CS231a Final Project Report: Automatic Camera Network Topology Recognition

Christian Elder, Tony Vivoli, Judson Wilson
Mentor: Alexandre Alahi

***Abstract*—Setting up and manually calibrating networks of surveillance cameras is a costly and time-consuming task. We present an algorithm to automatically determine the external parameters of a network of cameras from inference on pedestrians moving naturally through the network's collective field of view. Our method uses time-windowing and a voting scheme to determine most likely tracklet matches between cameras to find pairwise geometry between cameras. MDS–MAP, an SDP relaxation, and the Levenberg–Marquardt algorithm are then used to solve the relative positions and angles of the camera set. We performed tests using two different simulated data sources, a Linear-Dynamical System based tracklet generator, and output generated using the RVO2, as well as real world data from Gare de Lausanne, a railway station in Lausanne, Switzerland.**

I. Introduction

A. The Problem

As the price of digital cameras has dropped, systems of networked cameras have become more prevalent. Furthermore, the cost of computation has dropped low enough that video processing is a viable option for many applications of camera networks. Such applications include analysis of consumer patterns in marketplaces, or use in crime prevention.

While the hardware is now mature and available on the market, there are still many problems to solve and areas for improvement. Manual calibration of such systems still remains an exhausting and expensive task. Cameras that are pre-calibrated and RGB-D (color and depth) equipped will soon be the norm, but this only solves half of the problem. To make real use of a network of cameras, the position and orientation of each cameras in the world must be known, at least relative to a common frame of reference. Traditionally this requires many hours of manual work, and does not scale well as more cameras are added.

In theory, this calibration information can be inferred from the spatial and temporal information of the scene. Existing techniques can be applied to identify moving objects within the scene from RGB-D data. In the course of this project, we endeavour to create an algorithm which will use the estimated trajectories, or tracklets of the objects in the scene, combined with statistical inference techniques and geometry, to estimate the relative locations and orientations of the cameras from which the data originated.

B. Algorithm Hypothesis and Overview

The underlying idea behind the algorithm is that, given the image of a line in both cameras in a planar world, and a known distance between points imaged on the line in each camera, we can fully constrain the geometry between the two cameras within the planar world.

While there are many ways that such lines and points could be identified, we chose to use the paths of moving objects. If an object is moving at constant speed and direction, the path will be a perfect line. If the object produces a linear tracklet in one camera, then a speed and direction estimate can be determined. Afterwards, any observation by any other camera at a later time can be extrapolated from the path in the first camera to calculate a possible correspondence. This is enough information to determine the relative position and angles of the two cameras in the 2D floorplan, assuming planar motion. This process is illustrated in Figure 1, steps 1 and 2.

Given a sufficient number of correspondences between many tracklets in many cameras, their external calibration can be determined. In our formulation of the problem, we do not know which tracklet corresponds to which object, so we use time windowing and group behaviors to help match tracklets between cameras. Still, the process will produce many outliers from false correspondences. We employ a voting scheme on the geometric parameters between the orientations of camera pairs, as seen in step 3 of Figure 1. If certain voting thresholds are met, we create a geometric constraint between a pair of cameras.

Once we have determined a (usually overconstrained) set of geometric relationships between all cameras, we attempt to find the optimal relative positions and angles of the entire set of cameras on the ground plane, as shown in steps 4 and 5 of Figure 1.

II. Related Work & Key Contribution

A. Prior Art

The problem of determining relationship between cameras with non-overlapping views is not new, and there are several approaches to the problem. Other work has been done to infer the network topology of the camera system [3], [5]. These algorithms typically estimate connectivity between cameras, in the sense of estimating transition probabilities of objects traversing a network, where these objects are observed at the camera nodes. These methods give little insight into the geometric relationship between the cameras. However, these network topologies may be useful in determining corresponding objects between cameras.

In [7] Rahimi et. al. give an algorithm to determine the relative position and rotation of cameras within a network, however it requires that a single object move through the network during a manual calibration phase. Similar calibration steps are needed in [4], and the algorithm is not well detailed.

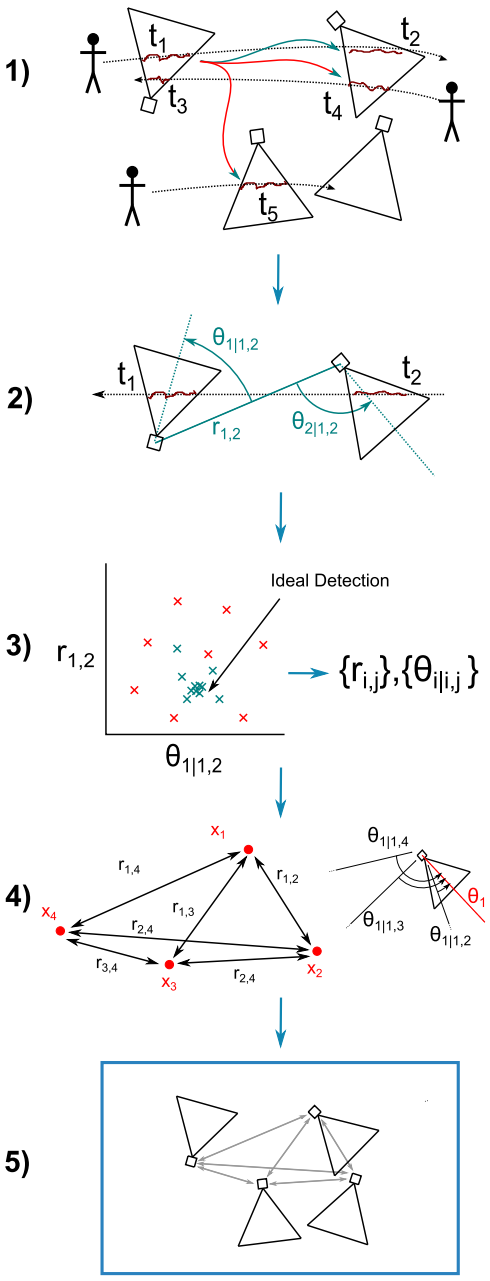

Fig. 1. Overview of the algorithm.

B. Key Contribution

Unlike other contemporary methods which require prior scene information or manual calibration, the goal of this algorithm is to automatically estimate the external parameters of a network of cameras from objects naturally passing through the network's collective field of view. In addition, we will be incorporating the technological advances offered by RGB-D cameras.

We propose a series of algorithms that can generate the full external calibration of a network of cameras using only tracklet data which does not maintain labellings across cameras. We rely heavily on established fundamental algorithms such as time windowing and voting, and use localization algorithms from wireless sensor localization. However, to our knowledge we are the first to combine the techniques in the way proposed, using time windowing and voting to estimate pairwise camera calibration, and then using the pairwise constraints to generate the calibration for the entire network.

The combination of time windowing and voting methods employed by our algorithm are particularly well suited for the unsupervised, automated case where tracklet correspondences are not known.

III. Algorithm

A. Determining Correspondences

The core datapoint for our algorithm is the *tracklet correspondence*, which we will often refer to as simply a *correspondence*. These are chosen pairs of tracklets that are assumed to correspond to the path of a single object on a trajectory through the field of view of two cameras. This step is depicted in the overview Figure 1, step 1.

We begin by chronologically ordering all tracklets across all camera fields of view. A sliding time window of width Δt is then applied to the chronologically-ordered data. Initial correspondences are generated at each timestep by pairing each tracklet within the sliding window from each camera with all tracklets from all other cameras within the time window, ignoring repeated pairs. This method attempts to find tracks intersecting two nearby camera FOVs in a short time window, minimizing error caused by changes in trajectory as time accumulates.

In real world applications we would sort longer time windows by the number of observations, and operate on the set with the fewest observations, to minimize the number of objects in the scene (and thus minimize false correspondences). In our testing with limited data, we were required to manually make a trade-off to ensure enough correspondences to be useful.

Once the initial list of correspondences is constructed, we use group-behavioral cues to further match corresponding tracklets. The list is then filtered by what we refer to as a *group shape*. A group refers to a group of people that move together through the camera frames. The motivation for this is that groups of people will stay together the entire time they move through the camera network, and characterising groups can lead to better correspondence matching. (Note, this method is useless for data generated by simulations that do not model this behavior, so we only use it on real-life data.) To characterise a group shape descriptor, the distances to the four closest people to a person are used to calculate a feature vector.

The feature vector is a four element vector, where the the element i is a weight value that is calculated from the radial distance of the i^{th} closest person. The weight function used is:

$$w(r) = \begin{cases} 1 & \text{if } r < 1 \\ e^{-\frac{(r-1)^2}{2}} & \text{if } r \geq 1 \end{cases}$$

A continuous function is desirable so that if small changes in distance of one person would not cause a large change in the feature vector which allows the shape of a group to change slightly and still have a high probability of matching.

To calculate a shape descriptor for a tracklet, a shape descriptor is calculated at each point along the tracklet. A KD-Tree is used to find the nearest 4 neighbors that will be used to determine the feature vector. The shape descriptor for the entire tracklet is the average of the shape descriptors of each point along the tracklet.

Once each tracklet has a shape descriptor, the previously generated list of correspondences is filtered using these descriptors. We compare the group shape of the two tracklets in each correspondence pair by taking the euclidean distance between the two group shapes. If this distance is greater than a minimum threshold value, the correspondence pair is discarded. In addition, all correspondence pairs of people that are determined to not be part of a group are filtered out to eliminate matching individual people to many other non-corresponding individual people. To check if a person is not part of a group, the weight for the first element of the feature vector should be sufficiently low signifying that there was usually no one within four meters of this person.

B. Calculating Pairwise Camera Geometry Relationships Implied by a Tracklet Correspondence

Real world objects, such as people and automobiles will never follow a perfectly straight path at a constant speed, but by underlying assumptions of inertia and an intended direction, we model their behavior as linear with smooth velocity with additive Gaussian noise. For each tracklet in a correspondence we produce a best-fit-line estimate of the tracklet path using PCA to derive an orthogonal least squares estimate. The process consists of centering the data about its centroid and using SVD to determine directions of greatest and least variance. The direction of greatest variance is chosen as the direction of the line, and the perpendicular direction is considered a dimension of noise.

Average speed is calculated as the difference in position of the first and last points in the tracklet, projected onto the best fit line, divided by the difference in time that the points were measured. The time at the centroid is estimated to be average of the first and last times of the tracklet observations.

From the lines, average speeds, and centroid-times calculated for each of the two corresponding tracklets in cameras i and j, we can extrapolate a correspondence, from which we can generate an estimate of the relative camera positions in the form $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$. Here, $r_{i,j}$ is the distance between cameras i and j, i.e. the length of a chord between them, and $\theta_{i|i,j}$ and $\theta_{j|j,i}$ are the angles to each camera i and j, respectively, from the chord drawn between them. They are depicted in the overview Figure 1, step 2. We refer to these parameters as a ***camera-relation*** estimate.

The calculation of $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ is trivial knowing the centroid, lines, and speeds of of the two tracklets. The constraints are that the two tracklet lines must be co-linear (with the velocity vectors pointing in the same direction), and that the distance between the centroids must match their estimated difference in time multiplied by the average speed of the two tracklets.

One step that we found helps later estimates is to center the camera origin in it's observed tracklets. In other words, transform the camera coordinate systems such that the centroids of all the pointwise observations seen by each camera lie at the camera origin. After the algorithm is complete, this adjustment will need to reversed to find the true camera position. This decreases the error dependency between the angle and distance measurements. If the camera origin is located relatively far away from the tracklet observations, a small error in the tracklet path angle may lead to a rather large error in the estimate distance between camera origins, which is undesirable.

C. Camera Relation Estimation by means of Voting and Median Filtering

From many of these noisy tracklet observations we generate many noisy $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ camera relation estimates. For the simple case described, we would fit a Gaussian distribution to the camera position coordinates and determine the most likely coordinate over all observations for the camera pair. (This would be individually applied to every camera pair.) However, this method has a couple of problems. The first problem is that the angular estimates are periodic, so a Gaussian distribution does not apply. The second problem is that false correspondences, caused by assuming two unrelated tracklets were generated by the path of a single object, lead to additional observations of an unknown, second distribution. This is illustrated in the overview Figure 1, step 3. The green points represent measurements of true correspondences, and the red points represent false correspondences. Note the illustration is of a 2d parameter space, but there is actually 3 parameters: $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$.

We are targeting a real world case where we cannot ensure that we know an object observed in one camera is the same as one observed in another. The intention is that there will be only one object in the environment being observed, and it will take a somewhat straight path through several cameras.

It is clear that we desire a low number of false correspondences. The previously mentioned time windowing, and group shape filtering techniques will help mitigate the problem, but it will still exist.

If the true corresponding tracklets are clean enough and linear enough, the $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ camera-relation estimations should occupy a rather tight cluster in the parameter space, while the false correspondences should be distributed more randomly.

Based on this idea, we use a voting scheme to help identify the true correspondences, similar in principle to a Generalized Hough Transform. Each $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ camera-relation estimate from each tracklet pairing of the two cameras constitutes a vote in a discretized 3D parameter space. The parameter space is divided into a $9 \times 9 \times 9$ set of bins, and the best vote is determined by a $3 \times 3 \times 3$ sliding-window (stretching over 1/3rd of the space in each dimension).

There are a couple points to note about this. The angular bins are periodic, so care must be taken with the sliding window to allow wrapping. Also, distant outliers could stretch the range of the voting space in $r_{i,j}$, so we size the bins such that 8 out of 9 equally sized bins encapsulate the range of 0

to the 90^{th} percentile largest $r_{i,j}$, and allow the 9^{th} bin to account for extra margin.

This peak window does not exhibit periodicity in the dimensions for the angles (since it is bounded), so we can now apply a center estimate. We use the median filter in each dimension of $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ over the votes in the peak window.

We only accept this estimate if there are at least 5 votes total. The acceptance threshold is 50% of votes when there are less than 10 total votes, and 10%-20% of votes otherwise. This ensures a certain qualitative level of confidence in the vote, even for low sample sizes.

These thresholds build in a certain amount of acceptable false correspondence tolerance. Intuitively, if the correct cluster of votes is tight, it will only be accepted if there is a high enough ratio of true correspondences to false correspondences, assuming the false correspondences are widely distributed. This is an important point to understand when analyzing vote data and system performance.

To see examples of the voting scheme results on simulated data, see Figures 3 and 4 in Section IV: *Experiments and Results*.

D. Camera Localization and Pose Estimation

The estimation of the camera-relation geometry between pairs of cameras is carried out for every pairing of the cameras, and results in a possibly overconstrained set of distances and angles. These constraints can be viewed as an undirected graph, with each node as a camera and each edge representing the relative geometry between them. The graph may be fully or partially connected. We will only discuss cases when the graph is connected, i.e. there is a path from any node to any other node, depending on whether the voting procedure rejects or accepts the best voting window.

From these pairwise geometric relationships we can estimate the global location and pose of all the cameras as a set, up to a rotation and a translation. This is depicted in the overview in Figure 1, step 4 and 5. The problem of finding both the optimal location and angles, in a least squares sense, is both non-convex and nonlinear, but there are methods to produce approximate solutions. We use three different methods, all with origins in the large body of work regarding sensor localization: MDS-MAP, Biswas' and Ye's SDP relaxation method, and Levenberg–Marquardt nonlinear least squares optimization.

1) MDS-MAP Localization: MDS-MAP is a based on the (MDS) Multidimensional Scaling technique from statistics, with adaptation to the wireless sensor problem. Given a set of n sensors, the traditional method would require distance measurements r_{ij} from all n nodes to their $n-1$ neighbors. Using SVD based techniques, the algorithm returns the best locations estimates $\{x_i\}$ to minimize the cost c:

$$c = \sum_{i,j} \left(||x_i - x_j|| - r_{ij} \right)^2$$

This requires having full connectivity, but a simple method is used to fill the missing values. Any missing distance is estimated by measuring the shortest path distance of known distance estimates. (Therefore there must be a known path from every node to every other node on the graph.) We perform this step using a dynamic programming approach.

Note, in general, these estimates for the missing paths will be longer than the actual path. Some literature discusses weighting these distances to decrease path length, but we followed the method of not scaling the distances.

It's also worth noting that these shortest-path distances, while non ideal, may help yield better results than other cost functions that simply omit the missing distance constraints. Those other methods may allow the structure to "fold inwards" when the graph is weakly connected, to reduce cost. However, this solution is likely not correct, because it implies that there are other nearby neighbors in the folded solution that should generate pairwise correspondences, but we know that they did not. Therefore, the distortion of the shortest-path distances may often produce a better result.

We refer you to the literature for further details about the MDS-MAP algorithm. Shang et al. introduce the method in [8]. Performance analysis and a simple formulation of the algorithm are given by Oh et al. in [6]. The algorithm is widely used and there are many other sources of information.

The MDS-MAP algorithm produces an optimal positioning up to a translation, rotation, and reflection (since it is based on distances only). We can now easily estimate the camera angles. Using the solved locations, for each relevant $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ constraint we can produce a vote for the absolute angle of the cameras i and j. We simply use the average of these absolute angle votes as the estimated rotation.

To resolve the issue of the positional reflection ambiguity (which does not exist when the camera angle relations are considered), we generate a second estimate by reflecting the position estimates about one of the axes, and recomputing the absolute angles. We then choose the set of position and angles that produces the best angular cost, defined as follows: Assume the n cameras are located at $\{(x_1, y_1),\ (x_2, y_2),\ \ \dots\ \ , (x_n, y_n)\}$, and we have a set of m relation constraints of the form $\theta_{i|i,j}$ which are the angle of camera i as measured from the chord between camera i and camera j. The cost of a set of absolute camera angles $\{\alpha_1, \alpha_2, ..., \alpha_n\}$, parametrized by the locations $\{(x_1, y_1),\ (x_2, y_2)\ , \dots, (x_n, y_n)\}$ and the angle constraints $\{\theta_{i|i,j}\}$ can be formulated as a sum of squared differences:

$$C_{\{(x_i,y_i)\},\{\theta_{i|i,j}\}}(\ \{\alpha_i\}\ ) = \sum_i^n \sum_{\{\theta_{i|i,j}\}} \left(\alpha_i - (\theta_{i|i,j} + \operatorname{atan2}(y_i - y_j,\ x_i - x_j)) \right)^2$$

Note that because the angles are periodic, we must consider the difference terms in their periods that minimize this error. Once they are aligned to the correct period, our angle solution method by averaging, discussed above, is the minimizer of this cost for a given $\{(x_1, y_1),\ (x_2, y_2)\ , \dots, (x_n, y_n)\}$ and the corresponding $\{\alpha_1, \alpha_2, ..., \alpha_n\}$.

This procedure produces an estimate of the location and rotation of all of the individual cameras, up to a global translation and rotation.

2) SDP Relaxation Method: In general, choosing the positions of the cameras that minimize the sum-of-squared-error of the position distances and the estimated constraints is a non-convex problem. Biswas and Ye presented a relaxation of the problem in the form of an SDP. We refer you to their paper [1] for the formulation, although the formulation in [9] is simpler.

The method has potential benefits MDS–MAP. It does not require complete pairwise distances between every node. It is also solvable using the readily available convex optimization software cvx by CVX Research, unlike some other approaches.

One drawback of the method is that it requires 3 anchor locations of known location (up to a rotation and translation) to function. Because we do not have anchors, we first started with the MDS-MAP solution. We then randomly choose 3 nodes to act as anchors, solve the SDP, and see if the solution improves cost. In this case we use the following positional cost function for positions of n cameras $\{(x_1, y_1),\ (x_2, y_2)\ , \ldots, (x_n, y_n)\}$ given distance constraints between cameras of the form $\{r_{i,j}\}$:

$$C_{\{r_{i,j}\}}(\{(x_1, y_1),\ (x_2, y_2)\ , \ldots, (x_n, y_n)\}) = \sum_{\{r_{i,j}\}} \left(r_{i,j} - ||(x_i - x_j)^2 + (y_i - y_j)|| \right)^2$$

If the cost has decreased, the new answer is accepted as a better candidate solution. Then the processes is repeated: choosing 3 random points as anchors, solving the SDP, and comparing the cost. When no improvement in cost is made after a fixed number of attempts, the solution stops.

The camera angles are calculated using the same averaging method as was done when using MDS-MAP. Like MDS-MAP, because this method initially localizes using only distance constraints, there is a reflection ambiguity in the position solving step. The camera angles are computed for both the original position solution, and its reflection about an axis. The solution that produces the lowest angular cost is chosen as the best fit.

3) Levenberg–Marquardt Nonlinear Least Squares Optimization: The Levenberg–Marquardt algorithm is a method for solving non-linear least squares problems. Given a set of "residual" functions $\{f_i(x)\}$, where x is a vector, and $f_i(x)$ is not necessarily a linear function, the algorithm attempts to find a solution to the following problem:

$$\arg\min_x \sum_i (\ f_i(x)\ )^2$$

The behavior of the algorithm is adaptive, and as suggested by Ellis and Hazas in [2], this method, which they call NLR (for Non-Linear Regression) has some distinct advantages over MDS-MAP in sensor localization, especially in the case when angular relationships are known. They claim that angular costs also help maintain the layout structure of the sensors.

We use the same cost residuals as Ellis and Hazas. In terms of the equations given above, and camera positions $\{(x_i, y_i)\}$ and absolute angles $\{\alpha_i\}$ we have a distance cost term for each pairwise distance constraint r_{ij}:

$$f_{d,ij}(\ \{(x_1, y_1), \ldots, (x_n, y_n)\},\ \{\alpha_1, \ldots, \alpha_n\}\ ) = ||(x_i - x_j)^2 + (y_i - y_j)^2|| - r_{ij}$$

and angular cost terms parametrized by each camera's angular estimates $\theta_{i|i,j}$ that constrains the angle between camera i and the chord drawn from (x_i, y_i) to (x_j, y_j):

$$f_{a,i|ij}(\ \{(x_1, y_1), \ldots, (x_n, y_n)\},\ \{\alpha_1, \ldots, \alpha_n\}\ ) = \alpha_i - (\theta_{i|i,j} + \operatorname{atan2}(y_i - y_j,\ x_i - x_j))$$

The implementation of this cost function must choose the period of each angle that minimizes the cost.

Conceptually, we can see from the underlying idea of these geometric constraints that this algorithm has some powerful advantages over other methods. Placing cameras using only distance information requires a certain level of triangulation to be fully constrained. However, with the non-linear least squares method, any connected network of cameras will be fully constrained by a distance and an angle constraint. This is very advantageous for various camera network topologies or situational conditions where the cameras are connected by relatively few constraints, such as a network of cameras in a mine-shaft. It also helps when poor voting results in a small set of constraints.

After applying one of these localization and orientation algorithms, we have solved the problem up to a global rotation and a translation.

IV. Experiments and Results

For our experiment, we used three different sources of data. First we used a simple Linear-Dynamical System model to generate non-linear tracks. We then used the RVO2 library to generate more human-like tracks. Finally, we tested our methods on a set of real-world camera data collected at the Lausanne railway station in Lausanne, Vaud, Switzerland.

A. Simple LDS Simulated Dataset

We produced a simple algorithm for generating track and tracklet data using a linear dynamical system (LDS) model of a moving object. First, a set of cameras was created and placed in the world. Then tracks were generated. Track coordinates (x_t, y_t) were computed for each timestep according to the following LDS:

$$\mathbf{x_{t+1}} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ x'_{t+1} \\ y'_{t+1} \\ x''_{t+1} \\ y''_{t+1} \\ x'''_{t+1} \\ y'''_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ x'_t \\ y'_t \\ x''_t \\ y''_t \\ x'''_t \\ y'''_t \end{bmatrix}$$

The initial $\mathbf{x_t}$ was chosen such that (x_t, y_t) were at random locations on the bounding box of the cameras in the world.

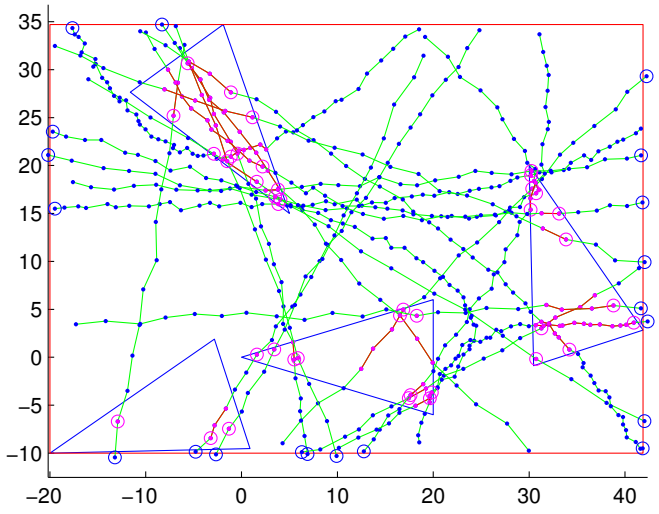

Fig. 2. 20 tracks and their observed tracklets generated by the LDS, in our experimental camera placement. Tracklets chosen randomly using our "Standard Settings."

Velocity (x'_t, y'_t) was chosen to point in a random direction inward from the bounding box, with speed drawn from a Gaussian distribution, and then clipping values outside a given range to the nearest boundary of the range. The higher order difference terms (x''_t, y''_t) and (x'''_t, y'''_t) were also formed by drawing magnitudes from Gaussian distributions and clipped, and using the direction of the initial velocity.

To form the observations, each (x_t, y_t) was calculated and independent Gaussian observation noise was added. Visually, the tracks appear to be similar to either noisy lines or noisy parabolas. See Figure 2 for sample tracklets generated using our standard settings, in our standard camera configuration.

Tracklets are then formed by finding the subset of the tracks in each camera field of view, and transforming them to the cameras' coordinate systems. Note that in our test cases the cameras were not overlapping. In cases where they are, it would be better to use different additive observation noise for each camera.

Once the tracklets are generated, true and false correspondences are created by randomly choosing how many tracks are in a time window. Generally, a probability mass function is used to choose between 1 to 8 tracks in any given time window, although for most tests we randomly choose between 1 and 3 tracks in a given time window.

We completed a series of tests, presented in Table I. The goal was to demonstrate behavior for different kinds of input to qualitatively see if the expected behaviors resulted. For a variety of input conditions using the camera configuration shown in Figure 2, we tried the algorithm with all 3 different localization/orientation algorithms, and provided the cost and error (which will be described shortly). These are single test instances, so they have questionable statistical significance, but they were chosen with very little supervision. We specifically did not compare errors and search for specific characteristic behavior. In the future it would be better to randomly generate series of tests using the same sets of parameters, and compare the average error of all the tests generated with the same parameters. We were unable to do this at this time.

The position cost is as defined in Section III-D2 where it used to compare potential solutions in the SDP optimization, and the angular cost is described in Section III-D1 when discussing how to solve for camera angles after computing the MDS-MAP camera position solution. In general these costs can be compared between algorithms when using the same test input, because the correspondence voting results are the same for each version of the algorithm.

Positional error is defined as the minimum sum of squared differences between the estimated camera locations and the ground truth locations, when applying any arbitrary global rotation and translation to the estimated positions (because we only know the solution up to a rotation and translation). The optimal rotation / translation are solved using the Procrustes transform (MATLAB function `procrustes`).

Angular error is defined in a similar way to the positional error, with some special considerations. It is defined as the minimum sum of squared differences between the estimated camera angles and the ground truth camera angles, when applying an arbitrary global rotation to the estimated cameras. We approximate the optimal, error minimizing rotation as the angle-mean of the individual differences of the estimated camera angles and the ground truth camera angles. The angle-mean is defined as the angle of the sum of unit vectors pointed in the direction of each angle being averaged over. This technique is used because the angles are periodic, so calculating a "mean" is ambiguous. Adding this angle to all of the camera angles, the differences between the ground truth and offset estimated camera angles are calculated, squared, and summed. Note the differences are calculated using the angle periods that minimize the differences - an important implementation note.

We also present "Connectivity" in Table I, which is a count of the number of pairwise geometric constraint tuples $(\theta_{i|i,j}\ ,\ r_{i,j}\ ,\ \theta_{j|j,i})$ from our voting scheme, vs. the maximum number possible.

These tests were all performed with the same ground truth camera setup, consisting of 4 cameras in close proximity, as shown in Figure 2.

The first two tests in the table were constructed such that the tracks were completely straight with constant velocity, no observation noise, and no false correspondence tracklet matches. As expected, if there enough tracklets, the pairwise geometric constraint voting picks the exact distance and angle constraints (because all votes are identical), and MDS-MAP chooses the perfect solution (to within working precision). The SDP and LM NLR algorithms do as well, which is expected, since they use the MDS-MAP solution as a starting estimate. The second test has poor connectivity, due to too few tracks, and we can see that MDS-MAP and the SDP relaxation have very bad error, even though the positional cost is very low. This suggests that the distance constraints do not properly triangulate at least one camera, so positional cost of the localization solution is small while the actual error is

		1. Noiseless, Const. Vel., 0FC, High Conn.	2. Noiseless, Const. Vel., 0FC, Low Conn.	3. Constant Velocity, Noisy, 0FC	5. Low Noise/Curvature, 0FC	5. Standard, 0FC	6. Standard - 100 Tracks	7. Standard - 400 Tracks	8. Standard - 1600 Tracks
	Test Tracks	100	50	400	400	400	100	400	1600
	Connectivity	6/6	4/6	6/6	6/6	6/6	5/6	5/6	6/6
MDS MAP	Pos. Cost	0.000	0.80	0.013	0.14	0.033	136.91	13.73	1.49
	Angle Cost	0.000	6.00	0.002	0.000	0.014	1.66	0.88	0.019
	Pos. Error	0.000	3571.86	1.04	0.20	1.58	311.29	284.11	2.54
	Angle Error	0.000	6.00	0.000	0.000	0.000	0.12	0.41	0.004
SDP relaxation	Pos. Cost	0.000	0.80	0.012	0.13	0.033	3.07	0.73	1.49
	Angle Cost	0.000	6.00	0.002	0.000	0.014	0.021	0.016	0.019
	Pos. Error	0.000	3571.86	1.03	0.21	1.59	4.89	9.68	2.54
	Angle Error	0.000	6.00	0.000	0.000	0.000	0.017	0.033	0.004
LM NLR	Pos. Cost	0.000	0.000	0.008	0.081	0.020	1.02	0.46	0.87
	Angle Cost	0.000	0.000	0.001	0.000	0.014	0.023	0.010	0.011
	Pos. Error	0.000	0.000	1.00	0.10	1.57	2.70	6.90	1.13
	Angle Error	0.000	0.000	0.000	0.000	0.000	0.016	0.029	0.003

TABLE I: **Selected Results** - Performance using MDS-MAP, SDP relaxation and LM NLS over selected examples to demonstrate characteristic behavior of the algorithms. Note "0FC" stands for "Zero False Correspondences".

much larger. The angular cost is high, but these two methods solve the camera angles after position, and the poor positions cause this high angle cost. The LM algorithm produces a zero-error solution (to within working precision). This is to be expected, since the relation graph is fully connected, so fully constrained by perfect constraints, when you include the angular constraints. The algorithm uses location and angular constraints to choose both the positions and angles of the cameras, simultaneously, so is able to arrive at the correct solution.

Tests 3-5 show a few cases of non-ideal measurements, adding observation noise, curvature, and speed changes to the tracks. All 3 cases are constructed such that they never have multiple tracks in the same time window, and thus no false tracklet correspondences. We can see that, compared to all the tests overall, they perform quite well, with the LM algorithm consistently performing with a slight edge. We attribute this performance to the lack of false correspondences. We can see in Figure 3 the relatively few outliers compared to the cluster in the peak window.

Tests 6-8 use our standard track generation settings (noise and curvature as bad as or worse than tests 3-5), with 100, 400, and 1600 tracks, where the tracks of the smaller test samples are subsets of the tracks of the larger samples. Each time window has either 1 track with 30% probability, 2 tracks with 40% probability, or 3 tracks with 30% probability. When multiple tracks are in the same time window, 30% of tracks will intersect only a single camera's field of view. All other tracks have a tracklet intersecting 2 cameras' fields of view.

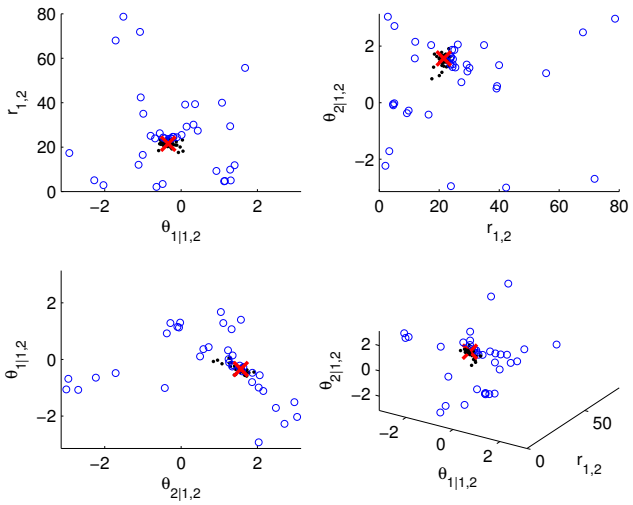

Fig. 3. Voting results for Test 5, for camera relations between cameras 1 and 2. Votes in the peak window are solid dots, and those outside are hollow. The chosen median value is the red-x. Note this is the relation between the camera observation centroids, not their origins.

We can see in Figure 4, that the false correspondences in Test 7 produce many more outliers in the camera relation votes for the pairing of camera 1 and camera 2, compared to Test 5 (shown in Figure 3) which uses the same tracks except some tracks fall in the same time window, causing false

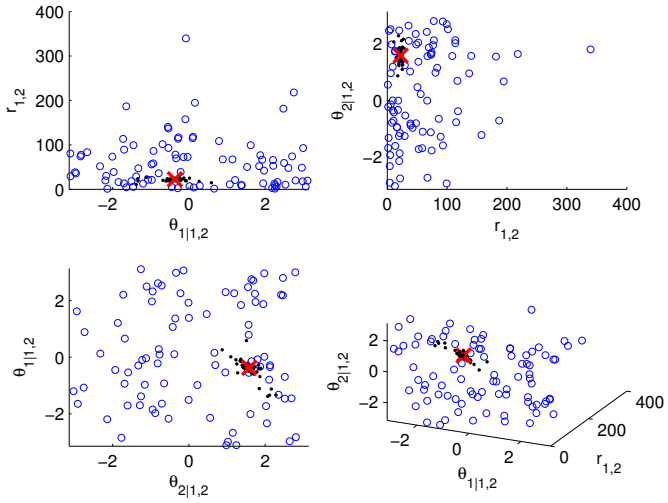

Fig. 4. Voting results for Test 7, for camera relations between cameras 1 and 2. Votes in the peak window are solid dots, and those outside are hollow. The chosen median value is the red-x. Note this is the relation between the camera observation centroids, not their origins.

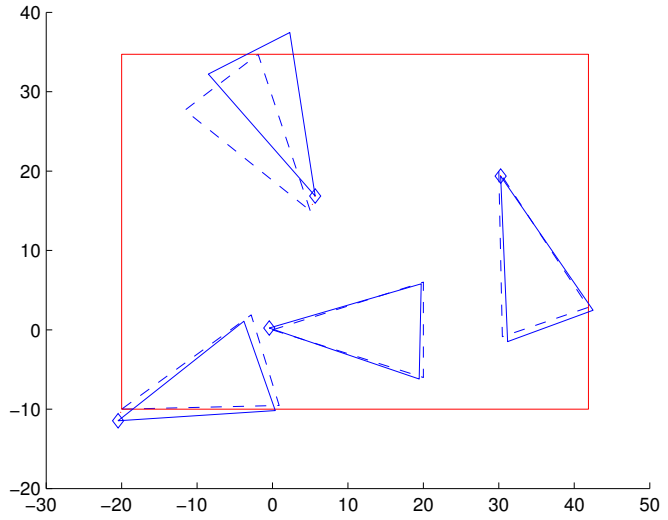

Fig. 5. Test 7 LM NLR result camera locations and orientations (solid) after a global rotational and translational alignment to the ground truth (dashed).

correspondences.

Generally we see better performance with more tracklets, an effect you may expect to see. The increase in connectivity for the 1600 tracks case is also promising, but again, these single tests aren't statistically significant for drawing hard conclusions. We can see that MDS-MAP performs the worst, followed by the SDP relaxation, with the LM algorithm performing the best.

We can see that the algorithm works, and provides sensible results and characteristic behavior, for the data as generated.

To visualize a result, see at Figure 5. This is the resulting positions and camera angles using the LM NLR algorithm, after applying the optimal translation and rotation for alignment (because we can only solve up to a translation and rotation).

B. RVO2 Simulated Dataset

In addition to the initial simulator described previously, a second, more realistic simulator was implemented. This simulator utilizes the RVO2 library developed at the University of North Carolina at Chapel Hill to model the interactions between pedestrians and obstacles moving through a scene. With this simulator, scene geometry, camera locations, and pedestrian attributes such as preferred velocity and frequency and location of entry and exit are easily controllable, enabling for a better approximation of real world data while retaining a controlled test environment. An example environment is shown in Figure 6, which is geometrically similar to the Lausanne train station. Unfortunately, the computational requirements of the RVO2 library limited testing to simulations of two hours with up to 15 pedestrians in the scene at a given time.

Testing with this more advanced simulator was composed of determining locations of groups of cameras (see Figure 6) under various traffic frequencies using the three algorithms described previously. The groups tested were cameras 1 and 2, cameras 2 and 3, cameras 1, 2, and 3, and cameras 5, 6, 7, 8, and 9, and pedestrian frequencies were 1, 5, and 15 pedestrians in the network's field of view at a given time.

Under these testing conditions, we saw the corroboration of several of the conclusions from testing with initial simulator. In the majority of testing, the LM NLS algorithm produces the least total error between estimated and actual camera locations. Furthermore, as the number of pedestrians in the scene increased, the increase in the number of tracks improved the error of all three algorithms. However, this increase has the secondary effect of introducing more false correspondences due to more pedestrian observations per time window, which seems to have negated some of this benefit. As the number of pedestrians in a single time window increases to the levels expected in a busy train station corridor, we expect that the number of false correspondences generated would completely obscure our Hough space clustering. Therefore, an augmented correspondence algorithm is required beyond 15 pedestrians per time window.

It should be noted that, in the test of the group of 5 cameras with 15 pedestrians in the scene, not enough graph connections were established to perform any of the localization algorithms. Upon closer inspection, this may have been caused by the tendency of individual pedestrians to walk across the scene at a approximately constant vertical displacement, thus appearing in only cameras 6, 8, and 9 or only in cameras 5 and 7. This led to high connectivity within these two subgroups, but very little inter-subgroup connectivity.

C. Lausanne Railway Station, Real World Dataset

Our final dataset comes from a camera network installed in Gare de Lausanne, a railway station in Lausanne, Switzerland, provided by Alexandre Alahi. Figure 7 illustrates the camera layout. The network uses RGB-D cameras to detect moving objects through the station, and a software pipeline is used

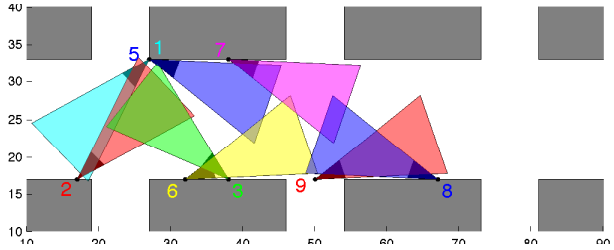

Fig. 6. Layout of simulated corridor with numbered camera fields of view overlay. The corridor consists of 8 entrances/exits, separated by grey walls.

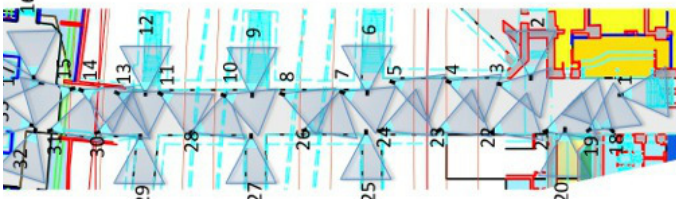

Fig. 7. Layout of corridor at Gare de Lausanne with numbered camera fields of view overlay

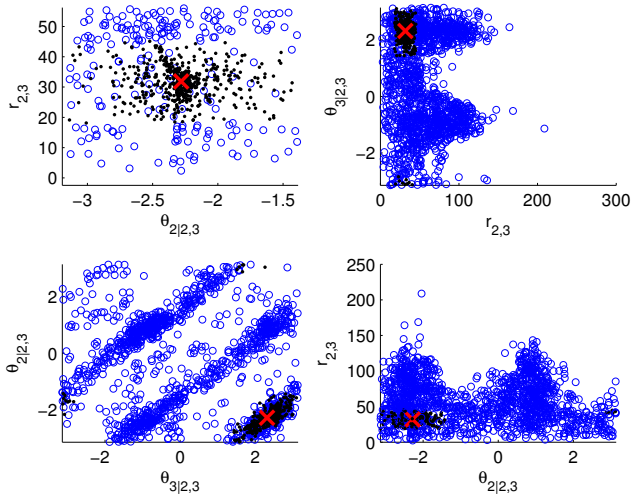

Fig. 8. Voting results for the Gare de Lausanne test data, relation between cameras 18 and 19. This is without applying the Group Filter. Votes in the peak window (25% acceptance threshold) are solid dots, and those outside are hollow. The chosen median value is the red-x. Note this is the relation between the camera observation centroids, not their origins.

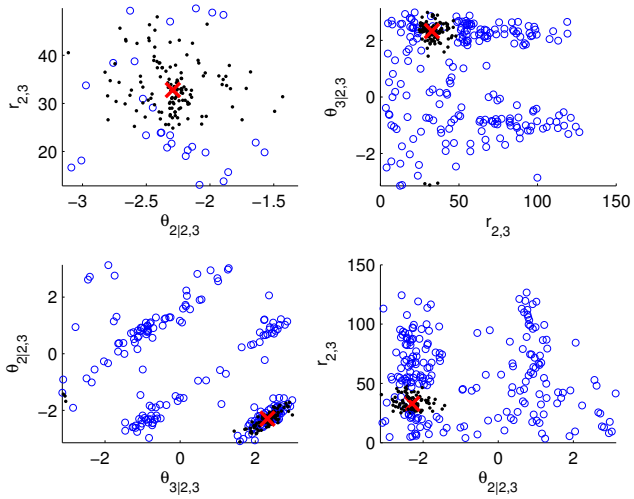

Fig. 9. Voting results for the Gare de Lausanne test data, relation between cameras 18 and 19. This is after applying the Group Filter. Votes in the peak window (25% acceptance threshold) are solid dots, and those outside are hollow. The chosen median value is the red-x. Note this is the relation between the camera observation centroids, not their origins.

to preprocess the data, creating tracklet data which describes the motion of objects through the cameras fields of view. The data is given in planar (x,y) coordinates for each timestep, in the coordinate system of the observing camera, as the object moves along a track. Object labels are not tracked between cameras.

The two data samples we were given were of 23 and 40 minutes in length, during times of moderate crowd density. The data contained fields of view for all 33 cameras, of which a representative subset were tested, due to computational constraints.

We began testing using cameras 1, 18, and 19. Because there are often several objects within the network's field of view within a designated time window, the data was filtered to remove times during which a large crowd is visible. This filtered data is then processed by the algorithm described in section III. We compare results of the relation between overlapping cameras 18 and 19 with the group filter turned on and off. As shown in Figures 8 and 9, the voting scheme found a strong cluster in both instances, using a 25% threshold for peak window vote acceptance. The result using the group filter has $\sim$82% fewer points, with a larger fraction in the peak voting window than the unfiltered results: 129 out of 336 votes vs. 546 out 1958 votes. We consider this a strong indicator that the group shape filter can produce better results, at least in the correct conditions. Camera 1 had relatively few observed tracklets, thus the ratio of false correspondences was too high to accept a voting window.

We also tested using cameras 4,5,7,23 and 25, which are within close proximity in the cooridor. At 25% peak voting window threshold, the voting method found 1/10 of the possible pairwise camera relations with the group shape filter turned off, and 4/10 with the filter turned on. Neither set contained a connected graph of camera relations. Using a 15% peak voting threshold, these fractions were 6/10 for the unfiltered case and 9/10 for the filtered case. We can see that the group shape filter tends to produce more camera relations, however visually inspecting the resulting camera layouts in Figures 10 and 11 and comparing to Figure 7, it is not clear whether the result is better.

Generally it appears that the cameras in Figures 10 are placed in a somewhat correct placement, but reflected, with very bad angular alignments. The result in Figure 11 appears much worse in terms of placement and angular alignment. Visually juding the votes between cameras 4 and 5, shown in Figure 12, it appears the false correspondences are not well distributed, with a high dependence on angle. We believe this is caused by unequal angular distribution of tracklets: most tracklets are parallel to the walls.

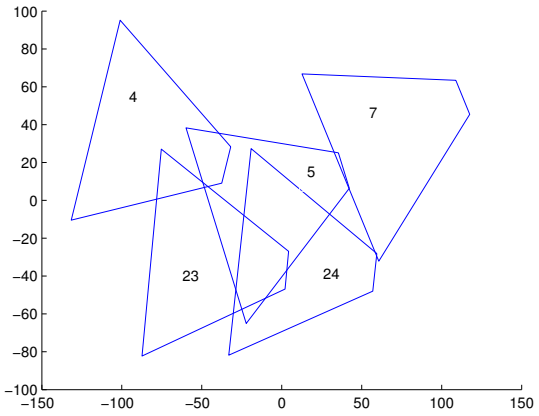

Fig. 10. Results for Gare de Lausanne data using LM NLR algorithm without group shape filtering.

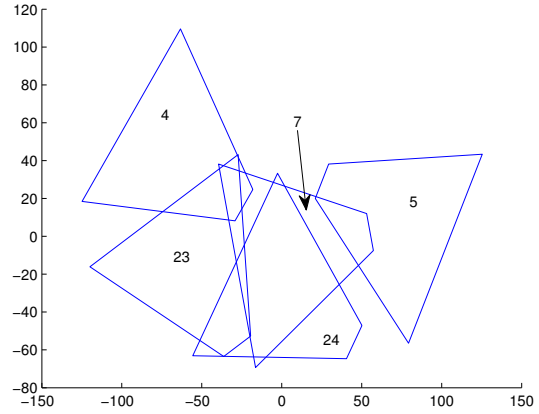

Fig. 11. Results for Gare de Lausanne data using LM NLR algorithm after group shape filtering.

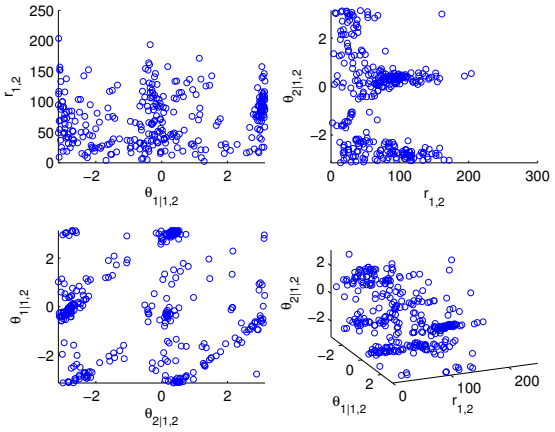

Fig. 12. Voting results for the Gare de Lausanne test data, relation between cameras 4 and 5 with a 25% peak bin voting acceptance threshold. The peak bin was not accepted, yet there is a dense cluster region.

V. Conclusions

As we have seen in the simulations, when presented with data from a few objects passing through a scene, the presented algorithm can correctly estimate the external parameters of a network of cameras. However, as seen from the real-world data, having too many people in the field of view will result in a high number of false correspondences, which leads to errors in parameter estimation. To retain the effectiveness of the algorithm during periods with visible crowds, a heuristic to remove false correspondences must be implemented. The radial grouping heuristic presented here proved to be simple and provide improvement in some situations, but it is by no means the only method of filtering. Another approach, similar to the algorithms presented in the related works, could involve first determining the network's approximate topology and applying weights to subsequent correspondences based on these graph connections.

If the ratio of true and false correspondences cannot be kept within a usable threshold for the voting scheme, a more elaborate method is needed to classify true/false correspondences.

Also, to decrease the noise of the true correspondences, we could attempt a better fit to each tracklet. Presently we are using a line to estimate the entire tracklet, but a curvilinear path based on a Kalman Filtering or other technique could better approximate the speed and direction of tracklets as an object enters and leaves the field of view of the cameras.

Overall, the algorithm shows promise, and we have demonstrated the viability of our contributions. The voting scheme works very well in our simulations, and will likely work much better in real world situations with much more tracklets taken with fewer objects in the scene. The overall algorithm can properly place cameras on the ground plane to within an acceptable amount of error in situations where tracklet matching between cameras has some low to moderate ambiguity, but will poorly-tolerate cases with too many objects in the scene in a given time window.

References

[1] P. Biswas, Y. Ye, *Semidefinite programming for ad hoc wireless sensor network localization*, in 3rd Annu. Symp. Information Processing in Sensor Networks, 2004, pp. 46-54.

[2] C. Ellis, M. Hazas, *A comparison of MDS-MAP and non-linear regression*, in Indoor Positioning and Indoor Navigation (IPIN), 2010 Int. Conf., 2010, pp.1-6.

[3] T. Ellis, D. Makris, J. Black, *Learning a multi-camera topology*, in Joint IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), 2003, pp. 165-171.

[4] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, *KNIGHT: a real time surveillance system for multiple and non-overlapping cameras*, in Proceedings of 2003 International Conference on Multimedia and Expo, 2003, pp. I-649.

[5] D. Makris, T. Ellis, J. Black, *Bridging the gaps between cameras*, in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. II-205.

[6] S. Oh, A. Montanari, A. Karbasi, *Sensor network localization from local connectivity: Performance analysis for the MDS-MAP algorithm*, in Information Theory Workshop (ITW), 2010, pp. 1-5.

[7] A. Rahimi, B. Dunagan, and T. Darrell, *Simultaneous calibration and tracking with a network of non-overlapping sensors*, in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. I-187.

[8] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, *Localization from mere connectivity*, in Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, 2003, pp. 201-212

[9] P. Tseng, *Second-order cone programming relaxation of sensor network localization*, SIAM J. Optim., 2007, vol. 8, no. 1, pp. 156185.