Dynamic Task Decomposition for Probabilistic Tracking in Complex Scenes

Tao Hu[1,2], Stefano Messelodi[1], Oswald Lanz[1]
[1]Fondazione Bruno Kessler, Trento, Italy [2]ICT Doctoral School, University of Trento, Italy
Email: {hutao,messelod,lanz}@fbk.eu

***Abstract*—The employment of visual sensor networks in surveillance systems has brought in as many challenges as advantages. While the integration of multiple cameras into a network has the potential advantage of fusing complementary observations from sensors and enlarging visual coverage, it also increases the complexity of tracking tasks and poses challenges to system scalability. A key approach to tackling these challenges is the mapping of the demanding global task onto a distributed sensing and processing infrastructure. In this paper, we present an efficient and scalable multi-camera multi-people tracking system with a three-layer architecture, in which we formulate the overall task (i.e. tracking all people using all available cameras) as a vision based state estimation problem and aim to maximize utility and sharing of available sensing and processing resources. By exploiting the geometric relations between sensing geometry and people's positions, our method is able to dynamically and adaptively partition the overall task into a number of nearly independent subtasks, each of which tracks a subset of people with a subset of cameras. The method hereby reduces task complexity dramatically and helps to boost parallelization and maximize the real-time throughput and available resources of the system while accounting for intrinsic uncertainty induced, e.g., by visual clutter, occlusion, and illumination changes. We demonstrate the efficiency of our method by testing it with a challenging video sequence.**

Keywords: multi-camera tracking, object tracking, distributed tracking, resource allocation, task assignment

I. Introduction

Visual tracking has been a vigorous research topic in the computer vision domain, due to its widespread applications in such fields as ambient assisted living, sports analysis, traffic control, urban surveillance etc.. Nowadays visual surveillance often involves monitoring large, open areas (e.g., airports) with multiple networked cameras. The employment of visual sensor networks in surveillance systems has brought in as many challenges as advantages. While the integration of multiple cameras into a network has the potential advantage of fusing complementary observations from sensors and enlarging visual coverage, it also increases the complexity of tracking tasks and poses challenges to system scalability. Traditional methods of tracking all the targets with a joint likelihood can easily incur the curse of dimensionality as the number of targets increases. On the other hand, tracking with totally independent particle filters often results in the problem of hijacking [1]. Another issue arises from the enlarged area to be monitored and the increased number of views to be processed. Larger area means more computational resources to be allocated for the detection process and more views means more data to be processed, hence engendering more computational overhead. How to curtail the computational load so as to maintain real-time tracking without losing frames is then a big issue. Besides, camera networks usually have limited resources such as communication bandwidth and sensors typically have limited or even no computational capabilities, the way of information gathering, sharing and processing among cameras is then of crucial importance. A possible approach to tackling these challenges would then be to dynamically map the demanding global task onto a decentralized or distributed sensing and processing infrastructure. In this paper, we demonstrate that the overall tracking task can be split into nearly independent subtasks corresponding to the tracking of subsets of people, thereby reducing computational load and saving communication bandwidth due to branched image transfer. Our dynamic task decomposition strategy brings out a three-layer architecture for the tracking system (Fig. 1), in which cameras constitute the bottom layer and are dynamically grouped into clusters (or agencies, the middle layer) which track a subset of targets. The top layer is a supervisor process which takes care of target detection and release, and task decomposition. For real-time consideration, we design a simple but effective cost function which measures the cost of assigning a subset of targets to an agency, and propose a method for dynamic task decomposition.

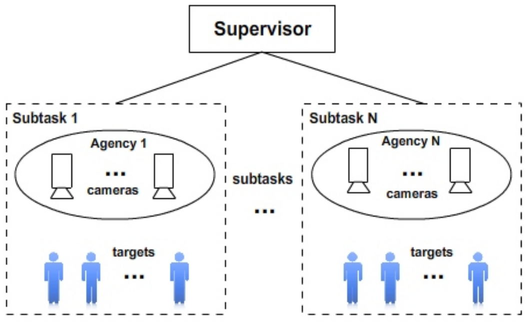

Fig. 1: The three-layer architecture of our tracking system.

The remainder of the paper is structured as follows. Section 2 covers some state-of-the-art work to our interest. Section 3 presents how we formulate the problem. Section 4 details how we implement dynamic task decomposition. Experimental results and conclusions are given in Section 5 and 6 respectively.

II. Related work

In the context of multi-camera multi-object tracking with overlapping fields of view (FOVs), a number of issues have to be considered in order to achieve reliable real-time tracking, balance system computational load and data flow, and thus enhance system scalability.

One of these issues is how to minimize computational cost and circumvent the curse of dimensionality. A possible strategy is to reduce observations for evaluation, since observation and likelihood computation is the most costly part during tracking. Tessens et al. [9] demonstrate that not all cameras are equally needed for tracking a given target in a distributed smart camera network and propose to select a subset of cameras for tracking each target independently. The selection of the subsets is done by computing a suitability value which is based on the Dempster-Shafer (DS) theory of evidence. Song et al. [18] propose to divide the particle set into two subsets: one with high-quality particles and the other with low-quality particles. Observations are only performed for the high-quality subset. Instead of dividing particles, in [15] cameras are dynamically divided into three subsets: active, passive and inactive. Active cameras are used for tracking, while passive cameras are not actually used for tracking but available for evaluation of states returned by active cameras, and inactive cameras are either disabled due to power saving or do not have the targets in view.

Another issue is how to circumvent the curse of dimensionality. Kembhavi et al. [14] propose to divide targets into interaction groups according to a similarity score and perform tracking with a joint filter for each group of targets. Lanz [25] proposes a hybrid joint-separable filter, an approximate Bayesian tracker that propagates independent representations for each target with a joint likelihood to manage occlusions, which has a computational complexity that grows quadratically with the number of targets. Our proposed method is similar to [14], however, we differ in that:(1) our approach is decentralized; (2) we divide targets into groups according to occlusion constraints; (3) we use an HJS filter [16] to track each group of targets.

A third issue is how to allocate and manage resources (e.g., computation load, system bandwidth). This is typically done by best camera(s) assignment approaches [6, 5, 7], i.e., assigning the best camera(s) for each target. Li and Bhanu [6] propose to use game theory and bargaining mechanisms to perform camera assignment and hand-off in a VSN. Cameras act as players and bargain with each other to reach an optimal solution which maximizes the global utility given a number of criteria. Similarly, Esterle et al. [3] introduce a market mechanism to address object handover during tracking. The camera assignment problem can also be formulated as a Constraint Satisfaction Problem (CSP) [7] or a distributed CSP [4] or even a Stable Marriage Problem [10]. A comparison of methods for camera selection and handoff can be found in [8]. Different efficiency measures have been proposed, such as the camera utility measures [6], trackability measure [19] and suitability value [9]. In this paper, we design a simple but effective cost function which measures the cost of assigning an agency to a subset of targets. Different from the aforementioned measures which mainly focus on the FOVs of cameras and the size of the target in the image plane, our cost function takes into account occlusions between targets and the computational overhead of the agency.

Besides the above issues, how to fuse the estimates among cameras or surveillance clusters is of equal importance. As the monitored scene gets larger and the number of cameras increases, the tracking system has to be distributed for the sake of scalability. In distributed tracking systems, usually cameras track independently and data fusion can be done by in-network aggregation [12], weight average [2] or Kalman-consensus filtering [13]. Although distributed tracking algorithms use less amount of data at every individual node, decentralized algorithms can share larger amount of data among view clusters and transmit estimates among fusion centers through a centralized processor to validate the estimates in return. Typically decentralized frameworks form cameras into clusters [17] or hierarchical Fault Containment Units [11], which are managed by a base station. For a comparison of distributed and decentralized trackers, please refer to [21] as a survey. The authors implement some representative decentralized and distributed algorithms for tracking with multiple overlapping cameras and multiple targets. Experimental results show the overall performance favors decentralized algorithms due to fewer fusion centers compared to distributed algorithms. In this paper, we also adopt a decentralized framework.

III. Problem formulation

The application scenario addressed in this paper suggests the choice of a probabilistic framework, since measurements may convey intrinsic uncertainty which cannot be eliminated due to occlusion, target similarity and background clutter. Besides, for efficiency reasons, illumination effects (e.g. shadows) are neglected and only coarse shape and appearance models can be used, leading to a noisy measurement process. Estimates are therefore inherently inaccurate and a deterministic framework (such as Mean-shift [22], graph-matching [23]) would not adequately account for this.

A. Sequential Bayesian framework

In this paper, object tracking is interpreted as a state estimation problem. Basically, the dynamic components of interest of the monitored environment are described with a vector x of numbers, the *state*, which is evolving in continuous time and observed at discrete times t through a measurement vector z_t. More specifically, in this paper the state is composed of the target's 2D position and velocity w.r.t. the floor plane. The aim is then to estimate its posterior distribution, or *belief*, $p(x_t|z_{1:t})$ at t conditioned on a sequence of observations $z_{1:t}$ obtained up to t. In order to support sequential estimation imposed by real time applications, signal x_t is modeled as a first order Markov process, and observations $z_{1:t}$ are assumed to be conditionally independent given a sequence of states $x_{1:t}$. This enables us to compute a new estimate $p(x_t|z_{1:t})$ solely from the actual observation z_t and its previous estimate $p(x_{t-1}|z_{1:t-1})$ using stochastic propagation and Bayes law

$$p(x_t|z_{1:t}) \propto p(z_t|x_t) \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1}. \quad (1)$$

Eq. 1 provides a widespread algorithmic framework called *Sequential Bayes Filter* built upon a probabilistic model which is fully described by an initial distribution $p(x_0)$, the dynamical model $p(x_t|x_{t-1})$ and the observation model $p(z_t|x_t)$.

B. *Distributed estimation via dynamic factorization*

The task of monitoring a large open environment may be formalized as a single, global state estimation problem. A Bayes filter can then estimate the *joint* configuration x_t of all targets (i.e. the vector containing one component for each target) with the set z_t of images captured by all the cameras at the same time (for unsynchronized streams z_t may contain a single image). It is easy to imagine that applying this *joint filter* to real time monitoring may be unaffordable. A more feasible solution would be to instantiate several sub-filters with reduced, local competence which together may perform more efficiently in terms of computational complexity, load balance on a processor web and data flow management. From a theoretical point of view it is easy to show that such a choice is certainly justified when dynamical and observation models are both separable. While assuming independence among the motion patterns of targets may be acceptable at least for short predictions (the Bayes filter is first order in time), this is certainly not true for the observation model when people arrange in groups causing frequent and persistent occlusions. Under occlusion, the appearance of one target cannot but be explained in relation to that of the occluder, which requires therefore a joint analysis of captured images. On the other hand, the same scene may be seen occlusion-free from another viewpoint where indeed the resulting image can be elaborated independently for locating each target, thus, at lower computational cost and in parallel. This is the key observation leading to our task decomposition algorithm described in the next section: given known view geometry and the targets' estimated positions, we determine camera-target associations that maximize parallelization while guaranteeing consistent occlusion handling.

IV. Task decomposition

A key issue in monitoring large open areas is the mapping of the demanding global task onto a distributed sensing and processing infrastructure [24]. As mentioned in the previous section, a principled approach may be derived by exploring dependency relations among environment dynamics (target trajectories) and measurements (utilized cameras). While such dependencies may appear in many different facets (group behavior, illumination effects such as shadows), the most essential one arises from occlusions. The approach proposed next accounts for this, providing a solution derived from knowledge about camera placement and target positions under given constraints.

A. *Task decomposition: an example*

An example is shown in Fig. 2: tracking four targets with three cameras. Suppose all four targets are visible in all three cameras, a trivial solution is to instantiate a single task with all the three cameras tracking all the four targets jointly. However, if we have two processing units, we can decompose the task into two subtasks with reduced complexity and implement them on the two processing units in parallel. Furthermore, closeness of targets B, C and D suggests that they should be grouped together, whereas target A might be handled separately. This can be derived from occlusion reasoning. The closeness of targets B, C and D indicates potential occlusions among them in the three views within a certain time interval,

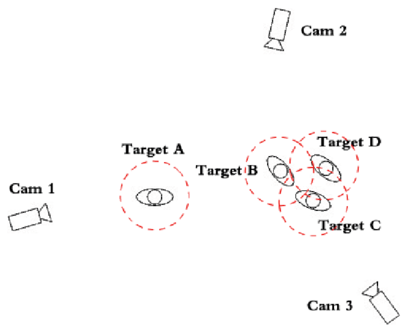

Fig. 2: Tracking four targets with three cameras: how to optimally assign cameras, targets to independent tracking processes? The red dashed circles centered around the targets are the predicted supports which define all possible positions each target can reach within a short time period Δt.

as can be seen from the intersecting red dashed circles. Due to the potential occlusions, targets B, C and D should be tracked together. Suppose we allocate agency A_1 to track A and agency A_2 to track (B,C,D), then in the camera views of A_1 target A must not be occluded by (B,C,D), and in the camera views of A_2 (B,C,D) must not be occluded by A. Cam 1 observes (B,C,D) occluded by A and can therefore not be used to track (B,C,D) only (i.e., if Cam 1 is used to track (B,C,D), it must also track A at the same time.). On the other hand, Cam 1 could take care of tracking A alone because (B,C,D) do not influence the appearance of A in Cam 1 although they are also visible[1]. Cam 2 and Cam 3 could all be associated to both (A) and (B,C,D) as there are no occlusions among them. An appropriate choice would then be to instantiate two subtasks:

$$\text{Sub}_1 = \big\{(A),(1,2,3)\big\}; \qquad \text{Sub}_2 = \big\{(B,C,D),(2,3)\big\}.$$

This solution does not consider constraints about resources. In practice, we may have requirements about (1) the minimum number of cameras in an agency to ensure tracking quality; (2) maximum number of cameras in an agency for computational overhead consideration; (3) maximum number of agencies (which must not be larger than the number of targets) to ensure there are enough available processing units for every subtask to be mapped into. Besides, this solution is only the snapshot decomposition in Fig. 2. Task decomposition should not be carried out with a frequency comparable to the observation rate. A lower rate may be employed while accounting for potential occlusions within a given time interval. To sum up, a task decomposition algorithm must be able to: (1) detect potential interactions (i.e. occlusions) among targets , and (2) reason about efficiency (or cost) and suitability of a possible decomposition, which will be detailed in the next sections.

B. *Occlusion reasoning*

As outlined in Sec. III-B, the allocation of a set of subtasks remains sustainable as long as the measurement model remains separable over the support of estimated beliefs. Verifying this at full resolution is not required: we may limit ourselves to

[1]We extract color information of the target by projecting a 3D shape model to the image plane as in [16]. The color information can be extracted for likelihood evaluation as long as the target is not occluded in the camera view.

detect potential occlusions while reasoning on a horizontal reference plane, with a discrete loss function which takes value 1 where a potential occlusion is detected and value 0 elsewhere.

To further reduce complexity, the prior belief utilized is taken uniform on a circular support centered around the estimated target position (an MAP estimate over the belief, or simply its expectation). Its predicted support is then obtained as the envelope of all the positions that can be reached within a time span at a predefined maximal velocity. This corresponds to drawing a circle around the current position with a radius R equal to the product of the maximum velocity and a time span (the red dashed circles in Fig. 2):

$$R = V(\Delta T + \tau). \tag{2}$$

where V is the predefined velocity, ΔT is a fixed short time interval, and τ is the estimated time needed for task decomposition. While V and ΔT can be defined a priori, τ can be computed as the average time for task decomposition for different number of targets in the scene. To detect potential occlusion in each view, we draw a circle around the camera viewpoint and project the support outline of each target to the circle. If the projection of outlines have intersection, then occlusion is detected (Fig. 3).

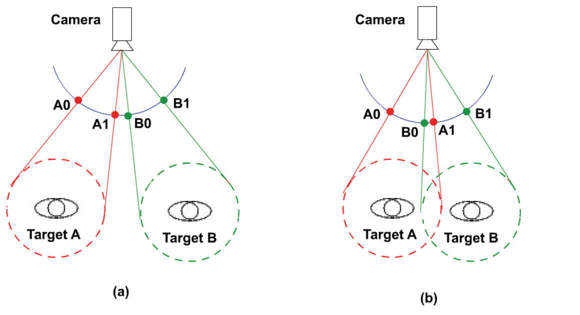

Fig. 3: Potential occlusion detection. $\overset{\frown}{A_0A_1}$ and $\overset{\frown}{B_0B_1}$ are the projections of the support outlines of targets A and B to the circle centered at the viewpoint of the camera. (a) No occlusion between A and B when $\overset{\frown}{A_0A_1}$ and $\overset{\frown}{B_0B_1}$ have no intersection. (b) Occlusion is detected when $\overset{\frown}{A_0A_1}$ and $\overset{\frown}{B_0B_1}$ have intersection.

C. Cost measure for task decomposition

Since in most cases there are many feasible solutions for task decomposition even given a number of constraints, we need to quantify the cost for each decomposition. The optimal solution is the one with the lowest cost.

Efficient task decomposition relies on the choice of an appropriate function to quantify the cost. It should account for computational load as well as for sensing and networking issues. When splitting a joint filter into agencies with competence on a subset of K_i targets, the computational complexity reduces from $e^{\sum \kappa_i}$ to $\sum e^{K_i}$ (in case of the more efficient HJS filter [16] from $(\sum K_i)^2$ to $\sum K_i^2$). Another factor that influences performance is the number and spatial distribution of attached sensors. A higher number guarantees better visual coverage, but at the cost of higher observation overhead, power

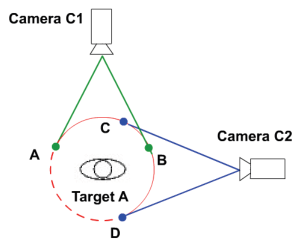

Fig. 4: The coverage rate of a target.

consumption (camera cannot be switched off) and networking load (image transfer). To measure agency utility, we compute the overall coverage rate of the support outlines of all the targets in the agency. Fig. 4 shows an example how the coverage rate is computed. In the example, the coverage of target T is $C_{ov}(T) = L_{\overset{\frown}{ACBD}}/C$, where $L_{\overset{\frown}{ACBD}}$ is the length of the arc $\overset{\frown}{ACBD}$ and C is the circumference of the support outline. Although simple, the coverage rate is able to reflect the common sense that the utility depends on the relative positions of the cameras. Here we do not design a very sophisticated and impeccable utility measure because it is not worth the efforts and the computational cost is a major concern. With all these factors considered, a suitable cost function to be minimized is

$$\sum_i n_t^2(\mathcal{A}_i) + \alpha \sum_j (1 - \mathcal{C}_{ov}(T_j)) \tag{3}$$

where n_t counts the number of targets of agency $\mathcal{A}_i$, and parameter α modulates the importance of agency utility. Although simple, Eq. 3 already considers the most important aspects of task distribution over a complex infrastructure. With this cost measure, we can dynamically partition the cameras and targets and select the best subtask configuration that results in the least cost.

D. Dynamic task decomposition

Decomposing the overall task is basically to find the optimal configuration of subtasks under given constraints. Let's denote the number of cameras by N_c, the number of targets by N_t, the maximum number of agencies by MAX_a, the minimum and maximum number of cameras each agency can have by MIN_c and MAX_c, the maximum number of agencies each camera can be assigned to by M_i. The idea is first to partition the camera set $\{C_{11}...C_{1M_1}, ..., C_{i1}...C_{iM_i}, ..., C_{N_c1}...C_{N_cM_{N_c}}\}$ to produce agencies (subsets of cameras), and partition the target set $\{T_1, T_2...T_{N_t}\}$ to produce groups of targets, where $C_{i1}...C_{iM_i}$ are multiple copies of camera C_i and M_i is the maximum number of agencies camera C_i can be assigned to (we introduce multiple copies to allow each camera to be assigned to more than one agency). Camera partitions that do not meet the constraints of MAX_a, MIN_c and MAX_c should be precluded. In this way, we get a camera partition set S_c and a target partition set S_t. Each camera partition contains a set of agencies and each target partition contains a set of target groups. Then for each camera partition and each target partition we enumerate all the possible configurations of subtasks (combinations of agencies and target groups) and check whether they meet our required occlusion constraint (i.e., the targets tracked in one agency must not be occluded by targets tracked in another agency). If so, compute the cost according to Eq. 3. Finally the configuration with the least cost is selected. In practice, the partitions of cameras and targets as well as the combinations of agencies and target groups can be computed offline and stored in a lookup table. A piece of pseudo code is shown in Algorithm.1.

Algorithm 1 Dynamic task decomposition

```
compute camera partition set $S_C$;
compute target partition set $S_T$;
$cost_{min} = MAX\_DOUBLE; config_{best}$; //$config_{best}$ is the best configuration
for each camera partition $PC_i \in S_C$ do
  for each target partition $PT_j \in S_T$ do
    if $PC_i.size == PT_j.size$ then
      generate all possible configurations ($S_{config}$) of subtasks (combinations of camera subsets and target subsets ) ($SubC_k, SubT_l$); where $SubC_k \in PC_i, SubT_l \in PT_j$;
      for each configuration $conf_m \in S_{config}$ do
        if Occlusion constraint is met then
          compute the cost $cost_m$;
          if $cost_m < cost_{min}$ then
            $cost_{min} = cost_m$;
            $config_{best} = conf_m$;
          end if
        end if
      end for
    end if
  end for
end for
```

V. Experiments and results

We tested our algorithm with a challenging sequence taken in our lab, where four cameras are installed at the corners. The sequence is about 3.5 minutes long captured at 15 Hz with a resolution of 640×480 pixels . Through the sequence, people enter, walk around, sit down and exit the room randomly, causing frequent occlusions. The maximum number of people in the scene at the same time is 7 and the ground truth was done by manual labeling. The sequence together with ground truth and calibration files are available at: http://tev.fbk.eu/databases/lab.html.

In the experiment, we imposed the following constraints: $MAX_a = 3; MIN_c = 2; MAX_c = 3; M_i = 2, 1 \leq i \leq 4$ (this enforces that each camera can be assigned to maximum 2 agencies). We set V to 1.5m/s and ΔT to 0.2s in Eq. 2. The time used for task decomposition depends on the nummber of cameras and the number of targets. In the case of 4 cameras and 7 targets, it is negligible. So $R \approx 1.5 * 0.2 = 0.3m$. The choice of R influences the performance. A larger R makes the task less decomposable (e.g., when R becomes infinity, the task cannot be decomposed because of potential occlusions among all targets so the tracker turns out to be centralized), and a smaller R means more frequent decomposition which may compromise the performance since target handover takes time. The initialization of the tracker was the same as in [20]. We verified the efficiency of our approach by measuring the MOTA/MOTP (Multiple Object Tracking Accuracy/Precision) metrics as proposed in [26], which provides a systematic evaluation to compare the performance of different trackers in terms of tracking accuracy and consistent labeling. We compared the proposed approach with a joint HJS tracker proposed in [16]. Table 1 and 2 show the evaluation results with 5 runs. As can be seen, the average MOTA score of the proposed approach has over 10% gain compared to the centralized approach. This is because the centralized tracker lost more frames during tracking. As the number of targets increases, the centralized tracker may not be able to handle the computational burden and has to lose frames, thus producing unreliable tracking results (the MOTA score is very low sometimes as can be seen in the table). However, the proposed approach is able to decompose the task into a number of subtasks (maximum 3) and implement them on different processors in parallel. Therefore, the evaluation scores are very stable. The centralized tracker has a better MOTP score because it uses 4 cameras for each target, while the decentralized one use at most 3 cameras for each target. Fig. 5 shows a map of the environment and a screen shot of the tracker. At the specific instant, there are 7 targets in the scene ($T_1 \sim T_7$, the supports of which are denoted by the circles in the map). The tracker instantiates 3 agencies: A_1 ={Cam 1, Cam 2}, A_2 ={Cam 2, Cam 3}, A_3 ={Cam 1, Cam 3, Cam 4}, which tracks the target groups G_1 ={1}, G_2 ={2, 3, 4, 5}, G_3 ={6, 7} respectively. We can see this decomposition is reasonable from the geometric relations of the cameras and the estimated positions of the targets on the map.

TABLE I: Evaluation of the centralized HJS tracker.

Run	MOTP	miss	fp	mismatches	MOTA
1	61mm	17.4%	2.3%	6	80.3%
2	70mm	21.4%	13.2%	7	65.4%
3	59mm	20.1%	2.1%	12	77.7%
4	59mm	24.4%	8.9%	8	66.6%
5	67mm	17.3%	3.5%	6	79.2%
Average	63.2	20.12%	6.0%	7.8	73.84%

TABLE II: Evaluation of the proposed decentralized tracker.

Run	MOTP	miss	fp	mismatches	MOTA
1	85mm	8.9%	2.1%	4	88.9%
2	84mm	10.0%	3.3%	5	86.7%
3	83mm	8.7%	3.6%	2	87.7%
4	93mm	12.8%	2.8%	16	84.3%
5	89mm	9.1%	3.5%	5	87.4%
Average	86.8	9.9%	3.06%	6.4	87.0%

VI. Conclusion

Visual tracking is challenging in large complex scenes where there are many people occluding each other. A major issue is how to maintain reliable real-time tracking, balance system computational load and data flow, and thus enhance system scalability. In this paper, we proposed a decentralized multi-target multi-object tracking system, which has a three-layer architecture. With the proposed cost measure, the approach is able to dynamically decompose the overall task into a number of nearly independent subtasks, each of which tracks a subset of targets with an agency. The association of agencies and groups of people is based on sensing geometry and the estimated target positions. Experimental results demonstrate that the method is able to reduce task complexity and helps to

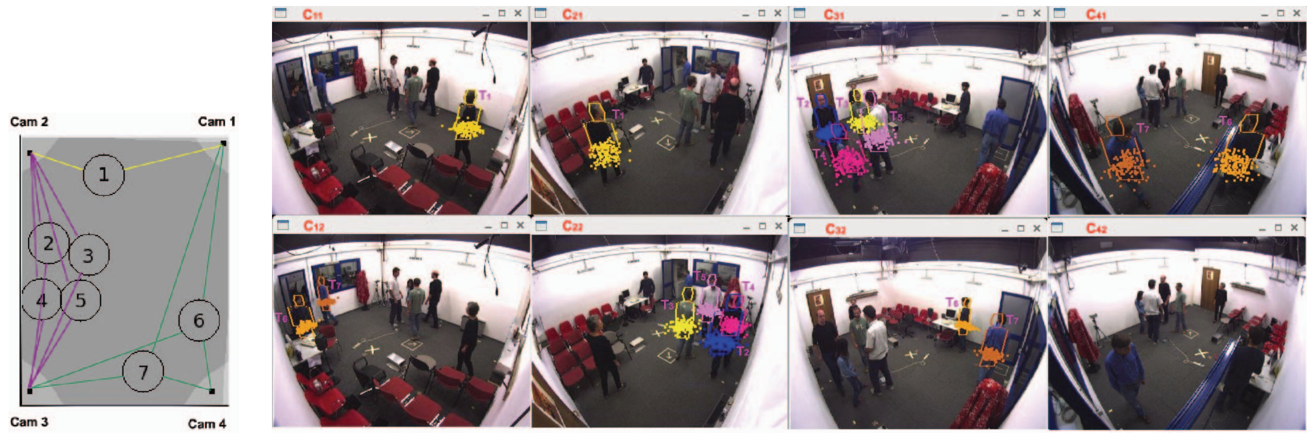

Fig. 5: A screenshot of the tracker. The left is the map. On the right there are 8 camera views, with the second row being a copy of the first row (i.e., C_{12} and C_{11} both refer to Cam 1.). The lines connecting a target to cameras indicate the target is tracked by the corresponding cameras. The dots around the targets are particles (hypotheses).

improve real-time tracking performance. Future work will be focused on how to dynamically decompose the task in a more efficient way.

References

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

[2] F. Rezaei and B. H. Khalaj: Distibuted human tracking in smart camera networks by adaptive particle filtering and data fusion. ICDSC 2012.

[3] L. Esterle, P. Lewis, M. Bogdanski, B. Rinner, and X. Yao: A socio-economic approach to online vision graph generation and handover in distributed smart camera networks. ICDSC 2011.

[4] M. Bramberger, B. Rinner, H. Schwabach: A method for dynamic allocation of tasks in clusters of embedded smart cameras. IEEE International Conference on Systems, Man and Cybernetics, vol.3, pp.2595-2600 2005'

[5] Y. Li and B. Bhanu: Task-oriented camera assignment in a video network. ICIP 2009.

[6] Y. Li and B. Bhanu: Utility-based dynamic camera assignment and hand-off in a video network. ICDSC 2008.

[7] F.Z. Qureshi, D.Terzopoulos: Multi-camera Control through Constraint Satisfaction for Persistent Surveillance. IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance, pp.211,218, 1-3 Sept. 2008.

[8] Y. Li and B. Bhanu: A comparison of techniques for camera selection and handoff in a video network. Third ACM/IEEE International Conference on Distributed Smart Cameras, 2009.

[9] L. Tessens, M. Morbee, H. Aghajan, W. Philips: Camera Selection for Tracking in Distributed Smart Camera Networks. In ACM Transactions on Sensor Networks, vol. 10, no. 2, 2014.

[10] Cenedese, A.; Cerruti, F.; Fabbro, M.; Masiero, C.; Schenato, L.: Decentralized task assignment in camera networks. 2010 49th IEEE Conference on Decision and Control, pp.126,131, 15-17 Dec. 2010

[11] D.R. Karuppiah, R.A. Grupen, z. Zhu, and A.R. Hanson: Automatic resource allocation in a distributed camera network. Machine Vision and Applications, vol. 21, pp. 517-528, 2010

[12] Manish Kushwaha, Xenofon D. Koutsoukos. 3D target tracking in distributed smart camera networks with in-network aggregation. Fourth ACM/IEEE International Conference on Distributed Smart Cameras, 2010. P. 25-32.

[13] C. Soto, B. Song, A.K. Roy-Chowhury: Distributed multi-target tracking in a self-configuring camera network. IE Conference on Computer Vision and Pattern Recognition, 2009.

[14] A. Kembhavi, W. Schwartz, and L. Davis: Resource allocation for tracking multiple targets using particle filters. Eight International Workshop on Visual Surveillance (VS2008), 2008.

[15] S. Spurlock, R. Souvenir: Dynamic subset selection for multi-camera tracking. ACM-SE '12 Proceedings of the 50th Annual Southeast Regional Conference.

[16] O. Lanz: Approximate Bayesian multi-body tracking. In IEEE Trans. Pattern Analysis and Machine Intelligence,2006.

[17] H. Medeiros, J. Park, and A. KaK: Distributed object tracking using a cluster-based kalman filter in wireless camera networks. IEEE Journal of Selected Topics in Signal Processing, 2(4):448-463, Aug. 2008.

[18] C. Song, J. Son, S. Kwak and B. Han: Dynamic resource allocation by ranking SVM for particle filter tracking. Proceedings of the British Machine Vision Conference, PP. 103.1-103.11. September 2011.

[19] C.H. Chen, Y. Yao, D. Page, B. Abidi, A. Koschan, and M. Abidi: Camera handoff with adaptive resource management for multi-camera multi-object tracking. Image Vision Comput. 28, 6 (June 2010), 851-864.

[20] Lanz, O.; Messelodi, S.: A Sampling Algorithm for Occlusion Robust Multi Target Detection. Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, 346–351, 2-4 (Sept.2009)

[21] M. Taj and A. Cavallaro: Distributed and decentralized multi-camera tracking. IEEE Signal Processing Mag., vol. 28, no. 3, pp. 4658, May 2011.

[22] D. Cornaniciu, V Ramesh and P.Mcer, 2003. Kernel-based object tracking. IEEE Transaction on PaRern Analysis and Machine Intelligence, vol. 25, no. 5, pp. 564-577, 2003

[23] C. Gomila; F. Meyer. Graph-based object tracking. Proceedings. 2003 International Conference on Image Processing, vol.2, no., pp.II,41-4 vol.3, 14-17 Sept. 2003

[24] M. Rosencrantz, G. Gordon and S. Thrun. Decentralized sensor fusion with distributed particle filters. In Uncertainty in Artificial Intelligence (UAI-03). Morgan Kaufmann Publishers, 2003, pp.493-500.

[25] O. Lanz, R. Manduchi. Hybrid joint-separable multibody tracking. CVPR 2005. vol.1, pp. 413,420 vol. 1.

[26] K. Bernardin, R. Stiefelhagen. Evaluating multiple object tracking performance: the CLEAR MOT metrics. J. Image Video Process. 2008(3), 2008.