Camera Network Topology Estimation Using Sparse Methods

Francisco Miguel Borges Morais Lourenço

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Alexandre José Malheiro Bernardino
Prof. Jorge Dos Santos Salvador Marques

November 2018

Declaration

Acknowledgments

This work was partially funded by FCT under the project PTDC/EEI-PRO/ 0426/2014.

I would like to thank my supervisors, professors Alexandre Bernardino and Jorge Salvador Marques, for the support they gave me. I would also like to thank my friends and family.

Resumo

O conhecimento da estrutura topológica de uma rede de câmaras pode facilitar tarefas como a re-identificação de pessoas ao limitar as possíveis correspondências para cada pessoa observada. Em situações onde não é prático determinar manualmente a disposição das câmaras, a estimação automática da topologia de uma rede de câmaras pode ser utilizada para descobrir as ligações entre as áreas observadas pelas diferentes câmaras.

Este trabalho propõe dois métodos para descobrir a topologia de uma rede de câmaras sem sobreposição baseados na estimação de matrizes de covariância inversa esparsas. Utilizando a estimação de matrizes de covariância inversa esparsas, uma matriz de covariância empírica obtida a partir de dados observados pode ser utilizada para descobrir um modelo probabilístico gráfico esparso que representa a estrutura de dependências condicionais entre os seus nós. O primeiro método estabelece correspondências directas entre observações da mesma pessoa em diferentes câmaras através de *thresholding* baseado na cor, enquanto que o segundo método estima um grafo representativo das ligações entre eventos baseado nas características observadas sem estabelecer correspondências directas.

São apresentados resultados experimentais favoráveis para ambos os métodos, utilizando dados de trajectórias geradas aleatóriamente através de um simulador de eventos. Foi adicionado ruído sintético aos descritores de cor gerados pelo simulador para testar a robustez dos métodos desenvolvidos face a dados com ruído.

Palavras-chave: aprendizagem de grafos, topologia, redes multi-câmara

Abstract

Knowledge of the topological structure of a camera network can help with tasks such as person re-identification by limiting the correspondence possibilities for each observed person. In situations where manually determining the camera layout is not practical, automated camera network topology estimation can be used to discover the connections between the different camera views.

This work proposes two methods for discovering the topology of a non-overlapping camera network based on sparse inverse covariance estimation. Using sparse inverse covariance estimation an empirical covariance matrix obtained from observed feature data can be used to discover a sparse graphical model representing the conditional dependence structure between its nodes. The first method establishes direct correspondences between appearances of the same person in different cameras via color based thresholding, whereas the second method estimates a graph representing connections between events based on observed features without directly establishing correspondences.

Favorable experimental results are shown for both methods using randomly generated trajectory data obtained from an event simulator. Synthetic noise was added to the generated color descriptors to test the methods' robustness to noisy data.

Keywords: graph learning, topology, multi-camera network

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Video surveillance is increasingly common nowadays [1], being used in both public and private environments such as offices, hospitals, homes and even outdoor spaces like roads. In many large environments a network composed of multiple cameras is necessary for an effective surveillance [2]. One of the most important aspects of surveillance in a network of cameras is the task of following targets across multiple camera views.

In order to properly follow a target through the various cameras composing the network, especially cameras with non-overlapping fields of view, it can be helpful to know the spatial relations between them. While precise metric information of each camera's position can be useful, it is often enough to find a topological representation of the camera network map. This topological map can take the form of a graph where each vertex represents either a camera or an entry/exit zone of a camera's field of view. This latter option allows for a better understanding of the network, especially in more closed indoor areas where different hallways and doorways in the same camera view can lead to different places. The graph edges represent connections between these zones - if two vertices are connected by an edge in the graph, then there is an off-camera path between the two zones and it is possible for objects that disappear in an entry/exit zone associated with one of the two vertices to appear in the zone associated with the other vertex.

This knowledge of the topological network map can be used to help with the task of object re-identification. Object re-identification consists in identifying the same object in images taken by different cameras [3], and it is an essential step in following a target across multiple non-overlapping cameras in a network. By knowing the possible off-camera paths connecting the different camera views the task of re-identification can be made significantly easier, as for each object sighted in a camera the list of previously sighted objects it can correspond to is limited to objects that have disappeared through an entry/exit zone connected to the camera the current object is being sighted in.

While the topological layout of a camera network can be determined manually, for larger networks or networks where the layout can be changed often it can be impractical to do so. Therefore, it can be useful to devise a method that allows for an automated inference of the topological network map.

1.2 Objectives

The goal of this work is to develop, implement and evaluate a camera network topology inference method based on sparse inverse covariance estimation. We will be focusing on finding graphs representing the off-camera paths in a network of non-overlapping cameras. The camera network graphs are simple unweighted undirected graphs where nodes represent specific zones in the cameras' field of view from which people appear and disappear, and edges connecting two nodes indicate that people who disappear from one zone may appear in the other.

1.3 Contributions

The main contributions of this work are the two developed methods for the automated estimation of the topology of a camera network. While based on an existing graph estimation method, the two different applications of this method to the specific problem of camera network topology discovery represent novel contributions.

A method to generate random Black-Value-Tint (BVT) histogram color descriptors was also developed for the trajectory simulator [4] used to generate the test data. These histograms are generated according to the characteristics of the BVT histogram (detailed in section 5.1) and constitute a more complex descriptor than the 2-dimensional feature vector originally developed for the simulator.

During this work the footage collected with one of the yet unlabeled cameras for the HDA+ dataset [5] was fully labeled with manually assigned bounding boxes identifying all the observed people.

1.4 Thesis Outline

Chapter 2 contains the state of the art, where previous work on different approaches to the problem of camera network topology estimation is reviewed. Chapter 3 provides an overlook of the topology estimation problem as well as mathematical methods used to solve it. In chapter 4 the two camera network topology estimation methods proposed in this work are explained. Chapter 5 details the experiments run to test and evaluate the implemented methods, including an overview of the dataset used for these tests. In chapter 6 the results of the testing are presented and analyzed. Chapter 7 provides the conclusions taken from this work as well as possible future work.

Chapter 2

State of the art

The problem of camera network topology estimation has been approached in several different ways. Kamenetsky [6] divides this problem into two main types: overlapping and non-overlapping. In overlapping camera networks the fields of view of the different cameras overlap, partially observing the same area. In the topology graphs for these problems two cameras are linked by an edge if their fields of view overlap, forming a *vision graph*. In non-overlapping problems different cameras observe different parts of the environment, without any overlap in their fields of view. In these cases the relationships between the different cameras are represented by a *communication graph*. These are undirected graphs where each node represents one camera and an edge connecting two nodes indicates that transitions between the two cameras are possible. These edges can also have weights corresponding to transition probabilities and times. Our work is focused on these non-overlapping camera networks, where the graph nodes represent specific zones through which people enter and exit the camera's field of view.

In addition to this distinction, Li et al. [7] state that non-overlapping problems can be approached with correspondence based methods and correspondence-free methods. Correspondence based methods require direct correspondences between people sighted in different cameras, that is, they need to identify the same person in different camera views. This can be done manually or through an automatic re-identification method. Correspondence-free methods have no such requirements. The method proposed by the authors in [7] belongs to this category. First, the entry and exit zones of the different cameras are identified using a clustering algorithm. Then, the transition time distributions between the various nodes are computed by accumulated cross correlation and Gaussian fitting. Finally, the mutual information is used to refine the final estimated topology by eliminating connections between pairs of nodes that have a mutual information value below a chosen threshold. This method was tested with both simulated and real data in small camera networks with four and five nodes respectively. In addition to the existence of connections between nodes, the parameters of the transition time distributions obtained by fitting a normal distribution to the cross correlation data were also compared to the temporal distributions of the test data to evaluate the estimated distributions.

Another correspondence-free method is proposed by Makris et al. [8]. For this method the entry and exit zones of the different cameras are learned automatically from observed trajectories using an expectation-maximization algorithm. The graphs obtained through this method include both visible and invisible connections, that is, connections between nodes that belong to the same camera and connections between nodes belonging to different cameras. The visible connections are learned through a standard on-camera object tracking algorithm. An additional virtual node is used to represent entries from and exits to the outside of the network. Invisible connections are obtained from the cross-correlation data for each pair of nodes. Peaks in the cross-correlation above a chosen threshold are used to identify valid links. For each link in the final graph, the most common transition time among all the trajectories between the two nodes connected by that edge is used as that link's transition time. Depending on the the transition time associated with each of these links, they may be classified as invisible connections (for positive transition times), additional visible connections (for negative transition times) or overlapping entry/exit zones (for transition times of approximately 0). This method was tested with a dataset collected from a six camera network with 30466 trajectories.

Cho et al. [9] propose a correspondence based iterative method to estimate camera network topology from person re-identification data. First, a random forest classifier is used to for person re-identification within a set time window. Correspondences for each person that disappears from a camera are found in other cameras. Only highly reliable correspondences (correspondences with a high similarity score) are taken into account. The time differences of these correspondences are used to make a normalized histogram for each pair of cameras, representing the transition distribution. The resulting histograms are fitted to a normal distribution, and two cameras are considered connected if the variance and fitting error of the distribution are low enough. A similar process is then used to find connections between specific manually identified entry and exit zones belonging to connected cameras. When a person disappears at an exit zone in one camera, correspondences are searched within a set time frame in all entry zones in connected cameras. As before, highly reliable correspondences are used to make a histogram of the time between exit from one zone and entrance in another for each pair of exit and entry zones in different cameras. A normal distribution is fitted to each histogram and if the resulting variance and fitting error are low enough the two zones are considered to be connected. This results in an initial topological map of the camera network. The re-identification process is then repeated based on the inferred network topology. A new time frame for the correspondence search is chosen based on the estimated transition distributions and correspondences are only searched within entry zones connected to the exit zone in question. This new re-identification data is used to update the topology estimation, and these two processes are repeated until the estimated topology converges. The topology estimation was evaluated by computing the average Bhattacharyya distance between the estimated transition distributions and those obtained from the ground truth in a dataset provided by the authors, containing 2632 people in a space observed by 9 cameras.

In this work we propose two camera network estimation methods focused on discovering an undirected graph of invisible connections between specific manually determined entry and exit zones belonging to non-overlapping cameras. An edge connecting two entry/exit zones indicates that a person disappearing from one of those zones may reappear in the other, and no transition probabilities or time distributions are computed. The first method is correspondence based, while the second is a correspondence-free method. Both methods use a sparse graph learning algorithm to solve the topology estimation problem by using observed feature data to discover the graph that represents the camera network. This work builds upon the work developed by Silva [4] by using the same tools to implement and test more complex and realistic graph estimation methods, rather than the simple method based on pre-established correspondences and using a two-dimensional feature vector to characterize the simulated data.

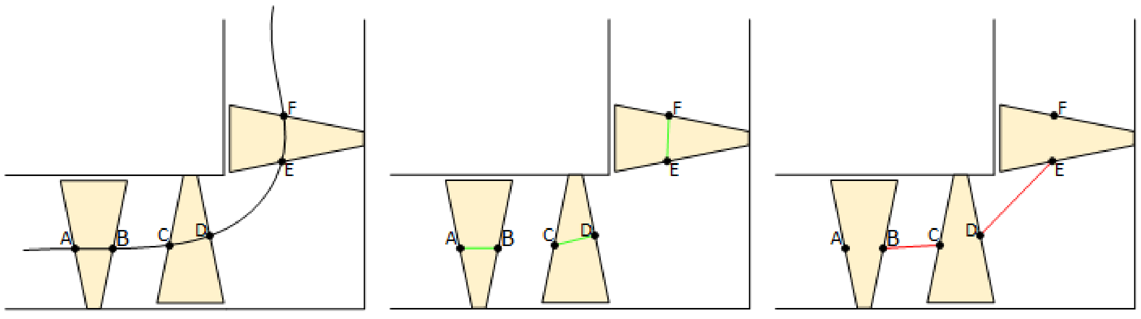
Chapter 3

Graph estimation problem

3.1 Problem definition

As people move through the area observed by the network of non-overlapping cameras, they will either be on-camera (within a camera's field of view) or off-camera (outside the field of view of every camera). The transition between the on-camera and off-camera areas is done through entry and exit zones, which are the parts of a camera's field of view from which people appear and disappear. All of these zones are simultaneously entry zones and exit zones, as any zone through which people can enter a camera's field of view can also be used to exit it. These entry/exit zones will serve as the nodes in the camera network graphs we want to estimate, and these graphs will represent the connections between the different entry/exit zones - an edge connecting two zones indicates that a person can travel between those two zones without passing through any other.

We can divide each trip made by a person into its on-camera (*visible*) and off-camera (*hidden* or *invisible*) portions, as can be seen in figure 3.1.



(a) Example trip (b) Visible paths (c) Hidden paths

Figure 3.1: Example of a path travelled by a person as well as its visible and hidden portions. The areas in yellow represent the fields of view of three different non-overlapping cameras in a camera network.

The points A, B, C, E and F are entry/exit zones of the fields of view of three cameras in a network. The example trip shown in figure 3.1(a) passes through all of these zones and can be divided into five sequential segments: $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$ and $E \rightarrow F$. Segments where both the beginning and the end nodes belong to the same camera's field of view ($A \rightarrow B$, $C \rightarrow D$ and $E \rightarrow F$)

form visible paths, while segments where the two nodes are in the fields of view of different cameras ($B \rightarrow C$ and $D \rightarrow E$) form hidden paths. Hidden paths are of particular interest as they are not visible in the images captured by the cameras that form the network, and therefore are more difficult to identify. Our goal is to find the graph that represents all the hidden paths in a camera network, that is, all the paths that can be travelled off-camera. These graphs are undirected, as all paths can be travelled both ways - if a person can travel from node A to node B, they can also travel from node B to node A.

These hidden path graphs will be found based on entry and exit events. These events correspond to a person entering or exiting a camera's field of view. Associated with each entry/exit event is the entry/exit zone through which it happened, the instant when it happened and a descriptor of the person involved in the event. This information can be used to match different entry/exit events, forming trips travelled by people in the camera network and thereby identifying possible paths. Unlike with entry/exit zones, the distinction between entry and exit events is important, as entry events can not be matched with other entry events, or exit events with other exit events.

As the descriptors used to identify events associated with the same person are obtained from the camera images, they are non-ideal. Even ignoring issues such as noise, the same person in two different images can have different characteristics that result in different descriptors. For example, if using a color descriptor such as a simple color histogram, depending on the lighting conditions and the properties of different cameras the observed colors can be different for the same person. Moreover, even the angle a person is seen from in an image can alter the resulting descriptor, as different sides of their clothing can have different colors. This means that a graph estimation method must be robust to these differences in the descriptors in order to correctly infer the hidden path graphs. Our goal then is to develop a method that can estimate adequate graphs with non-ideal data.

3.2 Graph estimation method

An undirected weighted graph can be fully represented by its Laplacian matrix [10]. The Laplacian matrix $\Delta = [\Delta_{ij}]$ of an undirected weighted graph G with n nodes is given by

$$\Delta_{ij} = \begin{cases} -w_{ij}, \ i \neq j \\ \sum\limits_{k} w_{ik}, \ i = j \end{cases} , \ i,j = 1,...,n, \tag{3.1}$$

where w_{ij} is the weight of the graph edge connecting nodes i and j. These weights are nonnegative (that is, $w_{ij} \geq 0 \ \forall i,j = 1,...,n$) and it is considered that if two nodes i and j are not directly connected by an edge in the graph, the weight w_{ij} between the two nodes is 0. As the diagonal of the Laplacian contains the weighted degree of the corresponding nodes and can be obtained from the non diagonal entries of Δ, the graph can be fully represented by these non diagonal entries alone. Therefore, it is sufficient to estimate the weighted adjacency matrix $W = [w_{ij}]$ with $i,j = 1,...,n$. This matrix is symmetric as the graph is undirected, and it is considered that $w_{ii} = 0$ for all $i = 1,...n$.

The weighted adjacency matrix W can be estimated as described by Lake & Tenenbaum [11] by defining the following *maximum a posteriori* (MAP) estimate:

$$\begin{aligned}\hat{W} &= \underset{W}{\text{argmax}}\ p(W|D) \\ &= \underset{W}{\text{argmax}}\ p(D|W) \cdot p(W).\end{aligned} \tag{3.2}$$

The matrix D is an $n \times m$ feature matrix with each of the m columns representing a feature vector $f^{(k)} = (f_1^{(k)}, ..., f_n^{(k)})^T$ where $f_i^{(k)}$ represents the value of feature k pertaining to node i. These features are assumed to be independent and identically distributed draws from a normal distribution

$$\begin{aligned}p(f^{(k)}|W) = (2\pi)^{-\frac{n}{2}} |\tilde{\Delta}^{-1}|^{-\frac{1}{2}} \cdot \\ \cdot \exp\left(-\frac{1}{4}\sum_{i,j} w_{ij}\left(f_i^{(k)} - f_j^{(k)}\right)^2 - \frac{1}{2}\sum_i \frac{\left(f_i^{(k)}\right)^2}{\sigma_i^2}\right).\end{aligned} \tag{3.3}$$

This is equivalent to $p(f^{(k)}|W) \sim N(0, \tilde{\Delta}^{-1})$, where $\tilde{\Delta}$ is a regularized graph Laplacian given by $\tilde{\Delta} = \Delta + \text{diag}\left(1/{\sigma_i}^2\right)$, with $\text{diag}\left(1/{\sigma_i}^2\right)$ being a diagonal matrix containing n regularization terms. This regularization of the graph Laplacian matrix is important as the unregularized matrix is singular [12].

The term $-\frac{1}{4}\sum_{i,j} w_{ij}(f_i^{(k)} - f_j^{(k)})^2$ imposes a form of feature smoothness, as two nodes i and j connected by a large weight w_{ij} must have similar values of $f^{(k)}$ to maximize the value of the likelihood function $p(f^{(k)}|W)$. As the various features $f^{(k)}$ are assumed to be independent, the likelihood function pertaining to the full feature set D can be defined as

$$p(D|W) = \prod_{k=1}^{m} p(f^{(k)}|W). \tag{3.4}$$

Assuming each weight w_{ij} is independently drawn from an exponential distribution $p(w_{ij}) \sim \text{Exponential}(\beta)$, the graph weight prior can be defined as

$$p(W) = \prod_{1 \le i < j \le n} p(w_{ij}) = \prod_{1 \le i < j \le n} \beta e^{-\beta w_{ij}}. \tag{3.5}$$

This prior results in sparse graphs by encouraging weights close to zero due to the nature of the exponential distribution.

The MAP estimate of W can then be written as

$$\begin{aligned}\hat{W} &= \underset{W}{\text{argmax}}\ p(D|W) \cdot p(W) \\ &= \underset{W}{\text{argmax}}\ \prod_{k=1}^{m} p(f^{(k)}|W) \cdot \prod_{1 \le i < j \le n} p(w_{ij}).\end{aligned} \tag{3.6}$$

Using the result from (3.3) we have

$$\begin{aligned}\prod_{k\ =1}^{m} p(f^{(k)}|W) = (2\pi)^{-\frac{nm}{2}} |\tilde{\Delta}^{-1}|^{-\frac{1}{2}m} \cdot \\ \cdot \exp\left(\sum_{k=1}^{m}\left(-\frac{1}{4}\sum_{i,j}^{n} w_{ij}\left(f_i^{(k)} - f_j^{(k)}\right)^2 - \frac{1}{2}\sum_{i=1}^{n}\frac{\left(f_i^{(k)}\right)^2}{\sigma_i^2}\right)\right),\end{aligned} \tag{3.7}$$

which gives us

$$\hat{W} = \underset{W}{\text{argmax}} \ (2\pi)^{-\frac{nm}{2}} |\tilde{\Delta}^{-1}|^{-\frac{1}{2}m} \cdot \\ \cdot \exp\left(\sum_{k=1}^{m} \left(-\frac{1}{4} \sum_{i,j}^{n} w_{ij} \left(f_i^{(k)} - f_j^{(k)} \right)^2 - \frac{1}{2} \sum_{i=1}^{n} \frac{\left(f_i^{(k)} \right)^2}{\sigma_i^2} \right) \right) \\ \cdot \prod_{1 \leq i < j \leq n} \beta e^{-\beta w_{ij}}. \tag{3.8}$$

Maximizing the expression in (3.8) is equivalent to maximizing its logarithm, as the logarithm is a monotonically increasing function. Therefore, we can solve

$$\hat{W} = \underset{W}{\text{argmax}} \ -\frac{nm}{2}\log 2\pi - \frac{m}{2}\log|\tilde{\Delta}^{-1}| - \frac{1}{2}\operatorname{Tr}\left(\tilde{\Delta}DD^T\right) + \frac{n(n-1)}{2}\log\beta - \frac{\beta}{2}||W||_1, \tag{3.9}$$

with $||W||_1 = \sum_{i,j=1}^{n} |w_{ij}|$.

As we are only interested in finding the matrix W that maximizes (3.9), we can ignore constants that do not depend on W, resulting in the final expression

$$\hat{W} = \underset{W}{\text{argmax}} \ \log|\tilde{\Delta}| - \frac{1}{m}\operatorname{Tr}\left(\tilde{\Delta}DD^T\right) - \frac{\beta}{m}||W||_1. \tag{3.10}$$

This MAP estimation with a Gaussian likelihood function and an exponential prior can also be seen as a Gaussian maximum likelihood estimation with an ℓ_1-norm penalty term similar to the commonly used Lasso regularization [13]. The penalty parameter β controls the trade-off between the likelihood of the data in matrix D and the sparsity inducing penalty term, with larger values of β encouraging sparser graphs in detriment of the likelihood.

The relationship between the regularized graph Laplacian $\tilde{\Delta}$ and the weight matrix W can be written as $\tilde{\Delta} = -W + \text{diag}\,(q_i) + \text{diag}\,(1/{\sigma_i}^2)$, where diag (q_i) is a diagonal matrix with each entry being $q_i = \sum_j w_{ij}$. As $\tilde{\Delta}$ can be easily obtained from W, discovering $\hat{W}$ is essentially equivalent to discovering $\hat{\tilde{\Delta}}$ and therefore solving (3.10) corresponds to estimating the precision or inverse covariance matrix $\tilde{\Delta}$ of the normal distribution $p(f^{(k)}|W) \sim N(0, \tilde{\Delta}^{-1})$. If this distribution describes a Gaussian Markov random field [14] then zero entries in its precision matrix indicate conditional independence between variables - if $\tilde{\Delta}_{ij} = 0$ for $i \neq j$ then $f_i^{(k)}$ and $f_j^{(k)}$ are conditionally independent given all other variables $\left\{ f_a^{(k)} : a \neq i \wedge a \neq j \right\}$. Therefore by using the empirical covariance matrix $\frac{1}{m}DD^T$ obtained from m samples of $p(f^{(k)}|W) \sim N(0, \tilde{\Delta}^{-1})$ we can estimate a sparse precision matrix that defines a graphical model representing the dependencies between the different nodes. This graphical model is the graph we are estimating by solving (3.10), with the absence of an edge connecting two nodes indicating their conditional independence. Therefore this problem can also be seen as the estimation of a sparse Gaussian Markov random field by computing its precision matrix from empirical covariance data.

The final optimization problem (3.10) can be solved using a generic optimization solver. Different coordinate descent methods [15, 16, 17, 18] have also been developed to solve optimization problems of this type.

Chapter 4

Proposed methods

The goal is to estimate a graph representing the hidden paths, i.e., the paths not directly observable by the camera network. This corresponds to paths travelled by a person between their exit from a camera's entry/exit node and their entry into another camera's entry/exit node.

In order to estimate the camera network topology two different approaches were developed, based on the graph estimation method detailed in section 3.2. Section 4.1 describes a pairwise method that identifies correspondences between pairs of events. In section 4.2 a correspondence-free method that estimates a single graph representing the relations between all observed events is detailed.

4.1 Pairwise method

In the first method specific trips made by a person between two nodes were identified and the color descriptors of the person at the entry and exit from a camera's field of view were used as corresponding features for each of the two nodes. This is done by considering only entry and exit events, denoted by e_+ and e_- respectively. The algorithm starts by iterating over the various exit events, which are the start points of the hidden trips. For each exit event e^i_-, the corresponding entry event $a(e^i_-)$ that completes the trip must be found. First, the set $\hat{S}^i_+ = \left\{e^j_+ : d_{BHATT}(f(e^i_-), f(e^j_+)) < \epsilon_{color},\ t(e^j_+) - t(e^i_-) > 0\right\}$ containing all the entry events pertaining to what can be considered the same person and taking place after the chosen exit event is found. This is done by selecting only entry events where a distance metric $d_{BHATT}(f(e^i_-), f(e^j_+))$ based on a modified Bhattacharya coefficient [19] between the color descriptors $f(e^i_-)$ and $f(e^i_+)$ is below a certain similarity threshold ϵ_{color}. This accounts for small differences in color, but it is assumed that the color descriptors of the same person in different events will be mostly similar. Then, the end of the trip is chosen as

$$a(e^i_-) = \underset{e^j_+ \in \hat{S}^i_+}{\text{argmin}} \quad t(e^j_+) - t(e^i_-), \tag{4.1}$$

where $t(e^i)$ is the instant when event e^i takes place.

If no entry events are found within the acceptable threshold and $\hat{S}^i_+ = \emptyset$, no path is considered and the chosen exit event e^i_- is ignored.

Once the trips are identified, a feature matrix D as described in section 3.2 is built for each pair of nodes using the color descriptors associated with the exit and entry events of all trips between the two nodes. As only two nodes are being considered, each feature matrix is a $2 \times pq$ matrix of the form $D = \begin{bmatrix} \mathbf{f_1^1} & \mathbf{f_1^2} & \dots & \mathbf{f_1^p} \\ \mathbf{f_2^1} & \mathbf{f_2^2} & \dots & \mathbf{f_2^p} \end{bmatrix}$, where $\mathbf{f_i^j}$ is a row vector of length q containing the color descriptor associated with the entry/exit event of the jth trip at the ith node of the trip, and p is the total number of trips between the two nodes.

These feature matrices are then used to compute the weight matrix W as described in section 3.2. The optimization problem defined in (3.10) was solved using CVX, a package for specifying and solving convex programs [20, 21], using the code shown in appendix A. This process is memory intensive but can be applied in this case due to the reduced size of the graphs being estimated. As only two nodes are being considered at a time, this corresponds to computing a single weight w_{ij}, which is the weight of the connection between nodes i and j, for each pair of nodes. In effect this means that rather than using the event data to estimate a single graph representing the full camera network, one graph is inferred for each pair of nodes, which consists in determining whether or not the nodes are connected by an edge. The final camera network graph is the union of all the graphs obtained through this method.

While this approach does not make use of all the properties of the sparse inverse covariance estimation due to considering each pair of nodes individually, the sparsity penalty does provide an advantage over using a more typical distance metric to compare the color descriptors associated with two nodes by avoiding the need for further thresholding, as the penalty term $\frac{\beta}{m}||W||_1$ in expression (3.10) helps eliminate weights connecting dissimilar nodes.

4.2 Event graph method

Unlike the method described in section 4.1, the second approach does not need to establish one-to-one correspondences between events. Rather than identifying events belonging to the same trip, each entry or exit event is used as a node to estimate a graph representing connections between events. The data pertaining to all the observed events is stored in a $2n \times (q+1)$ matrix of the form

$$D = \begin{bmatrix} \mathbf{f}(e^1_+) & \tau t(e^1_+) \\ \mathbf{f}(e^1_-) & \tau t(e^1_-) \\ \mathbf{f}(e^2_+) & \tau t(e^2_+) \\ \mathbf{f}(e^2_-) & \tau t(e^2_-) \\ \dots & \\ \mathbf{f}(e^n_+) & \tau t(e^n_+) \\ \mathbf{f}(e^n_-) & \tau t(e^n_-) \end{bmatrix},$$

where $\mathbf{f}(e^i_+)$ is a row vector of length q containing the color descriptor associated with the ith entry event and $\mathbf{f}(e^i_-)$ is a row vector of length q containing the color descriptor associated with the ith exit event. It is important to note that two events e^i_+ and e^i_- are not necessarily corresponding entry and exit events, they are simply the ith entry event to be recorded and the ith exit event to be recorded, respectively. Maintaining this ordering of the matrix rows is not necessary to obtain the event graph, but having a consistent order facilitates the treatment of the resulting data. The instant when each event takes place is used as an additional feature. These time values are multiplied by a parameter τ in order to control the trade-off between the weights of the color descriptors and the moments when the events take place, with larger values of τ placing a larger emphasis on the temporal factor over the color descriptors in the trade-off. Two corresponding entry and exit events (events belonging to the same trip) should have similar color descriptors, and it is considered that most trips should be relatively short. Therefore, the value of τ should be such that among events with similar color descriptors those that are closer in time should be more likely to be connected in the final graph but with the color similarity being the more important factor.

The feature matrix D is then used to estimate a weight matrix W^{ev} representing the graph of all events $G^{ev} = (V(G^{ev}), E(G^{ev}))$ as described in section 3.2 using the R package *dpglasso* [16, 22] with the code shown in appendix B, as, unlike in section 4.1 the size of the graph being estimated makes using CVX impossible due to memory constraints. The DP-Glasso algorithm improves upon the original Graphical Lasso [15] by solving potential convergence issues. In the resulting graph each vertex represents an event, with an edge between two vertices indicating that both events form a trip. In order to obtain the final graph $G^{net} = (V(G^{net}), E(G^{net}))$ representing the camera network, a new matrix W^{net} is initialized with every entry w_{ij}^{net} set to 0. For each edge $\{u, v\} \in E(G^{ev})$ the entry/exit zones z_a and z_b associated with the events represented by vertices u and v are identified, and the absolute value of the weight w_{uv}^{ev} of edge $\{u, v\}$ is added to entry w_{ab}^{net} of matrix W^{net}. The edge set of the final camera network graph is given by $E(G^{net}) = \left\{\{u, v\} \in (V(G^{net}))^2 : u \neq v, w_{uv}^{net} > \epsilon_{thr}\right\}$. The threshold ϵ_{thr} is important to remove false positives from the final graph. As the original graph being estimated is a graph of single events, a single false positive can result in a connection between unrelated events. It is assumed that for large enough data sets real paths will be travelled multiple times and large numbers of false positives are unlikely to occur in the same path. As the final weight values in matrix W^{net} are the sums of every edge weight associated with each possible path, applying the threshold ϵ_{thr} allows us to exclude most false positives while not affecting real paths.

Unlike the pairwise method described in section 4.1, this method makes no distinction between visible and hidden paths. As the goal is to find the hidden paths, after obtaining the full network graph we simply consider only connections between entry/exit zones belonging to different cameras. This ensures that visible paths are not included in the final graph.

Chapter 5

Experiments

In this chapter the experiments used to evaluate the performance of the implemented methods will be detailed.

5.1 Color descriptor

The developed methods do not need a specific descriptor to be used, as the implementation is generic. The descriptor used in this work for the purposes of testing and evaluation was the Black-Value-Tint (BVT) histogram [23].

The BVT histogram is a variant of the commonly used Hue-Saturation-Value (HSV) histogram [24]. To obtain this histogram, a standard Value histogram of the image is obtained. All the pixels belonging to the first bin in this Value histogram (comprised of black and near-black pixels, which have a Value close to 0) are counted separately as the number of black pixels. Then, a 2D histogram of Hue and Saturation is built for the non-black pixels (the pixels belonging to the remaining bins of the Value histogram). This is done because the Tint (Hue and Saturation) values of the black or near-black pixels in an image are essentially random noise. By counting the black pixels for the Value histogram but ignoring them for the 2D Tint histogram this noisy data can be avoided. In our case, the Value histogram was built with 10 bins, and 10 bins were also used for both the Hue and Saturation, resulting in a 2D Tint histogram with 100 bins. This means that the final BVT histograms used as color descriptors had 110 bins in total.

5.2 Data set

The data used during this work was based on the HDA+ data set [5]. This data set consists of footage recorded by 18 cameras distributed over three floors of a university research department for approximately 30 minutes. It contains footage of 83 identified people, as well as manually labeled ground truth bounding box data for 13 of the cameras. An occlusion flag indicates whether the person contained in the bounding box is occluded or fully visible, and crowds of unidentifiable people are labeled as a crowd.

A floor plan of the parts of the network environment used in this work can be seen in figure 5.1, along with the position and field of view of the 13 fully labeled cameras as well as all the visible and hidden paths corresponding to trips observed in the recorded footage.

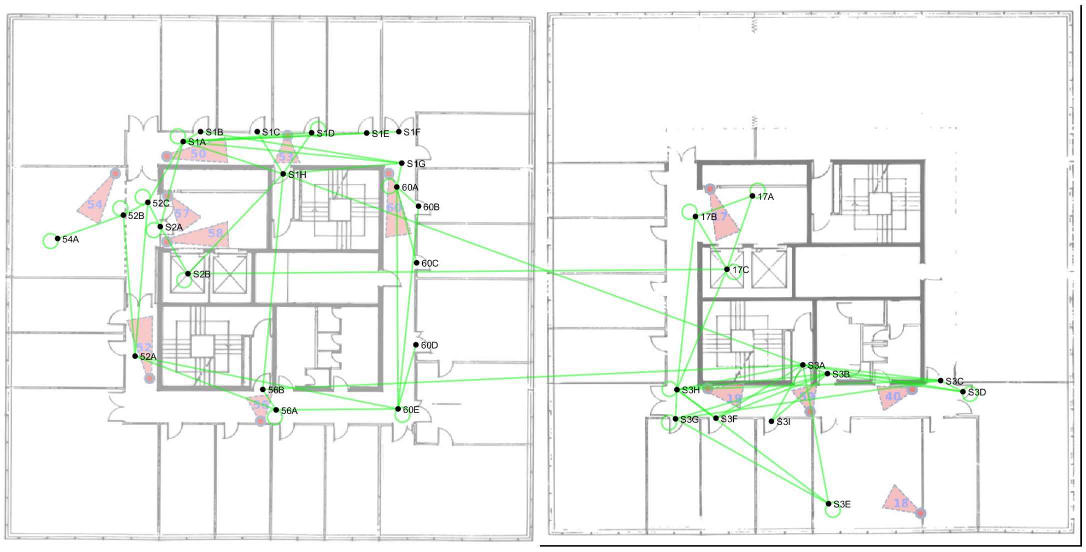
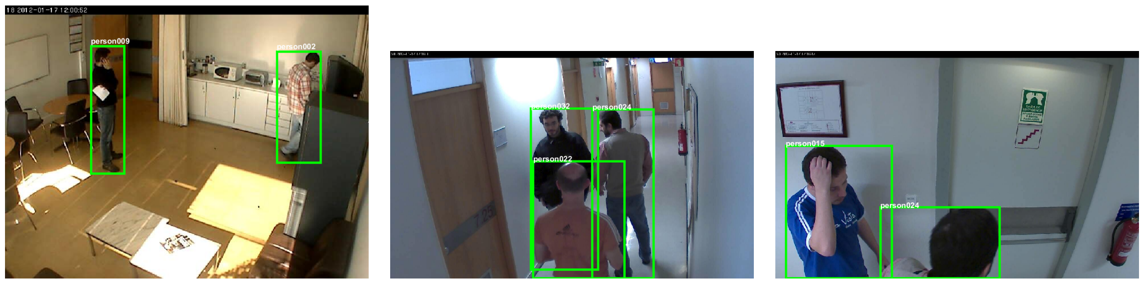


Figure 5.1: Floor plan of the two floors observed by the camera network. The 13 cameras are numbered and their position is represented by red circles, with the triangles representing their fields of view. The green lines represent the paths in the recorded footage.

Figure 5.2 shows examples of the images captured by three of the cameras in the network as well as the manually assigned bounding boxes.



(a) Camera 18 (b) Camera 50 (c) Camera 53

Figure 5.2: Example of images captured by different cameras in the network, including the manually annotated bounding boxes.

As can be seen in figure 5.2 the different cameras provide images taken from different angles and under varied lighting conditions.

The footage taken by the 13 fully labeled cameras includes 980 entry and exit events. As this constitutes a relatively small dataset, we use an event simulator to generate larger amounts of data based on the characteristics of the real environment observed by these cameras, as detailed in section 5.3.

5.3 Event simulator

The data used to evaluate the performance of the implemented methods was obtained using a modified version of the event simulator developed for [4]. This simulator randomly generates events based on the transition probabilities and temporal distributions obtained from the HDA+ data set described in section 5.2. The number of simulated events generated can be controlled by adjusting two parameters - the probability p_{new} of a new person entering the camera network in each second, and the total time t_{sim} to simulate (in seconds). First, the total number of people entering the network during the entire simulation is determined. For each second of the simulation, a new person is added to the list of people in the network with probability p_{new} and the node through which they entered the network is chosen randomly from among all the possible entry nodes. Then, for each person, the remaining entry and exit events are generated. Starting from the initial entry event that brings the person into the camera network, an exit event that completes the first visible trip is chosen taking into account the transition probabilities, with the length of the trip depending on the temporal distributions. Then, an entry event that completes the off-camera trip is chosen in the same way, with the possibility that the person may exit the camera network instead of entering another camera's field of view. This process is repeated for every person in the simulation until they exit the network.

The version of the simulator adapted for this work generates a random BVT histogram for each person. The BVT histogram associated with a person is used as the color feature for all the events pertaining to that person. In order to generate a random BVT histogram, the simulator starts by randomly selecting a number of pixels n_{total}. Then, a number of black and near-black pixels $n_{black} < n_{total}$ is randomly chosen. This number corresponds to the frequency of the first bin of the Value histogram. The remaining pixels $n_{remaining} = n_{total} - n_{black}$ are then randomly split between the 9 remaining bins of the Value histogram. This is done by obtaining 8 random numbers $\{n_i : 0 \leq n_i \leq n_{remaining} \quad \forall i = 1, ..., 8\}$ which are then used in the following algorithm:

```
$array \leftarrow [0, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_{remaining}]$
$array \leftarrow sort(array)$
for $i = 1 : size(array) - 1$ do
    $frequencies(i) \leftarrow array(i+1) - array(i)$
end for
```

This results in 9 random frequencies that add up to $n_{remaining}$, which constitute the 9 non-black bins of the Value histogram. The same method is used with 99 random numbers to obtain 100 random frequency values adding up to $n_{remaining}$ for the Tint histogram, resulting in the complete BVT histogram. The final histogram is then normalized so that the sum of all its values is 1.

It is also possible to add uniformly distributed random noise with a variable maximum amplitude to the features associated with each generated event.

5.4 Evaluation

To evaluate the performance of the graph estimation methods, the estimated graphs are compared to an ideal graph obtained which represents all 15 paths between cameras in the network. Edges that are present in both the estimated and ideal graphs are considered *true positives*. Edges that are present in the estimated graph but not the ideal one are considered *false positives*. Edges that are present in the ideal graph but not the estimated one are considered *false negatives*.

The main metrics used to evaluate the estimated graphs are the precision and recall [25]. The precision of an estimated graph represents the proportion of the identified edges that is correct and is defined as

$$precision = \frac{true\ positives}{true\ positives + false\ positives}. \tag{5.1}$$

The recall of an estimated graph represents the proportion of all the existing edges that is identified by the estimation algorithm and is defined as

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}. \tag{5.2}$$

A low precision value indicates that an estimated graph includes too many nonexistent connections, while a low recall value indicates than an estimated graph failed to recognize too many existing connections. A graph that includes every possible connection would have a recall value of 1 but low precision and a graph that consists of only a single existing connection would have a precision of 1 but a low recall value, with most possible graphs representing a trade-off between the two metrics. In our case, this trade-off can be controlled by the sparsity parameter β. Higher values of β increase the sparsity of the estimated graphs, which means they include fewer connections as edges. Therefore, it is important to identify the values of β that optimize the trade-off between precision and recall to obtain good estimates.

One way to do this is by using the *F-measure* [26], which is the harmonic mean of precision and recall and is given by

$$F - measure = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{5.3}$$

By computing the F-measure for each test we can simply choose the value of β that results in the highest F-measure as the best value of the sparsity coefficient.

5.5 Pairwise method

To test the pairwise method, each set of simulated data was used to estimate a graph with different values of the sparsity coefficient β, in order to evaluate its effect on the resulting estimate.

In all the tests the value used for the color descriptor similarity threshold was $\epsilon_{color} = 0.1$. A first test with ideal noiseless simulated data was run, and in subsequent tests the maximum amplitude of the uniform additive noise added to the normalized histogram values was $r = 2 \times 10^{-3}$. For each test, five simulations were made and the final results being presented are the median of the five runs. This means that in the resulting graphs, each edge is present if it was present in the results of at least three of the five simulations.

For the first noiseless test, the chosen parameters were a test length of $t_{sim} = 10800\ s$ (which corresponds to 3 hours) with a probability of a new person entering in each second of $p_{new} = 0.01$.

Two tests were run with the noisy data. For the first one the parameters $p_{new} = 0.01$ and $t_{sim} = 86400\ s$ (which corresponds to 24 hours) were used. The second test used $t_{sim} = 10800\ s$ and $p_{new} = 0.01$. This shorter simulation time results in a smaller set of data, which should make it more difficult to correctly estimate the hidden paths.

5.6 Event graph method

Due to the increased computational complexity of the algorithm used in this method, it is not practical to run tests with larger amounts of data such as those resulting from using a simulation time $t_{sim} = 86400\ s$. Therefore, testing for this method was focused on a simulation with $t_{sim} = 10800\ s$ and $p_{new} = 0.01$, with the goal of improving upon the results obtained with the pairwise method for the same simulation parameters. An initial test was run with ideal noiseless data, followed by tests using data with uniform additive noise of amplitude $r = 2 \times 10^{-3}$.

As with the tests done for the pairwise method five sets of simulated data were generated and the estimation algorithm was applied to each of them, with the final results being the median of the five resulting graphs. The time coefficients used for the testing were $\tau_1 = 2 \times 10^{-4}$ and $\tau_2 = 1 \times 10^{-4}$, and the threshold used for the creation of the final camera network graph G^{net} was $\epsilon_{thr} = 500$.

Chapter 6

Results

In this chapter the results of the the experiments detailed in chapter 5 are presented. For each test the precision value, the recall value and the F-measure are shown. We also present the total number of edges included in each estimated graph to evaluate the sparsity of the obtained estimates. Figures representing the estimated graphs are also shown for relevant results. In these figures only the hidden paths are shown, as those are the paths we are discovering.

6.1 Pairwise method

In this section the results obtained through the approach detailed in section 4.1 are presented. The first test had no added noise, and the test parameters were $t_{sim} = 10800\ s$ and $p_{new} = 0.01$. The values of β used for this test were $\beta = \{0, 1000, 1500, 2000, 3000\}$. The results of this test can be seen in table 6.1.

β	Number of detected edges	Precision	Recall	F-measure
0	15	1	1	1
1000	15	1	1	1
1500	15	1	1	1
2000	15	1	1	1
3000	12	1	0.8	0.889

Table 6.1: Results of the pairwise method for $t_{sim} = 10800\ s$ without added noise.

The ideal simulated data without added noise results in perfect estimations for most lower values of the sparsity coefficient β. For higher values of β, the sparsity encouraging term grows large enough to exclude correct connections from the estimated graph, resulting in a lower recall value. The precision was 1 for all tested values of β as no false positive connections are identified. As there is no random noise added to the color descriptors, all events associated with the same person have the exact same descriptor. This means that in order for a false positive connection to be obtained, the random color descriptors generated for two different people must be within the chosen similarity threshold of $\epsilon_{color} = 0.1$, which is highly unlikely.

The results obtained for a second simulation with $r_{max} = 2 \times 10^{-3}$ using $t_{sim} = 86400\ s$, $p_{new} = 0.01$ and $\beta = \{0, 3500, 7000, 10000, 20000\}$ are presented in table 6.2.

β	Number of detected edges	Precision	Recall	F-measure
0	84	0.179	1	0.303
3500	21	0.714	1	0.833
7000	15	1	1	1
10000	13	1	0.867	0.929
20000	10	1	0.667	0.8

Table 6.2: Results of the pairwise method for $t_{sim} = 86400\ s$.

Unlike the results obtained with the noiseless data, in this test the precision value is no longer 1 for every value of β. This happens because as each instance of a person's color descriptor is altered with significant random additive noise, it is possible for the noisy data to result in false positives. As the sparsity coefficient β increases, the number of edges in the estimated graph decreases. Lower values of β result in low precision and high recall, with the opposite occurring for higher values of β. For $\beta = 7000$ it was possible to estimate a graph that perfectly represents all the hidden paths, as indicated by the fact that both precision and recall have a value of 1. This means that there are no false positives or false negatives - all existing paths are identified in the estimate, and no false paths are incorrectly identified. Figure 6.1 shows the resulting estimated graph for this value of β, which corresponds to the ideal graph of hidden paths.

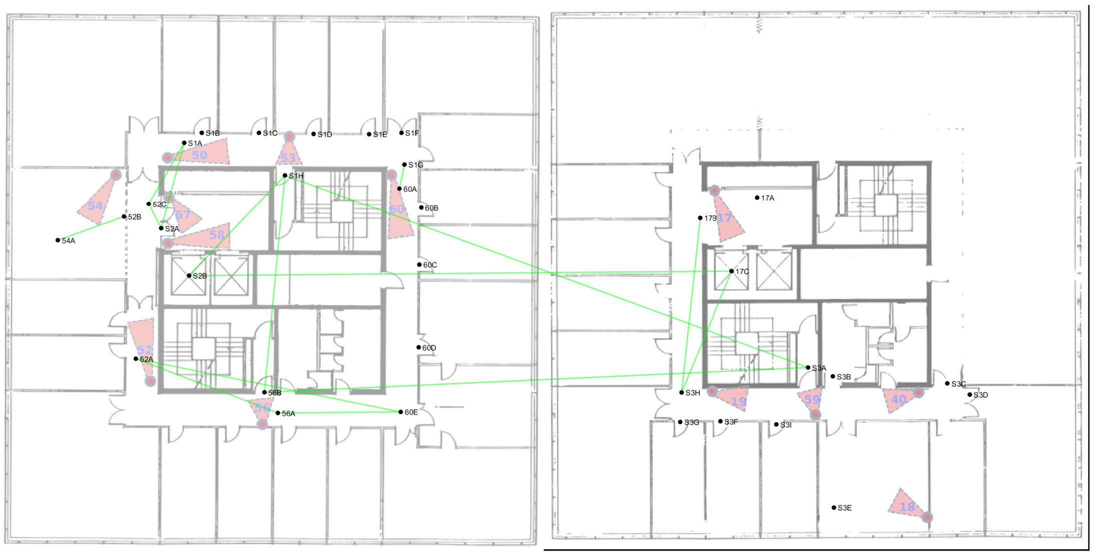


Figure 6.1: Estimated graph of hidden paths obtained with the pairwise method for $t_{sim} = 86400\ s$ and $\beta = 7000$. Green lines represent correctly identified paths.

A third test was run with the parameters $p_{new} = 0.01$, $t_{sim} = 10800\ s$ and $r_{max} = 2 \times 10^{-3}$. The results for this test for $\beta = \{0, 1000, 1500, 2000, 3000\}$ can be seen in table 6.3.

As with the previous test, increasing the value of the sparsity coefficient β decreases the number of

β	Number of detected edges	Precision	Recall	F-measure
0	38	0.395	1	0.566
1000	19	0.79	1	0.882
1500	15	0.933	0.933	0.933
2000	11	1	0.733	0.846
3000	9	1	0.6	0.75

Table 6.3: Results of the pairwise method for $t_{sim} = 10800\ s$.

edges in the estimated graph as well as the recall value, while increasing the precision. However, due to the smaller amount of data available for the estimation of the graph, in this test it was not possible to obtain a completely correct estimation, i.e. a graph with both precision and recall values of 1. In this case the results represent a trade-off between both metrics, with lower values of β resulting in a recall of 1 but lower precision values, higher values of β resulting in lower recall but a precision of 1, and $\beta = 1500$ resulting in a high value for both metrics but with neither of them being perfect. Figure 6.2 shows the resulting estimated graph for $\beta = 1500$, where both one false positive and one false negative can be seen.

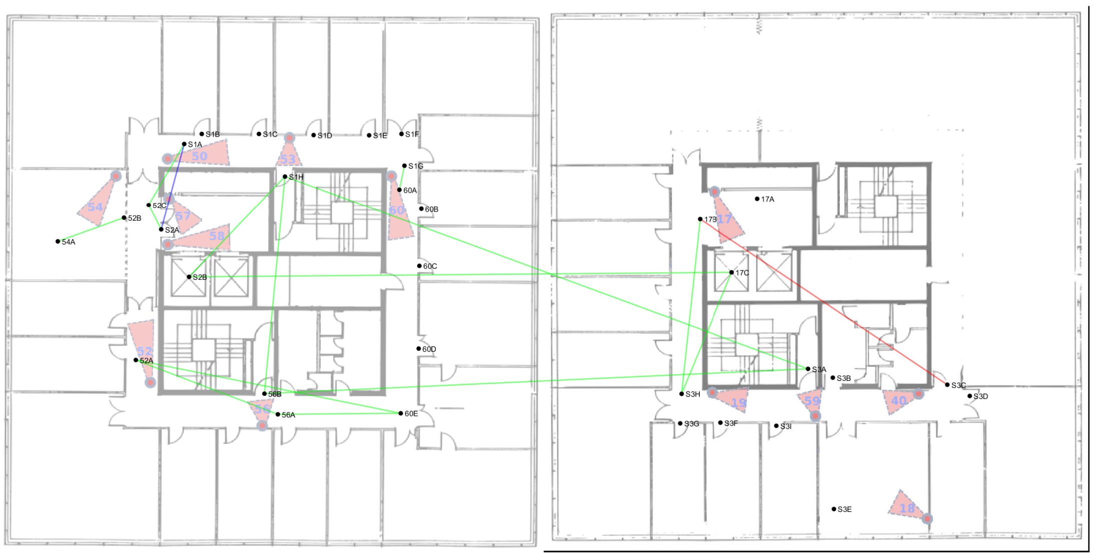


Figure 6.2: Estimated graph of hidden paths obtained with the pairwise method for $t_{sim} = 10800\ s$ and $\beta = 1500$. Green lines represent correctly identified paths, blue lines represent false negatives and red lines represent false positives.

6.2 Event graph method

In this section the results obtained through the approach detailed in section 4.2 are presented.

An initial test was run with noiseless data with a test duration of $t_{sim} = 10800\ s$ and $p_{new} = 0.01$ for $\beta = \{0, 0.025, 0.075, 0.125, 0.2\}$. The value used for the time coefficient was $\tau = 2 \times 10^{-4}$. The results of this test can be seen in table 6.4.

β	Number of detected edges	Precision	Recall	F-measure
0	175	0.086	1	0.158
0.025	29	0.517	1	0.681
0.075	14	1	0.933	0.966
0.125	13	1	0.867	0.929
0.2	10	1	0.667	0.8

Table 6.4: Results of the event graph method for $t_{sim} = 10800\ s$ without added noise.

For the lower values of β, the resulting graph is less sparse than those resulting from the pairwise method. This happens because while the pairwise method utilized a thresholding based on the color descriptors associated with each event prior to applying the graph estimation algorithm, this method relies only on the sparsity penalty to eliminate connections in the resulting graph. This means that when using $\beta = 0$ every possible hidden path is included in the estimated graph, resulting in an extremely low precision value. The graph obtained for this value of β simply corresponds to all paths between two entry/exit zones in different cameras.

It is also worth nothing that despite the fact that the color descriptors used for this test had no added noise, it was not possible to obtain a perfect estimate. This is due to the usage of the time of the event as an additional feature, which can result in real paths that take a particularly long time to walk being excluded from the estimate. Figure 6.3 shows the estimated graph obtained for $\beta = 0.075$, which was the best result obtained as it includes no false positives while missing only a single connection. This missing connection is a path between two different floors, which takes longer to walk than other paths in the network.

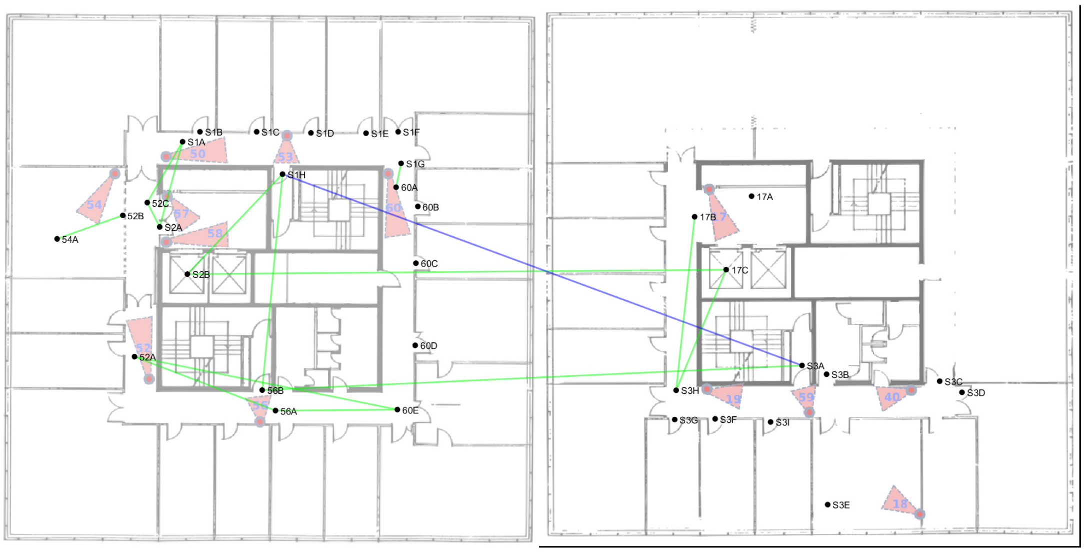


Figure 6.3: Estimated graph of hidden paths obtained with the event graph method for $t_{sim} = 10800\ s$ and $\beta = 0.075$ without added noise. Green lines represent correctly identified paths and blue lines represent false negatives.

A second test was run, this time with uniform additive noise of amplitude $r = 2 \times 10^{-3}$ being added to the color descriptors. The test parameters used were $t_{sim} = 10800\ s$ and $p_{new} = 0.01$ and the test was run for $\beta = \{0, 0.025, 0.075, 0.125, 0.2\}$, once again with $\tau = 2 \times 10^{-4}$. The results of this test can be seen in table 6.5.

β	Number of detected edges	Precision	Recall	F-measure
0	175	0.086	1	0.158
0.025	35	0.429	1	0.6
0.075	14	1	0.933	0.966
0.125	13	1	0.867	0.929
0.2	8	1	0.533	0.696

Table 6.5: Results of the event graph method for $t_{sim} = 10800\ s$.

For $\beta = 0$ we obtain a graph containing every possible invisible path again. As before, increasing the value of β increases the sparsity of the obtained graphs while also increasing the precision and decreasing the recall value of the estimate. For $\beta = 0.075$ it was possible to achieve a precision of 1 and a recall of 0.933. In addition to having the same precision and recall values as the result obtained from the noiseless data, this constitutes an improvement over the results obtained with the pairwise method. The graph obtained for this value of β is represented in figure 6.4, where it can be seen only one existing path was not included in the estimated graph.

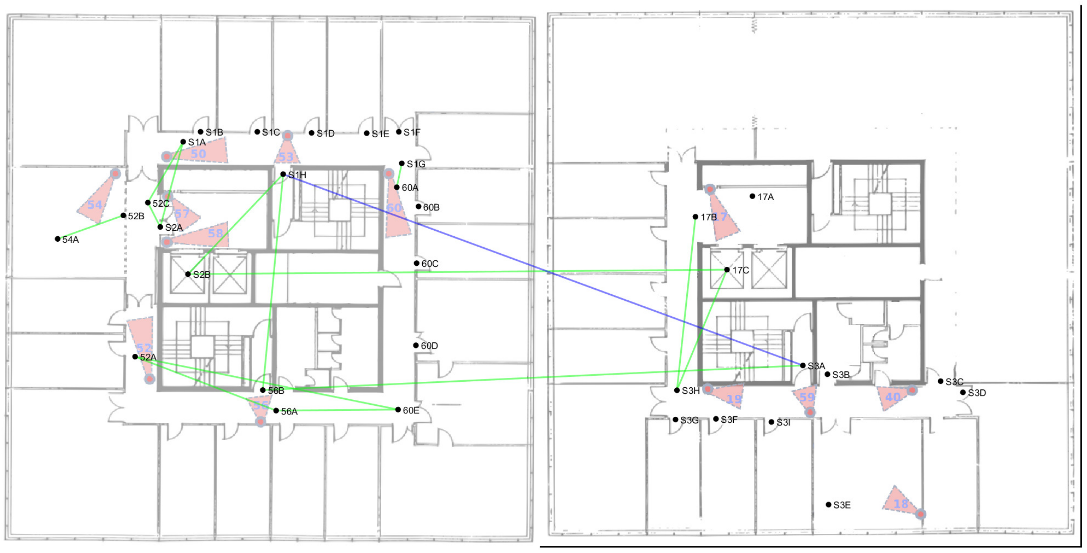


Figure 6.4: Estimated graph of hidden paths obtained with the event graph method for $t_{sim} = 10800\ s$ and $\beta = 0.075$. Green lines represent correctly identified paths and blue lines represent false negatives.

As with the previous test using noiseless data, the path that the algorithm failed to identify connects two different floors of the testing environment. As the moment when each event takes place is used as a feature in this method, paths that take a long time to complete (such as the path in question) are less likely to be included in the estimated graph, as the difference between the temporal features in two events that form a trip between two particularly distant nodes will be large. Reducing the value of the time coefficient τ can help include these paths in the estimated graph, but it also makes it more likely that incorrect paths connecting two nodes that were visited by the same person but not consecutively will be included.

In order to evaluate the effect of the time coefficient τ another set of tests was run with $\tau = 1 \times 10^{-4}$ for $t_{sim} = 10800\ s$ and $\beta = \{0, 0.025, 0.075, 0.125, 0.2\}$ with $r = 2 \times 10^{-3}$. The results of these tests can be seen in table 6.6.

β	Number of detected edges	Precision	Recall	F-measure
0	175	0.086	1	0.158
0.025	35	0.429	1	0.6
0.075	15	0.933	0.933	0.933
0.125	13	1	0.867	0.929
0.2	8	1	0.533	0.696

Table 6.6: Results of the event graph method for $t_{sim} = 10800\ s$ and $\tau = 1 \times 10^{-4}$.

While the results are mostly identical to those obtained with $\tau = 2 \times 10^{-4}$, for $\beta = 0.075$ the precision value is lower. The graph obtained for this value of β can be seen in figure 6.5.

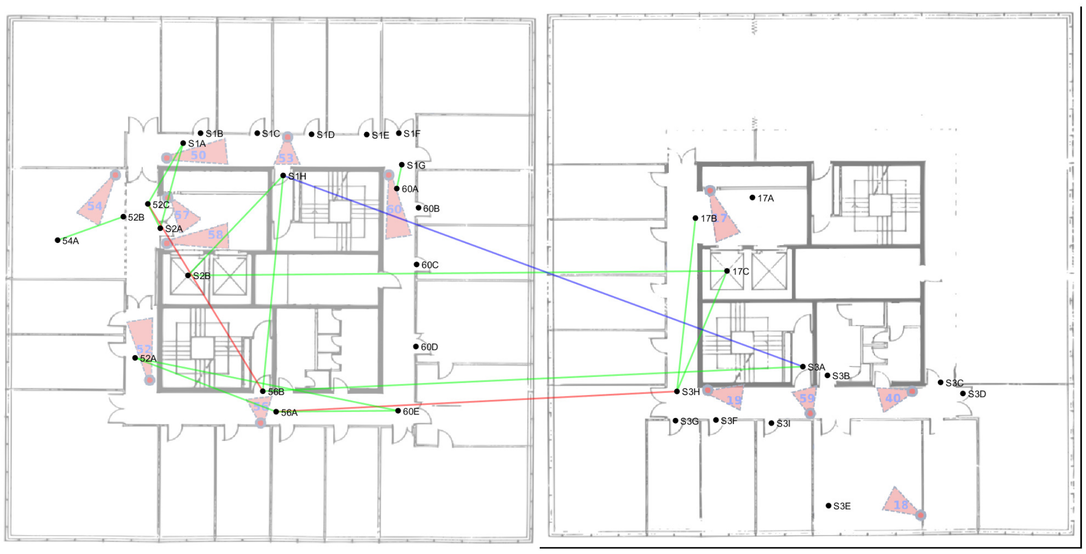


Figure 6.5: Estimated graph of hidden paths obtained with the event graph method for $t_{sim} = 10800\ s$, $\tau = 1 \times 10^{-4}$ and $\beta = 0.075$. Green lines represent correctly identified paths, red lines represent false positives and blue lines represent false negatives.

In addition to the fact that the missing path between the two floors is still not included in the estimated graph, there are two new incorrect paths in the estimate. This means that even relaxing the time coefficient τ enough to match events that do not directly follow one another is still not enough to include paths with such a long travel time as the one connecting the two opposing stairwells on different floors.

These results demonstrate that while the event graph method can not be as easily used with large data sets as the pairwise method, for smaller amounts of data it can obtain more precise results. Moreover, while the pairwise method showed a better performance with the ideal noiseless color descriptors due to not using temporal features, the event graph method is more robust to noisy data, being able to obtain the same estimated graph when using noiseless data as well as color descriptors with uniform additive noise of amplitude $r = 2 \times 10^{-3}$.

Chapter 7

Conclusions and future work

In this work two automated methods for camera network topology estimation were tested with different sets of simulated data. The graphs obtained during the testing were compared with the ideal graphs containing all the off-camera trajectories within the network, and the test results were evaluated based on precision and recall metrics. For each test multiple values of the sparsity coefficient β were tested with the goal of finding the values of β that result in the best estimated graph. In general there is a trade-off between the two metrics we want to maximize so we used the F-measure to select the best value of β for each test.

We were able to obtain favorable results with both methods, with the event graph method demonstrating more robustness to noise in the color descriptor data but performing worse than the pairwise method with ideal data due to the effect of using the event time as a feature. This suggests that the pairwise method may be a better option for data with lower noise values, while the event graph method can be used when there is more noise in the data.

For the pairwise method, data sets with different numbers of events were used for testing. The best values of β depend on the number of events, with larger data sets requiring larger values of the sparsity coefficient to obtain similar results. As future work it would be important to test this method with data from a different environment to evaluate how the same values of β perform with different data sets. While we used different simulations with different parameters and different randomly generated trajectories, they were all based on the real data from the HDA+ data set and shared the same transition probabilities and temporal distributions, which means that the method was not tested with varied data containing different types of trajectories in different environments.

The event graph method was only tested with smaller sets of data due to its greater computational complexity making it impractical to use larger simulations for testing. The best results were consistently obtained for the same value of β across all the different tests, although as with the pairwise method it would be useful to use data from a different environment for future testing.

Both of the methods perform more poorly with smaller data sets, requiring a sizeable amount of data to provide favorable results. As the real data from the HDA+ data set amounts to a relatively small number of events it was impossible to obtain adequate estimates of the camera network topology from the real data. Testing with a larger amount of real data would be an important next step for future testing, as all the testing done so far used simulated color descriptors with additive uniform noise which do not properly reproduce all the challenges of using real color descriptors obtained from camera images.

Bibliography

[1] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(3):334–352, 2004.

[2] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1): 3–19, 2013.

[3] A. Bedagkar-Gala and S. K. Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, 2014.

[4] I. Silva. Estimação de topologia de uma rede de câmaras usando métodos esparsos (in Portuguese). Master's thesis, Instituto Superior Técnico, 2017.

[5] D. Figueira, M. Taiana, A. Nambiar, J. Nascimento, and A. Bernardino. The hda+ data set for research on fully automated re-identification systems. In *European Conference on Computer Vision*, pages 241–255. Springer, 2014.

[6] D. Kamenetsky. Camera network topology discovery literature review. Technical Report DSTO-GD-0667, Defence Science and Technology Organisation, 2011.

[7] X. Li, W. Dong, F. Chang, and P. Qu. Topology learning of non-overlapping multi-camera network. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(11):243–254, 2015.

[8] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, pages 205–210. IEEE Computer Society, 2004.

[9] Y.-J. Cho, J.-H. Park, S.-A. Kim, K. Lee, and K.-J. Yoon. Unified framework for automated person re-identification and camera network topology inference in camera networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2601–2607, 2017.

[10] F. Chung and R. M. Richardson. Weighted laplacians and the sigma function of a graph. *Contemporary Mathematics*, 415:93—-107, 2006.

[11] B. Lake and J. Tenenbaum. Discovering structure by learning sparse graphs. In *Proceedings of the Annual Meeting of the Cognitive Science Society, 32*, pages 778–783, 2010.

[12] X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical Report CMU-CS-03-17, Carnegie Mellon University, 2013.

[13] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[14] H. Rue and L. Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[15] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[16] R. Mazumder and T. Hastie. The graphical lasso: New insights and alternatives. *Electronic journal of statistics*, 6:2125, 2012.

[17] J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse gaussians. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 153–160. AUAI Press, 2008.

[18] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

[19] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE, 2000.

[20] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Mar. 2014.

[21] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[22] R. Mazumder and T. Hastie. *dpglasso: Primal Graphical Lasso*, 2012. URL `https://CRAN.R-project.org/package=dpglasso`. R package version 1.0.

[23] D. S. Cheng, M. Cristani, M. Stoppa, L. Bazzani, and V. Murino. Custom pictorial structures for re-identification. In *Proceedings of the British Machine Vision Conference*, pages 68.1–68.11. BMVA Press, 2011. ISBN 1-901725-43-X.

[24] G. H. Joblove and D. Greenberg. Color spaces for computer graphics. *SIGGRAPH Comput. Graph.*, 12(3):20–25, Aug. 1978. ISSN 0097-8930. doi: 10.1145/965139.807362. URL `http://doi.acm.org/10.1145/965139.807362`.

[25] M. Buckland and F. Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19, 1994.

[26] N. Chinchor. Muc-4 evaluation metrics. In *Proceedings of the 4th conference on Message understanding*, pages 22–29. Association for Computational Linguistics, 1992.

Appendix A

Matlab optimization code for pairwise method

```
% Inputs: D - feature matrix;
%         beta - sparsity penalty coefficient.
% Outputs: w - weight of the single edge being considered.
D = D - mean(D(:));
m = size(D,2);
covar = (1/m)*D*D';

D = D/sqrt(max(covar(:)));
covar = (1/m)*D*D';

cvx_begin
variables delta(2,2);
variables w12;
variables w21;
variables teta;
maximize(log_det(delta)-trace(delta*covar)-(beta/m)*(abs(w12)+abs(w21)));
subject to
teta <= 1/1e-4;
w12 >= 0;
w21 >= 0;
delta(1,1) == w21 + teta;
delta(1,2) == -w12;
delta(2,1) == - w21;
delta(2,2) == w12 + teta;
cvx_end
w = w12;
```

Appendix B

R optimization code for event graph method

```
# Inputs: D - feature matrix;
#         beta - sparsity penalty coefficient.
# Outputs: W - regularized graph Laplacian matrix.
m=dim(D)[2]
D=t(scale(t(D),center=TRUE,scale=FALSE))
covar = (1/m)*D%*%t(D)
D = D/sqrt(max(covar))
covar = (1/m)*D%*%t(D)

struct=dpglasso(Sigma=covar,rho=beta)
W=struct[["X"]]
```