CS231a Milestone: Automatic Camera Network Topology Recognition

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

I. Introduction

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 precalibrated 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 many hours of manual work, and does not scale well as more and 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.

II. Related Work

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 [1], [3]. 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 [4] 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. Similiar calibration steps are needed in [2], and the algorithm is not well detailed. One of the goals of our algorithm is that it will be able to be calibrated with normal daily traffic in the system, and not require any manual calibration at all.

To our knowledge, no paper has been published which thoroughly outlines an algorithm which can estimate the external calibration of a network of cameras without prior information or manual calibration steps.

III. Algorithm

A. Algorithm Hypothesis

The underlying idea behind the algorithm is that, for the planar world case, relative camera orientations can be determined by identifying the image of a line in both cameras, which constrains the relative positions of two cameras to a line parallel to the imaged line, and constrains the difference in camera angles to a specific value. The position of the camera on this line is then further constrained to a point by identifying a point correspondence on the line image.

While there are many ways that such a line and point 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 the first camera, then a speed and direction estimate can be determined. Afterwards, any observation by any camera at a later time can be extrapolated to calculate a corresponce to a specific point on this line in the first cameras coordinate system. If a second camera observes this object, a point correspondence and a line correspondence will be known. This is enough information to determine the relative position and angles of the two cameras in the 2D floorplan, assuming planar motion.

Given a sufficient number of correspondences between many cameras, their external calibration can be determined.

B. Algorithm Overview

Our algorithm first attempts to estimate the relative position and orientation of camera pairs, for each pairing independently, using tracklet observations. This requires algorithms to filter data, and identify the best estimate from noisy observations and false correspondences. This generates a distance measurement between cameras, as well as the angle between each camera and the chord between them. The set of distances between cameras is converted to a best-fit 2D position map using the MDS-MAP algorithm [5]. This determines both the camera coordinates, as well as the positions and relative angles of the chords between them. The camera angles are then determined using a best fit method from the independently estimated angular constraints from each camera pair.

C. Algorithm Theory and Practice

Real world objects, such as people and automobiles will rarely follow a straight path at a constant speed, but we model their behavior as linear with additive Gaussian noise. From each tracklet correspondence we produce a best-fit-line estimate of the path using PCA to derive an orthogonal least squares estimate. Also, a line centroid, average speed, and time-of-centroid estimate are derived. From these lines, velocities, and times, we can extrapolate a correspondence, from which we can generate an estimate of the relative camera positions in the form (θ_1, r, θ_2). Here, r is the distance between cameras, i.e. the length of a chord between them, and θ_1 and θ_2 are the angles between each camera and the chord drawn between them. We refer to these parameters as a ***camera-relation*** estimate. The calculation of (θ_1, r, θ_2) is trivial knowing the correspondence relation between a line and a point in both cameras.

From many of these noisy paths we generate many noisy (θ_1, r, θ_2) estimates. For the simple case described, we would fit a gaussian distribution to the camera position coordinates and determine the most likely coordinate. However, when we also consider cases with false correspondences, this model may not be sufficient.

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. To help ensure this case, once we begin using real data, we will filter our correspondence samples to include the time windows with the fewest number of objects observed summed over all cameras. However, there may still be more than one object, and the tracks may weave through cameras in a non-linear path, which will generate false temporospatial correspondences. This is a side effect of assuming that any object observed in two cameras in a short time period must be the same object. These extra false correspondences will manifest as false camera-relations, and we model them as noise that has very low correlation to the actual camera-relation.

We expect the true correspondences to produce camera-relation parameter estimates with relatively low variance compared to those resulting from false correspondences. Therefore, inferring the identities of the true-correspondences would be achieved by identifying a high density region in the distribution, or a cluster.

Separating two mixed distributions will become much more difficult as the probability of a measurement belonging to the unwanted class increases. Metaphorically, we want a high signal to noise ratio. It is clearly beneficial to reduce the number of false correspondences. As previously mentioned, the first filtering step is to choose data from time windows with the fewest number of observations within the window, which should correlate to there being very few moving objects in the environment. Other cues may be used to identify potential false correspondences, such as a difference in behaviors in different views, including speed or crowd grouping. We would expect these characteristics to change little between views.

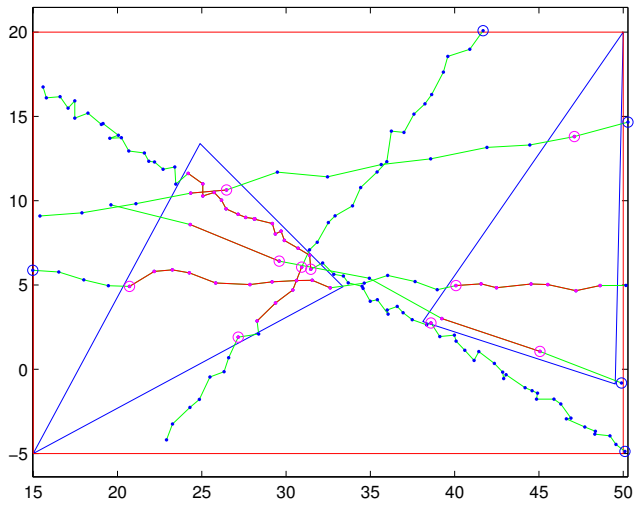

Fig. 1. **Noisy Simulation** Simulation introduces Gaussian noise in both speed and direction of movement across the world

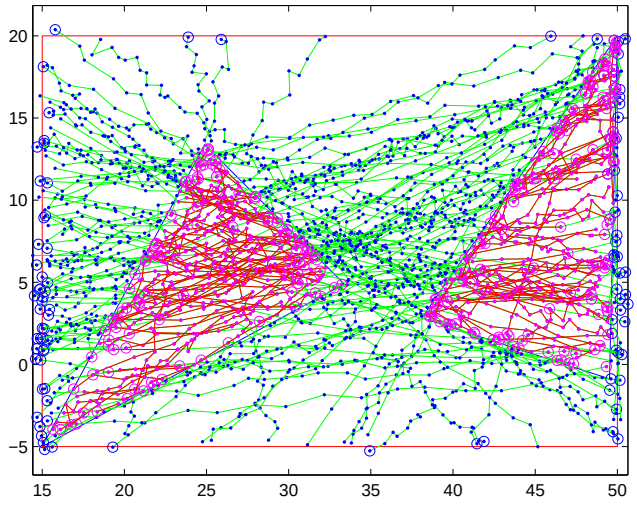

Fig. 2. **Simulated Track Map** Simulation produces tracks across world space (green), which intersect cameras field of view (pink triangle). Data input to localization algorithm is shown in pink

Using various techniques, we hope that a good estimate of the cluster can be produced. We are currently experimenting with histogramming methods and mixtures-of-gaussian methods to perform this task, but work is in its early stages.

Once a cluster is identified, a statistical model can be used to infer the camera-relation between a given pair. An obvious estimate is the mean camera-relation coordinates (θ_1, r, θ_2) within the cluster.

This process may be repeated to estimate pairwise camera-relations between a large subset of cameras. Given the estimated distances from many camera locations, a relative positioning of all cameras can be generated. In general, enforcing

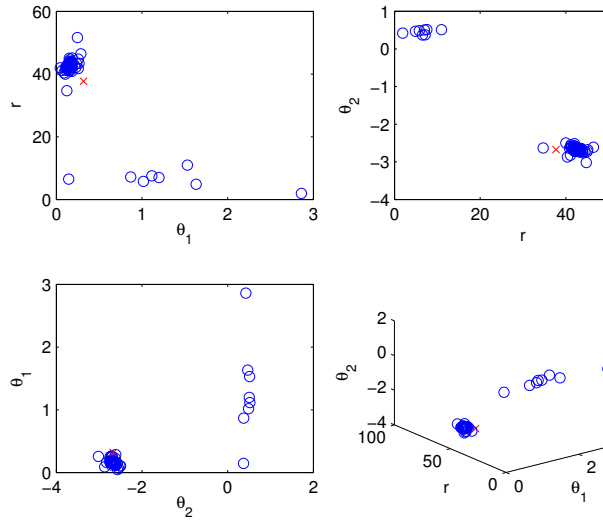

Fig. 3. **Estimated Camera Relation** The algorithm produces an estimate of the camera relation between two cameras for each correspondence. The above plots show the estimates for the camera relation from Figure 2.

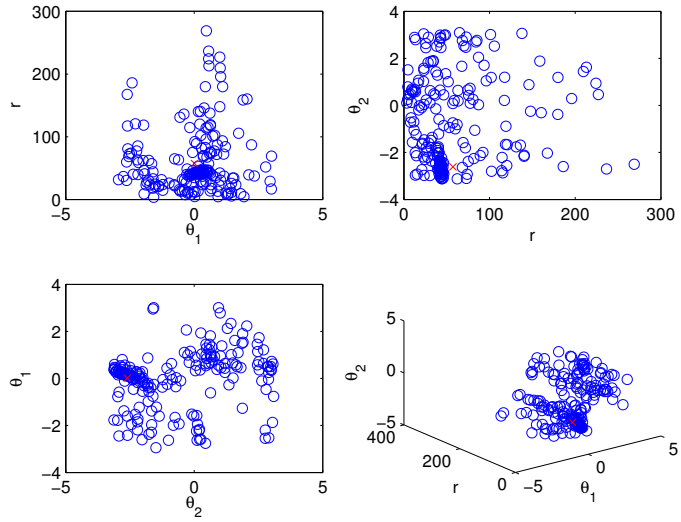

Fig. 4. **Estimated Camera Relation with More False Correspondences** The estimates above are produced from tracks which produce more false correspondences due to multiple tracks entering and exiting the fields of view at a given timestep.

the relative positions within a plane will be an overconstrained problem. The estimated distances between points will be very noisy, so we require an algorithm which will optimize point locations to preserve the planar constraint while minimizing deviation from all estimates. We will use the MDS-MAP algorithm, which essentially uses techniques from PCA to find the plane of highest variance in the data, and the variance in the remaining dimension is the orthogonal noise. This will give optimal relative planar positions between all camera positions, but will not determine their angle, or pose.

Once the points are constrained, the chords between the vertices, and the angles between the chords are fixed. For any camera, the estimated camera-relation angles dictate an estimated angle between the camera and any chord between cameras (since all such chords between a camera and the other cameras are a determined constant). Thus a simple mean may be used to estimate this angle from the camera-relation estimates.

D. Future Work

As mentioned in the discussion of the algorithm, we have several portions of the project underway. We are currently investigating clustering algorithms, and will implement MDS-MAP shortly. Upon completing this step we will have an estimate of the full orientation map for all of the cameras.

We also need to start using real data. The dataset is quite large, and the filtering steps described above will be somewhat time consuming. For example, we need to first determine time windows with few visible objects, to reduce false correspondences. This is a clear next step. We hope to first experiment with the simplest subsets of camera views, modify the algorithm as needed, and scale up to more views or harder situations. During the process and after completion we will evaluate performance compared to the ground truth. We may find that we need to use more elaborate cues to reduce the number of noisy or false correspondences, which will incur further algorithmic filtering steps.

It should also be noted that the algorithm given above treats the global optimization problem as a series of smaller optimization problems, and will likely not reach a global optimal camera configuration given the tracklets. We hope that we can find a tractable method to increase the scope of, and reduce the number of the optimization steps, to yield a better optimum.

IV. Dataset

Our dataset comes from a camera system installed in the Lausanne railway station, provided by Alexandre Alahi. The system uses RGB-D cameras to detect moving objects through the station, and a software pipeline is used to create 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 identity is not tracked between cameras.

V. Results

We ran our current algorithm on simulated data with varying path curvature and additive Gaussian noise (see Figures 1,2), generating plots of the estimated camera relation for each correspondence (see Figure 3). When we injected no noise or false correspondences, the resulting camera relations mapped to exactly the correct values. As we introduced noise, curved paths, varying velocity, and false correspondences, the resulting estimates became more diffuse. However, the data points are still clustered around the correct values.

Figure 3 depicts estimates computed from tracks with little

noise and few tracks per time interval, resulting in very dense clusters about the actual relative orientations of the cameras. Figure 4 shows camera relation estimates produced when multiple tracks occur per time interval, resulting in more false correspondences. There is, however, still a distinct cluster around the actual relative orientations of the cameras.

VI. Milestone Goals

See Table I and II for achieved and remaining milestones.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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

Week	**Milestone**
1/26-2/1	- Determine Project Topic - Write Project Proposal **Thurs 1/30 - Proposal Due**
2/2-2/8	- Literature Survey - Familiarize with Data Set - Begin Algorithm Dev. - Toy Problem
2/9-2/15	- Finish Toy Problem - Algorithm Dev. - Noisy Artificial Data - Write Milestone Progress Report **Thurs 2/13 - Milestone Progress Due**

TABLE I: Milestones Achieved

Week	**Milestone**
2/16-2/22	- Implement algorightm to generate correspondences - Incorporate MDS-MAP - Algorithm Dev. - Handpicked Data - Expand Dataset, Test, Develop
2/23-3/1	- Expand Dataset, Test, Develop
3/2-3/8	- Measure Performance with Fewer Cameras
3/9-3/15	- Produce Graphical Demo of Results **Tues 3/11 - Project Presentations (I)** **Thurs 3/13 - Project Presentations (I)**
3/16-3/19	- Write Final Report **Wed 3/19 - Final Report Due**

TABLE II: Remaining Milestones