

Chapter 1

SINGLE CAMERA MULTIPLEXING FOR MULTI-TARGET TRACKING

Chiraz BenAbdelkader,
Philippe Burlina,
and Larry Davis

*Computer Vision Laboratory
University of Maryland
College Park, MD 20742 USA
chiraz.burlina,lsd@cfar.umd.edu*

1. INTRODUCTION

Recent years have seen a continued increase in the need for, and use of automatic video surveillance, both in urban (civilian) and military airborne applications. A surveillance system is typically comprised of one or more video sensors (such as cameras), each of which is mounted on a mobile pan/tilt platform. Because the platform hardware is expensive, making the most use out of each sensor is a worthy goal. To this end, we consider the design of a novel real-time surveillance system that uses a single camera equipped with pan/tilt and zoom capabilities, to 'keep track' of as many targets as possible for as long as possible.

We assume the targets are scattered anywhere within the camera's field of regard ¹, i.e. they may not necessarily be captured within one field of view. Hence, to accommodate multiple targets at once requires that we aim the camera at different points of the field of regard, for limited amounts of time. Two main issues arise. The first is, how (and whether it is possible) to maintain the trajectory of a target without con-

¹The camera field of regard (FOR) is defined as the union of all fields of view over the entire range of the pan and tilt rotation angles.

tinuously tracking it. The second is, how to manage the time-allocation of the camera among these contending targets.

The solution we describe in this paper consists of two independent main modules that operate in a cyclical loop: a *planning module* and a *control-tracking module*, as shown in the block diagram of Figure 1.1. The planning module handles the time-allocation problem by selecting one target at a time to be tracked for a prescribed amount of time. The control-tracking module tracks the 3D motion of the designated target within the allotted amount of time, based on image measurements taken by the camera at a fixed frame rate.

The goal of the control-tracking module is to minimize the error covariance of the motion estimate, and to spend as little time as possible on each target without sacrificing certainty. We design this module as a recursive data filter and a (negative) feedback controller. The filter estimates the target’s motion based on noisy measurements of its position in the image, and the feedback controller controls the orientation of the camera to make sure the target stays within the camera’s field of view during tracking. When successful, the control-tracking module returns the target’s motion trajectory, which the planning module subsequently uses to predict forward the motion of this target during the time that it is not being tracked.

The planning module essentially schedules access to the control-tracking module among a *maximal* set of (contending) targets. We design a scheduling scheme that is a function of two key parameters to be defined for each target: (i) the length of each target’s time slot, and (ii) the time between any two consecutive time slots allocated to it. Obviously, the former corresponds to the amount of time the control-tracking module may spend on the target, and the latter corresponds to its inter-tracking time. In order to maximize the number of tracked targets without overloading the system, the proposed scheme also dynamically adjusts the workload (number of targets handled at once) based on measured performance of the system. Performance analysis and modeling here is done within the framework of a single-server closed queuing model. This model is particularly fitting, since we can view the system as a set of users (the targets) contending over access to a scarce resource (the camera).

The rest of the paper is organized as follows. In Section 2, a brief survey of previous related work is presented. In Sections 4 and 5 we discuss the implementation details of the control-tracking module and planning module, respectively. In Section 6, we present the methodology and results of simulations carried out on the system with data extracted

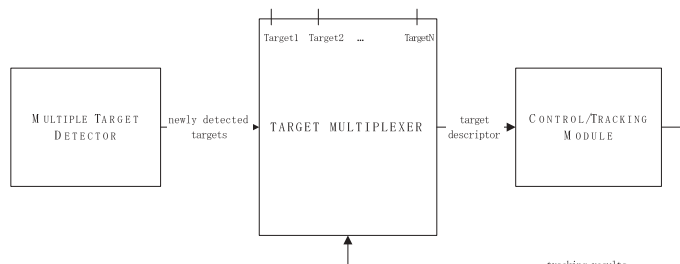


Figure 1.1 System Architecture.

from real sequences. Finally, in Section 7 we give a conclusion of our findings and directions for future work.

2. RELATED WORK

While to our knowledge the problem of using a single camera to track multiple targets in its FOR has not been addressed before, target tracking is a recurrent problem in computer vision [7] [4] [9]. The scope and approach of each attempt at this problem vary in many ways; in terms of the number of cameras used (i.e. monocular, binocular, etc.), whether the tracking is done in 2D or 3D, whether the camera is stationary or moving, the use of kinematic models and/or dynamic constraints to represent the motion of the moving objects, the motion computation technique (e.g. correlation matching, temporal gradients), and the motion correspondence algorithm [5]. Since we are using only one camera, the literature on monocular tracking is of special interest to us.

Another useful body of work comes from the robotics and control literature, specifically that regarding visual servoing and camera fixation [14] [15] [13] [2]. The problem here is: how to aim the camera such that the projection of some desired 3D point of a moving object lies in some desired position in the image. The solution approach usually consists of a feedback control mechanism that uses visual information (namely a measurement of the location of the desired feature in the image) to adjust the attitude and/or position of the camera. Some of the issues involved in this problem are handling camera jitter, as well as turn delays (and possibly computational delays) in the control loop.

3. ASSUMPTIONS

- The platform is stationary, and equipped with a pan/tilt mechanism to rotate the camera. The internal calibration parameters

and the position of the camera are assumed to be known at all times.

- An external mechanism, such as a Moving Target Indicator (MTI), does the initial detection of moving targets in the camera's FOR and cues our system with their initial 3D positions.
- The targets are moving on a known surface and their motions are sufficiently modeled with first-order dynamics.

4. CONTROL-TRACKING MODULE

4.1 OVERVIEW OF OPERATION

The goal of this module is to track a given target for a prescribed amount of time, and return the estimated motion trajectory. This task requires acquisition (i.e. recovery) of the target track, maintaining it for the given amount of time, and adaptively controlling (repositioning) the camera so that the target remains visible throughout tracking. The module is decomposed into three main computational sub-modules, operating in a feedback loop as illustrated in Figure 1.2.

The *target recognition* sub-module determines the 2D location of the target of interest in each image frame. This can be achieved for example via appearance matching within a search window around the target's predicted position in the image. The *data filter* sub-module estimates the target's 3D motion trajectory based on noisy image measurements provided by the detection sub-module. Here we shall use a first-order dynamic model to represent target motion, and the Extended Kalman Filter (EKF) algorithm to recursively estimate the parameters of this model (i.e. position and velocity) [1] [6] [8]. Finally, the *camera control* sub-module computes the rotation of the camera required to keep the target visible during the tracking interval, based on the predicted 3D position of the target and calibration parameters of the camera.

4.2 EXTENDED KALMAN FILTER

The Kalman filter is a recursive stochastic estimator that is known to be optimal (in the minimum mean square, MMSE, sense) for linear systems driven by Gaussian white noise, but is otherwise only the best linear estimator [1] [6]. Albeit there exist less restrictive (more distribution-free) estimators, such as the condensation algorithm which is particularly useful for multi-modal noise [10], we shall use the Kalman filter in this paper, due to its simplicity and computational efficiency.

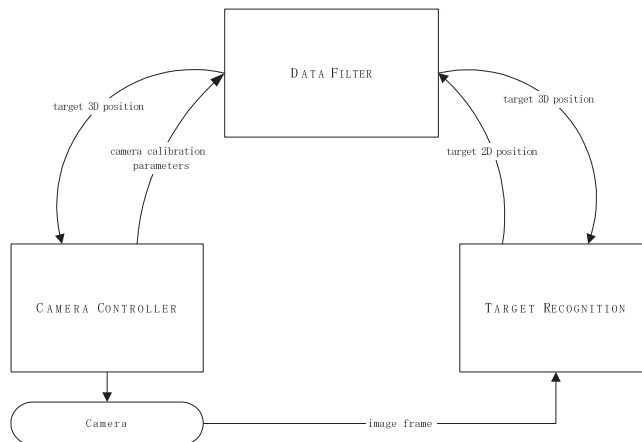


Figure 1.2 Control-Tracking Module.

4.2.1 Stopping Criteria. The filter need only be iterated until the state estimate has become *consistent*, which means both that the error covariance of the state estimate has converged stochastically to a constant value, and that its error covariance is equal to the filter-calculated covariance P_k :

$$\lim_{k \rightarrow \infty} P_k \equiv \lim_{k \rightarrow \infty} E\{[\hat{x}_k - x]^2\} = \bar{P}.$$

The filter-calculated covariance P_k indeed converges to a steady value \bar{P} (which corresponds to the solution of the Ricatti equation) only if the system is stationary (time-invariant) and driven by white noise [1]. Moreover the rate of convergence depends on the magnitude of modeling errors, observability problem and roundoff errors, all of which may cause the filter to diverge [3].

In real data applications, we can check whether the above consistency condition holds to within a reasonable level, in a statistical sense, via two hypothesis tests on the innovations [1]. One test checks whether the innovations are unbiased and consistent with the filter-calculated covariance, and the other checks whether the innovations are white (i.e. uncorrelated in time). The acceptance of these two hypotheses tests at some confidence level γ ($\gamma = 95\%$ is reasonable) can hence be used as the filter stopping criterion.

5. PLANNING MODULE

5.1 OVERVIEW OF OPERATION

This module implements the high-level multiplexing of the camera among multiple targets moving anywhere within the camera's field of regard. For this, it shall maintain a set of candidates, the *tracking set*, from which it selects one target at a time to pass on to the control-tracking module, in some round-robin-like fashion.

As we explained in the previous section, the control-tracking module uses a first-order dynamic model to represent target motion, and provides an estimate of its motion trajectory of the target during the tracking interval. The planning module can then use the final position and velocity estimates of a target (i.e. at the last step of its tracking interval) to predict forward its trajectory during the time that it is not being tracked (and until the target is scheduled again for tracking). A flowchart of the planning algorithm is depicted in Figure 1.3.

The tracking set is initially empty. Tracking candidates, consisting of moving objects that are independently detected by the external detection mechanism, will be inserted into this set as necessary, as we shall explain shortly. The detection mechanism presumably provides the initial position and time of detection for each candidate. Each target in the tracking set is characterized by two parameters. The maximum tracking time, denoted by τ , specifies the maximum time that the control-tracking module is allowed to spend on the target. The maximum inter-tracking time, denoted by δ , specifies the maximum time the target can be left un-tracked. Finally, the tracking set is implemented as a priority queue, in which targets are stored in order of *expiration time*, defined as δ time units from when the target was last tracked; A target that is not scheduled on or before its pre-computed expiration time is presumed lost, and is removed from the tracking set.

In the following two sections, we discuss the algorithm's two main computational modules in more detail. The first computes the two characteristic parameters of each target (τ and δ), and the other computes system performance and adjusts the tracking set accordingly.

5.2 COMPUTATION OF TARGET PARAMETERS

5.2.1 Maximum Tracking Time. This parameter is an upper bound on the amount of time the control-tracking module needs to spend on a target. As discussed in Section 4.2.1, this is determined by how long the Kalman filter takes to reach a consistent state (at a given level of

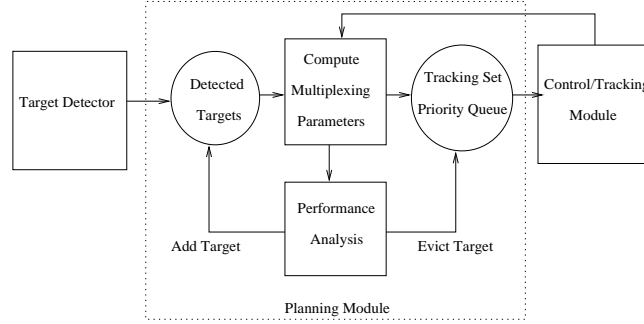


Figure 1.3 Flow chart of planning module.

confidence). Because we cannot determine this value a priori, instead we estimate it empirically as follows. When a target is first added to the tracking set, its τ is initialized to some large value τ_{max} . Then, after each time the target is tracked, we set τ to the median of all the actual tracking times (i.e. the time that the Kalman filter took to reach a consistent state) observed so far for that target.

5.2.2 Maximum Inter-tracking Time. When a target is not being tracked, its motion trajectory (and error covariance) is predicted forward until it is scheduled again for tracking. Prediction error accumulates over time due to uncertainty of the motion model, and at some point will become so large as to impede target re-acquisition upon its next visit to the server. Obviously, re-acquisition is possible only if the actual position of the target is inside the camera's field of view at this time. As such, we shall define δ as the largest inter-tracking interval such that the target can still be found within $FOV(\hat{X}_\delta)$ with some confidence $1 - \alpha$, namely:

$$\delta = \text{Max}_k \{Pr\{X_k \in FOV(\hat{X}_k)\} \geq 1 - \alpha\}$$

where X_k and \hat{X}_k denote the true and predicted positions respectively at time k , Π_k the error covariance of the latter, and $FOV(\hat{X}_k)$ the camera's field of view when its optical axis is aligned with \hat{X}_k .

Since prediction error increases over time, and assuming the area of the field of view does not increase over time significantly enough, the probability $\gamma_k = Pr\{X_k \in FOV(\hat{X}_k)\}$ is a decreasing function of k . In other words, our confidence in detecting the target within the predicted field of view diminishes over time. Thus, δ can be determined via a *binary search* on k , with γ_k as the objective function, and using some appropriate initial search interval $[k_{min}, k_{max}]$.

Exact evaluation of γ_k in real-time can be prohibitively expensive, as it requires integration of the probability density function of \hat{X}_k over the FOV area, and the latter generally takes on the shape of an irregular quadrangle. We circumvent the need for this expensive computation by exploiting a property of the iso-probability contours of the Gaussian distribution. Namely, assuming the Gaussian white noise assumption holds, X_k has a bivariate Gaussian distribution with mean \hat{X}_k and error covariance Π_k . Furthermore, for any bivariate random variable y with mean μ , the $(1 - \alpha)100\%$ iso-probability contours are ellipses with the following property:

$$Pr \left[(y - \mu)^T \Sigma^{-1} (y - \mu) \leq \chi_p^2(\alpha) \right] = 1 - \alpha$$

where $\chi_p^2(\alpha)$ is the $(1 - \alpha)100\%$ percentile of the Chi-square distribution. Thus we shall instead binary-search for the largest k such that the $(1 - \alpha)100\%$ -contour is circumscribed within $FOV(\hat{X}_k)$, a relatively straightforward task, so that $\gamma_k \geq 1 - \alpha$, as desired.

5.3 QUEUING-BASED PERFORMANCE ANALYSIS

This is the adaptive component of the planning module algorithm that dynamically adjusts the tracking set in order to maximize system performance.

Because our system can be viewed as multiple users (targets) contending for access to one scarce resource (the camera), we shall use the single-server closed queuing system shown in Figure 1.4 to model its performance. Performance modeling is a powerful tool in systems analysis. It is useful for predicting the values of performance measures of a system from a set of workload parameter values [11] [12].

In this queuing model, customers correspond to targets, the service center represents the control-tracking module, and the delay center the planning module. Each customer remains for a certain period of time in the delay center, then proceeds to the service center where it waits in the server queue for its turn (on a First-In-First-Out basis). Finally, once a customer is serviced it either goes back to the delay center (if it needs more rounds of service), or exits the system altogether.

The queuing system is defined by its input parameters and performance indicators. The input parameters are the *Think Time*, T_i , defined as the average time a customer remains at the delay center, the *Service Requirement*, S_i , defined as the average amount of time a customer needs in any one visit to the server, and the *Degree of Multiprogramming*, N , defined as the number of customers present in the system at once, and

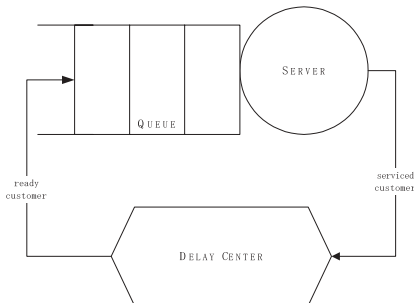


Figure 1.4 System Queueing Model.

hence corresponds to the number of targets in the tracking set. The *workload* of the system is collectively defined by the think time and service requirement values of all customers currently in the system. The performance indicators of interest to us are the *Server Utilization*, U , defined as the percentage of time the server is being used, during a given time, and the *Response Time*, W_i , defined for each customer as the average time elapsed between when the customer enters the queue and the time it is serviced.

In order to control (adjust) the degree of multiprogramming to maximize the performance of the system without overloading the system, we devise an iterative algorithm wherein the workload is initialized by setting $S_i = \tau_i$ and $T_i = \delta_i - W_{MAX}$ for each customer ($1 \leq i \leq N$). W_{MAX} is an initial guess of the maximum response time. T_i is initialized this way so that $\delta_i \leq T_i + W_i$, as desired. The algorithm iterates for a given maximum number of iterations, or until it reaches steady state, i.e. when $\forall i, \delta_i = T_i - W_i$ and $U \geq U_{MAX}$. U_{MAX} is a design parameter specifying the upper bound on system utilization. The Mean Value Analysis (MVA) [11] [12] is a technique used to determine the values of the various performance measures as a function of the workload parameter values.

6. EXPERIMENTS

The methods developed herein were implemented in C++ and tested on a Dual Pentium processor. For current lack of the hardware necessary for real-time operation (i.e. a controllable pan/tilt camera), we simulated our system with synthetic data as well as data extracted from real video sequences. This data constitutes the actual (i.e. 'ground truth') 3D trajectories of the targets, which are then 'perturbed' to obtain the measurements that drive the data filter. Obviously this data

replaces the vision sub-module. In the sequel, we shall only explain the simulations done with real sequences, because they are more interesting (and because of space limitations).

The video sequence depicts a typical urban outdoor surveillance setting, wherein eleven people are walking randomly, in different directions, at a various paces in a more or less linear fashion. The camera used has a wide-angle field of view (a focal length of 67mm and image size of 720x480 pixels), and is kept stationary throughout the sequence. The actual target trajectories in the sequence were determined via a simple 2D detection tracking algorithm (see for example [7]). The results were subsequently inspected by hand to correct any errors in the algorithm. In order to simulate the switching of the camera viewpoint, we let the actual field of view be the (virtual) camera FOR, and the image plane be 50x30 pixels (instead of the actual 720x480) corresponding to a virtual narrow field of view. Figure 1.5 shows the virtual fields of view (the yellow quadrangles), with their centers connected by the red poly-line, over the first 100 frames of the sequence.

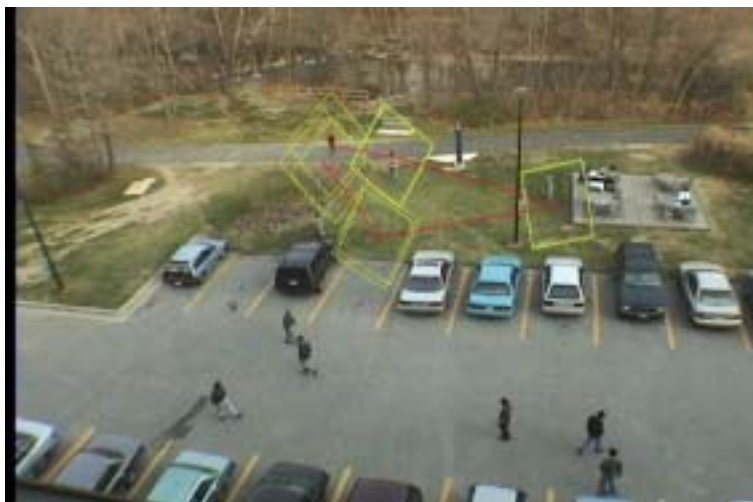


Figure 1.5 Virtual field of view of the camera for the first 100 frames of the sequence.

6.0.1 Results. Figure 1.7 below shows a plot of the degree of multiplexing throughout the sequence, and Figure 1.6 shows the trajectories of two of the tracked targets. The red line represents the actual trajectory and the green line represents the filter-estimated trajectory.



Figure 1.6 Estimated and measured trajectories of two of the multiplexed targets.

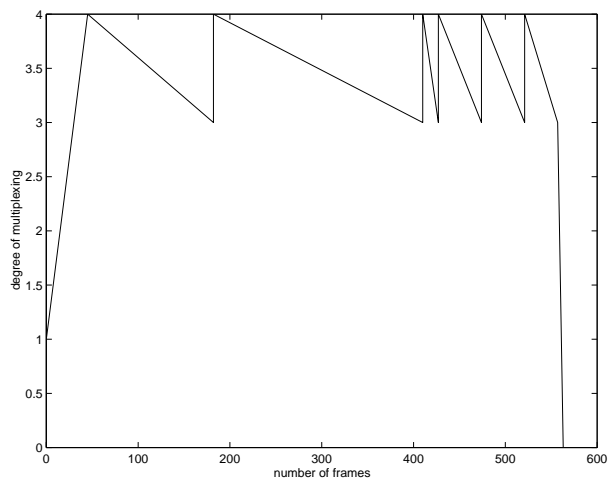


Figure 1.7 Degree of multiplexing vs. time.

7. CONCLUSIONS

This paper presented the design of a novel surveillance system for tracking multiple targets in a wide area using a single narrow-field-of-view camera. The system so far has only been tested off-line on synthetic data. In the future, we aim to extend the current implementation of the system to handle real-time operation. Alternative, more robust heuristics for computing the two target parameters will also be investigated.

References

- [1] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, Academic Press, New York, NY, 1988.
- [2] C. Brown, "Gaze Controls with Interactions and Delays," *IEEE Transactions on Systems, Man, and Cybernetics*.
- [3] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, New York, 1983.
- [4] I. Cohen and G. Medioni, "Detecting and Tracking Moving Objects for Video Surveillance," in *IEEE Proc. Computer Vision and Pattern Recognition*, June 1999.
- [5] I. J. Cox, "A Review of Statistical Data Association Techniques for Motion Correspondence," *International Journal of Computer Vision*, Vol. 1, No. 10, pp. 53–66, 1993.
- [6] A. Gelb, *Applied Optimal Estimation*, MIT Press, Cambridge MA, 1974.
- [7] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4S: A Real-Time System for Detecting and Tracking People in 2 1/2 D," in *Proc. European Conf. Computer Vision*, June 1998.
- [8] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1986.
- [9] D. Huttenlocher, J. Noh, and W. Rucklidge, "Tracking non-rigid objects in complex scenes," in *Proc. Int. Conf. Computer Vision*, May 1993.
- [10] M. Isard and A. Blake, "Condensation—conditional density propagation for visual tracking," *International Journal of Computer Vision*, No. 15, pp. 234–44, 1998.
- [11] R. Jain, *The Art of Computer Systems Performance Analysis Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, 1991.
- [12] E. Lazowska, J. Zahorjan, G. S. Graham, and K. Sevcik, *Quantitative System Performance*, Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1984.
- [13] T. Matsuyama, "Cooperative Distributed Vision - Dynamic Integration of Visual Perception, Action, and Communication -," in *Proc. Int. Conf. Computer Vision*, 1998.
- [14] N. Papanikolopoulos, P. K. Koshla, and T. Kanade, "Vision and Control Techniques for Robotic Visual Tracking,"

- [15] e. a. Wixson, L., "Image Alignment for Precise Camera Fixation and Aim," in *Proc. Conf. Computer Vision and Pattern Recognition*, 1998.