Occlusions in Camera Networks and Vision: The Bridge between Topological Recovery and Metric Reconstruction

Edgar J. Lobaton

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-67
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-67.html

May 18, 2009

Acknowledgement

I want to thank Prof. Shankar Sastry, Prof. Ruzena Bajcsy and Prof. Robion Kirby for all of their valuable feedback and excellent guidance, and Ram Vasudevan and Parvez Ahammad for their contributions to this work. This research work was partially funded by the ARO MURI grant W911NF-06-1-0076, and AFOSR grant FA9550-06-1-0267.

Occlusions in Camera Networks and Vision:
The Bridge between Topological Recovery and Metric Reconstruction

by

Edgar J. Lobaton

B.S. (Seattle University) 2004

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor S. Shankar Sastry, Chair
Professor Ruzena Bajcsy
Professor Robion Kirby

Spring 2009

The dissertation of Edgar J. Lobaton is approved:

Chair Date

Date

Date

University of California, Berkeley

Occlusions in Camera Networks and Vision:

The Bridge between Topological Recovery and Metric Reconstruction

Edgar J. Lobaton

Abstract

Occlusions in Camera Networks and Vision:

The Bridge between Topological Recovery and Metric Reconstruction

by

Edgar J. Lobaton

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor S. Shankar Sastry, Chair

Camera networks are widely used for security and tracking. Knowledge of camera locations and geometric constraints in the environment are usually assumed in order to accomplish these tasks. However, many of these tasks do not require actual localization. Topological information about the network coverage is many times sufficient. In this work, a simplicial representation called the CN-Complex is presented which captures accurate topological information about the coverage of the network. The construction process of this representation relies on the detection of occlusion events. Occlusions are shown to occur when certain generalized topological invariants are violated. The use of these sparse events leads to algorithms which require the extraction of information from continuous observations. The CN-Complex is shown to be useful for navigation and path identification purposes. Augmenting this representation leads to the discovery of relations between camera pairs

providing relative positions at different degrees of accuracy. These relations create a bridge between a purely topological model and a fully localized network. Several theoretical results are shown for occlusion detection and topology recovery, which are then validated by simulations and experiments.

Professor S. Shankar Sastry
Dissertation Committee Chair

To my family,

Everyone that supported me through this journey, and

The Maple that kept me alive

through those moments of dispair.

Contents

List of Figures

List of Tables

Acknowledgments

I want to thank Prof. Shankar Sastry, Prof. Ruzena Bajcsy and Prof. Robion Kirby for all of their valuable feedback and excellent guidance, and Ram Vasudevan and Parvez Ahammad for their contributions to this work. This research work was partially funded by the ARO MURI grant W911NF-06-1-0076, and AFOSR grant FA9550-06-1-0267.

Chapter 1

Introduction

Sensor networks are widely used for tasks such as surveillance, monitoring, and tracking. In order to accomplish these tasks, knowledge of localization information such as camera locations and other geometric constraints about the environment (e.g. walls, rooms, and building layout) are typically considered to be essential. However, there are situations in which the localization of the sensors is unknown (e.g. unavailability of GPS or an ad-hoc network setup). A common approach to overcoming this challenge has been to determine the exact localization of the sensors and reconstruction of the surrounding environment. Nevertheless, there is evidence supporting the hypothesis that many of the tasks at hand may not require exact localization information. In particular, this information is not required for tasks such as estimating the topology of the network coverage, or coordinate-free object tracking and navigation.

In this manuscript, we consider a sensor network with camera sensors where each camera node can perform local computations, and they can extract symbolic/discrete obser-

vations to be transmitted for further processing. This conversion to symbolic representation alleviates the communication overhead for a wireless network. This is a significant benefit as self-localization algorithms can be computationally expensive and require the exchange of large volumes of data. These discrete observations are used to build a model of the environment without any prior localization information of objects or the cameras themselves. Once such non-metric reconstruction of the camera network is accomplished, this representation is used for tasks such as coordinate-free navigation, target-tracking, and path identification.

One of the fundamental questions in the context of camera networks is whether a network is limited to perform only tasks that a single camera can perform, but at a larger scale, or if the total network is "greater" than the sum of the parts. Imagine a camera network where no inter-relationship between the cameras is known. It is natural to ask what the spatial relationship between cameras is. For surveillance applications in which multiple views are certainly useful, it is investigated how object tracking information from multiple cameras can be aggregated and analyzed. A related and important question here involves how to manage the processing and flow of data between the cameras. Note that all of these questions can be approached using knowledge of the topology of the coverage of the network. In particular, topology awareness makes it possible to design more efficient routing and broadcasting schemes as it is discussed by M. Li et al [28]. This knowledge in turn can also aid the control mechanism for more energy-efficient usage.

Let us describe two scenarios in which some weak geometric information can aid in tracking and navigation for a large non-localized camera network:

1. Consider tracking of a target through an urban environment. In this scenario, it may

be of interest to classify the path followed by the target. For example, it would be desirable for the network to specify whether the target went around a specific landmark instead of returning a list of cameras in which the target was visible. This can be accomplished by identifying paths that are homotopic to each other (i.e. that can be deformed continuously from one to another). This allows distinguishing between paths that go around a building clockwise or counter-clockwise without worrying about specific cameras visited. Note that this cannot be done by knowledge of pairwise connectivity between cameras alone (as in the case of so called connectivity/vision graphs).

2. A second scenario where topological information is useful is navigation through an urban environment. This task can be accomplished by making use of local target tracking and a set of directions such as where to turn right, and when to keep going straight. In this case, a general description of the surroundings and the target location is sufficient to guide the target around obstacles.

Figure 1.1 serves as a didactic tool to understand the information required for the approach to coordinate free tracking and navigation problems. Observe that the complete floor plan (left) and corresponding abstract representation (right) serve an equivalent purpose. The abstract representation allows us to track a target and navigate through the environment. The goal in this context is to use the continuous observations from camera nodes to extract the necessary symbols to create this representation.

It turns out that the most useful symbolic information that is extracted comes from occlusions. In the field of computer vision, occlusions are typically considered to be a

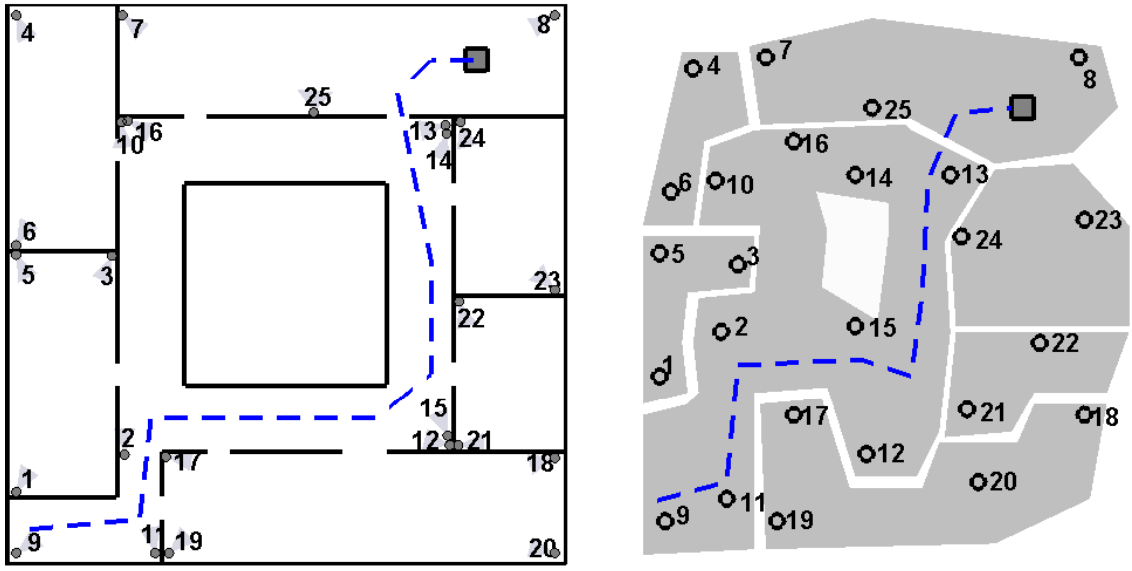

Figure 1.1: A physical layout (left) and an abstract layout (right) for an building floor are shown. Note that the abstract layout contains enough information about the topological structure of the environment and characterizes the space up to a certain degree. This abstraction is sufficient to perform general surveillance and navigation tasks. A dashed path in the physical layout is easily mapped to a path in the abstract layout.

nuisance and are either ignored or overcome. However, in this context, occlusions will be the source of information for the model.

In order to illustrate how symbols can be extracted from occlusions, let us consider a 2D configuration like the one shown in figure 1.2 where cameras and objects are fixed, cameras are capable of detecting targets in their field of views, and the coverage of each camera is a cone. Also, consider a target moving through the environment (as shown in the right plot). Again, the goal is to extract symbols that capture geometric information about this setup.

A prominent geometric feature of the coverage of camera 1 in figure 1.2 is the dashed line drawn in the right plot. This line corresponds to the line of sight in which the target that was once visible by this camera disappears. In figure 1.3, which shows

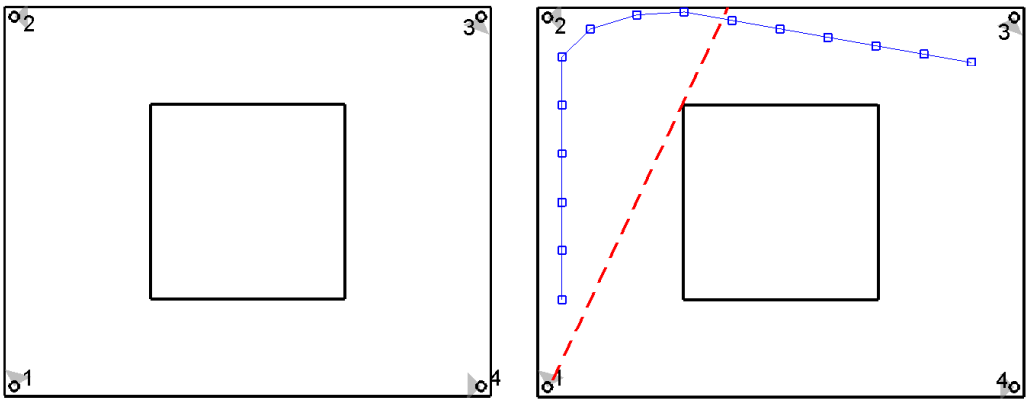

Figure 1.2: A simple layout where four cameras are placed in a configuration that resembles a circular hallway (left). A path for a target and the line of sight (dashed line) in which an occlusion is detected for camera 1 (right).

the detections from each camera over time, it is observed that the target is visible by camera 1 until it crosses this line and moves behind the occluding object in the scene. This corresponds to an **occlusion event** at this time.

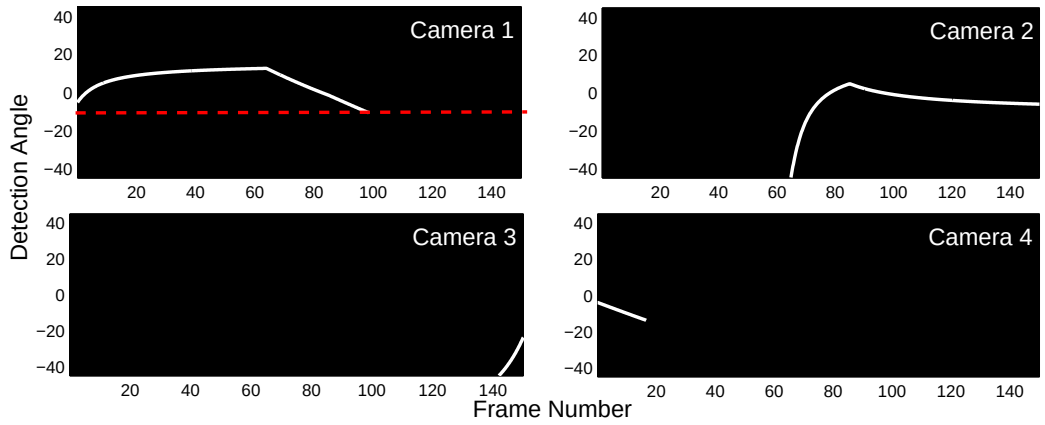

Figure 1.3: Detections over time for the path shown in figure 1.2 (right). The dashed line in the observations of camera 1 correspond to the line of sight in the previous figure.

Assuming that each camera can perfectly track a target, an occlusion event is the event in which a target is "lost" or a new target is "found" by a camera. Hence, there can

be occlusions due to a target leaving the camera's field of view through the boundary of the image domain or due to an occlusion by an object inside the camera's field of view. Of course, the occlusions also have an orientation since the occluding object can be to the left or the right of the location at which the occlusion event was detected. Also, we can distinguish between "appear" or "disappear" events. The following table summarizes the different labels that can be assigned to an occlusion event:

'|'	–	Occlusion due to boundary
'['	–	Occlusion due to object to the right of occlusion angle
']'	–	Occlusion due to object to the left of occlusion angle
'A'	–	Appearance event
'D'	–	Disappearance event

Table 1.1: Labels for different occlusion events

Hence, $D1[$ denotes a disappearance event in camera 1 where the object is inferred to be to the right of the occlusion angle (with respect to the camera coordinate frame), which is consistent with the detections in figure 1.3 for camera 1. Another example is $A2|$ which is observed for camera 2 and corresponds to an appearance event through the boundary.

Occlusions for the trajectory shown in figure 1.4 are shown next:

Detection:	$D4[$	$A2[$	$D1[$	$A3\|$	$D2[$	$A4\|$	$D4\|$	$A4\|$	$D3[$	$A3[$
Angle:	-13^o	-45^o	-11^o	-45^o	-15^o	-45^o	-45^o	-45^o	-6^o	-6^o
Time:	15.3	60.1	97.2	130.3	166.5	203.5	249.3	279.7	312.8	362.6

These symbols and their timing information will be utilized for the construction of a simplicial representation called the CN-Complex which captures appropriate topological information about the camera network coverage.

The rest of this manuscript is organized as follows: Chapter 2 will review the

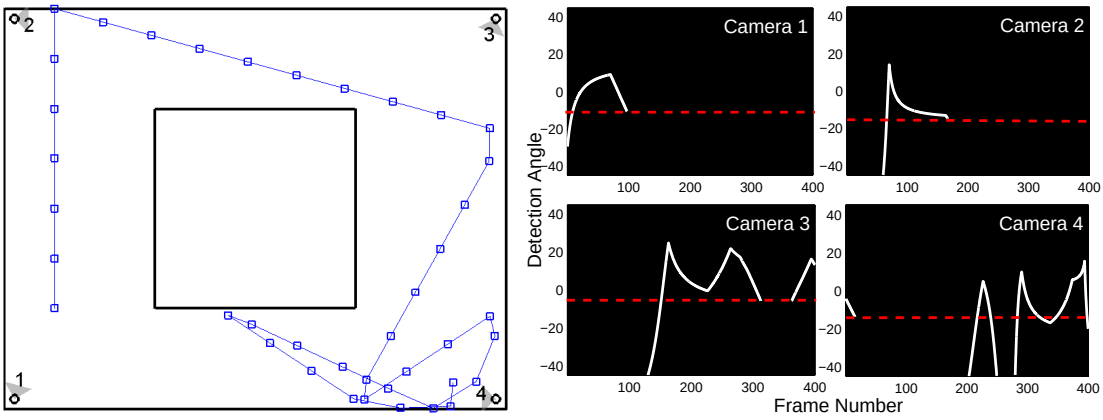

Figure 1.4: A longer trajectory in the environment (left) and corresponding observations over time (right).

mathematical tools necessary to make our discussion formal; chapter 3 will present the construction of a simplicial representation called the CN-Complex under a single target assumption which will address the problem of topology recovery of a camera network coverage; chapter 4 will deal with the robust construction of the CN-Complex in the presence of multiple targets and noise in the observations; chapter 5 will discuss some applications of the CN-Complex to tracking, navigation, and path identification; chapter 6 will discuss how extracting more information from the network can help in the discovery of spatial relations between the cameras leading toward weak localization of the network; finally, chapter 7 will present a mathematical framework for occlusion detection which allows for the generalization of this approach for non-static backgrounds and mobile cameras.

Chapter 2

Mathematical Background

In this section the concepts from algebraic topology that will be used throughout this manuscript are covered. This section contains material adapted from [19, 12] and it is not intended as a formal introduction to the topic. For a proper introduction to the topic, the reader is encouraged to read [36, 25, 19].

2.1 Simplicial Homology

Definition 1. *Given a collection of vertices V, a* k**-simplex** *is a set $[v_1\ v_2\ v_3\ \dots\ v_{k+1}]$ where $v_i \in V$ and $v_i \neq v_j$ for all $i \neq j$. Also, 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 some $N > 0$, is the simplicial complex where vertex v_i corresponds to the set S_i and its k-simplices*

correspond to non-empty intersections of $k+1$ distinct elements of $\mathcal{S}$.

The following statements define some algebraic structures using these simplices.

Definition 4. *Let $\{s_i\}_{i=1}^N$ (for some $N > 0$) be the k-simplices of a given complex. Then, the* **group of k-chains** *C_k is the free abelian group generated by $\{s_i\}$. That is,*

$$\sigma \in C_k \quad \textit{iff} \quad \sigma = \alpha_1 s_1 + \alpha_2 s_2 + \cdots \alpha_N s_N$$

for some $\alpha_i \in \mathbb{Z}$. If there are no k-simplices, then $C_k := 0$. Similarly, $C_{-1} := 0$.

Definition 5. *Let the* **boundary operator** *∂_k applied to a k-simplex $s = [v_1\ v_2\ \cdots\ v_{k+1}]$, be defined by:*

$$\partial_k s = \sum_{i=1}^{k+1} (-1)^{i+1} [v_1\ v_2\ \cdots\ v_{i-1}\ v_{i+1}\ \cdots\ v_k\ v_{k+1}],$$

and extended to any $\sigma \in C_k$ by linearity.

A k-chain $\sigma \in C_k$ is called a **cycle** *if $\partial_k \sigma = 0$. The set of k-cycles, denoted by $\mathbf{Z_k}$, is the $ker\ \partial_k$ and forms a subgroup of C_k. That is,*

$$Z_k := ker\ \partial_k.$$

A k-chain $\sigma \in C_k$ is called a **boundary** *if there exists $\rho \in C_{k+1}$ such that $\partial_{k+1}\rho = \sigma$. The set of k-boundaries, denoted by $\mathbf{B_k}$, is the image of ∂_{k+1} and it is also a subgroup of C_k. That is,*

$$B_k := im\ \partial_{k+1}.$$

Even further, we can check that $\partial_k(\partial_{k+1}\sigma) = 0$ for any $\sigma \in C_{k+1}$, which implies that B_k is a subgroup of Z_k.

Observe that the boundary operator ∂_k maps a k-simplex to its $(k-1)$-simplicial faces. Further, the set of edges that form a closed loop are exactly what we denote by the group of 1-cycles. We will be interested in finding out holes in our domains; that is, cycles that cannot be obtained from boundaries of simplices in a given complex. This observation motivates the definition of the homology groups.

Definition 6. *The* k**-th homology group** *is the quotient group*

$$H_k := Z_k / B_k.$$

The **homology** *of a complex is the collection of all homology groups. The rank of* H_k*, denoted the* k**-th betti number** β_k*, gives us a coarse measure of the number of holes. In particular,* β_0 *is the number of connected components and* β_1 *is the number of loops that enclose different "holes" in the complex.*

2.2 Example

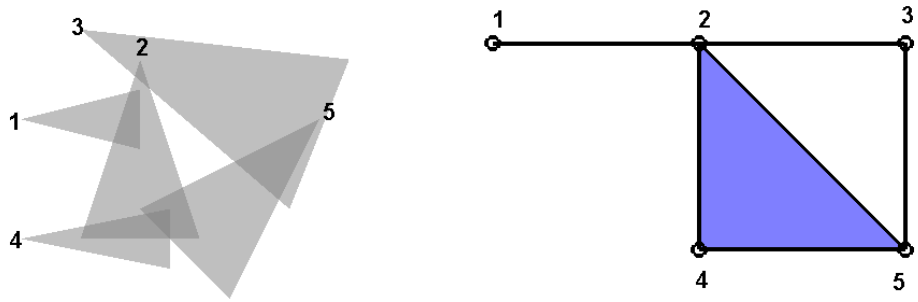

Figure 2.1: A collection of sets (left) and corresponding nerve complex (right). The complex is formed by simplices: [1], [2], [3], [4], [5], [1 2], [2 3], [2 4], [2 5], [3 5], [4 5] and [2 4 5]. Pictorially, 1-simplices can be represented by edges and 2-simplices by triangles.

In figure 2.1 we observe a collection of triangular shaped sets labeled from 1 to 5. The nerve complex is obtained by labeling the 0-simplices (i.e., the vertices) in the same

way as the sets. The 1-simplices (i.e., the edges in the pictorial representation) correspond to pairwise intersection between the regions. The 2-simplex correspond to the intersection between triangles 2, 4 and 5.

For the group of 0-chains C_0, we can identify the simplices $\{[1], [2], [3], [4], [5]\}$ with the column vectors $\{v_1, v_2, v_3, v_4, v_5\}$, where $v_1 = [1, 0, 0, 0, 0]^T$ and so on.

For C_1, we identify $\{[1\ 2], [2\ 3], [2\ 4], [2\ 5], [3\ 5], [4\ 5]\}$ with the column vectors $\{e_1, e_2, e_3, e_4, e_5, e_6\}$, where we define $e_1 = [1, 0, 0, 0, 0, 0]^T$ and so on.

Similarly for C_2, [2 4 5] is identified with $f_1 = 1$.

As mentioned before, ∂_k is the operator that maps a simplex $\sigma \in C_k$ to its boundary faces. For example, we have:

$$\partial_2[2\ 4\ 5] = [4\ 5] - [2\ 5] + [2\ 4] \quad \text{iff} \quad \partial_2 f_1 = e_6 - e_4 + e_3,$$

$$\partial_1[2\ 4] = [4] - [2] \quad \text{iff} \quad \partial_1 e_3 = v_4 - v_2.$$

That is, ∂_k can be expressed in matrix form as:

$$\partial_1 = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad \partial_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}.$$

Since $C_{-1} = 0$,

$$H_0 = Z_0/B_0 = ker\ \partial_0/im\ \partial_1 = C_0/im\ \partial_1.$$

It can be verified that

$$\beta_0 = dim(H_0) = 1.$$

Hence, we recover the fact that there is only one connected component in the diagram of figure 2.1. Similarly, it can be verified that

$$\beta_1 = dim(H_1) = dim(Z_1/B_1) = dim(ker\ \partial_1/im\ \partial_2) = 1,$$

which tells us that the number of holes in our complex is 1. Also, $H_k = 0$ for $k > 1$ (since $C_k = 0$).

2.3 Čech Theorem

Now we introduce the Čech Theorem which has been used in the context of sensor networks with unit-disk coverage [12] and has been proved in [9]. Before proceeding any further, the following definitions are required:

Definition 7. *Given two spaces X and Y, a* **homotopy** *between two continuous functions $f_0 : X \to Y$ and $f_1 : X \to Y$ is a continuous 1-parameter family of functions $f_t : X \to Y$ for $t \in [0, 1]$ connecting f_0 to f_1.*

Definition 8. *Two spaces X and Y are said to be of the same* **homotopy type** *if there exist functions $f : X \to Y$ and $g : Y \to X$ with $g \circ f$ homotopic to the identity map on X and $f \circ g$ homotopic to the identity map on Y.*

Definition 9. *A set X is* **contractible** *if the identity map on X is homotopic to a constant map.*

In other words, two functions are homotopic if it is possible to continuously deform one into the other. Also, a space is contractible if it is possible to continuously deform it to a single point. It is known that homologies are an **invariant** of homotopy type; that is, two spaces with the same homotopy type will have the same homology groups.

Theorem 1. (Čech Theorem) *If the sets* $\{S_i\}_{i=1}^{N}$ *(for some* $N > 0$*) and all nonempty finite intersections are contractible, then the union* $\bigcup_{i=1}^{N} S_i$ *has the homotopy type of the nerve complex.*

That is, given that the required conditions are satisfied, the topological structure of the union of the sets is captured by the nerve. It is observed that in figure 2.1 all of the intersections are contractible. Therefore, we can conclude that the extracted nerve complex has the same homology as the space formed by the union of the triangular regions.

2.4 Persistent Homology

In this section, we informally develop the notion of persistent homology and barcodes which were introduced by Carlsson and Zomorodian [10, 56]. Consider a collection of complexes Σ_τ for $\tau \in [0, 1]$ such that $\Sigma_p \subset \Sigma_q$ for $p < q$. Note that simplicial complexes can be built from a collection of pixels S_τ by defining simplices between pixels that are neighbors of each other. Throughout this section binary images will be used to illustrate these concepts. An example of such a collection is shown in figure 2.2. A collection of images S_τ is introduced such that the simplicial complexes Σ_τ built from these images have the desired inclusion property.

It is possible to track topological features as a function of the parameter τ. Most

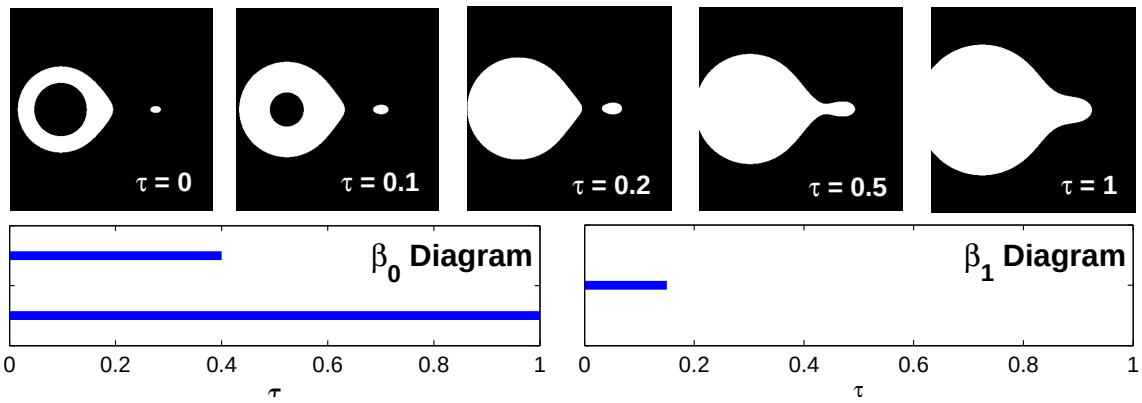

Figure 2.2: Snapshots of a family of images S_τ for $\tau \in [0, 1]$. The collection starts with two connected components which merge at $\tau = 0.4$ as shown in the β_0 diagram. A hole is present until $\tau = 0.15$ as depicted in the β_1 diagram.

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 2.2 shows the collection of pixels start with two connected components which eventually merge at $\tau = 0.4$, and it also shows the collection start with a hole that disappears at $\tau = 0.15$. This information is depicted as a persistence diagram / barcode at the bottom of figure 2.2. Carlsson and Zomorodian prove that the computation of these life spans is equivalent to calculating the roots of a polynomial.

The persistence of topological features will be used to make computations robust to the choice of parameter τ. Consider the example in figure 2.3 resulting from a color segmentation scheme and for which it is our goal to determine the number of connected components. The left image in the figure corresponds to the output of a segmentation algorithm. By using eight-neighbor connectivity, 4 connected components are computed. By comparing this result to the persistence diagram (the right image in figure 2.3), this result is identified with an oversensitivity to a particular choice of segmentation. By exploiting the

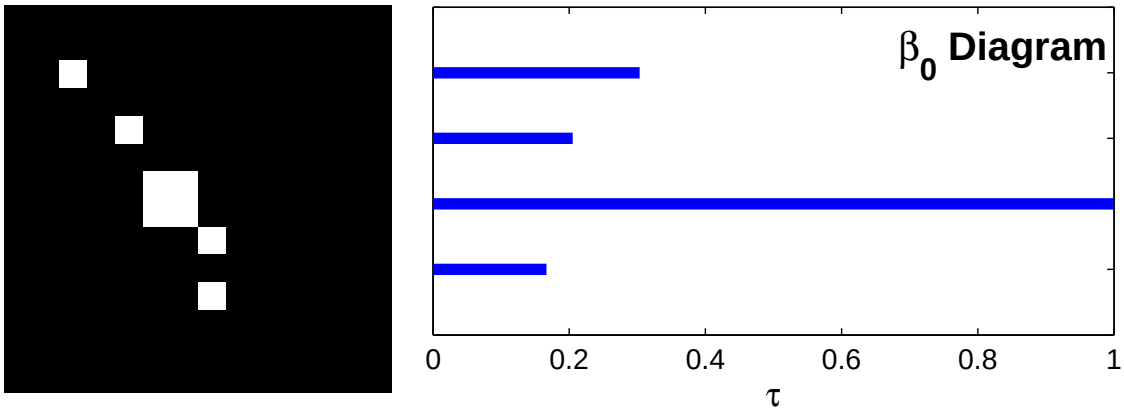

Figure 2.3: A binary image S_0 resulting from color segmentation (left) and the corresponding β_0 diagram (right). The collection of segmentations S_τ correspond to dilating the image S_0 over a chosen range. Given the persistence diagram, we could conclude an average of 1.6 connected components or a single persistent connected component.

persistence diagram, it is possible to 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.

Chapter 3

The CN-Complex

This chapter deals with the recovery of topological information from the coverage of a camera network. The goal of this chapter is to formalize the mathematical framework in which a simplicial complex (called the CN-Complex) is built and proof that it captures the appropriate topological information. In later chapters it will be shown how to construct this complex robustly, how to refine this model, and how to utilize it for navigation, tracking and path identification. The work presented in this chapter is adapted from the work by Lobaton et al[29].

The rest of the discussion is as follows: Section 3.1 gives a brief discussion about different approaches to capturing topological information in sensor networks and the related work in this domain; sections 3.2 and 3.3 contain the main theoretical contributions defining the problems and the assumptions made for topological recovery of the camera network coverage; simulations and an experiment are discussed in sections 3.4 and 3.5. The contributions of this chapter include the introduction of formalized topology recovery prob-

lems by making explicit assumption about the environment model, and the utilization of topological information for tracking and navigation.

3.1 Related Work

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

A. van den Hengel et al [50] 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 not consistent with their observations. An evaluation of the method and datasets are made in [20]. Their method only relies on detections of the target and it resembles one of the algorithms proposed in this chapter.

Marinakis et al [31] work on finding connectivity between non-overlapping coverage of cameras by using only reports of detection and no description of the target. They use a Markov model for modeling the transition probabilities and minimize a functional using Markov Chain Monte Carlo Sampling. They also present a different formulation of the same problem with "time-stamp free" observation with only ordering available (still no target description) [32].

Cheng et al [11] 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 [53] utilize a random projection based framework to exchange compact feature descriptors in a rate-efficient manner to establish correspondence between various camera views.

Connectivity between overlapping camera views by determining the correspondence models between cameras and extracting homography models has been approached by Stauffer and Tieu [47]. L. Lo Presti and M. La Cascia [43] also compute homographies by approximating tracks using piecewise linear segments and apparence models. M. Meingast et al [34] utilizes tracks and radio interferometry to fully localize the cameras.

Other approaches to solving the same problem with target identification have been explored by Zou et al [57]. Camera network with overlaps have been studied using the statistical consistency of the observation data by Makris et al[30]. Rahimi et al [44] 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 [17] introduce a distributed algorithm for simultaneous localization and tracking with a set of overlapping cameras.

Finding the topology of a domain embedded in $\mathbb{R}^2$ is closely related to detecting holes. There has been much work on the detection and recovery of holes using topological methods in sensor networks, most of which considers symmetric coverage (explicitly or implicitly) or high enough density of sensors in the field. In particular, Vin de Silva and

Ghrist [12] obtain the Rips complex based on the communication graph of the network and compute homologies using this representation. These methods assume some symmetry in the coverage of each sensor node (such as circular coverage), however, such assumptions are not valid for camera networks.

3.2 The Environment Model

In this section the assumptions made about the environment are made explicit. Even though they may seem very restrictive, they are introduced in order to simplify the problem and facilitate the analysis.

3.2.1 The Problem in 2.5D

The problem will be defined in terms of the detection of a target moving through an environment. For the sake of mathematical clarity, we first focus on the case of a single target moving through the environment. Let us start by describing our setup:

The Environment in 2.5D : We consider a domain in 3D with the following constraints:

- All objects and cameras in the environment will be within the space defined by the planes $z = 0$ (the "floor") and $z = h_{max}$ (the "ceiling").
- Objects in the environment consists of static "walls" erected perpendicular to our plane from $z = 0$ to $z = h_{max}$. The perpendicular projection of the objects to the plane $z = 0$ must have a piecewise linear boundary. Objects must enclose a non-zero volume.

Cameras in 2.5D : A camera α has the following properties:

- It is located at position o_α^{3D} with an arbitrary 3D orientation and a local coordinate frame Ψ_α^{3D}.

- Its **camera projection in 3D ,** $\Pi_\alpha^{3D} : \mathcal{F}_\alpha \to \mathbb{R}^2$, is given by

$$\Pi_\alpha^{3D}(p) = (p_x/p_z, p_y/p_z),$$

  where p is given in coordinate frame Ψ_α^{3D}, and $\mathcal{F}_\alpha \subset (\{(x, y, z) \,|\, z > 0\})$, referred to as the **field of view (FOV)** of the camera, is an open convex set such that its closure is a convex cone based at o_α^{3D}. The image of this mapping, i.e. $\Pi_\alpha^{3D}(\mathcal{F}_\alpha)$, will be called the **image domain** Ω_α^{3D}.

The Target in 2.5D : A target will have the following properties:

- The target will be a line segment perpendicular to the bounding planes of our domain which connects the points $(x, y, 0)$ to (x, y, h_t), where x and y are arbitrary and $h_t \leq h_{max}$ is the height of the target. The target is free to move along the domain as long as it does not intersect any of the objects in the environment.

- A target is said to be **detected** by camera α if there exists a point $p := (x, y, z)$ in the target such that $p \in \mathcal{F}_\alpha$ and $\overline{o_\alpha^{3D}\, p}$, where $\overline{o_\alpha^{3D}\, p}$ is the line segment between o_α^{3D} and p, does not intersect any of the objects in the environment.

Note that these assumptions may seem very restrictive, but they are satisfied by most camera networks in indoor and outdoor environments. Also, some of these choices in our model (such as the vertical line target and polygonal objects) are made in order to simplify our analysis. We will see that the approach works in real-life scenarios through the experiments.

The example in figure 3.1 shows a target and a camera with its corresponding FOV.

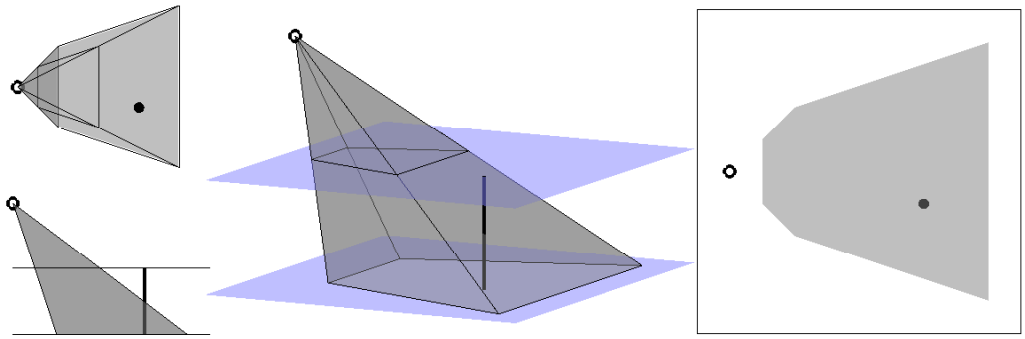

Figure 3.1: Mapping from 2.5D to 2D : A camera and its FOV are shown from multiple perspectives (left and middle), and its corresponding mapping to 2D (right). For the 2.5D configuration, the planes displayed bound the space that can be occupied by the target.

Problem 1. (2.5D Case): *Given the camera and environment models in 2.5D , our goal is to develop a representation that has the same homotopy type as the* **detectable set for a camera network** *(i.e., the union of the sets in which a target is detectable by a camera). The construction of this representation should not rely on camera or object localization.*

The formulation of the problem is very generic. We are choosing a simplicial representation because we are after a combinatorial representation that does not contain metric information. We are also after a distributed solution, i.e. processing information at local nodes.

3.2.2 Mapping from 2.5D to 2D

The structure of the detectable set for a camera network becomes clear through an identification of our 2.5D problem to a 2D problem. Since the target is constrained to move along the floor plane, it is possible to map our problem to a 2D problem. In particular:

- Cameras located at locations (x, y, z) are mapped to location (x, y) in the plane.
- Objects in our 2.5D domain are mapped to objects with piecewise linear boundaries in the plane.
- We can also do a simple identification between the FOV of a camera to a domain $\mathcal{D}_\alpha$ of a camera in 2D . A point (x, y) in the plane is in $\mathcal{D}_\alpha$ if the target located at that point intersects the FOV $\mathcal{F}_\alpha$. The set $\mathcal{D}_\alpha$ is the orthogonal projection (onto the xy-plane) of the intersection between $\mathcal{F}_\alpha$ and the space between $z \geq 0$ and $z \leq h_{target}$. Since the latter is an intersection of convex sets, and orthogonal projections preserve convexity, then $\mathcal{D}_\alpha$ is convex. We can also check that $\mathcal{D}_\alpha$ will be open.
- Also, we can give a 2D description of the coverage of a camera. A point (x, y) is in the coverage $\mathcal{C}_\alpha$ of camera α if the target located at (x, y) is detectable by the camera.

3.2.3 The Problem in 2D

We now proceed by characterizing our problem after mapping the original configuration from a 2.5D space to 2D . The following definitions are presented to formalize our discussion.

The Environment: The space under consideration is a 2D layout where cameras are located in the plane, and only sets with **piecewise-linear** boundaries are allowed (including object and paths). We assume a finite number of objects in our environment.

Cameras: A camera object α is specified by: its **position** o_α in the plane; and an open convex domain $\mathcal{D}_\alpha$, referred to as the **camera domain**.

The camera domain $\mathcal{D}_\alpha$ can be interpreted as the set of points visible from camera

α when no objects occluding the field of view are present. The convexity of this set will be essential for some of the proofs. Some examples of camera domains are shown in figure 3.3.

Definition 10. *The subset of the plane occupied by the* i**-th object**, *which is denoted by* $\mathcal{O}_i$, *is a closed connected subset of the plane with non-empty interior and piecewise linear boundary. The collection* $\{\mathcal{O}_i\}_{i=1}^{N_o}$, *where* $N_o < \infty$ *is the number of objects in the environment, will be referred to as the* **objects** *in the environment.*

Definition 11. *Given a camera* α, *a point* $p \in \mathbb{R}^2$ *is said to be* **visible from camera** α *if* $p \in \mathcal{D}_\alpha$ *and* $\overline{o_\alpha\, p} \cap \left(\bigcup_{i=1}^{N_o} \mathcal{O}_i\right) = \emptyset$, *where* $\overline{o_\alpha\, p}$ *is the line segment between the camera location* o_α *and* p. *The set of visible points is called the* **coverage** $\mathcal{C}_\alpha$ *of camera* α.

We consider the following problem:

Problem 2. (2D Case): *Given the camera and environment models in 2D , our goal is to develop a simplicial representation that has the same homotopy type as the* **coverage of the camera network** *(i.e., the union of the coverage of the cameras). The construction of this representation should not rely on camera or object localization.*

Observation 1. *Note that the camera network coverage has the same homology (i.e. topological information) as the domain* $(\mathbb{R}^2 - \bigcup O_i)$ *if these two sets are homotopic (i.e., we can continuously deform one into the other).*

3.3 The CN-Complex

The goal of this section is the construction of a simplicial complex that will capture the homology of the union of camera coverages $\bigcup \mathcal{C}_\alpha$. One possible approach for accom-

plishing this task is to obtain the nerve complex (see chapter 2) using the set of camera coverage $\{\mathcal{C}_\alpha\}$. However, this approach will only work for simple configurations without objects in the domain. An example illustrating our claim is shown in figure 3.2.

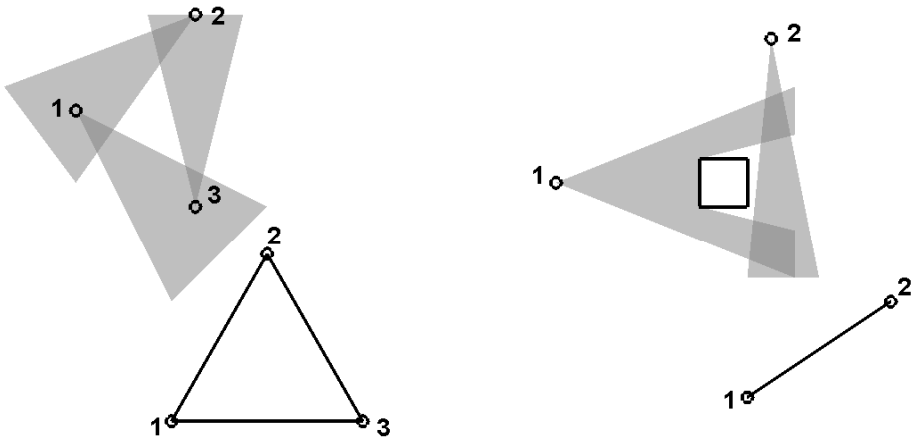

Figure 3.2: Nerve complexes obtained from the collection $\{\mathcal{C}_\alpha\}$. One complex captures the correct topological information (left) but the other does not (right).

The reason figure 3.2 (right) does not capture the topological structure of the union of camera coverage is because the hypothesis of the Čech Theorem (see chapter 2) is not satisfied (in particular, $\mathcal{C}_1 \cap \mathcal{C}_2$ is not contractible). From the physical layout of the cameras and the objects in the environment, it is clear how we can divide $\mathcal{C}_1$ in order to obtain contractible intersections. We are after a decomposition of the coverage that can be achieved without knowing the exact location of objects in the environment.

3.3.1 The Decomposition Theorem

Before proceeding let us consider the following useful definitions:

Definition 12. *Given the objects* $\{\mathcal{O}_i\}_{i=1}^{N_o}$*, a piecewise linear path* $\Gamma : [0,1] \to \mathbb{R}^2$ *is said to*

be **feasible** *if* $\Gamma([0,1]) \cap (\bigcup \mathcal{O}_i) = \emptyset$.

Definition 13. *Given camera* α *with camera domain* $\mathcal{D}_\alpha$ *and corresponding boundary* $\partial\mathcal{D}_\alpha$*, a line* L_α *is a* **bisecting line** *for the camera if:*

- L_α *goes through the camera location* o_α.
- *There exists a feasible path* $\Gamma : [0,1] \to \mathbb{R}^2$ *such that for any* $\epsilon > 0$ *there exists a* δ *such that* $0 < \delta < \epsilon$, $\Gamma(0.5-\delta) \in \mathcal{C}_n$, $\Gamma(0.5+\delta) \notin \mathcal{C}_\alpha$, $\Gamma(0.5) \in L_\alpha$, *and* $\Gamma(0.5) \notin \partial\mathcal{D}_\alpha$.

If we imagine a target traveling through the path Γ, the last condition in the definition of a bisecting line identifies when an **occlusion event** is detected (i.e., the target transitions from visible to not visible, or vice versa). However, occlusion events due to the target leaving through the boundary of the camera domain $\mathcal{D}_\alpha$ are ignored.

Definition 14. *Let* $\{L_{\alpha,i}\}_{i=1}^{N_L}$ *be a finite collection of bisecting lines for camera* α*. Consider the set of adjacent cones in the plane* $\{K_{\alpha,j}\}_{j=1}^{N_C}$ *bounded by these lines, where* $N_C = 2 \cdot N_L$*, then the* **decomposition of** $\mathcal{C}_\alpha$ **by lines** $\{L_{\alpha,i}\}$ *is the collection of sets*

$$\mathcal{C}_{\alpha,j} := K_{\alpha,j} \cap \mathcal{C}_\alpha.$$

Note that the decomposition of $\mathcal{C}_\alpha$ is not a **partition** since the sets $\mathcal{C}_{\alpha,j}$ are not necessarily disjoint.

The construction of the **camera network complex** (CN-Complex) is based on the identification of bisecting lines for the coverage of each individual camera. This construct will capture the correct topological structure of the union of coverage of the network.

Figure 3.4 displays examples of CN-Complexes obtained after decomposing the coverage of each camera using their corresponding bisecting lines. The CN-Complex cap-

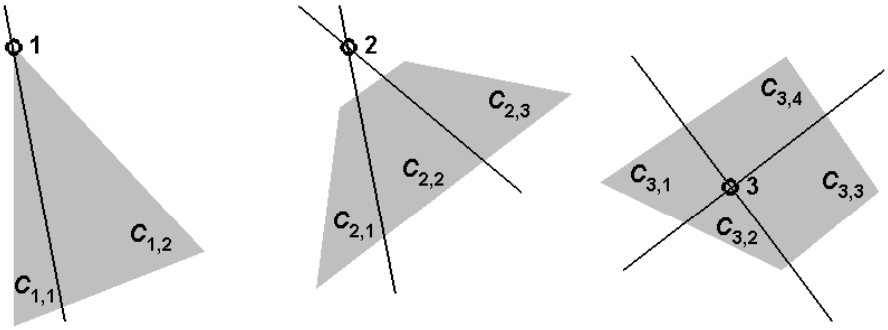

Figure 3.3: Three examples of camera domains $\mathcal{D}_\alpha$. Cameras can be inside or outside their domains. Our camera model spans projection models from perspective cameras to omni-directional cameras. Decompositions are shown for each set.

tures the correct topological information, given that the assumptions made for the model described in section 3.2 are satisfied. The following theorem (see appendix A for proof), states this fact.

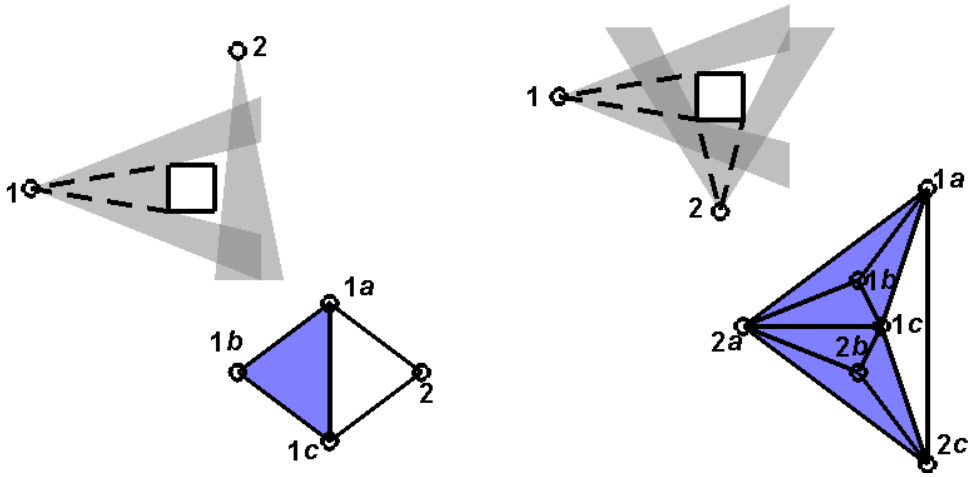

Figure 3.4: Examples of CN-Complexes. In both cases, camera 1 is decomposed into three regions, each of which becomes a vertex in the complex.

Theorem 2. (Decomposition Theorem)

Let $\{\mathcal{C}_\alpha\}_{\alpha=1}^{N}$ be a collection of camera coverage where each $\mathcal{C}_\alpha$ is connected and N is the number of cameras in the domain. Let $\{\mathcal{C}_{\alpha,k}\}_{(\alpha,k)\in A_D}$ be the collection of decomposed

sets by all possible bisecting lines, where A_D is the set of indices in the decomposition. Then, any finite intersection $\bigcap_{(\alpha',k')\in A} \mathcal{C}_{\alpha',k'}$, where $A \subset A_D$, is contractible.

By the previous theorem, the hypothesis of the Čech Theorem is satisfied if we have connected coverage which are decomposed by all of their bisecting lines. This implies that computing the homology of the CN-Complex returns the appropriate topological information about the network coverage as a whole.

Observation 2. *Note that there are many ways to decompose a set in order to obtain subsets with contractible intersections. However, by using the bisecting lines, we ensure that the decomposition can be done locally (at each camera node) without knowledge of the physical structure of the environment.*

We note that the steps required to build the CN-Complex are two-fold:

1. Identify all bisecting lines and decompose each camera coverage.

2. Determine which of the resulting sets intersect.

The first step makes sure that any intersection will be contractible. The second step allows us to find the simplices for our representation. These two steps can be completed in different ways which depend on the scenario under consideration. In sections 3.4 and 3.5, the construction of the CN-Complex for a scenario with a single target is demonstrated.

3.3.2 From 2D to 2.5D

The CN-Complex can be built by decomposing each camera coverage using its bisecting lines and determining which of the resulting sets intersect. However, a physical

camera only has access to observations available in its image domain Ω^{3D}. Therefore, it is essential to determine how to find bisecting lines using information in the image domain.

We note that occlusion events occur when the target leaves the coverage $\mathcal{C}_\alpha$ of camera α along the boundary of the camera domain $\mathcal{D}_\alpha$ or along a bisecting line. We can verify that a target leaving through the boundary of $\mathcal{D}_\alpha$ will be detected in the image domain Ω_α^{3D} as having the target disappearing/appearing through the boundary of Ω_α^{3D}. If the target leaves $\mathcal{C}_\alpha$ through one of the bisecting lines, an occlusion event in the interior of Ω_α^{3D} will be observed. Note that bisecting lines in the 2D domain correspond to vertical planes in the 2.5D configuration, whose intersection with the FOV of the camera map to lines in Ω_α^{3D}. Hence, all that is required is to find the line segment in which an occlusion event takes place in the image domain. From an engineering point of view, this can be done by performing some simple image processing to find the edge along which target disappears/appears in an image. The result will be a decomposition of the image domain Ω_α^{3D} which will correspond to a decomposition of the camera coverage $\mathcal{C}_\alpha$. These computations can be done locally at a camera node without any need to transmit information.

The problem of finding intersections of the sets for the 2D problem corresponds to having concurrent detections at corresponding cameras for the case of a single target in the environment. Finding overlap between these regions can be solved for the multiple-target case by using approaches such as the ones outlined in [31, 32, 57, 53, 11] in which correspondence and time correlation are exploited.

3.4 Simulations in 2D for Single Target

In this section, a 2D scenario is simulated in which a wireless camera network is deployed and no localization information is available. Camera nodes will be assumed to have certain computational capabilities and they can communicate wirelessly with each other.

The assumptions for this particular simulation are:

The Environment in Simulation: The objects in the environment will have piecewise linear boundaries as described earlier. The location of the objects will be unknown. The location and orientation of the cameras is also unknown.

Cameras in Simulation: A camera α has the following properties:

- The domain $\mathcal{D}_\alpha$ of a camera in 2D will be the interior of a convex cone with field of view $\theta_\alpha < 180^o$. This model is used for simplicity in the simulations.
- A local camera frame Ψ_α^{2D} is chosen such that the range of the field of view is $[-\theta_\alpha/2, \theta_\alpha/2]$ when measured from the y-axis.
- Its **camera projection** $\Pi_\alpha^{2D} : \mathcal{D}_\alpha \to \mathbb{R}$, is given by

$$\Pi_\alpha^{2D}(p) = p_x/p_y,$$

  where p is given in coordinate frame Ψ_α^{2D}. The image of this mapping, i.e. $\Pi_\alpha^{2D}(\mathcal{D}_\alpha)$, will be called the **image domain** Ω_α^{2D}.

The Target in Simulation: A single point target is considered in order to focus on the construction of the complex without worrying about correspondence/identification of our target.

Throughout our simulations the target will move continuously through the environment. At each time step the cameras compute their detections of the target and use their observations to detect bisecting lines. Observations at the regions obtained after decomposition using the bisecting lines are stored. These observations are then combined to determine intersections between the regions which become simplices in the CN-Complex.

As mentioned before, the topology of the environment can be characterized in terms of its homology. In particular we will use Betti numbers β_0 and β_1 (see section 2.1). The β_0 number tells us the number of connected components in the coverage while β_1 gives the number of holes. The PLEX software package [1] is used for homology computations and corresponding Betti numbers.

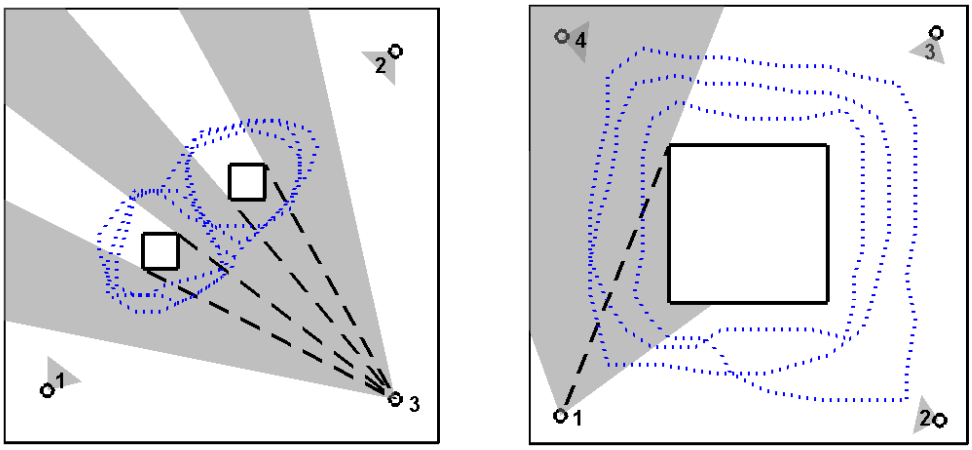

Figure 3.5: A layout with two objects where $\mathcal{C}_3$ is shown (left). A circular hallway configuration where $\mathcal{C}_1$ is shown (right). Dashed lines represent corresponding bisecting lines. Dotted curves represent the paths followed by the target during the simulation.

Figure 3.5 (left) shows a three-camera layout with two objects in their field of view. In this case, we observe three bisecting lines for camera 1, two for camera 2, and four for camera 3. Note that cameras 1 and 2 have different number of bisecting lines since

there are not placed symmetrically in the diagram. The coverage $\mathcal{C}_3$ is decomposed into 5 regions, namely $\{\mathcal{C}_{3,a}, \mathcal{C}_{3,b}, \mathcal{C}_{3,c}, \mathcal{C}_{3,d}$ and $\mathcal{C}_{3,e}\}$. The list of maximal simplices obtained by our algorithm is: $[1a\ 1b\ 1c\ 1d]$, $[2a\ 2b\ 2c]$, $[3a\ 3b\ 3c\ 3d\ 3e]$, $[1a\ 1b\ 2c\ 3c]$, $[1d\ 2a\ 3c]$, $[2a\ 2b\ 3a]$, $[1a\ 2b\ 2c\ 3a]$, $[1a\ 2c\ 3a\ 3b]$, $[1a\ 2c\ 3b\ 3c]$, $[1a\ 2c\ 3c\ 3d]$, $[1a\ 1b\ 1c\ 2c\ 3d\ 3e]$, $[1c\ 1d\ 3e]$ and $[1d\ 2a\ 3e]$. The homology computations returned Betti numbers: $\beta_0 = 1$ and $\beta_1 = 2$. This agrees with having a single connected component for the network coverage and two objects inside the coverage of the cameras.

In figure 3.5 (right) we observe similar results for a configuration that can be interpreted as a hallway in a building floor. There is a single bisecting line for all cameras. The algebraic analysis returns $\beta_0 = 1$ and $\beta_1 = 1$. The latter identifies a single hole corresponding to the loop formed by the hallway structure. The list of maximal simplices recovered by the algorithm is: $[3b\ 4a\ 4b]$, $[2b\ 3a\ 3b]$, $[1b\ 2a\ 2b]$, $[1a\ 1b\ 4b]$.

3.5 Experimentation

In order to demonstrate how the mathematical tools described in the previous sections can be applied to a real wireless sensor network, an experiment tracking a robot in a simple maze is presented. Figure 3.6 shows the layout to be used. A sensor network consisting of CITRIC camera motes [13] is setup in the maze. The CN-Complex is constructed for this particular configuration and used for tracking in this representation. Homology computations are performed using the PLEX software package [1].

Time synchronization is required in order to determine overlaps between the different camera regions. The Flooding Time Synchronization Protocol (FTSP) [33] was used

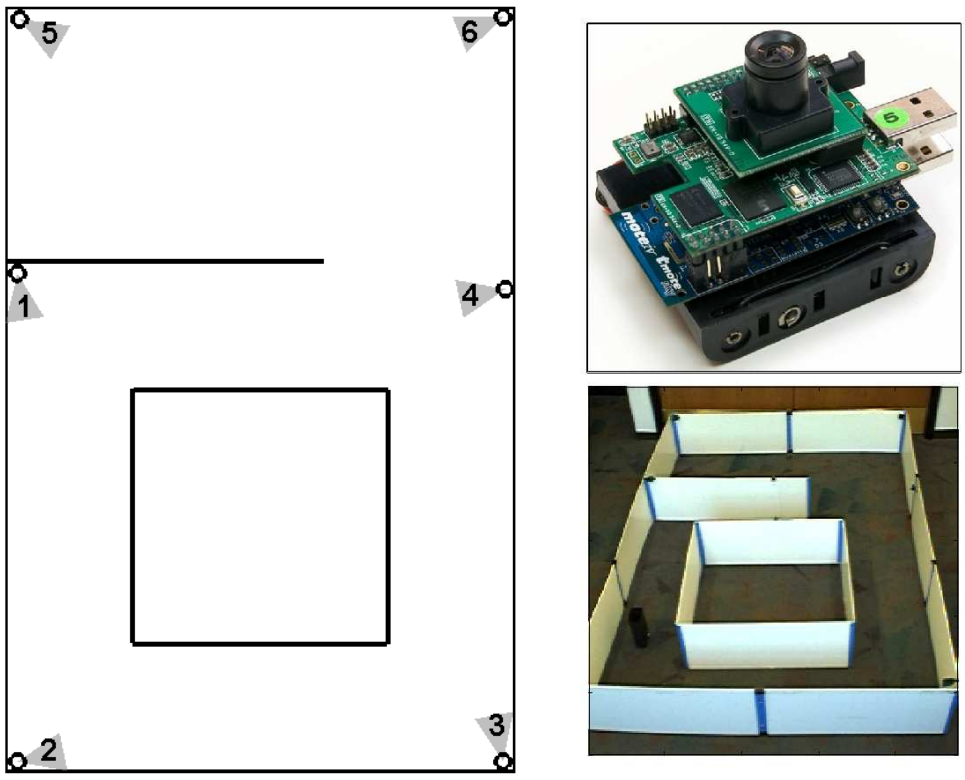

Figure 3.6: Layout used for our experiment. A diagram showing the location of the different cameras (left). A picture of our experimental maze (bottom-right). The CITRIC camera motes used for our experiments (top-right).

for this purpose.

At each camera node, background subtraction is performed at each frame. Once a target is detected, further processing to detect bisecting lines (as shown in figure 3.7) is performed. However, note that the bisecting line processing occurs sparsely and hence power consumption is mostly due to background subtraction. Statistics on the power consumption for the CITRIC platform can be found in [13]. Note that the information extracted from each camera node is just a decomposition of the image domain with a list of times at which detections were made.

For the experiments the camera motes were capable of processing grayscale images

at 4 frames per second at a resolution of 320×240 pixels. Symbolic information was then extracted and transmitted at a rate of 1 packet of 100 bytes every 10 seconds. Transmissions were performed regularly even when there were no observations to transmit. If raw image data (without any compression) was to be streamed over the network, this would correspond to about 300 $kBytes/s$ of data from a single mote. Instead, transmitting symbolic information in our experiment only accounts for 10 $Bytes/s$.

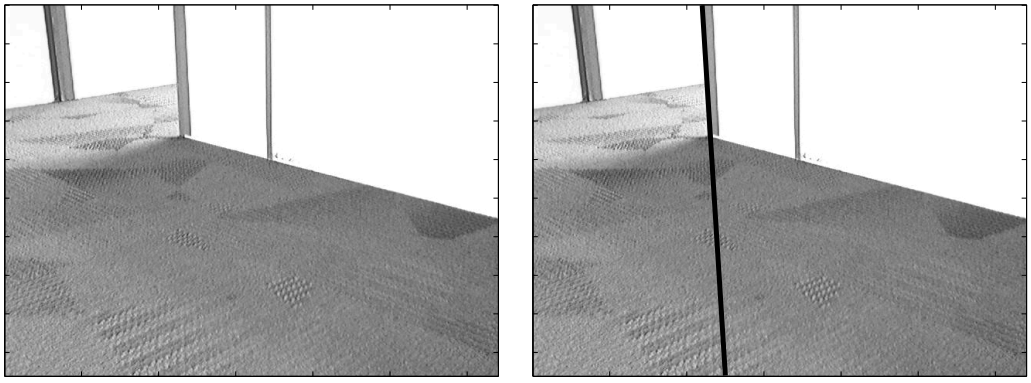

Figure 3.7: View of camera 5 from the layout in figure 3.6 before (left) and after (right) a bisecting line is found.

The complex is built by combining all local information from the camera motes. Each camera mote transmits the history of its detections wirelessly to a central computer that creates the CN-Complex. The resulting complex contains the maximal simplices: [1a 1b 4b], [1b 2a 2b], [2b 3a 3b], [3b 4a 4b 5b], [3b 5b 6], and [5a 5b 6]. A pictorial representation of the complex is shown in figure 3.8 (right plots).

As mentioned earlier, this representation can then be used for tracking and navigation without actual metric reconstruction of the environment. Figure 3.8 shows a set of recorded paths for the robot. By determining which simplices are visited by the robot's path we can extract a path in the complex as shown by the dashed path in the complexes

of figure 3.8. The main advantage of this representation is that the path in the complex gives a global view of the trajectory of the robot, while local information can be extracted from single camera views.

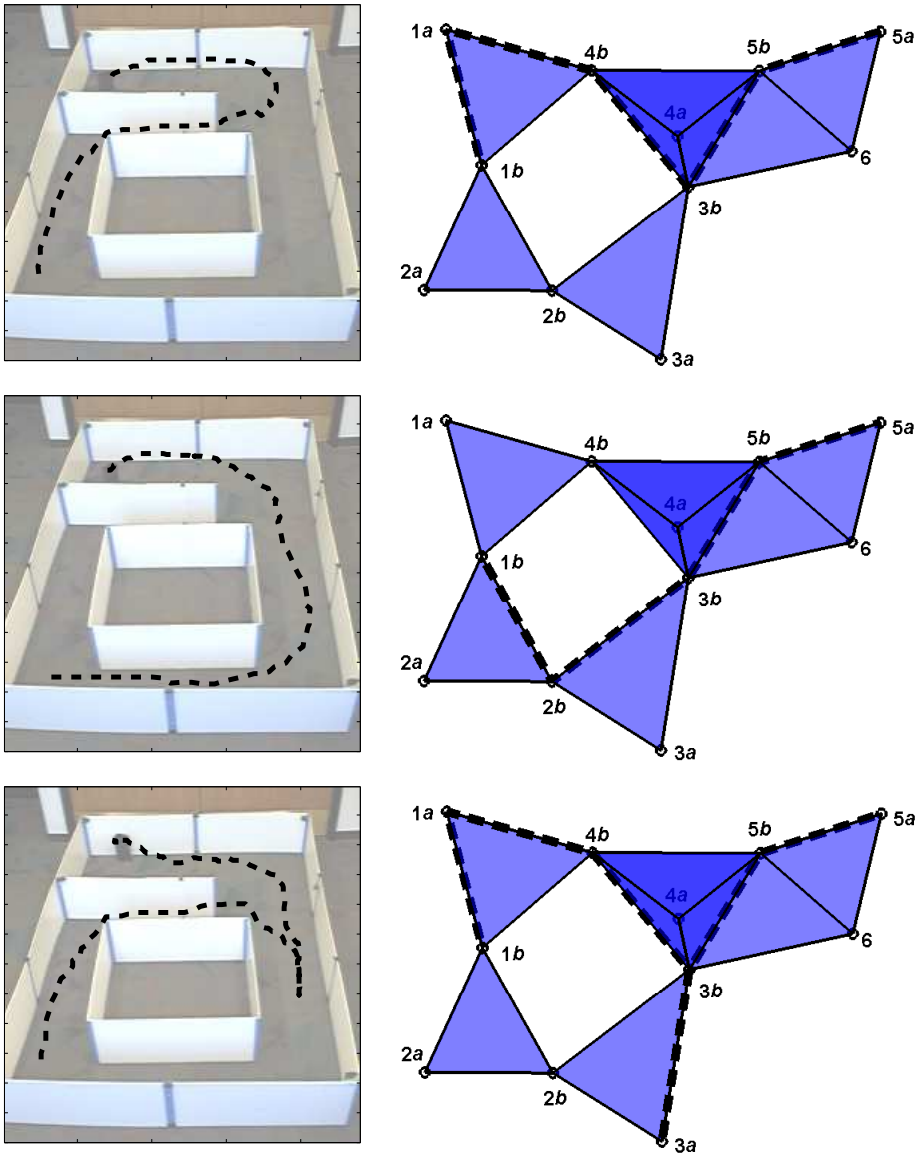

Figure 3.8: Paths traveled by the robot in the maze (shown in dashed lines): In the physical layout (left), and in the CN-Complex (right). These paths can be compared by using the algebraic topological tools covered in chapter 2.

It is possible to identify paths in the simplicial representation that are homotopic (i.e., that can be continuously deformed into one another). The tools required for these computations are already available to us from chapter 2. In particular, by taking two paths that start and end at the same locations forming a loop, we can verify that they are homotopic if they form the boundary of some combination of simplices. Equivalently, since a closed loop σ is just a collection of edges in C_1, we need to check whether the loop σ is in B_1 (i.e., in the range of ∂_2). This is just a simple algebraic computation. By putting the top and middle paths from figure 3.8 together we note that the resulting loop is not in the range of ∂_2 (i.e., they are not homotopic). On the other hand, the top and bottom paths can be easily checked to be homotopic.

3.6 Extensions

In this section extensions of the CN-Complex to several scenarios are discussed. First, more general 3D configurations are considered and the limitations of this approach are explored. Finally, extensions to mobile agents and the role of occlusion detection in these scenarios are discussed.

3.6.1 Challenges in 3D

Let us discuss a scenario for which the construction of the CN-Complex would fail to capture topological information about the environment. Consider the case in which walls are short, i.e. the target is taller than the walls. A very basic example is illustrated in figure 3.9. For this configuration, we would like to recover the fact that the set of feasible

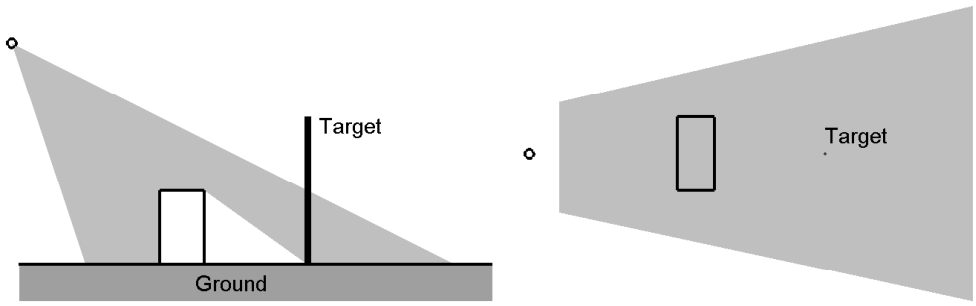

Figure 3.9: 3D layout in which a hole in the set of feasible locations for the target is not captured by the CN-Complex: A side view (left) and top view (right) of the configuration showing a target in the scene. Note that it is not possible to detect bisecting lines in this configuration.

locations for the target contains a hole due to the object present in the scene. However, no bisecting line for the configuration can be found if we consider a vertical target moving in the environment. On the other hand, even if bisecting lines were found along the boundary of the object, enough connections between the regions would be discovered to identify the coverage as simply connected. Note that even after partitioning the image domain as much as possible we still run into this problem. Nevertheless, one may argue that if the target is taller than the obstacle then it can probably get over it. However, there are other situation when considering hills (not just a flat ground) in which such a naive argument will not be valid.

3.6.2 Extensions to Mobile Agents

A dual formulation of the 2D problem under consideration is the 2D scenario in which all observations are made from a mobile agent with omnidirectional vision, the cameras are replaced by oriented and uniquely identifiable markers, and the agent is capable of detecting markers and their orientation with respect to its reference frame. Every time a

marker disappears / appears from the field of view of the agent, the angle of the location of the agent with respect to the reference frame of the marker is recorded. This angle plays the same role as the bisecting lines in the CN-Complex. Overlap between the resulting regions is found by concurrent detections of markers. Hence, it is possible to build a simplicial complex as before.

A natural restriction of the setup for the CN-Complex is the assumption of static cameras. However, it is also possible to exploit occlusion information in a scenario where cameras move in the plane under known trajectories and occlusions are detected as they move around the environment. By assuming that a mobile agent is capable of detecting occlusions between different objects in the environment, the agent can then move through the layout finding bisecting lines which are used to decompose the 2D environment. Connectivity and possible occupancy between the resulting regions can then be validated to obtain an actual map of the layout.

3.7 Discussion

In this chapter, an algebraic representation of a camera network coverage is obtained through the use of discrete observations from each camera node. The mathematical tools used for this purpose are those of algebraic topology. In particular, it is shown that given enough observations the model does capture the correct topological information.

The experiment using wireless camera motes illustrates how the representation can be used to track and compare paths in a wireless camera network without any metric information. For coordinate-free navigation, the representation can give an overall view of how

to arrive at a specific location, and the transitions between simplices can be accomplished in the physical space by local visual feedback from single camera views. Using this proposed model allows for local processing at each node and minimal wireless communication. A list of times at which occlusion events were observed is all that needs to be transmitted. Also, all algebraic computations can be performed using integer operations as described in [25], which opens the doors to implementation on platforms with low-computational power. The homology computations in the experiment are done in a centralized fashion, however, distributed algorithms such as the ones introduced by A. Muhammad and A. Jadbabaie [35] can be used.

The next chapter will show how to obtain the CN-Complex robustly. Chapter 5 will discuss tracking, navigation, and path identification applications using this representation. Finally, chapter 6 will demonstrate how to generalize this representation to incorporate more information about the environment extracting relative position between cameras in the form of relations.

Chapter 4

Robust Complex Building

The CN-Complex was proven to have the same homotopy type as the coverage of a camera network in a 2.5D environment in chapter 3, simulations and a simple experiment were shown using a single target and perfect foreground detection assumptions. However, no algorithm was presented for handling multiple targets and noisy observations. This chapter will present an approach to building the CN-Complex in a robust way to be able to handle multiple targets and detection errors. This approach will only utilize temporal correlation between observations, i.e. neither actual matching is performed nor an appearance model is constructed in order to extract connectivity information between cameras. A distributed version of the algorithm will be outlined for which data is processed and stored in a distributed fashion. The end result will be a collection of simplices with assigned probabilities of occurrence. It is possible to choose a threshold in order to select the most likely simplices; however, diagrams depicting the persistence of topological features over all possible threshold values will be analyzed instead.

An example of the construction of the CN-Complex is illustrated in figure 4.1 in which three overlapping views are considered (left plot). The steps of the construction of the complex involve first bisecting the fields of view (middle plot) and then finding all possible overlap (right plot). In this example, the simplicial complex contains a single hole which corresponds to the column that acts as an occluding object in the physical coverage.

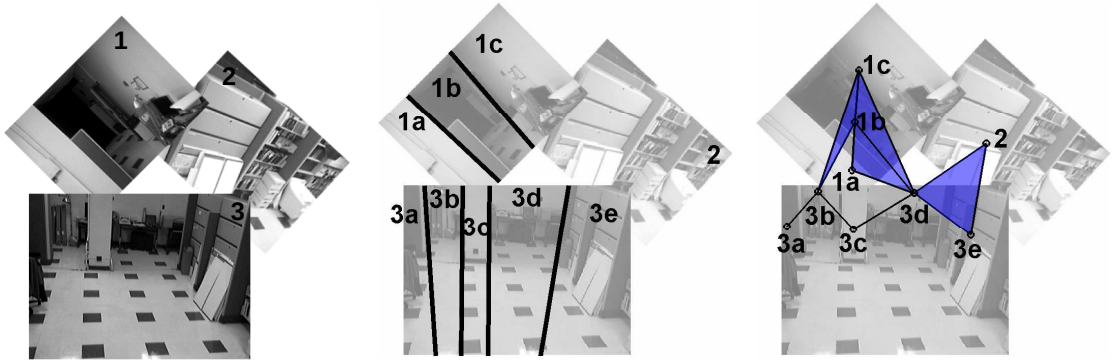

Figure 4.1: The CN-Complex for a network of three cameras constructed using the methodology presented in this chapter. The views from different cameras (left). Bisected views due to occluding objects (middle). The simplicial complex built by finding the overlap in the coverage of the cameras (right). 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.

The rest of the discussion is as follows: Section 4.1 shows how to find bisecting lines in a robust way; section 4.2 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 4.3 through a real life experiment with multiple targets.

4.1 Finding Bisecting Lines

In this section, the problem of detecting the bisecting lines that decompose the image domain of a camera is addressed. To this end, a simple background subtraction algorithm (e.g. thresholding with respect to a background image) is assumed. We then utilize the algorithm presented by B. Jackson et al. [24], which consists of accumulating the boundary of foreground objects wherever partial occlusions are detected. In our case, only the detections at times when full occlusion events occur are stored. 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 4.2 (top-left), a camera view with several occluding boundaries due to walls and a column is observed. The accumulated boundaries of the foreground detections are shown on the top-right plot. Initial estimates for the bisecting lines 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.

4.2 Finding Intersect Points

In this section, it is assumed that the bisections within each camera view have been calculated. The emphasis is only on determining the connectivity between camera pairs. More specifically, we look for **intersect points** (i.e. points in the intersection of the

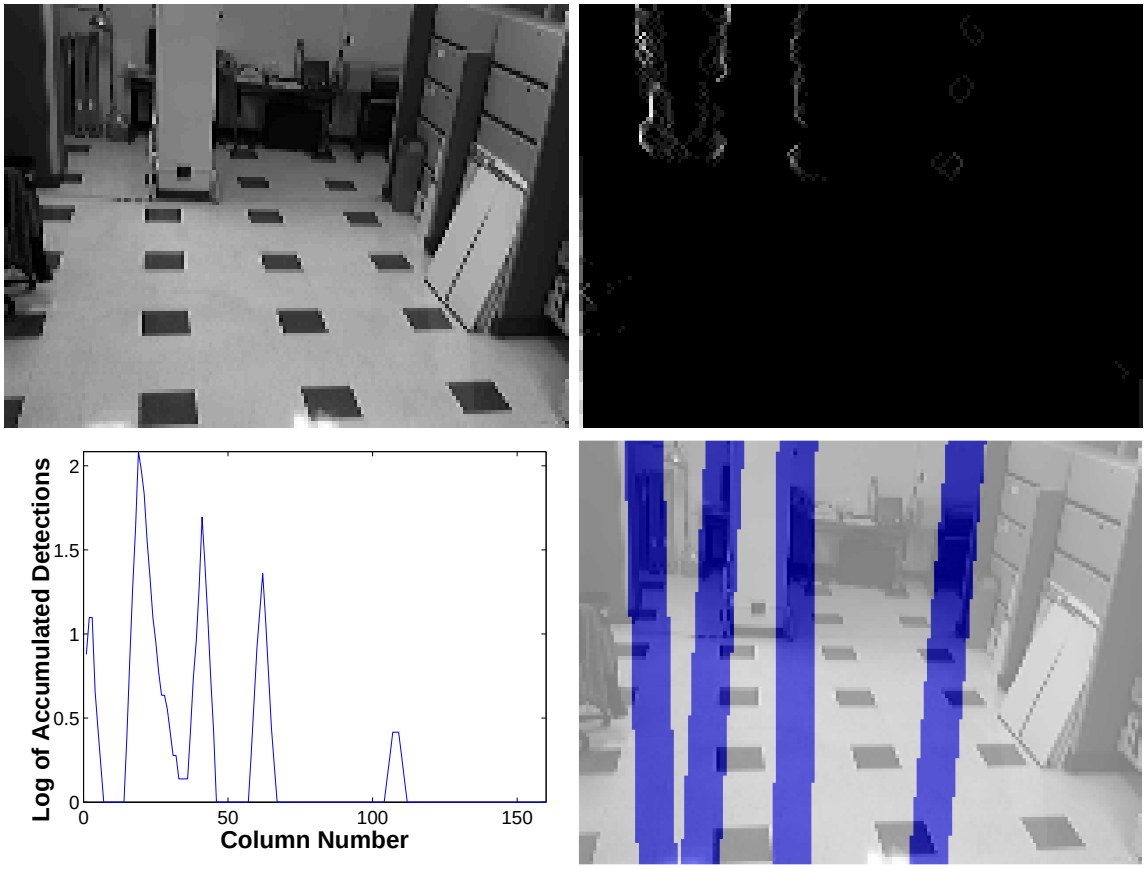

Figure 4.2: Steps for finding bisecting lines: For the original view (top-left), the boundaries of the foreground masks are accumulated whenever occlusion events are detected (top-right). Vertical bisecting lines are estimated by aggregating observations over all rows and obtaining the indices of the 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).

field of views of the cameras).

In the following discussion, it is also assumed for simplicity that each sensor will be able to uniquely identify any point in its coverage. In other words, a homeomorphism between the image domain from each camera and its coverage exists. No target identification will be necessary, but localization and recurrence over time will be exploited.

The approach is illustrated by first considering the example in figure 4.3. Assume 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, and 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, assume that a target in R_i is detected if and only if it is actually present (i.e. there are no errors in the detections). Hence,

$$P(D_1^t) = 1 - P(\overline{D_1^t}) = 1 - |R_1^c|^N, \tag{4.1}$$

and

$$\begin{aligned} P(D_1^t \wedge D_2^t) &= 1 - P(\overline{D_1^t \wedge D_2^t}) \\ &= 1 - P(\overline{D_1^t} \vee \overline{D_2^t}) \\ &= 1 - \left(P(\overline{D_1^t}) + P(\overline{D_2^t}) - P(\overline{D_1^t} \wedge \overline{D_2^t})\right) \\ &= 1 - |R_1^c|^N - |R_2^c|^N + |(R_1 \cup R_2)^c|^N \end{aligned} \tag{4.2}$$

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

$$\begin{aligned} P(D_2^t|D_1^t) &= \frac{P(D_1^t \wedge D_2^t)}{P(D_1^t)} \\ &= 1 - \left(\frac{|R_2^c|^N - |(R_1 \cup R_2)^c|^N}{1 - |R_1^c|^N}\right). \end{aligned} \tag{4.3}$$

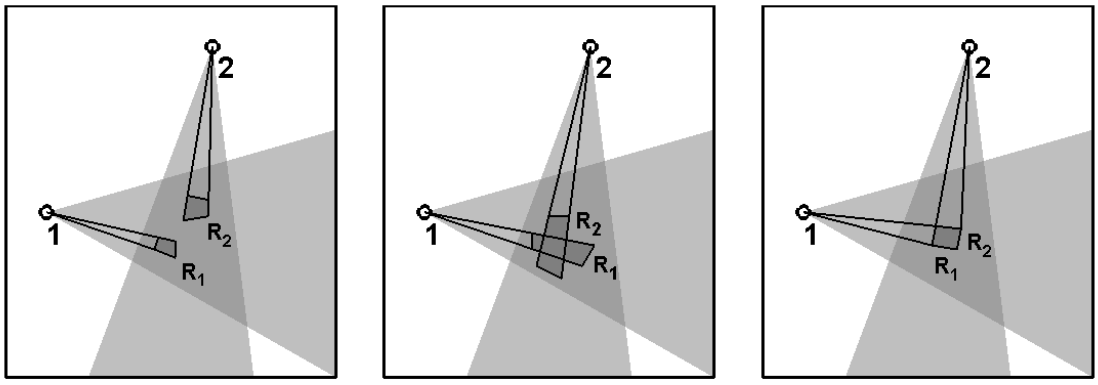

Figure 4.3: Geometric depiction illustrating different overlapping configurations and corresponding detection probabilities for 3 targets. Intuitively, whenever R_1 and R_2 are disjoint we expect a low value of $P(D_2^t|D_1^t)$ (left). If $|R_1| \approx |R_2| \approx 0.005$ then $P(D_2^t|D_1^t) \approx 0.01$ by using equation 4.3. 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 $P(D_2^t|D_1^t) = 1$.

Intuitively, whenever R_1 and R_2 are disjoint we expect $P(D_2^t|D_1^t) << 1$. For a partial overlap, a larger probability is expected. For a perfect overlap, $P(D_2^t|D_1^t) = 1$. These observations are illustrated in figure 4.3. Note that $P(D_2^t|D_1^t)$ can be 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. $P(D_2^t \wedge D_3^t|D_1^t)$. Also, $P(D_2^t|D_1^t)$ 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 maybe impossible. Therefore, the filtration process is employed to robustly analyze the observed data in order to avoid making undue

assumptions.

4.2.1 The Algorithm

In this section, an algorithm to estimate distributedly the probabilities $P(D_2^t|D_1^t)$ and $P(D_2^t \wedge D_3^t|D_1^t)$ is described. 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 the simulations, it is shown that this subset will be sufficient to determine whether there is an overlap in coverage between cameras.

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 1, which is executed every time a new frame is captured, describes how intersect 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* 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

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
```

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 intersect 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 $IPts$.

An overlap between two cameras (i.e. a 1-simplex) can be concluded if there is an entry in $IPTS$ between these cameras with a high detection probability. Intersections between three cameras (i.e. 2-simplices) can be found similarly. Note, only 2-simplices are required for the construction of the CN-Complex since only planar information about the coverage is recovered.

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

4.2.2 Simulations for Multiple Targets

In this section, the previous algorithms are used to build the CN-Complex in a simulated environment made up of objects with piecewise linear boundaries and point targets moving around it. 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 4.4. In this simulation, two targets moving around independently are considered. 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 4.4 (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 4.4 show some intersect points found by selecting a threshold of $\tau = 0.5$.

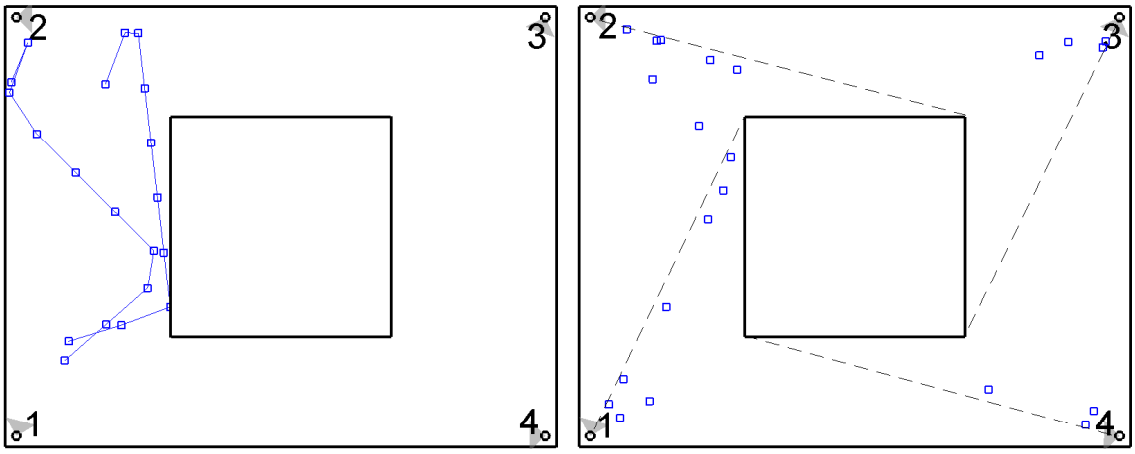

Figure 4.4: Layout of a circular corridor setup with two targets moving through the environment (left). Intersect points found, plotted as squares, for a threshold value $\tau = 0.5$ with corresponding bisecting lines, plotted as dashed lines (right).

As described in section 2.4, it is possible to analyze the topological structure of the data over all values of τ by employing the persistence of topological features. Figure 4.5 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

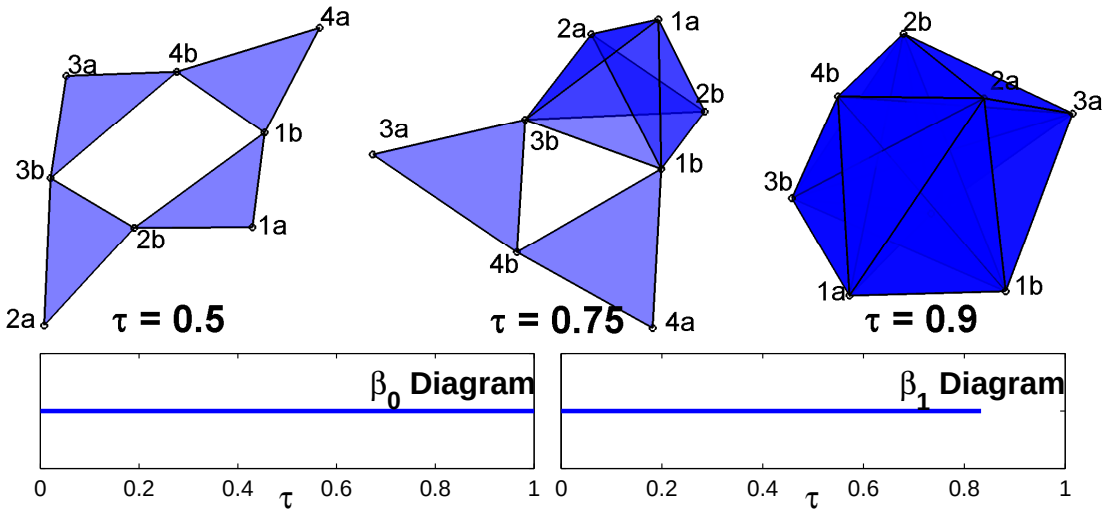

Figure 4.5: CN-Complexes for several values of τ (top) and persistence diagrams (bottom) are shown for the circular corridor setup in figure 4.4. We observe that the diagram shows a persistent single connected component and a persistent loop.

the layout. In fact, choosing a value of $\tau \in [0, 0.65]$ gives the correct simplices.

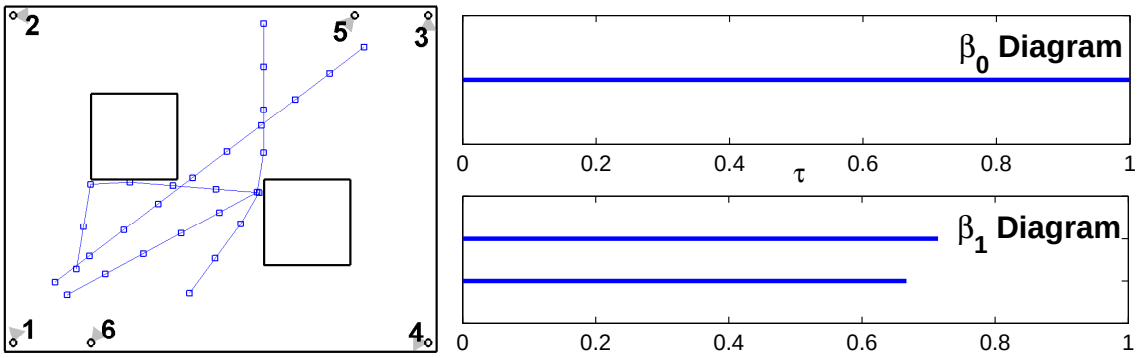

Figure 4.6: Layout of an environment with two objects and three targets (left). Corresponding persistence diagrams showing a single persistent connected component and two holes (right).

Figure 4.6 (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.

4.3 Experimentation

In this section, an experimental setup with three cameras placed in indoors is considere. Three computers that are synchronized using the Network Time Protocol (NTP) are utilized 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 off-line, the amount of computation and data transmission required are small enough to occur distributedly on a sensor network platform such as CITRIC [13]. 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 the experiment is shown in figure 4.7. 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 region in camera 2, and five regions in camera 3.

Intersect points and corresponding frequencies are computed as described in the previous section. However, instead of considering every possible pixel as an intersect point, the image domain is split into blocks of size 10×10 and treat these regions as possible intersect points. In the experiment, only points that have been observed more than 10

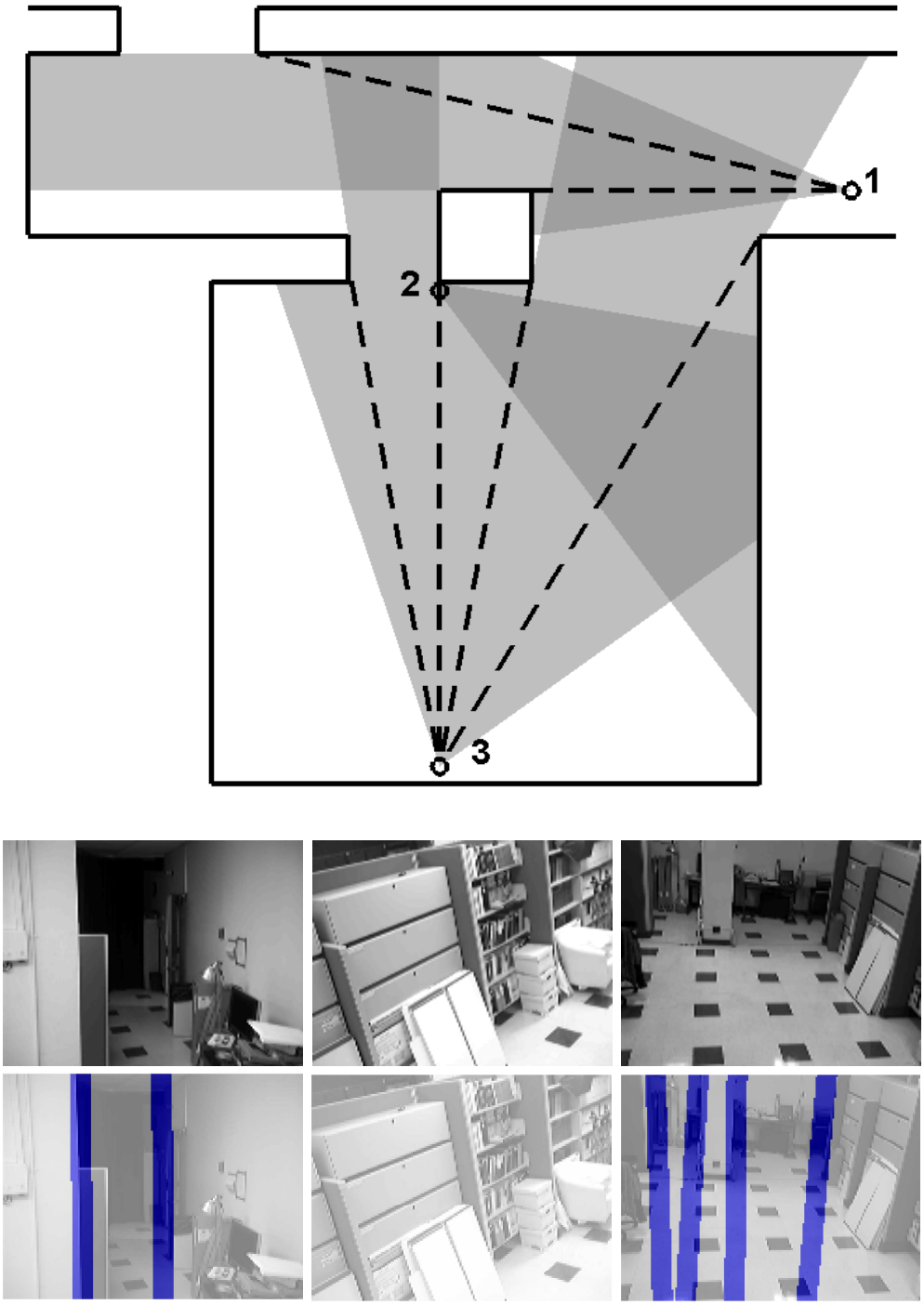

Figure 4.7: Experimental setup: Physical layout for cameras in the experiment (top). Views for cameras 1 (middle-left) through 3 (middle-right), and corresponding detected bisecting lines (bottom).

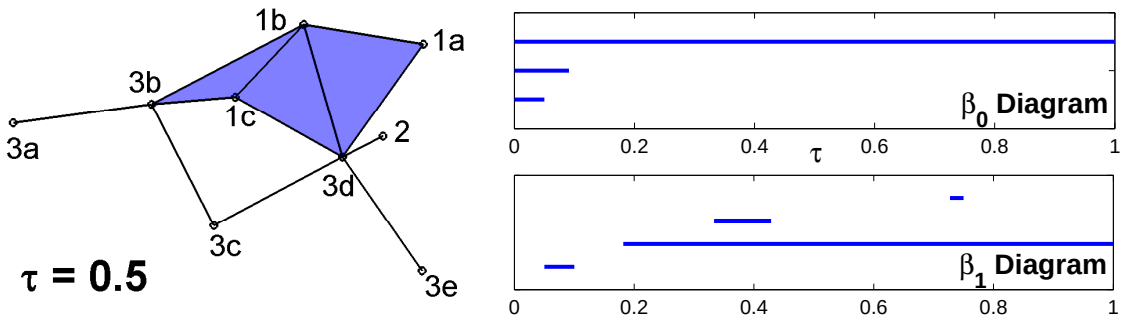

Figure 4.8: CN-Complex found for our experiment using a threshold value of $\tau = 0.5$ (left). Persistence diagrams for the filtration obtained from the experiment (right). Note, a single connected component and hole are the correct persistent features. The hole is due to the column in the middle of the room.

times are considered. Figure 4.8 (left) shows the corresponding simplex when thresholding 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.

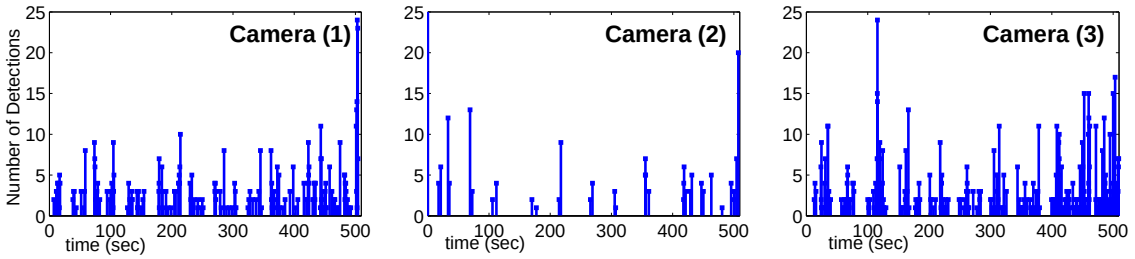

Figure 4.9: Plots of the number of block detections per occlusion event over time. Note that the events are relatively sparse (over an 8.5 minutes period), and the number of blocks detected at each time step is under 15 in most cases.

Table 4.1: 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

Since detections are only transmitted after an occlusion event, the transmission

rate is low. Figure 4.9 shows a summary of the number of blocks in which a detection was made for each camera over time. Table 4.1 shows the total number of blocks detected, number 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.

4.4 Discussion

In this chapter, a method to construct the CN-Complex for a camera network was presented. The approach presented in this chapter 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 an experiment are used to validate the approach and demonstrate its efficiency in terms of low communications costs.

Now that the construction process for the CN-Complex has been established, the following chapters will overview applications and extensions of this representation.

Chapter 5

Navigation and Path Identification

In this chapter, applications to navigation and path identification using the CN-Complex are presented. Paths in the coverage of a network will be characterized in terms of their homotopy class. The analysis will require to map trajectories from the physical space to the CN-Complex (which is trivially done by determining the visibility of the target by the cameras), and from the complex to the physical space. The latter will be accomplished through the use of intersect points (as discussed in the previous chapter).

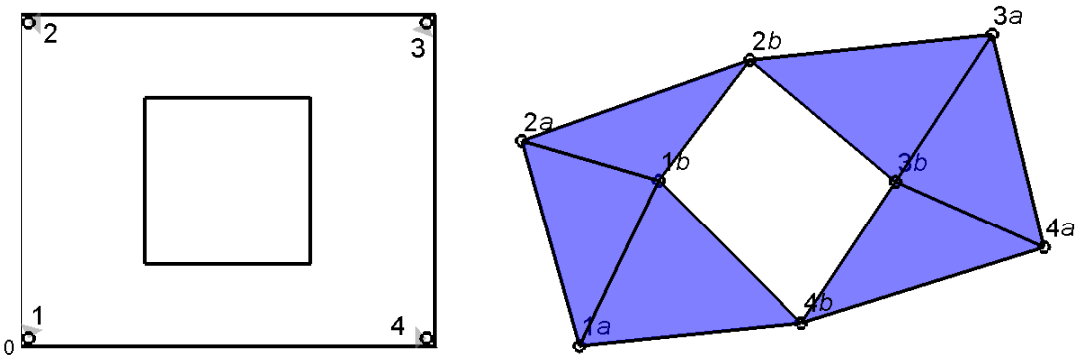

Figure 5.1: A simple circular hallway layout (left) and corresponding CN-Complex (right). Note that each camera node has been split into vertices a and b.

The layout in figure 5.1 will be used as an illustrative example throughout this chapter. Throughout this chapter, the 0-simplices $\{[1a], [1b], [2a], \cdots, [4b]\}$ will be identified to the standard basis $\{v_1, v_2, v_3, \cdots, v_8\} \subset \mathbb{R}^8$. Similarly, the 1-simplices $\{[1a\ 1b], [1a\ 2a], [1a\ 4b], [1b\ 2a], [1b\ 2b], [1b\ 4b], [2a\ 2b], [2b\ 3a], [2b\ 3b], [3a\ 3b], [3a\ 4a], [3b\ 4a], [3b\ 4b], [4a\ 4b]\}$ are identified to the standard basis $\{e_1, e_2, e_3, \cdots e_{14}\} \subset \mathbb{R}^{14}$. Finally, $\{[1a\ 1b\ 2a], [1a\ 1b\ 4b], [1b\ 2a\ 2b], [2b\ 3a\ 3b], [3a\ 3b\ 4a], [3b\ 4a\ 4b]\}$ are identified to the standard basis $\{f_1, f_2, f_3, \cdots, f_6\} \subset \mathbb{R}^6$. Then, the boundary operators can be represented as:

$$\partial_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

and

$$\partial_1 = \begin{bmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Chapter 2 offers a review of the algebraic topological concepts used here.

The rest of the discussion is as follows: section 5.1 shows how to find a path between any two vertices in a simplicial complex; section 5.2 describes how to navigate in the physical layout given a path in the complex; section 5.3 gives a way to identify homotopic paths in the complex; finally, an algorithm for recovering homotopically distinct paths is outlined in section 5.4.

5.1 Finding a Path in a Complex

In this section we discuss how to find sets of edges which form paths joining two vertices of a complex. Note that a path $\sigma \in C_1$ joining vertices p to q must satisfy

$$\partial_1 \sigma + [p] - [q] = 0.$$

Using the example above, it may be of interest to find a path going from vertices $1a$ to $4a$, in which case the equation above can be expressed as a linear system of equations

$A\,x - b = 0$, where $A = \partial_1$ (represented as a matrix), $b = [-1, 0, 0, 0, 0, 0, 1, 0]^T$, and x is the vector representation of σ. Solving for the minimal norm solution of the system gives $x = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1]^T$, which corresponds to $\sigma = [1a\ 4b] + [4b\ 4a]$. Hence, we have found a path from $1a$ to $4a$ using the algebraic information in the CN-Complex (see figure 5.3(a)).

This approach is contrasted with the graph based approach given by applying Dijkstra's algorithm to the connectivity graph obtained from the edges of the complex. Clearly, this will give a valid path (i.e. the shortest in terms of the number of edges visited), which turns out to be the same path as the one described above.

Note that neither the algebraic nor the graph based approach use higher order information contained within the constructed CN-Complex. A hybrid approach that utilizes aspects from previous approaches will lead to an algorithm that characterizes homotopic paths between two points. First, we specify how to identify homotopic paths.

5.2 Mapping from Complex to Physical Layout

The previous section gives a way of computing a path in the complex between any two vertices that are connected. However, this does not specify how to navigate through the physical environment in order to create a physical path. This is the goal of this section.

In the following, when referring to cameras we mean the virtual cameras resulting from our decomposition when construction the CN-Complex. Navigation in 2D and 2.5D scenarios are discussed next.

5.2.1 2D Navigation

Let us consider the environment assumptions made for the simulation environment described in section 3.4. That is, assume that cameras are in the 2D plane and can only provide bearing angular information of the target with respect to their local environment and the field of views are conic. Note that in this case targets can be localized through the use of two cameras.

Assume that the target at location p is visible by cameras α and β, then we can easily move the target to a final location q still in the intersection of both cameras. First, let the target move along the line of sight of α until the target's projection in β is aligned to the projection of q. Then, move the target along the line of sight of β until its projection in α is aligned to the projection of q, at this point the target has arrived to position q. If the target gets to the position of camera α before having its projection aligned to the projection of q in camera β, then move the target in the direction from α to q. If the target is about to leave the coverage of one of the cameras before it arrives to its destination, let the roles of α and β to be exchanged. This gives a general way to move in the intersection of the coverage of two cameras.

Given cameras α, β and γ, where $p \in \mathcal{C}_\alpha \cap \mathcal{C}_\beta$, $q \in \mathcal{C}_\beta \cap \mathcal{C}_\gamma$ and $\mathcal{C}_\alpha \cap \mathcal{C}_\beta \cap \mathcal{C}_\gamma \neq \emptyset$, then it is possible to move from p to q by first moving from p to a point in the intersection of all three camera coverage and then to q.

The set of transitions between simplices that are allowed by the previous algorithm can be represented by a graph in which nodes represent 1-simplices and edges represent 2-simplices. Note that navigation is only possible through regions that are covered by at least

two cameras.

5.2.2 2.5D Navigation

Let us consider the environment assumption made in section 3.2.1 (i.e., cameras in 3D where there are only walls to worry about) and assume that cameras are located high enough so a single camera is sufficient to localize a target. That is, there is a one-to-one mapping between the detection in the image plane and the location of the target. Of course, since there is no calibration information, it is not possible to absolutely determine where the target is except in the local coordinate frame of the camera.

Consider a target with current location p and final destination q. Local navigation between two points p and q in a single camera α can be accomplished by having the target move in the direction $q-p$ given in the coordinates of camera α. If the target hits an obstacle the target is to continue moving along the boundary of the obstacle in the direction that has a positive component when projected to $q-p$. It is easy to proof that by following this strategy the target will arrive to q without leaving the coverage of camera α. This is due to camera α having no bisecting lines in its coverage. Throughout the simulation it is observed that a straight linear path is sufficient in most cases.

In order to have the target travel from location $p \in \mathcal{C}_\alpha$ to $q \in \mathcal{C}_\beta$ where $\mathcal{C}_\alpha \cap \mathcal{C}_\beta \neq \emptyset$, camera α can guide the target to a point in the intersection with camera β, and then camera β can aid in the navigate to q. We can find general path from p and q in a connected component of the coverage of the network by following pairwise intersections. This corresponds to traveling through a graph which is the 1-skeleton of the CN-Complex. Representative points in the intersection of the coverage of two cameras can be found by

picking an arbitrary intersect point (see section 4.2).

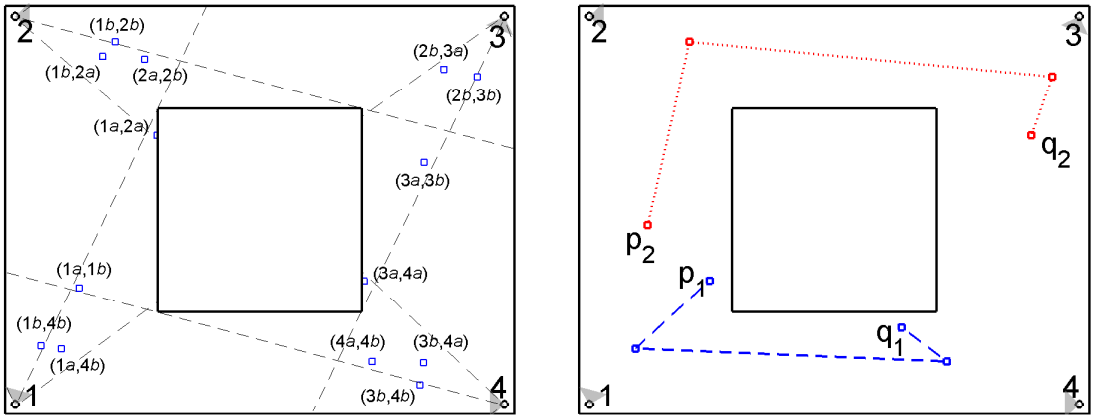

Figure 5.2: Navigation in the layout of figure 5.1: Diagram showing representative intersect points found in the layout (left) and two navigation paths obtained using the CN-Complex (right).

Since the 2.5D navigation is of most interest for real applications where the view from a single camera is sufficient for navigation, we will focus on the analysis of this approach. The following algorithm specifies how to navigate between two points in a connected coverage:

Algorithm 3 (Target Navigation in 2.5D). *Given that the target's initial location $p \in \mathcal{C}_\alpha$, and its destination $q \in \mathcal{C}_\beta$, then:*

1. *Find a path between α and β using Dijkstra's algorithm as described in section 5.1. If the path does not exist then there is no way to move from p to q without leaving the coverage of the network.*

2. *Move from p to an intersect point in the first edge of the result path by local navigation through camera α. Repeat this step as necessary until the target arrives to q.*

Figure 5.2 illustrates the outcome of the algorithm described above. The left plot shows intersect points found between camera pairs. Two navigation paths are shown in the right plot. The local navigation between intersect points is represented as straight lines. The path p_1 to q_1 corresponds to the example discussed in section 5.1.

5.3 Identifying Homotopic Paths

Let σ_1 and σ_2 be two paths joining vertices p and q which are expressed as 1-chains. Then

$$\sigma := \sigma_1 - \sigma_2$$

forms a loop. If the previous paths are homotopic, then $\sigma \in B_1 := im\,\partial_2$ (see section 2.1). Hence, we can easily verify that these paths are homotopic.

Algorithm 4. *Given the paths σ_1 and σ_2, these paths are homotopic if and only if*

$$rank(im\,\partial_2) = rank([im\,\partial_2 \mid \sigma]).$$

Let us demonstrate this process by using the paths depicted in figure 5.3. Call σ_a the path in plot (a), and so on for plots (b) and (c). Then, we can define

$$\sigma_{ab} := \sigma_a - \sigma_b = [-1, 0, 1, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, -1]^T$$

which is depicted in plot (d), and

$$\sigma_{bc} := \sigma_b - \sigma_c = [1, -1, 0, 0, 1, 0, -1, -1, 1, 0, -1, 1, 0, 0]^T$$

depicted in plot (e).

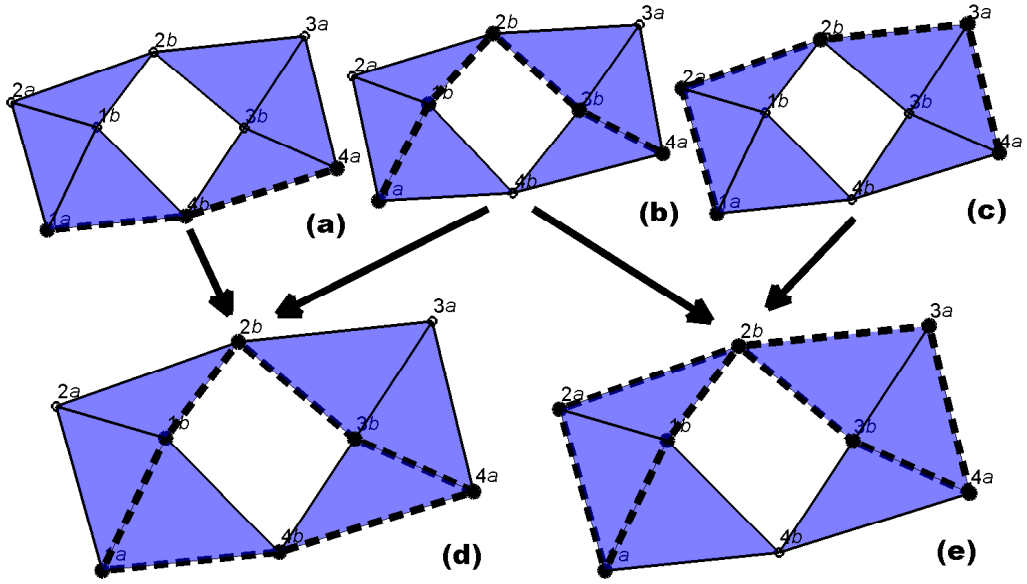

Figure 5.3: Plots (a)-(c) show several paths joining vertices $1a$ and $4a$. Plots (d) and (e) show the corresponding loops formed using these paths.

Note that $rank(im\,\partial_2) = 6$ and $rank([im\,\partial_2 \mid \sigma_{ab}]) = 7$, which leads us to conclude that σ_a and σ_b are not homotopic. This fact is clear from figure 5.3(d) which shows this loop enclosing a hole. However, $rank([im\,\partial_2 \mid \sigma_{bc}]) = 6$, which tells us that σ_b and σ_c are homotopic as we can see from figure 5.3(e).

5.4 Finding Homotopically Distinct Paths

In this section an algorithm is presented for computing homotopically distinct paths between two points. As usual, assume that all cameras have been decomposed and the CN-Complex has been built. In the algorithm each node keeps track of homotopically distinct paths that arrive to this location and broadcasts to its neighbors (as defined by the complex) all new homotopically distinct paths that arrive at a given time.

Algorithm 5. *The goal is to find homotopically distinct paths between nodes α and β. Let P_k be the set of paths found at k-th node.*

1. *We initialize all sets to be empty except for $P_\alpha = \{\alpha\}$ and broadcast this path to all neighbors.*

2. *Let σ be a path received by the k-th node. If σ does not contain k and is not homotopic to any of the paths in P_k, then k is added to the path σ and it is stored in P_k and transmitted to its neighbors (except the neighbor that it came from).*

3. *Repeat step 2 as many times as necessary.*

The set of paths P_β contains homotopically distinct simple paths.

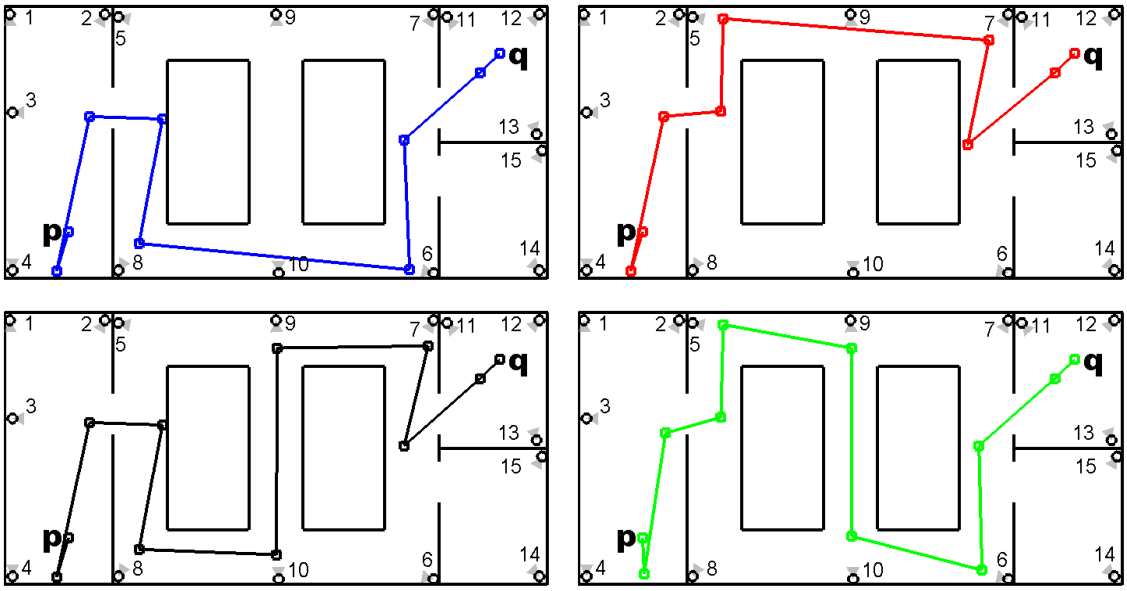

Figure 5.4: Homotopically distinct paths found connecting point p to point q in the environment. Paths are graphically depicted as piecewise linear paths connecting intersect points.

An example showing the resulting paths found by algorithm 5 is shown in figure

5.4. In the figure, four homotopically distinct paths between p and q were found and depicted graphically using line segments between intersect points.

5.5 Discussion

Several application in navigation and path identification using the CN-Complex have been demonstrated in this chapter. The complex is utilized to find trajectories between two points while remaining in a connected coverage. The representation can also be used to identify homotopic paths in the complex. It is possible to extend this work by learning distances between intersect points to solve for minimal length problems between two points.

As seen in section 3.6.2, it is possible to switch the roles of the target and the cameras for navigation purposes. The setup described in this manuscript corresponds to cameras guiding a blind target. If instead we consider an omnidirectional camera in a mobile agent and cameras replaced by unique markers, then it is possible to build a simplicial representation by making local observations from the agent's point of view. This simplicial representation can then be utilized to navigate through the environment.

The navigation procedure described in this chapter can be identified with a hybrid system in which having a mobile agent visible by camera α corresponds to being in discrete state α. Navigation is accomplished by switching between different states and utilizing very simple control laws locally. Utilizing the simplicial complex gives a way to characterize and identify paths in the hybrid system. This is useful for analysis of hybrid systems independent of their particular state structure by extracting global topological features.

Chapter 6

Camera Relations

In this chapter, geometric relations between cameras in an unlocalized network are discovered which provides information about the layout of the network which is complementary to the topological structure recovered by the CN-Complex. For simplicity, cameras that satisfies the assumption for a 2D configuration as specified in section 3.4, i.e. cameras in the plane with only bearing angle information and conic views, are considered. The relations will specify the relative positioning between cameras at different degrees of accuracy. Throughout this discussion, the fact that intersections between cameras can be found robustly will be exploited.

Consider points in the image domain of cameras α and β which can be identified with lines of sight in the physical environment. By determining the pattern of intersection between these lines it is possible to determine something about the relative position between the cameras. Throughout this work it is assumed that the cameras do not have bisecting lines in their fields of view, i.e. the coverage decomposition necessary for obtaining the

CN-Complex has already taken place.

The rest of the discussion is as follows: section 6.1 presents the types of relation that can be obtained between camera pairs; section 6.2 outlines an algorithms for discovering these relations; and section 6.3 illustrates the discovery of relations through some examples.

6.1 Defining Relations

In figure 6.1, several configurations with corresponding lines of sight are considered. Note that if only one line of sight is considered for both cameras (see left plot), we can virtually position both cameras in any location and we will be able to intersect or not intersect these lines of sight, i.e. it is not possible to determine any relation between the cameras.

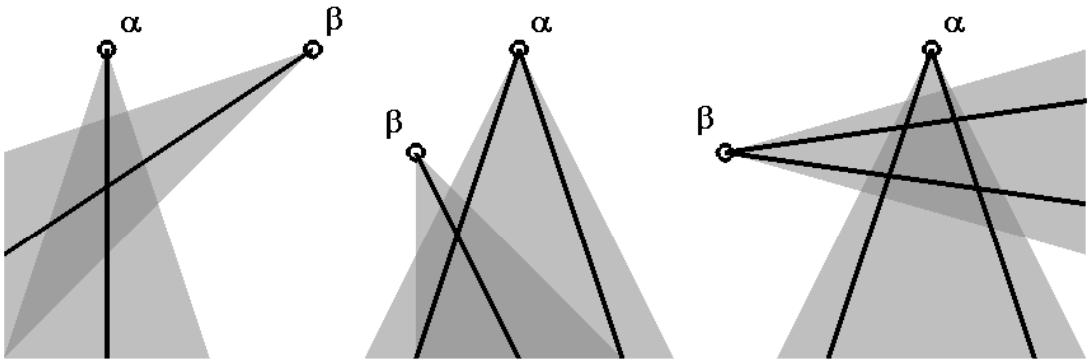

Figure 6.1: Sample configuration for one line of sight for each camera (left). Note that if we assume that these lines intersect and fix the line of sight of camera α, camera β can be virtually anywhere. Sample configuration for two lines of sight for camera α and one for camera β (middle). Sample configuration for two lines of sight for both cameras (right).

In order to analyze the case when there are two lines of sight for camera α and one for camera β, assume that the lines are labeled L_α^l and L_α^r for the left and right lines from camera α, and L_β for the one from β. Also, assume a canonical configuration for camera

α and consider different configuration for β as seen in figure 6.2 (left). Note that there are essentially four regions where camera β can be located with respect to the lines of sight of camera α.

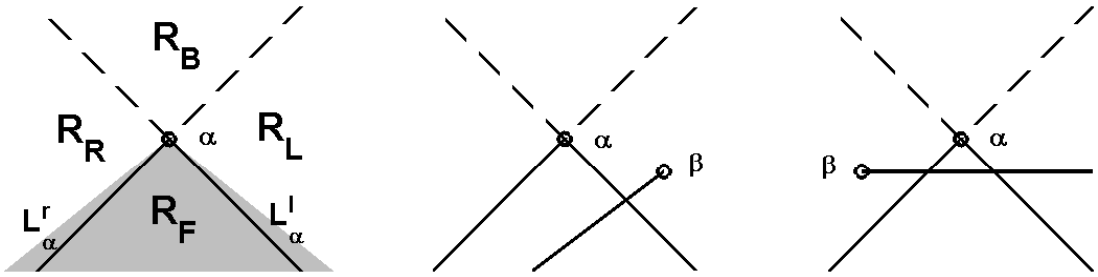

Figure 6.2: Configuration space for camera β given a fixed camera α with line of sights L^l_α and L^r_α (left). We note there are essentially four regions in which camera β can be (i.e. to the left in R_L, to the right in R_R, in the front in R_F, or behind in R_B). A configuration in which $L^l_\alpha \cap L_\beta \neq \emptyset$ and $L^r_\alpha \cap L_\beta = \emptyset$ (middle) is shown. A configuration in which $L^l_\alpha \cap L_\beta \neq \emptyset$ and $L^r_\alpha \cap L_\beta \neq \emptyset$ (right) is shown.

From the point of view of the unlocalized network, it is only possible to determine if $L^l_\alpha \cap L_\beta$ and $L^r_\alpha \cap L_\beta$ are empty or not. Hence, relations between cameras can only be extracted in terms of these quantities. The following theorem summarizes relations that can be found for this scenario.

Theorem 3. *Given camera α with lines of sight L^r_α to the right of L^l_α, and camera β with line of sight L_β, then we have the following relations for β with respect to α:*

$$\begin{aligned} &(00) \rightarrow \beta \in R_L \cup R_R \cup R_F \cup R_B && (01) \rightarrow \beta \in R_L \cup R_F \cup R_B \\ &(10) \rightarrow \beta \in R_R \cup R_F \cup R_B && (11) \rightarrow \beta \in R_L \cup R_R \end{aligned} \tag{6.1}$$

where R_L, R_R, R_F, and R_B are given as in figure 6.2 (left). For simplicity (00) *is used to represent $L^r_\alpha \cap L_\beta = \emptyset \wedge L^l_\alpha \cap L_\beta = \emptyset$,* (01) *represents $L^r_\alpha \cap L_\beta = \emptyset \wedge L^l_\alpha \cap L_\beta \neq \emptyset$, so on.*

A similar result can be shown by considering lines of sight L^l_α and L^r_α for camera α and L^l_β and L^r_β for camera β:

Theorem 4. *Given camera α with lines of sight L^r_α to the right of L^l_α, and camera β with lines of sight L^r_β to the right of L^l_β, then we have the relations shown below for β with respect to α:*

$$
\begin{aligned}
&(0000) \rightarrow \beta \in R_L \cup R_R \cup R_F \cup R_B && (0001) \rightarrow \beta \in R_L \cup R_F \cup R_B \\
&(0010) \rightarrow \beta \in R_L \cup R_F \cup R_B && (0011) \rightarrow \beta \in R_L \cup R_F \cup R_B \\
&(0100) \rightarrow \beta \in R_R \cup R_F \cup R_B && (0101) \rightarrow \beta \in R_L \cup R_R \\
&(0110) \rightarrow \beta \in R_F && (0111) \rightarrow \beta \in \emptyset \\
&(1000) \rightarrow \beta \in R_R \cup R_F \cup R_B && (1001) \rightarrow \beta \in R_F \cup R_B \\
&(1010) \rightarrow \beta \in R_L \cup R_R && (1011) \rightarrow \beta \in R_L \\
&(1100) \rightarrow \beta \in R_R \cup R_F \cup R_B && (1101) \rightarrow \beta \in R_R \\
&(1110) \rightarrow \beta \in \emptyset && (1111) \rightarrow \beta \in R_L \cup R_R
\end{aligned}
\tag{6.2}
$$

where R_L, R_R, R_F, and R_B are given as in figure 6.2 (left). For simplicity we use (0000) *to represent $L^r_\alpha \cap L^r_\beta = \emptyset \wedge L^r_\alpha \cap L^l_\beta = \emptyset \wedge L^l_\alpha \cap L^r_\beta = \emptyset \wedge L^l_\alpha \cap L^l_\beta = \emptyset$,* (0001) *to represent $L^r_\alpha \cap L^r_\beta = \emptyset \wedge L^r_\alpha \cap L^l_\beta = \emptyset \wedge L^l_\alpha \cap L^r_\beta = \emptyset \wedge L^l_\alpha \cap L^l_\beta \neq \emptyset$, and so on.*

The relations in both of the previous theorems can be easily proved by combinatorially considering all possibilities.

6.2 Finding Relations

By relaxing the assumption of lines of sight to using small conic regions in the field of view, it is possible to use the same algorithm used for finding intersection points (see section 4.2) to find intersections between lines of sight. The following is a simple algorithm to discover relations between camera pairs:

Algorithm 6. *Given two cameras α and β (with no bisecting lines; otherwise, their coverage is decomposed), the process is initialized by choosing two points from each field of view, then:*

for *i = 1* **to** *N*
FindIntersections
UpdateRelations
AddLinesOfSight
end

where N is a fixed number of iterations / refinements.

The algorithm starts by choosing two lines of sight from each camera, it finds all intersections between the resulting lines, updates any relations that are discovered between the two cameras, and then adds more lines of sight to further refine the relations. More relations will be discovered as more lines are added, which gives better localization estimates between cameras.

6.3 Examples

In order to illustrate the results from this chapter we consider some basic examples.

In figure 6.3, two cameras with overlapping field of view are shown. The fact that both of these cameras are located within their fields of view can be easily discovered by considering the relations between the lines of sight displayed in the figure.

Next, we consider a more complex example (see figure 6.4) in which there is an object occluding the field of view of one of the camera.

For the configuration in figure 6.4, it is observed that the field of view of camera α needs to be bisected as seen in plot (b). Utilizing the relations between the lines of sight shown in the diagram it is possible to conclude that camera β is to the left of camera α.

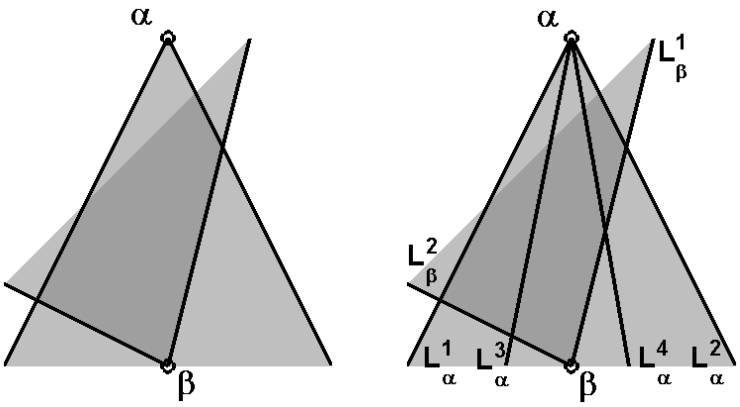

Figure 6.3: Finding relations between two cameras: Initial setup of the cameras (left). Cameras after adding some bisecting lines to their field of view (right). Note that by considering the lines L^1_α, L^2_α, L^1_β and L^2_β we conclude that camera β is in front of camera α between L^1_α and L^2_α. By considering L^3_α, L^4_α, L^1_β and L^2_β we can further conclude that β is in front of α between L^3_α and L^4_α.

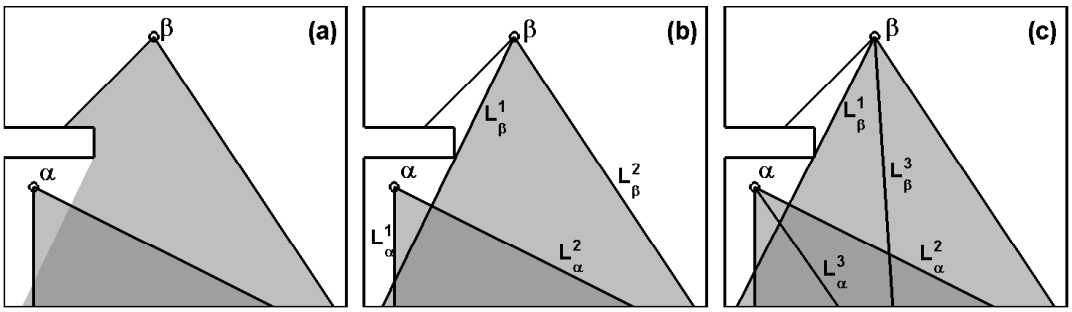

Figure 6.4: Original configuration of two cameras (left). Camera β is bisected in order to have a field of view without bisecting lines (middle). Note that the relation discovered from lines L^1_α, L^2_α, L^1_β and L^2_β tells us that camera β is either to the right or the left of camera α. Utilizing lines L^3_α, L^2_α, L^1_β and L^3_β we conclude that camera β is to the left of camera α.

6.4 Discussion

In this chapter relations between cameras were discovered by analyzing the intersection between lines of sight. The analysis becomes very simple when assuming that the coverage does not have any bisecting lines, hence constructing the CN-Complex is a prerequisite for this analysis. The more relations that are discovered, the more precise the relative position between cameras becomes. By utilizing the methods described above, it is possible to start from the CN-Complex and further refine the model to the point that relations (and hence localization) is available between all cameras pairs.

It is possible to extend this work to cluster cameras that satisfy certain geometric relations such as covering the same room in an office space. In this scenario, a camera can be used to identify the location of an entrance (perhaps by looking at the types of occlusions in its field of view), and then other cameras that are in appropriate relative positions can be identified as located on the same side of the entrance (and hence in the same room). Also, this local information could be integrated in order to build more sophisticated models of the environment such as a Voronoi diagram of the environment.

Chapter 7

Occlusion Detection in Non-Static Scenes

The CN-Complex of a camera network is built from detecting occlusions from each camera view. Its construction (as presented in chapter 4) utilizes detections of occlusions of targets given a static background model. However, background images can change due to small perturbations of a camera's position. Also, as described in section 3.6, it is useful to detect occlusions in non-static scenes when considering mobile extensions. In this chapter, occlusion detection in non-static scenes will be studied through the use of persistent topological features in order to deal with these scenarios.

Occluding contours are a commonplace in synthetic and natural scenes. Since occlusions correspond to locations in an image where one surface is closer to the camera than another, they provide critical cues about the 3D structure of a scene. Given this utility, it is unsurprising that their detection has numerous applications in shape extraction,

figure-ground separation, and motion segmentation, e.g. [3, 4, 14, 23, 39, 49, 51]. Occlusion boundaries also result in the appearance or disappearance of regions which make occlusions the source of notorious difficulty for many patch-based computer vision algorithms, e.g. [18, 21, 26, 55]. The goal of this chapter is to present a completely local, bottom-up approach to detect and localize occlusions in order to provide this powerful low-level information to higher-level reasoning methods (see figure 7.1 for an example).

Occlusions are undetectable from a single image and instead must be found by comparing several images. Traditional occlusion detectors rely almost entirely upon spatio-temporal derivatives or matching to detect the artifacts of occlusions. Generally, these artifacts fall into two categories: motion inconsistency or the classical T-junction. Unfortunately, both of these methods have shortcomings that render them unreliable as occlusion detectors. Motion inconsistencies are found by relying almost entirely on noise sensitive derivatives. Though there are numerous ways to find T-junctions, each method makes assumptions about the orientations of the occluding contour. Moreover, even after a T-junction has been detected, an occlusion may not be present. In contrast to these methods, we model the cause of occlusions and show that the proper measurement of certain robust topological invariants is a definitive indicator to the presence of an occlusion. Detections occur in regions where new color information becomes available, exactly the region at which motion estimates are the most unreliable. Therefore, the framework presented in this chapter can be viewed as complimentary to most motion estimation approaches. The strength of our framework is that it is able to operate at different scales providing information that may otherwise be unavailable while not relying on noisy derivatives, making rigid assump-

tions about the orientation of occluding contours, building complex appearance models, or performing any matching.

Figure 7.1: Three consecutive frames from a walking sequence are shown. The detection of occlusions using local topological invariants is made with respect to the middle frame (top-left). Appearances between the middle and bottom frames (bottom-left) are marked as red in the middle frame, and disappearances between the middle and top frames (top right) are marked as blue in the middle frame. All of the occlusions are compiled in the bottom right plot.

The main contributions of this work are four-fold. First, in section 7.2, an image model is introduced and it is shown that, under Lipschitz continuous camera or object movement, occlusions occur if topological invariants are not preserved. Second, in section 7.3, a robust measure to determine these topological invariants is developed for a given segmentation of the image. Third, in section 7.4, the topological invariants are rigorously defined to guarantee localization of occlusions within an image. Finally, in section 7.5, the

performance of the occlusion detector is demonstrated on synthetic and natural data.

7.1 Related Work

As described earlier, traditional approaches to occlusion detection can generally be divided into two categories: those that attempt to detect motion inconsistencies and those that detect T-junctions. Inspired by the classic work of Horn and Schunck [22] and the observation that the motion between two sides of an edge at an occlusion will be dissimilar, the motion inconsistency domain makes few assumptions about the camera path and instead relies on accurate estimates of local motion. These methods are completely local and employ derivatives that carry little information. As a result, they are often inaccurate and noisy. The algorithms in this domain can be classified by the varying rigidity of assumptions used in order to make the motion estimate robust. T-junctions as an indicator for occlusions traces its roots to an observation made by Irving Biederman [6]. Unfortunately, not all T-junctions are occlusions. Most algorithms in the T-junction detection domain can be classified according to the methodology they employ to detect and classify them.

At one extreme of motion estimation is the class of layered motion segmentation algorithms which segment regions based on the consistency of motion [40, 46, 52, 54]. These techniques use a parametric motion model that is restricted to near-planar, rigidly-moving regions for each layer and employ a variety of techniques for estimating these models. They estimate motion accurately by assuming a known, fixed number of layers in the scene and do not scale well as the number of layers increases. We argue that attempting to explain the scene in terms of a fixed number of motion-consistent connected regions is

too demanding a requirement. We propose that the low-level reasoning provided by the occlusion detector presented in this chapter can provide more appropriate cues to high-level reasoning algorithms like those performing layered motion segmentation.

At the other extreme of motion estimators are those that make the estimate robust by smoothing the velocity field spatially [2] or temporally [7]. Regrettably, this has the unintended consequence of making the motion estimate inaccurate at boundaries, which is where occlusions occur. An alternative to this smoothing approach is the use of an implicit model, either learned from local motion cues estimated from training data or based on some fixed model of the distribution of motion cues in the vicinity of occluding boundaries [8, 15, 38, 49, 48]. Though these approaches are appealing because they rely on well-defined statistical models, they are still sensitive to deviations of the actual data from the trained model. Even though they improve the robustness of the calculation of the motion estimate, they still rely entirely upon it.

T-junction detection has a rich history. Until recently, there have been two predominant approaches to T-junction detection: gradient or filter-based approaches [5, 16, 27, 42, 45] and model-based template matching [41]. These approaches work singularly to detect the T-junctions rather than distinguish an occluding T-junction from a non-occluding T-junction. More recently, Favaro et al. define what they call a proper T-junction as a T-junction at which an occlusion takes place [14]. They detect these proper T-junctions by exploiting a rank constraint on a data matrix of feature tracks that would normally be classified as outliers in a multiple-view geometry problem. Although mathematically correct, the method has the deficiency of being overly sensitive to even slight deviations

from the given rank condition. Inspired by this work, other alternatives have exploited a discriminative framework to classify these proper T-junctions [3, 4]. Unfortunately, these methods utilize 2D spatio-temporal slices instead of volumes which mean that detections can only be made in fixed orientations.

In contrast to prior work, we show that under a well-defined imaging model, occlusions occur when images are not related by a deformation. Given this powerful result, any properly defined topological invariant could detect the existence of an occlusion. In this chapter, we will construct several such robust invariants that are able to operate at different scales providing information that may otherwise be unavailable, while not relying on the calculation of motion models or hinge upon the detection and classification of T-junctions.

7.2 Notation and Image Model

This section is meant to introduce the mathematical framework to be used throughout the rest of the chapter. We begin this section by introducing our imaging model and reviewing some simple topological concepts. We conclude this section by describing a naïve occlusion detector. For a more formal discussion of the topological concepts presented here, we refer the interested reader to Hatcher [19] or Munkres [37].

Definition 15. *An* **image** *is a collection of disjoint open sets and colors denoted by* $\mathcal{I} := \{(E_i, c_i)\}_{i=1}^{Ns}$*, where* $E_i \subset \mathbb{R}^2$*,* $c_i \in \mathbb{R}^d$*, each* E_i *consists of a finite number of open connected components, and* $N_s < \infty$*. Similarly we denote an image sequence by* $\mathcal{I}^t := \{(E_i^t, c_i^t)\}_{i=1}^{Ns}$ *for* $t \in \mathbb{Z}$*.*

Observe the sets E_i do not need to form a partition of $\mathbb{R}^2$. For simplicity, we

assume $d = 1$.

Definition 16. *An* **image deformation** *is a continuous function $G : \mathbb{R}^2 \times [0,1] \to \mathbb{R}^2$ for which $G(x,0)$ is the identity map and $G(\cdot, s)$ is a homeomorphism for each t. An image deformation G is* **Lipschitz** *if there exists a constant $C > 0$ such that for all $s_1, s_2 \in \mathbb{Z}$ and $x \in \mathbb{R}^2$*

$$||G(x, s_2) - G(x, s_1)|| \leq C\,|s_2 - s_1|. \tag{7.1}$$

Definition 17. *A* **topological invariant** *is a property of a topological space which is invariant under homeomorphisms. Two examples of such invariants are of particular interest:* **Betti Zero**, *denoted by $\beta_0(E)$, which counts the number of connected components in a set $E \subset \mathbb{R}^2$; and* **Betti One**, *denoted by $\beta_1(E)$, which counts the number of holes or loops in a set $E \subset \mathbb{R}^2$.*

Topological invariants are extremely useful properties of topological spaces since they allow for the comparison of spaces without explicit matching. In addition, if the definition of these invariants is made carefully they can allow us to locally compare spaces.

Definition 18. *Given an image $\mathcal{I} = \{(E_i, c_i)\}$ and a partition $\mathcal{B} = \{B_k\}_{k=1}^{Np}$ of $\mathbb{R}$, the $\beta_\mathbf{0}$* **histogram at a closed set** *K, denoted by $\alpha(\mathcal{I}|K)$, is a vector with entries given by*

$$\alpha_k(\mathcal{I}|K) := \sum_{c \in A_k} \beta_0 \left(\bigcup_{c_j = c} E_j \cap K \right), \tag{7.2}$$

where $A_k = \{c_i \mid c_i \in B_k\}$. The β_0 **histogram with boundary identification**, *denoted by $\overline{\alpha}_k(\mathcal{I}|K)$, is defined in the same way except that β_0 is calculated after identifying the boundary of K to a single point. That is, sets that intersect the boundary of K are considered connected.*

The importance of the identification concept will be made clear in section 7.4.

Definition 19. *Under the same assumptions as the previous definition, the* β_1 **histogram at a set** K *for an image* $\mathcal{I}$*, denoted by* $\gamma(\mathcal{I}|K)$*, is a vector with entries given by*

$$\gamma_k(\mathcal{I}|K) := \sum_{c \in A_k} \beta_1 \left(\bigcup_{c_j = c} E_j \cap K \right). \tag{7.3}$$

Given these definitions, we make several observations. If an image is thought of as a collection of sets $E_i \subset \mathbb{R}^2$, coming from a single object, then small changes of perspective would be equivalent to Lipschitz image deformations. Making this more explicit, consider a Lipschitz image deformation G with Lipschitz constant C. If $E_i \subset D(y, r)$ for all i, where $D(y, r)$ is the disk centered at $y \in \mathbb{R}^2$ of radius $r > 0$, then $G(E_i, s) \subset D(y, r + C \cdot s)$. Since the collection of sets E_i and $G(E_i, s)$ are related via a homeomorphism, they must have the same number of connected components and holes. If, instead, these invariants were not preserved one of the sets must have disappeared which would be evidence of an occlusion. The preservation of topological invariants, like Betti Zero or One, would then serve as indicators of occlusions.

At this point, we have seemingly constructed a naïve occlusion detector, but three nontrivial problems remain to be addressed. First, Betti Zero and One as defined do not take advantage of the color of images. We introduce a procedure to exploit the color of images to improve the robustness of the calculations of these topological invariants by using definitions 18 and 19. Second, the procedure as described provides no localization of an occlusion within a given image. We show that via a careful redefinition of Betti Zero and One, we can localize an occlusion. Finally, there exists innumerable ways of dividing the image domain into a collection of sets $E_i \subset \mathbb{R}^2$, each producing a different number of

connected components and holes. Calculations of these invariants can be made robust to this choice of segmentation by exploiting the persistence of topological features.

7.3 Histogram Flows

This section, as presented, may seem a sort of digression. This is done purposefully as the material discussed here has greater utility than the one suggested herein. The goal of this section is to define a relation between feature vectors based on a partition and the allowed variations of the color space ($\mathbb{R}$ in our case). The work in this section will allow us to extend the concepts of Betti Zero and One to exploit the color of images.

Definition 20. *Let $\mathcal{I}^t = \{(E_i^t, c_i^t)\}_{i=1}^{Ns}$ be an image sequence. A partition $\mathcal{B} = \{B_k\}_{k=1}^{Np}$ of $\mathbb{R}$ for this image sequence, where the B_k are an ordered sequence of intervals of the form $[a_k, b_k)$ each with length $|B_k|$, is said to be a* **simple partition** *if the following conditions are satisfied for all $t \in \mathbb{Z}$ and all $1 \leq i \leq N_s$*

- $-|B_{k-1}| < c_i^{t+1} - c_i^t < |B_{k+1}|$ *whenever* $c_i^t \in B_k$ *and* $1 < k < N_p$,
- $c_i^{t+1} - c_i^t < |B_2|$ *whenever* $c_i^t \in B_1$, *and*
- $-|B_{Np-1}| < c_i^{t+1} - c_i^t$ *whenever* $c_i^t \in B_{Np}$.

The sets B_k are referred to as **bins**.

The conditions above guarantee that over one unit of time the color can only move from one bin to its neighbor.

Proposition 1. *Given a simple partition* $\mathcal{B} = \{B_k\}_{k=1}^{Np}$ *for an image sequence* $\mathcal{I}^t = \{(E_i^t, c_i^t)\}_{i=1}^{Ns}$. *Whenever* $c_i^t \in B_k$ *we have that* $c_i^{t+1} \in B_1 \cup B_2$ *if* $k = 1$, $c_i^{t+1} \in B_{k-1} \cup B_k \cup B_{k+1}$ *if* $1 < k < N_p$, *and* $c_i^{t+1} \in B_{Np-1} \cup B_{Np1}$ *if* $k = N_p$.

The proof of this proposition can be found in appendix B. In section 7.4, we will show how a simple partition $\mathcal{B}$ can generate histogram vectors, which count features in different bins. It is possible to treat these vectors as feature vectors in $\mathbb{R}^{Np}$, and compare them by using Chi-Squared or Euclidean distance; however, small variations in the color of an image can generate large changes with respect to these norms. Therefore, it is critical to determine the allowable variations between histograms.

7.3.1 Defining Flows

Although between times t and $t+1$ the histogram vectors given in definitions 18 and 19 may change, the types of transitions are limited by the simple partition $\mathcal{B}$. A set with color $c_i^t \in B_k$ can only change color to bins B_{k-1} or B_{k+1}. We define a right flow f_k^R from B_k to B_{k+1} corresponding to the number of sets that migrated from B_k to B_{k+1} and a corresponding left flow f_k^L from B_k to B_{k-1}. Figure 7.2 illustrates this idea for a given histogram vector v.

First, by our definition of simple partition the flows must satisfy the following

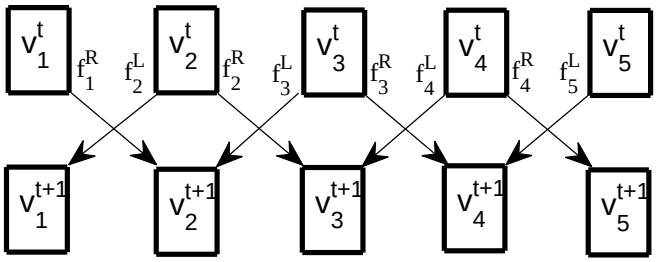

Figure 7.2: Diagram illustrating flows f^R and f^L between adjacent bins for a histogram vector v where $N_p = 5$.

constraints:

$$\begin{aligned} f_k^R &\geq 0,\ 1 \leq k \leq N_p \\ f_k^L &\geq 0,\ 1 \leq k \leq N_p \\ f_1^R &\leq v_1^t \\ f_1^L &= 0 \\ f_k^R + f_k^L &\leq v_k^t,\ 1 < k < N_p \\ f_{Np}^R &= 0 \\ f_{Np}^L &\leq v_{Np}^t \end{aligned} \tag{7.4}$$

Given flows from frame t to frame $t+1$, we observe:

$$v^{t+1} = v^t + A^R f^R + A^L f^L, \tag{7.5}$$

where the matrices $A^L \in \mathbb{R}^{Np \times Np}$ and $A^R \in \mathbb{R}^{Np \times Np}$ have entries:

$$A_{ij}^L = \begin{cases} -1 & \text{if } i = j \\ 1 & \text{if } i = j - 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

and

$$A^R_{ij} = \begin{cases} -1 & \text{if } i = j \\ 1 & \text{if } i = j+1 \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

When the flow conditions go unsatisfied, an anomaly between a pair of images is detected.

Definition 21. *A histogram vector w is said to be* **explained** *based on histogram vector v, denoted as $w \prec v$, if there exist flow vector f^R and f^L that satisfy equation 7.4 and*

$$w \leq v + A^R f^R + A^L f^L,$$

where the inequality is satisfied at every entry of the vector.

If the histogram vector of an image at time $t+1$ cannot be explained by the vector at time t, then we know something significant has changed. This will be made more precise in section 7.4.

7.3.2 Determining Histogram Flows

In this section, a greedy algorithm to calculate flows to explain a desired histogram is described. Given a base histogram v, a desired histogram w that we want to explain, and vector flows f^R and f^L, we will want to know by how much the vector $v' := v + A^R f^R + A^L f^L$ fails to satisfy the condition $w \leq v'$. We introduce the score:

$$\rho(f^R, f^L \mid v, w) = \sum_k (w_k - v'_k) \cdot \quad \{w_k - v'_k > 0\}. \tag{7.8}$$

We employ the following greedy algorithm to calculate the flows:

Algorithm 7. *The inputs are the histogram to be explained w and the base histogram v. The outputs are the flows f^R and f^L and the resulting histogram v' (i.e. v after applying the flows).*

$v' = v; \quad f^R = 0; \quad f^L = 0$

for $k = 2$ **to** $(N_p - 1)$

$d = w_k - v'_k$

if $d \leq 0$

continue

end

$u_L = \max(\min(d, v_{k-1} - f^L_{k-1} - f^R_{k-1}), 0)$

$u_R = \max(\min(d + u_L, v_{k+1} - f^L_{k+1} - f^R_{k+1}), 0)$

$f^R_{k-1} = f^R_{k-1} + u_L$

$f^L_{k+1} = f^L_{k+1} + u_R$

$v'_{k-1} = v'_{k-1} - u_L$

$v'_k = v'_k + u_L + u_R$

$v'_{k+1} = v'_{k+1} - u_R$

end

The algorithm proceeds by updating the current flows and histogram entries based on the values of the neighboring entries only if it will improve the score ρ.

7.4 Occlusion Indicators

The purpose of this section will be the construction of a localized occlusion detector by combining the construction presented in the previous section with the definition of Betti

Zero and One. In order to illustrate the difficulty of simply naïvely comparing the count of connected components to detect occlusions, we consider the sets in figure 7.3.

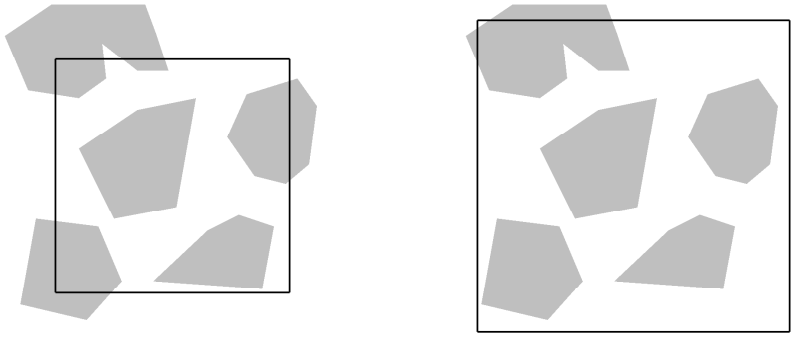

Figure 7.3: Illustration of how to count connected components for neighborhood K_r (left) and K_{r+C} (right). There are 5 connected components in K_{r+C}. There are 6 connected components in K_r and 3 connected components after boundary identification. Without boundary identification we could erroneously conclude that a set disappeared.

In this case, the two images are related by an image deformation with Lipschitz constant C. A simple comparison of the number of connected components in the drawn neighborhoods, K_r and K_{r+C}, suggests that an occlusion has occurred (the left image has 6 connected components and the right image has 5 connected components); however, this is clearly not the case. The problem arises because we count the same set twice. We can remedy this problem by identifying every point on the boundary of the left image as belonging to the same set. This is the inspiration for the definition of the β_0 histogram with boundary identification. Making this small change in how we perform the calculation of connected components guarantees that under Lipschitz image deformations, the number of connected components in K_r is always less than the number of connected components in K_{r+C}.

If we employ this procedure in the example and compare the number of connected

components, we find that it has increased, which means no occlusion has taken place (the left image under identification has 3 connected components and the right image still has 5 connected components). This result is easily combined with the histogram flow procedure presented in the previous section to produce a robust measure of topological invariants:

Theorem 5. *Given a collection of Lipschitz image deformations G^t with constant C, images $\mathcal{I}^t$ and a simple partition $\mathcal{B} = \{B_k\}$, such that $G^t(E_i^t, 1) = E_i^{t+1}$ for every i and t, then*

$$\overline{\alpha}(\mathcal{I}^t|K_r) \prec \alpha(\mathcal{I}^{t+1}|K_{r+C}) \tag{7.9}$$

where $K_r := \{(x_1, x_2) \mid max(|x_1|, |x_2|) \leq r\}$.

Theorem 6. *Under the same assumptions as the previous theorem, we have*

$$\gamma(\mathcal{I}^t|K_r) \prec \gamma(\mathcal{I}^{t+1}|K_{r+C}). \tag{7.10}$$

If the criteria presented in these theorems goes unsatisfied, then an occlusion must have taken place between times t and $t+1$. The roles of $\mathcal{I}^t$ and $\mathcal{I}^{t+1}$ can be interchanged and the corresponding condition will remain true as long as no object has appeared between times t and $t+1$. The proof of these theorems can be found in the appendix C.

7.5 Analysis

In this section, we will analyze the performance of the occlusions detectors presented in this chapter on a dataset that includes both natural and synthetic image sequences. In order to simplify our analysis, we focus our attention on the β_0 histogram in tandem with equation 7.9. Though a few datasets exist with labeled occlusions, particularly the

one built by Stein et al. [48], they all capture images at an inadequate frame rate, which makes applying the framework presented in this chapter difficult.

Our analysis is performed by first decomposing the image domain into squares of length r_1, which we identify as the sets K_r from our previous discussion. We also consider squares of length $r_2 > r_1$, which we identify as the sets K_{r+C}, with coincident centers with the previous squares. We take an image $\mathcal{I}^1$ and try to explain the observations in the squares of length r_1 by using the observations in $\mathcal{I}^0$ and $\mathcal{I}^2$ specifically within the squares of length r_2. In order to do this, we construct a segmentation of each square by utilizing the results of section 2.4, then compute the histograms and their flows by using algorithm 7. Whenever the conditions in equation 7.9 go unsatisfied, we mark this as a detection.

In the formulation of theorem 5, the radius of the window and the type of partition of the color space are free parameters to be set. The Lipschitz constant, on the other hand, is something that must be well approximated. In order to better understand the role of the partition $\mathcal{B}$ and the radius r of the window, we consider the performance of our model under extreme choices of these parameters.

First, consider the extreme cases for the partition of the color space: either a single interval for all $\mathbb{R}$ or a simple partition whose length goes to zero. When the partition is equal to $\mathbb{R}$, it is only possible to detect whether a set is present or not. If instead a simple partition whose length goes to zero is considered, then the colors in the image can never change. Since real images are quantized, this is equivalent to assuming the images are piecewise constant.

Second, consider the extreme cases for the radius of the window: either the window

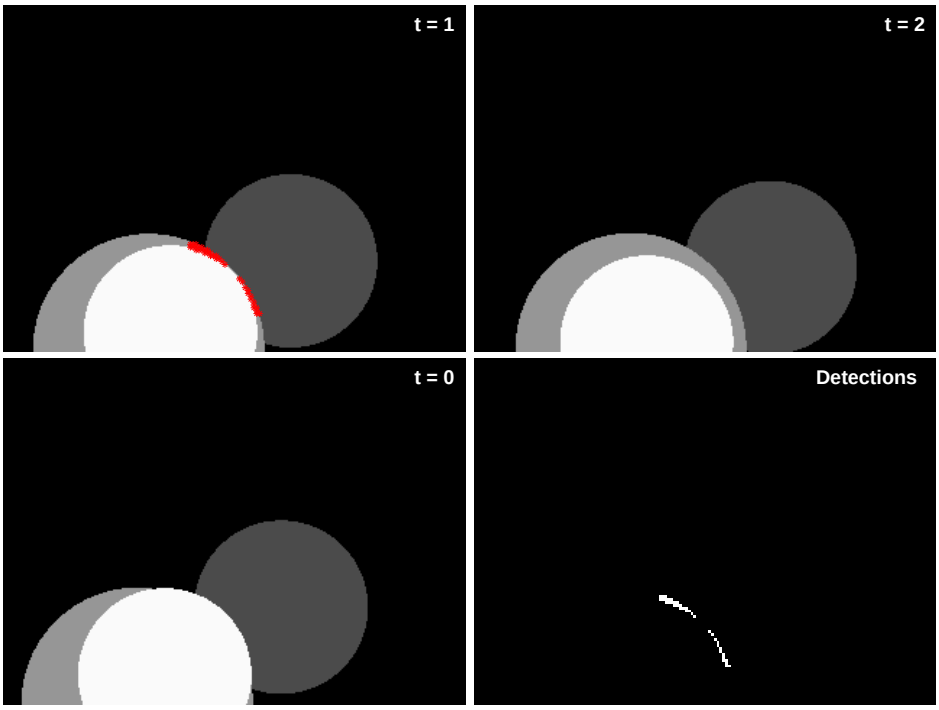

Figure 7.4: Three consecutive frames from synthetic sequence are shown. The color used for the sets are: 0, 75, 150 and 250. The detection of occlusions using local topological invariants is made with respect to the middle frame (top-left). Appearances between the middle and bottom frames (bottom-left) are marked as red in the middle frame, and disappearances between the middle and top frames (top right) are marked as blue in the middle frame. All of the occlusions are compiled in the bottom right plot.

is the size of the whole image or the window is the size of a single pixel. Although using a whole image window size allow for a more robust segmentation, it allows for no localization of the occlusion. For a window the size of a single pixel, due to interpolation during sampling and image noise, we may have to choose larger partitions of the color space which has the unintended consequence of making our detections sparser.

The detection process is illustrated on a sequence of synthetic images. Each image in the sequence consists of three layers of circles moving in the plane as illustrated in figure 7.4. We perform detections by dividing the image into windows of length 2 and larger

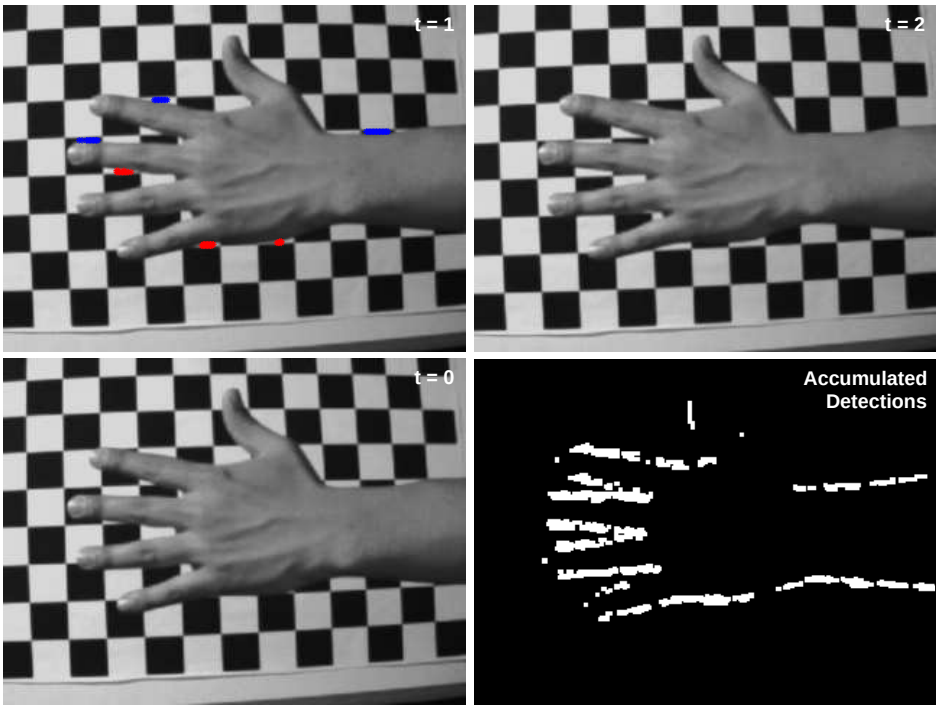

Figure 7.5: Three consecutive frames from a sequence of a hand in front of a moving Macbeth board are shown. The detection of occlusions using local topological invariants is made with respect to the middle frame (top-left). Appearances between the middle and bottom frames (bottom-left) are marked as red in the middle frame, and disappearances between the middle and top frames (top right) are marked as blue in the middle frame. The accumulated occlusions over the entire sequence are compiled in the bottom right plot.

windows of length 8. Our simple partition of the color space corresponds to bins of length 32 from 0 to 256. At this scale, we are able to detect the appearance of the larger circle from behind the smaller one. We can also perform the computation at the scale of the whole image. In this case, we observe that $\overline{\alpha}(\mathcal{I}^1|K_r) = \alpha(\mathcal{I}^0|K_{r+C}) = \alpha(\mathcal{I}^2|K_{r+C}) = (1,0,1,0,1,0,0,1)^T$. Hence, no occlusion is detected at the scale of the whole image. This simple example illustrates the importance of selecting the correct window size. A similar analysis is performed on several natural image sequences and depict a few of the results in figures 7.1 and 7.5.

7.6 Discussion

In this chapter, a mathematical framework for performing occlusion detection by employing local topological invariants was presented. Under an imaging model that allows for deformation and color variation, occlusions are detected without employing matching or spatio-temporal derivatives. The framework quickly generalizes to higher dimensional datasets and other higher dimensional topological invariants.

Occlusion detection has numerous applications in shape extraction, figure-ground separation, and motion segmentation. The work presented here can be easily extended to perform robust figure-ground separation when the background model undergoes Lipschitz deformations. Anomalies in the observations can be identified as foreground. Most current such algorithms employ a fixed statistical model for the variation allowed in the background, but the framework presented here is more general and can work in tandem with a statistical model attached to the histogram flows.

The framework described in this chapter is general enough to create topological histograms which carry more information than just color. The histograms can be interpreted as descriptors of an image and can be compared via theorems 5 and 6. For example, they can include gradient information which would allow us to track model deformation in addition to color changes. This work can be extended by incorporating prior estimates of displacement which can be utilized to choose smaller windows. Statistical learning methods, in particular, can be used to select appropriate parameters.

Note that the idea of the histogram bins can also be extended to bins in the plain (rather than just in the color space). This leads to solving for flows of pixels between

different regions which in turn generalizes to solving for motion flow in the image plane. This is how refinement of our model and assumptions connects simple topological constraints to more general ones.

Also, having new sets introduced in an image can be thought of as new information in which new objects have been discovered. Utilizing this insight, it is possible to guide changes of perspective such that unseen regions become visible. This can lead to developing algorithm for "topological exploration" of a scene, looking for views that maximize the discovery of new topological features. This idea can also be coupled with the construction of the CN-Complex.

Bibliography

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

[2] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 283–310, 1989.

[3] N. Apostoloff and A. Fitzgibbon, "Learning Spatiotemporal T-Junctions for Occlusion Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[4] ——, "Automatic video segmentation using spatiotemporal T-junctions," in *British Machine Vision Conference*, vol. 3, 2006, p. 1089.

[5] D. Beymer, M. I. of Technology, and A. I. Laboratory, *Finding Junctions Using the Image Gradient*. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1991.

[6] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 2, pp. 115–147, 1987.

[7] M. Black and P. Anandan, "Robust dynamic motion estimation over time," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, 1991, pp. 296–302.

[8] M. Black and D. Fleet, "Probabilistic detection and tracking of motion discontinuities," *International Journal of Computer Vision*, vol. 38, no. 3, pp. 231–245, 2000.

[9] R. Bott and L. Tu, *Differential Forms in Algebraic Topology*. Springer, 1995.

[10] 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.

[11] 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.

[12] 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.

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

[14] P. Favaro, A. Duci, Y. Ma, and S. Soatto, "On exploiting occlusions in multiple-view geometry," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 479–486.

[15] D. Fleet, M. Black, and O. Nestares, "Bayesian inference of visual motion boundaries," 2003.

[16] W. Freeman and E. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, 1991.

[17] 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.

[18] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," in *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings.*, 1997, pp. 858–863.

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

[20] 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.

[21] H. Hirschmüller, P. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 229–246, 2002.

[22] B. Horn and B. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[23] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 5–16, 1994.

[24] 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.

[25] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*. Springer, 2003.

[26] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory andexperiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 920–932, 1994.

[27] D. Li, G. Sullivan, and K. Baker, "Edge detection at junctions," in *Proceedings Alvey Vision Conference*, vol. 2, 1989.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the Second international conference on Embedded networked sensor systems*, 2004.

[34] 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.

[35] A. Muhammad and A. Jadbabaie, "Decentralized computation of homology groups in networks by gossip," in *Proceedings of the American Control Conference*, 2007.

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

[37] ——, *Elements of algebraic topology.* Addison Wesley Publishing Company, 1993.

[38] O. Nestares and D. Fleet, "Probabilistic tracking of motion boundaries with spatiotemporal predictions," in *IEEE Conference on Computer Vision and and Pattern Recognition*, vol. 2, 2001, pp. 358–365.

[39] S. Niyogi and E. Adelson, "Analyzing and recognizing walking figures in XYT," in *IEEE Conference on Computer Vision and Pattern Recognition.*, 1994, pp. 469–474.

[40] A. Ogale, C. Fermuller, and Y. Aloimonos, "Motion segmentation using occlusions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 6, pp. 988–992, 2005.

[41] L. Parida, D. Geiger, and R. Hummel, "Junctions: detection, classification, and reconstruction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 7, pp. 687–698, 1998.

[42] P. Perona, "Steerable-Scalable Kernels for Edge Detection and Junction Analysis," in *Proceedings of the Second European Conference on Computer Vision*. Springer-Verlag London, UK, 1992, pp. 3–18.

[43] 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.

[44] 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.

[45] E. Simoncelli and H. Farid, "Steerable wedge filters for local orientation analysis," *IEEE Transactions on Image Processing*, vol. 5, no. 9, pp. 1377–1382, 1996.

[46] P. Smith, T. Drummond, and R. Cipolla, "Layered motion segmentation and depth ordering by tracking edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, pp. 479–494, 2004.

[47] 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.

[48] A. Stein, D. Hoiem, and M. Hebert, "Learning to Find Object Boundaries Using Motion Cues," in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[49] A. Stein and M. Hebert, "Local detection of occlusion boundaries in video," *Image and Vision Computing*, 2008.

[50] 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.

[51] B. Wu and R. Nevatia, "Detection and segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses," *International Journal of Computer Vision*, vol. 82, pp. 185–204, 2009.

[52] J. Xiao and M. Shah, "Accurate motion layer segmentation and matting," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 2, 2005.

[53] 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.

[54] P. Yin, A. Criminisi, J. Winn, and I. Essa, "Tree-based classifiers for bilayer video segmentation," in *Proc. CVPR*, 2007.

[55] C. Zitnick and T. Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 675–684, 2000.

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

[57] 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.

Appendix A

Proof of Decomposition Theorem 2

In this appendix, a detailed proof of the result stated in theorem 2 for the 2D configuration described in section 3.2.3 is provided.

Throughout this appendix we consider a finite set of cameras indexed by $\alpha \in \{1, 2, 3 \cdots N_c\}$ with corresponding domains $\mathcal{D}_\alpha$ and coverages $\mathcal{C}_\alpha$. Each camera coverage is decomposed by all possible bisecting lines $\{L_{\alpha,i}\}$. The collection $\{\mathcal{C}_{\alpha,j}\}$ is the result of this decomposition, where $\mathcal{C}_{\alpha,j} := \mathcal{C}_\alpha \cap K_{\alpha,j}$ and $K_{\alpha,j}$ is the convex cone resulting from decomposing the plane using the lines $\{L_{\alpha,i}\}$ (see definition 13).

Observation 3. *It may be useful for the reader to think of the set $\mathcal{C}_n$ (the visible set after object occlusions have been removed) as the intersection of a convex set (i.e., the camera domain) with a star convex set (due to visibility from o_α).*

Observation 4. *The number of bisecting lines for a given camera in our environment is finite since we are considering finite number of objects in the coverage with piecewise linear boundaries.*

Definition 22. *The* **line segment joining points** p **and** q *is denoted by* $\overline{p\,q}$. *The* **line passing through points** p **and** q *is denoted by* $L(p,q)$.

Definition 23. *The* **triangle formed by points** a**,** b **and** $c \in \mathbb{R}^2$ *is the convex hull of these three points and it is denoted* $\Delta_{a,b,c}$.

Lemma 1. *Given that* $o_\alpha, p \in \mathcal{C}_\alpha$ *then* $\overline{o_\alpha\, p} \in \mathcal{C}_\alpha$.

Proof. Since, o_α and $p \in \mathcal{C}_\alpha \subset \mathcal{D}_\alpha$, then $\overline{o_\alpha\, p} \subset \mathcal{D}_\alpha$ due to convexity of $\mathcal{D}_\alpha$. Let $r \in \overline{o_\alpha\, p}$. If r is not visible then $\overline{o_\alpha\, r} \cap \bigcup \mathcal{O}_i \neq \emptyset$ (where $\{\mathcal{O}_i\}$ is the collection of objects in the environment). However, this implies that $\overline{o_\alpha\, p} \cap \bigcup \mathcal{O}_i \neq \emptyset$. Hence, we conclude that p is not visible, which is a contradiction. Therefore, r must be visible. Since r was arbitrary then $\overline{o_\alpha\, p}$ is visible. □

Lemma 2. *Given that* $p, q \in \mathcal{C}_\alpha$ *with*

$$L(p, o_\alpha) = L(q, o_\alpha),$$

then $\overline{p\,q} \in \mathcal{C}_\alpha$. *That is, if* p *and* q *are visible and are in the same line of sight, then the line joining them is visible too.*

Proof. This follows from the definition of $\mathcal{C}_\alpha$ and the domain of a camera $\mathcal{D}_\alpha$. We know that $\mathcal{D}_\alpha$ is convex, so $\overline{p\,q} \subset \mathcal{D}_\alpha$ since $p, q \in \mathcal{C}_\alpha \subset \mathcal{D}_\alpha$.

From our assumption $L(p, o_\alpha) = L(q, o_\alpha)$, it is possible to conclude that for $r \in \overline{p\,q}$ then $r \in \mathcal{D}_\alpha$, and $r \in \overline{o_\alpha\, p}$ or $r \in \overline{o_\alpha\, q}$. Basically, there are only two cases, both p and q on the same side of o_α or on opposites sides. Either way, r must be in $\overline{o_\alpha\, p}$ or $\overline{o_\alpha\, q}$.

Without loss of generality, assume $r \in \overline{o_\alpha\, p}$. If r was not visible, the

$$\overline{o_\alpha\, r} \cap \bigcup \mathcal{O}_i \neq \emptyset$$

(where $\{\mathcal{O}_i\}$ is the collection of sets representing the objects in the space). This implies that

$$\overline{o_\alpha\, p} \cap \bigcup \mathcal{O}_i \neq \emptyset,$$

since $\overline{o_\alpha\, r} \subset \overline{o_\alpha\, p}$. This implies that $p \notin \mathcal{C}_\alpha$ which is a contradiction. Therefore, r must be visible too. □

Lemma 3. *Given a closed path $\Gamma([0,1]) \subset \mathcal{C}_\alpha$, then the space enclosed by Γ is also in $\mathcal{C}_\alpha$.*

Proof. Let $\mathcal{R}$ be the enclosed area by the path Γ. Since $\Gamma : [0,1] \to \mathbb{R}^2$ is bounded, then

$$\exists M > 0 \text{ such that } ||\Gamma(t) - o_\alpha|| < M,$$

where o_α is the location of camera α. Hence,

$$r \notin \mathcal{R} \text{ if } ||r - o_\alpha|| > M.$$

Also, if a point r' is connected to $r \notin \mathcal{R}$ through a path γ that does not cross Γ, then $r' \notin \mathcal{R}$.

Let $p \in \mathcal{R}$ and define

$$\mathcal{L} := L(p, o_\alpha) \cap \Gamma([0,1])$$

(i.e., points in Γ and in the line passing through p and o_α), then there must be points $q_1, q_2 \in \mathcal{L}$ such that and $p \in \overline{q_1\, q_2}$. Otherwise, there would exist a point $r \in L$ with $||r - o_\alpha|| > M$ (i.e., $r \notin \mathcal{R}$) such that $\overline{r\, p}$ does not intersect $\Gamma([0,1])$. This implies $p \notin \mathcal{R}$ which is a contradiction. Therefore, $p \in \overline{q_1\, q_2}$.

Next, we consider three cases:

- Assume $q_1 \neq o_\alpha$ and $q_2 \neq o_n$. Since $q_1, q_2 \in \Gamma([0,1]) \subset \mathcal{C}_n$ with $L(q_1, o_\alpha) = L(q_2, o_\alpha)$, then $p \in \overline{q_1\, q_2} \subset \mathcal{C}_n$ by lemma 2 (which makes p visible).

- Assume $q_1 \neq o_\alpha$ and $q_2 = o_\alpha$. Then $p \in \overline{o_\alpha q_2} \subset \mathcal{C}_\alpha$ by lemma 1.
- Assume $q_1 = q_1 = o_\alpha$. Then, $p = o_\alpha \in \mathcal{C}_\alpha$.

In all cases p is visible, and since p was arbitrary we conclude that $\mathcal{R}$ is visible. □

The previous lemmas are also true if we replace $\mathcal{C}_\alpha$ by the set $\mathcal{C}_{\alpha,j}$ resulting from a decomposition of the coverage. The reason why it works is because we can think of $\mathcal{C}_{\alpha,j}$ as being the coverage of a camera with a domain

$$\mathcal{D}_{\alpha,j} := \mathcal{D}_\alpha \cap K_{\alpha,j},$$

where $K_{\alpha,j}$ is the corresponding convex cone that generates the region $\mathcal{C}_{\alpha,j}$. This new domain is still convex which is the property used in the previous lemmas. However, note that this $\mathcal{D}_{\alpha,j}$ is not open.

Lemma 4. *Every connected component of $\bigcap_{(\alpha,j)\in A} \mathcal{C}_{\alpha,j}$, where A is a finite set of indices, is simply connected.*

Proof. Let Γ be a closed loop in $\bigcap_{(\alpha,j)\in A} \mathcal{C}_{\alpha,j}$. By the previous lemma, the space enclosed by Γ is inside $\mathcal{C}_{\alpha,j}$ for all $(\alpha, j) \in A$. □

Definition 24. *Let $\Gamma : [0,1] \to \mathbb{R}^2$ be a path connecting points p to q (i.e. $\Gamma(0) = p$ and $\Gamma(1) = q$). We define the* **region enclosed by** Γ*, denoted by $\mathcal{R}(\Gamma)$, to be the region enclosed by the set $\Gamma([0,1]) \cup \overline{p\,q}$.*

Definition 25. *A path $\Gamma : [0,1] \to \mathbb{R}^2$ connecting points p and q is said to be a* **convex path** *if $\mathcal{R}(\Gamma)$ is convex.*

Definition 26. *A non-intersecting path* $\Gamma : [0,1] \to \mathbb{R}^2$ *is* **monotone with respect to camera** α *if for any* $p \in S^1_\alpha$*, where* S^1_α *is the unit circle centered at* o_α*, we have that* $\Gamma([0,1]) \cap L(p, o_\alpha)$ *has a single connected component.*

Lemma 5. *Let* $\mathcal{R}$ *be a bounded convex set contained between the lines* $L(p, o_\alpha)$ *and* $L(q, o_\alpha)$*, where* p *and* $q \in \mathcal{R}$*. Then, either* $\overline{p\,q}$ *is the only path in* $\mathcal{R}$ *joining* p *to* q*, or there are exactly two distinct images of monotone paths connecting* p *to* q *(only intersecting at the end points), which form the boundary of* $\mathcal{R}$*.*

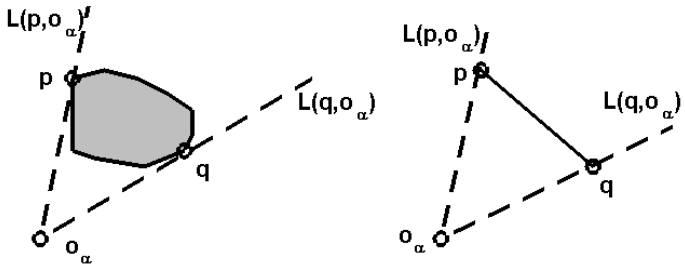

Figure A.1: Cases for lemma 5: Two monotone paths forming the boundary of the set (left). A line segment joining p to q (right)

Figure A.1 illustrates the cases described in the previous lemma.

Lemma 6. *Given that* $\mathcal{C}_\alpha$ *is connected with* p *and* $q \in \mathcal{C}_\alpha$*, then there exists a path* Γ *connecting these points that is convex and monotone with respect to camera* α *with* $\Gamma([0,1]) \subset \mathcal{C}_\alpha \cap \Delta_{p,q,o_\alpha}$*.*

Proof. We present an outline of the proof of this result.

Let p and $q \in \mathcal{C}_\alpha$, where $\mathcal{C}_\alpha$ is connected.

The reader may be tempted to try the path $\overline{p\,o_\alpha} \cup \overline{o_\alpha q}$. However, we are not assuming $o_\alpha \in \mathcal{C}_\alpha$. Our proof takes care of this case too.

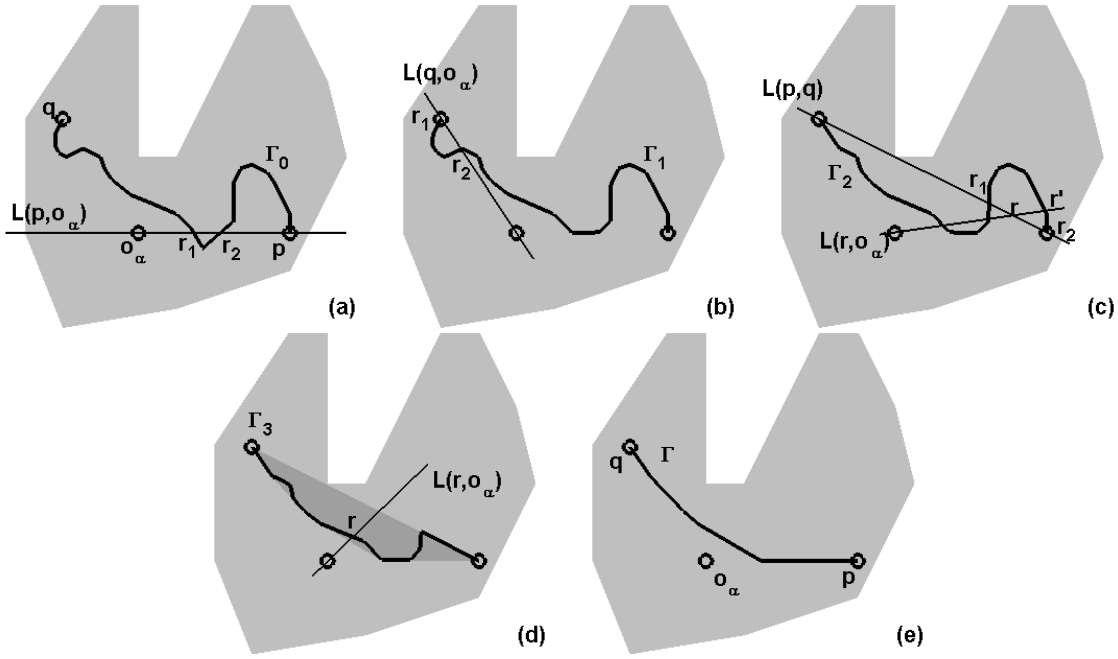

Figure A.2: Construction steps of a monotone convex path for lemma 6.

Since $\mathcal{C}_\alpha$ is connected then there exists a path Γ_0 that connects p to q with $\Gamma_0([0,1]) \subset \mathcal{C}_\alpha$. We illustrate this in the diagram in figure A.2 (a) in which the gray region corresponds to the coverage under consideration.

Our first objective will be to construct a path that is contained within Δ_{p,q,o_α}.

We start with path Γ_0 and consider the line $L(p, o_\alpha)$ (see figure A.2 (a)). This line will intersect the Γ_0 at points $\{r_k\}$. By lemma 2, we know that the line segments between them are visible, so we can construct path Γ_1 (as shown in figure A.2 (b)) which does not cross $L(p, o_\alpha)$.

Next, we consider the intersections between $L(q, o_\alpha)$ and Γ_1 (see figure A.2 (b)). As in the previous case, we can construct a path Γ_2 which does not cross $L(q, o_\alpha)$.

If we consider the line $L(p, q)$, then it will intersect the line Γ_2 at points $\{r_k\}$ (see

figure A.2 (c)). Consider a segment of Γ_2 that is outside of the triangle Δ_{p,q,o_α}, which intersects $L(p,q)$ at r_1 and r_2. For any $r \in \overline{r_1\, r_2}$, we see that $r \in \mathcal{D}_\alpha$ since $r_k \in \mathcal{D}_\alpha$ and $\mathcal{D}_\alpha$ is convex. Also, there exists a point $r' \in L(r, o_\alpha) \cap \Gamma_2([0,1])$ which is further away from o_α than r. Otherwise, the line segment in Γ between r_1 and r_2 would not be outside the Δ_{p,q,o_α}. Therefore, if r was not visible then r' would not be visible which is a contradiction. Hence, r must be visible.

This implies that we can connect r_1 to r_2 by the line segment $\overline{r_1\, r_2}$ and construct path Γ_3 which is inside Δ_{p,q,o_α}.

In order to make Γ_3 into a convex path, we take the convex hull of Γ_3 and by lemma 5 we know that there are at most two monotone paths to choose from (see figure A.2 (d)). We choose the path Γ that is closest to o_α. Clearly Γ is convex. We can see that Γ is visible since for any line $L(r, o_\alpha)$ for $r \in \Gamma_3([0,1])$, the line will have to intersect Γ at some location s closer to o_α than r.

This process yields the desired monotone and convex path Γ (see figure A.2 (e)) which images is in $\mathcal{C}_\alpha \cap \Delta_{p,o_\alpha,q}$. □

Lemma 7. *Given that $\mathcal{C}_\alpha$ is connected with p and $q \in \mathcal{C}_{\alpha,j}$ for some j, then there exists a path Γ connecting these points that is convex and monotone with respect to camera α with $\Gamma([0,1]) \subset \mathcal{C}_{\alpha,j} \cap \Delta_{p,q,o_\alpha}$.*

Proof. Since p and $q \in \mathcal{C}_{\alpha,j}$, then p and $q \in C_\alpha \cap K_{\alpha,j}$. By the previous lemma, we know that there exists a path Γ such that $\Gamma([0,1]) \subset \mathcal{C}_\alpha$. Note that $\Gamma([0,1])$ is inside the cone formed by the lines $L(p, o_\alpha)$ and $L(q, o_\alpha)$ by construction. This cone must be contained within $K_{\alpha,j}$, otherwise p and q could not be in $K_{\alpha,j}$. Therefore, $\Gamma([0,1]) \subset K_{\alpha,j} \cap \mathcal{C}_\alpha = \mathcal{C}_{\alpha,j}$. □

Lemma 8. *Let* $\Gamma : [0,1] \to \mathbb{R}^2$ *be a feasible monotone path connecting* p *and* $q \in \mathcal{C}_\alpha$ *with* $\Gamma([0,1]) \subset \mathcal{D}_\alpha$ *for some camera* α*. If an object* $\mathcal{O}$ *is within the region enclosed by* $\overline{o_\alpha\, p} \cup \Gamma([0,1]) \cup \overline{q\, o_\alpha}$ *then there exists a bisecting line* L *passing through a point in* Γ *that does not intersect* $L(p, o_\alpha)$ *and* $L(q, o_\alpha)$ *(not including these lines).*

Proof. For simplicity we just give an outline of this proof.

Since $\Gamma([0,1]) \subset \mathcal{D}_\alpha$, we know that no point in Γ will be in the boundary of $\mathcal{D}_\alpha$ since $\mathcal{D}_\alpha$ is open.

Since $\overline{o_\alpha\, p} \cup \Gamma([0,1]) \cup \overline{q\, o_\alpha}$ encloses an object, there exists a transition between having a visible and a not-visible point in the path (i.e. an occlusion event). This is guaranteed since at least a point in the path is visible, and not all the points can be visible due to the object $\mathcal{O}$.

Assume that the transition event occurs in $L(p, o_\alpha)$ or $L(q, o_\alpha)$ at some point $r \in \Gamma([0,1])$ and nowhere else. Without loss of generality assume that $\overline{r\, p} \subset \Gamma([0,1])$ (due to monotonicity of path). The object would have to occlude r too (since objects are closed). Then either the path is not feasible or p is not visible which contradicts our assumption. Therefore, a transition must occur at some other point along Γ and not in these lines. □

Theorem 7. (Decomposition Theorem) *Let* $\{\mathcal{C}_\alpha\}_{\alpha=1}^{N}$ *be a collection of camera coverages where each* $\mathcal{C}_\alpha$ *is connected and* N *is the number of cameras in the domain. Let* $\{\mathcal{C}_{\alpha,k}\}_{(\alpha,k)\in A_D}$ *be the collection of decomposed sets by all possible bisecting lines, where* A_D *is the set of indices in the decomposition. Then, any finite intersection* $\bigcap_{(\alpha',k')\in A} \mathcal{C}_{\alpha',k'}$*, where* A *is a finite set of indices, is contractible.*

Proof. For simplicity we just give an outline of this proof for two cameras. The proof for

multiple cameras can be completed by induction.

Let p and $q \in \mathcal{C}_{\alpha_1,k_1} \cap \mathcal{C}_{\alpha_2,k_2}$ for some indices (α_i, k_i).

Part I:

First, consider cameras α_1 and α_2 on the same side of the line $L(p,q)$. We know that there exist convex monotone paths Γ_i connecting p to q such that $\Gamma_i([0,1]) \subset \mathcal{C}_{\alpha_i,k_i} \cap \Delta_{p,o_{\alpha_i},q}$ for $i = 1, 2$ (see left plot in figure A.3).

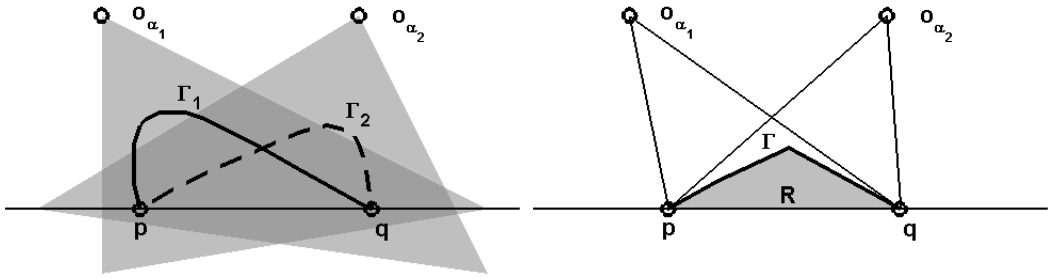

Figure A.3: Illustration for the construction of Γ.

By lemma 5, we can choose a path Γ corresponding to a segment of the boundary of $\mathcal{R} := \mathcal{R}(\Gamma_1) \cap \mathcal{R}(\Gamma_2)$ (see right plot in figure A.3). We choose the path that consists of segments from Γ_1 and Γ_2 so Γ will be feasible. We note that lemma 5 also tells us that Γ is monotone with respect to camera α_i (since $\mathcal{R}$ is between $L(p, o_{\alpha_i})$ and $L(q, o_{\alpha_i})$). Also,

$$\Gamma \subset \mathcal{R} = \mathcal{R}(\Gamma_1) \cap \mathcal{R}(\Gamma_2) \subset \mathcal{D}_{\alpha_1} \cap \mathcal{D}_{\alpha_2}$$

due to convexity of $\mathcal{D}_{\alpha_i}$.

By lemma 8, we know that there are no objects inside the regions enclosed by $\overline{p\, o_{\alpha_i}} \cup \Gamma([0,1]) \cup \overline{q\, o_{\alpha_i}}$ (since otherwise there would be a bisecting line and we assumed that we already decomposed using all bisecting lines). Hence, $\overline{s\, o_{\alpha_i}}$ does not intersect any object for $s \in \Gamma([0,1])$, which implies that Γ is visible by both cameras (i.e. $\Gamma([0,1]) \subset \mathcal{C}_{\alpha_1,k_1} \cap \mathcal{C}_{\alpha_2,k_2}$).

Part II:

Now we consider cameras α_1 and α_2 at opposite sides of the line $L(p, q)$. There are two main cases to consider.

Case 1:

For the first case we consider a configuration as seen in figure A.4 (left).

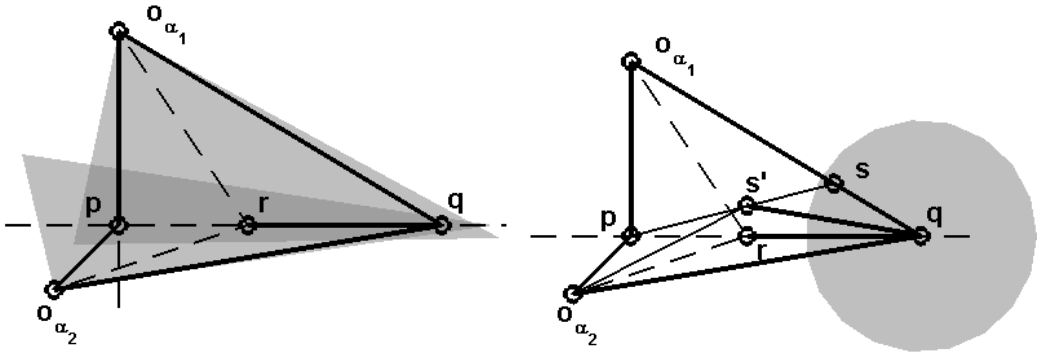

Figure A.4: Illustrations for Case 1.

We would like to conclude that the path $\overline{p\,q}$ is visible by both cameras. Assume that it is only visible up to a point r (not including this point since objects are closed). Then, an object must intersect $\overline{r\,o_{\alpha_1}}$ or $\overline{r\,o_{\alpha_2}}$. By lemma 8, we notice that the interior of $\Delta_{r,o_{\alpha_i},q}$ must be empty; otherwise, there would be a bisecting line.

Also, since $\mathcal{D}_{\alpha_2}$ is open, we can find a ball B around q that is contained in $\mathcal{D}_{\alpha_2}$ (see right plot in figure A.4(right)). We choose a point $s \in B \cap \overline{q\,o_{\alpha_1}}$.

Then, $\Delta_{p,s,q} \subset \mathcal{D}_{\alpha_1} \cap \mathcal{D}_{\alpha_2}$. From there, we can choose a point s' as shown in the diagram such that $\overline{s'\,q} \subset \mathcal{D}_{\alpha_2}$ is feasible. This is possible since there are no objects in $\Delta_{r,o_{\alpha_1},q}$. However, if there was an object intersecting $\overline{r\,o_{\alpha_2}}$, then $L(r, o_{\alpha_2})$ would be a bisecting line. But, it is not. So, there are no objects intersecting $\overline{r\,o_{\alpha_2}}$.

Similarly, there are no objects intersecting $\overline{r\, o_{\alpha_1}}$. This means that r is visible by both cameras, which contradicts our initial assumption. This shows that $\overline{p\, q}$ is visible to both cameras, i.e. $\overline{p\, q} \subset \mathcal{C}_{\alpha_1,k_1} \cap \mathcal{C}_{\alpha_2,k_2}$.

Case 2:

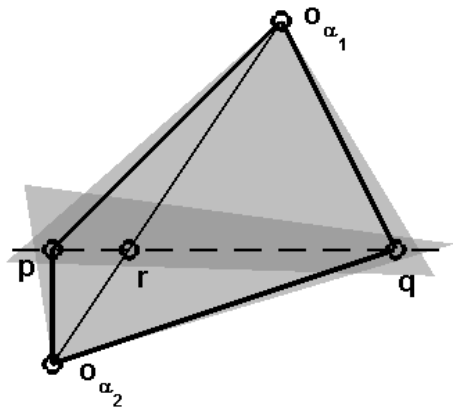

Figure A.5: Illustrations for Case 2.

For the second case, we consider a configuration as shown in figure A.5.

By following the same analysis as before, we can show that $\overline{p\, r} - \{r\}$ and $\overline{r\, q} - \{r\}$ must be visible by both cameras. However, we could have an object in $\overline{o_{\alpha_1}\, o_{\alpha_2}}$. Nevertheless, objects must enclose some area which does not allow an object to be contained in this line. Therefore, $\overline{p\, q} \subset \mathcal{C}_{\alpha_1,k_1} \cap \mathcal{C}_{\alpha_2,k_2}$. □

Appendix B

Proof of Proposition 1

Letting $c_i^t \in B_k = [a_k, b_k)$ for $1 < k < N_p$, we verify the previous claim by noting that the first condition in the definition tells us:

$$\begin{aligned} c_i^{t+1} &> c_i^t - |B_{k-1}| = c_i^t - b_{k-1} + a_{k-1} \\ &> c_i^t - b_k + a_{k-1} \\ &\geq a_{k-1}. \end{aligned} \tag{B.1}$$

The second line in the inequality follows from the fact that $b_{k-1} < b_k$, and the last inequality follows from the fact that $c_i^t \geq b_k$ since $c_i^t \in B_k$. Similarly, we can conclude that:

$$\begin{aligned} c_i^{t+1} &< c_i^t + |B_{k+1}| = c_i^t - a_{k+1} + b_{k+1} \\ &< b_{k+1}. \end{aligned} \tag{B.2}$$

Combining the two previous results, we have $a_{k-1} \leq c_i^{t+1} < b_{k+1}$ which implies that $c_i^{t+1} \in B_{k-1} \cup B_k \cup B_{k+1}$.

Similarly, if $c_i^t \in B_1$ then $c_i^{t+1} \in B_1 \cup B_2$, and if $c_i^t \in B_{N_p}$ then $c_i^{t+1} \in B_{Np-1} \cup B_{Np}$.

Appendix C

Proof of Theorems 5 and 6

C.1 Proof of Theorem 5

In order to simplify the notation, we make the following definition: $\alpha(\mathcal{I}) := \alpha(\mathcal{I}|\mathbb{R}^2)$, $\alpha(\mathcal{I}|r) := \alpha(\mathcal{I}|K_r)$, and corresponding definitions for $\overline{\alpha}$ and γ.

Throughout this proof we make use of the following property.

Property 1. *Given vectors v, v', w and w', where $v \leq v'$, $w' \leq w$, and $v' \prec w'$, then $v \prec w$. We note that the flow that explains v' based on w' also explains v based on w.*

Property 2. *Given vectors v, v', w and w' where $v \prec w$ and $v' \prec w'$, then $v + v' \prec w + w'$. The flow that explains $v + v'$ based on $w + w'$ is the sum of the other flows.*

This theorem is proven by induction. Assume that all images are resulting from Lipschitz image deformation, and $\mathcal{B}$ is a simple partition for all the images. In particular, consider images that are the collection of single connected components sets.

First Step of Induction

Let the collection $\mathcal{I}^t = \{(E_1^t, c_1^t)\}$ be an image sequence, where $c_1^t \in B_p$, $c_1^{t+1} \in B_q$, and E_1^t is a single connected component.

Case 1: $E_1^t \cap K_r = \emptyset$

Then, $\overline{\alpha}(\mathcal{I}^t|r) = 0$ and $0 \leq \alpha(\mathcal{I}^{t+1}|r + C)$. Hence, zero flow gives

$$\overline{\alpha}(\mathcal{I}^t|r) \prec \alpha(\mathcal{I}^{t+1}|r + C).$$

Case 2: $E_1^t \cap K_r \neq \emptyset$

Then, $\overline{\alpha}(\mathcal{I}^t|r) = e_p$, where e_p is the standard basis vector with p-th entry equal to 1 and all other entries equal to 0. This is true by the following reasoning: If $E_1^t \cap K_r$ consists of multiple components, they will have to intersect the boundary since E_1^t is a single connected component.

Since $E_1^t \cap K_r \neq \emptyset$ then $E_1^{t+1} \cap K_{r+C} \neq \emptyset$ due to Lipschitz condition of image deformation. That is, the intersection with E_1^{t+1} will contain one or more connected components. So, $e_q \leq \alpha(\mathcal{I}^{t+1}|r + C)$.

Finally, there is a flow that explains $\alpha(\mathcal{I}^t) = e_p$ based on $\alpha(\mathcal{I}^{t+1}) = e_q$ since $\mathcal{B}$ is a simple partition which implies that p and q are adjacent bins. Then

$$\overline{\alpha}(\mathcal{I}^t|r) = e_p \prec e_q \leq \alpha(\mathcal{I}^{t+1}|r + C)$$

which, by property 1, implies

$$\overline{\alpha}(\mathcal{I}^t|r) \prec \alpha(\mathcal{I}^{t+1}|r + C).$$

k + 1-th Step of Induction

Let the collection $\mathcal{I}^t = \{(E_i^t, c_i^t)\}_{i=1}^{k+1}$ be an image sequence, where $c_{k+1}^t \in B_p$, $c_{k+1}^{t+1} \in B_q$, and all E_i^t consist of single connected components. Also, define the images $\mathcal{J}^t = \{(E_i^t, c_i^t)\}_{i=1}^{k}$. We assume that the desired condition is satisfied by $\mathcal{J}^t$.

Case 1: $E_{k+1}^t \cap K_r = \emptyset$

Then, $\overline{\alpha}(\mathcal{I}^t|r) = \overline{\alpha}(\mathcal{J}^t|r)$. Since adding new sets to an image can only increase the number of connected components, we have $\alpha(\mathcal{I}^{t+1}|r+C) \geq \alpha(\mathcal{J}^{t+1}|r+C)$. Then, by assumption, we have

$$\overline{\alpha}(\mathcal{I}^t|r) = \overline{\alpha}(\mathcal{J}^t|r) \prec \alpha(\mathcal{J}^{t+1}|r+C) \leq \alpha(\mathcal{I}^{t+1}|r+C)$$

which implies

$$\overline{\alpha}(\mathcal{I}^t|r) \prec \alpha(\mathcal{I}^{t+1}|r+C).$$

Case 2: $E_{k+1}^t \cap K_r \neq \emptyset$

Then, $\overline{\alpha}(\mathcal{I}^t|r) \leq \overline{\alpha}(\mathcal{J}^t|r) + e_p$. This is true by the following reasoning: If $E_{k+1}^t \cap K_r$ does not intersect the boundary then $\overline{\alpha}(\mathcal{I}^t|r) = \overline{\alpha}(\mathcal{J}^t|r) + e_p$; also, if $E_{k+1}^t \cap K_r$ does intersect the boundary but it does not have the same color as any other sets $E_i^t \cap K_r$ that intersect the boundary then $\overline{\alpha}(\mathcal{I}^t|r) = \overline{\alpha}(\mathcal{J}^t|r) + e_p$; othewise, we have $\overline{\alpha}(\mathcal{I}^t|r) = \overline{\alpha}(\mathcal{J}^t|r)$.

It can be also verified that

$$\alpha(\mathcal{J}^{t+1}|r+C) + e_q \leq \alpha(\mathcal{I}^{t+1}|r+C),$$

since $E_i^{t+1} \cap K_{r+C}$ will add at least one more connected component to the histogram count.

As before, $e_p \prec e_q$ since we have a simple partition. It is also known that $\overline{\alpha}(\mathcal{J}^t|r) \prec \alpha(\mathcal{J}^{t+1}|r+c)$ which, by property 2, implies

$$\overline{\alpha}(\mathcal{J}^t|r) + e_p \prec \alpha(\mathcal{J}^{t+1}|r+c) + e_q.$$

Finally, since $\overline{\alpha}(\mathcal{I}^t|r) \leq \overline{\alpha}(\mathcal{J}^t|r) + e_p$ and $\alpha(\mathcal{J}^{t+1}|r+C) + e_q \leq \alpha(\mathcal{I}^{t+1}|r+C)$ we have, by property 1,

$$\overline{\alpha}(\mathcal{I}^t|r) \prec \alpha(\mathcal{I}^{t+1}|r+C).$$

This concludes the proof of equation 7.9.

C.2 Proof of Theorem 6

In order to show that

$$\gamma(\mathcal{I}^t|K_r) \prec \gamma(\mathcal{I}^{t+1}|K_{r+C}),$$

we first consider $\mathcal{I}^t = \{E_1^t, c_1^t\}$. It is clear that any hole in $E_1^t \cap K_r$ will have to be in $E_1^{t+1} \cap K_{r+C}$. Hence, $\gamma(\mathcal{I}^t|K_r) = n_h e_p$ and $\gamma(\mathcal{I}^{t+1}|K_{r+C}) \geq n_h e_q$ for some p and q, n_h is the number of holes in $E_1^t \cap K_r$, and $e_p \prec e_q$. The general proof follows by induction as before.