ESTIMATING CAMERA OVERLAP IN LARGE AND GROWING NETWORKS

Henry Detmold, Anton van den Hengel, Anthony Dick, Alex Cichowski, Rhys Hill, Ekim Kocadag, Yuval Yarom, Katrina Falkner and David S. Munro
{henry,anton,ard,alexc,rhys,ekim,yval,katrina,dave}@cs.adelaide.edu.au

The Australian Centre for Visual Technologies
The University of Adelaide

ABSTRACT

Large-scale intelligent video surveillance requires an accurate estimate of the relationships between the fields of view of the cameras in the network. The exclusion approach is the only method currently capable of performing online estimation of camera overlap for networks of more than 100 cameras, and implementations have demonstrated the capability to support networks of 1000 cameras. However, these implementations include a centralised processing component, with the practical result that the resources (in particular, memory) of the central processor limit the size of the network that can be supported. In this paper, we describe a new, partitioned, implementation of exclusion, suitable for deployment to a cluster of commodity servers. Results for this implementation demonstrate support for significantly larger camera networks than was previously feasible. Furthermore, the nature of the partitioning scheme enables incremental extension of system capacity through the addition of more servers, without interrupting the existing system. Finally, formulae for requirements of system memory and bandwidth resources, verified by experimental results, are derived to assist engineers seeking to implement the technique.

1. INTRODUCTION

Video surveillance networks are increasing in scale: installations of $50,000$ camera surveillance networks are now being reported [1], and networks of more than 100 cameras are common place. It has been reported that Washington D.C. police have access to $5,000$ cameras [2], for instance. Even for networks of ten cameras, human operators require assistance from software to make sense of the vast amounts of data in video streams, whether it be to monitor ongoing activity or to search through archives for a specific event. Computer vision research has made significant progress in automating processing on the very 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.

This work was supported by ARC Discovery Grant DP0770482 and the Government of South Australia PSRF scheme.

A promising approach to tackling large surveillance networks is to identify a core set of network wide *common services*, needed by many visual processing approaches, and then focus research effort on providing these services on large networks. A key example of such a service is estimation of *activity topology*. The activity topology of surveillance network is a graph describing the spatial and temporal relationships between the fields of view of the network's cameras. An accurate and up-to-date estimate of activity topology supports reasoning about events that span multiple cameras. In particular, activity topology supports efficient solution of the *camera handover* problem, which is concerned with the continuation of visual processing (*e.g.* tracking) when a target leaves one camera's field of view and needs to be resumed using data from other cameras (*i.e.* those adjacent in the topology).

The exclusion approach [4] has two desirable properties as an implementation of activity topology estimation. First of all, it produces topology estimates of sufficient accuracy (precision and recall) to be useful in tracking [5]. Secondly, it has proven ability to provide on-line estimation for networks of up to 1,000 cameras [6], whereas no other approach has demonstrated the ability to scale beyond 100 cameras.

However, previous implementations of exclusion, including that described in [6] require a central server component. Specifically, the scale of the surveillance systems these implementation can support in limited by the physical memory on this central server, with the practical consequence that for systems with more than a few thousand cameras, it is not possible to use commodity equipment for the central server, and system implementation thus becomes prohibitively expensive.

The first major contribution of this paper is that it reports experimental results for a decentralised and partitioned memory implementation of activity topology estimation by exclusion. This overcomes previous implementations' dependence on a single central server, and as a result provides a much cost effective approach to implementation of exclusion for large surveillance networks. Results comparing partitioned and non-partitioned exclusion demonstrate that advantages of partitioning outweigh the costs. The second major contribution concerns the properties of the partitioning

978-1-4244-2665-2/08/$25.00

scheme. Specifically, the scheme enables partitions to execute independently. This both enhances performance (through increased parallelism) and, more importantly, permits partitions to be added without affecting existing partitions. This property results in an activity topology estimation sub-system that can grow in capacity as the number of cameras increases, whilst remaining on-line 24×7. The final contribution of this paper is the derivation of formulae for the network and memory requirements of partitioned exclusion. These formulae, verified by experimental results, enable engineers seeking to use exclusion to determine the capacity required from the implementation platform.

2. ACTIVITY TOPOLOGY AND EXCLUSION

An estimate of the activity topology of a surveillance network makes feasible a number of processes critical within on-line video surveillance. The nodes of the activity topology graph are the fields of view of individual cameras, or alternatively regions within those fields of view. Each such region is labelled a *cell* and denoted c_x. The edges of the graph represent the connections between cells. These connections may be used to represent the overlap of the cells or, by including timing information, to describe the movement of targets through the graph. Overlap is an important special case of the more general notion of topology, and we focus on this special case in this paper. Our approach to the estimation of activity topology is termed *exclusion*.

2.1. Formulation of Activity Topology

The activity topology graph is defined as follows:

1. Edges are directed, such that (c_i, c_j) represents the flow from c_i to c_j whereas (c_j, c_i) represents the (distinct) flow from c_j to c_i. Directed edges can be converted to undirected edges if required, but the exclusion algorithm estimates each direction independently and thus we retain this information.

2. Each edge has a set of labels, $p_{i,j}^{[a,b]}$ for various time delay intervals $[a, b]$, each giving the probability that activity leaving c_i arrives at c_j after a delay between a and b. In this paper, each edge has exactly one such label, that for $[-\epsilon, \epsilon]$ where ϵ is some small value large enough to account for clock skew between cameras. Thus $p_{i,j}^{[-\epsilon,\epsilon]}$ describes overlap between cameras.

Actual activity topologies are constrained by building layout, camera placement and other factors. Typical topologies contain sub-graphs with many edges between the nodes within the same sub-graph and few edges between nodes within different sub-graphs. These nearly isolated cliques are termed *zones* within the activity topology. Figure 1 shows a recovered activity topology for a network of over a hundred cameras, with zones represented by circles.

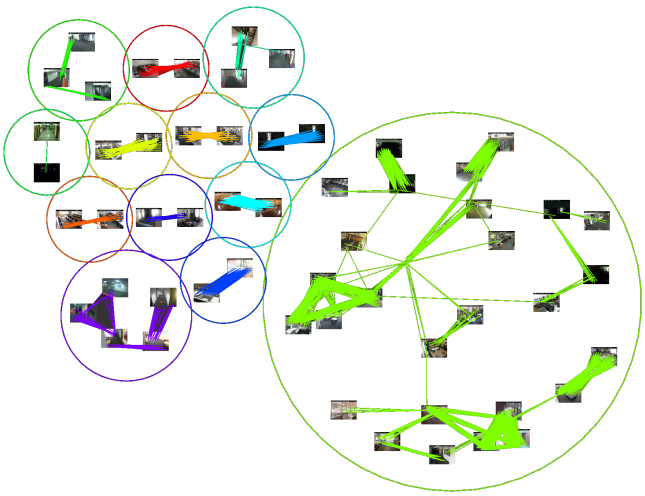

Fig. 1. Estimated activity topology for a real camera network. Edges linking cameras are shown as coloured lines, while zones are pictured as circles. Singletons are omitted.

2.2. Estimating Activity Topology by Exclusion

Consider the problem of determining overlap for a set of N cameras. The set of cameras generates N images at time t, with each image partitioned into a grid of cells. Application of Stauffer & Grimson foreground detection [7] to all camera images produces a set of foreground blobs, each of which can be summarised into a position given by a single cell within the containing camera. At any given time t, each cell is labelled *occupied* or *unoccupied* depending on whether it contains a summarised foreground object.

Exclusion is based on the observation that a cell which is *occupied* at time t cannot be an image of the same area as any other cell that is simultaneously *unoccupied*. Given that cells tend to be unoccupied more often than they are occupied, this observation can be used to eliminate a large number of cell pairs as potentially viewing the same area at each time instant. The process of elimination can be repeated for each frame of video to rapidly reduce the number of pairs of image cells that could possibly overlap. This is the opposite of most previous approaches: rather than accumulate positive information over time about overlap between cells, we seek negative information allowing the instant elimination of impossible overlaps. Such overlaps are referred to as having been *excluded* [4].

In this paper, the focus is on the performance of a new distributed implementation of exclusion to detecting overlap in large surveillance networks. Note however, that the technique is not limited to this special case of activity topology, but can also be applied to the general case (connections between non-overlapping cameras) through the use of varying time offsets in the operands to the exclusion operation. Future papers will evaluate this scenario.

3. PARTITIONING OF THE EXCLUSION MATRIX

Exclusion estimates camera overlap through the maintenance of an overlap certainty matrix, C_{ij}, which gives the likelihood that the regions of the scene corresponding to cells c_i and c_j overlap. The overlap estimate is further strengthened by exploitation of the bi-directional nature of overlap, we consider cells c_i and c_j to overlap only when the following Boolean function is true:

$$X_{ij} = C_{ij} > C^* \wedge C_{ji} > C^* \tag{1}$$

with C^* a threshold value. The effect of varying this threshold, in terms of the precision and recall achieved by the estimator, is extensively evaluated in [5].

The overlap certainty matrix is defined as follows:

$$C_{ij} = \frac{O_{ij} - E_{ij}}{O_{ij}} \tag{2}$$

where O_{ij} is the *exclusion opportunity matrix*, giving the number of times an exclusion contradicting overlap of cells c_i and c_j could possibly have been found, and E_{ij} is the *exclusion matrix*, giving the number of times an exclusion contradicting overlap of cells c_i and c_j has been found. The precise definition of O_{ij} is:

$$O_{ij} = \sum_{t=1}^{T} o_{it} \wedge v_{jt} \tag{3}$$

where v_{jt} is true if data for cell c_j at time t is available, and o_{it} is true if cell c_i is occupied at time t. Similarly, we define:

$$E_{ij} = \sum_{t=1}^{T} o_{it} \ominus p_{jt}. \tag{4}$$

where p_{jt} is true if cell c_j or any of its immediate neighbours are occupied at time j and $\ominus$ is an operator which has the value true if the left operand is true and the right operand is false. It is not necessary to record the v_{jt}, o_{it} and p_{jt} values in persistent matrices; instead we acquire and process this data in relatively short batches (a few seconds) and incorporate the results into the persistent O_{ij} and E_{ij} matrices.

In previous implementations of exclusion, these matrices are maintained in the memory of a centralised processing node. Furthermore, because these matrices are large and dense (at least in the case of E_{ij}), the memory available on this central node places an overall limit on the size of network that can be supported. For example, an instantiation of exclusion with 108 (12×9) cells per camera, 1000 cameras, and 16-bit (2 byte) exclusion counts will require:

$$(108 \times 1000)^2 \times 2 = 23,328,000,000$$

bytes (or approximately 24GB) to represent E_{ij}. Some optimisation is possible; for example, our previously reported implementation [6] used byte-sized counts and a selective reset procedure (division by two of sections of by E_{ij} and O_{ij}, so as to maintain approximately correct C_{ji} ratios) to support 1000 cameras within 12GB.

Nevertheless, there are two obstacles to further increases in the scale of systems that can be built using exclusion:

1. The requirement that the whole exclusion matrix be stored in a single server means that the memory (and processing) capacity of that server limits the maximum size of the networks that is feasible. For example, the current limit for easily affordable server hardware is less than 100GB, and only incremental improvements can be expected, so centralised implementations of exclusion are limited to supporting networks of a few thousand cameras.

2. The requirement for n^2 memory (however distributed) means that even if it is possible to increase system scale by the addition of more hardware, it becomes increasingly expensive to do so, and at some point it ceases to be feasible.

Both of these challenges need to be overcome; in this paper we focus on the first.

3.1. Partitioning Requirements

Our approach is to partition the exclusion computation across multiple computers. Each such computer is termed an *exclusion partition*. The aims are as follows:

1. Distribute the memory required to store the E_{ij} and O_{ij} matrices across multiple (affordable) computers.

2. Avoid communication (and in particular, synchronisation) between exclusion partitions, in order to permit processing within each partition to proceed in parallel.

3. Permit the system to grow, through addition of exclusion partitions, whilst the existing partitions continue processing and hence the extant system remains online.

4. Quantify the volume of communication required between the exclusion partitions and the rest of the surveillance system, and ensure that this requirement remains within acceptable bounds.

5. Size of partitions to be uniform.

3.2. Partitioned System Model

The role of exclusion within a surveillance system is to derive activity topology from occupancy. We adopt a layered approach, with an exclusion layer that consumes occupancy information (produced from a lower layer) and produces activity topology information (to be consumed by higher layers).

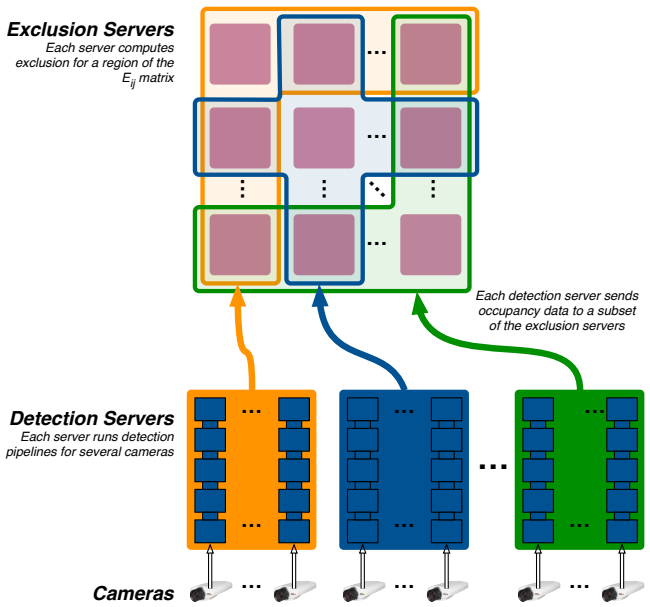

Fig. 2. Architecture of partitioned system model

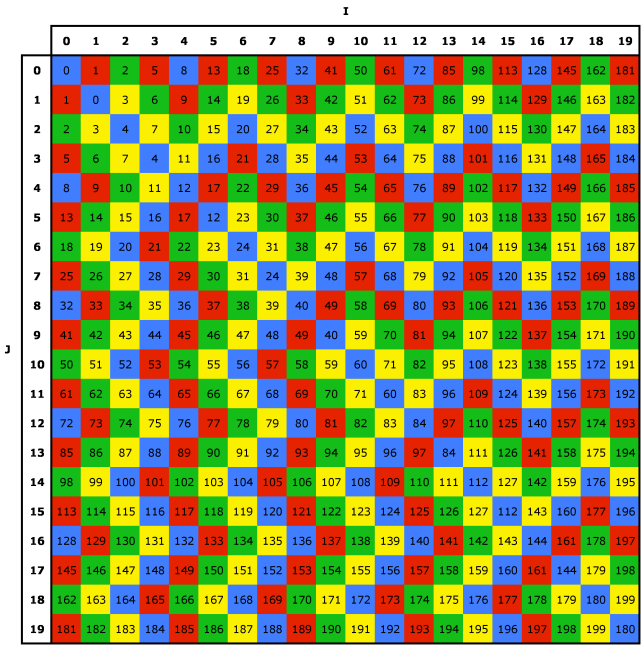

Fig. 3. The partitioning scheme for 200 partitions

This system model is shown in Figure 2, with the exclusion layer shown as a collection of exclusion partitions.

Topology estimates must be made available to higher levels of the surveillance system. The possibilities include:

1. As exclusion partitions derive topology estimates they forward significant changes in those estimates to a central database. These changes include both increases in likelihood of an edge in the topology and decreases in likelihood. In the extreme, this includes edges disappearing completely, reflecting changes in activity topology over time, and hence those edges being removed from the central topology database.

2. Option 1, but with the central database replaced with a distributed database.

3. Higher layers obtain topology information by querying the exclusion layer partition(s) holding it. In effect, the exclusion partitions act as a distributed database.

The experiments reported in this paper concern only the estimation of activity topology by exclusion, not any further use of the estimated topology within a surveillance system. This is closest to option 3, where topology data is stored only on exclusion partition nodes. Option 1 presents a simpler query model, but has the risk that the centralised database becomes a bottleneck. Note, however, that because the topology database only stores information for edges having a likelihood exceeding some threshold, it remains sparse with respect to the set of all possible edges, and thus avoids replicating the (problematic) requirement for n^2 memory on a single node. Option 2 is a trade-off between the other two options.

Parameter	Definition
n	the number of cameras.
N	the number of partitions.
r	the number of cells into which each camera's field of view is divided.
R	the length of a half partition, in terms of the number of distinct whole cameras for which that half partition contains data.

Table 1. Partitioned Exclusion System Parameters

3.3. A Partitioning Scheme

Observe from Equation 1 that calculation of overlap (X_{ij}) for given i and j requires both C_{ij} and C_{ji}, and hence (from Equation 2) each of E_{ij}, E_{ji}, O_{ij} and O_{ji}. Whilst it would be possible to perform the final overlap calculation separately from the calculation (and storage) of E_{ij} and O_{ij}, we assume that it is not practically useful to do so, which implies that for given i and j, each of E_{ij}, E_{ji}, O_{ij} and O_{ji} must reside in the same partition (so as to avoid inter-partition communication). This constraint drives our partitioning scheme, along with the aims identified previously.

Figure 3 shows partitioning across 200 exclusion partitions; each partition contains two distinct square regions of the E_{ij} matrix, such that the required symmetry is obtained. These square regions are termed *half partitions*. The O_{ij} matrix can be partitioned in the same way. However, given that the O_{ij} matrix contains significant redundancy (the O_{ij} values for all j in a given camera are the same), some optimisa-

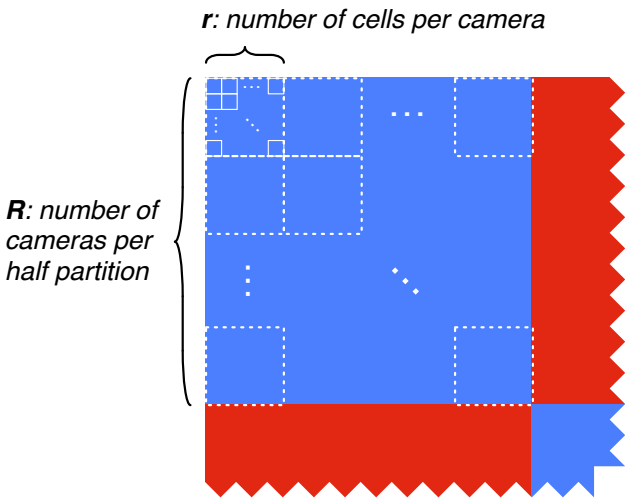

Fig. 4. Partitioned system parameter detail

tion is possible. Each of the square regions within Figure 3 contains sufficient rows and columns for several whole cameras worth of data. Table 1 defines the system parameters for partitioned exclusion, with Figure 4 illustrating the r and R parameters. Note also:

$$R = \frac{n}{\sqrt{2N}} \tag{5}$$

relates n, N and R where $N \in 2\mathbb{N}^2$ and $N \geq 2$.

Now, for a given (cell) co-ordinate pair (i, j) within E_{ij} we can determine the *partition co-ordinates*, (I, J) of the half partition containing the data for (i, j), as follows:

$$(I, J) = (\frac{i}{rR}, \frac{j}{rR}) \tag{6}$$

From the partition co-ordinates of a given half partition, (I, J), we determine the *partition number* of the (whole) partition to which that half-partition belongs, using the following recursively defined *partition numbering function*:

$$PN(I,J) = \begin{cases} PN(J,I) & if \quad J > I \\ PN(I-1, J-1) & if \quad I = J \wedge \\ & \quad I \bmod 2 = 1 \\ \left\lceil \frac{I^2}{2} \right\rceil + J & otherwise \end{cases} \tag{7}$$

This recursive function gives the partition numbers shown in Figure 3. More importantly, it is used within the distributed exclusion implementation to locate the partition responsible for a given region of E_{ij}. Detection pipelines producing occupancy data use Equation 7 to determine the partitions to which they should send that occupancy data, and clients querying the activity topology may use it to locate the partition hold the information they seek.

The inverse relation maps partition numbers to a set of two half partition co-ordinate pairs. This set, $\mathbb{P}$ for a given partition is:

$$\mathbb{P} = \left\{ (\overline{I}, \overline{J}), (\underline{I}, \underline{J}) : \overline{J} < \underline{J} \right\} \tag{8}$$

The co-ordinate pair $(\overline{I}, \overline{J})$ is termed the upper half-partition, and the pair $(\underline{I}, \underline{J})$ is termed the lower half-partition; they are distinguished based on the y axis co-ordinate, as shown.

Now, the x co-ordinate of the upper half-partition, $\overline{I}$, is a function of the partition number, P:

$$\overline{I} = \left\lfloor \sqrt{2P} \right\rfloor \tag{9}$$

and the y co-ordinate of the upper half-partition, $\overline{J}$, is a function of the partition number and the x co-ordinate:

$$\overline{J} = P - \left\lceil \frac{\overline{I}^2}{2} \right\rceil \tag{10}$$

Combining equations 9 and 10 yields the following, *upper half-partition address function*:

$$UHPA(P) = \left(\left\lfloor \sqrt{2P} \right\rfloor, P - \left\lceil \frac{\left\lfloor \sqrt{2P} \right\rfloor^2}{2} \right\rceil \right) \tag{11}$$

Next, the lower half-partition co-ordinate pair, $(\underline{I}, \underline{J})$, is a function of the upper half-partition co-ordinate pair, $(\overline{I}, \overline{J})$, as follows:

$$(\underline{I}, \underline{J}) = \begin{cases} (\overline{I}+1, \overline{J}+1) & if \ \overline{I} = \overline{J} \\ (\overline{J}, \overline{I}) & otherwise \end{cases} \tag{12}$$

Combining equations 9, 10 and 12 yields the following, *lower half-partition address function*:

$$LHPA(P) = \begin{cases} \left(\left\lfloor \sqrt{2P} \right\rfloor + 1, \left(P - \left\lceil \frac{\left\lfloor \sqrt{2P} \right\rfloor^2}{2} \right\rceil \right) + 1 \right) \\ \quad if \ \left\lfloor \sqrt{2P} \right\rfloor = P - \left\lceil \frac{\left\lfloor \sqrt{2P} \right\rfloor^2}{2} \right\rceil \\ \left(P - \left\lceil \frac{\left\lfloor \sqrt{2P} \right\rfloor^2}{2} \right\rceil, \left\lfloor \sqrt{2P} \right\rfloor \right) \\ \quad otherwise \end{cases} \tag{13}$$

Equations 11 and 13 define the partition co-ordinates of the two half-partitions corresponding to a given partition number. This is exploited in an implementation strategy whereby partition creation is parameterised by partition number, and this mapping is used to determine the two rectangular regions of E_{ij} to be stored in the partition.

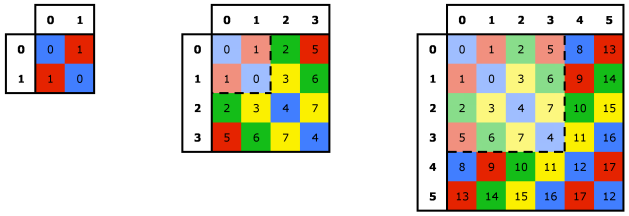

Fig. 5. Expansion from 2 to 8 and then to 18 partiions

Parameter	Definition
f	the number of frames per units time processed by the exclusion partitions.
d	the maximum time which occupancy data may be buffered prior to processing by the exclusion partitions.
b	the size of each exclusion count, in bytes.

Table 2. System Implementation Parameters

3.4. Incremental Expansion

A key property of the partitioning scheme is support for incremental expansion of E_{ij} and hence of the system. As shown in Figure 5, new partitions, with higher partition numbers, can be added on the right and bottom borders of the matrix, leaving the existing partitions unchanged in both partition number and content. Since the addition of new partitions is entirely independent of the existing partitions, expansion can occur whilst the system (*i.e.* the existing partitions) remains on-line.

Figure 5 shows expansion by two (partition) rows and (partition) columns each time. The matrix must remain square, and hence must grow by the same amount in each direction. The implication is that when growth is necessary, a large number of new partitions must be added (not just one at a time). Growth by two rows and columns at a time is the smallest increment that ensures all partitions are exactly the same size.

Growth by one row and column can lead to the latest partition on the diagonal being half the size of all the rest (it has only one half partition instead of two). At worst, this leads to under-utilisation of one computing node, and full utilisation will be restored at the next growth increment. It is also worth noting that whilst partitions have to be of fixed size, the mapping between partitions and computing nodes can be virtualised, allowing, for example, more recently added nodes, which are likely to have greater capacity, to be assigned more than a single partition.

Now, the number of partitions, N is expressed in terms of the length, L of the (square) partition grid:

$$N = \frac{L^2}{2} \tag{14}$$

Now suppose that at some point in its life time, a surveillance system has N partitions. Growth by two partitions in each direction results in partition grid length, $L + 2$, and hence the number of partitions in the system after growth, N', is:

$$N' = \frac{(L+2)^2}{2} = \frac{L^2 + 4L + 4}{2} = N + 2L + 2 \tag{15}$$

The growth in the number of partitions is then:

$$N' - N = 2L + 2 = \frac{2n}{R} + 2 \tag{16}$$

i.e. it is linear in the number of cameras in the system prior to growth.

4. ANALYSIS

Here we describe the expected properties for an implementation of distributed exclusion based on our partitioning approach. Section 5 evaluates the properties measured for a real implementation against the predictions made here. The properties of interest are:

- *Network bandwidth* – the input bandwidth for each exclusion partition and the aggregate bandwidth between the detection and exclusion layers.
- *Memory* – memory required within each partition.

Specifically, our aim is to relate these properties to the system parameters defined in Tables 1 and 2. Such relationships enable those engineering a surveillance system to provision enough memory and network hardware to prevent degradation of system performance, due to paging (or worse, memory exhaustion) and contention respectively, thus increasing the probability of the system maintaining continuous availability.

4.1. Network Bandwidth Requirements

The input bandwidth required by a partition is determined by the occupancy and padded occupancy data needed in the two half partitions constituting the partition. The occupancy data required in a half partition is that in the x co-ordinate range of E_{ij} which the half-partition represents. Similarly, the padded occupancy data required corresponds to the y-axis range. The size of the range in each case is the length (in cells) of a half-partition, that is:

$$l = Rr = \frac{nr}{\sqrt{2N}} \tag{17}$$

As with R in Equation 5, this is defined only where $N \in 2\mathbb{N}^2$ and $N \geq 2$.

Now, observe from Figure 3 that there are only two configurations of partitions:

1. For partitions on the diagonal of the partition grid, each of the two half-partitions occupies the same co-ordinate range in both x and y axes.

2. For all other partitions, the x co-ordinate range of one half partition is the y co-ordinate range of the other half-partition, and *vice versa*.

In both cases, a (whole) partition occupies a given (possibly non-contiguous) range in the x dimension and the same range in the y dimension, the total size of these ranges is $2l$. Given that padded occupancy, p_i, is computed from occupancy, o_k, for k in the set of cells including i and its immediate neighbours (within the same camera), all the padded occupancy data needed in a partition can be computed from the occupancy data that is also needed. Therefore the amount of data per frame needed as input into a partition is simply the amount of occupancy data, that is $2l$ bits. The unpartitioned case ($N = 1$) is handled separately: the number of inputs per frame is simply nr. With f frames per second this gives the partition's input bandwidth per second, β_P, in bits per second:

$$\beta_P = \left\{ \begin{array}{lll} nrf & if & N = 1 \\ 2lf = \frac{2nrf}{\sqrt{2N}} & if & N \in 2\mathbb{N}^2 \wedge N \geq 2 \end{array} \right. \quad (18)$$

Now, in practice, the information sent over the network will need to be encoded in some structured form, so the actual bandwidth requirement will be some constant multiple of β_P.

For N partitions, each on a separate host, the aggregate bandwidth per unit time, β_T, in bits per second, is:

$$\beta_T = \left\{ \begin{array}{lll} nrf & if & N = 1 \\ n\sqrt{2N}rf & if & N \in 2\mathbb{N}^2 \wedge N \geq 2 \end{array} \right. \quad (19)$$

4.2. Memory Requirements

Each exclusion partition requires memory for:

- Representation of two half partitions of the E_{ij} matrix.
- Representation of two half partitions of the O_{ij} matrix.
- Buffering occupancy data received from detection servers.
- Other miscellaneous purposes, such as parsing the XML data received from the detection servers.

Globally, the E_{ij} matrix requires one integer count for each pair of camera cells, (i, j). The number of cell pairs is r^2n^2, so the storage required for E_{ij} in each partition is:

$$\mu_E = \frac{r^2n^2b}{N} \quad (20)$$

The O_{ij} matrix is the same size as E_{ij}. However, for given i, O_{ij} for all j identifying cells within a given camera has the same value (as the cells identified by j are either all available or all unavailable at a given point in time), so the storage required for O_{ij} in each partition is:

$$\mu_O = \frac{rn^2b}{N} = \frac{\mu_E}{r} \quad (21)$$

The occupancy data processed within a given partition is produced by several detection servers. Hence it may be the case that different occupancy data pertaining to a given time point arrives at an exclusion partition at different times, and in fact some fraction of data (typically very small) may not arrive at all. To cope with this, occupancy data are buffered in exclusion partitions prior to processing. Double buffering is required to permit data to continue to arrive in parallel within processing of buffered data. Each data item requires at least two bits (to represent the absence of data as well as the occupied and unoccupied states). Using one byte per item, the storage required for buffering within a partition is:

$$\mu_B = 2d\beta_P = \left\{ \begin{array}{lll} 2dnrf & if & N = 1 \\ \frac{4dnrf}{\sqrt{2N}} & if & N \in 2\mathbb{N}^2 \wedge N \geq 2 \end{array} \right. \quad (22)$$

Finally, exclusion partitions require memory for parsing and connection management and for code and other fixed requirements, the storage required for parsing and connection management is proportional to the number of cameras processed by the partition, whereas the remaining memory is constant, so:

$$\mu_M = \left\{ \begin{array}{lll} n\mu_P + \mu_C & if & N = 1 \\ \frac{2n}{\sqrt{2N}}\mu_P + \mu_C & if & N \in 2\mathbb{N}^2 \wedge N \geq 2 \end{array} \right. \quad (23)$$

where μ_P and μ_C are constants determined empirically from a given implementation.

5. EVALUATION

5.1. Experimental Environment

Our experimental platform is a cluster of 16 servers, each with two 2.0Ghz dual-core Opteron CPUs and each server having 4 gigabytes of memory. We instantiate up to 32 exclusion partitions (of size up to 2GB) on this platform.

Results are reported for running distributed exclusion for surveillance networks of between 100 and 1,400 cameras and between 1 and 32 partitions. The occupancy data is derived by running detection pipelines on 2 hours footage from a real network of 132 cameras then duplicating occupancy data as necessary to synthesise 1,400 input files, each with 2 hours of occupancy data. These files are then used as input for the exclusion partitions, enabling us to repeat experiments. The use of synthesis to generate a sufficiently large number of inputs for the larger tests results in input that contains an artificially high incidence of overlap, since there is complete overlap within each set of input replicas. The likely consequence of this is that the time performance of exclusion computations is slightly worse than it would be in a real network.

Parameter	Value
N	either 1, 2, 8 or 32 partitions.
n(for $N = 1$)	100, 200 or 300 cameras
n(for $N = 2$)	100, 200 or 400 cameras
n(for $N = 8$)	200 to 800 cameras in steps of 200
n(for $N = 32$)	200 to 1400 cameras in steps of 200
r	each camera's field of view is divided into $12 \times 9 = 108$ cells.
R	determined from N and n.
f	10 frames per second.
d	at most 2 seconds buffering delay.
b	2 bytes per exclusion count.
μ_P	250 KB storage overhead per-camera.
μ_C	120 MB storage overhead per-partition.

Table 3. Experimental Parameters

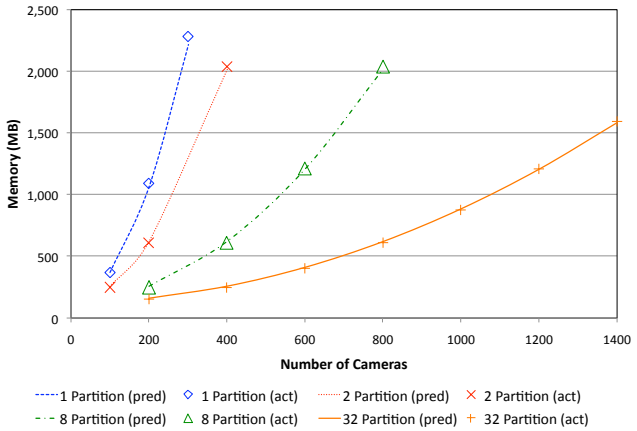

Fig. 6. Memory used within each partition

5.2. Verification

Results are verified against the previous, non-partitioned implementation of exclusion. It is shown in [5] that this previous implementation exhibits sufficient precision and recall of the ground truth overlap to support tracking. The partitioned implementation achieve very similar results for the same data. The differences arise because the distributed implementation choose a simpler approach to dealing with clock skew. Adopting the more sophisticated previous approach would be feasible in the partitioned implementation, and would not affect memory or network requirements, but would increase CPU time requirements.

5.3. Performance Results

The parameters for our experiments are shown in Table 3. Parameters n, N and (by implication) R are variables, whereas the remaining parameters have the constant values shown. The μ_P and μ_C constants have been determined empirically.

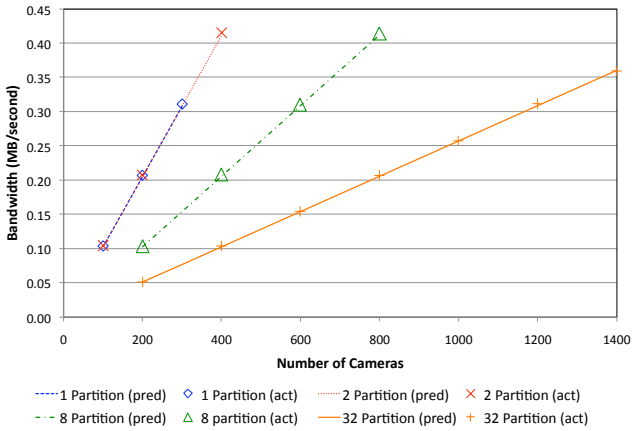

Fig. 7. Bandwidth into each partition

Figure 6 shows measurements of the arithmetic mean memory usage within each partition for the various configurations tested. Also shown are curves computed from the memory requirement formulae derived in Section 4.2. As can be seen, these closely match the measured results. The standard deviation in these results is at most 1.0×10^{-3} of the mean, for the 8 partition/200 camera case.

Figure 7 shows measurements of the arithmetic mean input bandwidth into each partition for the various configurations tested. Also shown are curves computed from the network bandwidth requirement formulae derived in Section 4.1, scaled by a constant multiple (as discussed earlier), which turns out to be 8. As one would expect, these closely match the measured results. Notice that the bandwidth requirements of the two partition case are the same as for the unpartitioned case: both partitions require input from all cameras. The standard deviation in these results is at most 3.7×10^{-2} of the mean, for the 32 partition/1200 camera case. The explanation for this variance (in fact any variance at all) is that we use a compressed format (sending only occupied cells) in the occupancy data.

Figure 8 shows measurements of the arithmetic mean CPU time within each partition for the various configurations tested. Recall that the footage used for experimentation is two hours (7,200 seconds) so all configurations shown are significantly faster than real time. The standard deviation in these results is at most 7.4×10^{-2} of the mean, for the 32 partition/200 camera case. Partitions in this case require a mean of 77 seconds CPU time for 7,200 seconds real-time, with the consequence that CPU time sampling effects contribute much of the variance. In contrast, the 32 partition/1400 camera case requires a mean of 1,435 seconds CPU time for 7,200 second real time, and has standard deviation of 1.6×10^{-2} of the mean.

At each time step, exclusion executes $O(n^2)$ exclusion tests (one for each pair of cells). Thus, we fit quadratic curves (least squares) to the measured data to obtain the co-efficients of quadratic formulae predicting the time performance for

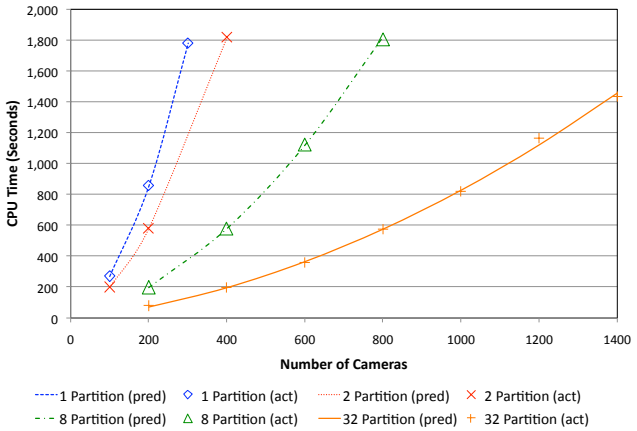

Fig. 8. CPU time used by each partition

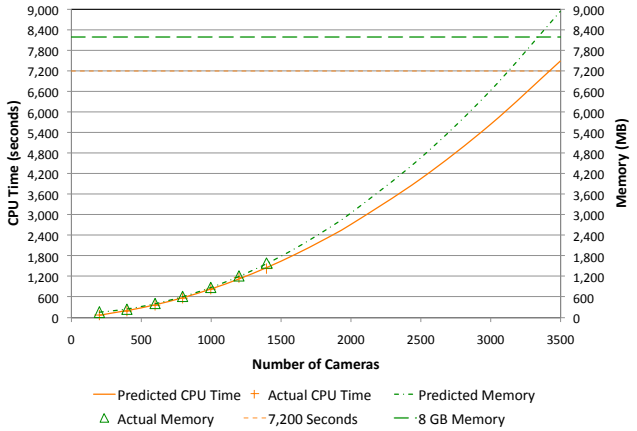

Fig. 9. Scale limit for 32 partition system

each distinct number of partitions. These formulae are shown as the predicted curves in Figure 8.

5.4. Discussion

Observe from Figures 6 and 8 that partitioning over just two nodes delivers significant increases over the unpartitioned implementation, at the cost of a second, commodity level, server. There is some additional network cost arising from partitioning, for example the total bandwidth required in the two partition case is twice that for the unpartitioned case with same number of cameras, however the total bandwidth required is relatively small in any case and thus is not expected to be problematic in practice. The total memory required for a given number of cameras is almost independent of the number of partitions, and the cost of this additional memory is more than outweighed by the ability for the memory to be distributed across multiple machines, thus avoiding any requirement for expensive machines capable of supporting unusually large amounts of memory.

The experiments validate the memory requirement formulae from Section 4.2 and (trivially) the network bandwidth formulae from Section 4.1. Using the memory formulae together with the empirically derived quadratic formulae fitted to the curves in Figure 8, it is possible to determine the current scale limit for exclusion to operate in real-time on typical commodity server hardware. We take as typical a server with 16 GB memory and 2 CPUs: the current cost of such a server is less than the cost of ten cameras (including camera installation). We instantiate two 8 GB partitions onto each such server. Figure 9 shows the predicted memory and CPU time curves for the 32 partition case extended up to 3,500 cameras. As can be seen, the memory curve crosses the 8 GB requirement at about 3,200 cameras, with the CPU time curve crossing 7,200 seconds at about 3,400 cameras, leading the conclusion that a 16 server system can support over 3,000 cameras; significantly larger scale than any previously reported results.

This paper focuses on the use of exclusion to estimate camera overlap, as it is only for this special case of activity topology that the accuracy of the approach has (as yet) been validated. Exclusion can be applied to general topology estimation, and produces results that seem reasonable (however, the extreme difficulty of obtaining ground truth has thus far prevented quantitative validation). When applied to the general case, the memory and processing requirements are constant multiples of those for overlap; thus extrapolation from results here is valid for the general case, in terms of prediction of resource requirements.

6. PREVIOUS WORK

Activity topology has typically been learnt by tracking people as they appear and disappear from camera fields of view (FOVs) over a long period of time. For example, in [8] the delay between the disappearance of each person from one camera and their appearance in another is stored to form a set of histograms describing the transit time between each camera pair. The system is demonstrated on a network of 3 cameras, but does not scale easily as it requires that correspondences between tracks are given during the training phase when topology is learnt.

Previous work by one of the authors [9] suggests an alternative approach whereby activity topology is represented by a Markov model. This does not require correspondences, but does need to learn n^2 transition matrix elements during a training phase and so does not scale well with the number of cameras n, due to the number of observations required for the Markov model. The training phase required in this and similar work is problematic in large networks, chiefly because the camera configuration, and thus activity topology, changes with surprising frequency; as cameras are added, removed, moved and fail. Approaches requiring a training phase to complete before operation would have to cease operation each

time there is a change, and only resume once re-training has completed. This is an intolerable restriction on the availability of a surveillance network. Instead, on-line automatic approaches, where topology is estimated concurrently with the operation of surveillance, are desirable.

Ellis et al. [10] do not require correspondences or a training phase, instead observing motion over a long period of time and accumulating appearance/disappearance information in a histogram. Instead of recording known correspondences, it records every possible disappearance that could relate to an appearance. Over time, actual transitions are reinforced and extracted from the histogram with a threshold. A variation on this approach is presented in [11], and has been extended by Stauffer [12] and Tieu et al. [13] to include a more rigorous definition of a transition based on statistical significance, and by Gilbert et al. [14] to incorporate a coarse to fine topology estimation. These methods rely on correctly analysing enough data to distinguish true correspondences, and have only been demonstrated on networks of less than 10 cameras.

The patented approach of Buehler [15] appears to scale to networks of about 100 cameras. It works by minimising a distance metric between images in cameras within a set (initially one camera?) having known field-of-view relationships and cameras whose relationships are as yet unknown, in order to assign relationships to the latter set. In operating thus, it supports growth of the network. In fact, it seems to rely on discovering field-of-view relationships incrementally, and it is unclear how this technique would cope with change in the underlying topology (which might invalidate the known set).

7. CONCLUSION

This paper reports experimental results for a decentralised and partitioned memory implementation of activity topology estimation by exclusion. This overcomes previous implementations' dependence on a single central server, and as a result provides a more cost effective approach to activity topology estimation for large surveillance networks. Results comparing partitioned and non-partitiioned exclusion demonstrate the advantages of partitioning outweigh the costs. The partitioning scheme enables partitions to execute independently; this both enhances performance (through increased parallelism) and, more importantly, permits partitions to be added without affecting existing partitions. Finally, formulae are derived for the network and memory requirements of partitioned exclusion. These formulae, verified by experimental results, enable engineers seeking to use exclusion to determine the resources required from the implementation platform.

8. REFERENCES

[1] "Personal communication, manufacturer of large surveillance systems," November 2007.

[2] G. Emerling, "D.c. police set to monitor 5,000 cameras," *The Washington Times*, vol. April 9, 2008.

[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. R. Dick, and R. Hill, "Activity topology estimation for large networks of cameras," in *Proceedings of the IEEE International Conference on Advanced Video and Signal-basedSurveillance*. November 2006, IEEE.

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

[6] 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*. September 2007, pp. 195–202, IEEE.

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

[8] O. Javed, Z. Rasheed, K. Shafique, and M. Shah, "Tracking across multiple cameras with disjoint views," in *IEEE Int. Conf. Computer Vision*, 2003, pp. 952–957.

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

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

[11] T. H. Ko and N. M. Berry, "On scaling distributed low-power wireless image sensors," in *Proc. 39th Annual Hawaii International Conference on System Sciences*, 2006, vol. 09, p. 235.

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

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

[14] A. Gilbert and R. Bowden, "Tracking objects across cameras by incrementally learning inter-camera colour calibraion and patterns of activity," in *European Conference on Computer Vision*, 2006, vol. 2, pp. 125–136.

[15] C. J. Buehler and Intellivid Corporation, "Computerized method and apparatus for determining field-of-view relationships among multiple image sensors," *United States Patent 7286157*, 2007.