Resource Allocation for Tracking Multiple Targets Using Particle Filters

Aniruddha Kembhavi[†] William Robson Schwartz[⋆] Larry S. Davis[⋆]
[†]Department of Electrical Engineering [⋆]Department of Computer Science
University of Maryland, College Park, MD, USA
anikem@umd.edu, schwartz@cs.umd.edu, lsd@cs.umd.edu

Abstract

Particle filters have been very widely used to track targets in video sequences. However, they suffer from an exponential rise in the number of particles needed to jointly track multiple targets. On the other hand, using multiple independent filters to track in crowded scenes often leads to erroneous results. We present a new particle filtering framework which uses an intelligent resource allocation scheme allowing us to track a large number of targets using a small set of particles. First, targets with overlapping posterior distributions and similar appearance models are clustered into interaction groups and tracked jointly, but independent of other targets in the scene. Second, different number of particles are allocated to different groups based on the following observations. Groups with higher associations (quantifying spatial proximity and pairwise appearance similarity) are given more particles. Groups with larger number of targets are given a larger number of particles. Finally, groups with ineffective proposal distributions are assigned more particles. Our experiments demonstrate the effectiveness of this framework over the commonly used joint particle filter with Markov Chain Monte Carlo (MCMC) sampling.

1 Introduction

The problem of object tracking has been the subject of a very large body of research. The vast improvement in the performance and the reduction in camera costs, coupled with ever increasing computing resources has led to the development of many sophisticated single and multi-target tracking algorithms. Specifically, the advent of intelligent surveillance systems has focused research efforts on the problem of detecting and tracking multiple humans.

More recently, target tracking has been dominated by sequential Monte Carlo methods. The most popular Monte Carlo tracking method is the Condensation algorithm [8], commonly referred to as a particle filter. Particle filters have been used both widely and successfully to track objects in video sequences. However, as the number of targets to be jointly tracked increases, the dimensionality of the state space increases and the number of particles needed to sample this space rises exponentially. For a large number of targets often seen in surveillance videos, particle filters require an infeasible number of computations.

Using multiple independent particle filters alleviates this tractability problem at the cost of performance. Independent particle filters often suffer from the problem of *hijacking* [10]. When two or more targets come close to one another, the target with the best likelihood score often hijacks the other filters, leading to tracking errors. Hijacking is boosted under the following conditions. Targets that lie in close proximity to one another in the state space may hijack the other's filter. The proximity of two targets in the state space can be estimated by a measure of the overlap of their posterior probability distribution functions. Hijacking also increases when the interacting targets have similar appearance models.

We present a particle filtering framework along with an overlying graph structure that allows us to deal with the problem of hijacking without increasing the computational cost exponentially. A graph, whose nodes are the targets in the image and whose edges are a measure of the proximity and appearance similarity between targets is used to cluster the targets into multiple and possibly overlapping *interaction groups*. Targets that might hijack each other's filters are grouped into a single group and tracked jointly, while targets in different groups are tracked independently. Our framework allows us to track all targets in the scene, while seamlessly allowing interaction groups to be formed and split at any time instant. The overall tracking framework is discussed in Section 3. An example of targets being clustered into different groups at different time instants is shown in Figure 1.

We also address the problem of allocating resources to the different interaction groups identified in the scene. In the particle filtering framework, the resources are essentially the total number of particles, which directly determines the computational cost. Given an upper bound on the number of particles to be used at any time instant, how

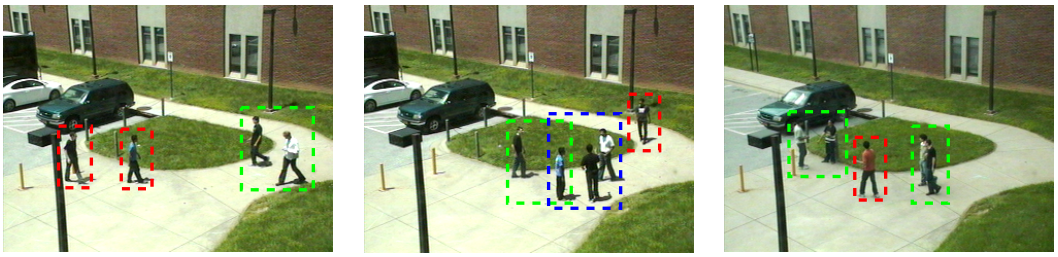

Figure 1. 3 frames from surveillance camera 1 monitoring a parking lot. The bounding rectangles show targets that are grouped together into a single *interaction group* and tracked jointly. Red rectangles show targets that are grouped individually, whereas the green and blue rectangles show groups of two and three targets respectively. Notice that the groups may overlap.

should we distribute them to minimize the number of tracking errors? Our solution to this optimization problem is based on the following observations. First, we assign more particles to groups with a larger number of targets due to their increased state space dimensionality. Second, we assign more particles to groups with higher *associations*. The association of a group quantifies the spatial proximity and pairwise appearance similarities of the targets in the group. Third, we assign more particles to groups whose filters have a higher effective particle sample size at the previous time instant. This gives a measure of the ineffectiveness of the particle proposal distributions for each group. We further elaborate on the issue of resource allocation in Section 4.

We demonstrate our particle filtering framework on video sequences captured from two surveillance cameras overlooking a pedestrian walkway and a parking lot. Sequences from Camera 1 have up to 5 people simultaneously, while those from Camera 2 have up to 9 people simultaneously in the scene with an average height of just 27 pixels in the image. We provide comparisons to two other methods - first, using multiple independent particle filters and second, using a single joint particle filter with MCMC as a speed-up mechanism. Our results clearly demonstrate the superiority of our particle filtering framework over the others, while using the same number of particles.

2 Related Work

Given the extensive amount of work carried out in the field of object tracking, we are unable to provide an exhaustive literature review. Here, we present some of the more relevant bodies of work. For a good survey, we refer the reader to [16].

Many Bayesian approaches to the tracking problem have been explored before. When the posterior density of the targets at every time step can be assumed to be Gaussian, Kalman filters provide the optimal solution [3]. When such constraints are invalid and need to be relaxed, the optimal solution typically becomes intractable. This has led to several approximation algorithms such as the extended Kalman filter, approximate grid-based filter [1] and the particle filter. The particle filter, originally introduced as the Condensation algorithm in the computer vision community, was proposed in [8], and has been widely and very successfully used for tracking targets [7][9][12].

There has also been research aimed at speeding up particle filters, so as to be able to track multiple targets in video sequences for real-time applications. Khan et al. replace the traditional importance sampling step in the particle filtering framework by a Markov Chain Monte Carlo (MCMC) sampling step. This leads to an efficient sampling of the target posterior distribution [10]. They also incorporate a Markov Random Field (MRF) to model interactions between targets to prevent hijacking. Zhao et al. [18] use MCMC with jump/diffusion dynamics to sample the posterior distribution. They also perform a detailed target occlusion analysis and are able to track many humans together in a crowded environment. In Section 5, we compare our particle filtering framework with a single joint particle filter using MCMC as a speed-up mechanism. Our method obtains superior results when using the same number of particles, due to the appropriate particle distribution scheme.

Yang et al. use a Hierarchical Particle Filter [15], whereby they break down the multi-feature observation likelihood to be computed in a coarse to fine manner. This allows the computation to quickly focus on more promising regions and speeds up the entire system. They also employ optimized computational techniques such as Integral Histograms [13]. Khan et al. [11] introduce an efficient method for using subspace representations in a particle filter. They apply Rao-Blackwellization to integrate out the subspace coefficients in the state vector. Since part of the posterior is analytically calculated, the number of particles required decreases, which in turn speeds up the system. Brandao et

al. [2] speed up the particle filtering approach by dividing the search space into subspaces that can be estimated separately. Low correlated subspaces are estimated with parallel or serial filters and their probability distributions are combined by a special aggregator filter. Sankaranarayanan et al. [14] present a method for implementing the particle filter using the Independent Metropolis Hastings sampler, that is highly amenable to pipelined implementations and parallelization. Zhou et al. [19] use an adaptive number of particles determined by the variance of the adaptive noise model at the current time step. As the variance decreases, the computational cost of the filter reduces. Gupta et al. [5] use an approach for camera selection and inference ordering to reduce the computational cost required to track people in a multi-camera framework.

Yu et al. [17] present a decentralized approach to multiple tracking for the emerging application of sensor networks. In order to distribute the computation amongst multiple sensing units, they employ a set of autonomous and collaborative trackers. Dowdall et al. [4] also employ a distributed network of individual trackers. The interactions of these trackers are modeled using coalitional game theory. As in our tracking framework, such decentralized approaches address the tracker coalescence problem (the problem of hijacking filters). Our framework further allows us to allocate resources intelligently. This leads to superior tracking performance using a lesser number of particles.

3 Tracking Framework

Let X_t denote the state of the system and Z_t denote the observation at time t. In a Bayesian tracking framework, we wish to estimate the posterior distribution $P(X_t|Z_t)$ given in Equation (2). In a particle filter, the posterior distribution at time t is approximated by a set of particles with weights π^r, as shown in Equation (3). $P(Z_t|X_t)$ denotes the likelihood function and $P(X_t|X_{t-1})$ denotes the proposal distribution. In a regular particle filter, at every time step, new particles are generated from the particles at the previous time step using the proposal distribution, and their weights are obtained using the likelihood function. This gives the posterior distribution for the current time step. The number of particles needed to sample a higher dimensional state space increases exponentially with the dimensionality of the space (determined by the number of targets in the scene).

$$P(X_t|Z_t) = \frac{P(Z_t|X_t)P(X_t)}{P(Z_t)} \quad (1)$$

$$= \Phi P(Z_t|X_t) \int_{X_{t-1}} P(X_t|X_{t-1}) P(X_{t-1}|Z_{t-1}) \quad (2)$$

$$\approx \Phi P(Z_t|X_t) \sum_r \pi^r_{t-1} P(X_t|X^r_{t-1}) \quad (3)$$

Φ denotes the normalization constant. Let N_t be the total number of targets in the scene at time t. In our particle filtering framework, at every time t, we cluster targets into possibly overlapping *interaction groups*, so that targets within a single group are tracked jointly, but targets in different groups are tracked independently of each other. This group structure can change at every time instant. The groups at time t are denoted as $g^j_t \ \forall \ j = \{1, 2, .., G_t\}$, where G_t is the total number of groups at the time instant t. When referring to a single group, we will often drop the superscript j for ease of reading. We use λ^g_t to denote the set of all targets in group g_t, and k^n_t to be the set of all groups of which target n is a member at time t. As an example consider Figure 2(a). Based on the proximity and appearance of the 4 targets in the scene, 3 groups have been formed. Thus, $\lambda^1_t = \{1\}$,$\lambda^2_t = \{2, 3\}$ and $\lambda^3_t = \{2, 4\}$. $k^1_t = \{1\}$, $k^2_t = \{2, 3\}$, $k^3_t = \{2\}$ and $k^4_t = \{3\}$.

The overall framework is summarized in Algorithm 1 and illustrated with an example in Figure 2. At time $t-1$, N_{t-1} targets are clustered into G_{t-1} interaction groups. Each group is characterized by a joint distribution given by $P(X^{\lambda_g}_{t-1}|Z_{1:t-1})$. For each group g^j_{t-1}, new particles are proposed and their joint likelihoods are calculated (this is done independent of other groups in the scene). The resulting posterior distribution for each group is then marginalized to obtain the marginal posterior distribution for every target in the group. Targets belonging to multiple groups will thus have multiple such distributions. Inspired by work in the field of sensor fusion based on particle filters [6], the particle filter corresponding to each group in the scene can be considered a logical sensor. For any given target, multiple distributions can be thought of as being generated by multiple logical sensors. Similar to [6], they can be combined linearly using appropriate mixture weights κ^g_{t-1}, to obtain a resultant posterior distribution per target. Thus, for every target n in the scene,

$$P(X^n_t|Z_{1:t-1}) = \sum_{j \in k^n_{t-1}} (\kappa^j_{t-1} P_j(X^n_t|Z_{1:t-1})) \quad (4)$$

Every group has a mixture weight which represents a measure of the group tracking confidence. This confidence value for each group is obtained at the previous time instant, and is determined by the likelihood estimates of the particles representing the corresponding group posterior distribution at time $t-1$. Combining information from multiple filters has the added advantage of yielding more robust resultant distributions for each target in the scene. An example can be seen in Figure 2(b,c), where filter 2 has erroneous particles with large weights. However, since the mixture weight for filter 3 is higher than that of filter 2 (obtained from the previous time instant), the resultant posterior distribution for target 2 is improved.

Using the resultant posterior distribution for each target

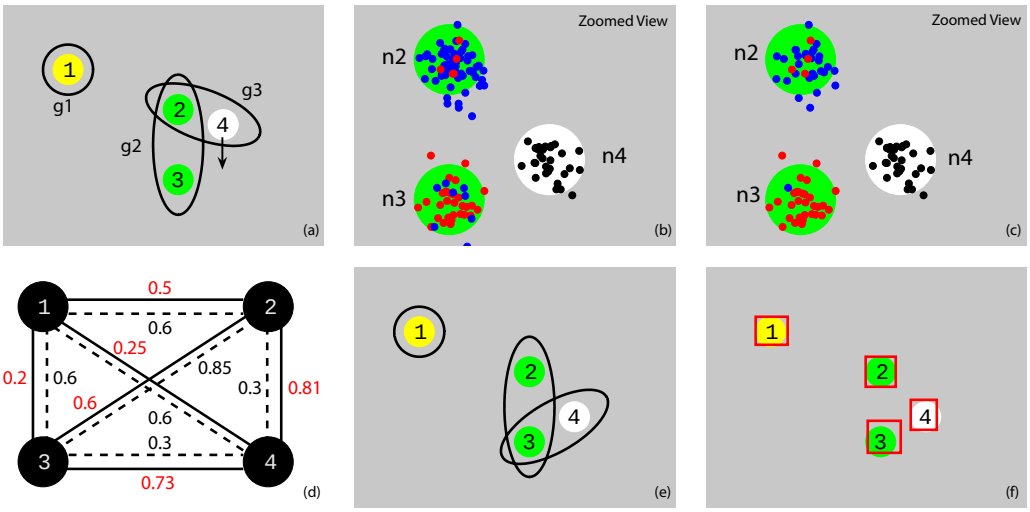

Figure 2. A synthetic example demonstrating our tracking framework. (a) 4 targets n_1 to n_4 clustered into groups g_1 to g_3 at time $t-1$, where n_4 is moving south. (b) (Zoomed in view at time t). Particles representing the marginal distributions of the targets (blue for n_2, red for n_3, black for n_4). n_2 has twice the number of particles, representing two marginal distributions. Filter for g_2 has erroneous particles with high weights since n_2 and n_3 have very similar appearances. However, filter for g_2 has a lower mixture weight than the filter for g_3 based on the likelihood estimates of the previous frame (not shown). (c) The resultant marginal distributions are thus more robust and have fewer erroneous particles with high weights. (d) Proximity (solid) and appearance similarity graphs (dashed). (e) New group structure at time t. (f) Maximum likelihood particles shown for all 4 targets in the scene.

in the scene, we build a *proximity graph* for all targets in the scene, where each node in the graph is a target and an edge between two nodes represents the similarity between the posterior distributions of the two targets. We first estimate these non-parametric distributions from the corresponding particles using Kernel Density Estimation (KDE), and use the Kullback-Liebler (KL) distance between the two. Our state space for each target is 4-dimensional (x-position(x), y-position(y), width(w), height(h)). Using gaussian kernels K and M particles, the density estimate is given by,

$$\hat{p}(x,y,w,h) = \frac{1}{M}\sum_{m=1}^{M} K_{\sigma_x}(x-x_m)K_{\sigma_y}(y-y_m) \\ K_{\sigma_w}(w-w_m)K_{\sigma_h}(h-h_m) \quad (5)$$

An *appearance similarity graph* is also built with each edge weight set equal to the similarity between the appearance models of the two targets. We model the appearance of targets using histograms in each color channel, and use the L2 distance as a distance measure. The proximity graph (PG) and appearance similarity graph (AG) are combined linearly to form a single similarity graph (SG) which is used to obtain a new group structure for the current time step t.

$$SG(i,j) = \alpha_{PG}PG(i,j) + \alpha_{AG}AG(i,j)...\forall i,j \quad (6)$$

The issue of clustering targets into interaction groups based on the similarity graph is dealt with in the following section. Given the marginal distributions for every target and the new group structure $g_t^j, j = \{1,2,..,G_t\}$, we need to obtain the appropriate joint distributions for each group. For every group g_t^j, we sample the marginal distributions of the interacting targets to obtain smaller subsets of particles, and combine them combinatorially to obtain particles for the joint distributions. For every new particle however, the joint likelihoods must be recalculated. Resampling is again performed on these particle distributions. The number of particles given by the resampling algorithm is set by the resource allocation function described in the following section. The number of particles assigned to each group at time t determines the computational cost incurred at the next time instant $t+1$.

4 Resource Allocation

At every time step t, we use the Similarity Graph (SG) described in Equation (6) to distribute our resources so as to reduce the number of tracking errors. In a particle filtering framework, our resources are the total number of particles

Algorithm 1 Overall tracking framework

for $t = 2$ to T **do**
  N_{t-1} targets in the scene are divided into G_{t-1} groups, each represented by a joint density.
  for every g_{t-1}^j **do**
    Propose particles and calculate their joint likelihoods.
    Obtain marginal distributions for each target in the group (may lead to multiple marginal distributions for each target in the scene).
    Resample each marginal distribution.
  end for
  for $n = 1$ to N_{t-1} **do**
    Combine marginal distributions for each target to obtain a resultant distribution per target.
  end for
  Build a *proximity* and an *appearance graph*.
  Obtain new groups g_t^j using the greedy algorithm (Section 4).
  for every g_t^j **do**
    Combine marginals to form a new joint density.
    Recalculate joint likelihoods for all particles representing the new joint distribution.
    Resample to obtain appropriate number of particles given by the resource allocation function (Section 4).
  end for
  Given the joint likelihoods for every group, update group mixture weights.
end for

that can be used at every time step. Distribution of these particles at every time step is carried out in two stages.

Targets that have a high similarity score with each other (nodes with large edge weights between them in the SG) must be grouped together (tracked jointly). Hence, in the first stage, we cluster targets in the scene into interaction groups based on the SG. We define a binary decision variable $\phi(i,j)$ that determines if targets i and j will be grouped together, and a cost function $C(g)$ that determines the cost of tracking targets in group g jointly. The optimization problem can be stated as follows,

$$\max \sum_{\forall i,j} \phi(i,j) SG(i,j) \quad \text{such that} \quad \min \sum_{i=1}^{G_t} C(g_t^i) \quad (7)$$

We solve the above optimization problem using an approximate iterative greedy algorithm for the purpose of efficiency. First we set $\phi(i,j) = 0, \forall i,j$. At every iteration of the algorithm, we select an edge (i',j') with the maximum edge strength, set $\phi(i',j') = 1$ and recalculate the structure of the groups formed by the result of the addition of the edge (i',j'). If the cost of this group structure is less than the predetermined maximum cost, we go to the next iteration. If the cost of the group is above the maximum, we reset $\phi(i',j')$ to 0 and the algorithm terminates. At every such iteration, given a graph with edges $\phi(i,j)$, the structure of the interaction groups is given by the set of all maximal cliques in the graph. Since we add only a single edge to the graph at every iteration, we compute an approximation to the set of all maximal cliques by updating the set of maximal cliques in the previous iteration with the new edge. Thus the computation required at each iteration is kept low. The cost function $C(g)$ is set to be quadratic in the number of targets in group g. In this stage, all groups having the same number of targets are assigned an equal cost, irrespective of the corresponding edges in the similarity graph. Thus the first stage outputs the group structure for the current time step.

In the second stage of resource allocation, we distribute particles amongst groups based on three criteria. The first criterion is the strength of the Similarity Graph edges between targets in the group, given by the *association* of the group. We define the association of group g_t as the average edge strength of all edges in the group,

$$Assoc(g_t) = \frac{1}{\binom{len(g_t)}{2}} \sum_{\forall i,j \in g_t} SG(i,j) \quad (8)$$

where $len(g_t)$ is the number of targets in group g_t. This encapsulates the proximity between the targets in the state space as well as the similarity of the appearances of all targets in the group. The number of particles assigned to the group increases with the association.

The second criterion to allocate resources to a group is motivated by the degeneracy phenomenon seen in particle filters [1]. A common problem with particle filters is that with every iteration, the number of particles with non-negligible weights decreases. The rate of degeneracy depends on the effectiveness of the particle proposal distributions (which involves the motion models learnt from the previous frames and variance estimates of the noise models). Though the degeneracy problem is overcome by the

use of resampling, we argue that a measure of group degeneracy, calculated before the particle resampling stage, should be used to allocate a larger or smaller number of particles to that group. This is because greater the effect of degeneracy, poorer the proposal distribution, larger the number of particles that must be assigned to ensure an effective search of the state space and reduce tracking errors. A suitable measure of degeneracy of the group particle filter is given by the Effective Sample Size (P_{eff}) of the particles characterizing the posterior distribution of the group at the previous time instant [1]. This can be obtained as,

$$P_{eff} = \left[\sum_{r=1}^{P_s} \left(\pi_{t-1}^{r} \right)^2 \right]^{-1} \tag{9}$$

where π_{t-1}^{r} represents the weight of the r'th particle at time instant $t-1$, and P_s is the total number of particles. We define the fraction of effective particles (F_{eff}) as $F_{eff} = P_{eff}/P_s$ and use it as the second criterion to allocate resources. The number of particles assigned to a group increases with a decrease in the value of F_{eff} for the group.

The third criterion to allocate resources to a group is the mean likelihood of the particles $\Pi_{mean}(g_t)$ forming the posterior distribution of the group at the previous time instant. The mean likelihood (weights) of the particles determines the effectiveness of the corresponding particle filter to sample the posterior distribution. A low mean likelihood at the previous time instant necessitates an increase in the number of particles and an increase in the variance of the noise model. This ensures better sampling of the posterior distribution and thus more accurate tracking.

If a group g_t is a new group formed as a result of a change in the group structure at time t, it will have no corresponding group at the previous time instant. In such cases, g_t is assigned a default mean likelihood score.

$$\Pi_{mean}(g_t) = \frac{1}{P_s} \sum_{r=1}^{P_s} \pi_{t-1}^{r} \tag{10}$$

The association $Assoc(g_t)$, fraction of effective particles $F_{eff}(g_t)$ and mean likelihood $\Pi_{mean}(g_t)$ are linearly combined and used to determine the allocation of particles to each group from the total set of available particles. We also ensure that the number of particles assigned to a group lies within pre-defined minimum and maximum values. These bounding values are based on the size of the group.

5 Experiments

We evaluated our tracking framework on video sequences collected from two surveillance cameras (resolution 320x240 pixels and framerate 15fps). Camera 1 overlooks a pedestrian walkway adjoining a parking lot (Figure 1). Camera 2 overlooks the parking lot (Figure 3). Multiple people enter and leave the scene. The maximum number of simultaneous people being tracked is 9. The tracking task in camera 2 is quite challenging since the average height of persons in the scene is only 27 pixels, and frequent occlusions are observed due to the large number of people.

We compared our tracking framework to two other methods - first, multiple independent particle filters and second, a single joint particle filter using MCMC as a speed up mechanism [10][18]. In our method, targets within a single interaction group were tracked using a traditional joint particle filter. However, an MCMC sampling particle filter can be used here to further speed up tracking. For all three methods, we used the same likelihood function, which is similar to that used in [18]. The maximum number of particles used in the entire scene at every time instant was kept the same for all three methods. Furthermore this upper bound on the number of particles was kept low despite the large number of people in the scene, to ensure a low computational cost, essential for any real time surveillance system.

Figure 3 shows tracking results obtained by our particle filtering framework for sample frames from Camera 2. The maximum number of particles was set to 2000 per frame. The top row shows the tracked targets with their IDs, while the bottom row shows the interaction groups formed at the corresponding time instants and the number of particles assigned to each group. Given a small set of particles, our results show the importance of wisely allocating more particles to those groups, whose targets are more prone to be tracked erroneously. The frame in the left column shows a group with a larger number of people (Ids 1,2,3,4) being assigned more particles than smaller groups. In the middle frame, the group of two persons (Ids 1,2) gets assigned a large number of particles per person as compared to the group of 4 persons, due to the high appearance similarity between the targets. The frame in the right column shows an example of a group of two highly occluding targets (Ids 3,4) getting assigned a larger number of particles as compared to a group with three targets (Ids 1,2,5). The other particle filtering frameworks that we compare to, treat all targets equally. Thus, resources are wasted in some parts of the scene, while they are insufficient for other parts. This leads to more tracking errors.

Figure 4 shows the histogram of the size of the largest group in every frame for videos from both cameras. For camera 1, the largest group has size 3. For camera 2, the largest group has size 4 although up to 9 persons are simultaneously present in the scene. Each group in the scene is tracked using a particle filter with a corresponding state space. Thus, the size of the largest group determines the maximum dimensionality of all these state spaces. Note that for a majority of frames in both cameras, groups of at most size 2 are present, which restricts the state spaces to a low dimensionality. Thus, even small sets of particles are able

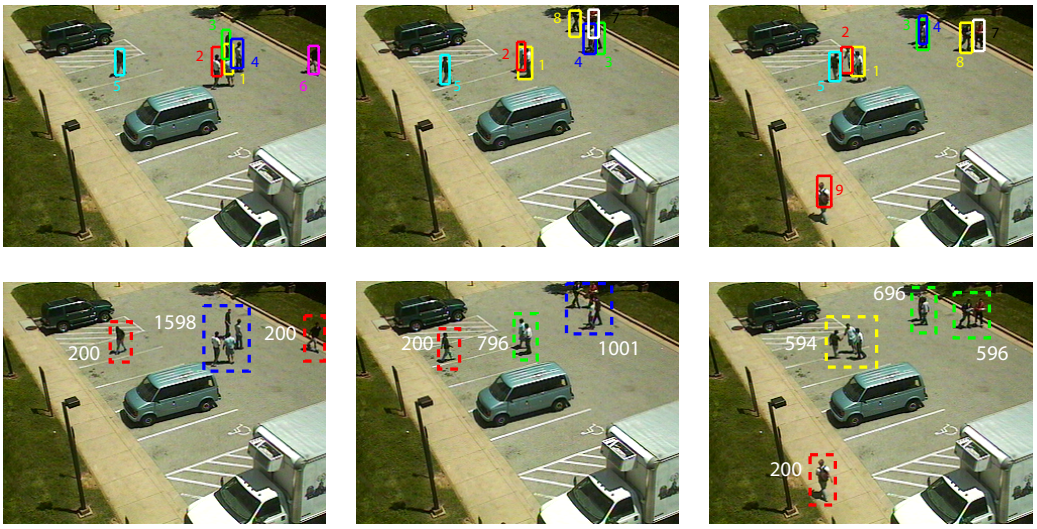

Figure 3. Tracking results for Camera 2 shown for three frames. The bottom row shows interaction groups formed by our framework. Groups of the same size are marked with the same color. The number of particles assigned to each group is also displayed. (See text for detailed discussion.)

to densely sample these spaces and provide good results.

We measure our tracking accuracy by comparing the predicted bounding boxes to manually marked ground truth locations for all targets. The tracking error for each target is defined as the degree of overlap (Θ) between the predicted bounding box (B_p) and ground truthed bounding box (B_g).

$$\Theta = \frac{Area(B_p \cap B_g)}{Area(B_p)} + \frac{Area(B_p \cap B_g)}{Area(B_g)} \qquad (11)$$

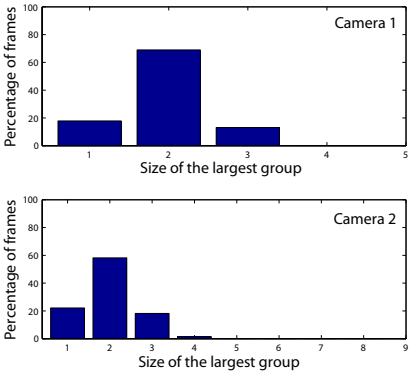

Figure 4. Histogram of the size of the largest group in every frame of the video. The size of the largest group determines the maximum dimensionality of the state space.

Large values of Θ indicate a more precise tracking result. When Θ falls below a threshold Θ_{thr} for a target, we manually reinitialize the bounding box for that target and resume tracking. The length of a track is then defined as the number of frames between two reinitializations. Figure 5 shows tracking results for both video sequences. The average track length is plotted against Θ_{thr}. Clearly, longer track lengths indicate a better system performance. Larger the value of Θ_{thr}, lower the error tolerance. This leads to frequent reinitializations giving shorter tracks.

Figure 5(a) shows results for Camera 1. The maximum number of particles is set to 500 per frame. The independent particle filter has many hijackings that take place and needs to be reinitialized often. The joint particle filter with MCMC sampling also shows a poor performance, as compared to our tracking framework using resource allocation. Figure 5(b) shows results for Camera 2. Here we use a total of 2000 particles per frame. We notice a considerable gain in performance which is very encouraging given that video sequences from Camera 2 are very challenging.

The independent particle filter gives good results when targets are far away from each other, but suffers from hijacking when interactions take place. The joint filter with MCMC shows poor results because the small set of particles is clearly insufficient for the high dimensional state spaces that need to be sampled, when a large number of people are simultaneously present in the scene. Given the same number of resources, our method clearly outperforms the other two, demonstrating the importance of resource allocation.

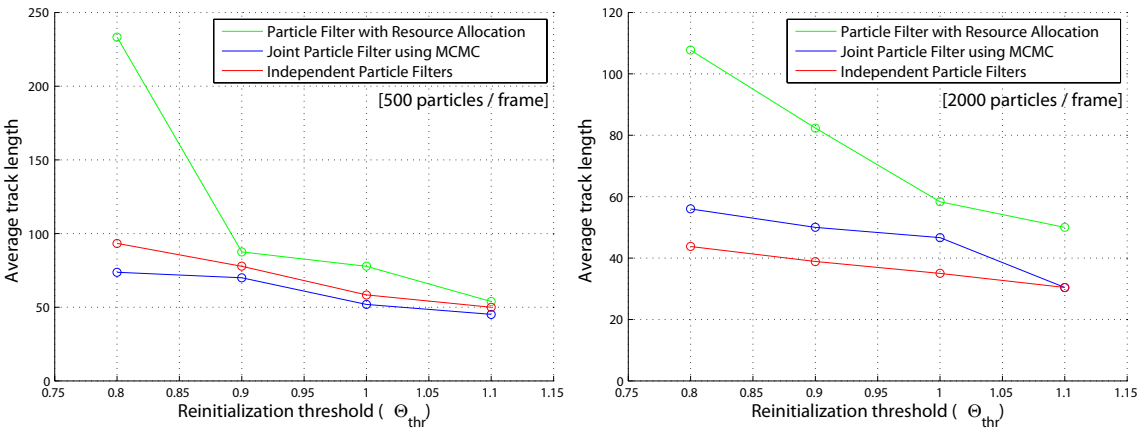

Figure 5. Tracking results for video sequences from Cameras 1 (frames shown in Figure 1) and 2 (frames shown in Figure 3). The system is manually reinitialized when the degree of overlap between the predicted and ground truthed bounding boxes goes below a threshold Θ_{thr}. The average length of tracks (measured as the number of frames between two reinitializations) is plotted against Θ_{thr}. Our system clearly outperforms the other two tracking methods. (See text for discussion.)

6 Conclusions

We present a particle filtering framework that uses an intelligent resource allocation scheme to track a large number of targets using a small set of particles. The number of particles assigned to each target depends the number of targets it is interacting with, the proximity and appearance models of the interacting targets and the tracking confidence at the previous time instant. We demonstrate the advantages of our method on sequences from two surveillance cameras and compare it to commonly used tracking frameworks.

7 Acknowledgements

This research was funded in part by the U.S. Government VACE program. W. R. Schwartz acknowledges Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior (CAPES - Brazil, grant BEX1673/04-1). The authors also thank Ryan Farrell and Vlad Morariu for useful discussions.

References

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. Signal Proc.*, (2), 2002.

[2] B. Brandao, J. Wainer, and S. Goldenstein. Subspace hierarchical particle filter. In *SIBGRAPI*, pages 194–204, 2006.

[3] T. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE PAMI*, 8, 1986.

[4] J. Dowdall, I. Pavlidis, and P. Tsiamyrtzis. Coalitional tracking in facial infrared imaging and beyond. In *CVPR Workshop*, 2006.

[5] A. Gupta, A. Mittal, and L. Davis. Cost: An approach for camera selection and multi-object inference ordering in dynamic scenes. In *ICCV*, 2007.

[6] B. Han, S. Joo, and L. Davis. Probabilistic fusion tracking using mixture kernel-based bayesian filtering. In *ICCV*, 07.

[7] B. Han, Y. Zhu, D. Comaniciu, and L. Davis. Kernel-based bayesian filtering for object tracking. In *CVPR*, 2005.

[8] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *IJCV*, 1998.

[9] M. Isard and J. MacCormick. Bramble: a bayesian multiple-blob tracker. *ICCV*, pages 34–41, 2001.

[10] Z. Khan, T. Balch, and F. Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV*, 2004.

[11] Z. Khan, T. Balch, and F. Dellaert. A rao-blackwellized particle filter for eigentracking, 2004.

[12] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple targets. In *ICCV*, 1999.

[13] F. Porikli. Integral histogram: A fastway to extract histograms in cartesian spaces. In *CVPR*, 2005.

[14] A. Sankaranarayanan, A. Srivastava, and R. Chellappa. Algorithmic and architectural optimizations for computationally efficient particle filtering. *IEEE Trans. Image Proc.*, 08.

[15] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. In *ICCV*, 2005.

[16] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[17] T. Yu and Y. Wu. Decentralized multiple target tracking using netted collaborative autonomous trackers. In *CVPR*, 2005.

[18] T. Zhao and R. Nevatia. Tracking multiple humans in crowded environment. *CVPR*, pages 406–413, 2004.

[19] S. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Trans. Image Proc.*, 2004.