A Framework for Determining Overlap in Large Scale Networks

Anton van den Hengel, Henry Detmold, Christopher Madden, Anthony Dick, Alex Cichowski, Rhys Hill
The Australian Centre for Visual Technologies
School of Computer Science
University of Adelaide
Email: {anton,henry,cmadden,ard,alexc,rhys}@cs.adelaide.edu.au

***Abstract*—This paper presents a novel framework designed for calculating the topology of overlapping cameras in large surveillance systems. Such a framework is a key enabler for efficient network-wide surveillance, e.g. inter-camera tracking, especially in large surveillance networks. The framework presented can be adapted to utilise numerous contradiction and correlation approaches to identify overlapping portions of camera views using activity within the system. It can also utilise a various arbitrary occupancy cells which can be used to adjust both the memory requirements and accuracy of the topology generated. The framework is evaluated for its memory usage, processing speed and the accuracy of its overlap topology on a 26 camera dataset using various approaches. A further examination of memory requirements and processing speed on a larger 200 camera network is also presented. The results demonstrate that the framework significantly reduces memory requirements and improves execution speed whilst producing useful topologies from a large surveillance system at real-time speeds.**

I. Introduction

Video surveillance is becoming increasingly ubiquitous in society, and is often seen by the public as a quick and easy tool for obtaining accurate, verifiable information to evaluate incidents. It is perceived that video footage can provide an undeniable record of observable events, so video footage is increasingly being used in court cases. Video surveillance networks are therefore increasing in scale, with networks of multiple thousands of cameras becoming common. For example, the Washington D.C. police have access to a network of 5,000 cameras [1] and Singapore's Local Transport Authority runs a network of nearly 6,000 cameras [2]. The scale of these networks demands intelligent software to assist human operators in making sense of the vast amounts of data produced. Computer vision research has made significant progress in automating the processing of this data on the small scale (see [3] for a survey), but there has been less progress in scaling these techniques to the much larger networks now being deployed.

Recent approaches aimed at analysing data from large scale surveillance networks have focused upon important network-wide services to support visual processing. *Activity topology estimation* [4] is an example of such a service that provides a description of the spatial and temporal relationships between the fields of view of a network's cameras in the form of a graph. Such a graph can be used to identify where one camera's field of view overlaps with that of an adjacent camera. The *camera overlap topology* is an important sub-graph of the full activity topology that only contains those links where cameras observe common regions of the scene. Thus camera overlap supports the same operations as a general topology; however it is easier to obtain automatically. The estimation this overlap topology from cell occupancy is the focus of the framework described in this paper. This overlap topology aids an operator in predicting the movement of an individual from the field of view of one camera to that of an adjacent camera. It can also allow software applications to provide efficient handover of objects being tracked across cameras, and be used to optimise camera position and orientation through the selection of appropriate degrees of overlap.

In early surveillance systems the overlap topology was either manually specified or derived from camera calibration to common ground planes. More recent systems have been demonstrated that can at least partly automate the process by analysing video from the cameras. These systems often utilise networks containing fewer than 10 cameras [3], but have requirements that mean they do not scale well to networks an order of magnitude larger. For example, [5] requires manually marked correspondences between images of the fields of view of different cameras. [6] requires a training stage where only one object is observed in order to accurately extract the true camera correspondences. [7], [8] and [9] all require many correct detections of objects as they appear and disappear from cameras over a long period of time. They also rely upon the accurate matching of appearance-based representations across cameras, which may have differing intrinsic properties and colour responses, and will almost certainly be observing their fields of view under different illumination conditions. [10] investigates a different approach where exclusion is used to build an overlap topology by gradually removing potential connections based on the results of foreground segmentation [11].

The camera overlap topology is at its most effective in large scale systems with hundreds or thousands of cameras. Such large scale systems are very difficult to effectively map using manual processes. In order to effectively deal with this size of network, a framework is needed that provides scalability and an ability to function automatically with the disturbances that can occur during routine operation of large systems. This requires an ability to operate in a distributed fashion and

978-1-4244-4620-9/09/$25.00

handle other potential problems, such as cameras becoming offline for indeterminate periods. Such a framework also needs to develop an overlap topology that produces overlap results without excluding any possible connections from areas of the system that might see little activity.

The main contribution of this paper is the development of a novel framework that can be used to support the distributed estimation of overlap topologies for large scale surveillance systems. In order to provide the basis of the framework, a number of key concepts, including flexible cell systems for occupancy correlation and time padding. These are described in detail in Section II. Following this, the design of the framework itself is presented in Section III, with some key implementation details discussed in Section IV. As the framework provides a system that can support many overlap estimators, some of the common overlap estimators that have been implemented within the framework are described in Section V. The datasets used to evaluate the framework are then described in Section VI. The framework is then evaluated against several key criteria, including memory requirements, execution time and accuracy of the estimated overlap results, in Section VII.

II. Background

The following sections detail the fundamental concepts upon which the software framework was designed. An overview is provided of the concept of *cell occupancy* and how this concept may be employed in the context of overlap estimation, the novel concept of *cell systems* is defined, and the concept of *time padding* is presented.

A. Cell Occupancy

A *cell* can be defined as an arbitrary region of image-space within a camera's field of view. A cell can be in one of three states at any given time: *occupied* (O), *unoccupied* (U), or *unknown* (X). A fourth pseudo-state of being *present* is defined as the cell being in either of the occupied or unoccupied states, but not the unknown state. A cell is considered occupied when a foreground target is currently within it, and unoccupied if no target is within it. A cell enters the unknown state when no data from the relevant camera is available. Thus for a given camera all cells are always either unknown or present simultaneously. This limited number of states does not include other object or background feature information to limit the processing required for comparisons.

One can build upon this concept by defining a *cell system* to be a set of cells in a surveillance network. A simple cell system could be formed, for example, by dividing the image space of each camera into a regular grid, and taking the cell system to be the collection of all cells of every grid in each camera. Cell systems are an important abstraction within the developed framework, as they enable flexibility in specifying what image-space regions are of interest, and the detail to which overlap is to be discerned. Choice of cell systems involves trade-offs between accuracy, performance, and memory requirements of the resulting system. The number of cells affects the memory requirements of the system, and the pattern and density of cells can affect the accuracy of information in the derived overlap topology. A cell system need not include cells from every camera, and thus cell systems may also aid in processing sub-regions of the overall surveillance network on nodes in a distributed system, such as a system using the partitioning scheme recently described for exclusion [12].

Local cell systems are differentiated from the overall, global cell system as they are defined within a single camera. In the developed framework, a set of local cell systems is combined to form the overall *global* cell system, with no requirement that identical or similar local cell systems be used throughout the overall system.

Overlap may be sought between cells of two disparate cell systems defined on the same surveillance network, in which case we refer to the two cell systems involved as *cell system A* and *cell system B*. Examples of cell system choices include *camera-camera*, where the entire field of view of each camera forms a cell, with cell system A and cell system B being the same cell system, and similarly *pixel-pixel* where each pixel forms a cell. We could also consider *pixel-camera* overlap analysis, with cell system A using pixel-based cells, and cell system B using whole-camera cells. Any level of detail between the extremes of *camera-camera* and *pixel-pixel* may be specified by choosing appropriate cell systems.

B. Correlating Cell Occupancies

Cell occupancy correlation approaches rely on analysing cell state samples taken at a given set of time points to determine relationships between cells. The choice of cells thus determines the image-space regions for which overlap is to be determined. The basic mechanism for the correlation approaches considered here is to count correlations between cell states for each cell pair c_i,c_j across a set of time points in the given input (with c_i being drawn from cell system A and c_j from cell system B, where two different cell systems have been selected for overlap estimation). For example, we may count the number of times c_i and c_j have been occupied together, which is denoted OO_{ij}, or the number of times c_i has been occupied when c_j has been unoccupied, denoted OU_{ij}. Indeed all possible combinations of the four states O, U, X and P can be explored, though not all of this correlation data needs to be stored directly as some of it can be derived from other data to reduce redundancy.

With this *correlation data*, we can support a wide range of overlap estimators. The exclusion approach [10] can be implemented, for example, based primarily on the OU count, which is essentially an exclusion count. For statistical approaches to overlap estimation, probabilities can be estimated as in the following example:

$$P(c_i \text{ occupied}, c_j \text{ occupied}) \approx \frac{OO_{ij}}{PP_{ij}} \tag{1}$$

where PP_{ij} denotes the number of times c_i and c_j have been simultaneously present. Other probabilities can similarly be determined from other correlation counts. The overall overlap

estimation process can thus be divided into the two stages of initial *correlation*, where the correlation data is derived from input data, and a subsequent *estimation* stage where a particular overlap estimator is applied to the correlation data to estimate cell overlap.

C. Time Padding

Since obtaining occupancy samples for all cameras at precisely synchronised times is difficult, a mechanism known as *time padding* is employed. We consider all cells for a camera to be unknown for any period of time where the frame rate drops below a certain threshold. For all other times t, we consider a cell occupied if there is any frame where the cell is occupied within a specified *time padding distance* from time t. The precise variation of time padding varies with the choice of overlap estimator used. Future work will also investigate the alternative to time padding of interpolating target locations between frames so that cell occupancy can be precisely sampled at any given time for all cameras.

III. Framework for Overlap Estimation

The software framework was developed as an extensible, general framework for implementing a variety of systems within the context of cell occupancy correlation for large scale surveillance networks. This focus upon occupancy removes any reliance upon the extraction and comparison of features that might be computationally expensive to perform. One key area of flexibility required was supporting arbitrary cells and cell systems as discussed in Section II-A. Another was the data structure used for correlation. Whilst basic correlation counting is a simple process, a fast correlator suitable for large-scale surveillance networks requires a highly tuned data structure to correlate. The correlator implementation was designed to support the investigation of new and more efficient data structure designs, whilst handling various practical issues such as frame rate variation and lack of camera synchronisation. The framework also supports the use of a variety of overlap estimation techniques, including those outlined in Section V.

The processing pipeline of the framework is illustrated in Figure 1. These are explained in the following sections.

A. Occupancy Input

The occupancy input component handles the definition and instantiation of cell systems. This involves the practical aspects of streaming the input data upon which final cell occupancy is calculated. Arbitrary input graphs can be defined and constructed to specify how occupancy data is generated for a whole cell system. A variety of transform nodes are supported to generate occupancy data in various ways. For example, a transform node may be used to resample pre-computed occupancy on a grid-based cell system to a lower grid resolution. This component also supports buffering and time padding for performing on-line, live correlation where data from cameras may arrive at any time.

B. Correlator

The correlator provided by the framework includes support for applying time padding as discussed in Section II-C, and appropriately handling cases where cameras go offline. The data flow pipeline throughout the various components within the correlator itself are shown in Figure 2.

The initial stage of the pipeline is *grouping*, where frames at roughly coincident times are grouped into time points. For each time point, the available occupancy data is mapped into a pair of boolean *occupancy vectors*, one for cell system A and another for cell system B. These describe whether each cell in the cell system is occupied. These occupancy vectors are accompanied by similar *presence vectors* describing which cameras, or more precisely, which local cell systems within the overall global cell systems, are present. For a given time point, any cameras for which no frame was mapped are considered not present. Occupancy and presence vectors are arranged according to standard indexing schemes defined over the cell systems being used. For a given camera, all cells are either unknown or present, i.e. occupied or unoccupied, together. Thus, the full state (X, U or O) of every cell in the system is completely encoded. These vectors are typically populated sparsely, and are maintained in the run-length encoded representation described in Section IV-A for performance.

These occupancy and presence vectors are then passed through the time time padding process. This maintains the vectors over a range of time points sufficient to compute the occupancy and presence vectors for "cell system A state" and "cell system B state" in Figure 2. The configuration of time range and time padding buffers used depends on the precise time padding strategy selected. This process also masks out those cameras having low frame rates. Following this process, each time point will be left containing full, time-padded occupancy data from those cameras which are currently streaming useful video.

C. Overlap Estimation

In the final stage of the framework pipeline, the correlation data for each cell pair c_i,c_j is processed by the selected overlap estimator. This determines which cell pairs are overlapping. Extensive support is provided for dealing with both correlation data and overlap topologies, making the implementation of any given overlap estimation technique based on occupancy data straightforward. The list of overlapping cells resulting from the overlap estimator forms the overlap topology.

IV. Implementation

This section discusses the important implementation details of the framework, notably the correlation data storage methods and the multi-threading components of the system.

A. Correlation Data Storage

As mentioned in Section II-B, there is significant redundancy implied in storing all correlation matrices. Various minimal subsets can be stored without losing the ability to

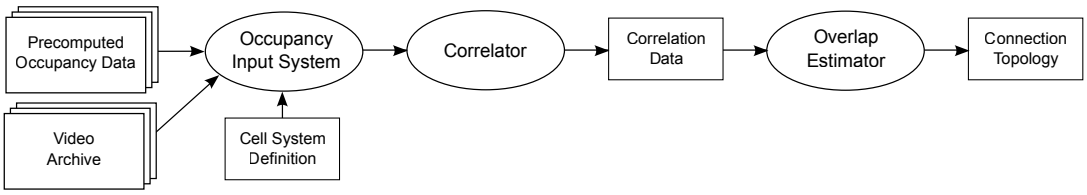

Fig. 1. Data flow pipeline formed by components in the developed framework. Rectangles indicate data that may be saved to disk. Ellipses indicate framework components. Dashed lines indicate component dependencies.

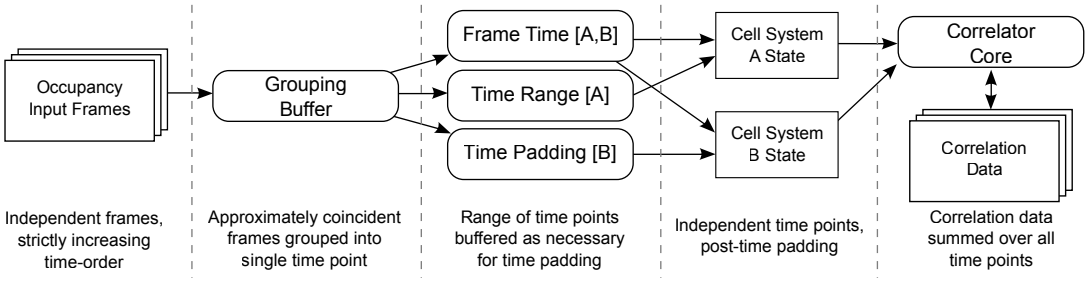

Fig. 2. Data flow between correlator components and nature of data at each stage of the pipeline, when time padding is enabled for cell system B but not cell system A. Rectangles indicate data. Ellipses indicate components. Text in square brackets denotes which cell system data relates to.

reconstruct the correlation data. Correlating upon minimal subsets gives different performance characteristics. For example, using the OU matrix would yield very poor performance, as most cells tend to be unoccupied rather than occupied, and the OU matrix needs to be updated in proportion to the number of unoccupied cells. This leads to significant updating of the data without adding information. Various minimal subsets including OO, OX, XO, and XX were found to provide the best performance.

Support for performing correlation directly on compressed correlation matrix representations was also added, with compression performed by run-length encoding (RLE). In RLE mode, correlation matrix rows usually stored as arrays are replaced with run-length encoded lists. These are rewritten each time any entry in a row needs to be updated. Run-length encoding is effective in compressing correlation matrices due to contiguous regions resulting from large areas of cells remaining wholly unoccupied. Unfortunately run-length encoding was found to incur a penalty in execution time, with Section VII discussing this trade-off.

B. Multi-threading

To take advantage of the parallel processing capabilities and the expected scale the framework to larger surveillance networks, support for multi-threading was required within the framework. Profiling revealed that the vast majority of CPU time is spent updating the correlation matrices in the core component of the correlator. Thus, the multi-threading enhancement has focused on parallelising the correlator core.

A basic approach to multi-threading was implemented involving parallelising individual executions of the correlator core for each time point. In this approach, a set of worker threads is created at startup. When updating the correlation matrix for a given time point, an array describing matrix row updates to be performed is created. Row updates are then divided up equally among worker threads, which are then signalled to begin work. This simple approach ensures no contention between threads, with each updating separate rows of the correlation matrices, hence no locking is required. This approach does not take advantage of distributing the workload across multiple time points, and tends to give a performance penalty on smaller datasets. It also has difficulty in achieving more than a $2.5\times$ speedup on larger datasets when executed on a system with 8 CPU cores.

A second approach to multi-threading was implemented to enable matrix updates for different time points to be undertaken simultaneously by different threads. In this approach, matrix row updates are distributed out to worker threads as before, but each worker thread is allowed to continue working continuously on its own queue of row updates for the duration of execution. The main thread regularly adds work to each worker thread's queue, in each instance choosing the best thread to balance the workload evenly.

The workload balancing of the second approach was successful, but introduces the potential for two threads to be attempting to update the same matrix row simultaneously. Locking was hence required for each matrix row. Self-profiling results generated by the framework showed a large degree of contention in performing row updates, and this manifested in the difficulty in obtaining speedups higher than $3\times$ on the same system with 8 CPU cores, as seen in Section VII.

V. Camera Overlap Estimators

This section describes several methods implemented within the software framework for estimating camera overlap based upon correlation results. These estimators are derived using

several common probabilistic representations: $P_X(x)$ represents the probability of X being in state x, $P_Y(y)$ represents the probability of Y being in state y, and $P_{XY}(x,y)$ represents the probability that both X is in state x and Y is in state y at the same time. Approximate values for such probabilities can be obtained from correlation data as shown in Equation 1.

A. Exclusion Estimator

The exclusion approach is based on the observation that if, at a given point in time, cell i is occupied and cell j unoccupied, then cells i and j do not overlap (*i.e.* there is evidence contradicting overlap)[10]. Efficient implementation requires several extensions of the basic exclusion principal: i) lowest visible extent is applied to occupancy blobs to place cells i and j on the same solid surface, ii) accumulation of contradictions over time improves efficiency and overcomes errors in the occupancy signal output by foreground detection, iii) exploitation of the bidirectional nature of overlap to strengthen the evidentiary value of exclusions and iv) temporal padding of the occupancy signal to overcome clock skew and codec latency effects.

B. Mutual Information Estimator

A correlation approach based on the mutual information of cell pairs can be supported by information theory, to produce a more statistical analysis of the data available. The mutual information represents the amount of information that is obtained about a second variable when one knows the value of the first variable. The mutual information, $I(X;Y)$ for two occupancy cells represented as binary random variables X and Y is given by:

$$I(X;Y) = \sum_{y \in Y, x \in X} p_{XY}(x,y) log \left(\frac{p_{XY}(x,y)}{p_X(x) p_Y(y)} \right) \quad (2)$$

High values of $I(X;Y)$ indicate a high probability of overlap, as one cell being occupied is very predictive of another cell being occupied. Only a single parameter parameter required to threshold from the $I(X;Y)$ domain, although a function could be determined to derive the probability of overlap instead.

C. Conditional Entropy Estimator

Another correlation approach that could form the statistical analysis of the overlapping topologies is conditional entropy. The conditional entropy quantifies the amount of entropy remaining about a random variable when a second random variable is already known. For the overlap case, one would expect the conditional entropy to be minimal where cells are overlapping. The conditional entropy, $H(X|Y)$ for two occupancy cells represented as binary random variables X and Y is given by:

$$H(X|Y) = - \sum_{y \in Y, x \in X} p_{XY}(x,y) log p_{XY}(y|x) \quad (3)$$

Low values of $H(X|Y)$ indicate a high probability of overlap. Again a single parameter is required to threshold from the $H(X|Y)$ domain.

D. IntelliVid's Correlation Estimator

IntelliVid have patented several correlation estimators [13]. The estimator investigated here is based on $lift(X,Y)$ for two occupancy cells represented as binary random variables X and Y:

$$lift(X,Y) = \frac{p_{XY}(1,1)}{p_X(1) p_Y(1)} \quad (4)$$

Values of $lift(X,Y)$ significantly greater than 1.0 provide evidence in favour of overlap.

VI. Datasets

This paper analyses the framework using two datasets that represent both a smaller scale system and a larger scale system. The first dataset consists of 26 surveillance cameras placed around an office and laboratory environment as shown in Figure 3. Each camera has some degree of overlap with at least one other camera's field of view. The data was obtained at full frame rate over a period of approximately four hours. The views of these cameras have been manually analysed to extract the ground truth of overlap for accuracy testing.

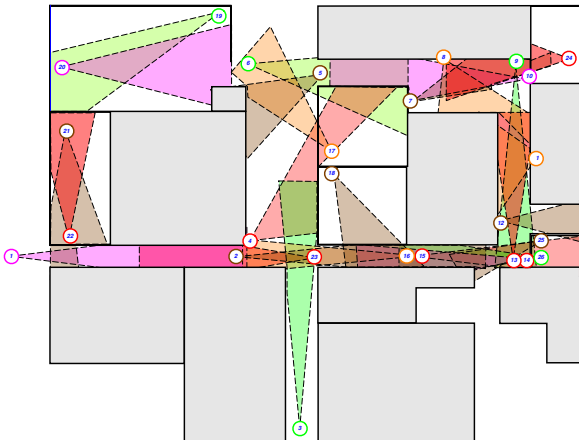

Fig. 3. A floor plan showing the 26 camera dataset. The camera positions are shown by circles, with coloured triangles designating each camera's field of view. White areas are open spaces, whilst light blue regions showing areas that are impassable.

A second, larger dataset consisting of over 200 camera streams was also captured. These cameras spanned a campus environment, and cannot be readily viewed on a single plan view of the area. However, Figure 4 shows an example overlap topology for this camera network where coloured lines link camera views that were found to be overlapping. These links are clustered into groups, where disconnected groups are deemed to be non-overlapping. Such sparsely connected overlap graphs are typical of larger camera networks. Any camera not connected with another camera was omitted in Figure 4. The video sequences were obtained across a period of approximately two hours, but were captured at varying frame rates of between 5 to 10 frames per second, depending on the performance of individual cameras. This dataset has not been fully ground truthed, and so is only used to analyse

the speed and memory requirements for dealing with such a large surveillance network in the framework.

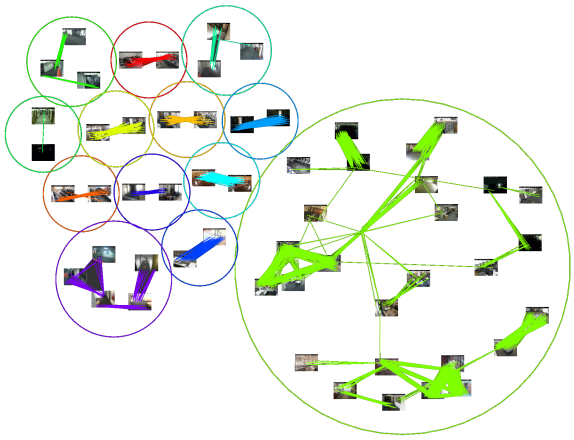

Fig. 4. An example overlap topology for the 200 camera network showing connected components.

VII. Results

The main focus of this paper has been the development of a novel framework to estimate overlap topologies. The evaluation of this framework is performed in two parts to separate the evaluation of the memory and execution speed of the framework from the evaluation of the accuracy of the resulting overlap topologies produced by the system. This separation is made to highlight the fact that the memory and execution speed of the generic correlation stage of the framework is independent from the overlap estimator to be applied to the data. It is a relatively insignificant step in terms of execution time and memory requirements, with the accuracy of the results of the overlap estimator are largely indicative only of the overlap estimator used.

The correlation performance results presented in the following section demonstrate the effectiveness of the developed framework for the correlation component of the overall overlap estimation process. It is important to note that these results were generated using pre-computed occupancy data, and do not reflect the overheads involved in analysing video streams to extract foreground object data from which cell occupancy may be extracted. However, we note that foreground detection has the potential to be performed directly within modern IP cameras, and deriving cell occupancy from foreground detection results is trivial.

The results of the overlap topology estimation are also presented to demonstrate the accuracy of the topologies generated by the final, overlap estimation stage of the framework. Due to the difficulties of obtaining large scale ground truth, these results are based upon the 26 camera dataset.

Except where otherwise indicated, the cell systems used in these results consisted of cells on a regular 12×9 grid for each camera in the network, yielding a total of 108 cells per camera. The analysis presented here was executed upon a current high-end desktop-level computer system, consisting of two 2.83GHz Intel Xeon CPUs, each with 4 cores, and a total of 8GB of RAM. It should be noted that this process operates an order of magnitude faster than real-time, so more limited hardware could still operate in a real-time system.

A. Correlation Performance Results

The correlation performance results presented here analyse a range of important factors affecting the correlation stage within the overall overlap estimation system. We begin with the analysis of the memory requirements of the system when using various cell systems. This is followed by an exploration of the use of RLE-based representations rather than simple arrays in storing correlation data in memory during the correlation process, as discussed in Section IV-A. A comparison with the existing system presented by [14] examining relative performance under different configurations. Finally, the execution speeds of single-threaded and multithreaded instantiations of the system are investigated.

The initial consideration of any large scale system is the memory requirements for storing information during processing. The memory usage resulting from configuring cell systems A and B of correlation on a (12×9 grid) cell to (12×9 grid) cell basis, cell to camera basis, and camera to camera basis (see Section II-A for details) follow the expected trends. The camera-camera requirements are very small as expected; however the usefulness of such results is also minimal, as they not give any indication of which *areas* of which cameras overlap, merely which cameras overlap. A cell-camera overlap topology provides additional information about where an operator or automated process might begin to look for objects moving across cameras, and yet only requires less than 50MB for processing over 200 cameras. Determining full cell-cell correlations provides the most detail, but requires over 2000MB of memory, which may be prohibitive for a system that is not dedicated to the task. This size of the required data structure primarily results from the storage of the OO matrix over all cell pairs. Since each camera has a 12×9 grid, and therefore 108 cells, and as there is a total of 210 cameras in the dataset, there is thus a total of 22680 cells in the system. With the correlation count for each cell pair being stored in 32 bits, the total size of the OO matrix is:

$$4 \times 22680^2 = 2057529600 \text{ bytes} \approx 2GB$$

Note, however, that the use of RLE helps to greatly reduce this memory overhead. The cell to cell configuration provides much more information about how regions within camera views might overlap, which improves the localisation of objects in a given camera to specific regions of the views of other cameras. The particular configuration of the framework that will be most desirable is generally a trade-off involving the processing power and memory available. The relative memory usage of the above options is similar for the 26 camera dataset; however the full memory required for cell-cell correlation is approximately 36MB due to the smaller number of cameras.

Figure 5 demonstrates both the execution time and memory requirements of using the exclusion system presented in [14]

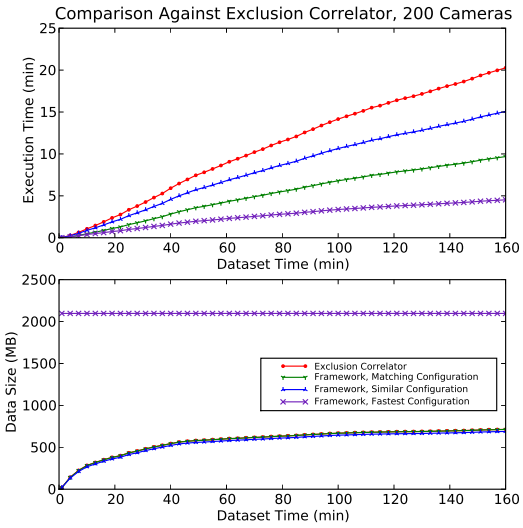

Fig. 5. Analysis of [14] versus the developed framework for the 200 camera dataset.

compared with the developed framework in various configurations. It demonstrates the reduced memory requirements of the framework when using the RLE option. The "matching configuration" result configures the framework to operate in a fashion closely matching that of the exclusion system, which also uses RLE, and effectively correlates only on OU and OP matrices. It can be seen that for this equivalent configuration, the developed framework greatly outperforms the exclusion system. The "similar configuration" result changes the correlation matrix configuration so that the full range of correlation data can be extracted. This has a performance impact, but the framework still outperforms the exclusion system, despite producing fully generalised correlation results not specific to exclusion. Finally, the "fastest configuration" result disables RLE (a configuration unsupported by the exclusion correlator), which can be seen to give by far the fastest performance given that sufficient memory is available. It can be seen here that RLE memory usage increases over time due to decreasing compressibility of correlation data, but this tends to stabilise at approximately a quarter of the memory size of the full array-based method. Regions of the correlation data structures corresponding to never-occupied cells and non-overlapping cell pairs do not need to be stored directly with the RLE method, and large numbers of such cells and cell pairs are typical of large camera installations which do not have high numbers of overlapping cameras.

The main problem with RLE is that updating correlation matrix rows requires the temporary unpacking of RLE data so that counts within rows can be updated, reducing the execution speed. Figure 5 shows that the overall process still operates at approximately half the speed of the full array-based data structure. This may be a highly desirable option in some cases, enabling systems with limited memory to process large, 200 camera datasets at reasonable speeds. Note that all execution speeds presented here are well above real-time performance levels. The results follow a similar trend for the 26 camera dataset, at speeds an order of magnitude higher.

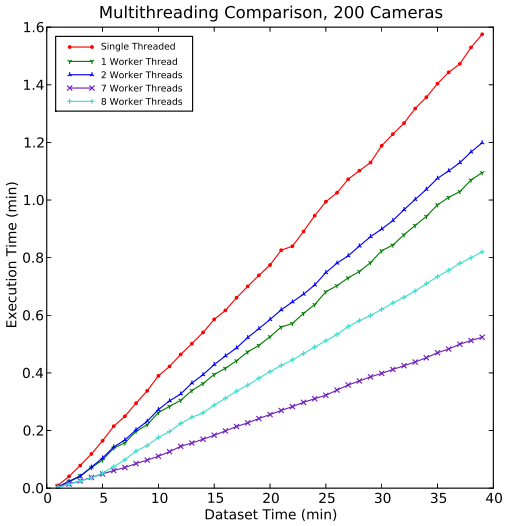

Fig. 6. The execution time versus the dataset time for multithreaded analysis on the 200 camera datasets

Figure 6 shows the effect of running the system in multithreaded mode for the 200 camera dataset with various numbers of threads. Here, it can be seen that introducing a single worker thread greatly aids the performance of the system by allowing the core of the correlator to run entirely in parallel with the remainder of the framework. There is little opportunity for contention between the main thread and the worker thread. Adding a second worker thread leads to a small performance decrease due to contention between worker threads. The best configuration is seen to be 7 worker threads in addition to the main thread, providing optimal usage of all 8 CPU cores. Adding an additional thread again leads to a performance decrease due to additional contention without the benefit of another CPU core to utilise.

B. Overlap Topology Estimation Results

The focus of this paper has been to describe the framework developed for the estimation of camera overlap topologies. The results presented here relate to the accuracy of the estimators, discussed in Section V, that have been utilised within the framework. The accuracy analysis of each estimator is based upon the results of the estimator being compared to a manually determined ground truth for the 26 camera dataset. This dataset consists of 26 cameras placed for surveillance purposes throughout an indoor environment and is described in detail in Section VI.

In each case, results are presented as a P-R curve. Precision and recall are standard metrics for the accuracy of a classifier [15], particularly in the information retrieval context. Given a classifier with true positives, TP, false positives, FP, and false negatives, FN, precision, P, is given by:

$$P = TP/(TP + FN) \tag{5}$$

and recall, R, is given by:

$$R = TP/(TP + TN) \tag{6}$$

In order to obtain values to use in the equations above, a threshold must be applied the the scalar value returned by each estimator (see Section V) to select which links are considered to be overlapping or non-overlapping. A P-R curve can be generated by adjusting this threshold and plotting the P and R values over the range of usable threshold values, which will depend upon the particular estimator being used.

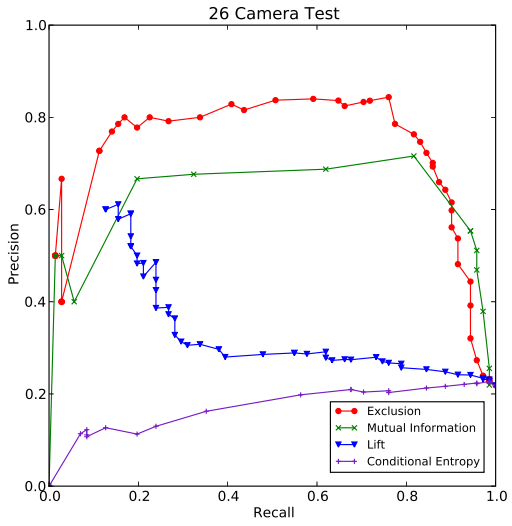

Fig. 7. The accuracy of the overlap topology results

Figure 7 shows precision-recall results for the four different estimators presented in Section V applied to the 26 camera dataset. These results demonstrate that using some of these estimators, a reasonable level of precision can be obtained, even for a relatively high level of recall. The *lift* and conditional entropy estimators provide very poor precision across the range of recall values. The mutual information-based estimator provides a considerable improvement to the point where, whilst the results are not highly accurate, they would provide useful overlap information to subsequent processes. The exclusion estimator provides an additional level of precision up to the point of 80% recall of the ground truth information, though it is outperformed by the mutual information estimator for very high levels of recall.

VIII. Conclusion

This paper has presented a novel framework that can utilise numerous contradiction and correlation methods to estimate an overlap topology of a surveillance system where cameras have overlapping fields of view. The framework presented is highly extensible, and has the flexibility to deal with many real world implementation problems, such as camera unreliability and cameras not being fully synchronised. The framework enables the use of fully customisable cell systems for determining image-space regions to be correlated, allowing a high degree of control regarding the involved trade-offs between memory requirements, processing speed, and overlap topology accuracy. The results presented demonstrate that the framework can provide a useful overlap topology through faster-than-real-time analysis of cell occupancy data without maximising the CPU or memory usage of a current desktop PC system, even when analysing occupancy data from 200 individual camera streams. Such a framework will be useful in many large scale surveillance applications, and will also be valuable for future comparisons of the accuracy of a variety of overlap estimation methods.

References

[1] G. Emerling, "D.C. Police set to monitor 5,000 cameras," 2008, washington Times.

[2] J. Griffin, "Singapore deploys march networks vms solution," 2009, iP Security Watch. [Online]. Available: http://www.ipsecuritywatch.com/web/online/IPSW-News/Singapore-deploys-March-Networks-VMS-solution/512$14948

[3] M. Valera Espina and S. A. Velastin, "Intelligent distributed surveillance systems: A review," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192–204, April 2005.

[4] A. van den Hengel, A. Dick, and R. Hill, "Activity topology estimation for large networks of cameras," in *AVSS '06: Proc. IEEE International Conference on Video and Signal Based Surveillance*, 2006, pp. 44–49.

[5] O. Javed, Z. Rasheed, K. Shafique, and M. Shah, "Tracking across multiple cameras with disjoint views," in *Proc. IEEE International Conference on Computer Vision*, 2003, pp. 952–957.

[6] A. Dick and M. J. Brooks, "A stochastic approach to tracking objects across multiple cameras," in *Proc. Australian Joint Conference on Artificial Intelligence (AI'04)*, 2004, pp. 160–170.

[7] T. J. Ellis, D. Makris, and J. Black, "Learning a multi-camera topology," in *Joint IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, 2003, pp. 165–171.

[8] C. Stauffer, "Learning to track objects through unobserved regions," in *IEEE Computer Society Workshop on Motion and Video Computing*, 2005, pp. II: 96–102.

[9] K. Tieu, G. Dalley, and W. Grimson, "Inference of non-overlapping camera network topology by measuring statistical dependence," in *Proc. IEEE International Conference on Computer Vision*, 2005, pp. II: 1842–1849.

[10] R. Hill, A. van den Hengel, A. R. Dick, A. Cichowski, and H. Detmold, "Empirical evaluation of the exclusion approach to estimating camera overlap," 2008, proceedings of the International Conference on Distributed Smart Cameras.

[11] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[12] H. Detmold, A. van den Hengel, A. R. Dick, A. Cichowski, R. Hill, E. Kocadag, Y. Yarom, K. Falkner, and D. Munro, "Estimating camera overlap in large and growing networks," in *2nd IEEE/ACM International Conference on Distributed Smart Cameras*, 2008.

[13] C. Buehler, "Computerized method and apparatus for determining field-of-view relationships among multiple image sensors," 2007, united States Patent 7286157.

[14] H. Detmold, A. van den Hengel, A. R. Dick, A. Cichowski, R. Hill, E. Kocadag, K. Falkner, and D. S. Munro, "Topology estimation for thousand-camera surveillance networks," in *Proceedings of First ACM/IEEE International Conference on Distributed Smart Cameras*. IEEE, September 2007, pp. 195–202.

[15] V. Raghavan, P. Bollmann, and G. S. Jung, "A critical investigation of recall and precision as measures of retrieval system performance," *ACM Trans. Inf. Syst.*, vol. 7, no. 3, pp. 205–229, 1989.