Non-overlapping Distributed Tracking using Particle Filter

Wilson Leoputra, Tele Tan, and Fee Lee Lim

Department of Computing, Curtin University of Technology, Perth, Western Australia
E-mail: {leoputra,teletan,limfl}@cs.curtin.edu.au

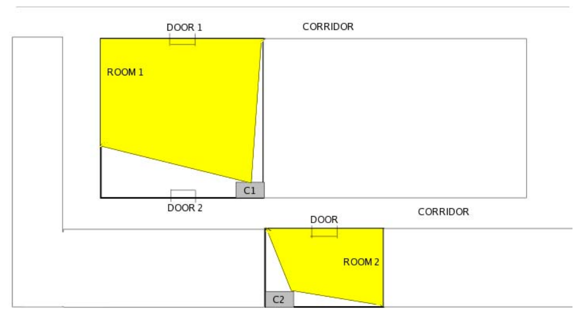
Abstract

Tracking people or objects across multiple cameras is a challenging research area in visual computing especially when these cameras have non-overlapping field-of-views. The important task is to associate a target of interest with its previous appearances across time and space within the camera network. In this paper, we propose a unified tracking framework using Particle Filter to efficiently switch between track prediction (to deal with non-overlapping region tracking) and visual tracking. The Particle Filter tracking system uses a map to provide the possible trajectory information of the target as it moves within the non-overlapping regions. We implemented and tested this tracking approach in an in-house multiple cameras system. Promising results were obtained which suggested the feasibility of such an approach.

1. Introduction

Consider the problem of in-door surveillance involving hundreds or even thousands of cameras distributed across key sections of the environment like in an airport or large building complexes. For economic reasons, most of the cameras do not have overlapped field-of-view (FOV) which makes tracking of people or objects of interest as they move around a difficult and challenging task. We wish to monitor the flow of these targets in this situation such that a summary report of target movement (for example where and when a target of interest appears within the footprint of the surveillance camera). Such function is useful for the purpose of piecing together a complete surveillance picture of the targets movement when one has hundreds or even thousands of video footages to deal with. A simple example is shown in Figure 1, comprising two rooms. The target appears in room 1 (at frame 350), and later, he reappears at room 2 (frame 800). The aim is to correctly associate the target across a network of this distributed camera system.

Previous works such as [6, 10] used primarily Bayesian approaches to deal with the non-overlapping region tracking. Kettnaker and Zabih [6] presented a Bayesian technique using prior information, such as the set of allowable paths and the transition technique, to track people across non-overlapped cameras. Likewise, Porikli and Divakaran [10] presented a framework that included a Bayesian technique to perform the inter-camera target correspondence matching. However, when the size of the environment becomes large, the complexity of the transition probability computation dramatically increases.



(a) Layout of the two rooms and corridors

(b)

(c)

Figure 1. Example of target movements from room 1 at frame 375 (left) to room 2 at frame 800 (right).

Vehicle tracking along freeways is another good example of non-overlapping camera tracking, where cars are identified by the camera based on their appearance and positional information. Huang *et al.* [4, 9] presented their works relating to this problem using the appearance model. Huang [4] combined Bayesian technique and the appearance model to track the car, and the work was further extended by Pasula [9] using Markov Chain Monte Carlo (MCMC) technique. In general, vehicle tracking follows a well-defined path which makes the problem slightly easier to deal with.

When the probabilistic transition models can be supplied to the system, Javed [5]uses these models to match the moving target model. On the other hand, Makris *et al.* [7] do not use such prior information. Instead, they proposed a technique for learning the accumulated information of histogram from the object coming in-and-out of the room, over a certain period of time, which is then use to predict the target's motion.

Dunagan *et al.* [2, 11, 12] proposed techniques for non-overlapping region tracking by estimating the target's position. In the implementation, the distances between cameras are known, along with an assumption that the speed of the person is unknown but constant, which is not suitable for the real-world situation.

In this paper, we introduce a Particle Filter technique to perform tracking between non overlapping cameras. Besides the Particle Filter, we advocate the use of a *map* providing details about the room distribution and the connecting paths between the rooms. The *map* is a readily available resource found within the Geographic Information System (GIS) of most buildings. Once a target leaves a monitored-room, a new sampling technique of the Particle Filter is introduced, which essentially uses the *map* to predict the movement of the targets in the non-overlapped regions. In the literature, the use of *map* was implemented by [1] for a shooting game application. His work uses Particle Filter as the robot's field-of-view to automatically track down the victims and terminate them. This work is well documented in the gaming arena. We will extend it further to real with this surveillance task.

The organization of the paper is as follows. In the next section, we present the fundamentals of the Particle Filter used in our work. In Section 3, we discuss on the new approach using Particle Filter together with *map* to perform the non-overlapping region tracking. In Section 4 we demonstrate the results of our system for non-overlapping region tracking. Section 5 concludes the paper.

2. Background

2.1. Dynamic Model

Generally, tracking problem are inherently dynamic and can be formulated in equation 1 and 2:

$$x_k = f_k(x_{k-1}, u_k) \tag{1}$$
$$z_k = h_k(x_k, v_k) \tag{2}$$

where k is a discrete time index, x_k is the state sequences, and z_k represents the observation measurement. The notation u_k and v_k are noise metrics. The first equation, $x_k = f_k(x_{k-1}, u_k)$, is a function that predicts the current state given the previous state and a noise vector, u_k. The second equation, $z_k = h_k(x_k, v_k)$ defines a measurement model that predict how well the current state is based on current observation. The main objective is to recursively predict the state x_k based on historical observations of $z_{1:k}$.

2.2. Particle Filter

Particle Filter is a probabilistic technique which is based on random measurement density approximate by a set of weighted particles. Each particle is a 2-tuple consisting the state domains and its corresponding probability (weight), denoted by $\{x^i_{1:k}, w^i_k\}^{N_s}_{i=1}$, where i is the particle number and N_s is the number of particles. At each time instance, $x^i_{1:k}$ represents the predicted state of the target by particle i.

Particle Filter is a three phase algorithm, that is Sampling, Importance, and Resampling. Sampling is a process of locating all particles randomly to predict the next target's location. This is done using a distribution, which is usually called a Probability Density Function (pdf). Mathematically, the sampling technique can be expressed as [3]:

$$x^i_k \sim q(x^i_k | x^i_{k-1}, z_k) \tag{3}$$

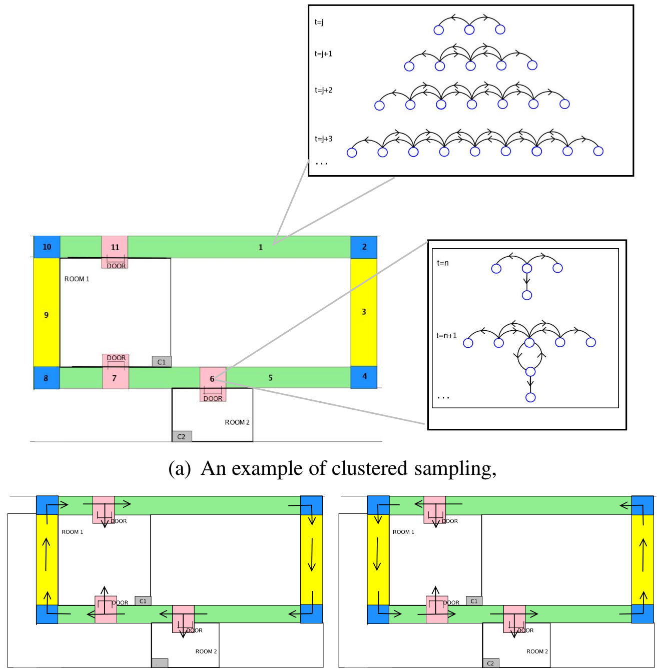
where $q(x^i_k | x^i_{k-1}, z_k)$ is the pdf. In the context of tracking, equation 3 can be worded as: *how can the system predicts the target's position, at current time period* k*, given a knowledge of where the target might be at an instant before* $k-1$*?*. The answer is: *given an input particle at* (x, y) *coordinate, the posterior distribution is equal to the addition of some random numbers with the particle's position of* (x, y) *coordinate*. This implies that the target may move randomly, but still be found by a number of particles [1].

In order to measure the wellness of the states prediction, all the particles are weighted based on how close they are to the observation. The weighting function refers to the computation on the previous weight times the probability of the real observation to the states divided by the sampling factor, which is defined in equation 4 and normalised in equation 5

$$w^i_k = w^i_{k-1} \times \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{k-1}, z_k)} \tag{4}$$
$$w^i_k = \frac{w^i_k}{\sum^{N_s}_{i=1} w^i_k} \tag{5}$$

In other words, both equations 4 and 5 are expressed in the following sentences: *Given what the system can see, update my belief about where the target is.* Blobs are chosen as the current observation measurements. If there exists such observation, then all particles are weighted proportional to their distance from the target. Otherwise, all the particles are assigned with the same weight.

Resampling procedure maintains large weight particles while removing small weight particles. Over time, the weight of a number of particles will become very large while the remaining particles have value that is relatively small, which are negligible. Thus, the resampling stage ensures the preservation of the larger-weighted particles. There are several ways to perform resampling and in this paper, we stick to the original resampling method proposed in [3].



(a) An example of clustered sampling,

(b)

(c)

Figure 2. An example of Particle Filter sampling given known environment on Figure 1.

3. Non-Overlapping Region Tracking using Particle Filter

3.1. Non-Overlapping Region Prediction

To extend the original Particle Filter technique into the non-overlap region tracking, a geographic information of the environment is needed, namely a *map*. Let n be the number of particles and C_k be a set of clusters $C_k = \{C_k^1, C_k^2, ..., C_k^s\}$ where s is the number of clusters at time k. Each cluster, j, $[C_k^j]$ contain a set of particles that resides in the same position.

$$x_k^i \in \begin{cases} C_k^j & Distance(x_k^i, C_k^j\{x_k^n\}_{l=0}^{Ns}) = 0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Hence the new sampling procedure can be expressed in equation 7, named as clustered sampling:

$$\begin{aligned} x_k^i &\sim q(x_k^i|x_{k-1}^i, m) \\ &\sim q(x_k^i|x_{k-1}^i \in C_{k-1}^j, m) \end{aligned} \quad (7)$$

where m is the *map* information and C_k^j is the j^{th} cluster that contains particle i. Note that in the sampling method in equation 7, there is no measurement z_k and the addition of parameter m, as compared to the generic Particle Filter. This implies that no measurement is available and the particles are sampled based on the cluster and the *map*. The whole process can be depicted in Figure 2.

Initially the prediction of Particle Filter will have each cluster containing more than one particle. This implies that it is able to be sampled in all different directions, specified by the *map*. When a cluster contains only a single number of particles, its movement follows one exact path. In the case of our example, it will employ either the clockwise path or anticlockwise path, as shown in figure 2(b) and 2(c) respectively. However, if a particle encounters a group, which is of same speed, the particle will again perform the clustered sampling.

The number of particles required is calculated based on the environment's area. The larger the environment, the greater the number of particles required. In the implementation, we divided the particles into three different speed models, which is enough to cover the whole area. As a rule of thumb, the number of particles required for each group can be expressed in equation 8

$$N_s = \frac{N_A}{N_g} \quad (8)$$

where N_s is the number of particles needed for each group, N_A is number of steps to cover the whole area, and N_g is number of groups (in terms of speed models).

Once a target re-appear in the camera's field-of-view, one of the particles will be able to associate it. Then, the corresponding appearance model is checked from the database. The details of the association process is presented in Subsection 3.2. The final matching procedure is based on weighting equation:

$$\text{match} = (1-\alpha) \times \text{PF} + \alpha \times \text{HM} \quad (9)$$

where α is the proportional weighting importance for Particle (PF) and Histogram Matching (HM). α is weighted based on the time. The longer the target spends its time in the non-overlapping region, the lower the Particle Filter's prediction. In other words, it indicates a larger number of possibility would be of the target's movement. Hence, we slowly transit our emphasis on color matching. In addition, we also incorporate *timeout* in our work. Considering the problem where the target completely leave the environment, after certain *timeout* period we ignore the prediction, assuming the target has left the environment. In the implementation, we set the *timeout* to be 5 minutes.

3.2. Association

The features used in the tracking process are velocity and colour. This section describes the colour matching technique. When the target is in the view, Particle Filter is used to keep track of its trajectory. At the same time, the target's colour information is obtained, which is represented by RGB values. This value is stored in a 3D colour histogram (CHist[r,g,b]), i.e. an $8 \times 8 \times 8$ histogram. Over time, these colour histograms is used to represent the appearance profile of the person.

A "main database" is used to store all the target's color histograms. In the experiment, up to twenty histograms were kept for each person. The main database is divided into three sub databases. The first sub database contains all the targets currently in the camera field-of-view. The second sub database contains the target that is under particle filter's prediction phase. The entries in the second sub database will be used when a new target reappears in one of the camera field-of-view. The last sub database stores only the log of the target that has left the surveillance environment.

Recall that the appearance based association will only be used under two conditions, either when a target leaves one room and reappears in another camera field-of-view, or when a number of individual objects split after merging within the field-of-view of the camera. A *histogram intersection* technique proposed by Swain & Ballard [13], which returns a matching score determining the similarity or dissimilarity between two histograms. This matching score has value between 0 (dissimilar) to 1 (similar) inclusively. We set a threshold of 0.7 for the matching histograms. When a target is checked against the model in the second sub database, the highest probability of matching score between all the models in database is used to confirm the target.

3.3. Within field-of-view

When a target is track in a monitored region, a classical Particle Filter based on Section 2 is implemented. That is implementing the original Particle Filter algorithm by [8].

4. Experimental Results

The system was tested on environment under surveillance, comprising two rooms and corridors situated within the Smart House Lab. Figure 1(a) shows the layout of the experimental environment. Note that, room 1 has two entry/exit doors labeled as door1 and door2, while room 2 has one entry/exit door.

A selected number of frames of the experiment is shown in figure 3. This experiment shows that Particle Filter with *map* is able to track a number of targets for non-overlapped region tracking, while the complete sequence of the tracking process is available at http://www.computing.edu.au/∼12482661/avi/index.html.

For the testing purposes, twenty-five sets of video sequences were collected. Each video consists of 1500 to 2200 frames with 53 transactions (1 to 5 transactions for each video sequence) in total. The outcome is the system has successfully tracks and associate targets for twenty sets of video sequences. These video sequences include various kind of walking pattern such as merging and splitting, target turns back, time out, and occlusion. For the remaining 5 video sequences where the system fails for a number of tracking situations. This is due to the system false positively identifying some similar colour profile targets. Another reason was when targets spend a large amount of time on the non-overlapped area. This leads to the prediction of Particle Filter becomes less accurate. Overall, the following table shows the quantitative measurement for all the number of transactions.

System performance	
Hit	44
Missed	9
Total	53

From the above table, we conclude that the system is able to perform with accuracy rate of 83.01% ($\frac{44}{53}$). Due to the time constraint, only a limited number of tests is presented. However, in the future, this system can be further tested under a larger environment with more targets in the scene.

5. Conclusions

In this paper, we have proposed, implemented, and tested a unified framework of tracking human subject across a distributed camera system. A Particle Filter-based approach was used with the ability to switch between visual tracking, when the subject is seen by the camera and track prediction, when the subject has left the field of view of the camera and enters into the non-overlapping region of the system. The latter prediction function is accomplished using *map* of the surveillance area. The *map* acts as the main input to the Particle Filter, which control the movement of the Particle Filter in the surveillance space. The use of a *map* provides a scalable solution to deal with an expandable area of interest (e.g. scaling from building level to street level tracking) typical of real-world surveillance applications. Experiments conducted so far have been encouraging. One of our near-term task is to increase the number of cameras in the network and measure its performance under real-world conditions.

References

[1] C. Bererton. State Estimation for Game AI Using Particle Filters. In *AAAI workshop on challenges in game AI*, Pittsburgh, PA, USA, 2004.

[2] A. R. Dick and M. J. Brooks. A Stochastic Approach to Tracking Objects Across Multiple Cameras. In *Proc. Australian Joint Conference on Artificial Intelligence*, pages 160–170, Mawson Lakes, SA, 2004.

[3] N. Gordon, D. Salmond, and A. Smith. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. *IEE Proceedings-F on Radar and Signal Processing*, 140(2):107–113, 1993.

[4] T. Huang and S. J. Russell. Object Identification in A Bayesian Context. In *International Joint Conference on Artificial Intelligence*, pages 1276–1283, Nagoya, Japan, 1997.

[5] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking Across Multiple Cameras with Disjoint Views. In *International Conference on Computer Vision*, pages 952–957, Nice, France, 2003.

[6] V. Kettnaker and R. Zabih. Bayesian Multi-camera Surveillance. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 253–259, Miami, FL, USA, 1999.

[7] D. Makris, T. Ellis, and J. Black. Bridging The Gaps Between Cameras. In *Conference on Computer Vision and Pattern Recognition*, 2, pages 205–210, Washington, DC, USA, 2004.

[8] S. Maskell and N. Gordon. A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking. *IEE Target Tracking: Algorithms and Applications (Ref. No. 2001/174)*, 2:2/1 – 2/15, Oct 2001.

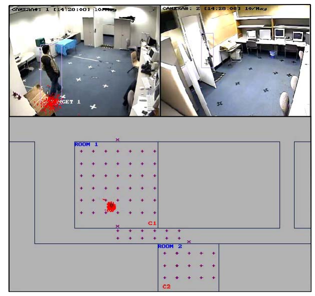
[9] H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov. Tracking Many Objects with Many Sensors. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1160–1171, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[10] F. Porikli and A. Divakaran. Multi-camera Calibration, Object Tracking and Query Generation. In *Proceedings International Conference on Multimedia and Expo*, volume 1, pages 653–656, Baltimore, Maryland, July 2003.

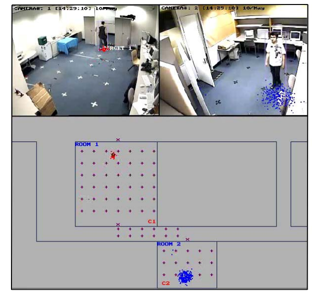
[11] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous Calibration and Tracking with A Network of Non-overlapping Sensors. In *Conference on Computer Vision and Pattern Recognition*, 1, pages 187–194, Washington, DC, USA, 2004.

[12] A. Rahimi, B. Dunagan, and T. Darrell. Tracking People with A Sparse Network of Bearing Sensors. In *European Conference on Computer Vision*, 4, pages 507–518, Prague, Czech Republic, 2004.

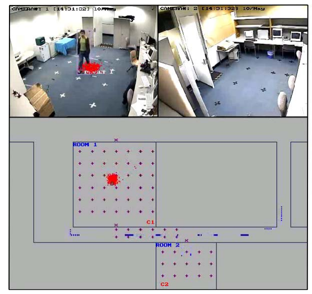
[13] M. Swain and D. Ballard. Indexing Via Color Histograms. In *Proceedings of the DARPA Image Understanding Workshop International Conference on Computer Vision*, pages 390–393, Pittsburgh, PA, USA, 1990.



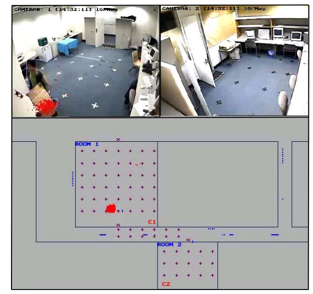
(a) Frame 260



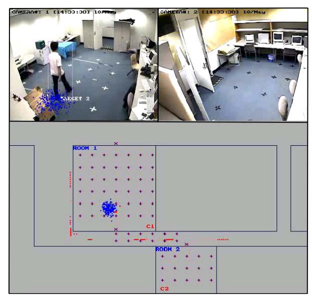
(b) Frame 370



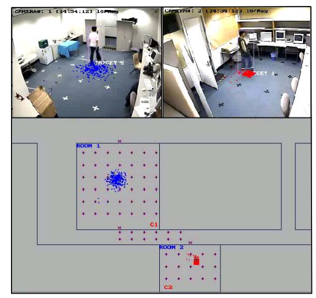
(c) Frame 620



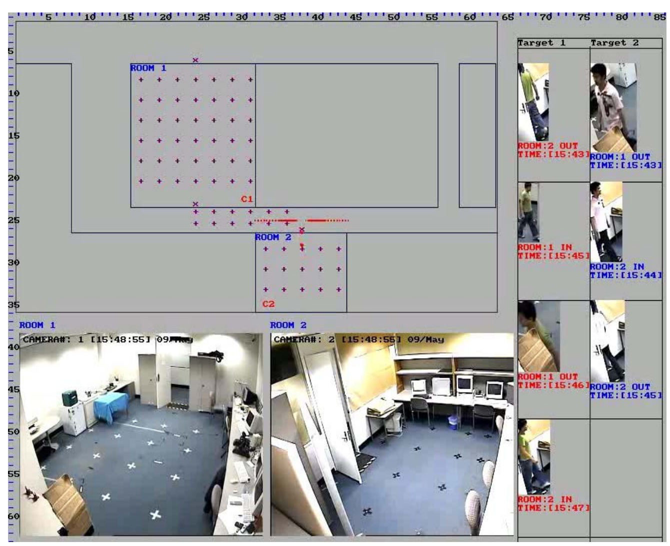
(d) Frame 680



(e) Frame 820



(f) Frame 880



(g) Snapshot after program execution

Figure 3. Experimental results.