Simultaneous Calibration and Tracking with a Network of Non-Overlapping Sensors

Ali Rahimi Brian Dunagan Trevor Darrell

{ali,bdunagan,trevor}@mit.edu
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA

Abstract

We describe a method for simultaneously recovering the trajectory of a target and the external calibration parameters of non-overlapping cameras in a multi-camera system. Each camera is assumed to measure the location of a moving target within its field of view with respect to the camera's ground-plane coordinate system. Calibrating the network of cameras requires aligning each camera's ground-plane coordinate system with a global ground-plane coordinate system. Prior knowledge about the target's dynamics can compensate for the lack of overlap between the camera fields of view. The target is allowed to move freely with varying speed and direction. We demonstrate the idea with a network of indoor wireless cameras.

1. Introduction

Networks of sensors are often used to instrument large spaces. To track moving people in large environments with as few cameras as possible, we have developed a multi-camera system where the fields of view of the cameras do not overlap. Calibrating such a network is difficult without access to ancillary information such as a map of the environment or additional sensors.

Our main contribution is to show that a model of dynamics for targets can compensate for the lack of overlap between the cameras by allowing us to estimate the target's position when it is not in the field of view of any sensor. Using the dynamics model and sporadic observations from a non-overlapping network of cameras, we can calibrate the camera network and track the target concurrently. We demonstrate our simultaneous calibration and tracking algorithm with a distributed system of wireless camera nodes.

Intuitively, we could estimate the velocity v of the target as it exits the field of view (FOV) of a camera. When, after some time δt, the target enters the FOV of another camera, we could estimate the relative displacement between the two cameras with $v\delta t$. The displacements between pairs of cameras can be combined to recover the relative location and orientation of all the cameras in the network. This velocity extrapolation approach works well when the target moves in a straight line while it is invisible, but does not work well in structured environments such as office buildings where people take sharp turns to enter and exit offices. We refine the velocity extrapolation solution to recover the camera poses and the target's trajectory simultaneously in a batch process that allows curved trajectories.

We recover the calibration parameters of the cameras and the target's trajectory using MAP estimation. Each camera observes the target for a few time steps at a time because the target moves between the fields of view. These observations impose a likelihood over both the trajectory and the calibration parameters. Knowledge about dynamics can be encoded with a prior probability distribution $p(x)$ over trajectories x. For example, a prior $p(\cdot)$ on the dynamics of cars will assign a higher probability to trajectories that obey the laws of physics as they apply to cars than to the trajectory of a pedestrian. One popular way to define $p(\cdot)$ is with a state space model (section 3), though our method can accommodate many smooth models.

We presume that each camera can map the image coordinate of the target to a local coordinate system laid on the ground-plane. Each camera's local ground-plane coordinate system must be aligned to a global ground-plane coordinate system up to a rotation and a translation prior to performing our procedure. We briefly discuss in section 2 how to find a transformation between image coordinates and a local ground-plane coordinate. The aim of this paper is to align these local ground-plane coordinate systems with a global ground-plane coordinate system by recovering the appropriate rotation and translation of each camera.

2. Related Work

We assume that the motion of the target is confined to a ground-plane. Cameras are placed at unknown locations and orientations above the ground-plane, with a portion of the ground-plane visible to each camera. Each camera can map the image plane location of a target to the ground-plane via a homography. Calibrating a network of cameras consists of recovering this homography for each camera.

Outside the computer vision community, ultra-sound or radio links between sensor nodes are the most popular way to recover the pose parameters of the sensors [17, 9, 5]. These approaches triangulate the locations of the sensors by computing pairwise distances between nodes from ultrasound or radio measurements.

Many methods explicitly designed for calibrating networks of cameras rely on overlapping fields of view [19, 20, 12]. Javed et al. [11] is one exception. They use an idea similar to velocity extrapolation to find the projection of the field of view lines of one camera onto another camera. Knowing these projections is tantamount to recovering calibration parameters. Because it is based on velocity extrapolation, this method requires people to walk in a straight line when they are not visible. This paper is most closely related to [10], 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 [10] 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. Our solution works indoors, where simpler solutions such as GPS or radio-based methods do not work reliably.

Most research involving non-overlapping multi-camera trackers has focused on maintaining consistent identity between multiple targets as they exit one field of view and enter another [22, 15]. This is known as the data association problem. These techniques provide machinery for establishing correspondences across disjoint views, but not for determining the real-world trajectory of targets.

To solve the data association problem, one could run the procedure presented in this paper inside an EM iteration that optimizes for the best trajectory and sensor configurations while marginalizing over target labels [8]. We assume that the targets are already labeled using either this, or one of [22, 15]. Once the identity of each target is known, we can treat each trajectory separately. So for the rest of this paper, we assume that the tracking problem is decoupled in this way and we only discuss tracking one target.

There are several methods for recovering the homography between a single camera and the ground-plane. [3] finds the homography that yields the best alignment between the ground-plane locations of known landmarks and the image plane location of the landmarks. [4] avoids relying on landmarks by watching trajectories of objects moving in a straight line with constant velocity. In each case, the problem is ill-posed for a network of cameras, unless the landmarks' locations are globally known, or unless the same linear trajectory crosses all sensors. For a network of cameras, these methods can only recover a mapping between each image plane and a local ground-plane coordinate system for that camera. These local ground-plane coordinate systems are arbitrarily rotated and translated with respect to the global ground-plane coordinate system.

By applying this mapping to the image coordinate of the target, a sensor can transform an image plane coordinate to its local ground-plane coordinate system, allowing the camera to behave like a virtual overhead camera. In the remainder of this paper, we presume that these homographies have been recovered up to a rotation and a translation using a method such as [4] or [3], and focus on aligning the local ground-plane coordinate system of each camera to the global ground plane coordinate system. In section 8, we use overhead cameras, so no further calibration is necessary.

Concurrently recovering calibration parameters and trajectories is reminiscent of SFM (Structure from Motion) and SLAM/CML (Simultaneous Localization and Mapping, a.k.a. Concurrent Mapping and Localization) [6]. In SFM and SLAM, a moving camera measures the image plane coordinate of scene features and recovers the trajectory of the camera and the 3D location of the scene features. Our problem is converse: A set of stationary cameras observe the image-plane coordinate of a moving feature (the target), and the goal is to recover the pose of the cameras and the trajectory of the feature. We cast our problem in an optimization framework similar to bundle adjustment [21].

Various SFM and SLAM techniques, particularly those based on the Kalman Filter [13, 7, 2, 1, 18], have a dynamics model for the motion of the camera. The major difference between our problem and SLAM or SFM is that due to the lack of overlap, at most one camera can observe the target at any time. This corresponds to an SFM setup where the moving camera observes only one scene feature in each frame. Because this is rarely the case in SFM, motion models are merely used to provide robustness against noisy measurements. But as the simple example in the introduction illustrates, a motion model is a critical component of our method.

In SFM, computing the camera trajectory is easy if the feature locations are known, and recovering the scene structure is easy if the camera motion is known. In our problem, finding the camera locations is easy if the target's motion is known, and estimating the target's motion is simple if the cameras are fully calibrated. This suggests an algorithm that begins with a guess for the camera poses, and iterates be-

tween estimating a trajectory and updating the camera poses with this new trajectory. In Structure from Motion, this algorithm is known to work, albeit slowly. We have proved in a technical report that this algorithm does not converge at all in our case [16]. Incidentally, this non-convergence proves that coordinate ascent cannot solve the Structure from Motion problem when only one feature is observed in each frame. Therefore, our algorithm estimates the trajectory and the calibration parameters simultaneously.

3. Trajectory Model

We represent dynamics with a distribution $p(\cdot)$ over the space of trajectories. $p(\cdot)$ will be used as a prior for the trajectory in a MAP framework. To define this function, we begin with a generative model for trajectories.

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. We assume that the state evolves according to linear Gaussian Markov dynamics:

$$x_{t+1} = \mathbf{A}x_t + \nu_t, \tag{1}$$

where ν_t is a zero mean Gaussian random variable with covariance $\mathbf{\Sigma}_\nu$.

The states form a Markov chain over time. x, the collection of states from time 1 to time T is itself a Gaussian random variable of dimension $1 \times 4T$:

$$\begin{aligned} p(x_t|x_{t-1}) &= \mathcal{N}\left(x_t \middle| \mathbf{A}x_{t-1}, \mathbf{\Sigma}_\nu\right) \\ p(x) &= \prod_{t=1}^{T} p(x_t|x_{t-1}) = \mathcal{N}\left(x \middle| 0, \mathbf{\Lambda}_x\right). \end{aligned} \tag{2}$$

The prior $p(x)$ is the probability that a given location and velocity trajectory emanated from our generative model of target dynamics.

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

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{\Sigma}_\nu &= \operatorname{diag}\left(\begin{bmatrix} 10^{-4} & 1 & 10^{-4} & 1 \end{bmatrix}\right) \end{aligned} \tag{3}$$

so that each x_{t+1} adds the velocities in x_t to the positions of x_t, and nudges the old velocities by Gaussian noise. The resulting poses are also nudged by a small amount of Gaussian noise. To extract the location components of x_t, we can multiply x_t by $\mathbf{C}$:

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

4. Observation Model

Measurements from the camera network provide the likelihood function over trajectories and camera parameters. When the target's location x_t falls within the field of view of camera i, the camera sees x_t in its own ground-plane coordinate system. This coordinate system is rotated by θ^i and translated by p^i with respect to the global ground-plane coordinate system. Let $\mu^i = \left[p^i; \theta^i\right]$ be the parameters of camera i, and we define $\mu = \left[\mu^1; \cdots; \mu^N\right]$ to be the collection of all the camera parameters. Let $\mathcal{Z}$ be a set of (t, i) pairs, where $(t, i) \in \mathcal{Z}$ iff camera i sees the target at time t.

With $\mathbf{R^i}$ the rotation matrix corresponding to a rotation of $-\theta^i$, a camera observing the target at x_t reports

$$y_t^i = \pi^i(x_t) + \omega_t = \mathbf{R}^i(\mathbf{C}x_t - p^i) + \omega_t, \tag{4}$$

where ω_t is white Gaussian noise with covariance $\sigma_y^2\mathbf{I}$. Equation (4) describes a likelihood function for the calibration parameters and the trajectory:

$$p(y_t^i|x_t, \mu^i) = \mathcal{N}\left(y_t^i|\pi^i(x_t), \sigma_y^2\mathbf{I}\right).$$

These measurements are independent of each other conditioned on x_t and μ^i. Given the trajectory and calibration parameters, the probability of $y = \{y_t^i|(t, i) \in \mathcal{Z}\}$, the collection of all measurements from all sensors, is:

$$p(y|x, \mu) = \prod_{(t,i)\in\mathcal{Z}} p(y_t^i|x_t, \mu^i). \tag{5}$$

The absence of a pair (t, i) from $\mathcal{Z}$ is informative. It says that at time t, the target was out of the field of view of all the sensors. However, we do not count the absence of an observation as a measurement, even though this type of field of view constraint could improve the recovered trajectory.

5. Finding the MAP configuration

The most a posteriori probable trajectory and calibration parameters are:

$$(x^*, \mu^*) = \arg\max_{x,\mu} p(y|x, \mu)p(x)p(\mu), \tag{6}$$

with the trajectory prior $p(x)$ defined by equation (2) and $p(y|x, \mu)$ defined by equation (5). $p(\mu)$ is a prior on the calibration parameters.

Any configuration of x and μ is equivalent to the same configuration arbitrarily rotated and translated. In Structure from Motion, this is known as a gauge freedom [21]. The prior $p(\mu)$ fixes the first sensor at the origin of the global ground-plane coordinate system, and points it in a given direction. If any other information about the sensor parameters is available a priori, for example, from GPS or other sensor localization algorithms, it can be encoded $p(\mu)$ as well.

The optimization of equation (6) is a non-linear least-squares problem over the variables μ and x:

$$\begin{aligned}(x^*, \mu^*) &= \arg\max_{x,\mu} \log p(y|x,\mu)p(x)p(\mu) \\ &= \arg\min_{x,\mu} \sum_{(t,i)\in\mathcal{Z}} \frac{1}{\sigma_y^2}\|y_t^i - \pi^i(x_t)\|^2 \qquad (7) \\ &\quad + x^{\mathrm{T}}\mathbf{\Lambda}_x^{-1}x + \frac{1}{\sigma_\mu^2}\|\mu^1 - \mu_0\|^2\end{aligned}$$

To find the optimal μ and x we use Newton-Raphson. Newton-Raphson produces a sequence of iterates $\chi^{(t)} = [\mu^{(t)}; x^{(t)}]$ that often converges to the bottom of the cost function. In the case of non-linear least squares, it iteratively linearizes the non-linearity inside the L2 norm and solves a linear least squares problem. Appendix A derives the quantities needed by each Newton-Raphson step.

6. Multiple Trajectories

So far, we have discussed recovering camera parameters by observing a single long trajectory. Long trajectories are difficult to obtain in real tracking systems because it is difficult to consistently label targets over long stretches of time. To ease the burden on the data association algorithm, we can process several shorter trajectories instead. In a future paper, we will investigate using the following method for handling multiple trajectories.

Let $\mathbf{X} = [X_1..X_M]$ be a collection of M trajectories of differing lengths. Let $\mathbf{Y} = [Y_1..Y_M]$ be the corresponding measurements made by the cameras for each of these trajectories. Assume that each X_m is a priori independent of the others. Because given X_m and μ the Y_m are independent, the best reconstruction of the sensor parameters μ and of the trajectories $\mathbf{X}$ is:

$$\begin{aligned}(\mathbf{X}, \mu) &= \arg\max_{\mathbf{X},\mu} p(X_1..X_M, \mu|Y_1..Y_M) \\ &= \arg\max_{\mathbf{X},\mu} p(\mu)\prod_{m=1}^{M} p(Y_m|X_m,\mu)p(X_m)\end{aligned}$$

This optimization can again be carried out with Newton-Raphson as before. More trajectories naturally result in better estimates for μ, and through μ, yield better trajectory estimates.

7. Synthetic Results

We simulated a point target traveling in a square environment with elastic walls. The target's trajectory was generated by a random walk that reflected off of walls. The resulting trajectory was smoothed to yield the setup shown in Figure 1(a). The synthetic cameras are depicted as triangular fields of view that can measure the location of the target without noise.

We used the $\mathbf{A}$ and $\mathbf{\Sigma}_\nu$ matrices of equation (3) to define the dynamics model. Notice that the generative model described in section 3 was not the model used for generating the synthetic trajectory. Our dynamics model does not model the bouncing behavior near walls. Our goal with these synthetic experiments was to see if such a large disparity between the dynamics model and actual target behavior had adverse effects on trajectory estimation and calibration.

The Newton-Raphson iterations begin with an initial guess for the configuration with all trajectory points and sensors at the origin, pointing to the right. Figures 1(b-c) show a few iterations 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.

8. Real Data

We have implemented this system on a network of wireless PDAs equipped with cameras. The PDAs were mounted on the ceiling in an indoor lab environment (see Figure 2). 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. 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 our optimization procedure already regularizes trajectories with the dynamics model.

In our first experiment, we install 4 cameras in an open area in our building. The fields of view of the cameras are about 1.5 meters on each side, and the cameras are 3-4 meters apart. One person traced a long trajectory, walking between the cameras at varying velocities. Figure 3(a) illustrates the setup. The ground truth μ was found by computing the FOVs of every camera and measuring their locations and orientations by hand. To find the trajectory, we set μ to the true sensor locations and optimized (6) for the trajectory only.

We used the same dynamics model as in the synthetic case, without any modification. The initial condition was the same as before. We set σ_y^2 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

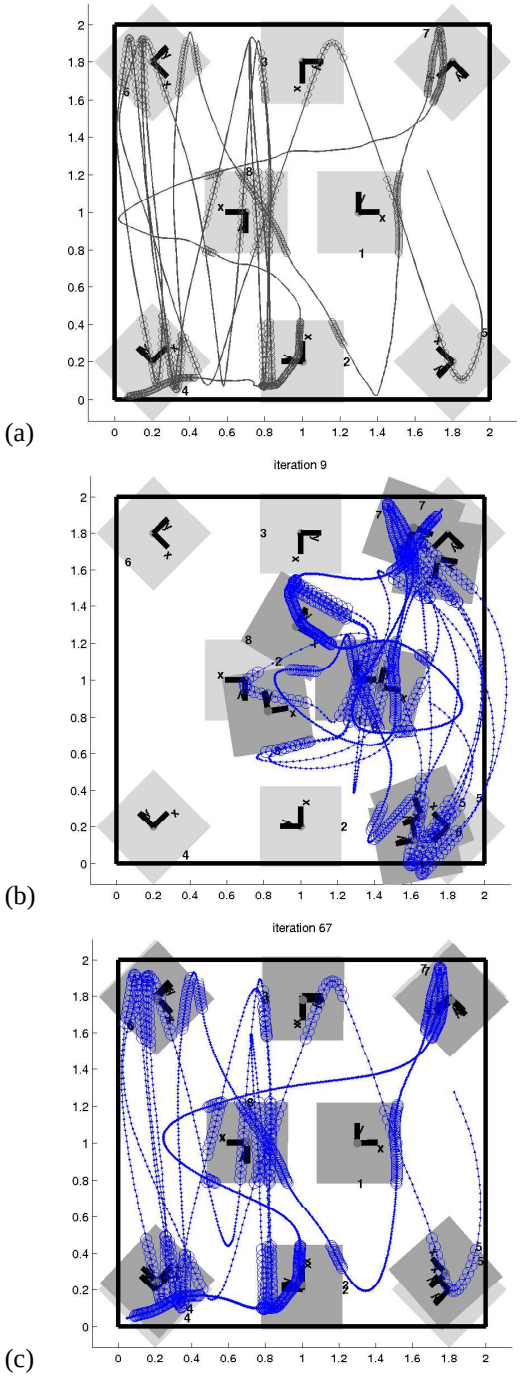


Figure 1: (a) 2,000 steps of a synthetic trajectory. Sensor fields of view are depicted as squares. Circles along the trajectories indicate time steps in which a synthetic camera measures the target's location relative to its coordinate system. (b) After 9 iterations. The gauge is fixed by fixing sensor 1 at its true location and orientation. Blue denotes the recovered trajectory. Gray are the recovered sensor fields of view. Dashed squares are the ground truth sensor locations from (a). (c) Convergence after about 65 iterations. The sensor locations are estimated correctly.

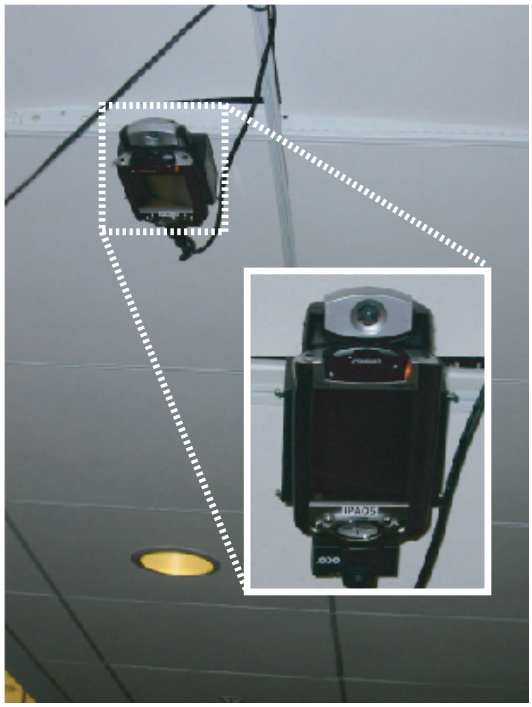


Figure 2: We use Compaq IPAQs as wireless camera nodes in our network. The IPAQs are mounted on the ceiling, with their camera image plane parallel to the ground plane.

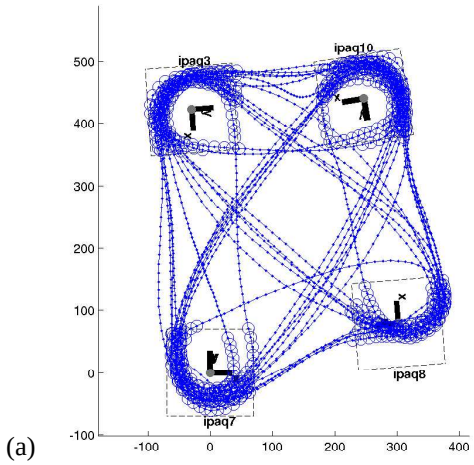
cm. The rotation of ipaq3 is off by $8°$. That of ipaq8 by $9°$ degrees. Ipaq10's rotation is off by $0.7°$.

In the second experiment, we instrumented the hallway to the south 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 by taking 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, the recovered configuration did not exhibit the turn. Adding ipaq9 (Figure 4(b)) provided enough information to recover the turn. This is because ipaqs7,9, and 5 form a triangle, and the angle 5-7-9 becomes constrained.

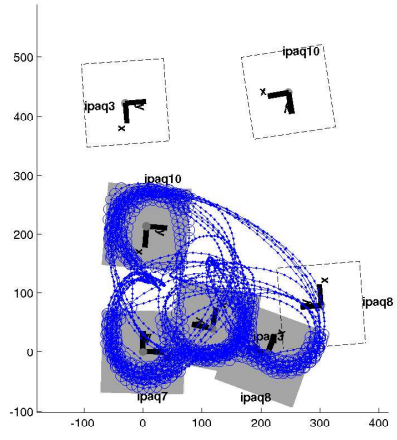
In contrast to the velocity extrapolation idea discussed earlier, our dynamics model allows curved trajectories. Figure 5 shows that our method works even when people take a sharp turn when entering the FOV of ipaq3. Using velocity extrapolation, the distance between ipaq7 and ipaq3 was found to be 518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

9. Conclusion

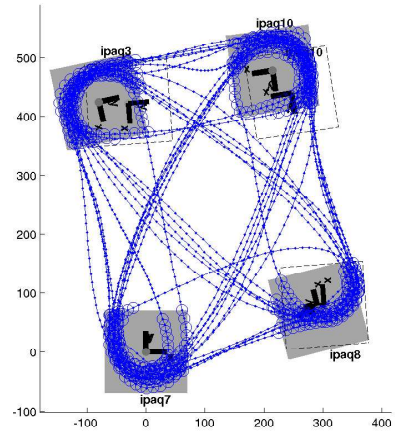
We have shown how to calibrate and track with a network of non-overlapping cameras. Our solution is framed as a joint MAP estimation over trajectories and camera pose parame-



(a)

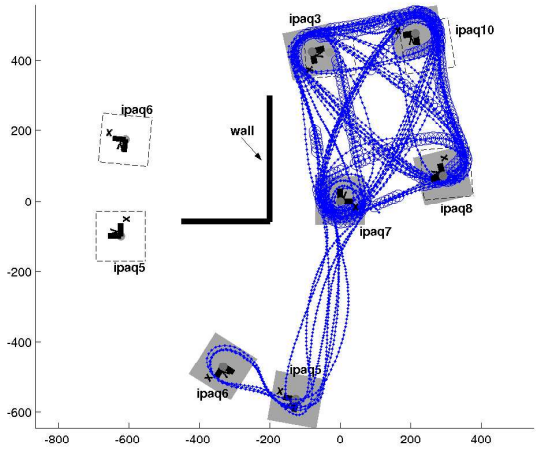


(b)

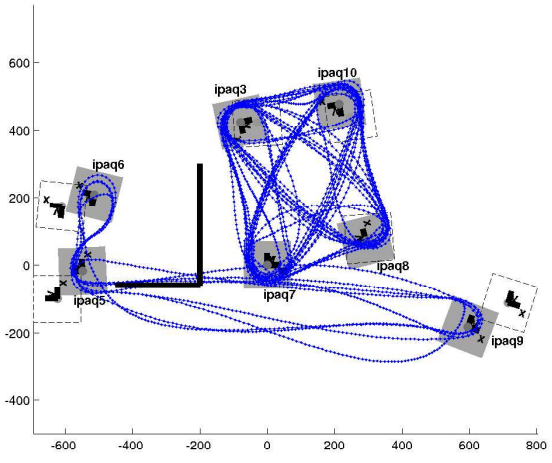


(c)

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.



(a)



(b)

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.

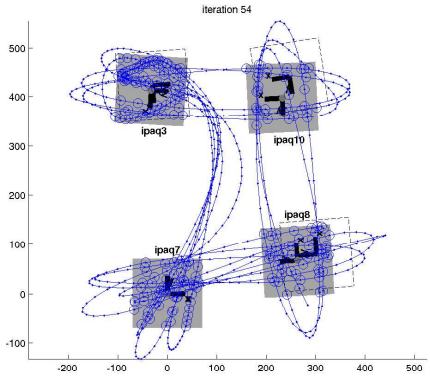


Figure 5: Recovering curved trajectories.

ters. The prior on the trajectory is a linear Gaussian Markov chain. This rather general model of dynamics accounts for both non-linear paths and speed changes even when the target is not visible. Information about the target's dynamics allows the system to reason about the behavior of the target when it is not visible.

We demonstrated our system with a network of battery-operated cameras that are easy to deploy. To ease the burden on the data-association algorithm, we have made provisions for operating on short trajectory segments instead of one long trajectory.

In the future, we plan to adopt a switching linear dynamics model to account for different regimes such as running, walking, stopping, taking a sharp turn, etc.

A. Finding the peak

This section derives the Newton-Raphson steps required to solve (7).

Denote by $\sqrt{\mathbf{\Sigma}_\nu}$ the Cholesky factor of the covariance $\mathbf{\Sigma}_\nu$ of the driving noise in the dynamics model[1]. We will also need the Cholesky factor $\mathbf{G}$ of $\mathbf{\Lambda}_x^{-1}$, the inverse covariance of the entire trajectory. The tth row of the $\mathbf{G}$ is:

$$[\mathbf{G}]_t = \left[\begin{array}{ccccc} 0 \cdots, & \sqrt{\mathbf{\Sigma}_\nu}\mathbf{A}, & -\sqrt{\mathbf{\Sigma}_\nu}, & 0 \cdots \end{array} \right] \quad (8)$$

where $[\mathbf{G}]_t$ is padded with zero to align its non-zero components with x_t and x_{t+1}. $\mathbf{G}$ is easily derived from the quadratic form inside $p(x)$ defined in equation (2).

To maximize (7), rewrite it as

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

with

$$r(\mu, x) = \left[\begin{array}{c} r_y \\ \hline r_x \\ \hline r_\mu \end{array} \right] = \left[\begin{array}{c} \sigma_y^{-1}\left(\mathbf{R}^i(\mathbf{C}x_t - p^i) - y_t^i\right) \\ \vdots \\ \hline \mathbf{G}x \\ \hline \sigma_\mu^{-1}(\mu^1 - \mu_0) \end{array} \right].$$

$r(\mu, x)$ is partitioned into three sections, corresponding to the different terms in (7). Each measurement $(t, i) \in \mathcal{Z}$ introduces an element into r_y.

To optimize a non-linear least squares cost function such as $r^{\mathrm{T}}r$, 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:

$$\chi^{(t+1)} = \chi^{(t)} - \arg\min_\delta \|\mathbf{J}\delta - r\|^2. \quad (9)$$

[1]The Cholesky factor of a positive definite matrix $\mathbf{A}$ is an upper triangular matrix $\mathbf{C}$ such that $\mathbf{A} = \mathbf{C}^{\mathrm{T}}\mathbf{C}$.

The process of computing r, $\mathbf{J}$, and applying equation (9) is repeated until until χ converges to a fixed point.

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

$$\mathbf{J} = \nabla r(\mu, x) = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{\mathbf{J}_\mu} & \mathbf{J}_{\mu\mathbf{x}} \\ \hline \multicolumn{2}{c|}{\mathbf{0}} & \mathbf{J}_{\mathbf{x}} \\ \hline \sigma_\mu^{-1}\mathbf{I} & \mathbf{0} & \mathbf{0} \end{array} \right].$$

The left block column of $\mathbf{J}$ corresponds to differentiating r with respect to μ, and its right block column corresponds to differentiation with respect to x. Like r, $\mathbf{J}$ is also partitioned vertically.

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

$$[J_\mu]_{z,i} = -\sigma_y^{-1}\left[\begin{array}{cc} \mathbf{R}^i, & \left((\mathbf{C}x_t - p^i)^{\mathrm{T}} \otimes \mathbf{I}\right)\frac{d\,\mathrm{vec}(\mathbf{R})}{d\theta} \end{array} \right],$$

where we have used the identity $\mathrm{vec}\,(\mathbf{XY}) = (\mathbf{Y}^{\mathrm{T}} \otimes \mathbf{I})\,\mathrm{vec}\,(\mathbf{X})$ [14], where $\otimes$ is the Kronecker product, $\mathrm{vec}\,(\cdot)$ stacks elements of a matrix into a column, and $\mathbf{I}$ is the 2×2 identity matrix.

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

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

where i is the number of the camera that made the zth observation.

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

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

References

[1] N. Ayache and O. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Tran. Robot. Automat.*, 5(6):804–819, 1989.

[2] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *PAMI*, June 1995.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *CVPR*, 1997.

[4] B. Bose and E. Grimson. Ground plane rectification by tracking moving objects. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2003.

[5] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. Technical Report Technical report 00-729, University of Southern California, Computer science department, April 2000.

[6] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tards. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Trans. Robot. Automat.*, 15:948–953, October 1999.

[7] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from motion causally integrated over time. *PAMI*, 24(4):523–535, April 2002.

[8] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun. Structure from motion without correspondence. In *CVPR*, June 2000.

[9] L. Doherty, K. S. J. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *INFOCOM*, volume 3, pages 1655 –1663, 2001.

[10] R. B. Fisher. Self-organization of randomly placed sensors. In *ECCV*, volume 4, pages 146–160, 2002.

[11] O. Javed, Z. Rasheed, O. Alatas, and M. Shah. $KNIGHT^{TM}$: A real time surveillance system for multiple overlapping and non-overlapping cameras. In *International Conference on Multimedia and Expo*, July 2003.

[12] S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10), October 2003.

[13] Philip F. McLauchlan. A batch/recursive algorithm for 3D scene reconstruction. *Conf. Computer Vision and Pattern Recognition*, 2:738–743, 2000.

[14] 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.

[15] H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *IJCAI*, pages 1160–1171, 1999.

[16] A. Rahimi. Coordinate ascent cannot recover trajectories and calibration parameters in a sparse sensor network. Technical report, *MIT CSAIL*, September 2003.

[17] C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conference*, June 2002.

[18] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Uncertainity in Artificial Intelligence*, 1988.

[19] C. Stauffer and K. Tieu. Automated multi-camera planar tracking correspondence modeling. In *CVPR*, volume 1, June 2003.

[20] 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.

[21] 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.

[22] R. Zabih V. Kettnaker. Counting people from multiple cameras. In *ICMCS*, volume 2, pages 267–271, 1999.