Tracking People with a Sparse Network of Bearing Sensors

A. Rahimi, B. Dunagan, and T. Darrell

MIT Computer Science and Artificial Intelligence Laboratory,
200 Technology Square,
Cambridge MA, 02139 USA
{ali,bdunagan,trevor}@mit.edu

Abstract. Recent techniques for multi-camera tracking have relied on either overlap between the fields of view of the cameras or on a visible ground plane. We show that if information about the dynamics of the target is available, we can estimate the trajectory of the target without visible ground planes or overlapping cameras.

1 Introduction

We explore the problem of tracking individuals using a network of non-overlapping cameras. Recent techniques for multi-camera tracking have relied on two kinds of cues. Some rely on overlap between the fields of view of the cameras to calculate the real-world coordinate of the target. Others assume that the ground plane is visible and map image points known to lie on the ground plane to the real world using homography. In this paper, we show that if information about the dynamics of the target is available, we can estimate trajectories without visible ground planes or overlapping cameras.

We are interested in instrumenting as large an environment as possible with a small number of cameras. To maximize the coverage area of the network, the camera fields of view (FOVs) rarely overlap. Further, in our indoor setting, we wish to use cameras that may not have a clear view of the ground plane due to occlusions or their horizontal orientation.

Without a visible ground plane, each camera can only estimate the bearing of the ray from the camera optical center to the target. Therefore the target's location can only be determined up to a scale factor with a single camera. However, information about a target's dynamics can be helpful in localizing it. For example, if a target is known to be moving at a given constant speed in the ground plane, its location can be fully recovered by matching its speed in the image plane to its ground-plane speed. See Figure 1(a).

If the target's speed is unknown but constant, it's trajectory can be estimated with two non-overlapping cameras. See Figure 1(b). The time interval in which the target leaves the first field of view and enters the second one is inversely proportional to the velocity of the target. Using the second camera helps us recover the speed, which in turns allows us to localize the target.

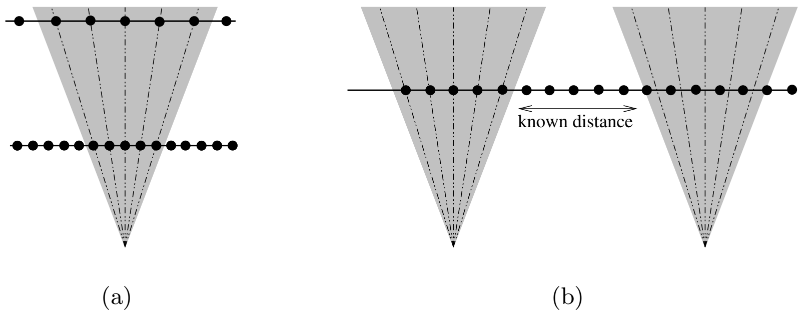

Fig. 1. (a) A horizontally mounted camera recovers the location of a target up to a scale factor. If the target's speed is known, the ambiguity disappears. For example, given its true speed, if the target is seen to be moving slowly, it must be far away. If it appears to be moving fast, it must be near. (b) A second camera provides enough information to estimate the (constant) speed of the target, making it possible to localize it.

The cases where speed is constant and known, or constant and unknown, are straightforward to handle. In this paper, we generalize to the case where the target moves with varying but smooth velocity. These dynamics can be modeled with a Gauss-Markov process. Given the dynamics of the target, we search for a trajectory that is most compatible with these dynamics and the observations made by the cameras. The resulting trajectories capture the gross features of the motion of the target and use the dynamics to interpolate sections of the trajectory not observed by the cameras.

We incorporate the smoothness of the trajectory as a prior in the Bayesian framework (§3). The camera measurements will provide observations that define a likelihood on trajectories (§4). The Maximum a Posteriori (MAP) trajectory can be recovered by iteratively solving a quadratic program (§5). We validate our system on both synthetic and real data (§6 and §7).

2 Related Work

Recently, there has been a significant amount of work in tracking people across multiple views. Some of the proposed approaches seek to hand off image-based tracking from camera to camera without recovering real-world coordinates [1]. We focus on those that recover the real-world coordinate of the person. Multi-camera person trackers can be categorized as overlapping systems and non-overlapping systems.

Tracking with overlapping cameras has relied on either narrow baseline stereo [4, 3] or wide baseline matching [5]. These methods use the correspondence across views to determine the location of the target in the real world.

Most research with non-overlapping cameras has focused on maintaining consistent identity between multiple targets as they exit one field of view and enter another [8, 6]. This is known as the data association problem. These techniques cannot help determine the real-world position of a target if individual cameras cannot make this determination individually. They provide machinery for establishing correspondences across disjoint views, but not for localization.

Caspi and Irani [2] provided another example where correspondence could be avoided. They showed how to align a pair of image sequences acquired from non-overlapping cameras, when the object being imaged spans the field of views of two non-overlapping cameras. The coherent motion of a planar object as seen by two nearby cameras can compensate for the lack of correspondence. In our case, coherent target dynamics provide this kind of coherence.

3 Trajectory Model

We assume that each camera can identify each person in its field of view from frame to frame. This allows us to track individuals independently of each other. For the rest of this paper, we assume that the tracking problem is decoupled in this way and we only discuss tracking each individual separately. This system could be augmented by more sophisticated person identification schemes than the one described in §7, such as the ones discussed in section §2.

We use a linear Gaussian state-space model to describe the smoothness of the trajectory on the ground plane. This will define a prior $p(X)$ on the trajectory to be estimated.

Define the state x_t of the target at time t as:

$$x_t = \begin{bmatrix} u_t \ \dot{u}_t \ v_t \ \dot{v}_t \end{bmatrix}^\top .$$

Where u_t and v_t are the x and y locations of the target on the ground plane, and $\dot{u}_t$ and $\dot{v}_t$ describe the target's instantaneous velocity. We assume that the state evolves according to linear Gaussian Markov dynamics:

$$x_{t+1} = Ax_t + \nu_t, \tag{1}$$

where ν_t is a zero-mean Gaussian random variable with covariance Σ_ν. For example, in the synthetic example of §6 we set

$$A = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ \Sigma_v = 10^{-6}\text{diag}\left(\begin{bmatrix} 10^{-4} \ 1 \ 10^{-4} \ 1 \end{bmatrix}\right),$$

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. The resulting poses are also nudged by a small amount of Gaussian noise.

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 $1 \times 4T$:

$$p(x_t|x_{t-1}) = \mathcal{N}\left(x_t \middle| Ax_{t-1}, \Sigma_\nu\right)$$

$$p(X) = \prod_{t=1}^{T} p(x_t|x_{t-1}) = \mathcal{N}\left(X \middle| 0, \Lambda_X\right), \tag{2}$$

where Λ_X^{-1} is tri-diagonal. We will use the Cholesky decomposition of Λ_X^{-1} in §5:

$$\begin{aligned} \Lambda_X &= G^\top G \\ G_t &= \left[0 \cdots, \sqrt{\Sigma_\nu} A, -\sqrt{\Sigma_\nu}, 0 \cdots \right], \end{aligned} \tag{3}$$

where G_t is row t of G and $\sqrt{\Sigma_\nu}$ is the Cholesky factor of Σ_ν. Equation (3) is easily derived from the quadratic form inside $p(X)$ defined in equation (2).

4 Observation Model

In general, we wish to consider an object tracked by a set of oblique non-overlapping cameras with no visible ground plane. To simplify our task, we consider the case where cameras are mounted horizontally, so that only the horizontal direction in the image plane is relevant in locating a person. In indoor settings, horizontal cameras are a way to cover a large area.

Let p^i be the location of camera i on the ground plane with respect to some reference, and let θ^i its rotation (yaw). Denote the focal length of the camera by f^i.

Ideally, the width of each person could be used to gauge the distance to the target. But our system uses background subtraction, which yields crude segmentation, partly because both the moving region and the uncovered region are identified as foreground pixels. Therefore we ignore the width of the clusters as a depth cue.

Let y_t^i be the horizontal location of the target as seen in the image plane of camera i at time t. This measurement is the bearing of the target with respect to the camera, and is computed by projecting the target's location onto the camera's focal plane:

$$y_t^i = \pi^i(x_t) + \omega_t = f^i \frac{\mathbf{R}_y^i(Cx_t - p^i)}{\mathbf{R}_x^i(Cx_t - p^i)} + \omega_t \tag{4}$$

$$C = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix}. \tag{5}$$

Here, $\mathbf{R}^i$ is the rotation matrix corresponding to θ^i, and ω_t is a zero mean Gaussian random variable with variance σ_ω^2. C is a 2×4 matrix that extracts the location of the target from its state.

When the target is within the field of view of a camera, (4) describes a likelihood model for each measurement. When the target is out of the field of view of a camera, that camera reports $\emptyset$:

$$p(y_t^i|x_t) = \begin{cases} \mathcal{N}\left(y_t^i|\pi^i(x_t), \sigma_\omega\right), & \text{if } \mathcal{I}^i(x_t) \\ \delta(y_t^i - \emptyset), & \text{otherwise.} \end{cases}$$

where $\mathcal{I}^i(x)$ is an indicator function that determines whether a given point falls within the field of view of camera i.

Conditioned on the true location of the target, the measurements are independent of each other. Then letting Y be the collection of all measurements from all sensors from time $t = 1$ to $t = T$,

$$p(Y|X) = \prod_{t=1}^{T} \prod_{i=1}^{N} p(y_t^i|x_t).$$

It will be useful to decompose the likelihood into constraints and observations:

$$p(Y|X) = \prod_{(t,i)\in\mathcal{O}} \mathcal{N}\left(y_t^i|\pi^i(x_t), \sigma_\omega\right) \prod_{(t,i)\notin\mathcal{O}} (1 - \mathcal{I}^i(x_t))$$

where $(t,i) \in \mathcal{O} \iff y_t^i \neq \emptyset$. The second product of factors is a set of constraints that insure the estimated trajectory only goes through fields of view that have actually seen the target. In §5, we will need to use these constraints in a quadratic program. However, quadratic programming requires a convex constraint set, and these constraints are not convex. So we relax them the constraints $\mathcal{I}^i$ by defining the function $\mathcal{J}^i$ so that $\mathcal{J}^i(x_t) = 0$ if x_t is behind camera i, and 1 if it is in front of it. The new likelihood becomes:

$$p(Y|X) = \prod_{(t,i)\in\mathcal{O}} p(y_t^i|x_t) \prod_{(t,i)\in\mathcal{O}} \mathcal{J}^i(x_t)$$

As is shown in the following section, this new constraint set is convex. It says that every measurement must have emanated from a sensor that had the target in front of it, but it does not penalize the situation where a trajectory crosses a field of view without generating an observation. We use this new, more permissive likelihood function to find the MAP trajectory.

5 MAP Trajectory Estimation

The most probable trajectory given all the camera observations is

$$X^* = \arg\max_X p(X|Y) = \arg\max_X p(X)p(Y|X) \tag{6}$$

A local maximum can be found by iteratively approximating this optimization problem with a quadratic program of the form:

$$\begin{aligned} X^* = \arg\max_X & \; X^\top Q X + C^\top X \\ \text{s.t.} & \; AX > b \end{aligned} \tag{7}$$

We show how to transform the optimization of equation (6) into a sequence of quadratic programs of the form of equation (7).

Taking the log of $p(X)p(Y|X)$ and dropping terms that don't depend on X yields a new quantity to minimize:

$$E(X) = X^\top G^\top G X + \frac{1}{\sigma_\omega^2} \sum_{(t,i)\in\mathcal{O}} (\pi^i(x_t) - y_t)^2 + \sum_{(t,i)\in\mathcal{O}} \log \mathcal{J}^i(x_t)$$

The last term serves as a constraint, so the maximization of E over X can take the form of a non-linear least-squares program:

$$\begin{aligned} X^* &= \arg\min_X \epsilon(X) = \arg\min_X \frac{1}{2} r(X)^\top r(X) \\ &\text{s.t. } \forall_{(t,i)\in\mathcal{O}} \mathcal{J}^i(x_t) = 1 \end{aligned}$$

where

$$r(X) = \begin{bmatrix} GX \\ \vdots \\ \frac{\pi^i(x_t)-y_t}{\sigma_\omega} \\ \vdots \end{bmatrix}, \quad (i,t) \in \mathcal{O}.$$

Each constraint $\mathcal{J}^i(x_t) = 1$ can be recast as a linear constraint for each observed point. Figure 2 shows that a point is in front of the sensor if its projection onto the camera optical axis is positive. Each field of view constraint becomes an inequality constraint:

$$\mathcal{J}^i(x_t) = 1 \iff n(\theta^i)^\top (x_t - p^i) > 0,$$

where $n(\theta^i)$ is a vector pointing along the optical axis of camera i. Let the rows of matrix A and the elements of vector b be:

$$\begin{aligned} A_\tau &= \begin{bmatrix} 0 \cdots & \cos(\theta^i) \; 0 \; \sin(\theta^i) \; 0 \cdots \end{bmatrix}, \\ b_\tau &= n(\theta^i)^\top p^i, \end{aligned}$$

with one row per observed trajectory point. The non-linear program becomes a non-linear program on a convex domain:

$$X^* = \arg\min_X \epsilon(X) \tag{8}$$

$$\text{s.t. } AX > b. \tag{9}$$

To convert equations (8,9) into a quadratic program, we linearize $r(X)$ about a guess X_0

$$\epsilon(X) \approx \|r(X_0) + J(X - X_0)\|^2, \tag{10}$$

where J is the Jacobian of $r(X)$:

$$J = \frac{\partial r}{\partial X} = \begin{bmatrix} G \\ \vdots \\ 0 \cdots, \frac{1}{\sigma_\omega} \frac{\partial}{\partial x} \pi^i, \cdots 0 \end{bmatrix},$$

where non-zero terms below G align with the element of X involved in each error term.

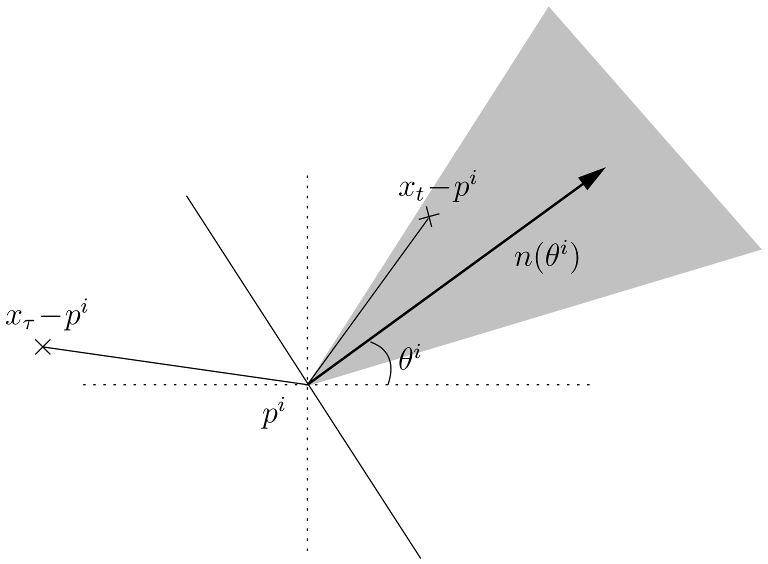

Fig. 2. The arrow is the camera optical axis $n(\theta)$. The gray region is the field of view of the camera. The dot product of $n(\theta)$ and a target location x is positive if the target is in front of the camera.

Substituting (10) into (8), we get a constrained least-squares problem:

$$\begin{aligned} X_1 &= \arg\min_X \|r(X_0) + J(X - X_0)\|^2, \\ \text{s.t.}\ & AX > b. \end{aligned}$$

This is a quadratic program in X. Notice that J is very sparse, with exactly 2 non-zero elements in each row. This expedites finding the optimum with QP solvers such as LOQO [9].

Iteratively linearizing r and solving this QP is similar to optimizing ϵ using Newton-Raphson with inequality constraints.

6 Synthetic Results

We simulated our approach with synthetic trajectories and sensor measurements. Figure 3 depicts the synthetic setup. Sensors are placed around a square environment, and the target's motion is generated randomly. Whenever the target hits the wall, it is reflected back. This trajectory is smoothed and passed to synthetic cameras which generate measurements. The state-space model of (1) cannot capture these operations, but we show here that state-space dynamics are sufficient to generate paths that capture the qualitative motion of the target.

The optimization must begin with an initial guess that satisfies the constraints of equation (9). This is because cutting plane QP solvers such as LOQO

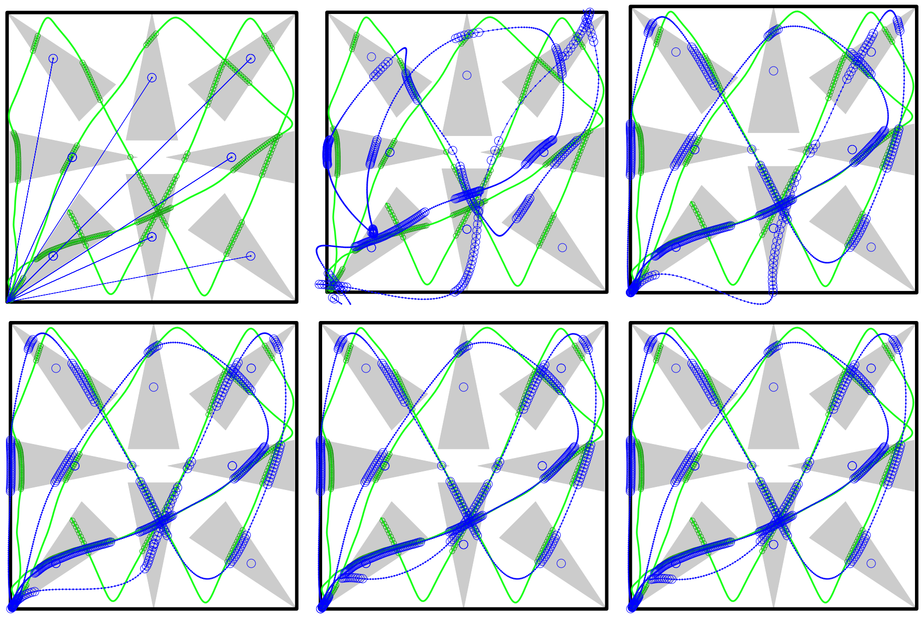

Fig. 3. Optimization begins with all points seen by a camera placed at the same location. Shown are iterations 0, 1, 3,9,11, and 17.

require an initial point within the convex constraint set. We set the initial iterate to have all unobserved trajectory points at the origin, and all observed trajectory points at one meter along the optical axis of the camera that observed it.

Figure 3 shows the estimated trajectory as it is refined. In early iterations, the likelihood term lines up the trajectory points along the ray from the camera optical center to the true target location. But initially, their distances from the optical centers are mis-estimated. The prior pulls the trajectory towards the right distance in subsequent iterations. Despite the mismatch between the dynamic models used in synthesis and estimation, the estimated trajectory is close to the true trajectory. Figure 4 shows the final answer of several more synthesized problems.

The field of view constraints are critical in recovering the trajectories. Without them, the dynamics pull the trajectory infeasible solutions. Figure 5 shows a sample trajectory estimated without the constraints.

7 Results: Real Data

We have implemented this system on a sensor network of wireless cameras with person trackers on board. Each node nodes in our network is a personal digital assistant computer equipped with low resolution camera. These devices have wireless network adaptors that allow them to communicate with a MATLAB

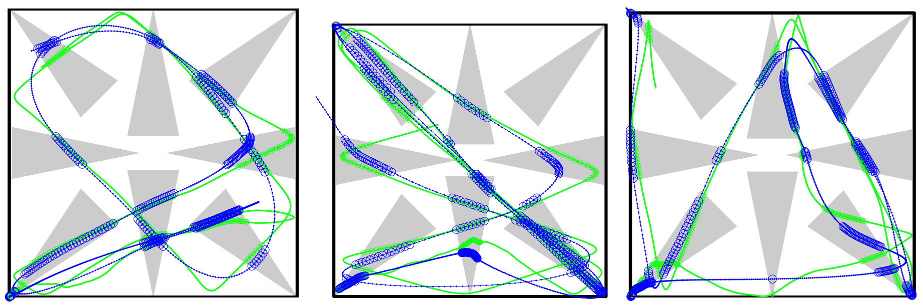

Fig. 4. More synthetic results. Notice that the ends of the trajectories do not match up with the ground truth. For these points, velocity information can only come from past points. Points in the middle of the trajectories, on the other hand, benefit from information from the past as well as the future.

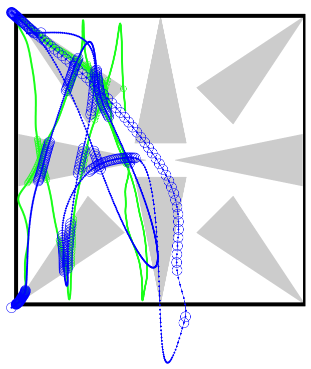

Fig. 5. Without the field of view constraints, the dynamics can pull the trajectories behind the cameras.

process running on a base station. The real-time person tracker runs on each PDA and reports the time-stamped horizontal location of a person to the base station every 250 millisecond.

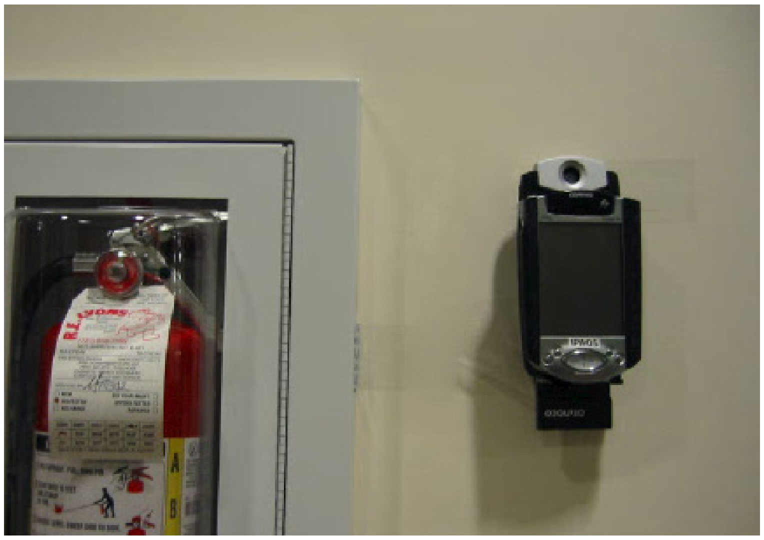

Fig. 6. IPAQ handheld computers equipped with a camera and a wireless adaptor serve as our sensor nodes. They are mounted on the walls in our office building.

We use background subtraction to locate a target within each image plane. Foreground pixels are clustered according to their image coordinates using EM on a mixture of Gaussians. Each cluster corresponds to one person in the field of view of the camera. The clusters have a prior covariance that matches the aspect of a human. This coalesces nearby small blobs into human-sized blobs and filters out isolated small blobs [7] . For each cluster, an appearance feature vector is computed and used to identify the person it represents. The identity of the person along with the horizontal component of the center of cluster is transmitted to a central processing station.

Before experimenting with non-overlapping cameras, we built a model of human motion using a pair of overlapping cameras. Stereopsis between the two cameras allowed us to recover real-world trajectories without knowledge of dynamics. A system identification procedure was applied to these trajectories to recover parameters A and Σ_ν of the dynamic model. We estimated Σ_ω by averaging the measurement error at various known locations in the room, at various target speeds.

A network of 4 PDAs observed a section of our floor. The cameras were mounted perpendicular to walls, so their orientations θ^i was easy to determine. We used the floor plan of our building to determine the location p^i of each camera. None of the fields of views overlapped. One test subject walked in the

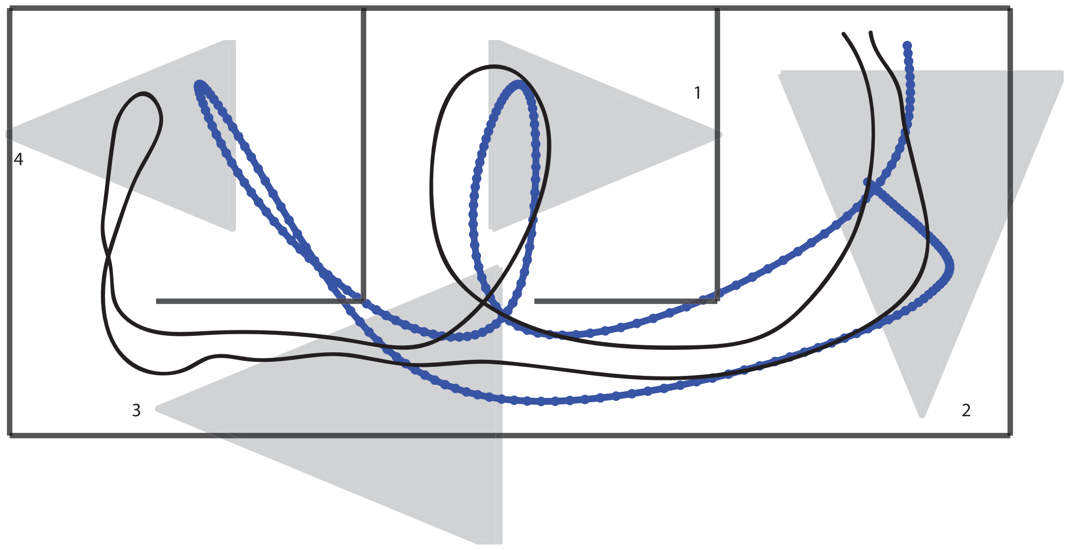

Fig. 7. Estimated real path.

environment for about one minute, beginning and ending at the same place. Figure 7 sketches the actual trajectory and plots the recovered trajectory. The small dots on the trajectory correspond to each discrete time step in the system (1/4th of a second apart).

Notice that the loop seen by camera 1 was successfully recovered. The forward and backward legs in this loops are correctly found to be at different depths. Without dynamics, it would have been impossible to determine that one path is at a different distance from the other. The long legs towards and away from camera 3 are not correctly recovered. It was impossible for the system to determine the motion of the target in this region because the subject moved along the same bearing on those legs. Notice also that the estimated trajectory goes through a wall between camera 3 and 4. Had we encoded these walls in the form of additional constraints, the legs through camera 3 might have been correctly estimated.

8 Conclusion

We have shown that some side information about the dynamics of a moving object can compensate for the lack of simultaneous correspondence between two cameras. Our method finds the trajectory that is most compatible with both the observations from the cameras and the expected dynamics. By using a convex visibility constraint, finding this trajectory can be expressed as a series of quadratic programs.

The method presented in this paper also obviates the need for a visible ground plane. This paper has focused on horizontally mounted cameras, but we plan to allow input from obliquely mounted cameras in the future.

As section 7 showed, known obstacles in the environment can provide helpful information in constraining the solution. We are working towards adding such constraints. Finally, the method we propose is batch. Batch processing is useful for this problem because the uncertaintly in the trajectory between observations can be very large. One could run this procedure in small time windows to obtain a time-lagged version.

References

1. Q. Cai and J.K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1241–1247, November 1999.
2. Y. Caspi and M. Irani. Alignment of Non-Overlapping sequences. In *ICCV*, pages 76–83, July 2001.
3. T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb. Plan-View trajectory estimation with dense stereo background models. In *ICCV*, pages 628–635, 2001.
4. M. Harville. Stereo person tracking with adaptive plan-view statistical templates. In *ECCV Workshop on Statistical Methods in Video Processing*, June 2002.
5. A. Mittal and L. S. Davis. M2Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *ECCV (1)*, pages 18–36, 2002.
6. H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *IJCAI*, pages 1160–1171, 1999.
7. A. Rahimi. Variational bayes for tracking moving mixtures of gaussians, 2000.
8. R. Zabih V. Kettnaker. Counting people from multiple cameras. In *ICMCS*, volume 2, pages 267–271, 1999.
9. R. J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11:451–484, 1999.