Localization of distributed wireless cameras

Paper ID: 108

***Abstract*— Cooperative cameras enable monitoring wide areas and detecting actions and events on a large scale. Due to hardware advancements and economic factors, distributed networks are becoming widely used for a variety of applications ranging from traffic monitoring and surveillance in shopping malls to sports coverage. However, the localization of a large number of cameras in a wide area is not a trivial task. Manual methods are time consuming and can be inaccurate over time. For this reason, we propose an algorithm that uses measurements from the observed objects to perform pair-wise automatic localization of a distributed set of cameras with non-overlapping fields of view. We use the temporal information derived from trajectory information to estimate unobserved trajectory segments, which are then used to estimate the position the cameras on a common ground plane. Furthermore, the exit-entrance direction of the moving objects is used to estimate the relative orientation of adjacent cameras. We demonstrate the algorithm on a distributed network of simulated cameras with wireless communication and compare it with state-of-the-art approaches. The abstract goes here.**

I. Introduction

Nowadays the deployment of a camera network is essential in many applications such as traffic monitoring, surveillance and sports coverage. Camera network provides substantial advantages over a single camera e.g., reduction in occlusions, fewer miss detections and less number of false negative/ positive alarms. However, the use of multiple cameras is not a trivial task. It requires the internal calibration of each camera as well as the external calibration (localization) with other cameras. More often, the high density of cameras and complexity of the environment makes it difficult or expensive to employ manual or GPS-based calibration methods. Therefore, a mechanism is needed that can calibrate the network of the cameras automatically using the information observed in each camera.

Furthermore, hardware advancements and high level software support have enabled the use of distributed wireless camera networks unprecidently at large scale. Each node contains signal processing and communication capabilities. However, these networks work on limited energy and memory resources. This forces the calibration process had to be enough accurate as well as resource efficient.

In this paper, we present a pair-wise camera localization algorithm using trajectory estimation (*CLUTE*) for a distributed wireless network. The algorithm addresses the problem of recovering the relative position and orientation of multiple cameras whose intrinsic parameters are known. *CLUTE* uses temporal information derived from the available trajectory information to estimate the unobserved trajectory segments, in case of cameras with non-overlapping fields-of-view and then used it to position the cameras on a common plane. The object motion information is used to estimate the relative orientation of the cameras. The algorithm is demonstrated on network of distributed cameras simulated in Network Simulator (NS2) and is compared its performance with a centralized approach.

This paper is organized as follows: Sec. II reviews related work in the field of sensor localization. Sec. III formalizes the problem under consideration. Sec. IV provides detailed description of *CLUTE*. Sec. V demonstrates the results and analysis of *CLUTE* on real and simulated data and also compares its performance with two state-of-the-art techniques. Lastly, Sec. VI draws the conclusions.

II. Related work

Localization of a sensor network has gained a considerable attention in recent years [?], [?]. The algorithms presented in this area can broadly be classified into *centralized* and *distributed*. Centralized algorithms ([?], [?], [?], [?], [?]) primarily works on single camera framework, where observations from each camera is stored on a server (e.g., base station), which localize the entire network in accordance with the coordinates of the reference camera. On the other hand, in distributed algorithms ([?], [?], [?], [?], [?]) each camera adjusts itself in accordance with information received from its neighbors. The details of each category of algorithms are given below.

Fisher *et al.* [?] presented a centralized algorithm, which exploits the moving scene features in the near and far fields to calibrate the cameras. A strong assumption is made that object motions are deterministic. Distant objects (e.g., stars) enable the recovery of the orientation (rotation) of the cameras, while close objects (e.g., people or cars) enable the recovery of the translation of the cameras up to a scale factor. Taylor *et al.* [?] used an iterative numerical approach network localization. Each iteration contributes to the reduction of the residual errors using the Newton Raphson's method. Although the simplicity of this approach is a key advantage, however, it heavily depends upon the proper initialization and increment rate to find the global minimum. Rahimi *et al.* [?] used the maximum a posterior (*MAP*) framework for simultaneous calibration and tracking. A network of non-overlapping cameras is localized by using the motion of a target. The *MAP* estimates for the calibration parameters are calculated using the trajectory prior (i.e., the motion model) and the likelihood function, which are constructed from the available observations. The *MAP* approach is highly computationally complex. Furthermore, it is also possible that the solution may place the target inside the field of view of another sensor for which no observations are available at that particular time instance. Javed *et al.* [?] use the concept of velocity extrapolation to project the field

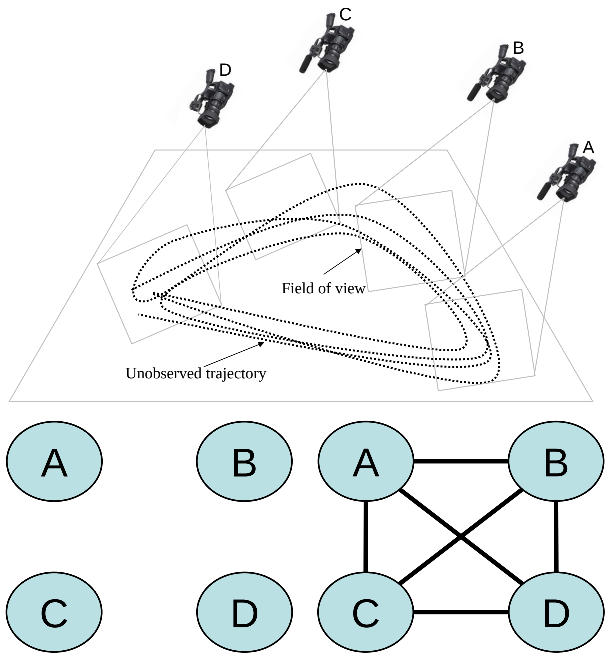

Fig. 1. Example of a scenario with a wireless camera networ alongwith the visual graph and communication graph.

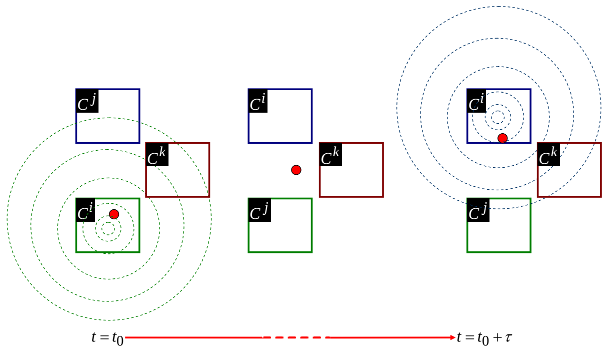

Fig. 2. An illustration of broadcasting information about object's pocession.

of view of one camera onto the other. The projection is then used as a tool to find the calibration parameters. However, the approach assumes that people walk in a straight line in the unobserved regions. Finally, Junejo *et al.* [?] use vanishing points to find the relative orientation of the cameras whose positions are already known.

Mantzel *et al.* [?] presented a localization approach of distributed camera network with partially overlapping fields-of-view. Several images from each camera are communicated across the entire network to learn the scene representation. Geometrical calibration is then employed to localize the network. The approach is simplistic, though, requires considerable resources for understanding the scene. Compressed images may make the approach more resource efficient. Alternatively, a moving target can be used to learn the scene with less resource requirements as used in [?] and [?]. In these approaches few cameras are considered as anchors and the rest are adjusted in the framework of the anchors. More recently, the approach presented by Medeiros *et al.* [?] works in cluster-based way using the moving object. Object tracks observed in overlapping regions are used to calculate the homography matrix for camera localization. However, these approaches are restricted to the network of cameras with overlapping fields-of-view.

III. Problem Formulation

Let $\psi = \{C^1, C^2, ..., C^N\}$ be a distributed network of N wireless cameras with non-overlapping fields-of-view. Let $T_k^i(m) = (x_k^i(m), y_k^i(m))$ be an observation of a moving object O^k within C^i. Also, let each camera C^i, with known field-of-view, provide a vertical top-down view of the scene (i.e., its optical axis is perpendicular to the ground plane or top-down view is approximated from the image plane view using [?]). Under this assumption, the number of parameters for the localization of each camera C^i is reduced to two, namely the camera *position*, $P^i = (p_x^i, p_y^i)$, and the *rotation angle*, ϕ^i. The rotation angle is the rotation of the camera about its optical axis measured with respect to the horizontal axis of the global ground plane. To summarize, the unknown parameters Θ^i for camera C^i are

$$\Theta^i = [p_x^i, p_y^i, \phi^i]. \tag{1}$$

Each camera contains a NxNx3 table with overall inforamtion of the configuration of the network.

A pair of cameras is considered adjacent if O^k object exits from the field-of-view of C^i and enters into the field-of-view of C^j, where $j = i + \eta$ and $\eta \neq 0$, without being observed by another camera. Figure 2 shows an example where an object initially observed by C^i, which propogates the information of holding the target across the network. After τ time, the object observed by C^j, which again propogates the information of seeing the object. Since no other camera propogates the information about holding the object. C^i and C^j becomes the adjacent cameras.

If O^k at position $(x_k^i(m), y_k^i(m))$ exits from C^i at instant m and then enters into the field-of-view of C^j at time $m + \tau$, where $\tau > 0$, then T_k^u represents the unoberserved trajectory segment between the cameras from m to $m + \tau$. A complete trajectory $\mathbf{T}_k$ belong to O_k is thus formed by integrating all the *observed* trajectory segment(s) and the *estimated* trajectory segment(s) in unobserved regions i.e.,

$$\mathbf{T}_k = \bigcup_{i=1}^{N_s} \left(T_k^i, T_k^u\right), \tag{2}$$

where $(C^i, C^j) \in \psi_s \subseteq \psi$ and $N_s \leq N$.

Once $\mathbf{T}_k$ is estimated, we place C^j on the global ground plane with respect to C^i according to the translation vector $\mathbf{D}^u$, which is calculated as:

$$\mathbf{D}^u = (T_k^u(m) - T_k^u(m + \tau)), \tag{3}$$

where t and $t + \tau$ are the instants when O^k exits from C^i and enters into the field-of-view of C^j.

Algorithm 1 One iteration of CLUTE incorporates T_k^i and $T_{\hat{k}}^j$ to place C^i and C^j.

```
$T_k^i = (x_k^i(m), y_k^i(m)) : 0 \leq m \leq M_k^i$; $T_{\hat{k}}^j = (x_k^i(n), y_k^i(n)) : M_k^i + \tau \leq n \leq M_{\hat{k}}^j$;
$Q_v$, $W_w$: Error covariannces; $K$: Kalman gain; $t_c \in T_{\hat{k}}^j$: instant of abrupt change
$(p_x^i, p_y^i, \phi)$=$(0, 0, 0)$
 1: Preprocessing: wavelet decomposition
 2:     $\bar{T}_k^i = \mathcal{G}(T_k^i)$
 3:     $\bar{T}_{\hat{k}}^j = \mathcal{G}(T_{\hat{k}}^j)$
 4: Estimate: unobserved trajectory segment
 5:     $K(m) = Q_v(m-1)L[L^T Q_v(m-1)L + W_w(m)]^{-1}$
 6:   *Forward estimation*
 7:     $\hat{T}_k^i(m+1) = \hat{T}_k^i(m) + K(m)(\bar{T}_k^j(m) - (L\hat{T}_k^i(m)))$: $m = 0 \leq m \leq M_k^i + \tau$
 8:   *Backward estimation*
 9:     $\hat{T}_k^j(m) = \hat{T}_k^j(m+1) + K(m+1)(\bar{T}_k^i(m+1) - (L\hat{T}_k^j(m+1)))$: $m = M_{\hat{k}}^j \geq m \geq M_k^i$
10:   *Fusion*
11:     $\hat{T}_k^u(m)$ = $\alpha(\hat{T}_k^i)(m)$+$\beta(\hat{T}_{\hat{k}}^j)(m)$: $m = 0 \leq m \leq M_k^i + \tau$: $\alpha$+$\beta$=1
12: Estimate: relative position and orientation
13:     $D^u$ = $(x_k^u(M_k^i + \tau) - x_k^u(M_k^i), y_k^u(M_k^i + \tau) - y_k^u(M_k^i))$
14:     $(p_x^j, p_y^j)$=$(p_x^i + D_x^u, p_y^i + D_y^u)$
15:     $\phi^{i,j} = \underset{\theta}{\arg\max} \ \mathcal{L}(\hat{T}_k^j(t_c), T_k^j(t_c) | \theta : \theta = -\pi, ..., +\pi)$
16:     $R^{i,j}$ = $\begin{pmatrix} cos(\phi^{i,j}) & -sin(\phi^{i,j}) \\ sin(\phi^{i,j}) & cos(\phi^{i,j}) \end{pmatrix}$
```

Furthermore, if $(x_{\hat{k}}^{(j)}(m+\tau), y_{\hat{k}}^{(j)}(m+\tau))$ and $(\hat{x}_{\hat{k}}^u(m+\tau), \hat{y}_{\hat{k}}^u(m+\tau))$ are the observed and estimated object positions in C^j on ground plane, then the orientation $\mathbf{R}^{i,j}$ of C^j with respect to C^i is calculated as:

$$\mathbf{R}^{i,j} = \begin{pmatrix} cos(\phi^{i,j}) & -sin(\phi^{i,j}) \\ sin(\phi^{i,j}) & cos(\phi^{i,j}) \end{pmatrix}, \quad (4)$$

where rotation angle $\phi^{i,j}$ is calculated from the observed and estimated object's positions.

IV. Proposed approach

To localize network of the cameras in the environment, we approximate the relative positions of the adjacent cameras using estimated trajectory segments in unobserved regions. If the object O^k is observed by C^i at $t = 0$, C^i broadcast a flag informing the rest of the cameras that the object is being seen by it. If $t = M_k^i$ object exits from C^i, the camera propogates this information as well and waits for it's adjacent camera's flag. Suppose at $t = M_k^i + \tau$ C^j observes the object, it also propagates this information across the network. C^j keeps recording the position of the object till the point of exit. C^j notify the exit of the object to the entire network. C^i and C^j share the trajectory segments observed in each camera's field-of-view to estimate the target's positions in unobserved region and then to find the relative position and orientation. The details of the process is given in subsequent sections.

A. Trajectory estimation in unobserved regions

In order to estimate the objects' positions in unobserved regions using the observations from each camera, we first smooth the trajectory segments as

$$\hat{T}_k^i = \mathcal{G}(T_k^i), \quad (5)$$

where, $\mathcal{G}(.)$ smooths T_k^i using multi-resolution wavelet decomposition method [**?**]. The next step is to reconstruct the trajectories across the unobserved regions. Suppose, the motion model of an object (O_k) is:

$$\hat{T}_k^i(m) = \mathcal{H}(\hat{T}_k^i(m-1), v(m-1)), \quad (6)$$

where, $\mathbf{v}$ is the noise with covariance $\mathbf{Q_v}$. For trajectory estimation between two non-overlapping cameras, we estimate the sequence of positions $\hat{T}_k^i(m+1), ..., \hat{T}_k^j(m+\tau)$ with m is the last state known in C^i and $(m+\tau)$ is the first state known in C^j. There exists a function $\mathcal{H}_f$ for which $\hat{T}_k^i(m) = \mathcal{H}_f(\hat{T}_k^i(m-1), v(m-1))$. This function is the forward model and it computes the current state of the object given the previous state. Similarly, $\hat{T}_k^j(m) = \mathcal{H}_b(\hat{T}_k^j(m+1), v(m+1))$ is the current state of the object given the next state information. We have approximated the state estimates using linear regression model of order 2, assuming that this order is sufficiently model the object's motion behavior. The next step is to combine $\hat{T}_k^i(k)$ and $\hat{T}_k^j(m)$ to have an estimated current state $\hat{T}_k^u(m)$. A viable solution is to take weighted average i.e.,

$$\hat{T}_k^u(m) = \alpha(m)\hat{T}_k^i(m) + \beta(m)\hat{T}_k^j(m), \quad (7)$$

where $\alpha(m) = m/m + \tau : m = 1, ..., m + \tau$ and $\beta(m) = 1 - \alpha(m)$. This weighting scheme gives higher weights to $\hat{T}_k^i(m)$ at the beginning of estimated trajectory and $\hat{T}_k^j(m)$ at the end of the estimated trajectory. The begin state (exit point in C^i) and end state (entrance point in C^j) are assumed to be with minimum error. However, the implicit underlying assumption is that both trajectories should equally contribute to the reconstructed trajectory.

For short unobserved trajectory it is very likely that this assumption is true. For longer unobserved trajectories it becomes more likely that one of the two trajectories performs much better than the other, so that an equal contribution will not result in the best possible reconstruction. To handle this, we use the Kalman filter to correct the forward and backward trajectories before they are combined. The filter is a two step approach. It propagates the state using a prediction and an update step. The state prediction equation (also see Eq. 6) and error covariance matrix are defined as

$$\begin{cases} \hat{T}_k^i(m+1) = \mathcal{H}(\hat{T}_k^i(m), v(m)) \\ Q_v(m+1) = \mathcal{Q}(\hat{T}_k^i(m)) \end{cases}, \quad (8)$$

where $\hat{T}_k^i(.)$ is state estimate. $\mathcal{H}$ translates the object's current state to the next state and $\mathcal{Q}(.)$ propagates the error covariance to next state given the current state estimate. The filter is updated by computing the Kalman gain, $K(m)$, as

$$K(m) = Q_v(m-1)L[L^T Q_v(m-1)L + W_w(m)]^{-1}, \quad (9)$$

where W_w is the covariance of the observation noise and L maps the state vector with the measurements. The object state

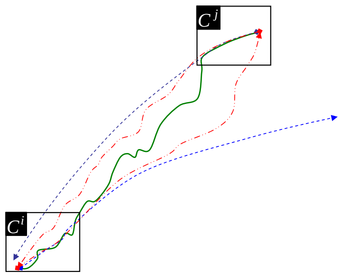

Fig. 3. An example of a comparison between trajectory estimation with and without Kalman filter corrections. *Green* (solid) line shows the original trajectory. *Blue* are the forward and backward estimations *without* corrections and *Red* are the forward and backward estimations *with* corrections. Two trajectories at the bottom show the forward estimation results. Two trajectories at top show the backward estimation results. Arrow head shows the direction of motion.

is updated using

$$\begin{cases} \hat{T}^i_k(m+1) = \hat{T}^i_k(m) + K(m)(Z(m) - (L\hat{T}^i_k(m))) \\ Q_v(m+1) = [I - K(m)L]Q_v(m) \end{cases}, \tag{10}$$

where Z is the observational model. Here, we have replaced $Z(m)$ with $\hat{T}^j_k$ in case of forward estimation i.e.,

$$\hat{T}^i_k(m+1) = \hat{T}^i_k(m) + K(m)(\hat{T}^j_k(m) - (L\hat{T}^i_k(m))), \tag{11}$$

This modification ensures that object will be in correct state at $m+\tau$ in accordance with the observation in C^j. Similarly, for backward estimation, we replace $Z(m)$ in Eq. 10 with $\hat{T}^i_k(m)$ i.e.,

$$\hat{T}^j_k(m) = \hat{T}^j_k(m+1) + K(m+1)(\hat{T}^i_k(m+1) - (L\hat{T}^j_k(m+1))). \tag{12}$$

Equation 13 ensures that estimation results in correct object's state at instant m in accordance with observation in C^i. Figure 3 shows an example of a comparison between trajectory estimation with and without Kalman filter corrections. From the figure it is visible that without Kalman corrections, both forward and backward estimates end far from the original state of the object in cameras' field-of-views. In this particular example, the forward estimation performs poorer than the backward. However, the corrected Kalman filtering method ensures that both forward and backward estimations result in correct objects' states. Equation 7 is then applied on the corrected forward and backward estimated trajectory segments to have a fused reconstructed trajectory segment.

B. Orientation estimation

The relative orientation ($\phi^{i,j}$) between two adjacent cameras C^i and C^j is computed by calculating the angle between the observed object position (x^j, y^j) and the corresponding estimated object position $(\hat{x}^u, \hat{y}^u)$ in camera C^j as described in Sec. **??**.

$$\phi^{i,j} = cos^{-1}\left(\frac{(x^j, y^j).(\hat{x}^u, \hat{y}^u)}{|(x^j, y^j)||(\hat{x}^u, \hat{y}^u)|}\right). \tag{13}$$

However for stable estimation, we calculate the direction of motion for the entire trajectory segment as

$$\mathbf{C} = tan^{-1}(\hat{T}^j_k(m+\tau+s)/\hat{T}^j_k(m+\tau+s-1)) : s = 1, ..., M^j_k. \tag{14}$$

To find the instant of change, we use

$$t_c = \begin{cases} m+\tau+s & \text{if} \quad |\mathbf{C}(s) - \mathbf{C}(s+1)| > \xi \\ m+\tau+M^j_k & otherwise \end{cases}, \tag{15}$$

We extrapolate further the trajectory estimate from instant $m+\tau$ to t_c. We rotate the observed chunk to all possible angles between $-\pi$ and $+\pi$ with an increment of $\pi/180$ (See Fig. **??**) and then the final relative orientation ($\phi^{i,j}$) between cameras C^i and C^j is calculated as

$$\phi^{i,j} = \arg\max_{\theta} \quad \mathcal{L}(\hat{T}^j_k(t_c), T^j_{\hat{k}}(t_c)|\theta : \theta = -\pi, ..., +\pi), \tag{16}$$

where $\hat{T}^j_{\hat{k}}(t_c)$ and $T^j_{\hat{k}}(t_c)$ are chunks of estimated and observed points in C^j to the instant t_c and the likelihood function $\mathcal{L}(.)$ is defined as

$$\mathcal{L} = 1/\mathcal{E}(\hat{T}^j_k(t_c), \mathcal{V}_\theta(T^j_k(t_c))), \tag{17}$$

where $\mathcal{V}_\theta(.)$ rotates $\hat{T}^j_{\hat{k}}(t_c)$ at an angle θ and $\mathcal{E}(,)$ calculates the L_2 distance between the estimated and (rotated) observed trajectory chunks. The localization paramters of the pair of cameras are then broadcast across the entire network.

C. Process termination criterion

Lastly we define the termination of the process as if the change in overall network localization is less than pre-defined ε. We consider the current and the previous locations of each camera for calculating point of stopping as as

$$\sqrt{\sum_{i=1}^{N} |(p^i_x)_{k-1} - (p^i_x)_k| + |(p^i_y)_{k-1} - (p^i_y)_k|} < \varepsilon, \tag{18}$$

where k signifies the current iteration.

V. Experimental results and analysis

A. Simulation of network of N wireless cameras

Fig. 4. Simulation of network of N wireless cameras

Fig. 5. Localization accuracy

VI. Conclusions