Dynamic Subset Selection for Multi-Camera Tracking

Scott Spurlock
UNC Charlotte
9201 University City Blvd.
Charlotte, NC 28223
sspurloc@uncc.edu

Richard Souvenir
UNC Charlotte
9201 University City Blvd.
Charlotte, NC 28223
souvenir@uncc.edu

ABSTRACT

While multi-camera methods for object tracking tend to outperform their single-camera counterparts, the data aggregation schemes can introduce new challenges, such as resource management and algorithm complexity. We present a framework for dynamically choosing the best subset of available cameras for tracking in real-time, which reduces aggregate tracking error and resource consumption and can be applied to a variety of existing base tracking models. We demonstrate on challenging video sequences of players in a basketball game. Our method is able to successfully track targets entering and exiting camera views and through occlusions, and overcome instances of single-view tracking drift.

Categories and Subject Descriptors

I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*Video analysis*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Tracking*

General Terms

Algorithms, Experimentation

1. INTRODUCTION

Accurately tracking people in video enables applications in surveillance, traffic monitoring, and video conferencing. Recent multi-camera methods have helped to overcome some of the issues associated with object tracking, such as drift and occlusion, that arise in the single-camera case. However, the integration of multiple cameras introduces new challenges in terms of resource consumption (e.g., power, computing, and networking), and algorithm complexity. Moreover, while tracking accuracy tends to increase with the number of cameras, the potential also increases for poor measurements from individual cameras to negatively affect aggregate tracking estimates.

In this paper, we present a framework for balancing the computational efficiency of single-camera tracking with the power of a distributed camera network by dynamically selecting the best subset of cameras for tracking. This method allows for more efficient use of network and computing resources, and can scale to reduce error with the allocation of additional cameras depending on the desired trade-off between performance and accuracy. In Section 2, we review related work, and our base tracking model is described in Section 3. In Section 4, we present our multi-camera coordination scheme. Results on a complicated dataset are described in Section 5, and we conclude in Section 6.

ACMSE'12, March 29-31, 2012, Tuscaloosa, AL, USA

2. RELATED WORK

Recently developed multi-camera approaches have an improved ability to mitigate mistracking due to occlusion because of the increased likelihood of an available, unoccluded view of the target [16]. Multi-camera methods tend to be one of two types: *measurement correspondence* approaches, where features are warped into a common view prior to state estimation, or *trajectory correspondence* approaches, where state estimates are computed independently in each view before being integrated [15].

Many measurement correspondence approaches make use of plane-to-plane transformations to warp detected objects from multiple camera views to an occupancy map in a reference plane, typically the ground plane [11]. Occlusions of the targets' feet and shadows mistakenly classified as foreground can create problems for methods that use the ground as a reference plane. More recent approaches [12, 5] address these issues by finding the ground plane occupancy using homographies to planes above and parallel to the ground plane. Most of these approaches focus on occupancy rather than using template-based object tracking. This can lead to difficulty in disambiguating nearby targets or discriminitive tracking in crowds. In [7], ground plane foreground occupancy masks are combined with color and motion models, which helps to address these issues.

Compared to measurement correspondence approaches, trajectory corresponce approaches tend to be less resource-intensive and more conducive to distributed processing. One of the early methods [2] used single-camera tracking until the system predicted the camera would no longer have a good view of the target and leveraged epipolar geometry to select a nearby camera to take over tracking. This method differs from ours in that our framework seeks to find the optimal subset of cameras to track the target *in every frame*, rather than simply switching to a different, single camera when necessary. More recent methods have attempted to incorporate the tracking information from all of the cameras in the net-

work. In [14], particle filtering with color-based detection is used to track a target independently in each camera. Their system also uses epipolar geometry to reinitialize a poorly tracking camera from a higher-scoring neighbor. In [17], single-camera detection is used to estimate the 3D target location and these estimates are combined using the Extended Kalman Filter. These methods differ from our proposed framework in that they aggregate information from all available cameras rather than the best subset. In [10], utility functions are defined that model which pair of cameras provides the optimal view of each physical location in the environment. However, the focus is on coverage of physical locations rather than the best view of a particular target, which can lead to problems with multiple occluding targets.

Several hybrid methods have also been proposed that combine the measurement and trajectory correspondence approaches. For example, one method integrates particle filter results from each camera and the ground plane in both the detection and prediction stages by making use of "principal axes" to find reliable target intersection at the ground plane [6]. Han et al. assign a particle filter to each camera and vary the number of particles and relative contribution from each camera using a Gaussian mixture model weighted based on sensor reliability [8]. These hybrid approaches report excellent tracking results; however, like measurement corresponce methods, these come at the price of increased computational and algorithmic complexity.

Our hybrid approach, which is more similar to the trajectory correspondence methods, allows for dynamic camera switching at each timestep and tracking from an arbitrary number of cameras at once. Cases such as targets entering or leaving a camera view, target occlusion, and drift are handled automatically as the framework selects the best cameras to track actively at any given time based on a novel voting and ranking scheme.

3. TRACKING MODEL

In our framework, each camera performs 2D object tracking independently. We chose to implement a particle filter-based tracker, but any number of other tracking methods could have been selected.

3.1 Object Model

Objects are represented as a weighted distribution, q, of color values of the pixels in image patch $J_{\mathbf{y},t} = \{\mathbf{y}' \in I \mid \mathbf{y}' \text{ is bounded by } \mathbf{y} \pm \langle w, h \rangle\}$, where h and w are the half-height and half-width of the bounding box centered at position $\mathbf{y}$, I is the image, and t is the frame number. The distribution, q, is over a quantized RGB color space. In our experiments, we use 16 bins per color channel, which results in a histogram of dimensionality $16 \times 16 \times 16 = 4096$. Each pixel location, $\mathbf{y}'$, is weighted based on:

- **Kernel Function:** Pixels closest to the center of the image patch, $\mathbf{y}$ are weighed more heavily than pixels at the edge using the Epanechnikov kernel [3]:

$$k(\mathbf{y}, \mathbf{y}') = \max\left[0, \frac{3}{4}\left(1 - \left|\left|\frac{\mathbf{y} - \mathbf{y}'}{\langle 2w, 2h \rangle}\right|\right|^2\right)\right] \quad (1)$$

- **Foreground Likelihood:** Pixels are additionally weighted on the likelihood of belonging to the foreground model. Let $B(\mathbf{y}')$ be the background model, represented as a pixel-wise Gaussian distribution with a mean equal to the mode at location $\mathbf{y}'$ in the video sequence, and standard deviation equal to the expected image noise. The foreground likelihood, ϕ, is:

$$\phi(\mathbf{y}') = 1 - P\left[B(\mathbf{y}') = I(\mathbf{y}')\right]$$

The resulting kernel is calculated as $\kappa(\mathbf{y}, \mathbf{y}') = k(\mathbf{y}, \mathbf{y}')\phi(\mathbf{y}')$. We define a function $u = f(\mathbf{y}')$ that maps each pixel from the target image patch into a discrete bin u based on the pixel's location in RGB color space. Taking these elements together, the probability of each feature $u = 1...m$ in the model for the target centered at location $\mathbf{y}$ within the image patch $J_{\mathbf{y},t}$ is given by:

$$q_u(\mathbf{y}) = C \sum_{\mathbf{y}' \in J_{\mathbf{y},t}} \kappa\left(\mathbf{y}, \mathbf{y}'\right) \delta[f(\mathbf{y}') - u] \quad (2)$$

where δ is the Kronecker delta function, and C is a normalization factor.

3.2 Reference Plane Transform

Each camera tracks objects using the coordinate system in a global reference plane. For a calibrated camera, projection matrix P converts a 3D point $\mathbf{X} = (X, Y, Z, W)^T$ to a 2D image location $\mathbf{y} = (x, y, w)^T$ as $\mathbf{y} = P\mathbf{X}$, where $\mathbf{y}$ and $\mathbf{X}$ are expressed as homogeneous coordinates. Denoting the j^{th} column of P as $\mathbf{p}_j$, we can derive a homography, $H_{ref} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_4 + Z_{ref}\mathbf{p}_3 \end{bmatrix}$, which maps points on the image plane to a plane Z_{ref} units above the ground plane along the vertical axis where Z_{ref} is expressed in world coordinates [4]. Ground-plane homographies ($Z_{ref} = 0$) are commonly used (e.g. [11, 7]); however, it has been shown that reference planes above the ground plane can improve tracking accuracy [5]. In our model, we choose Z_{ref} to be half the approximate height of a target. Figure 1(a) illustrates the reference plane used for tracking. Given projection matrix P and the free parameter Z_{ref}, we define the transform, τ, from reference coordinates, $\mathbf{x}$, to image coordinates $\mathbf{y}$:

$$\tau(\mathbf{x}, P) = (H_{ref})^{-1}\mathbf{x}. \quad (3)$$

Dynamic Target Resizing.

We additionally use the reference plane coordinates to resize the target bounding box dynamically during tracking. Figure 1 illustrates this process for two video frames of the same target. Given reference plane coordinates $\mathbf{x} = (x, y)$, we project the 3D points representing the bottom and top of the target, $\begin{bmatrix} x & y & 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} x & y & Z_t & 1 \end{bmatrix}$, respectively, to image coordinates using projection matrix P, where Z_t is the approximate height of the target. The difference gives us the height of the bounding box, in image pixels, and we assume the width to be $\frac{1}{3}$ of this value.

3.3 Particle Filter Tracking

A particle filter [1] estimates a nonparametric posterior distribution of the target object's state represented as a set of particles. We denote the j^{th} particle at time t as

$$s_t^{(j)} = \begin{bmatrix} \mathbf{x} & \dot{\mathbf{x}} \end{bmatrix} \quad (4)$$

where $\mathbf{x}$ and $\dot{\mathbf{x}}$ are the position and velocity, respectively, of the target. At time $t = 0$, the positions of the initial set of particles are manually initialized to the starting location

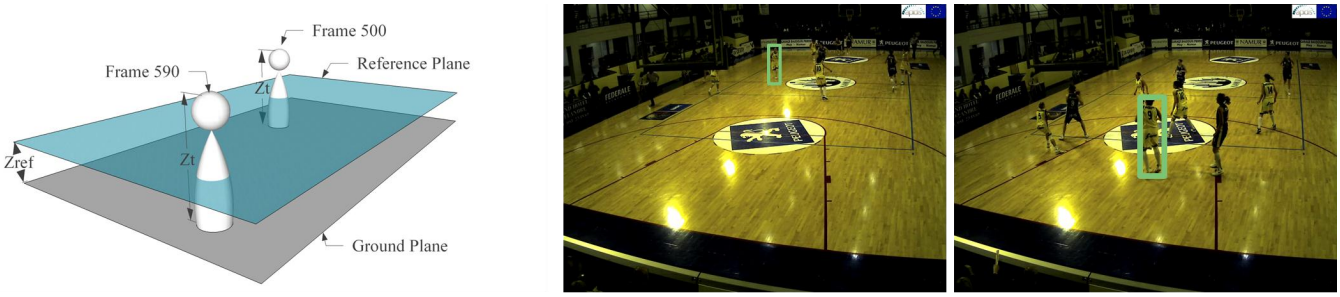


Figure 1: Targets are tracked in a reference plane parallel to and Z_{ref} units above the ground plane. To dynamically resize the bounding box for detection, the 3D points representing the top and bottom of the target are projected onto the image plane. The images in (b) and (c) show the automatically calculated bounding boxes for the same target at different times.

of the object. In the case where initial velocity estimates are not provided, we randomly sample the velocity from a 2D, zero-mean, Gaussian distribution where the covariance is set to be the average velocity of object motion. After object template initialization, we iterate through prediction, detection, and resampling at each timestep for online tracking.

Prediction.

For each particle $s_{t-1}^{(j)}$ we predict the new state at time t:

$$\mathbf{x_t^{(j)}} = \mathbf{x_{t-1}^{(j)}} + \dot{\mathbf{x}}_\mathbf{t}^{(\mathbf{j})}$$

$$\dot{\mathbf{x}}_\mathbf{t}^{(\mathbf{j})} = \dot{\mathbf{x}}_\mathbf{t-1}^{(\mathbf{j})} + \xi$$

where ξ is a noise term.

Detection.

To match two m-dimensional image patch histograms, q and p, for similarity, we employ the Bhattacharyya coefficient [9], a measure of similarity between two distributions:

$$\beta(q,p) = \sum_{u=1}^{m} \sqrt{q_u p_u} \tag{5}$$

Equation 5 is used to calculate the probability, $\rho(q,s,P)$ that the state represented by a given particle, $s_t^{(j)}$, coincides with the reference template, q_T. The position, in reference coordinates, $\mathbf{x_t^{(j)}}$, is converted to image coordinates using camera projection matrix, P and Equation 3, and the candidate object distribution q' is calculated using Equation 2. The *detection score*, ρ, is given as:

$$\rho(q_T, s, P) = \beta\left(q_T, q'\right) \tag{6}$$

For the set of particles $\mathbf{s_t}$, the detection scores ρ are normalized to give a non-parametric distribution of the target's state. The state of the object $\hat{s}_t$ at time t is the maximum likelihood estimate (MLE) of this posterior distribution. To continue tracking, the particles are resampled from this distribution to generate the set of particles for time $t+1$. The prediction and detection steps iterate on the next frame.

4. CAMERA COORDINATION MODEL

In the standard prediction-detection paradigm for object tracking, computation is typically dominated by the detection step, when many image locations are compared to the target's template. In this section, we describe our coordination model for multi-camera object tracking, which overcomes some of the issues associated with single-camera tracking, such as drift and occlusion, and also reduces overall computation by minimizing the aggregate number of detection processes needed in the complete system.

Let $\mathcal{C} = \{1, 2, \ldots, N\}$ be the set of identifiers for N cameras in a multi-camera network. We partition $\mathcal{C}$ into three subsets: *active* ($\mathcal{C}_A$), *passive* ($\mathcal{C}_P$), and *inactive* ($\mathcal{C}_I$) cameras. Active cameras track, as described in Section 3. Passive cameras are available, but are not actively tracking, and inactive cameras either do not contain the target within the field of view or are otherwise disabled (e.g., power saving). Our meta-algorithm for camera coordination iterates through tracking and reassignment steps.

Tracking.

At time step t, for each active camera $i \in \mathcal{C}_A$, we obtain the state estimate $s_t^{(i)} = \left[\begin{array}{cc} \mathbf{x}^{(i)} & \dot{\mathbf{x}}^{(i)} \end{array}\right]$ and detection score, $\hat{\rho}^{(i)}$, for the target. Each camera $j \in (\mathcal{C}_A \cup \mathcal{C}_P)$ evaluates the target state estimates, $\{\hat{s}_t^{(i)}\}$, identified by the active cameras and returns detection scores $\rho_j^{(i)}$. In the case of multiple active cameras ($|\mathcal{C}_A| > 1$), we determine an aggregate state estimate using a simple voting scheme. Each camera, j, selects the state estimate from active camera i with the highest value of $\rho_j^{(i)}$. The state estimate with the highest number of votes is selected as the network's estimate for time step t. This voting approach differs from previous multi-camera methods [14, 13] that directly compare detection scores across cameras.

Reassignment.

After tracking in frame t, cameras are reassigned to the active, passive, or inactive sets for frame $t+1$. Cameras whose state estimate indicates the object is out of the field of view or are otherwise disabled are labeled as inactive. Using a preference function, we rank the remaining cameras. This function could incorporate multiple factors, such as power remaining, long-term camera reliability, or priors from an environment model. In our experiments, we evaluate two preference functions. The first uses the detection scores as direct measures of tracking confidence. The second approach relies on the relative change in detection score. Due to differing color calibration, the size of the target in the frame, or occlusions in the scene, raw detection scores

Figure 2: Frames from the 7 cameras in the APIDIS dataset.

Scenario	Number of Frames	Target Acceleration	Occlusion Severity
1	420	Medium	Medium
2	200	Low	High
3	250	High	Low

Table 1: Description of the three test scenarios.

may not be meaningful between cameras. As an alternative, we also evaluated the relative change in detection score, $(\rho_t - \rho_{t-1})/\rho_{t-1}$, as a preference function. Based on the preference function, the top T cameras become active, and the remaining are passive.

We iterate through these two steps, tracking and reassignment, for each frame of the video. In the tracking step, the computation savings arise for inactive and passive cameras. Rather than evaluating a large number of locations in the detection step (typically hundreds using particle filters), only a few locations (those returned by the active cameras) need to be evaluated. This savings would be realized not only for particle filter tracking, but for any tracker using the prediction-detection model. In the next section, we show how this coordination method improves tracking on a challenging data set.

5. RESULTS

To test our method, we used the APIDIS dataset[1], which consists of footage from a basketball game recorded by seven calibrated, pseudo-synchronized, color cameras from various angles (Figure 2). This sequence includes 12 moving objects (10 players plus 2 referees) with uniforms similar in appearance to each other and the court, as well as challenging shadows and reflections. Ground truth image positions are included for each moving object in each camera at each time step for error calculation. To evaluate our method, we selected three tracking scenarios (summarized in Table 1), which contain a mix of sudden target acceleration, difficult occlusions across multiple cameras, and instances of the target exiting the view of one camera and entering another.

Single-Camera Tracking.

As a baseline, we performed single-camera tracking (as described in Section 3) using 200 particles per camera for tracking. The error metric is the distance on the reference plane from the target's estimated position to the ground truth position. Figure 3 (a) shows the mean tracking error for three different cameras. (Only cameras 2, 3, and 5 contain a view of the target for the duration of the tracking experiment.) The units correspond to a predefined coordinate system; 15 units equates to mistracking the target by one foot in real-world units. The high error values (i.e., mean error greater than 50) usually indicate the tracker became "lost" and the target was completely mistracked. For scenarios 1 and 3, the best-performing individual cameras (3 and 5, respectively) were able to track the target successfully. So, even in cases of high target acceleration or moderate occlusion, the base tracker can perform well. However, for scenario 2, no individual camera was able to successfully track the target due to an occlusion with multiple players in the same area.

Coordinated Tracking.

We tested our multi-camera coordinated model using up to four active cameras.[2] For reassignment, we evaluated the two strategies described in Section 4, which we call *Score* and *RelScore*. As with the single-camera experiments, each active camera used 200 particles for tracking. Figure 3 (b) shows the mean error for each coordination scheme over the same three scenarios from the APIDIS data. Both multi-camera approaches outperformed the individual cameras.

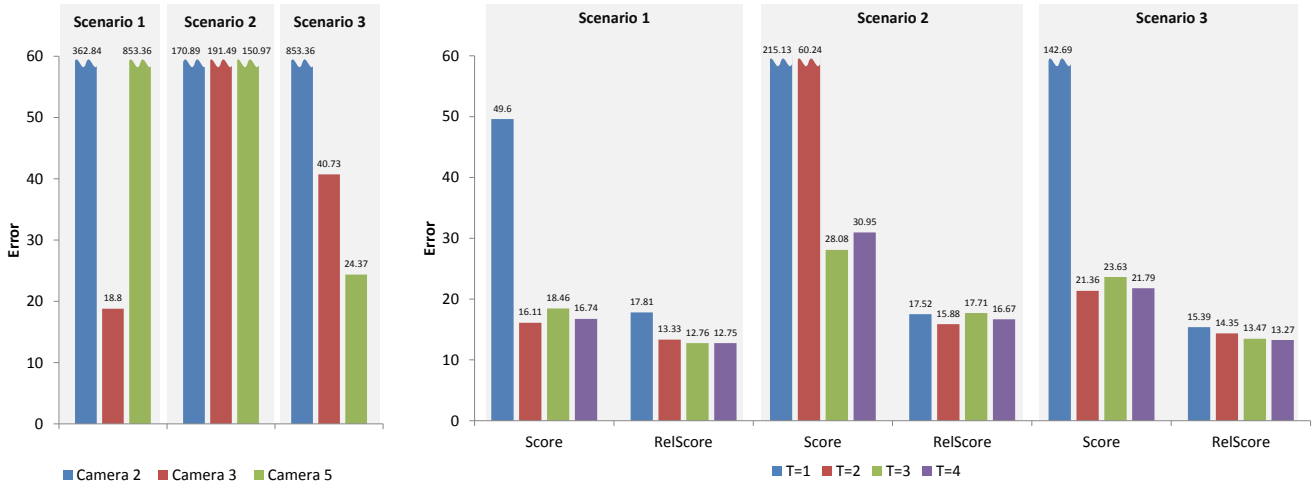
The first method, *Score*, which uses the raw detection scores as the preference function performs poorly in the case of a single active camera ($T = 1$), where the target is mistracked in 2 of the scenarios. This method is particularly vulnerable to a single high-scoring camera dominating the rankings. This effect appears to be mitigated at $T = 2$, even though the target is still mistracked in Scenario 2. With $T \geq 3$, the target in each scenario is successfully tracked, however with higher error rates than the *RelScore* approach. The second multi-camera method, *RelScore*, performed the best out of the three tracking methods. Even with a single active camera ($T = 1$), the method is able successfully track the target in all three scenarios, and error generally decreases as successive cameras are added. Over all the experiments, the *RelScore* scheme achieved a mean error of 15.07, compared to 53.73 and 211.55 for the *Score* and single-camera methods, respectively. Figure 4 illustrates how the system switches among cameras with $T = 2$ active cameras over scenario 2. Figures 5 and 6 show our method handling situations of mistracking and occlusion, respectively.

Computational Load.

To measure the computational load of our approach, we focus on what is typically the most computationally expensive step in tracking, evaluating a candidate location. In particle filter tracking, this step occurs once for each particle in each frame. In our approach, the number of evaluations is reduced, as only active cameras evaluate all of the particles, and non-active cameras perform just T evaluations, where T is the number of active cameras. For the multi-camera case with a single active camera ($T = 1$) and 200 particles for tracking, 206 detections are carried out per frame, a reduction of 85% of the computational effort compared to when all the cameras track independently, which is

[1] http://www.apidis.org/

[2] Even though the network contains seven cameras, a single object often does not appear in more than four camera views.



(a) Single-camera Error (b) Multi-camera Error

Figure 3: Mean error for (a) single-camera and (b) coordinated multi-camera tracking. Error units correspond to a predefined coordinate system; 15 units equates to mistracking the target by one foot in real-world units.

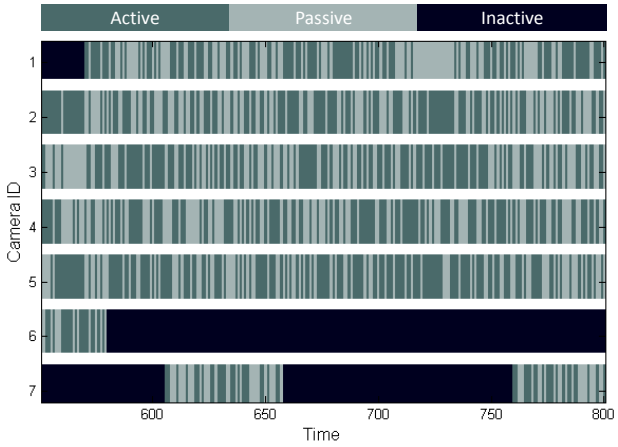


Figure 4: This timeline depicts the state of each camera for tracking an object using $T = 2$ active cameras.

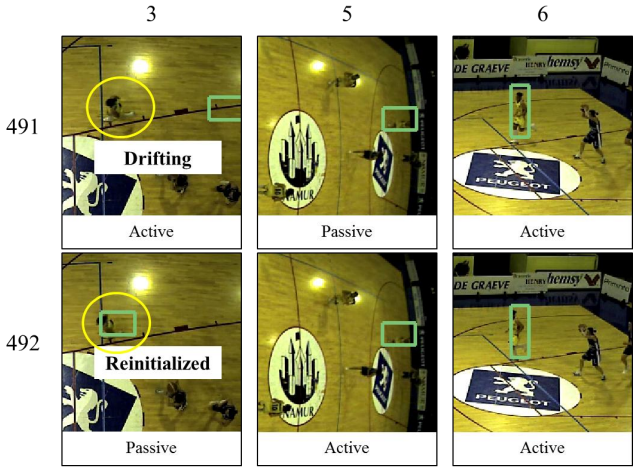


Figure 5: Cooperative camera switching for three cameras. In the top row (frame 491), camera 3 has drifted. In the bottom row (frame 492), camera 5 becomes active, and camera 3 reinitializes using the estimates from the best cameras.

the approach taken with typical trajectory correspondance tracking methods. While adding active cameras increases the computational load, this number can be selected to balance the tradeoff between efficiency and tracking accuracy.

6. CONCLUSIONS AND FUTURE WORK

We have presented a coordination framework for object tracking in multi-camera networks. The main contribution lies in the coordination model, which allocates resources dynamically to the best cameras at each time step. We introduced a voting scheme to aggregate multiple tracking estimates and a flexible preference function for dynamically switching between cameras. In contrast to typical approaches to multi-camera tracking, which aggregate data from all available sensors, our method reduces resource requirements, making it suitable for real-time application. We are investigating more complex preference functions, with the goal of predicting a camera's future suitability to detect a target by incorporating knowledge from environment or human motion models. In addition, we plan to investigate appearance models that can be shared among camera views to facilitate camera switching and re-initialization.

7. REFERENCES

[1] N. G. Arnaud Doucet, Nando de Freitas. *Sequential Monte Carlo Methods in Practice*, chapter An introduction to Sequential Monte Carlo Methods, pages 3–14. Springer-Verlag, 2001.

[2] Q. Cai and J. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1241 –1247, Nov 1999.

[3] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–575, 2003.

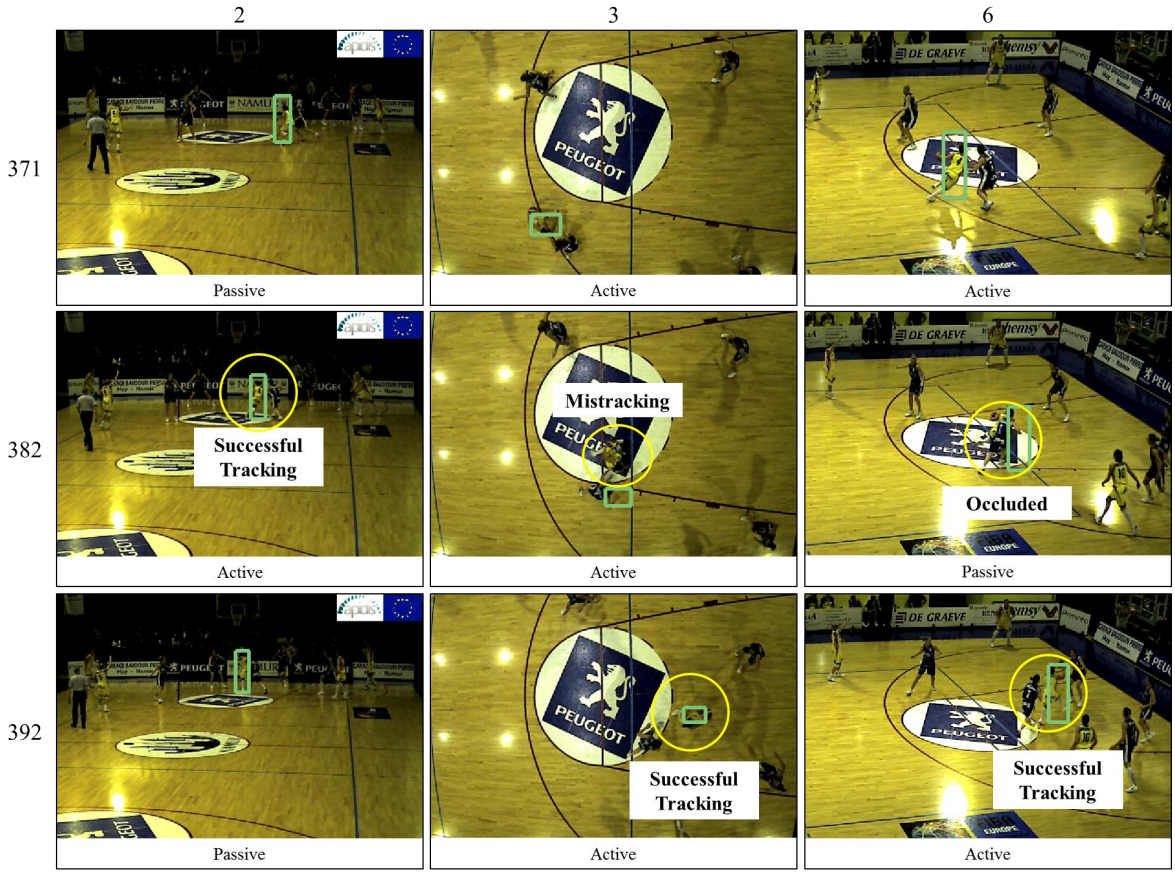


Figure 6: Pre-occlusion (frame 371), cameras 3 and 6 are active. During the occlusion (frame 382), camera 2 maintains a good view of the target and becomes active. Post-occlusion (frame 392), cameras 3 and 6 become active again.

[4] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *Intl J. of Computer Vision*, 40:123–148, 2000.

[5] D. Delannay, N. Danhier, and C. De Vleeschouwer. Detection and recognition of sports(wo)men from multiple views. In *International Conference on Distributed Smart Cameras*, pages 1 – 7, Sept 2009.

[6] W. Du and J. Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *Asian Conference on Computer Vision*, pages 365–374, 2007.

[7] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:267–282, 2008.

[8] B. Han, S.-W. Joo, and L. Davis. Multi-camera tracking with adaptive resource allocation. *Intl J. of Computer Vision*, 91:45–58, 2011.

[9] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Trans. on Comm. Tech.*, 15(1):52–60, 1967.

[10] D. Karuppiah, R. Grupen, Z. Zhu, and A. Hanson. Automatic resource allocation in a distributed camera network. *Machine Vision and Applications*, 21:517–528, 2010.

[11] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision*, volume 3954 of *Lecture Notes in Computer Science*, pages 133–146. Springer Berlin / Heidelberg, 2006.

[12] S. Khan and M. Shah. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 505–519, 2008.

[13] A. Mittal and L. S. Davis. M2Tracker: a multi-view approach to segmenting and tracking people in a cluttered scene. *Intl J. of Computer Vision*, 51:189–203, 2003.

[14] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. V. Gool. Color-based object tracking in multi-camera environments. *Lecture Notes in Computer Science*, 2781:591–599, 2003.

[15] M. Taj and A. Cavallaro. Distributed and decentralized multi-camera tracking: a survey. *IEEE Signal Processing Magazine*, 28(3), 2011.

[16] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38, December 2006.

[17] Q. Zhou and J. Aggarwal. Object tracking in an outdoor environment using fusion of features and cameras. *Image and Vision Computing*, 24(11):1244 – 1255, 2006.