cases} \in [0, NB]$$

$$hit(S_o,S_b)=\sum_{R_i \in S_o} \frac{NB-rank(R_i)}{NB} \in [0, length(S_o)]$$

$$recall(S_o,S_b)=\frac{hit(S_o,S_b)}{length(S_o)} \in [0,1] \quad (13)$$

$$precision(S_o,S_b)=\frac{hit(S_o,S_b)}{length(S_b)} \in [0,1]$$

$$F(S_o,S_b)=\frac{2recall(S_o,S_b)precision(S_o,S_b)}{recall(S_o,S_b)+precision(S_o,S_b)} \in [0,1]$$

D. Test data and Results

To simulate object tracking with small amount of observations, we selected path lengths L=1, 2, 3, 4, 5, 10, 15 and 20 to generate 8 sets of 500 object paths and noisy observations. **Figure 5** plots the *error count* and *noise ratio* of the noisy observations. The noisy observations deviate significantly from the object paths and the deviation increases monotonically with path length. At path length 1, 60% observations have over 2 errors. At path length 20, 100% observations have over 14 errors, which means that 66.66% vectors in an observation have some errors.

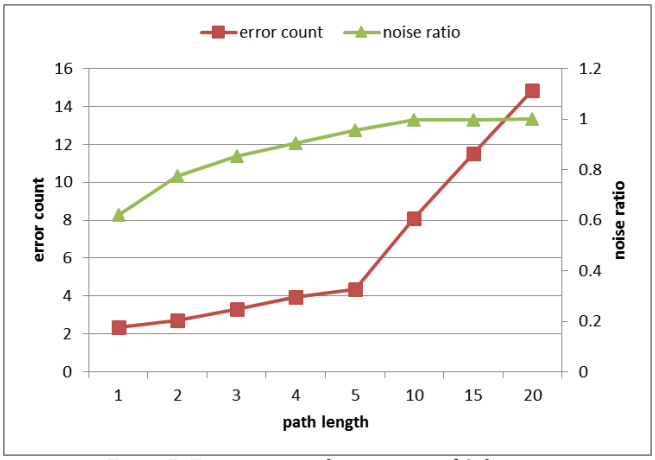

Figure 5. Error counts and noise ratios of 8 datasets

For each of the 8 datasets, we ran 6 configurations (Table I) of the tracking system, by activating different models and inference algorithms. We then averaged the F-measures of the 500 N-best paths as the F-measure of each configuration. The results are shown in Figure 6, where the FB family of configurations 1, 3, 5 clearly outperforms the VI family of configurations 2, 4, 6. The average F-measure of the FB family is 77.3%, with 47.9% relative improvement over the 52.2% of the VI family.

One possible reason that the VI family is more prone to the observation errors than the FB family is because the Viterbi algorithm selects the next optimal state based on the previous one such that one early mistake can derail the entire selections.

 ISBN: 978-1-61208-559-3

TABLE I. 6 CONFIGURATIONS OF TRACKING SYSTEM

	intention	motion	algorithm
1	used	not used	Forward-Backward (FB)
2			Viterbi (VI)
3	used	used	FB
4			VI
5	not used	used	FB
6			VI

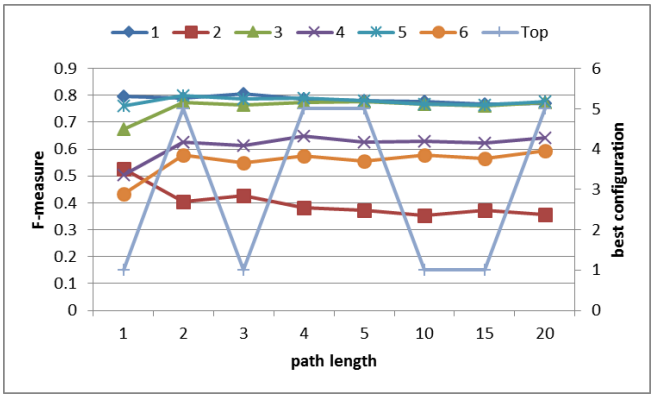

Figure 6. F-measures of 6 tracking configurations on 8 datasets, where the Top line connects the best configurations for each dataset

The top configurations for the 8 tests alternate between configurations 1 and 5, indicating that using the intention and motion models alone is better than combining them. We suspect that using both models may have over-fitted the HMM to the object paths that deviate significantly from the noisy observations.

The F-measures of the FB and VI algorithms are quite stable as the path length increases, with standard deviations of 0.010 and 0.017 respectively. This shows that the algorithms are robust to the increasing noises and errors in the observations shown in **Figure 5**.

All the tests were run on a 32-bit Windows 7 Lenovo T410 notebook computer with 2.67Ghz Dual Core CPU and 3GB RAM. The average tracking time is 26.6 milliseconds with a standard deviation of 0.68.

V. CONCLUSIONS

This paper described a probabilistic method to correct concurrent noisy observations about moving objects from unreliable recognizers (cameras). Our main contributions are:

- A BN to connect 3 probabilistic models of object behaviors and recognizer competence into causal relations to explain the noisy observations;
- An extension to HMM to support inference on concurrent observations from multiple cameras;
- An extension to FB and VI algorithms to return N-best optimal states.

As the simulated tests demonstrated that it is feasible to build a robust tracking system from unreliable recognizers, future work is needed to improve the accuracy. One important direction for future research is to learn the probabilistic models from noisy observations using statistical machine learning techniques [11][12]. It is also interesting to extend the tracking model to deal with more challenging errors in more realistic datasets.

REFERENCES

[1] M. Gunther, L. E. Shafey, and S. Marcel, "Face Recognition in Challenging Environments: An Experimental and Reproducible Research Survey," Face Recognition Across the Imaging Spectrum, Springer, pp 247-280, 2016.

[2] A. P. Patel, V. C. Gandhi, "Survey on Human Gait Recognition," IJCST Vol. 5, Issue 4, Oct - Dec 2014, pp 275-277, 2014.

[3] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," Circuits and Systems for Video Technology, IEEE Transactions on, 23(2), 311-325, 2013.

[4] J. Sochor, A. Herout, J. Havel, "BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3006-3015, 2016.

[5] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang, "Deep Relative Distance Learning: Tell the Difference Between Similar Vehicles," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2167-2175, 2016.

[6] M. Taj and A. Cavallaro, "Distributed and decentralized multi-camera tracking," IEEE Signal Processing Magazine, Volume 28, Issue 3, pp. 46-58, May 2011.

[7] P. Climent-Perez, D. N. Monekosso, and P. Remagnino, "Multi-view event detection in crowded scenes using tracklet plots," The 22nd International Conference on Pattern Recognition (ICPR), pp. 4370-4375, 2014.

[8] Y. Xu, X. Liu, Y. Liu and S.-C. Zhu, "Multi-view People Tracking via Hierarchical Trajectory Composition," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4256-4265, 2016.

[9] D.-T. Lin and K.-Y. Huang, "Collaborative Pedestrian Tracking and Data Fusion With Multiple Cameras," IEEE Transactions on Information Forensics and Security, Vol. 6, No. 4, pp. 1432-1444, December 2011.

[10] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera People Tracking with a Probabilistic Occupancy Map," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 30, No. 2, pp. 267-282, February 2008.

[11] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[12] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of IEEE, Vol. 77, No. 2, pp. 257-286, February 1989.

[13] MapCrow, Cities in United States, http://www.mapcrow.info/united_states.html, last accessed: April 3, 2017.

 ISBN: 978-1-61208-559-3