

# Localizing a Network of Non-Overlapping Cameras

Ali Rahimi      Trevor Darrell

{ali,trevor}@mit.edu

Computer Science and Artificial Intelligence Laboratory  
Massachusetts Institute of Technology  
Cambridge, MA

## Abstract

We describe a method for recovering the extrinsic calibration parameters of non-overlapping cameras in a multi-camera system. Non-overlapping camera networks can cover a much wider area than overlapping configurations, but calibrating them is more challenging. To compensate for the lack of overlap, our approach assumes that the calibration target follows a smooth trajectory over time, as described by a stochastic dynamics model. Our algorithm then tracks the target when it comes in the field of view of each camera, and searches for calibration parameters and target trajectories that are consistent with the observed tracks. We demonstrate the idea with a network of indoor wireless cameras.

## 1 Introduction

Networks of cameras have been shown to be an effective means for tracking people in surveillance settings. In these applications, the field of view of the cameras are usually made to overlap at least slightly, making it easier to track people as they move from the field of view of one camera to another, and in turn to recover the pose the cameras. Estimating these poses allows tracking to be performed in a globally consistent coordinate system. To instrument a wider area using fewer cameras, various authors have considered camera networks where the field of views of the cameras do not overlap [4, 7, 3]. Under these settings, cameras are placed in regions where interesting activity is expected to occur, and human motion is interpolated in uninstrumented regions.

We describe a calibration procedure for recovering the relative location and orientation of non-overlapping cameras in order to provide a globally consistent coordinate system for tracking. Our procedure requires a point calibration source that moves along the ground plane in smooth paths between the field of view of the

cameras. This path is not required to be straight, or to have constant velocity: we only assume that its evolution is consistent with a known stochastic dynamical model. Our main contribution is to show that a model of dynamics for the motion of the calibration target can compensate for the lack of overlap between the cameras.

When there is overlap between the fields of view of cameras, the appearance of the calibration target in a region of overlap provides information about the relative pose of the cameras. With enough such appearances, the relative orientation of the cameras can be recovered by taking advantage of stereopsis. Without information about the dynamics of the target, observing the calibration target where the cameras do not overlap only provides information about its pose relative to the camera that observed it, but not between the pose of two cameras.

For each camera, our algorithm recovers the two translation parameters and the rotation parameter in the ground plane. We rely on existing single-camera calibration techniques to recover the remaining extrinsic and intrinsic parameters before our algorithm is applied. The calibration task can be separated into these two tasks because single-camera calibration procedures can recover the pose of the camera relative to a local patch on the ground plane, up to a rotation and translation in the ground plane, in addition to recovering the intrinsic camera parameters. After such a procedure, our algorithm can be applied to globally align the coordinate system of the patches corresponding to each camera.

To perform this global alignment step, one could require that the target moves at constant velocity and in a straight line when it leaves the field of view of a camera. An intuitive approach would then be to have each camera track the target when the target passes through the camera's field of view, and estimate the target's velocity as it leaves the field of view. The travel time between the fields of view of two cameras, along with the

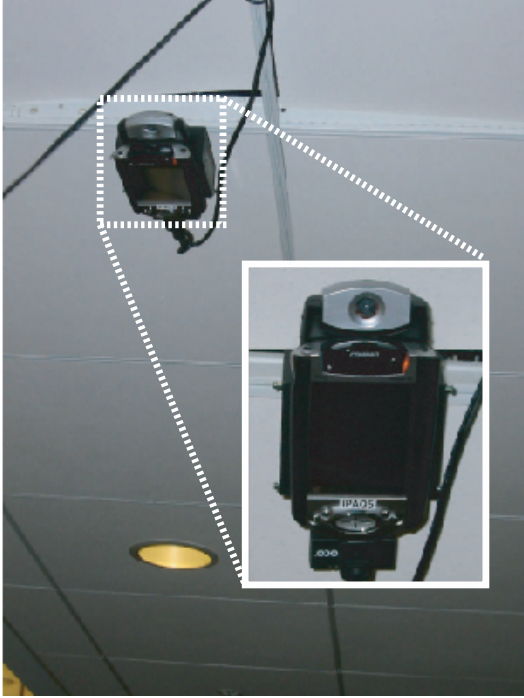


Figure 1: We use Compaq IPAQs as wireless camera nodes in our network. The IPAQs are mounted on the ceiling, with their camera image plane parallel to the ground plane.

target’s estimated velocity can be used to estimate the distance between the two fields of views. When enough such distances are obtained between pairs of cameras, they can be combined to recover the relative location and orientation of all the cameras in the network [3].

Our algorithm relaxes the assumption that the target moves in straight lines and at constant velocity. This allows the calibration target to be more casually carried by a person moving along curved paths. Our algorithm searches for camera calibration parameters and a smooth target trajectory that are consistent with the tracks observed by each camera while the target was in its field of view. This search is framed as finding the maximum *a posteriori* (MAP) camera calibration parameters and target trajectory, with a prior that prefers smooth target trajectories.

We have implemented our algorithm on a network of wireless PDAs equipped with cameras (see Figure 1). A real-time person tracker runs on each PDA which report trajectories to a central server that implements our algorithm.

## 2 Related Work

Many methods explicitly designed for calibrating networks of cameras rely on overlapping fields of view [8, 9, 5, 2]. The method of Javed et al. [3] is one exception, where an idea similar to velocity extrapolation is used to find the projection of the field of view lines of one camera onto the field of view of cameras. Knowledge of these projections is tantamount to recovering calibration parameters, but this method requires people to walk in a straight line and at constant velocity outside camera fields of view. Our work is most closely related to [1], which shows how to calibrate a network of non-overlapping cameras using distant objects (stars) to recover orientation, and nearby objects (airplanes) to recover relative position. Whereas [1] relies on strict constraints on the motion of the calibration targets (the motion of stars is parabolic, nearby targets must move in a line with constant velocity), our targets may move freely. Furthermore, we require only one type of target, and this target may be the same type of target the network will ultimately track.

Various researchers have addressed the problem of maintaining consistent identity between multiple targets as they exit one field of view and enter another [4, 7] in non-overlapping multi-camera systems. These techniques focus on recovering the topology of the network, and do not attempt to recover metric pose of the cameras.

## 3 Single-Camera Calibration

Before running our algorithm, each camera is calibrated so that it can map the image coordinate of the target to a local coordinate system laid on the ground-plane. The homography required to perform this mapping can be estimated for each camera individually [11]. For example, by laying a calibration object of known size on the ground plane, the focal length, height, pitch, and yaw of the camera can be estimated. This local homography can be used to rectify the image location of the target in each camera so that the camera appears to be virtually fronto-parallel to the ground plane. This leaves a translation and roll in the ground plane to be estimated by our algorithm.

In the remainder of this paper, we presume that the homography for each camera has been recovered up to a rotation and a translation in the ground plane, and focus on aligning the local ground-plane coordinate system of each camera to the global ground plane coordinate system. In our experiments, we used overhead cameras, so that only the height and the focal length of the cameras needed to be estimated during

the single-camera calibration phase.

## 4 Global Alignment

To globally calibrate the network, we jointly find the maximum *a posteriori* (MAP) estimate of the remaining calibration parameters and the trajectory of a point calibration target that moves smoothly in the network. Each camera tracks the point calibration target as it passes through its field of view. MAP estimation amounts to searching for calibration parameters and a trajectory that are consistent with the observations made by each camera and a dynamics model for the motion of the target. Since these can only be recovered up to a global rotation and translation in the ground plane, we fix the parameters of one of the cameras.

We assume that the state evolves according to linear Gaussian Markov dynamics. Define  $x_t$  to be the state of the target at time  $t$ . This state contains information about the location, velocity, or any other dynamic state of the target, and evolves with:

$$x_{t+1} = \mathbf{A}x_t + \nu_t, \quad (1)$$

where  $\nu_t$  is a zero-mean Gaussian random variable with covariance  $\Sigma_\nu$ .

In our experiments, we let  $x_t = [u_t; \dot{u}_t; v_t; \dot{v}_t]$ , where  $(u_t, v_t)$  is the ground-plane location of the target and  $(\dot{u}_t, \dot{v}_t)$  is its ground-plane velocity. We use the model parameters

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

so that each  $x_{t+1}$  adds the velocities in  $x_t$  to the positions in  $x_t$ , and nudges the old velocities by Gaussian noise. We use a diagonal  $\Sigma_\nu$  whose position components have much smaller value than its velocity components (by a few orders of magnitude), so that the velocities follow Brownian dynamics, and the resulting poses are nudged only by a small amount of Gaussian noise. Equation (1) defines a prior  $p(x)$  over a state trajectory  $x = \{x_t\}_{t=1}^T$ .

To extract the location components in  $x_t$ , we can multiply  $x_t$  by  $\mathbf{C}$ :

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

After taking into account the homography discussed in the previous section, camera  $i$  reports the location of the target in its own ground-plane coordinate system whenever the target is in its field of view. Let  $\theta^i$  and  $p^i$

denote the unknown rotation and translation of camera  $i$  with respect to the global ground-plane coordinate system. Let the index set of observations  $\mathcal{Z} = \{(t, i)\}$  be a collection of time index and camera pairs, where  $(t, i) \in \mathcal{Z}$  iff camera  $i$  sees the target at time  $t$ . Let  $\mu^i = [p^i; \theta^i]$  be the parameters of camera  $i$ , and define  $\mu = [\mu^1; \dots; \mu^N]$  to be the collection of all the camera parameters. Letting  $\mathbf{R}^i(\theta)$  denote the rotation matrix corresponding to a rotation of  $\theta^i$ , the location reported by a camera when the target is in its field of view is:

$$y_t^i = \pi(x_t; \mu^i) + \omega_t = \mathbf{R}^i(\mathbf{C}x_t - p^i) + \omega_t, \quad (3)$$

where  $\pi$  is the mapping between global and local camera coordinate systems. We assume  $\omega_t$ , the observation noise corrupting each measurement, is iid zero-mean and Gaussian with covariance  $\sigma_y^2 \mathbf{I}$ . Equation (3) defines a likelihood  $p(y|x, \mu)$  over trajectories and camera calibration parameters.

Any configuration of  $x$  and  $\mu$  will produce the same tracking measurements as the same configuration arbitrarily rotated and translated [10]. This ambiguity can be fixed by setting the parameters of one of the cameras to some arbitrary value. We use a prior that favors configurations where the parameters of the first camera are 0.

Using the likelihood defined by Equation (3) and the prior over  $x$  defined by Equation (1) and the gauge-fixing prior for the first camera, the most *a posteriori* probable trajectory and calibration parameters can be obtained by performing a nonlinear least squares optimization over  $\mu$  and  $x$ :

$$(x^*, \mu^*) = \arg \min_{x, \mu} \sum_{(t,i) \in \mathcal{Z}} \frac{1}{\sigma_y^2} \|y_t^i - \pi^i(x_t; \mu)\|^2 + \sum_{t=1}^T \|x_t - \mathbf{A}x_{t-1}\|_{\Sigma_\nu}^2 + \|\mu^1\|^2. \quad (4)$$

The first term in the cost function favors trajectories and parameters that would have generated the observed trajectory snippets. The second term favors trajectories that are consistent with the given dynamics by favoring trajectories where  $x_t$  can be predicted from  $x_{t-1}$  using the transition matrix of the dynamics model, and penalizing the prediction error by with a mahalanobis distance defined by the covariance of the driving noise. The third term fixes the gauge by setting the parameters of the first camera to zero.

To find the optimal  $\mu$  and  $x$ , we use Newton-Raphson with a first order approximation to the Hessian. This involves it iteratively linearizing the non-linearity inside the first term of Equation (4), transforming it to a linear least squares problem. This

least squares problem involves a minimization over  $3 * N + 4 * T$  variables with  $|Z|$  rows, where  $N$  is the number of cameras and  $T$  is the number of time steps in the trajectory to be estimated. It is also sparse, with  $O(|Z|)$  nonzero elements, and so can be solved efficiently using standard sparse least squares solvers. The appendix derives the quantities needed in each Newton-Raphson step.

## 5 Synthetic Results

Our goal with this synthetic experiment was to see if a disparity between the dynamics model and actual target behavior had adverse effects on trajectory estimation and calibration. We simulated a point target traveling in a square environment with elastic walls. The target’s trajectory was generated by Brownian motion that deflected off of walls. The resulting trajectory was smoothed to yield the setup shown in Figure 2(a). The synthetic cameras were front-parallel to the ground plane, and measured the location of the target without noise within their coordinate system (i.e. up to a rotation and translation of the true location of the target).

To define the dynamics model, we used the  $\mathbf{A}$  matrix of equation (2), and set  $\Sigma_\nu = \begin{bmatrix} 10^{-4} & 1 & 10^{-4} & 1 \end{bmatrix}$ . Notice that this dynamics model is different from the model used for generating the synthetic trajectory, as it does not model the bouncing behavior near walls, the final smoothing step, or the underlying Brownian motion.

The Newton-Raphson iterations were initialized with all trajectory points and camera parameters at the origin, pointing to the right. Figures 2(b-c) show an intermediate iteration and the converged estimate.

The estimated locations are wrong by an average of 0.03 size units, or 1.4% of the size of the environment. These experiments show that when there is no observation noise, the sensor parameters and trajectories are recovered very accurately, even if the target’s true dynamics don’t match those used in estimation.

## 6 Real Data

We implemented our algorithm on a network of wireless PDAs equipped with cameras. The PDAs were mounted on the ceiling in an indoor lab environment. The camera image planes are approximately parallel to the ground-plane so that computing the local ground-plane location of the person does not require any calibration beyond finding the focal length of the cameras.

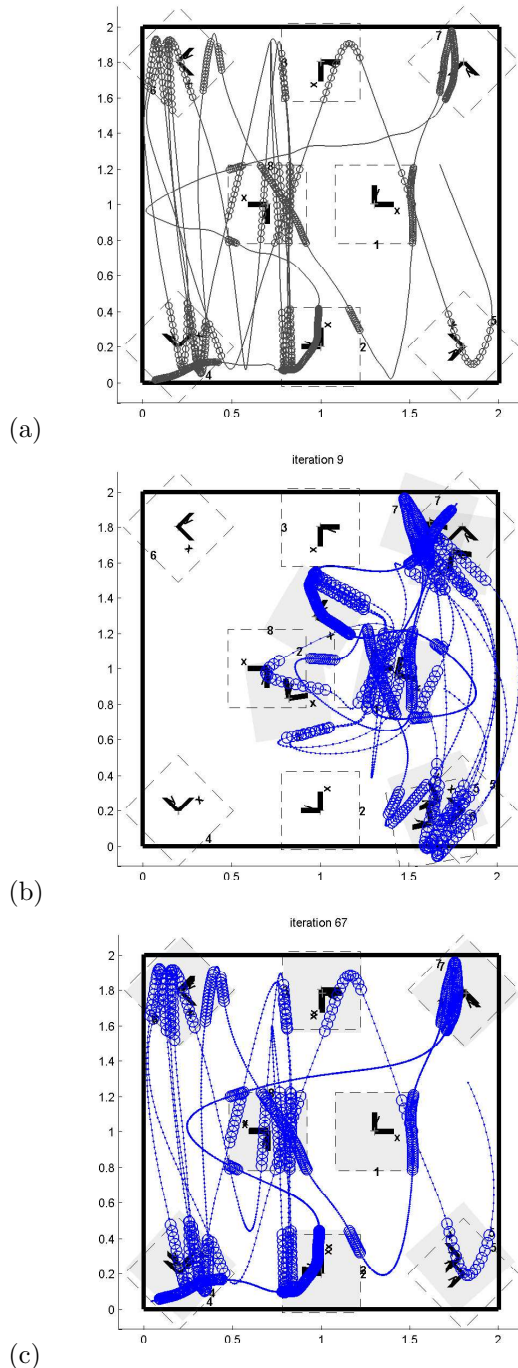


Figure 2: (a) 2,000 steps of a synthetic trajectory. True camera fields of view are depicted as dashed squares. Circles along the trajectories indicate time steps in which a synthetic camera observes the target’s location relative to its coordinate system. (b) After 9 iterations of the optimization procedure. The gauge is fixed by fixing sensor 1 at its true location and orientation. The blue (dark) path denotes the recovered trajectory. Gray squares are the recovered sensor fields of view. (c) Convergence after about 65 iterations. The sensor locations are estimated correctly.

A real-time person tracker runs on each PDA and reports to a base station the time-stamped location of a person with respect to the camera’s ground plane coordinate system every 250 ms. The person tracker uses background subtraction to extract the target and clusters the foreground pixels to compute the person’s location. The individual trackers do not need to filter or smooth the data, as the dynamics model automatically regularizes trajectories during the optimization procedure.

In our first experiment, we installed 4 cameras in an open area in our building. The fields of view of the cameras were about 1.5 meters on each side, and the cameras were 3-4 meters apart. One person walked between the cameras at varying velocities and served as the calibration target. Figure 3(a) illustrates the setup. The ground truth camera parameters were found by measuring the location and orientation of every camera by hand. To render the trajectory of the target in this figure, we set  $\mu$  to these ground truth camera parameters and optimized (4) for the trajectory only.

We used the same parameters for the dynamics model as in the synthetic case. The initial iterate for the optimizer was also the same as in the synthetic case. We set the observation noise  $\sigma_y^2$  to be very small, a factor of  $10^5$  smaller than the driving noise of the velocity. Figures 3(b-c) show the recovered trajectory and sensor locations. On average, the sensor were misplaced by 28 cm from the locations measured by hand. Cameras *ipaq3* and *ipaq10* are off by 50 cm. The rotation of *ipaq3* is off by  $8^\circ$ . That of *ipaq8* by  $9^\circ$  degrees. *ipaq10*’s rotation is off by  $0.7^\circ$ .

In the second experiment, we instrumented the hallway to the left of the open area with 2 additional cameras. The setup and results appear in Figure 4(a). The only way to reach *ipaq5* and *ipaq6* was to take a sharp right below *ipaq7*, outside its FOV. Because no camera ever witnessed this sharp right, and because straight trajectories are slightly favored by our dynamics model over turns, the estimated configuration did not recover the turn. Adding *ipaq9* (Figure 4(b)) provided enough information to recover the turn. This is because *ipaq7*, *ipaq9*, and *ipaq5* form a triangle, and the angle 7-9-5 becomes constrained.

In contrast to the velocity extrapolation idea discussed earlier, our dynamics model allows curved trajectories. Figure 5 shows that our method works even when people take a sharp turn when entering the FOV of *ipaq3*. Using velocity extrapolation, the distance between *ipaq7* and *ipaq3* was found to be 518 cm, whereas in reality, it was only 423 cm. Our method recovered this distance more accurately to be 415 cm.

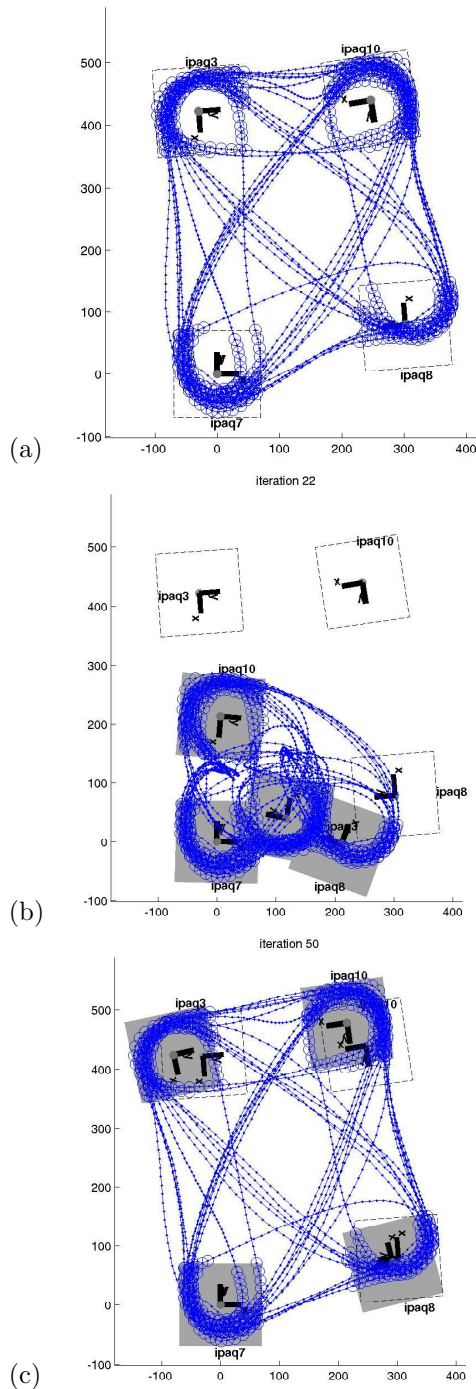


Figure 3: (a) The trajectory for the first experiment. Axes are labeled in centimeters. The coordinate system is fixed on *ipaq7*. (b) After 22 iterations. (c) Convergence after about 50 iterations.

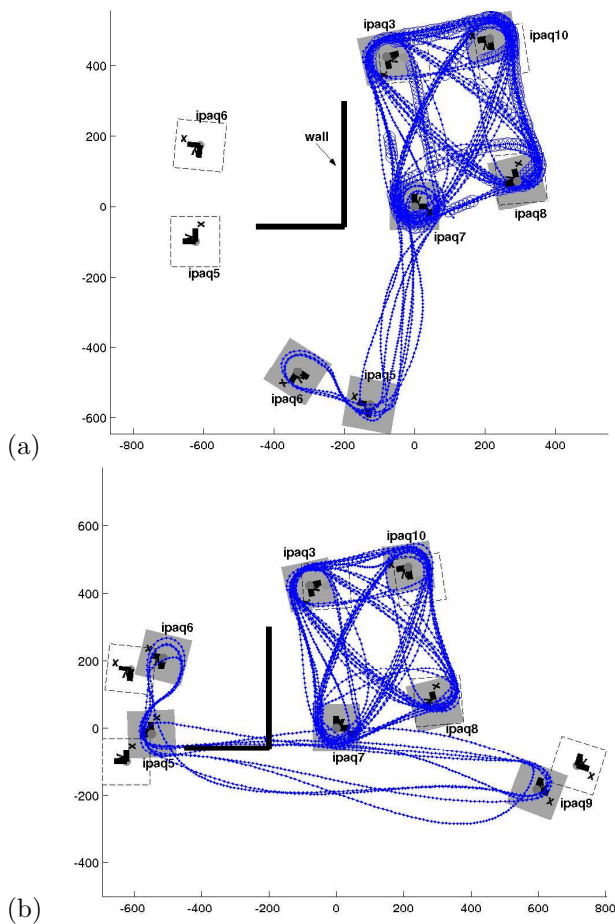


Figure 4: (a) A Wall forces people to take a sharp turn outside the FOV of `ipaq7`. This turn is never witnessed by any camera, so the algorithm does not deduce its existence. (b) Although `ipaq9` doesn't observe the turn directly, its presence provides enough information to determine that `ipaq7` and 6 cannot be below `ipaq7`.

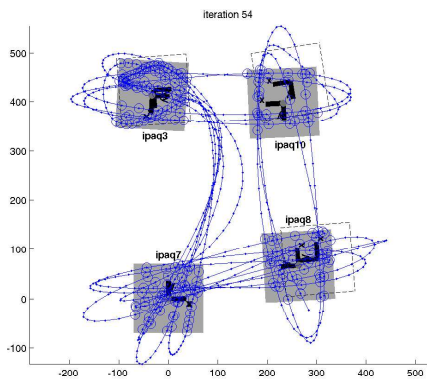


Figure 5: Recovering curved trajectories.

## 7 Conclusion

We have shown how to calibrate a network of non-overlapping cameras. Our solution is framed as a joint MAP estimation camera pose parameters and the trajectory of a calibration target. The prior on the trajectory is a linear Gaussian Markov chain, which allows for both nonlinear paths and speed changes when the target is outside the field of view of the cameras. Information about the target's dynamics allows the system to reason about the behavior of the target when it is not visible, compensating for the lack of overlap between the cameras. We demonstrated our system with a network of battery-operated cameras that are easy to deploy.

## References

- [1] R. B. Fisher. Self-organization of randomly placed sensors. In *European Conference on Computer Vision (ECCV)*, volume 4, pages 146–160, 2002.
- [2] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision (IJCV)*, 4:59–78, January 1989.
- [3] O. Javed, Z. Rasheed, O. Alatas, and M. Shah. *KNIGHT<sup>TM</sup>*: A real time surveillance system for multiple overlapping and non-overlapping cameras. In *International Conference on Multimedia and Expo*, July 2003.
- [4] V. Kettner and R. Zabih. Counting people from multiple cameras. In *ICMCS*, volume 2, pages 267–271, 1999.
- [5] S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *Pattern Analysis and Machine Intelligence (PAMI)*, 25(10), October 2003.
- [6] T.P. Minka. Old and new matrix algebra useful for statistics. Technical report, Media Lab, <http://www.media.mit.edu/~tpminka/papers/matrix.html>, 2001.
- [7] H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1160–1171, 1999.
- [8] C. Stauffer and K. Tieu. Automated multi-camera planar tracking correspondence modeling. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, June 2003.

- [9] G. P. Stein, R. Romano, and L. Lee. Monitoring activities from multiple video streams: Establishing a common coordinate frame. Technical Report AIM-1655, MIT AI Lab, 1999.
- [10] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.
- [11] R.Y. Tsai. Multiframe image point matching and 3-d surface reconstruction. *Pattern Analysis and Machine Intelligence (PAMI)*, 5(2):159–173, March 1983.

## A Optimization Procedure

This section derives the Newton-Raphson steps required to solve (4). We rewrite Equation (4) in nonlinear least squares form, linearize the nonlinearity, and describe one Newton-Raphson iteration.

The dynamics prior term  $\sum_{t=1}^T \|x_t - \mathbf{A}x_{t-1}\|_{\Sigma_\nu}^2$  can be written as a quadratic form in terms of  $x$ , the column vector of stacked states. Denote the Cholesky factor of the inverse covariance  $\Sigma_\nu^{-1}$  of the driving noise in the dynamics model by  $\Sigma_\nu^{-\frac{1}{2}}$ .<sup>1</sup> Define the matrix  $\mathbf{G}$  whose  $t$ th row is:

$$[\mathbf{G}]_t = \left[ 0 \dots, \Sigma_\nu^{-\frac{1}{2}} \mathbf{A}, -\Sigma_\nu^{-\frac{1}{2}}, 0 \dots \right] \quad (5)$$

where the zeros pad the matrix to align its non-zero components with  $x_t$  and  $x_{t+1}$ . Then the dynamics prior term can be rewritten as  $\sum_{t=1}^T \|x_t - \mathbf{A}x_{t-1}\|_{\Sigma_\nu}^2 = x^T \mathbf{G}^T \mathbf{G} x$ .

Equation (4) can now be rewritten as

$$(x^*, \mu^*) = \arg \min_{x, \mu} r(x, \mu)^T r(x, \mu),$$

with

$$r(\mu, x) = \begin{bmatrix} r_y \\ r_x \\ r_\mu \end{bmatrix} = \begin{bmatrix} \sigma_y^{-1} (\mathbf{R}^i (\mathbf{C}x_t - p^i) - y_t^i) \\ \vdots \\ \mathbf{G}x \\ (\mu^1 - \mu_0) \end{bmatrix}.$$

The column vector  $r(\mu, x)$  is partitioned into three sections, corresponding to the three terms in (4). Each

<sup>1</sup>The Cholesky factor of a positive semidefinite matrix  $\mathbf{A}$  is an upper triangular matrix  $\mathbf{C}$  such that  $\mathbf{A} = \mathbf{C}^T \mathbf{C}$ .

measurement  $(t, i) \in \mathcal{Z}$  introduces two elements into  $r_y$ .

To optimize a nonlinear least squares cost function such as  $r^T r$ , Newton-Raphson requires the Jacobian  $\mathbf{J}$  of  $r$ . Newton-Raphson maps an iterate  $\chi^{(t)} = [\mu^{(t)}; x^{(t)}]$  to the next iterate by solving a linear least-squares problem:

$$\chi^{(t+1)} = \chi^{(t)} - \arg \min_{\delta} \|\mathbf{J}\delta - r\|^2. \quad (6)$$

The process of computing  $r$ ,  $\mathbf{J}$ , and applying equation (6) is repeated until  $\chi$  converges to a fixed point.

For completeness, we derive  $\mathbf{J}$  here. It is block structured and very sparse:

$$\mathbf{J} = \nabla r(\mu, x) = \begin{bmatrix} \mathbf{J}_\mu & \mathbf{J}_{\mu x} \\ \mathbf{0} & \mathbf{J}_x \\ \mathbf{I} \ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (7)$$

The left block column of  $\mathbf{J}$  corresponds to differentiating  $r$  with respect to  $\mu$ , and its right block column corresponds to differentiating it with respect to  $x$ .

If the  $z$ th measurement in  $r_y$  came from camera  $i$ , the derivative of  $r_y$  with respect to parameters  $p^i$  and  $\theta^i$  of this camera introduces a block row into  $\mathbf{J}_\mu$  at location  $(z, i)$ .

$$[J_\mu]_{z,i} = -\sigma_y^{-1} \left[ \mathbf{R}^i, ((\mathbf{C}x_t - p^i)^T \otimes \mathbf{I}) \frac{d \text{vec}(\mathbf{R})}{d\theta} \right], \quad (8)$$

where we have used the identity  $\text{vec}(\mathbf{X}\mathbf{Y}) = (\mathbf{Y}^T \otimes \mathbf{I}) \text{vec}(\mathbf{X})$  [6], where  $\otimes$  is the Kronecker product,  $\text{vec}(\cdot)$  stacks elements of a matrix into a column, and  $\mathbf{I}$  is the  $2 \times 2$  identity matrix.

Differentiating the  $z$ th element of  $r_y$  with respect to the observed trajectory element  $x_t$  introduces a block into  $\mathbf{J}_{\mu x}$  at location  $(z, t)$ :

$$[\mathbf{J}_{\mu x}]_{z,t} = \sigma_y^{-1} \mathbf{R}^i \mathbf{C}, \quad (9)$$

where  $i$  is the number of the camera that made the  $z$ th observation.

Differentiating  $r_x$  with respect to  $\mu$  yields 0. Differentiating it with respect to  $x$  yields  $\mathbf{J}_x = \mathbf{G}$ .

$r_\mu$  is only a function of the parameters of the first sensor. Its derivative with respect to  $\mu^1$  is  $\mathbf{I}$ , where  $\mathbf{I}$  is the  $3 \times 3$  identity matrix.