MULTI-CAMERA TOPOLOGY RECOVERY FROM COHERENT MOTION

Zehavit Mandel *Ilan Shimshoni* *Daniel Keren*

Dept. of Computer Science Dept. of Management Information Systems Dept. of Computer Science
University of Haifa, Israel

ABSTRACT

Given a surveillance system with many cameras, which cover a wide and complex area, it is important to know the camera network topology. I.e., which cameras have overlapping fields of view. This is useful for inferring 3D structure and tracking. The computational model assumed in this paper is that each camera has its own computing unit able to perform simple processing operations and is connected via a communication network to all the other cameras. Due to the similar nature of the scenes photographed by the cameras, it might be hard to compute the overlap by matching features. This paper therefore suggests to accomplish the task automatically, using a distributed algorithm. Each camera detects motion locally and transmits the detected motion position to the other cameras. The overlap is detected by searching for correlations defined by simultaneous activity in image regions. The areas of these regions are chosen so that they optimize the number of frames required to determine whether there is an overlap and to reduce the number of false decisions. Each camera determines the number of regions based on the amount of motion detected in its field of view. The algorithm has been implemented and tested both in simulated and real multi-camera image sequences.

1. INTRODUCTION AND MOTIVATION

The concept of cooperative multi-camera system, informally a "forest" of sensors, has recently received increasing attention from the research community. The idea is of great practical relevance, since cameras typically have limited fields of view, but are now available at low cost. Thus, instead of having a high resolution camera with a wide field of view that monitors a wide area, far greater flexibility and scalability can be achieved by observing a scene "through many eyes", using a multitude of lower resolution COTS (commercial off-the-shelf) cameras.

Object tracking is an example of an application in a multiple camera environment. This kind of application requires information about the camera network topology. Usually this information is entered manually to the system. In this paper we suggest a method for learning the camera network topology automatically without the need of manual intervention, which is both laborious and error prone. The human user can also not be aware of changes which happen to the network topology which can be caused by camera movement and changes in the environment which can cause changes in the visibility of objects in the scene. Thus, an automatic network recovery procedure should be developed that can be run whenever needed.

There are a number of approaches we might use to automatically determine the connectivity or spatial adjacency of the camera network. An obvious method would be to extend the single camera tracking algorithm across multiple views. However in a busy environment where many objects are being tracked simultaneously, the main problem lies in establishing the correct correspondence. An alternative solution would be to calibrate the system by having only a single target move at a steady pace around all the typical paths in the environment, allowing the system to learn all the connections without ambiguity, as was used in [1].

Whilst such an approach is attractive in its simplicity, it has several drawbacks for a real surveillance system. Firstly, it requires an explicit setup phase in which this calibration is undertaken. If any of the cameras moves, the system will need recalibration. Secondly, it can not be applied to pre-recorded video data, where there is no access to the site prior to the video analysis.

Ellis, Makris and Black in [2] use detection of entry and exit zones in the camera's view fields in order to learn about the network topology. They adopt a two stage algorithm for detecting connected zones – first detecting exit and entry zones, then temporally "correlating" the disappearance and reappearance of tracked objects between zones. The entry and exit zones correspond to regions in the image where the majority of targets appear in or disappear from the camera FOV. They use statistical analysis to identify the topology; this analysis is based on trajectory data derived from single tracking modules attached to each camera of the system.

Khan, Omar and Shah in [3] find the FOV lines of the cameras. They employ the novel approach of finding the limits of the field of view of a camera as visible in the other cameras. When multiple people are in the scene and if someone crosses the edge of the FOV, all persons in the other cameras are picked as being candidates for the projection of an FOV line. Since the false candidates are randomly spread on both sides of the line whereas the correct candidates are clustered

1-4244-1354-0/07/$25.00 ©2007 IEEE

on a single line, correct correspondences will yield a line in a single orientation, but the wrong correspondences will yield lines in scattered orientations. They then use the Hough transform to find the best lines.

Cheng and Piccardi in [4] present a track matching algorithm based on the "major color" histograms matching and the post matching integration. By using the color distance and a given threshold, all pixels from a moving object $MO_{i,t}$ in a given frame t are clustered into a limited number of colors, with each color's frequency defined as the number of pixels with that color. Such colors are then sorted in descending frequency order and the first k used to represent the moving object. Given two arbitrary moving objects, $MO_{i,t}$ and $MO_{j,u}$ from two different frames t and u, a similarity criterion based on the major color representation is used to assess their matching.

Our algorithm makes the following assumptions about the settings of the system. Each camera has a processor which is able to perform image processing calculations and communicate with the other cameras; the communication bandwidth is limited; the image features detected by the cameras are not enough in themselves to compute the network topology; while the algorithm is running the camera locations are fixed.

Under these constraints the following approach is proposed. Each camera sends messages to the other cameras when it detects motion in its FOV. By receiving the messages from the other cameras in the network, each camera learns whether their FOVs overlap. From a probabilistic analysis developed in the paper we show that it is advantageous to divide the FOV of each camera into several regions and compute the optimal size of each region, depending on the amount of movement in the camera's FOV. The analysis is then performed on each pair of regions independently and from the results we infer the topology of the camera network.

The paper continues as follows. In Section 2 we develop our general approach and describe its various components. In Section 3 we present simulations and results on image sequences. Finally in Section 4 we present conclusions and future research directions.

2. THE ALGORITHM

Consider a network of cameras where each camera is equipped with its own processor. The goal of the algorithm is to determine the network topology correctly while keeping the network communication load low. We adopt a two stage algorithm for detecting connected cameras. First, each camera runs a background subtraction algorithm in order to detect motion in its view field (after learning its background). When motion is detected it is transmitted to the other cameras. The second stage of the algorithm is run in parallel. The camera receives the motion information from the other cameras and tries to correlate the motion events with its own detected motion using the Sequential Probabilistic Ratio Test (SPRT). Each camera maintains data about the other cameras in the network; this data includes the total number of messages it received from each other camera and the number of success messages (success means that camera C_i receives a message from camera C_j about motion in frame t, and camera C_i also detects motion in frame t). When C_i receives a message from C_j it updates its data and runs the SPRT statistical model test. The result of this test is either that C_j is a neighbor of C_i, or it is not a neighbor, or that at this stage a decision can not be taken with the required certainty. When camera C_i makes a decision it sends it to camera C_j, so cameras C_j and C_i can stop sending messages to each other.

While the cameras are learning the network topology many people may be walking through the scene simultaneously. This results in accidental simultaneous events. This can result in mistaken decisions made by the SPRT or will require extremely long image sequences to make a decision. We therefore exploit the fact that the overlap between the FOVs of a pair of cameras is usually contiguous. We divide the image into several regions and perform the algorithm on pairs of regions instead of pairs of images. Under this setting with high probability at least one pair of regions will have a large relative overlap enabling the SPRT to make the correct decision quickly. The optimal size of the regions is computed by each camera based on the amount of traffic in its FOV.

In the following subsections we describe the various components of the system. We will first review the motion detection we used which is our implementation of the codebook method presented in [5]. We then review the SPRT method. Finally we present our criterion to optimally choose the size of the regions in the image which minimizes the number of frames required to be analyzed by the algorithm until a correct decision is made.

2.1. Movement detection

Each camera in the network runs the background subtraction algorithm in order to detect motion in its FOV. Background subtraction is a commonly used class of techniques for segmenting out objects of interest in a scene for applications such as surveillance. It involves comparing an observed image with an estimate of the image constructed as if there are no objects of interest present. The image regions where there is significant difference between the observed and estimated images indicate the location of the object of interest.

We chose to implement the movement detection algorithm by using the codebook model described in [5]. In the rest of this section we present a short review of the codebook algorithm.

The codebook has the following key features:

1. An adaptive and compact background model that can capture structural background motion over a long period of time under limited memory.

2. It can handle local and global illumination changes.
3. Unconstrained training that allows moving objects in the scene during the initial training.
4. Multiple background layers.

Codebook algorithm:
For each pixel, the algorithm builds a codebook consisting of one or more codewords. We use the N first frames in order to build a codeword list for every pixel. We select the codewords with the highest frequency. Each codeword contains the RGB vector and the following fields:

- I_{min}, I_{max}: The min and max brightness, respectively, of all pixels assigned to this codeword.
- f: the frequency with which the codeword has occurred.
- λ : the maximum negative run-length (MNRL) defined at the longest interval during the training period.
- p, g : the first and last access times, respectively, in which the codeword was present.

Each codeword represents a cylinder in RGB space, illustrated in Figure 1. During runtime each pixel is checked to see whether its RGB vector lies within one of its codewords. Pixels which do not belong to any codeword assigned to them are declared foreground pixels. When a large number of pixels are determined to be foreground pixels, the frame is declared to include motion, and the bounding boxes of the moving objects are returned by the algorithm.

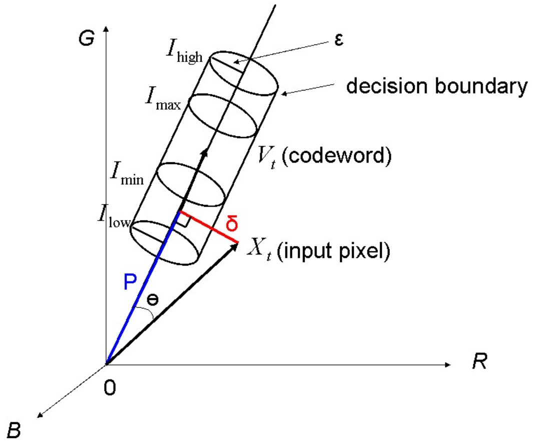

Fig. 1. Codebook illustration

2.2. Statistical Model

The image of every camera in the network is divided into several (possibly overlapping) regions. Whenever motion is detected in the image, the camera registers in which region the motion occurred, and sends this information to the other cameras in the network. Each camera maintains a data structure such as the one shown in Table 1 for each of its regions.

Camera ID	Region ID	Number of messages from camera	Success messages
C1	Region1	100	30
C1	Region2	85	28
...	...	...	...
C10	Region3	105	42
...	...	...	...

Table 1. Data structure used by each camera

When a camera detects motion, it sends a message with the time stamp, the camera ID, and the region ID where the motion occurred, to the other cameras in the network. When a camera receives such a message, it increases the number of messages for the camera that sent the message. It then checks if it also detected motion in this frame, and in which regions. When the motions occurred simultaneously, it increases the number of success messages. After the data structure has been updated, it checks if it can decide about the location of the other camera (neighbor or not) by using the Sequential Probability Ratio Test (SPRT) statistical model [6].

SPRT was designed under the assumptions that sampling is both expensive and time consuming; hence there are situations where it is more efficient to take samples sequentially, as opposed to all at once, and to define a stopping rule to determine the sampling process. At each stage, the data is assessed and only in when it is ambiguous do we continue sampling. The number of observations used by the SPRT is a random variable for fixed “alarm” and “miss” error probabilities. It is known that the SPRT consumes the least average number of samples to detect a hypothesis among all other hypothesis tests, for fixed false alarm and miss probabilities. The idea behind the sequential testing is that we collect observations one at a time, and when an observation has been made, we choose between the following options:

- Accept the null hypothesis and stop the observation.
- Reject the null hypothesis and stop the observation.
- Defer decision until we have collected more information.

We want to test the relationships between each pair of cameras using the ratio of simultaneous motion events with respect to the total number of motion events. When the view fields of the two cameras overlap, the average probability of simultaneous motion events is P_0, and when they do not overlap the average probability is P_1. Thus, we can transform these relationships into the respective equivalent hypotheses: $H_0 : p_0 = P_0$ and $H_1 : p_1 = P_1$. Define the Probability

Ratio PR as that of the binomial distribution, under H_0 and H_1:

$$\frac{P(y|H1)}{P(y|H0)} = \frac{Binomial(y;n,p_1)}{Binomial(y;n,p_0)} =$$

$$\frac{Kp_1{}^y(1-p_1)^{n-y}}{Kp_0{}^y(1-p_0)^{n-y}} = \frac{p_1{}^y(1-p_1)^{n-y}}{p_0{}^y(1-p_0)^{n-y}}$$

Now, we have to define the error probabilities α – the probability of rejecting the pair and deciding they are not neighbors although they are, and β – the probability of accepting the pair and deciding they are neighbors, although they aren't. Thus, we define $S(A,B)$, the "Sequential Probability Ratio Test" by the preceding equations, as one that compares PR with the values A and B at every stage n and decides :

1. accept H_0 if $PR < B$;
2. accept H_1 if $PR > A$
3. continue testing if $B < PR < A$.

We can simplify the equations by taking logarithms:

$$B < \frac{p_1{}^y(1-p_1)^{n-y}}{p_0{}^y(1-p_0)^{n-y}} = (\frac{p_1}{p_0})^y(\frac{1-p_1}{1-p_0})^{n-y} < A$$

$$\ln(B) < an + by < \ln(A)$$

where $a = \ln(\frac{1-p_1}{1-p_0})$ and $b = \ln(\frac{p_1}{p_0}) - \ln(\frac{1-p_1}{1-p_0})$.

The coefficients a and b of these equations are obviously, functions of p_0 and p_1. It can be shown (see [6, 7, 8]) that the constants A and B can be approximated by:

$$A \cong \frac{1-\beta}{\alpha}; B \cong \frac{\beta}{1-\alpha}.$$

To better track the SPRT test values, we need an equation comparing y, the number of "successes":

$$\begin{array}{rcl} \ln(B) < & an+by & < \ln(A) \\ \frac{\ln(B)}{b} - \frac{a}{b}n > & y & > \frac{\ln(A)}{b} - \frac{a}{b}n \\ h_1 + sn > & y & > h_0 + sn, \end{array}$$

where $h_1 = \frac{\ln(B)}{b}$ and $h_0 = \frac{\ln(A)}{b}$ and the slope $s = \frac{a}{b}$. The smaller the errors (α, β), the larger the intercepts (in absolute value). The result is intuitively appealing, since when we demand strong assurances (smaller errors) from the SPRT test procedure, the SPRT will necessarily require more information to be able to provide a decision that fulfills the error bounds α and β (see Figure 2).

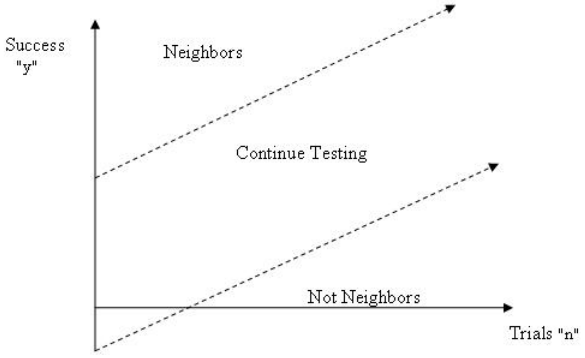

Fig. 2. Representation of SPRT

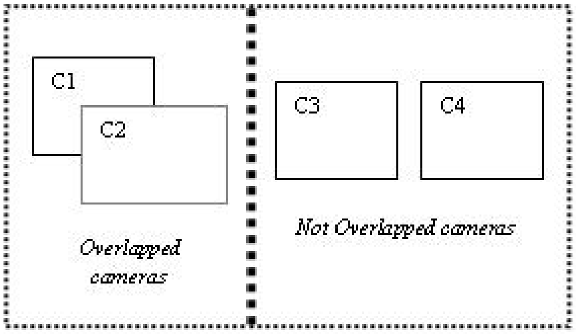

Fig. 3. Camera pair possible relationships

2.3. Optimal number of regions to split

Two cameras can have two kinds of relationships between them, as illustrated in Figure 3: overlapping and non-overlapping. The SPRT model is a natural candidate for determining the true relationships between the cameras due to the sequential nature of the collected evidence.

In our case the SPRT model has two inputs:

1. P_1 – the probability to have real simultaneous motion (e.g. motion having a common source) in two overlapped cameras.
2. P_0 – the probability to have accidental simultaneous motion in two non-overlapping cameras.

The values of these parameters depend on two additional basic values:

1. P – the probability of motion in a camera's field of view.
2. P_{ov} the minimum overlap area between two cameras, relative to the total view field.

Naturally, the values of these parameters have a strong effect on the number of frames that have to be analyzed by the

SPRT before it returns an answer. As the number of people moving in the scene increases, P increases, which causes an increase in the number of accidental simultaneous motions. This results in a longer run of the SPRT. On the other hand, if the overlap between the view fields is large, most simultaneous motion events will be caused by the same person moving in the scene, reducing the time required for the SPRT to reach a decision.

In a real application, both P and P_{ov} are not controlled by the user. We can however optimize their values by dividing the images into several regions and running the SPRT on pairs of regions instead of pairs of images. The result is the reduction in the probability of motion in a region and, for a small number of pairs of regions in the images of the two cameras, an increase in the relative overlap between them. So, if one pair of regions decides that they overlap, this results in the decision that the two images overlap. On the other hand, only when all region pairs belonging to a pair of images decide that they do not overlap, do we decide that the cameras do not overlap.

The main question which needs yet to be answered is: what is the optimal size of the regions which will yield the most accurate result in the shortest time?

Under the simplified assumption that motion occurs uniformly within the image, the area of a region is proportional to the probability of motion in it, which we shall denote P_m. Assuming that we estimated P for each camera by measuring the number of motion events in a short amount of time, the area of the region can be computed. Thus, all that remains is to estimate the optimal value of P_m under the requirement that if the overlap between the regions will be at least P_{ov}, the algorithm will terminate after a minimal number of frames.

The relationships between P_m and P_{ov}, and P_1 and P_0 are as follows:

$$P_1 = 1 - {P_{ov}}^2 P_m^2 + P_m P_{ov}. \qquad (1)$$

The first part of equation 1 is the probability for motion in non-overlapping regions and the second part is the probability for motion in overlapping regions.

$$P_0 = {P_m}^2. \qquad (2)$$

Under the binomial distributions for probabilities
$\mu_0 = P_0$ and $\mu_1 = P_1$ their standard deviations are
$\sigma_0 = \sqrt{\frac{P_0(1-P_0)}{N}}$ and $\sigma_1 = \sqrt{\frac{P_1(1-P_1)}{N}}$ respectively, where N is the number of frames.

For a given N, a measure of separation k_σ is defined by

$$k_\sigma = \frac{P_1 - P_0}{\sigma_1 - \sigma_0}.$$

Conversely, for a given k_σ the goal is to find P_m which yields the minimal N.

$$P_1 - P_0 = k_\sigma \left(\sqrt{\frac{P_1(1-P_1)}{N}} + \sqrt{\frac{P_0(1-P_0)}{N}} \right) \qquad (3)$$

Thus,

$$N = \left(\frac{k_\sigma}{P_1 - P_0} (\sqrt{P_1(1-P_1)} + \sqrt{P_0(1-P_0)}) \right)^2 \qquad (4)$$

Figure 4 depicts N as a function of P_m for $P_{ov} = 0.8$ and for a certain value of k_σ. The value of k_σ affects the value of N but does not change the optimal P_m.

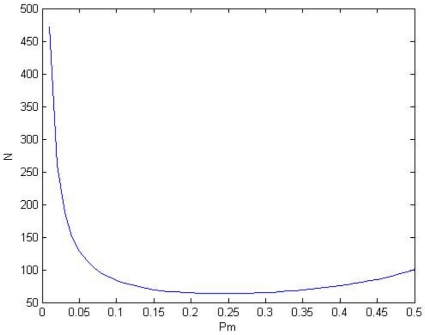

Fig. 4. Minimal number of frames, optimal P_m

After we have computed the optimal P_m we can decide for each camera, depending on P which was estimated from the motion detected in its FOV, the measure P_m/P of each region. The motion uniformity assumption in the image can be easily dropped by defining regions in which the empirical probability of motion is P_m.

3. RESULTS

3.1. Simulation

We implemented the topology detection algorithm and tested it first in a simulated environment, which consists of a square arena, in which simulated people follow a random walk pattern. Several cameras, with rectangular view fields, monitor the arena. In order to make the simulation more realistic we incorporated a random error into the motion detection process (if a person is in the camera field of view, with a probability of 0.05 we assume the system fails to detect the person). Figure 5 shows the camera configuration for which the following results are presented. In this configuration overlap exists only between C1-C2,C2-C3,C3-C4 and C4-C5.

In the first experiment we tested the veracity of our analysis of the optimal region area. The algorithm was run with three moving people. We ran the same experiment, each time changing the number of regions. Our algorithm chose to divide the view fields into 2 or 3 regions (See Table3(a)). The results are presented in Table 2. As can be seen, when the view field is not divided, the running time is short, however the algorithm decides that every pair of cameras is connected. On the other hand, when dividing into a number of regions larger than the optimal, the decision is correct, but the running time is considerably longer. Comparing the running times of the non-overlapping pair C2-C5, with the overlapping ones, we see that usually the running time of the non-overlapping pair is higher. This is because for a camera pair to be declared non-overlapping, all region pairs belonging to the two cameras have to decide that they are non-overlapping.

In another experiment we compared the performance of the algorithm when three and four people are moving in the environment. As can be seen in Table 3(a), the increase in probability of motion P results in the division of the view field into more regions. On the other hand the running times shown in Table 3(b) do not change consistently, suggesting that the algorithm's running time is not affected by the amount of motion.

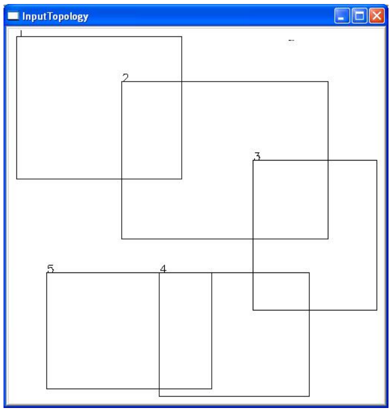

Fig. 5. Camera topology

In another experiment we look for the maximal number of people moving in the environment for which the algorithm still works. As can be seen in Figure 6 the time to get the correct results from the simulation using one to seven people moving in the environment is quite similar. This is not surprising because when there are more people in the environment we divide the camera view field into more regions, as can be seen in Table 4. As a result each region's behavior is invariant to the number of people moving within the environment.

When more than seven people are moving around in the environment, the algorithm begins to make mistakes (pairs of cameras that are not neighbors are considered to be neighbors by the algorithm). This is a result of more hypotheses being tested and even though the probability for a false positive is low, these rare events do occur. These false positive events are avoided by waiting for more than one pair of regions in the camera's view field to be considered related before declaring that the FOVs of the camera pair overlap. As in this case the camera's FOV is divided into more regions, we can assume that the overlap between the cameras will consist of more than one pair of regions. In our experiments two overlap events are required when eight or more people are moving in the environment and three events are required when eleven to thirteen people are moving in the environment. This comes of course at an increase in the number of frames required for the algorithm to reach a decision.

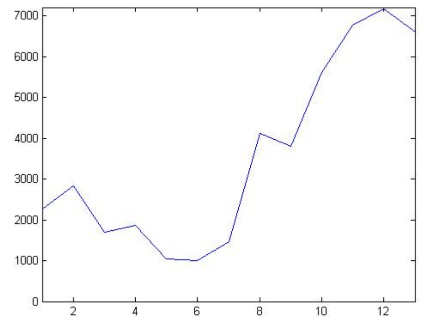

Fig. 6. Different number of people

No. of People	1	2	3	4	5	6	7
No. of Regions	2	2	3	3	4	5	5

Table 4. Number of regions to divide camera's view field

3.2. Lab results

In our experiments we used a network of three cameras of type VIVOTEK IP7135 denoted C1, C2 and C3. In our setup there was an overlap between the view fields of C1 and C2, and C2 and C3 (as can be seen in Figure 7 with the results of the background subtraction algorithm). To make the experiment more challenging we duplicated the motion outputs from the three cameras five times with small time shifts between them, thus adding many more non-overlapping camera pairs to the experiment. The algorithm was able to detect the network topology correctly. Figure 8 shows the motion detection results of the three cameras and when simultaneous motion occurred between camera pairs. For overlapping pairs of camera view fields (C1-C2 and C2-C3) the number of simultaneous motion events is much higher than between the non-overlapping camera pair C1-C3, even though accidental simultaneous motion events do happen. The algorithm exploits this information to make its decision. The times in seconds it took for several camera pairs to learn the network topology are given in Table 5.

Number of Regions	C_1-C_2	C_2-C_3	C_3-C_4	C_4-C_5	C_2-C_5	mistakes
Overlap?	Yes	Yes	Yes	Yes	No	
One region	293	450	1136	459	Overlap	6
Two Regions	421	655	1765	462	768	0
Our Division	578	655	1517	462	857	0
Six Regions	1103	1090	2482	968	907	0
Nine Regions	815	2886	2325	1358	1200	0

Table 2. Running times for different region divisions

	3 People		4 People	
	Probability of movement	regions	probability of movement	regions
C_1	0.4302	3	0.5236	3
C_2	0.5622	3	0.6542	4
C_3	0.3532	2	0.4278	3
C_4	0.3432	2	0.4358	3
C_5	0.3628	2	0.4384	3

(a) Estimates of P and the number of regions

	3 People	4 People
C_1-C_2	557	459
C_2-C_3	552	608
C_3-C_4	1564	950
C_4-C_5	570	369
C_1-C_4	1112	1864

(b) Running times

Table 3. Different number of people

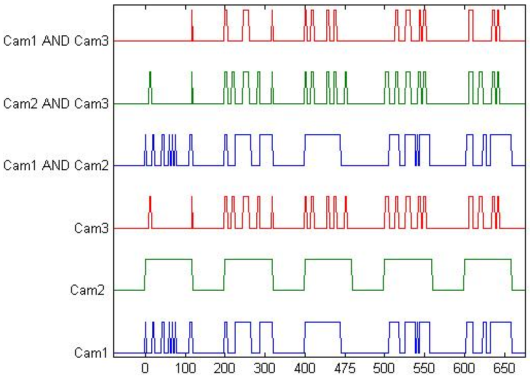

Fig. 8. A graph showing the motion detection performed by three cameras in the network as a function of time. Motion is represented as a high value. The graph also shows the combinations of every pair of cameras.

4. CONCLUSIONS

We presented an algorithm that uses well-established decision making techniques to reliably decide which camera pairs in a surveillance system share a field of view. The process is automatic. Further, the suggested solution minimizes the required communication, a critical issue in sensor networks. This is achieved by dividing the images to regions whose area is optimized with respect to the duration of the decision making process. The algorithm was successfully tested in both simulated and real environments.

5. REFERENCES

[1] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, "M-KNIGHT: a real time surveillance system for multiple overlapping and non-overlapping cameras," in *IEEE Conference on multi media and expo*, 2003.

[2] T. Ellis, D. Makris, and J. Black, "Learning a multicamera topology," in *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2003, pp. 165–171.

[3] S. Khan, O. Javed, and M. Shah, "Tracking in uncalibrated cameras with overlapping field of view," in *PETS*, 2001.

[4] E.D. Cheng and M. Piccardi, "Track matching by major color histograms matching and post-matching integration," *Image analysis and processing*, vol. 3617, no. 13, pp. 1148–1157, 2005.

[5] K. Kim, T.H. Chanlidabongse, D. Harwood, and L.S. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, June 2005.

[6] A. Wald, *Sequential Analysis*, Wiley, New York, 1947.

[7] A.J. Duncan, *Quality Control and Industrial Statistics*, Irwin, fourth edition, 1974.

[8] W.E. Wallace and C. Sontz, "MIL-HNDB-781D and MIL-STD 781D, reliability testing for engineering development, qualification, and production, DoD," October 1986.

C1-C2	C2-C3	C4-C5	C5-C6	C7-C8	C8-C9	C10-C11	C11-C12	C13-C14	C14-C15
246	910	983	1184	901	985	919	1280	326	952

Table 5. Time to obtain the network topology (seconds)

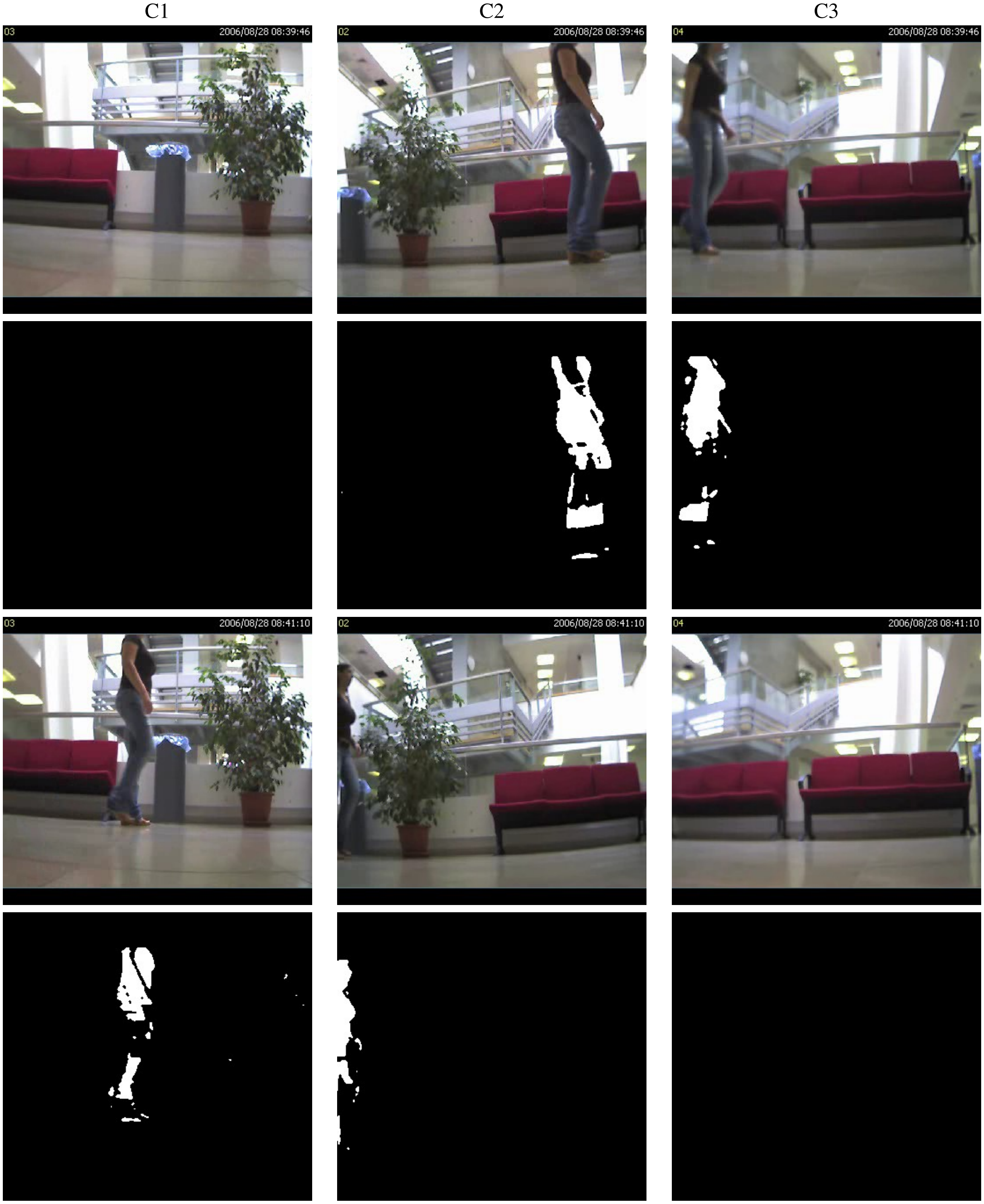

Fig. 7. Motion detection results: