Automated localization of a camera network

Nadeem Anjum and Andrea Cavallaro
Queen Mary University of London
Mile End Road, E1 4NS London, UK
{nadeem.anjum, andrea.cavallaro}@elec.qmul.ac.uk

***Abstract*—We present an algorithm for the automated external calibration (localization) of a network of cameras with non-overlapping fields of view, a type of network that is becoming widespread for monitoring large environments. The proposed algorithm exploits trajectory observations in each view and works iteratively on camera-pairs. First outliers are identified and removed from each camera observations. Next, spatio-temporal information derived from the available trajectory is used to estimate unobserved trajectory segments in areas uncovered by the cameras. The unobserved trajectory estimates are used to estimate the relative position of each camera-pair, whereas the exit-entrance direction of each object is used to estimate their relative orientation. The process continues and iteratively approximates the configuration of all cameras with respect to each other. Finally, we refine the initial configuration estimates with bundle adjustment, based on the observed and estimated trajectory segments. The accuracy and the robustness of the proposed approach are demonstrated using both simulated and real datasets, and compared with state-of-the-art approaches.**

I. Introduction

The deployment of camera networks is essential for the observation of large environments in applications such as traffic and behavior monitoring, remote surveillance and sporting events [1]. These networks are characterized by the presence of cameras with non-overlapping fields of view. Prior to applying video analytics in a camera network, internal as well as external calibration of each camera of the network has to be performed [2]. This calibration helps in solving camera hand-overs of targets and reconstructing complete trajectories from partial views [3]. External calibration (also referred to as localization) provides information about the relative position and orientation of a camera with respect to the rest of the network. When the number of cameras is large, it is impractical to employ manual localization techniques. Moreover, the use of GPS-based methods might not be viable all the time, as they are more expensive and there are scenarios when they are not suitable at all (e.g., when monitoring an underground train station). There is therefore the need for an automatic camera localization approach that can exploit the information observed in each camera view.

A. *Prior work*

Existing localization approaches for non-overlapping cameras primarily assume that objects (e.g., pedestrians or vehicles) move linearly in unobserved regions [4]. Under this assumption, the relative position of two cameras can be estimated by extrapolating the trajectory in unobserved regions using the velocity information learnt from the last few observed instances. To estimate the relative orientation of two cameras, vanishing points [5] or far objects with known trajectories (e.g., stars) can be used [6]. There exists various approaches that learn the topology of non-overlapping camera networks using graph representation [7]. In these approaches, camera nodes are considered as vertices and their spatial neighborhood as edges of a graph. The performance of these approaches degrades substantially when the dynamics of the camera network is complex and also when the object motion deviates considerably from the pre-modeled path in unobserved regions [8]. Other approaches track an object while simultaneously estimating the network localization parameters [9]. A Bayesian framework can be used to find unknown parameters (localization and trajectory), given the observations from each camera. Maximum a posterior (*MAP*) is estimated using the Newton-Raphson method that makes the method computationally expensive with increasing number of cameras and length of the trajectory. Furthermore, the performance degrades substantially with noisy observations. In order to address these problems, it is desirable to first estimate trajectory segments in unobserved regions and then to learn the localization parameters based on both observed and estimated trajectory segments [10].

B. *Our contributions*

In this paper, we present an automated localization algorithm that uses trajectories to estimate the configuration of a camera network and is also applicable when cameras have non-overlapping fields of view. The algorithm initially removes outliers from the observations of each camera view. Next, unlike other approaches [4], [7], [9], it iteratively uses spatio-temporal information derived from the available trajectory information to estimate the unobserved trajectory segment that is then used to position the cameras on a common plane. The exit-entrance direction of the object is used to estimate the relative orientation of the cameras. After this initial estimation, the configuration results are refined using bundle adjustment.

Our major contributions are: (a) the relaxation of the linearity constraint on the object motion in unobserved regions (the object can take a sharp turn); (b) no prior information about the environment or the motion model is required; and (c) the initial configuration of the network can be approximated using local pairwise processing followed by refined results achieved offline when batch processing is viable.

The paper is organized as follows. Sec. II formalizes the problem and presents the proposed approach. Sec. III discusses the results. Finally, Sec. IV draws the conclusions.

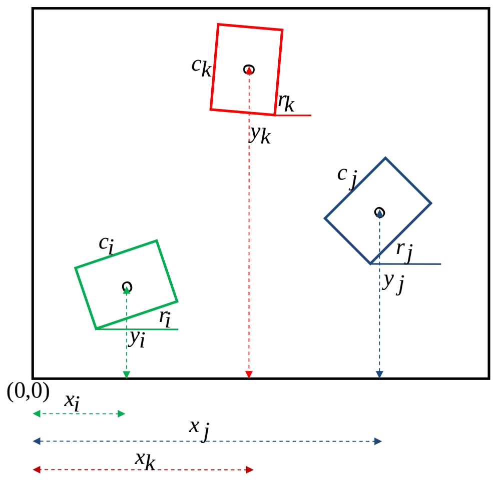

Fig. 1. Illustration of the unknown localization parameters for the definition of the configuration of the camera network. The center of each camera c_i is shown by a circle; (x_i, y_i) is the position of c_i and r_i is the orientation of c_i with reference to the horizontal axis.

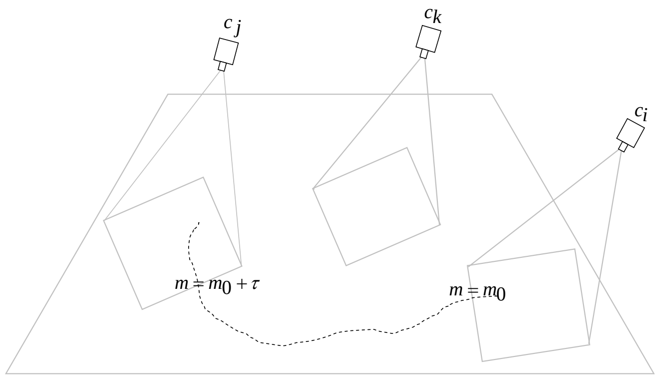

Fig. 2. Example of formation of a camera-pair (c_i and c_j): an object exits from c_i at m=m_0 and enters into the field of view of c_j at m=$m_{0+\tau}$ without being observed by another camera.

II. Proposed approach

A. Problem Formulation

Let $C = \{c_1, c_2, ..., c_N\}$ be a network of N cameras and $z^m = (x^m, y^m)$ the position of a moving object within the field of view[1] of camera c_i at time instant m. Let each camera provide a vertical top-down view of a portion of the scene, either because its optical axis is perpendicular to the ground plane or because its view is normalized. Under this assumption, the parameters for localizing camera c_i are its *position*, (x_i, y_i), and its *rotation angle*, r_i, about its optical axis, measured with respect to the horizontal axis of the ground plane. The set L of unknowns (Fig. 1) is therefore

$$L = \{(x_1, y_1, r_1), (x_2, y_2, r_2), ..., (x_N, y_N, r_N)\}. \quad (1)$$

Moreover, let us define camera c_i and camera c_j as a *camera-pair* i.e., when an object exits the field of view of c_i and enters the field of view of c_j, where $j \neq i$, without being observed by another camera. Note that, according to this definition, c_i and c_j may become a camera-pair, even if c_i and c_j are not physically close to each other (Fig. 2). The flow diagram of the proposed approach for an iteration is shown in Fig. 3. The details of algorithmic steps for the entire network are provided in Algo. 1 and described in the following sections.

B. Pre-processing

The observations in field of view of each camera (trajectory segments) are first pre-processed to remove outliers. As the trajectory segment can be corrupted by various types of estimation noises, we use RANSAC to smooth it before further processing. Instead of using all the available data, RANSAC starts with a small subset of the complete data and then adds data which are consistent with the assumed model. This makes RANSAC able also to filter out noisy observations that are unrelated to the instantaneous function of the object state (beyond the measurement error).

Let the motion parameters be approximated with a polynomial of degree l. We start with n_b observations chosen randomly from a trajectory segment. Next, within this subset, we find the observations that are within a given tolerance of the polynomial model. We repeat the process until the number of observations within this tolerance is more than 50% of the total number of observations of that trajectory segment in the camera view. The observations that fall outside the region of tolerance are considered outliers and therefore discarded from further processing.

Once the trajectories are filtered within the field of view of a camera, they are used to estimate the movement of the object in the unobserved regions, as described in the next section.

C. Trajectory estimation

In order to generate the complete trajectory

$$Z' = \{z^0, ..., z^{m_0}, \hat{z}^{m_0+1}, ..., \hat{z}^{m_0+\tau-1}, z^{m_0+\tau}, ..., z^{m_0+\tau+m_1}\} \quad (2)$$

of the moving object, we use the spatio-temporal information derived from available trajectory segments

$$Z = \{z^0, ..., z^{m_0}, z^{m_0+\tau}, ..., z^{m_0+\tau+m_1}\} \quad (3)$$

in the fields of view of the cameras in order to estimate the trajectory points in the unobserved regions

$$\hat{Z} = \{\hat{z}^{m_0+1}, ..., \hat{z}^{m_0+\tau-1}\}. \quad (4)$$

The estimation of the trajectories in the unobserved region between c_i and c_j is performed in three steps: (i) forward and backward estimations, (ii) Kalman filtering and (iii) fusion. First, we estimate the sequence of positions from time instant m_0, corresponding to the last observation in c_i, to $m_0 + \tau$, corresponding to the first observation in c_j. This estimation is performed in the forward and backward directions. The forward estimation generates the series of candidate positions

$$\tilde{Z}_f = \{\tilde{z}_f^{m_0+1}, ..., \tilde{z}_f^{m_0+\tau-1}\}; \quad (5)$$

[1]For clarity of notation we do not use the subscript i for the position of an object within camera c_i.

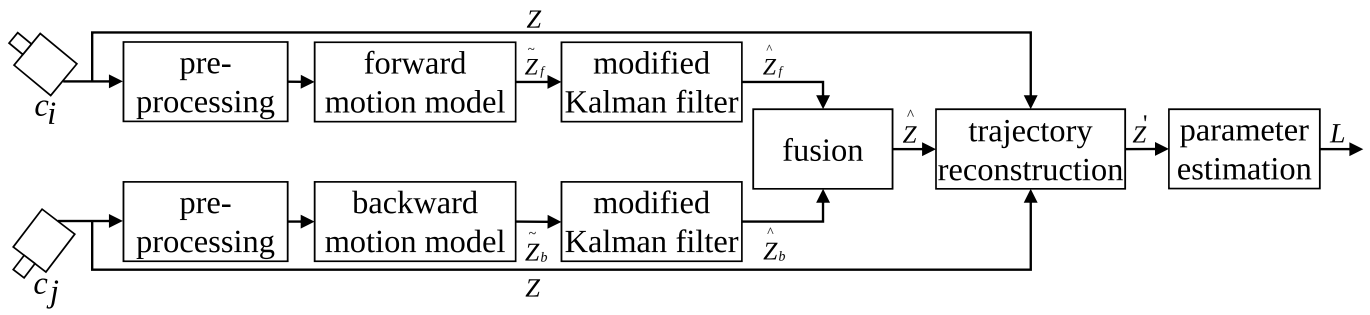

Fig. 3. Iteration of the parameter estimation for a camera-pair.

whereas the backward estimation generates the series of candidate positions

$$\tilde{Z}_b = \{\tilde{z}_b^{m_0+1}, ..., \tilde{z}_b^{m_0+\tau-1}\}. \tag{6}$$

Let each observation $\tilde{z}_f^{m+1}$=$(\tilde{x}_f^{m+1}, \tilde{y}_f^{m+1})$ be generated by a forward motion model as

$$\begin{bmatrix}\tilde{x}_f^{m+1}\\ \tilde{x}_{\nu,f}^{m+1}\\ \tilde{y}_f^{m+1}\\ \tilde{y}_{\nu,f}^{m+1}\end{bmatrix} = \begin{bmatrix}1 & a_{x,f} & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & a_{y,f}\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{x}_f^{m}\\ \tilde{x}_{\nu,f}^{m}\\ \tilde{y}_f^{m}\\ \tilde{y}_{\nu,f}^{m}\end{bmatrix} + \begin{bmatrix}v_x^m\\ v_{\nu,x}^m\\ v_y^m\\ v_{\nu,v}^m\end{bmatrix}, \tag{7}$$

where $(\tilde{x}_{\nu,f}^m, \tilde{y}_{\nu,f}^m)$ is the velocity of the object. Furthermore, $a_{x,f}$ and $a_{y,f}$ are computed independently for each available trajectory segment using the l^{th} order polynomial fitting on observations from c_i. Moreover, $\mathbf{v}^m(\mathcal{N}(0,\Sigma_v^m))$ is modeling additive noise with covariance Σ_v^m=$diag([1e^{-3}, 1e^{-3}, 1e^{-3}, 1e^{-3}])$.

Similarly, let each observation $\tilde{z}_b^m$=$(\tilde{x}_b^m, \tilde{y}_b^m)$ be generated by a backward motion model as

$$\begin{bmatrix}\tilde{x}_b^{m}\\ \tilde{x}_{\nu,b}^{m}\\ \tilde{y}_b^{m}\\ \tilde{y}_{\nu,b}^{m}\end{bmatrix} = \begin{bmatrix}1 & a_{x,b} & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & a_{y,b}\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{x}_b^{m+1}\\ \tilde{x}_{\nu,b}^{m+1}\\ \tilde{y}_b^{m+1}\\ \tilde{y}_{\nu,b}^{m+1}\end{bmatrix} + \begin{bmatrix}v_x^{m+1}\\ v_{\nu,x}^{m+1}\\ v_y^{m+1}\\ v_{\nu,v}^{m+1}\end{bmatrix}, \tag{8}$$

where $a_{x,b}$ and $a_{y,b}$ are calculated like $a_{x,f}$ and $a_{y,f}$, but using the observations from c_j.

Before fusing the forward and backward estimated trajectories, we filter them using a modified Kalman filtering to obtain smoothed estimated positions, i.e., $\hat{Z} = \{\hat{z}^{m_0+1}, ..., \hat{z}^{m_0+\tau-1}\}$. The Kalman gain, k^m, is calculated as

$$k^m = \Sigma_{v^{m-1}} A[A^T \Sigma_{v^{m-1}} A + \Sigma_{w^m}]^{-1}, \tag{9}$$

where Σ_{w^m} is the covariance of the observation noise at instant m and A maps the state vector with the measurements. The object position is updated using a forward motion model as

$$\hat{z}_f^{m+1} = \hat{z}_f^m + k^m(\tilde{z}_b^m - A(\hat{z}_f^m)). \tag{10}$$

Note that $\tilde{z}_b^m$ is used to calculate the residual $\tilde{z}_b^m - A(\hat{z}_f^m)$, which is the discrepancy between the forward estimation and the backward estimation. Zero residual means that the forward and the backward measurements are in agreement. This modification ensures that the object will be in the correct state at $m_0 + \tau$, in accordance with the observation in c_j.

Algorithm 1 Automated localization of a camera network.

```
Variables:
$ID^t$: ID of the camera observing the target at $t$
$I$, $J$: IDs of the first and second cameras in a camera-pair
$\zeta$: overall change in camera positions
$\delta$: minimal change threshold
$\kappa$: iteration number
Initializations
t=1; $\zeta$ = Inf; $\kappa$ = 0
 1: while $\zeta \geq \delta$ do
 2:     increment $\kappa$            % start iterations
 3:     $I = ID^{t-1}$
 4:     while $I$ == $ID^t$ do
 5:         $z_I^{t-1}$ = $(x_I^{t-1}, y_I^{t-1})$
 6:         $m_0$ = t-1
 7:         increment t
 8:     end while
 9:     if target is unobserved then
10:         increment t
11:     else
12:         $J$ = $ID^t$
13:         $m_{0+\tau}$ = t
14:         increment t
15:         while $J$ == $ID_t$ do
16:             $z_J^{t-1}$ = $(x_J^{t-1}, y_J^{t-1})$
17:             $m_{0+\tau+m1}$ = t-1
18:             increment t
19:         end while
20:         apply RANSAC on Z (Sec. II.B)
21:         compute $\tilde{Z}_f$ and $\tilde{Z}_b$ using Eq.(7) and Eq.(8), respectively
22:         compute $\hat{Z}_f$ and $\hat{Z}_b$ using Eq.(10) and Eq.(11), respectively
23:         compute $\hat{Z}$ using Eq.(12)
24:         compute $D_{I,J}^{\kappa}$ using Eq.(13)
25:         compute $r_{I,J}^{\kappa}$ using Eq.(18)
26:         $p_I^{\kappa}$ = $(x_I, y_I)$
27:         $p_J^{\kappa}$ = $(x_J, y_J)$+$D_{I,J}^{\kappa}$
28:         $L_{I,J}^{\kappa} \leftarrow (p_I^{\kappa}, p_J^{\kappa}, r_{I,J}^{\kappa})$
29:         t = $m_{0+\tau}$
30:         $\zeta$ = $\sum_{n=1}^{N} Norm_2(p_n^{\kappa}, p_n^{\kappa-1})$
31:     end if
32: end while                      % end iterations
33: compute $L^*$ using Eq.(19)   % non-iterative refinements
```

Similarly, the object position is updated using the backward motion model as

$$\hat{z}_b^{m+1} = \hat{z}_b^m + k^m(\tilde{z}_f^m - A(\hat{z}_b^m)). \tag{11}$$

Again, in the residual term $\tilde{z}_f^m - A(\hat{z}_b^m)$, $\tilde{z}_f^m$ is used to ensure that the estimated object state at instant m_0 is in accordance with the observation in c_i.

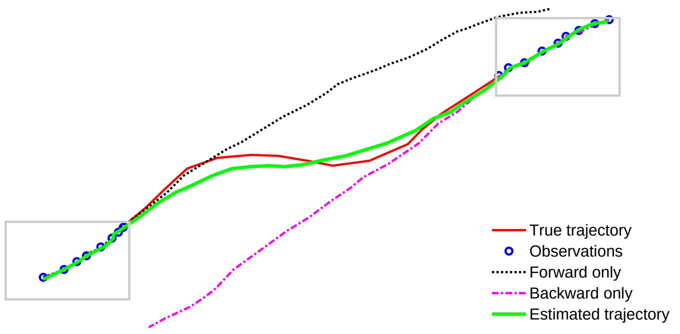

Fig. 4. Trajectory estimation with and without Kalman corrections (Key. *gray*: two non-overlapping cameras; *blue* circles: observations in each camera (Z); *red* line: the true trajectory; *black* dots: forward estimation ($\tilde{Z}_f$); *magenta* line-dots: backward estimation ($\tilde{Z}_b$); *green* line: estimated trajectory ($\hat{Z}$).)

The filtered forward and backward estimated segments are finally fused as:

$$\hat{z}^m = (1-\beta^m)\hat{z}_f^m + \beta^m \hat{z}_b^m, \tag{12}$$

where $\beta^m = m/m_0 + \tau$ for $m = 1, ..., m_0 + \tau$. The forward estimations get higher weights when the object is closer to c_i and these weights are progressively reduced when the object moves away from c_i.

Figure 4 shows a trajectory estimation comparison with and without Kalman filtering corrections. Without the corrections, both the forward and the backward estimates end far from the real object position in the fields of view of the cameras. Kalman filtering followed by fusion improves the estimation of the object position (green).

D. Position and orientation estimation

To localize the cameras we need to calculate their relative position (the distance between their centers) and their relative orientation. To estimate the relative position of c_j with respect to c_i, we use the displacement vector, D_{ij}, calculated as

$$D_{ij} = z^{m_0} - \hat{z}^{m_0+\tau}. \tag{13}$$

Furthermore, to estimate the relative orientation $\hat{r}_{ij}$ of the camera-pair, we could calculate the angle between the observed object position $z^{m_0+\tau}$ and the corresponding estimated object position $\hat{z}^{m_0+\tau}$ in camera c_j as

$$\hat{r}_{ij} = cos^{-1}\frac{(x^{m_0+\tau}, y^{m_0+\tau}).(\hat{x}^{m_0+\tau}, \hat{y}^{m_0+\tau})}{|(x^{m_0+\tau}, y^{m_0+\tau})||(\hat{x}^{m_0+\tau}, \hat{y}^{m_0+\tau})|}. \tag{14}$$

However, to increase the robustness of the orientation estimation process, instead of relying on single instances of observation and estimation, we use a sub-segment of the available trajectory segment in c_j. This sub-segment is extracted using the directional angle $\alpha(.)$, calculated as

$$\alpha(q) = tan^{-1}\frac{y^{m_0+\tau+q} - y^{m_0+\tau+q-1}}{x^{m_0+\tau+q} - x^{m_0+\tau+q-1}}, \tag{15}$$

where $q = 1, ..., m_1$ and m_1 is the number of consecutive observations in c_j. To find the time m_s at which the object changes its direction considerably, we use

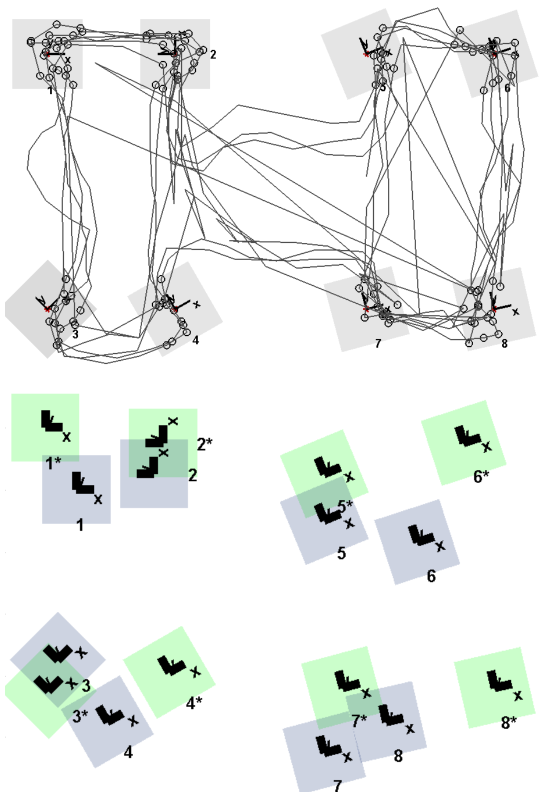

Fig. 5. An example of improvements of the estimation results using bundle adjustment: (top) input dataset and (bottom) localization of each camera, where the estimation before (blue) and after (green and * with each camera ID) bundle adjustments are superimposed.

$$m_s = \begin{cases} m_0 + \tau + q & \text{if} \quad |\alpha(q) - \alpha(q+1)| > \xi \\ m_0 + \tau + m_1 & otherwise \end{cases}. \tag{16}$$

Equation 16 explains that if the change in object direction is significant ($\xi = \pi/12$) at $m_0+\tau+q$ within the field of view of c_j, then we consider the trajectory segment only to that point for estimating the relative orientation; otherwise, the complete trajectory segment is used. To generate an estimated sequence

$$\hat{Z}^s = \{\hat{z}^{m_0+\tau}, ..., \hat{z}^{m_s}\}, \tag{17}$$

we extrapolate further the trajectory estimate from instant $m+\tau$ to m_s. We rotate the observed segments with $\theta = -\pi + n.\frac{\pi}{180}$ where $n = 0, 1, ..., 360$. The final relative orientation, r_{ij}, between c_i and c_j is calculated as

$$r_{ij} = \arg\min_{\theta} \quad \mathcal{D}(\hat{Z}^s, \mathcal{V}_\theta(Z^s)), \tag{18}$$

where $\mathcal{V}_\theta(.)$ rotates Z^s at an angle θ and $\mathcal{D}(.,.)$ calculates the L_2 distance between the estimated and (rotated) observed trajectory segments.

The process of estimating the relative position and orientation continues and iteratively approximates the configuration of all cameras with respect to each other. The process terminates when the total change in position estimation is smaller than a pre-defined threshold, δ, for example $\delta = 1\%$ of the area representing the field of view of a camera.

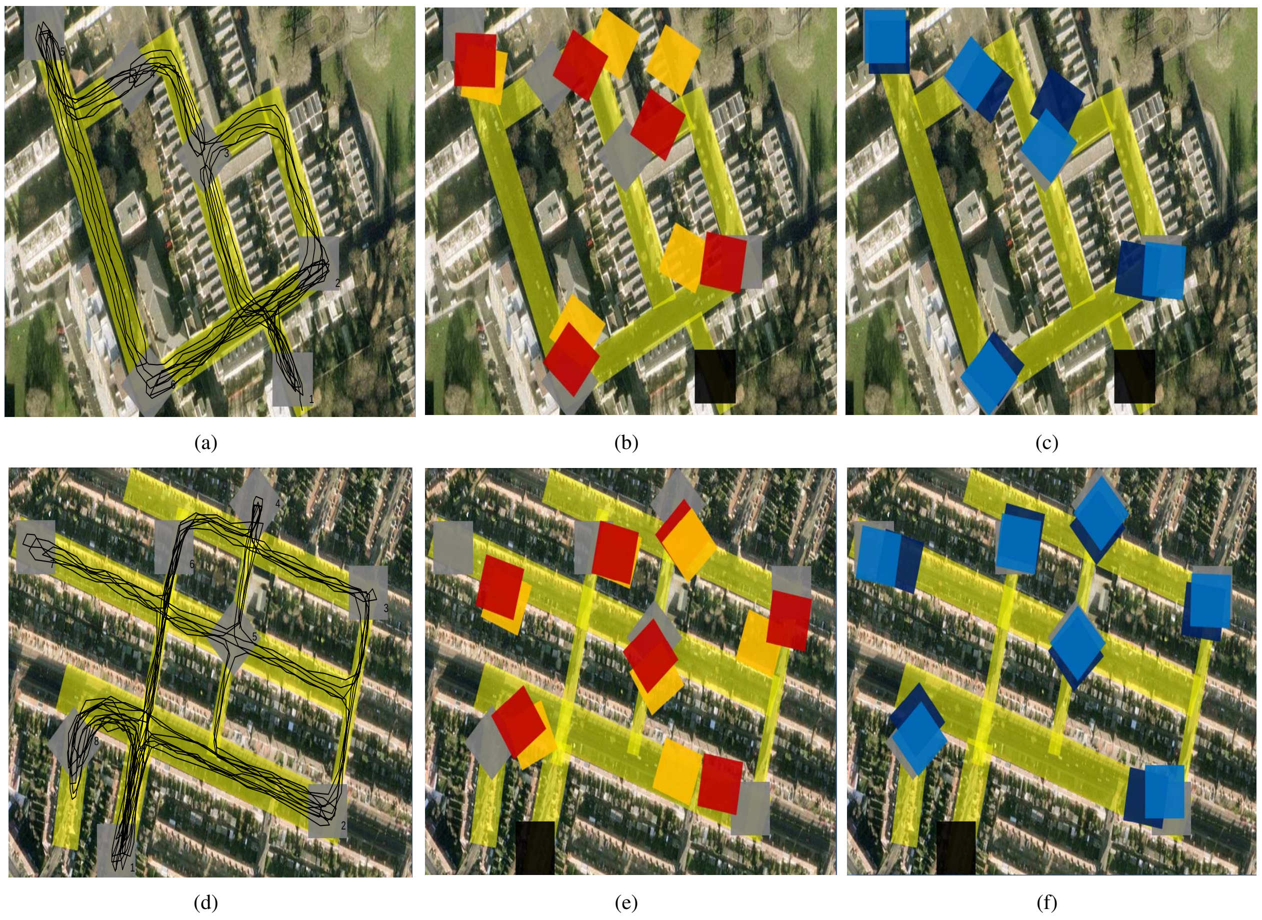

Fig. 6. Examples of camera network localization on simulated datasets: (a,d) trajectory of an object (black line) and field of view of each camera (gray) (Copyright 2010 Google - Map data); (b,e) localization results of MAP (orange) and MMAP (red) superimposed on the true camera configuration; (c,f) localization results of PBBA (dark-blue) and PABA (light-blue) superimposed on the true camera configuration. c_1 (black) is the reference camera.

Finally, we can refine the localization results, $L = \{(p_1, r_1), (p_2, r_2), ..., (p_N, r_N)\}$ with $p_i{=}(x_i, y_i)$, by employing bundle adjustment. The cost function for bundle adjustment is defined to ensure that the localization parameters allow for the minimal difference between observations and estimates:

$$L^* = \underset{L}{argmin} \sum_{i=1}^{N} ||Z_i - r_i(p_i + \hat{Z}_i)||^2, \quad (19)$$

where Z_i and $\hat{Z}_i$ are sets of observed and estimated positions in c_i. Figure 5 shows an example of refinements with bundle adjustment. Figure 5(top) shows the true camera configuration of the network along with the object trajectory. Figure 5(bottom) shows the initial configuration results (blue) that are refined after applying bundle adjustment (green). Significant improvements can be observed in the relative positioning of c_3, c_4, c_7 and c_8.

Additional results and the evaluation of the proposed approach are discussed in the next section.

III. Experimental results

A. Experimental setup

We evaluate the proposed approach and compare it with state-of-the-art algorithms on simulated trajectories with varying levels of complexity (Fig. 6) and on a real dataset generated from five cameras (Fig. 7). The simulated trajectories are grouped in two datasets consisting of six cameras and eight cameras, respectively. In the real dataset, a toy car moves across the network on a car track that is rotated during video acquisition to increase the variability of the trajectory. We use change detection to extract and to generate the trajectory. The coverage area is approximately 4.40 sq. meters. The dataset consists of 19137 frames.

We compare the results of the proposed approach before bundle adjustment (*PBBA*) and after bundle adjustment (*PABA*) with a state-of-the-art *MAP*-based approach [9]. Additionally, we initialize the MAP-based approach with the initial configuration estimated by the proposed approach (PBBA) and refer to it as Modified MAP (MMAP).

The algorithms are developed in Matlab (ver 7.5) and run on a Pentium (R) dual core CPU @ 2.00 GHz with 3.0 GB memory.

B. Discussion

Figure 6(a,d) shows the true camera configurations of two simulated camera networks. For both sequences, c_1 is the reference camera (black). Figure 6(b,e) shows the localization result using MAP (orange) superimposed on the true camera configuration. The results show that there are considerable

TABLE I
EVALUATION OF THE ACCURACY OF THE ALGORITHMS UNDER ANALYSIS ON THE SIMULATED DATASETS.

S	Algo.		c_2	c_3	c_4	c_5	c_6	c_7	c_8	**Avg.**
1	MAP	ϵ_t	20.86	19.83	14.58	11.52	3.46	-	-	**14.05**
		ϵ_r	25.00	16.00	16.00	10.00	10.00	-	-	**15.40**
	MMAP	ϵ_t	24.14	18.50	11.79	11.00	2.70	-	-	**13.63**
		ϵ_r	10.00	15.00	15.00	5.00	2.00	-	-	**9.40**
	PBBA	ϵ_t	2.01	1.12	1.24	0.54	0.51	-	-	**1.08**
		ϵ_r	10.00	15.00	10.00	5.00	0.00	-	-	**8.00**
	PABA	ϵ_t	0.25	0.34	0.12	0.06	0.16	-	-	**0.18**
		ϵ_r	5.00	5.00	4.00	0.00	0.00	-	-	**2.80**
2	MAP	ϵ_t	20.02	27.45	21.11	18.16	14.62	4.74	6.99	**16.16**
		ϵ_r	20.00	20.00	4.00	10.00	18.00	21.00	20.00	**16.14**
	MMAP	ϵ_t	23.83	29.97	20.52	17.61	14.44	4.16	5.84	**16.62**
		ϵ_r	0.00	10.00	2.00	6.00	13.00	16.00	10.00	**8.14**
	PBBA	ϵ_t	2.26	0.82	0.14	0.03	0.61	2.29	0.38	**0.93**
		ϵ_r	10.00	8.00	2.00	4.00	12.00	15.00	4.00	**7.86**
	PABA	ϵ_t	0.61	0.17	0.21	0.21	0.34	0.40	0.61	**0.36**
		ϵ_r	0.00	2.00	1.00	0.00	9.00	10.00	3.00	**3.57**

TABLE II
COMPARISON OF PROCESSING TIME FOR MAP, MMAP, PBBA AND PABA WHEN VARYING THE NUMBER OF UNKNOWN PARAMETERS (CAMERA LOCALIZATION PARAMETERS AND MISSING OBSERVATIONS).

S	No. of param	MAP	MMAP	PBBA	PABA
1	18+1187	310.11	280.39	120.12	173.06
2	24+2118	563.05	401.73	155.71	221.10

errors in approximating the cameras' positions. Specifically, in the first sequence, c_2, c_3 and c_4 are noticeably misplaced. Similarly, for the second sequence, c_2, c_3 and c_7 are far from the true camera configuration. This is primarily because MAP can handle only subtle turns in the unobserved regions. The localization results for MAP are improved when it is initialized with the output of PBBA. On the other hand, Figure 6(c,f) shows the results of the proposed approach, where the initial results (dark-blue) are improved by employing bundle adjustment (light-blue).

Table I summarizes the localization accuracy for MAP, MMAP, PBBA and PABA on the simulated datasets. The position estimation error, ϵ_t, is calculated with the L_2 norm whereas the rotation error, ϵ_r, is calculated with the L_1 norm. In the first sequence MAP has, on average, 13.87 size-units larger position estimation error compared to PABA. This error difference is reduced by 0.42 size-units with MMAP. In terms of rotation, MAP results in 12.60 deg. worse error than PABA. With MMAP this error difference is reduced to 6.6 deg. Similarly, for the second sequence, the average position estimation error for PABA is 0.36 size-units, which is 15.8 size-units smaller than that of MAP and 16.26 size-units smaller than MMAP.

Table II compares MAP, MMAP, PBBA and PABA in terms of processing time. For the first dataset the number of unknown parameters is 1205 (12 for the camera positions, 6 for the camera orientations and 1187 for the missing observations) and for the second dataset the number of unknown parameters is 2118 (16 for the camera positions, 8 for the camera orientations and 2094 for missing observations). By increasing the number of unknown parameters from 1205 to 2118, the processing time for PABA is increased by 27.74% and the processing time for MAP is increased by 81.56%. With a better initialization (i.e., MMAP) the processing time for MAP is increased by 43.02%, which is however still higher than PABA. This confirms that the proposed approach is less time consuming than MAP. This is an important aspect especially for the scalability of a network.

TABLE III
EVALUATION OF THE ACCURACY AND ROBUSTNESS TO CAMERA FAILURE OF THE ALGORITHMS UNDER ANALYSIS ON THE REAL DATASET.

Desc.	Algo.		c_2	c_3	c_4	c_5	**Avg.**
All cameras	MAP	ϵ_t	8.91	12.76	17.88	27.34	**16.72**
		ϵ_r	13.00	17.00	13.00	5.00	**12.00**
	MMAP	ϵ_t	8.90	13.04	17.59	26.98	**16.63**
		ϵ_r	7.00	11.00	13.00	5.00	**9.00**
	PBBA	ϵ_t	0.66	1.82	0.28	0.30	**0.76**
		ϵ_r	7.00	11.00	13.00	4.00	**8.75**
	PABA	ϵ_t	0.08	0.79	0.28	0.28	**0.36**
		ϵ_r	4.00	3.00	1.00	1.00	**2.25**
c_2 fails	MAP	ϵ_t	-	9.36	13.94	26.15	**16.48**
		ϵ_r	-	25.00	18.00	12.00	**18.33**
	MMAP	ϵ_t	-	10.24	16.74	26.43	**17.80**
		ϵ_r	-	18.00	15.00	10.00	**14.33**
	PBBA	ϵ_t	-	0.27	0.93	1.12	**0.77**
		ϵ_r	-	15.00	13.00	9.00	**12.33**
	PABA	ϵ_t	-	0.14	0.49	0.50	**0.38**
		ϵ_r	-	14.00	8.00	7.00	**9.67**
c_3 fails	MAP	ϵ_t	8.95	-	18.10	26.15	**17.33**
		ϵ_r	15.00	-	19.00	12.00	**15.33**
	MMAP	ϵ_t	8.91	-	18.07	26.43	**17.80**
		ϵ_r	15.00	-	17.00	10.00	**14.00**
	PBBA	ϵ_t	0.64	-	0.11	1.11	**0.41**
		ϵ_r	15.00	-	16.00	9.00	**13.33**
	PABA	ϵ_t	0.06	-	0.11	0.49	**0.22**
		ϵ_r	7.00	-	6.00	7.00	**6.67**
c_4 fails	MAP	ϵ_t	8.92	12.76	-	27.15	**16.28**
		ϵ_r	13.00	17.00	-	5.00	**11.67**
	MMAP	ϵ_t	8.89	13.04	-	27.15	**16.36**
		ϵ_r	7.00	11.00	-	5.00	**7.67**
	PBBA	ϵ_t	0.66	1.82	-	0.35	**0.94**
		ϵ_r	4.00	11.00	-	5.00	**6.67**
	PABA	ϵ_t	0.08	0.79	-	0.21	**0.36**
		ϵ_r	4.00	3.00	-	2.00	**3.00**

Figure 7(c) illustrates the localization results of MAP, MMAP, PBBA and PABA on the real dataset. The corresponding localization errors are summarized in Table III. The average position estimation error for the proposed approach is 0.36 mm, while the average position estimation errors for MAP and MMAP are respectively 16.72 mm and 16.63 mm. Similarly, the average rotation error for PABA is 2.25 deg., which is 9.97 deg. smaller than MAP and 6.75 deg. smaller than MMAP.

Furthermore, we also evaluate the robustness of the proposed algorithm to camera failure situations and compare it with MAP. To this end, we dropped all the observations from one camera at a time and applied the algorithms. The results are shown in Fig. 8. Table III summarizes the localization accuracy: c_3 is the most important camera in the network as it contains most of the sharp turns taken by the object; therefore, in case of its failure the localization errors for the rest of network increases substantially; c_4 is the least important camera. In fact, since the trajectory segments in c_4 can be approximated with the observations of c_2, c_3 and c_5, its failure does not affect considerably the performance of network localization.

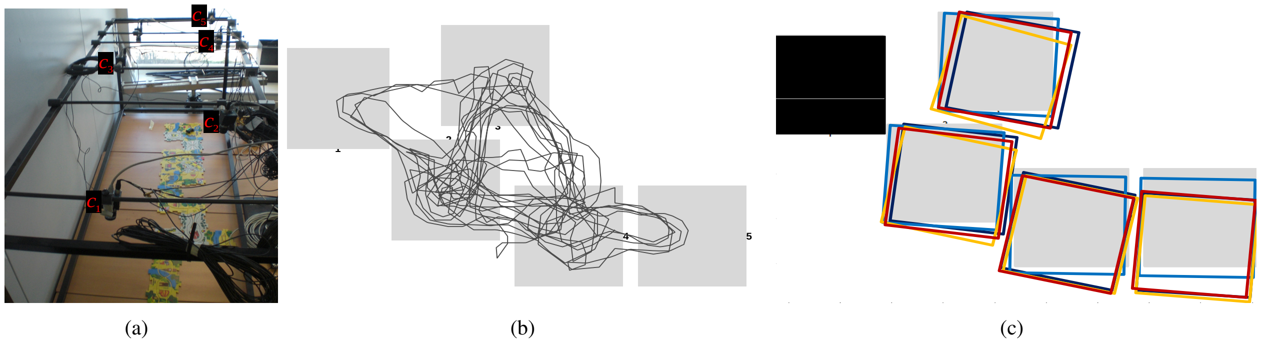

Fig. 7. Localization results on a real dataset: (a) the experimental setup; (b) trajectory (line) superimposed on the true camera configuration (gray); (c) localization results using MAP (orange), MMAP (red), PBBA (dark-blue) and PABA (light-blue) superimposed on the true camera configuration (gray).

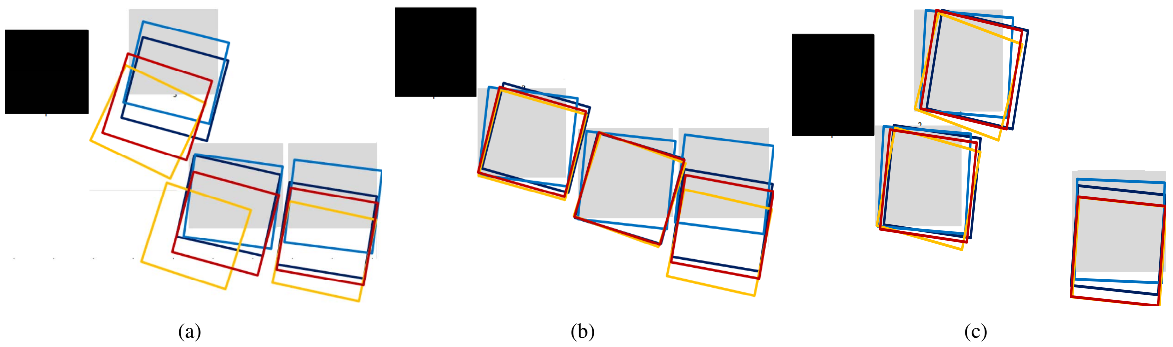

Fig. 8. Localization results in case of camera failure: MAP (orange), MMAP (red), PBBA (dark-blue) and PABA (light-blue) when (a) c_2 fails; (b) c_3 fails; and (c) c_4 fails.

IV. Conclusions

We proposed an algorithm for the localization of a network of non-overlapping cameras that allows us to relax the traditional linearity or smooth-turn constraint on the motion of objects. The algorithm recovers the position and the orientation of each camera based on two main steps: the estimation of the unobserved trajectory in regions not covered by the cameras' fields of view and the estimation of the relative orientations of the cameras. Kalman filtering is initially employed on the available trajectory segments to extrapolate their value in the unobserved regions, with the modification that the forward motion model provides measurements for the backward trajectory estimation and vice versa. This helps in improving the accuracy of the trajectory estimation in the unobserved regions. The relative orientation of the cameras is then obtained by using the estimated and the observed exit and entry point information in each camera field of view. The proposed approach can be applied using local pairwise processing (if a sub-optimal solution is acceptable) or offline, with a refinement based on bundle adjustment. The performance of the proposed approach was demonstrated using both simulated and real datasets.

Our current work includes using other sensor modalities, such as audio [11], to improve the estimation of the trajectory segments in the unobserved regions on real data sequences.

References

[1] A. Dore and C. Regazzoni, "Interaction analysis with a bayesian trajectory model," *IEEE Intelligent Systems*, vol. 25, no. 3, pp. 32–40, April 2010.

[2] D. Marinakis and G. Dudek, "Self-calibration of a vision-based sensor network," *Image and vision computing*, vol. 27, no. 1-2, pp. 116–130, January 2009.

[3] R. Pflugfelder and H. Bischof, "Localization and trajectory reconstruction in surveillance cameras with non-overlapping views," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 709–721, April 2010.

[4] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, "Knight: A real time surveillance system for multiple overlapping and non-overlapping cameras," in *Proc. of Int. Conf. on Multimedia and Expo*, Baltimore, USA, July 2003.

[5] I. N. Junejo, X. Cao, and H. Foroosh, "Autoconfiguration of a dynamic nonoverlapping camera network," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, pp. 803–816, August 2007.

[6] R. B. Fisher, "Self-organization of randomly placed sensors," in *Proc. of Eur. Conf. on Computer Vision*, Copenhagen, Denmark, May 2002.

[7] D. Makris and J. Black, "Bridging the gaps between cameras," in *Proc. of Intl. Conf. on Computer Vision and Pattern Recognition*, Washington-DC, USA, June 2004.

[8] D. Marinakis, G. Dudek, and D. J. Fleet, "Learning sensor network topology through monte-carlo expectation maximization," in *Proc. of Intl. Conf. on Robotics and Automation*, Barcelona, Spain, April 2005.

[9] A. Rahimi, B. Dunagan, and T. Darrell, "Simultaneous calibration and tracking with a network of non-overlapping sensors," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Washington-DC, USA, July 2004.

[10] N. Anjum, M. Taj, and A. Cavallaro, "Relative position estimation of non-overlapping cameras," in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, USA, April 2007.

[11] H. Zhou, M. Taj, and A. Cavallaro, "Target detection and tracking with heterogeneous sensors," *IEEE Journal of Selected Topics In Signal Processing*, vol. 2, no. 4, pp. 505–513, August 2008.