Robust Construction of the Camera Network Complex for Topology Recovery

Edgar Lobaton, Ram Vasudevan, Shankar Sastry, Ruzena Bajcsy
Electrical Engineering and Computer Sciences Department
University of California
Berkeley, CA 94720, USA
Email: {lobaton, ramv, sastry, bajcsy}@eecs.berkeley.edu

***Abstract*—While performing tasks such as estimating the topology of camera network coverage or coordinate-free object tracking and navigation, knowledge of camera position and other geometric constraints about the environment are considered unnecessary. Instead, topological information captured by the construction of a simplicial representation called the CN-Complex can be utilized to perform these tasks. This representation can be thought of as a generalization of the so-called vision graph of a camera network. The construction of this simplicial complex consists of two steps: the decomposition of the camera coverage through the detection of occlusion events, and the discovery of overlapping areas between the multiple decomposed regions. In this paper, we present an algorithm for performing both of these tasks in the presence of multiple targets and noisy observations. The algorithm exploits temporal correlations of the detections to estimate probabilities of overlap in a distributed manner. No correspondence, appearance models, or tracking are utilized. Instead of applying a single threshold on the probabilities, we analyze the persistence of the topological features in our representation through a filtration process. We demonstrate the validity of our approach through simulation and an experimental setup.**

I. Introduction

Though the identification of the exact location of targets and objects in an environment is considered essential in many applications in the realm of sensor networks, there exist numerous tasks wherein simple topological information about the network is sufficient. For example, knowledge of the topology of a camera network makes it possible to design efficient routing and broadcasting schemes as discussed by M. Li et al. [1], or it makes it possible to build powerful coordinate free tracking and navigation algorithms as discussed by E. Lobaton et al. [2]. Given its utility, the construction of this topological information becomes a critical task.

Topological information is traditionally captured through what are called vision graphs where vertices represent sensors and edges represent overlap between their fields of view. Unfortunately the topology of a domain embedded in $\mathbb{R}^2$ is intimately related to detecting holes, and these graphs are unable to detect this information. In order to deal with this shortcoming, E. Lobaton et al. introduced a combinatorial representation for a camera network called the Camera Network Complex, or CN-Complex [2]. It is a simplicial complex built after decomposing the fields of view of cameras at occluding contours and identifying the overlap of two or three sets with edges or triangular faces, respectively (see section III for more details). This representation, which can be thought of as a generalization of a vision graph, was proven to capture the appropriate topological information regarding the

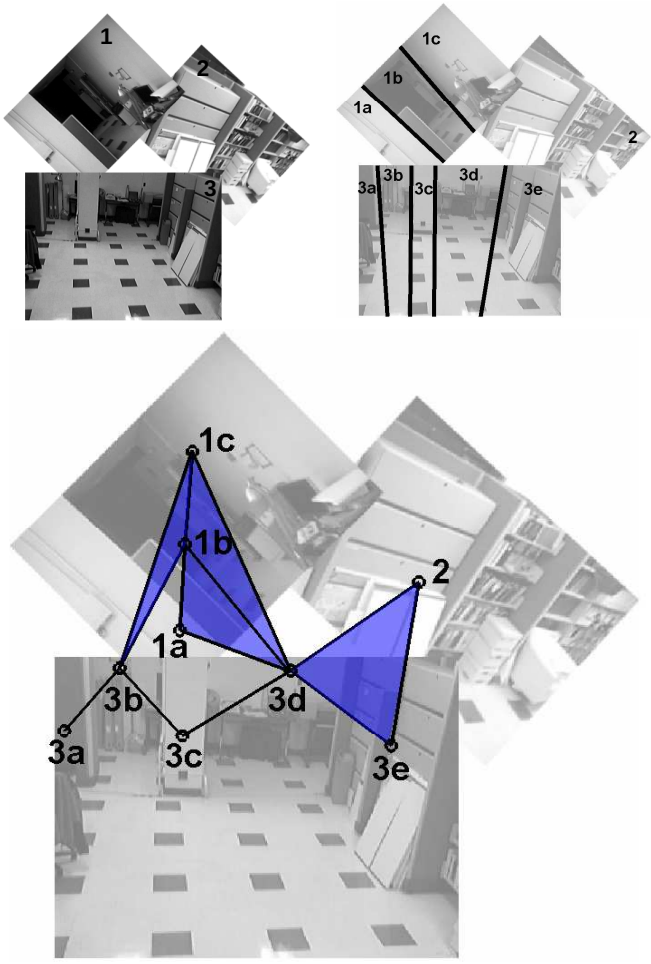

Fig. 1. The CN-Complex for a network of three cameras constructed using the methodology presented in this paper. The views from different cameras (top-left). Bisected views due to occluding objects (top-right). Simplicial complex built by finding the overlap in the coverage of the cameras (bottom). The simplicial complex, correctly, contains a single hole (i.e. the loop with vertices $1a$, $1b$, $3b$, $3c$ and $3d$) that corresponds to the column which acts as an occluding object in the physical coverage.

coverage of a camera network in scenarios where occlusions are due to vertical structures (e.g. walls and entrances). Though simulations and a simple experiment constructed with a single target and perfect foreground detection were presented in the original paper, no algorithm was given for handling multiple targets and noisy observations. This paper will present an approach to constructing the CN-Complex robustly (a sample construction is illustrated in figure 1).

Our approach will only employ temporal correlation between observations rather than a correspondence or appearance model, in order to determine the connectivity information between cameras. A distributed version of the algorithm will be outlined in which data will be processed and stored on sensors with limited computing capability. The result will be a collection of simplices with assigned probabilities of occurrence, which can then be thresholded to select the most likely simplices.

The rest of the discussion proceeds as follows: a brief review of similar work is presented in section II; the required mathematical background is introduced in section III; section IV shows how to find bisecting lines in a robust way; section V describes how to compute points in the intersection between different cameras and outlines a distributed implementation of the process; finally, the validity of this approach is verified in section VI through a real life experiment with multiple targets.

II. Related Work

The recovery of topological information about camera network coverage has generally been pursued by the computation of activity topology and vision graphs. Activity topology refers to the set of possible paths that moving targets can take through the field of views of cameras, and vision graphs are graphs where every node represents a camera view and edges specify the overlap between pairs of cameras. Usually, overlap is determined through the use of correlation between temporal detections, appearance models, or both.

A. van den Hengel et al. [3] introduce an exclusion approach to the solution of the activity topology recovery problem by starting with all possible combinations of topological connections and removing links that are inconsistent with their observations. An evaluation of the method and datasets are made available in [4]. Though their method only relies on detections of the target and avoids the use of appearance models, it has the unfortunate shortcoming of requiring the continuous streaming of detections from each camera. Other approaches include the study of camera networks with overlap through the use of the statistical consistency between observation by Makris et al.[5].

Rahimi et al. [6] describe a simultaneous calibration and tracking algorithm (with a networks of non-overlapping sensors) by using velocity extrapolation for a single target. Funiak et al. [7] introduce a distributed algorithm for simultaneous localization and tracking with a set of overlapping cameras. Stauffer et al. [8] determine the connectivity between overlapping camera views by calculating correspondence models between cameras in order to extract homography models. L. Lo Presti et al. [9] also compute homographies by approximating tracks using piecewise linear segments and appearance models. M. Meingast et al. [10] utilize tracks and radio interferometry to fully localize cameras. Although these approaches involve accurate camera localization, no actual information about the coverage of the network is recovered (i.e. no characterization of the objects occluding the field of view or holes in the coverage is provided).

Marinakis et al. [11] work on constructing a vision graph by finding the connectivity between the overlapping coverage of cameras by using only reports of detection. They employ a Markov model for modeling the transition probabilities and minimize a functional using a Markov Chain Monte Carlo Sampling method. They also present a different formulation of the same problem using "timestamp free" observation [12]. Both of these presentations are computationally expensive and therefore difficult to construct in a distributed manner on sensors.

Other approaches to obtaining a vision graph have been pursued by utilizing target identification as explored by Zou et al. [13]. Cheng et al. [14] build a vision graph in a distributed manner by exchanging feature descriptors from each camera view. In their work, each camera encodes a spatially well-distributed set of distinctive, approximately viewpoint-invariant feature points into a fixed-length "feature digest" that is broadcast throughout the network to establish correspondence between cameras. Yeo et al. [15] utilize a random projection based framework to exchange compact feature descriptors in a rate-efficient manner to establish correspondence between various camera views. Although both of these methods work efficiently, they suffer from being overly restrictive by using appearance models.

In addition to their variety of shortcomings, all of the approaches mentioned to this point have the common shortcoming of pursuing representations that are unable to characterize holes in the coverage. Unfortunately, as we discussed earlier, the topology of a domain embedded in $\mathbb{R}^2$ is intimately related to detecting holes. To address the detection and recovery of holes by topological methods for sensor networks, most assume symmetric coverage (explicitly or implicitly) or a high density of sensors. In particular, Vin de Silva et al. [16] obtain the Rips complex based on the communication graph of the network and compute homologies using this representation. Though this method captures the correct coverage of the sensor network, the assumptions made are generally invalid for camera networks.

E. Lobaton et al. introduce a combinatorial representation for a camera network called the CN-Complex [2] to overcome overly restrictive assumptions and prove that they can capture the appropriate topological information about the coverage of the camera network. This paper will present an approach to constructing the CN-Complex robustly.

III. Background

This section is meant to introduce the mathematical framework to be used throughout the rest of this paper. For a more

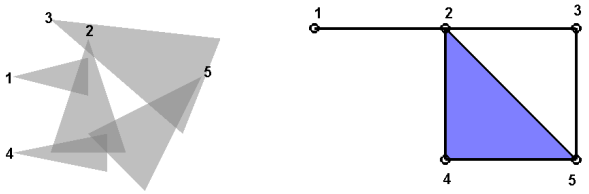

Fig. 2. A collection of sets (left) and corresponding nerve complex (right). The complex is formed by simplices: [1], [2], [3], [4], [5], [1 2], [2 3], [2 4], [2 5], [3 5], [4 5] and [2 4 5].

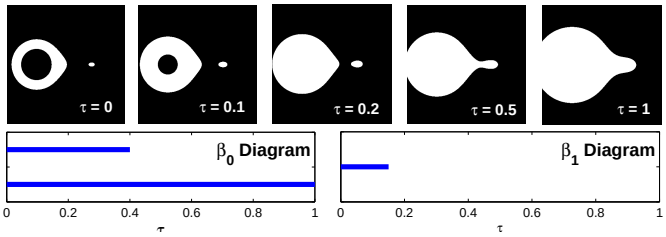

Fig. 3. Snapshots for a family of sets S_τ for $\tau \in [0, 1]$. The collection starts with two connected components which merge at $\tau = 0.4$ as shown in the β_0 diagram. A hole is present until $\tau = 0.15$ as depicted in the β_1 diagram.

formal discussion of the topological concepts presented here, we refer the interested reader to Hatcher [17] or Munkres [18].

A. Simplicial Complex

Definition 1: Given a collection of vertices V we define a k-**simplex** as a set $[v_1\ v_2\ v_3\ \ldots\ v_{k+1}]$ where $v_i \in V$ and $v_i \neq v_j$ for all $i \neq j$. If A and B are simplices and the vertices of B form a subset of the vertices of A, then we say that B is a **face** of A.

Definition 2: A finite collection of simplices is called a **simplicial complex** if whenever a simplex lies in the collection then so does each of its faces.

Definition 3: The **nerve complex** of a collection of sets $\mathcal{S} = \{S_i\}_{i=1}^N$, for $N > 0$, is the simplicial complex whose vertex v_i corresponds to the set S_i and whose k-simplices correspond to non-empty intersections of $k+1$ distinct elements of $\mathcal{S}$.

It is possible to define algebraic structures using this combinatorial representation called **homologies**, which capture all of the topological information about the union of the collection of sets. In particular, it is known that if every finite intersection of the collection of sets $\mathcal{S}$ is contractible, then we can recover the number of connected components and holes in the union of these sets from the nerve complex. This information is recovered from what are called **Betti numbers** where β_0 corresponds to the number of connected components and β_1 corresponds to the number of holes. Importantly, once these have been calculated one can perfectly recover the topological coverage of a sensor network. Software packages such as PLEX [19] can be used to compute these quantities. An example illustrating the construction of a nerve complex can be found in figure 2. In this case, $\beta_0 = 1$ since there is a single connected component and $\beta_1 = 1$ since there is a hole.

B. Persistent Homology

In this section, we informally develop the notion of persistent homology and barcodes, as introduced by Carlsson and Zomorodian [20], [21], to robustly calculate the topological features of a simplicial complex. In order to illustrate the idea, we will consider complexes built from binary images which are easier to visualize. Note that simplicial complexes can be built from a collection of pixels by defining simplices between neighboring pixels. We consider a collection of simplicial complexes, $\mathcal{C}_\tau$ for $\tau \in [0, 1]$ obtained from sets $\mathcal{S}_\tau$, constructed such that $\mathcal{C}_p \subset \mathcal{C}_q$ whenever $p < q$. An example of such a collection is shown in figure 3.

It is possible to track topological features as a function of the parameter τ. Most importantly, these features have a "life span" corresponding to the time at which the feature appears and the time at which it disappears. For example, figure 3 illustrates the collection of pixels start with two connected components which eventually merge at $\tau = 0.4$, and it also illustrates the collection start with a hole that disappears at $\tau = 0.15$. This information is depicted as a persistence diagram at the bottom of figure 3. Carlsson and Zomorodian prove that the computation of these life spans is equivalent to calculating the roots of a polynomial.

We will exploit the persistence of topological features in order to make computations robust to the choice of parameter τ. Consider, for example, figure 4 for which we are trying to calculate the number of connected components. The left image in the figure corresponds to the output of a segmentation algorithm. Using eight-neighbor connectivity we calculate 4 connected components. By comparing this result to the persistence diagram (the right image in figure 4), we recognize this result with an oversensitivity to a particular choice of segmentation. By exploiting the persistence diagram, we arrive at two more reasonable answers to the question at hand: either 1.6 connected components by computing the average number of components over the specified range or a single component since it is the most persistent number of components. We will exploit the persistence of topological features in order to make our computations robust to the choice of parameter τ.

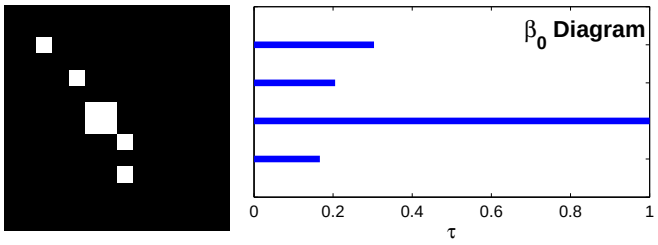

Fig. 4. The S_0 set corresponds to the segmentation of an image (left) and the corresponding β_0 diagram (right). The collection S_τ is obtained by dilating the set S_0 over a chosen range. Given the diagram, we could conclude an average of 1.6 connected components or a single persistent connected component.

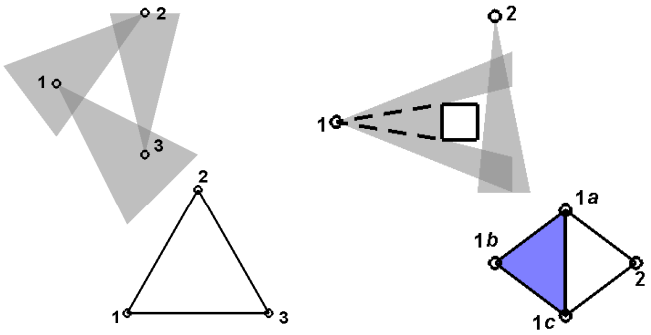

Fig. 5. CN-Complex for two camera networks. Three cameras where no bisecting lines are found due to occlusion (left). In this case, the complex is the same as the vision graph. Two cameras wherein the first camera is split into regions $1a$, $1b$ and $1c$ and the second camera is unsplit (right). The resulting simplicial complex captures the fact that there is a hole in the coverage. Note that a vision graph would only give two nodes and a single edge between them.

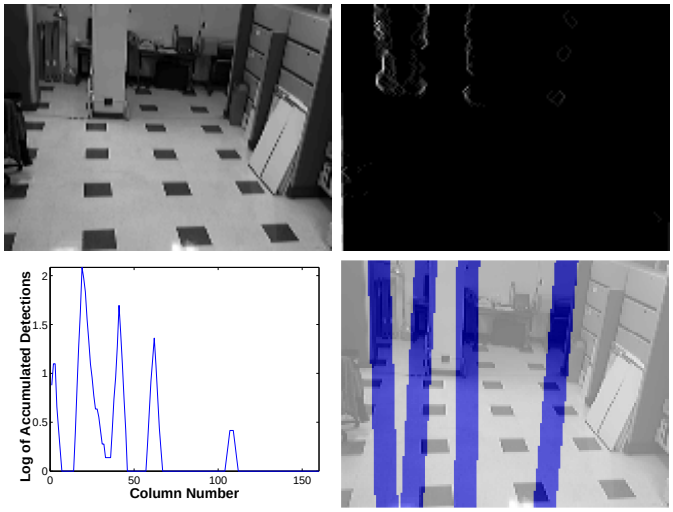

Fig. 6. Steps for finding bisecting lines: For the original view (top-left), the boundaries of the foreground masks are accumulated whenever occlusion events are detected (top-right). Vertical bisecting lines are estimated by aggregating observations over all rows and obtaining the indices of the column in which the highest detections were obtained (bottom-left). Bisecting lines are further refined through a linear fit procedure using the accumulated observations (bottom-right).

C. The CN-Complex

The construction defined here is applicable when targets are moving in a planar domain and walls are erected vertically in the domain. The cameras are at fixed, but unknown locations in this environment. Our goal will be to characterize the **detectable set** of the camera, i.e. the set of points in the domain in which a target is detected by any of the cameras. E. Lobaton et al. [2] proved that the topology of this set is captured by a simplicial representation called the CN-Complex as long as the detectable set for each camera is connected. Figure 5 compares the construction of the simplicial complex to the construction of the vision graph for two basic camera configurations. Notice that the vision graph is unable to detect the hole in the camera coverage.

As figure 1 illustrates, the construction of the CN-Complex proceeds in two steps: (1) image domain decomposition using vertical bisecting lines corresponding to the occluding edges of walls, and (2) finding the overlap between the fields of view of the cameras. The goal of this paper is the presentation of algorithms for the construction of the CN-Complex in the presence of multiple targets and noise in the observations. The analysis of the topology will be performed by employing the persistence of the topological features.

IV. Finding Bisecting Lines

In this section, we address the problem of detecting the bisecting lines that decompose the image domain of a camera. To this end, we assume that we have a simple background subtraction algorithm (e.g. thresholding with respect to a background image). We then utilize an algorithm presented by B. Jackson et al. [22], which consists of accumulating the boundary of foreground objects wherever partial occlusions are detected. In our case, we only store the detections at times when occlusion events occur. We are uninterested in the exact boundary of the objects, but only the bisecting lines. Hence, we will take the simpler approach of first approximating any occluding boundary with vertical lines and then refining the line fit.

In figure 6 (top-left), we observe a camera view with several occluding boundaries due to walls and a column. The accumulated boundaries of the foreground detections are shown on the top-right. Initial estimates for the boundaries are chosen at the peaks of the distributions of detections along each column (bottom-left). Finally, the estimates are refined by performing a least-squares fit on the data with respect to all the points on the boundary that are close to the vertical line estimates. The final result is shown in the bottom-right plot.

V. Finding Intersect Points

In this section, we assume that the bisections within each camera view have been calculated. We focus only on determining the connectivity between camera pairs. More specifically, we look for **intersect points** (i.e. points in the intersection of the field of views of the cameras). In the following discussion, we assume for simplicity that each sensor will be able to uniquely identify any point in its coverage. In other words, we assume a homeomorphism between the image domain from each camera and its coverage. No target identification will be necessary, but localization and recurrence over time will be exploited.

We illustrate our approach by first considering the example in figure 7. We assume that we have two cameras in a room of area 1 with region R_1 in the coverage of camera 1 and R_2 in the coverage of camera 2. We assume that we have N targets, where the probability of a target's location is uniformly distributed over the room. We define D_i^t as the event that there is a detection in R_i at time t, and $\overline{D_i^t}$ is its complement. For simplicity, we assume that we detect a target in R_i if and

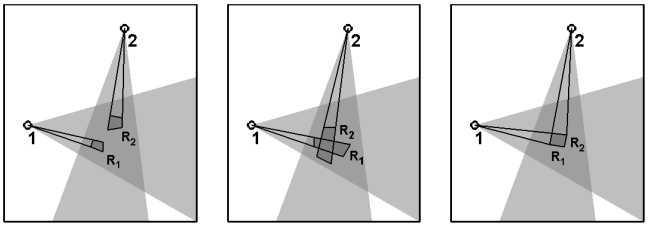

Fig. 7. Geometric depiction illustrating different overlapping configurations and corresponding detection probabilities for 3 targets. Intuitively, whenever R_1 and R_2 are disjoint we expect a low value of $P(D_2^t|D_1^t)$ (left). If $|R_1| \approx |R_2| \approx 0.005$ then $P(D_2^t|D_1^t) \approx 0.01$ by using equation 3. For a partial overlap, we expect a larger probability value (middle). If $|R_1| \approx |R_2| \approx 0.01$ and $|R_1 \cup R_2| \approx 0.015$ then $P(D_2^t|D_1^t) \approx 0.5$. For a perfect overlap, we observe that $P(D_2^t|D_1^t) = 1$.

only if it is actually present (i.e. there are no errors in our detections). Hence, we have

$$P(D_1^t) = 1 - P(\overline{D_1^t}) = 1 - |R_1^c|^N, \tag{1}$$

and

$$\begin{aligned} P(D_1^t \wedge D_2^t) &= 1 - P(\overline{D_1^t \wedge D_2^t}) \\ &= 1 - P(\overline{D_1^t} \vee \overline{D_2^t}) \\ &= 1 - \left(P(\overline{D_1^t}) + P(\overline{D_2^t}) - P(\overline{D_1^t} \wedge \overline{D_2^t})\right) \\ &= 1 - |R_1^c|^N - |R_2^c|^N + |(R_1 \cup R_2)^c|^N \end{aligned} \tag{2}$$

where A^c is the set complement for a set A, and $|A|$ is its area. Therefore, the probability of detecting a target in R_2 given a detection in R_1 is given by

$$\begin{aligned} P(D_2^t|D_1^t) &= \frac{P(D_1^t \wedge D_2^t)}{P(D_1^t)} \\ &= 1 - \left(\frac{|R_2^c|^N - |(R_1 \cup R_2)^c|^N}{1 - |R_1^c|^N}\right). \end{aligned} \tag{3}$$

Intuitively, whenever R_1 and R_2 are disjoint we expect $P(D_2^t|D_1^t) << 1$. For a partial overlap, we expect a larger probability. For a perfect overlap, we expect $P(D_2^t|D_1^t) = 1$. This observations are illustrated in figure 7. In our algorithm $P(D_2^t|D_1^t)$ is utilized as a direct measure of the confidence of the overlap between cameras 1 and 2. A similar argument can be made for detection probabilities between three cameras, i.e. $P(D_2^t \wedge D_3^t|D_1^t)$. Note that $P(D_2^t|D_1^t)$ can be approximated by counting the number of times that a target is detected in R_1 and R_2 whenever there are detections in R_1.

It is possible to bound these conditional probabilities such that values above a given threshold are guaranteed to correspond to a sufficient overlap between two regions. However, such a bound would require knowledge about the distribution of a target's location, the number of targets and the geometry of the environment, but this information maybe unavailable and calculating an arbitrary cut-off may be impossible. Therefore, we employ the filtration process to robustly analyze the observed data in order to avoid making undue assumptions.

A. The Algorithm

In this section, we describe an algorithm to estimate distributely the probabilities $P(D_2^t|D_1^t)$ and $P(D_2^t \wedge D_3^t|D_1^t)$ between two and three cameras respectively. Locally, each camera will make observations and store detections after every occlusion event. These detections will be transmitted to all other cameras, and every time a camera receives a detection message from another camera, the appropriate pairwise counts will be updated. Detections only occur at bisecting lines, which are a subset of the image domain. In our simulations, we will show that this subset will be sufficient to determine whether there is an overlap in coverage between cameras.

Algorithm 1, which is executed every time a new frame is captured, describes how intersection points can be computed and transmitted to all other cameras. The input is a local buffer of observations, Obs, containing detections from previous frames, a sequence of images, $imSeq$, around the current frame at time t, the current time, t, and the identification number, $camID$, of the camera transmitting the points. The function returns updated observations, Obs, and transmits a collection of points whenever an occlusion is detected.

Several functions are used within the previous algorithm. *UpdateObservations* updates a local buffer storing target detections over time using the current images and the corresponding times. *OcclusionDetected* determines if an occlusion event has occurred based on the observations. *ComputePts*

Algorithm 1: **Obs = TransmitPts(Obs,imSeq,t,camID)**

```
Obs = UpdateObservations(Obs,imSeq,t)
if OcclusionDetected(Obs,t)
    Pts = ComputePts(Obs)
    TransmitPts(Pts,t,camID)
end
```

Algorithm 2: **IPts = UpdatingIPts(IPts,Obs,Pts,t,camID)**

```
PPts = getPPts(Pts,Obs,t,camID)
foundPt = zeros(length(PPts),1)
for j = length(IPts) to 1
    if PointIsNotIPt(IPts(j))
        IPts = RemoveIPt(IPts,j)
        continue
    end
    idx = FindCamIDMatch(IPts(i),camID,PPts)
    if isempty(idx)
        continue
    end
    foundIPt = 0
    for i = 1 to length(idx)
        if PointMatch(IPts(j),PPts(idx(i)))
            foundPt(idx(i)) = 1
            foundIPt = 1
        end
    end
    if foundIPt
        IPts(j) = MatchFound(IPts(j),t)
    else
        IPts(j) = MisMatchFound(IPts(j),t)
    end
end
for i = 1 to length(PPts)
    if foundPt(i)==0
        IPts = AddIPt(IPts,PPts(i))
    end
end
```

compiles a list of coordinates for the points where a detection occurred before (if it was a disappearance event) or after (if it was an appearance event) an occlusion. *TransmitPts* periodically sends a list of detection points, a time t, and $camID$ to all other camera nodes.

Algorithm 2 describes what happens at each camera once a packet is received from another camera. The inputs are a list of current intersect points, $IPts$, between the current camera and all other cameras, a list of local observations, Obs, and a list of possible intersect point, Pts, at time t from camera $camID$. The output is an updated list of intersect points. Each entry in $IPts$ corresponds to a potential match between the current camera and another camera, and will contain the $camID$ of the other camera, the coordinates of the intersect point in both camera frames, and detection times. In order to compute the frequency of detections, each entry of $IPts$ will maintain a count of the number of times there were detections in $camID$ and the current camera (we refer to this as a **match**), and how many times there were detections in $camID$ but not the current camera (we refer to this as a **mismatch**).

In the algorithm, *getPPts* returns a list of potential intersect points $PPts$ between cameras by calculating all pairwise combinations between the received detections and the observations at time t. Each entry of $PPts$ will contain both coordinates for the intersection point (the one for the camera in question and the other for the transmitting camera), and also the $camID$ from which the detection points were received. *PointIsNotIPt* estimates the desired conditional probabilities using the formula

$$P(D^t_{local}|D^t_{moteID}) \approx \frac{\#Match}{\#Match + \#Mismatch},$$

and returns 1 if the frequency is too low, in which case we eliminate the intersect point using the function *RemoveIPt*. This is done to ensure the list does not grow too large. *FindCamIDMatch* is a function that returns the matches between the non-local coordinates in the provided intersect point and the list of $PPts$, if $camID$ matches the ID in the provided intersect point, otherwise, it returns an empty list. Coordinate matching is done by allowing for small variations in the coordinate values. *PointMatch* determines if the points match in local and non-local coordinates. *MismatchFound* updates the count of matches in the provided intersect point and stores the detection times. *MatchFound* updates the count of matches in the provided intersect point and detection times. *AddIPt* adds a new intersect point to the list. New points are added when no matches have been found in the original list $IPts$.

We conclude that there is an overlap between two cameras (i.e. a 1-simplex) if there is an entry in the $IPts$ between these cameras with a high detection probability. Similarly, we can find intersections between three cameras (i.e. 2-simplices). Note, we only require up to 2-simplices for the construction of the CN-Complex since we only expect to recover planar information about the coverage.

Data storage and processing occurs distributely. However, in order to analyze the CN-Complex, it is necessary to send the list of interior points to a central node. Of course, this only happens at the end of the observation period and the amount of data to be transmitted will be small.

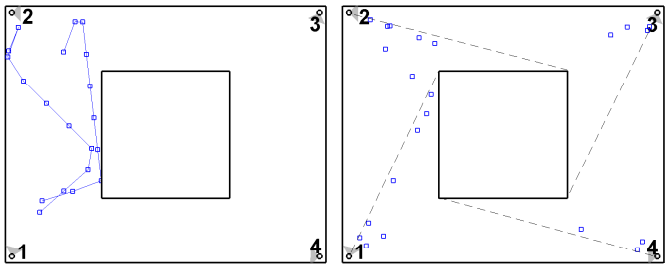

Fig. 8. Layout of a circular corridor setup with two targets moving through the environment (left). Intersect points found, plotted as squares, for a threshold value $\tau = 0.5$ with corresponding bisecting lines, plotted as dashed lines (right).

B. Simulations for Multiple Targets

In this section, we use the previous algorithms to build the CN-Complex in a simulated environment. The 2D environment is made up of objects with piecewise linear boundaries and point targets moving around the environment. Each camera has a conic field of view, is able to detect the targets, and records positions in its local reference frame. All cameras will be assumed to be perfectly synchronized.

As a first example, consider a setup similar to a corridor structure with four cameras located at each corner as shown in figure 8. In this simulation, we consider two targets moving around independently. A short path from each target is displayed in the plot on the left. The cameras bisect every time an occlusion is detected. The resulting bisecting lines are shown in figure 8 (right). After intersect points with corresponding frequencies have been calculated, we threshold on the frequencies of detections. Any frequency value greater than a threshold $1-\tau$ is considered a valid intersect point. We use $1-\tau$ since we want the number of valid intersect points to increase with τ (which guarantees inclusion of complexes as required for the persistence analysis). The right plot in figure 8 show some intersect points found by selecting a threshold of $\tau = 0.5$.

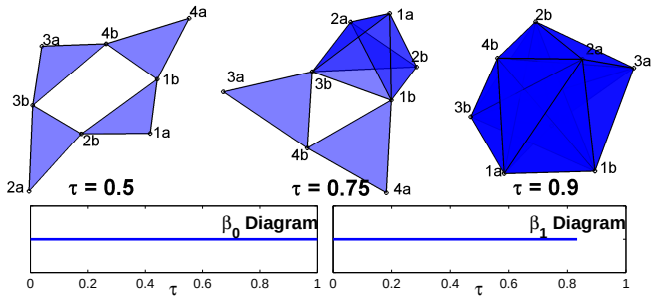

Fig. 9. CN-Complexes for several values of τ (top) and persistence diagrams (bottom) are shown for the circular corridor setup in figure 8. We observe that the diagram shows a persistent single connected component and a persistent loop.

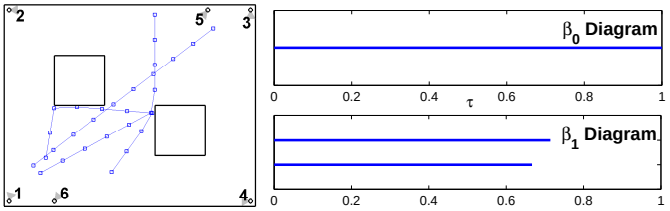

Fig. 10. Layout of an environment with two objects and three targets (left). Corresponding persistence diagrams showing a single persistent connected component and two holes (right).

As described in section III-B, it is possible to analyze the topological structure of the data over all values of τ by employing the persistence of topological features. Figure 9 illustrates the persistence diagrams and several simplices recovered at different thresholds. The diagrams clearly show the persistence of a single connected component and a hole in the layout. Choosing a value of $\tau \in [0, 0.65]$ gives the correct simplices.

Figure 10 (left) illustrates another example in which two objects are placed in an environment with six cameras and three targets. The persistence diagram in the right plot shows the persistence of a single connected component and two holes in the environment, as desired.

VI. Experimentation

In this section, we consider an experimental setup with three cameras placed in indoors. We utilize three computers that are synchronized using the Network Tiem Protocol (NTP) for data acquisition and employ no prior knowledge about the camera locations, no appearance or tracking models, or no knowledge about the number of targets. Though the processing is done offline, the amount of computation and data transmission required are small enough to occur distributedly on a sensor network platform such as CITRIC [23]. The sequence utilized for our analysis corresponds to about 8.5 minutes of recording with the first 3.2 minutes corresponding to a single target, the next 3.3 minutes corresponding to a different single target, and the last 2 minutes corresponding to two targets moving in the environment. Images were captured at about 10 frames per second at a resolution of 320×240.

The physical setup of our experiment is shown in figure 11. Views from the three cameras are shown in the middle row. Importantly, note that though there is overlap between the three cameras, it is nearly impossible to find common features between these views due to the large change in perspective. The decomposed camera views (after finding bisecting lines) are shown at the bottom of the figure. Note that there are three regions in camera 1, one in camera 2, and five in camera 3.

We compute intersect points and corresponding frequencies as described in the previous section. However, instead of considering every possible pixel as an intersect point, we split the image domain into blocks of size 10×10 and treat these regions as our points. In our experiment, we only consider points that have been observed more than 10 times. Figure 12 (left) shows the corresponding simplex when thresholding

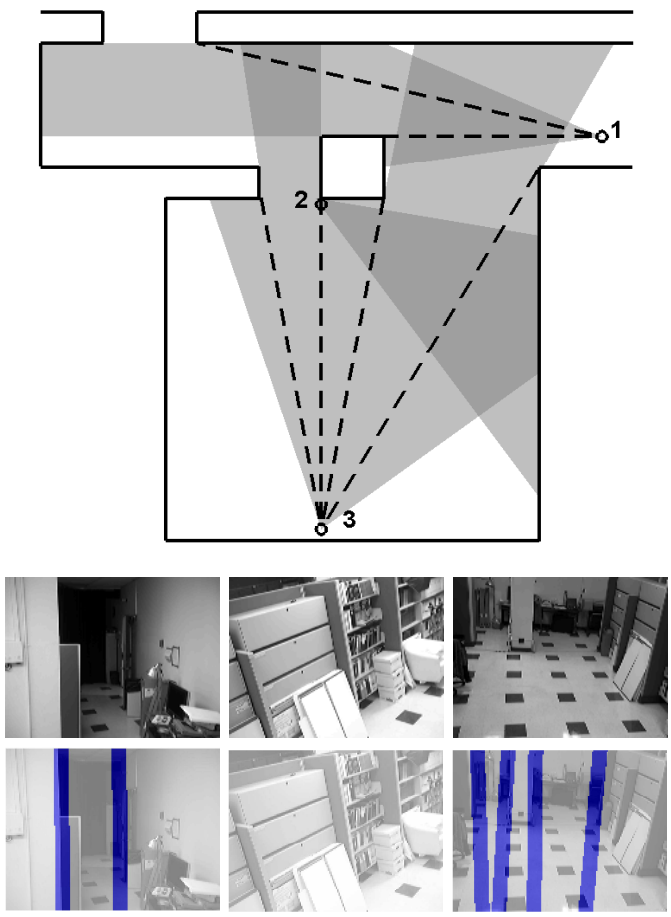

Fig. 11. Experimental setup: Physical layout for cameras in the experiment (top). Views for cameras 1 (middle-left) through 3 (middle-right), and corresponding detected bisecting lines (bottom).

with a value of $\tau = 0.5$. From the right plot, we observe that a single connected component and a single hole in the domain are the persistent topological features in the coverage, as desired.

Since detections are only transmitted after an occlusion event, the transmission rate is low. Figure 13 shows a summary of the number of blocks in which a detection was made for each camera over time.

Table I shows the total number of blocks detected, number

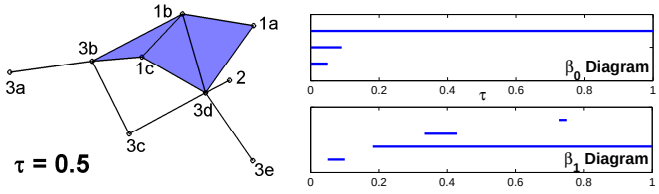

Fig. 12. CN-Complex found for our experiment using a threshold value of $\tau = 0.5$ (left). Persistence diagram for the filtration obtained from the experiment (right). Note, a single connected component and hole are the correct persistent features. The hole is due to the column in the middle of the room.

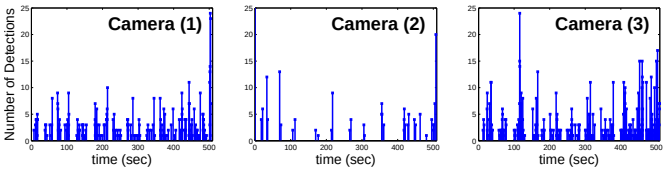

Fig. 13. Plots of the number of block detections per occlusion event over time. We note that the events are relatively sparse (over an 8.5 minutes period), and the number of blocks detected at each time step is under 15 in most cases.

TABLE I
SUMMARY OF DETECTIONS FOR THE WHOLE SEQUENCE

Camera	Total Blocks	Total Frames	Data to be Transmitted
1	609	172	1.9 kBytes / 8.5 min
2	261	37	0.7 kBytes / 8.5 min
3	880	260	2.7 kBytes / 8.5 min

of frames where there was a detection, and the estimated data size for transmission from each camera to the other cameras. The latter quantity is estimated by assigning two bytes to encode each block coordinate and four bytes to encode the time stamp. Note, no additional compression is performed.

VII. CONCLUSION

In this paper, we present a method to construct the simplicial representation of a camera network, which captures information about the connectivity and the number of holes in the coverage. The approach presented in this paper takes advantage of the temporal correlation between detections from different synchronized camera views. The method is designed to work with multiple targets and noisy observations by exploiting the persistence of topological features. Simulations and experiment are used to validate our approach and demonstrate its efficiency in terms of low communications costs.

Though we suggested an algorithm in which the data processing and storing could easily be done distributedly, the actual experimentation was done in a centralized fashion. In the near future, we expect to implement our algorithm on a platform like CITRIC [23] and test it on even larger camera networks.

ACKNOWLEDGMENT

This research work was partially funded by the ARO MURI grant W911NF-06-1-0076, and AFOSR grant FA9550-06-1-0267.

REFERENCES

[1] M. Li and B. Yang, "A survey on topology issues in wireless sensor network," in *Proceedings of the International Conference on Wireless Networks*, 2006.

[2] E. Lobaton, A. Parvez, and S. Sastry, "Algebraic approach to recovering topological information in distributed camera networks," in *Proceedings of the 8th international Conference on Information Processing in Sensor Networks*, 2009.

[3] A. van den Hengel, A. Dick, and R. Hill, "Activity topology estimation for large networks of cameras," in *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*, 2006.

[4] R. Hill, A. van den Hengel, A. Dick, A. Cichowski, and H. Detmold, "Empirical evaluation of the exclusion approach to estimating camera overlap," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

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

[6] A. Rahimi, B. Dunagan, and T. Darrell, "Simultaneous calibration and tracking with a network of non-overlapping sensors," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I–187–I–194.

[7] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of the fifth international conference on Information processing in sensor networks*, 2006.

[8] C. Stauffer and K. Tieu, "Automated multi-camera planar tracking correspondence modeling," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[9] L. L. Presti and M. L. Cascia, "Real-time estimation of geometrical transformation between views in distributed smart-cameras systems," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[10] M. Meingast, M. Kushwaha, S. Oh, X. Koutsoukos, A. Ledeczi, and S. Sastry, "Fusion-based localization for a heterogeneous camera network," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[11] D. Marinakis and G. Dudek, "Topology inference for a vision-based sensor network," in *Proceedings of the Second Canadian Conference on Computer and Robot Vision*, 2005.

[12] D. Marinakis, P. Giguere, and G. Dudek, "Learning network topology from simple sensor data," in *Proceedings of the twentieth Canadian Conference on Artificial Intelligence*, 2007.

[13] X. Zou, B. Bhanu, B. Song, and A. Roy-Chowdhury, "Determining topology in a distributed camera network," in *IEEE International Conference on Image Processing*, 2007.

[14] Z. Cheng, D. Devarajan, and R. Radke, "Determining vision graphs for distributed camera networks using feature digests," *EURASIP Journal on Applied Signal Processing*, vol. 2007(1), 2007.

[15] C. Yeo, P. Ahammad, and K. Ramchandran, "Rate-efficient visual correspondences using random projections," in *Proceedings of IEEE International Conference on Image Processing*, October 2008.

[16] V. de Silva and R. Ghrist, "Coordinate-free coverage in sensor networks with controlled boundaries via homology," *The International Journal of Robotics Research*, vol. 25, pp. 1205 – 1221, 2006.

[17] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002.

[18] J. Munkres, *Topology*, 2nd ed. Prentice Hall, 2000.

[19] "PLEX: A sytem for computational homology," Mar 2009, http://comptop.stanford.edu/.

[20] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas, "Persistence barcodes for shapes," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM New York, NY, USA, 2004, pp. 124–135.

[21] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete and Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.

[22] B. Jackson, R. Bodor, and N. Papanikolopoulos, "Learning static occlusions from interactions with moving figures," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[23] P. C. et al., "CITRIC: A low-bandwidth wireless camera network platform," in *Third ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.