Collaborative Multi-Camera Surveillance with Automated Person Detection

Trevor Ahmedali and James J. Clark
Centre for Intelligent Machines, McGill University, Montreal, H3A 2A7, Canada
{tahmeda, clark}@cim.mcgill.ca

Abstract

This paper presents the groundwork for a distributed network of collaborating, intelligent surveillance cameras, implemented with low-cost embedded microprocessor camera modules. Each camera trains a person detection classifier using the Winnow algorithm for unsupervised, online learning. Training examples are automatically extracted and labelled, and the classifier is then used to locate person instances.

To improve detection performance, multiple cameras with overlapping fields of view collaborate to confirm results. We present a novel, unsupervised calibration technique that allows each camera module to represent its spatial relationship with the rest. During runtime, cameras apply the learned spatial correlations to confirm each other's detections. This technique implicitly handles non-overlapping regions that cannot be confirmed. Its computational efficiency is well-suited to real-time processing on our hardware.

1. Introduction

The increased need for security in public and private areas has augmented the reliance on video surveillance systems as a cost-effective way of monitoring large areas with few security personnel. The nature of this work (monitoring multiple video feeds), however, can be tedious for the security guards involved, since it is mostly uneventful. This tedium, combined with the necessary multitasking, can make the observer prone to error. It is therefore desirable to automate as much of this process as possible, using an intelligent system to monitor the video feeds and report activity to a human operator.

The ability to detect people in an image is vital for automated video surveillance, as well as for many other real-world computer vision applications. The results of person detection can be used by higher-level tasks such as person tracking, event logging, content analysis, and low-rate video compression. In this work, the focus is on person detection itself, but its output can be used for any of the aforementioned applications.

Since areas requiring surveillance are typically larger than a single camera's field of view (FOV), surveillance setups typically use multiple cameras. While a setup composed of multiple, independent single-camera intelligent surveillance systems, such as [4, 15], would be an improvement over simple video feeds, even more benefits can be achieved by combining the sensors into a collaborative multi-camera surveillance system [2].

This paper presents a design for an intelligent multi-camera surveillance system capable of person detection and suitable for real-world applications. Person-detection is achieved with a machine-learning approach, based on work by Nair [10]. The system uses a parallel, distributed network of cameras, appropriate for wide-area surveillance. This system architecture implies several requirements that guide our design:

1) Real-time processing: In a security context, results must be available in a timely fashion. This requires computationally efficient techniques.

2) Low-cost hardware: A wide-area surveillance system can include tens or even hundreds of cameras, so each camera module uses low-cost, commercially available components.

3) Distributed processing: A distributed system scales easily to a large number of cameras. Our system therefore performs all processing on the individual camera modules, instead of a centralized server. This also eliminates the need to transmit images over the network: only high-level information, such person locations, is communicated. The lower bandwidth requirements also favour scalability.

4) Easy setup and calibration: Since this camera network is designed for wide-area surveillance, the initial setup of such a system is designed to be as simple as possible. Camera modules use a wireless network to maximize placement options. An easy setup also implies unsupervised learning: otherwise, each different camera view would require its own manually-labelled training set to account for differences in viewpoint, lighting, and camera properties. The

Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06)
0-7695-2542-3/06 $20.00 © 2006 IEEE

calibration for multiple cameras to learn their spatial relationships is also designed to be as simple as possible to perform.

5) Adaptability: The system must adapt to changes in people's appearance over time. In winter, for example, people might be wearing heavy garb, which alter their appearance and shape compared to what is seen in summer.

6) Multi-camera collaboration: By collaborating, the surveillance system can improve person detection accuracy, reducing the number of false positives and handling occlusion.

The contribution of this work is an intelligent multi-camera surveillance system that fulfills all the above criteria. Processing is distributed on low-cost embedded processor modules, which each learn a person detection classifier without supervision, thanks to an automatic labeller component that extracts and labels training examples on the fly. Detection accuracy is further improved with a novel spatial calibration technique, which allows each camera to learn the correspondence between its field of view and those of other cameras monitoring the same scene. Cameras then collaborate to confirm person detections and improve their performance.

The principle advantage of our multi-camera collaboration is that it offers an easy, unsupervised calibration and a computationally efficient means of runtime cooperation. The calibration procedure does not require detailed environmental models or knowledge of exact camera locations, as the Visual Surveillance and Monitoring (VSAM) [5] and Cooperative Distributed Vision (CDV) [9] projects do. The Intelligent Multi-Camera Surveillance and Monitoring (IMCASM) project [3] also offers unsupervised camera calibration, based on a homography mapping between camera views. However, applying this mapping during normal operation requires significant computation and would reduce the framerate on our hardware platform. The simplicity of our technique allows correspondences between camera views to be obtained with only a one-dimensional search through a row of the spatial association matrix.

Our method of automatic calibration (using a single moving target in overlapping camera FOVs) is closest to the work of Khan *et al.* [6]. During their calibration, a single person moves through the camera FOVs, allowing calculation of the FOV boundary lines for each camera. During normal detection operation, a camera can then tell which other cameras should be seeing a detected person, based on the person's current location (as determined by the position of their feet). Khan *et al.* use this correspondence to allow handoff of tracked targets between cameras, rather than to confirm detection. Although their approach is faster than the homography mapping used by IMCASM, it still requires more calculation than our approach.

Finally, it is important to note that, while person tracking could be added to the system as an additional layer, it is not required for camera collaboration. This is in contrast to nearly all the current multi-camera systems, including those discussed above. Our technique is ideal for embedded systems with slower frame rates, since it can function effectively without person tracking.

In the next section, we present our person detection technique within the context of a single camera, to facilitate understanding. In Section 3, we describe how this system can be extended to a multi-camera network using collaboration. Finally, Section 4 presents the results of our experiments.

2. Single-camera person detection

Each camera module is designed to be an independent unit capable of processing images, as well as communicating with the other cameras. To this end, we use a small, lightweight, embedded-microprocessor development board (the ARM-based Intrinsyc CerfBoard 250) as the core of each camera module. Each module also consists of an expansion card (CerfComm 250, needed for USB connectivity), a USB camera (D-Link DSB-C300 webcam), and a wireless network card (Linksys WCF12).

The single-camera person detection system is illustrated in Figure 1. The system is divided into three main components: the automatic labeller (Section 2.1), the classifier (Section 2.2), and the online learning algorithm (Section 2.3). Video frames received by the camera are sent to both the current classifier and the automatic labeller. The latter finds and labels person and non-person examples in the frame. These labels are compared with the current output of the classifier, and if the classifier was mistaken, the online learner updates the classifier.

2.1. Automatic labeller

The automatic labeller efficiently provides training examples on the fly, with which the online learner can train our classifier. The labeller does this with background subtraction [15]. Although computationally efficient, the initial results from background subtraction are prone to false positives. Since labeller provides training examples, it is important that it be reasonably accurate, while still being efficient enough for real-time processing.

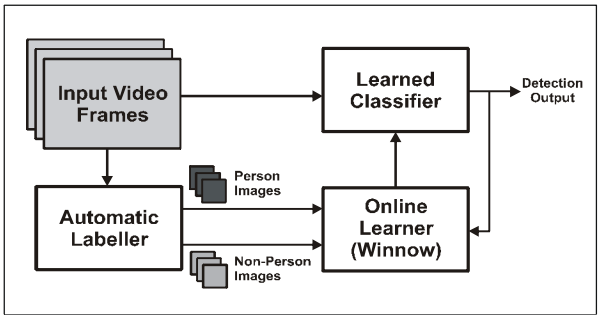

Figure 1. System diagram of the single-camera person detection application.

Background subtraction uses a statistical model of the background. The most accurate method is considered to be adaptive background subtraction [12, 16], but the increased calculations are too slow for our hardware platform, so a simplification is used: the background model is obtained by averaging 10 consecutive frames of video with little or no motion between them.

With the background model initialized, background subtraction is performed on each new frame in the RGB domain, unlike most systems, which use only greyscale intensity [3, 5, 10, 12]. Xu and Ellis [16] demonstrated significantly improved accuracy with chromaticity-based background subtraction. The per-pixel computation required for our background subtraction (three integer comparisons) is still less than even the simplest single-Gaussian-per-pixel greyscale adaptive background subtraction.

The automatic labeller features global failure avoidance, alerting it to cases where the whole frame may be unsuitable for use. This could be because of a lighting change or other large-scale environmental change in the scene, requiring re-calibration of the background model. If the percentage of foreground pixels in a frame is too high, the automatic labeller temporarily shuts itself off, ignoring the current frame and re-initializing the background model. In our setup, 44% was determined to be a good threshold percentage, although this depends on the camera sensor used and the scene being monitored.

Next, adjacent foreground pixels are grouped together into "blobs" of motion. Each blob undergoes a local failure avoidance check, designed to weed out erroneously-labelled person examples. A blob is discarded if it does not meet a minimum size requirement or if its aspect ratio does not approximate that of a person.

The segmented motion blobs that pass the checks are then labelled as "person" examples. Any motion blobs that failed the checks are left unlabelled, meaning they are not used for training. All other areas of the image are labelled "non-person". Rather than crop and store any images, the automatic labeller maintains a list of person example bounding boxes. During person detection, the current scan window can be quickly compared to the person example list to see if there exists a close match. If a match is found, the subimage's true label is "person". Otherwise, if no unlabelled areas exist in the scan window, the *true* label may be safely set to "non-person". This true label will be compared to the results of the classifier (the *predicted* label), as explained in the next section.

The performance of the labeller was suitable for real-time processing and made up only 5.6% of the per-frame processing time [1].

2.2. Feature-based person classifier

Person detection is performed with a classifier that learns the appearance of a person by using a set of image features, each of which is a wavelet template of the kind proposed by Oren *et al.* [11]. Each feature is a localized bandpass filter with a 2D structure. A window of interest is scanned across the image at different scales, and the classifier is applied to each subimage by evaluating all of its features. Feature values are then combined with the feature weights used by Winnow (see Section 2.3), to produce the classifier's prediction ("person" or "non-person").

Instead of using raw pixel values to compute each feature's value, we use a much faster alternative developed by Viola and Jones [14], called an integral image. The integral image is an intermediate representation, computed only once per frame, allowing rectangular regions of the image to be summed very quickly. It is represented as a 2D matrix of the same size as the original image. The integral image (*ii*) value at a point *x,y* is the sum of all image (*i*) pixels above and to the left of *x,y* (inclusive):

$$ii(x,y) = \sum_{x' \le x} \sum_{y' \le y} i(x',y') \qquad (2.1)$$

Any rectangular region can then be summed with only four array references and a sum of four terms. The sum of the pixels a rectangular region, whose vertices are labelled *A*, *B*, *C*, and *D* (clockwise from the top-left), is then simply:

$$sum = ii(D) - ii(C) - ii(B) + ii(A) \qquad (2.2)$$

During the classifier's initialization, an overcomplete feature set is created by varying the position and scale of the features across a basic 48x16 subwindow. As the subwindow is scaled during scanning, so are the classifier's features being applied. With the integral image technique, the computation time for the classifier is actually independent of the

scan window's size, allowing efficient use of the classifier and its features at a variety of scales.

This classifier functions as the system's internal representation of a person's appearance. Accordingly, using these features is preferable to raw pixel values, since we seek to capture general trends in the visual structure of a person, rather than specific pixel relationships. This allows our classifier to generalize and handle new person instances. In addition, starting with a large pool of features and using machine-learning techniques allows the classifier to capture non-intuitive details of a person's appearance that a hand-coded set of rules might not.

This classifier is a form of appearance-based person detection, which is well-suited for low-cost embedded hardware, as they do not require as much computation as model-based methods.

2.3. Online learning with Winnow

The Winnow learning algorithm [7] is an online-learning algorithm that trains a two-class classifier by learning a monotone disjunction from a high-dimensional feature vector. Winnow efficiently separates a number of relevant attributes from a much larger number of irrelevant ones. Consequently, this algorithm is ideally suited to our classifier, since we start with a large set of image features.

Winnow maintains a large pool of "experts" as a vector of Boolean variables with an associated vector of weights (one per variable). Each expert gives a prediction, and the prediction of the overall classifier is the dot product of the two vectors. If it is above the threshold θ, the classifier decides "person", otherwise it decides "non-person". The classifier is updated only when it is wrong, in which case experts are either increased or decreased by a weight update multiplier α.

Our classifier uses the balanced variation of the Winnow algorithm, which uses a weight vector for the features and another for their complements (see [7] for greater detail). Since Winnow adjusts weights by either multiplying or dividing them by a factor α, and that the weights are initialized to 1, each weight can be simply expressed as a power of α. Thus, only the exponents of α are stored, to save memory. The multiplier α is set to 2, allowing weight updates with bit-shifting.

Since Winnow uses Boolean features, we must map our image features (which return integer values) to a suitable vector of Boolean features. We therefore construct our Boolean feature vectors from the variables $x < t$, for each possible image feature x and each possible feature value t. Using this mapping, what could be nonlinear relationships between the original image pixels can be converted to linear relationships in this representation.

Since there is a total of 2,283 image features used in the classifier, each with a large range of possible values, we use the virtual weight algorithm [8], as Nair did [10], to efficiently represent the vectors. For a given image feature, each range of consecutive Boolean features with the same weight is represented virtually, by storing only a single weight and the delimitations of that range.

Additionally, since the vast majority of the Boolean features will be irrelevant for representing a person, we prune features whose weights are 2^{20} times smaller than the current biggest weight in the classifier, to further reduce memory and computation loads.

Winnow also provides the adaptability we seek, since learning is continuously performed online. Error accumulation is prevented, since if it starts to diverge from the automatic labeller's results, Winnow will compensate by training the classifier to correct any errors that it might initially make.

3. Multi-camera collaboration

Based on the observation that the single-camera system suffers from many false positives, and that these errors are generally the result of a scene feature that happens to be roughly person-shaped, it is reasonable to think that detection results can be improved by the collaboration of multiple cameras with overlapping FOVs. A scene artefact that appears person-shaped in one camera can look quite different from another point of view.

Collaboration is accomplished by learning the spatial relationship between camera FOVs, allowing each camera to report its local detections and receive confirmation. This technique uses an initial calibration procedure, explained in Section 3.1, through which each camera can learn the correspondences between points in its FOV and those in the other camera's FOV. Once this one-time calibration is complete, cameras can use these correspondences to confirm detections (see Section 3.2).

Currently, our system is run on two camera modules (referred to as Cam1 and Cam2), of the type presented in Section 2, arranged so that their FOVs overlap. The two cameras were placed in a corridor with heavy traffic. Cam1 was placed about a meter above the ground (located in region 3 of Cam2's view in Figure 2), and Cam2 was mounted on the ceiling (located in region 13 of Cam1's view in Figure 2). The detection application was run on 30,000 processed frames. The camera modules are each connected to a wireless network connected to the Internet, in order for the

Network Time Protocol (NTP) daemon on each module to synchronize the on-board clocks to a reference clock. NTP can provide accuracy to within a few milliseconds, which is sufficient for this design. During calibration, timestamps on each image are used to correlate detections, making it important that the clocks are aligned.

Once a person is detected in one camera, it sends out a notification to the other. Cameras use the spatial association matrix learned from calibration to determine if their local detections need confirmation and if so, what remote detections would correspond. Using passive notifications allows us to create a peer-based architecture (as opposed to a master-slave design) that is suitable for scaling up to large numbers of cameras.

3.1. Calibrating spatial associations

This multi-camera system uses a robust, unsupervised calibration procedure that is run only once after the cameras are first setup. The idea is to create a simple representation of the spatial relationship between the overlapping FOVs of each camera. This shifts much of the computational load of spatial association away from the actual person detection system, thus improving its real-time performance.

The calibration procedure is designed to be as simple as possible to run, making it suitable for large-scale installations. During this phase, one person simply walks around each camera's FOV, especially where they overlap. Each camera grabs and timestamps as many frames as it can. The actual calculations are performed after this step, in order to maximize the framerate of each camera.

Once the image capture stage is complete, a simple person detection algorithm, nearly identical to the automatic labeller explained in Section 2.1, is applied to the sets of frames. Since calibration occurs under controlled conditions (only one person in the scene and no sudden lighting changes), the results can be assumed to be fairly accurate. Motion-based detection is needed because the classifier has not yet been trained at this point.

Next, the center of mass of each detected blob is calculated. This is more accurate than using the center of the person bounding box, and since the calibration need only be performed once, the sacrifice of computational time is acceptable.

The system now has two timestamped lists of pinpointed person detection locations, one from each camera. Using a temporal alignment procedure similar to that used by Stein [13], it creates a list of all possible detection pairs (one from each camera) whose timestamps differ by less than a small time period, typically less than the slowest frame interval on the cameras.

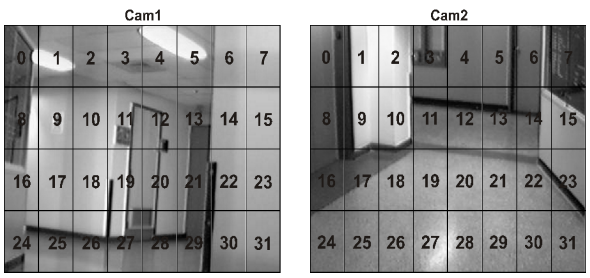

Figure 2. The 32 regions of each frame.

In order to increase generalization and reduce runtime complexity, each frame is divided into 32 regions (see Figure 2). The goal is to associate each region of a camera's frame with one of the 32 regions in the other camera's frame. The calibration program goes through each of the associated detection pairs and uses bilinear interpolation to increase association strength between the four closest region centres of the Cam1 detection and the four closest region centres of the Cam2 detection. The four closest region centres are used (weighted according to bilinear interpolation) instead of the single closest region, in order to maximize the applicability of the results. This allows us to build associations even in regions in which a person might not have been directly centred.

When this process is complete, a 32x32 matrix of association values between frame regions in each camera is produced. The values in this matrix can be interpreted as the strength or probability of the association between two regions. Values that are close to zero (below a certain minimum association threshold) suggest no association between those two regions. By finding the row with the highest association value for a given column (or vice-versa), we can determine the best match between a specific local region and a region in the remote camera. If the highest value found in such a search is below the minimum association threshold, this indicates that the given local region has no corresponding remote region, and is thus considered out of the other camera's FOV. This information will be useful for handling occlusions during runtime. Table 1 shows, for several regions in Cam1's frame, the best corresponding match in Cam2's frame, along with the association value (the region numbers refer to Figure 2).

Table 1. Spatial associations for several regions in Cam1. Region numbers refer to Figure 2.

Cam1 Region	Best Match Cam2	Assn. Strength

	Region	(out of 1.0)
8	15	0.42
9	12	0.67
10	4	0.19
13	10	0.23
14	10	0.17
15	No match	N/A

This procedure is used for two overlapping cameras. In a wide-area surveillance system with many cameras, this calibration procedure could start off using a custom protocol similar to the Dynamic Host Configuration Protocol (DHCP) to discover how many cameras are on the network and what their network addresses should be.

3.2. Collaborative peer-based camera network

The person detection system presented in Section 2 is now modified to make use of the association matrix learned during calibration. Figure 3 shows the complete multi-camera algorithm.

Each camera runs a listener thread, to wait for incoming detection notifications from the other camera. These notifications include a timestamp, and the location of a detected person. They are put in a remote detection queue and will confirm local person detections. NTP synchronizes the camera clocks, so that timestamps may be compared across cameras.

The main program thread is very similar manner to the stand-alone system in Section 2. The automatic labeller examines each new frame, then a person detection classifier is scanned across the frame to look for instances of people. Note that frame captures are now synchronized to the slowest camera. Online-learning is performed as before, by comparing the output of the classifier and the automatic labeller, then applying the Winnow algorithm when they differ. However, instead of simply using the predicted label (the classifier's output), the results are first verified with those of the other camera.

If the predicted label is "person", the spatial association matrix is used to check if this detection is visible to the other camera. The association values between the local detection region and all the remote regions are examined: if none of them are above a minimum association threshold, the detection is considered to be outside of the other camera's FOV. In this case, the predicted label is used the same way as in the single-camera system. Note that because the classifier is applied to a scan window, the exact person shape is not available; the camera must approximate the person's centre point with the centre of the detection bounding box.

Initialization:
1. Initialize background model (Section 2.1).
2. Initialize classifier's features and their weights.
3. Start listener thread to put incoming notifications into remote detection queue.

Online learning & person detection: (repeat for each frame)
4. Grab and timestamp new frame.

Automatic labeller: (Section 2.1)
5. Find foreground pixels using background subtraction.
6. If there is too much foreground, re-initialize background, *notify other camera of frame completion,* and go to Step 4.
7. Group foreground pixels into "blobs" of motion.
8. Label each region of image as either "person" or "non-person", or leave it unlabelled.

Person detection: (Section 2.2)
10. Convert image to greyscale and compute integral image.
12. Scan entire frame by shifting scan window across image, varying window's position and scale.

For each subimage defined by scan window:
13. Get subimage's "true" label from results of Step 8, if region was labelled (Section 2.1).
14. Get subimage's "predicted" label from current state of classifier (Section 2.2).
15. If predicted label is "person", use association matrix to check if other camera can see person. If not, output detection.
16. If person is in other camera's FOV, notify camera and check remote detection queue for recent detections spatially associated with local detection. If one is found, output detection. If nothing found, put detection in local queue, to compare with future remote detections.

Online learning: (Section 2.3)
17. If "true" label differs from "predicted" label, send both labels and computed classifier values to Winnow learning algorithm.

18. Send "end of frame" message to other camera and wait for same.
19. Clear local detection queue, since anything left is unconfirmed.
20. Clear any detections in remote queue with timestamp less than current frame's timestamp.

Figure 3. The multi-camera algorithm. Statements in *italics* are specific to the multi-camera version.

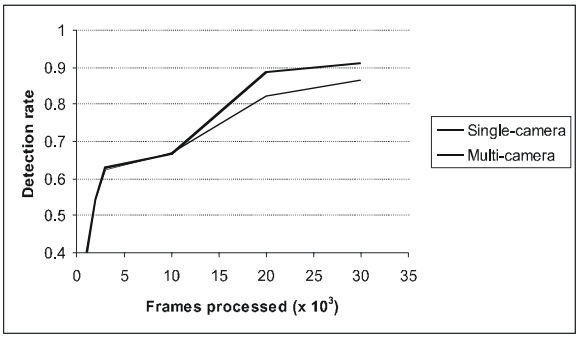

Figure 4: Cam1's detection rate, for single- and multi-camera implementations.

If the detected person is within other camera's FOV, the camera then seeks confirmation. A notification of the person detection is sent out. This is the type of message handled by the listener threads. The queue of remote detections is then examined for recent detections already sent by the other camera. If any temporal matches are found (within the approximate processing time of a single frame), the spatial association matrix is used to check for a spatial correlation. If a remote detection matches both temporally and spatially, the local detection is confirmed and the predicted label is set to "person". No explicit confirmation of this match is sent to the other camera, since it will come to the same conclusion on its own. If no match is found, the detection is placed into a local queue, for comparison with future remote detections as they are received by the listener thread. This provides robustness against network latencies and processing delays on each camera.

The simple symmetry of this communication, requiring only straightforward notifications, results in a flat, peer-to-peer network architecture that can be easily scaled up to a large number of cameras. Scalability is also increased by the low-bandwidth requirements of this system.

4. Results

The motivation behind the multi-camera collaboration was to improve the accuracy of person detections by reducing the number of false positives returned by the classifier, as compared to the single-camera implementation. In this respect, multi-camera collaboration was quite successful. At the end of 30,000 frames in which 1,349 subimages per frame were processed, both cameras achieved a noticeable improvement. For example, Cam1's detection rate, or the percentage of people correctly spotted, increased slightly from 86% to 91%, as shown in Figure 4. This modest improvement was expected, since the goal of

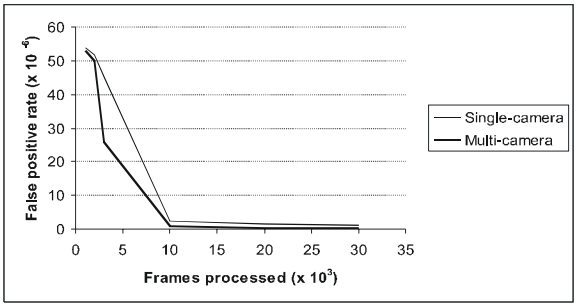

Figure 5. Cam1's false positive rate, for single- and multi-camera implementations.

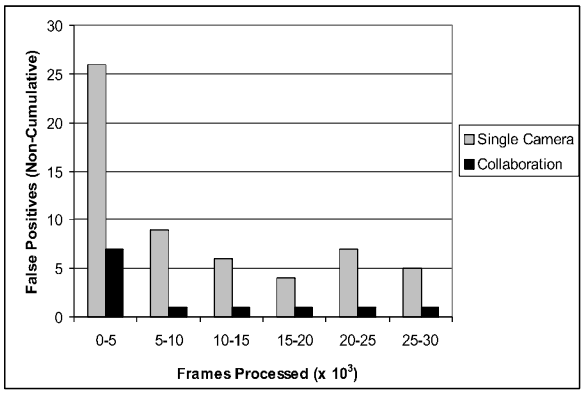

Figure 6. Distribution of false positives (grouped into bins of 5000 processed frames), for single- and multi-camera setups.

Figure 7. Examples of positive detections.

our collaboration was to reduce the false positive rate, shown in Figure 5. This rate is the number of non-person images mistakenly identified as people, divided by the total number of true non-person images. Figure 6 shows that the false positive rate dropped much faster in the multi-camera implementation and continued to be smaller. After 30,000 frames, the false positive rate was 0.21×10^{-6}, compared with 1.36×10^{-6} for the single-camera setup: collaboration thus reduced the number false positives by over 6.5 times.

Examples of the system's person detection results are shown in Figure 7. The right-most example shows an incorrectly-classified non-person (i.e. a false positive). The bounding boxes of detection results appear slightly off-center, due to the scan window moving at fixed, discrete steps during classification. The classifier returns the current location of the scan window when a positive result was detected, which approximates the location of the detected person.

5. Conclusions

We have presented our efforts towards an intelligent multi-camera surveillance system that uses a distributed network of camera modules, each capable of onboard processing and automatic person detection. A method for unsupervised, online learning of a person detection classifier was described, and this learned classifier was then integrated into a multi-camera system to improve detection performance. A novel, unsupervised method of camera calibration was presented, allowing cameras to learn the spatial relationships between each other in a one-time setup phase. Execution of this calibration step is simple enough to be used with a large network of cameras. The spatial associations learned from this calibration are then applied during person detection, allowing cameras to match and verify corresponding detections. The computational efficiency of this design is well-suited for real-time performance on low-cost hardware. The results of this multi-camera system were positive, showing that the system benefited from collaboration. The novel contributions of this work are the spatial calibration technique and the extension of Nair's work [10] over a multi-camera network to achieve performance gains.

6. References

[1] T. Ahmedali. *Collaborative Multi-Camera Surveillance Using Automated Person Detection*. M.Eng. Thesis, McGill University, 2005.

[2] J. Black and T. Ellis, "Intelligent image surveillance and monitoring", *Measurement and Control*, 35(8): 204-208, 2002.

[3] J. Black and T. Ellis, "Multi camera image tracking", *2nd Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2001)*, Kauai, Hawaii, USA, 2001.

[4] B. A. Boghossian and S. A. Velastin, "Image processing system for pedestrian monitoring using neural classification of normal motion patterns", *Measurement and Control*, 32(9): 261-264, 1999.

[5] R. T. Collins, A. J. Lipton, and T. Kanade, "A system for video surveillance and monitoring", *ANS 8th Int. Topical Meeting on Robotics and Remote Systems*, Pittsburgh, PA, USA, 1999.

[6] S. Khan, O. Javed, Z. Rasheed, and M. Shah, "Human tracking in multiple cameras", *IEEE ICCV 2001*, Vancouver, BC, Canada, 2001: 331-336.

[7] N. Littlestone, "Comparing several linear-threshold learning algorithms on tasks involving superfluous attributes", *12th Int. Conference on Machine Learning*, Tahoe City, CA, USA, 1995: 353-361.

[8] W. Maass and M. K. Warmuth, "Efficient learning with virtual threshold gates", *Information and Computation*, 141(1): 66-83, 1998.

[9] T. Matsuyama, "Cooperative distributed vision: dynamic integration of visual perception, action, and communication", *23rd Annual German Conference on Artificial Intelligence*, Bonn, Germany, 1999: 75-88.

[10] V. Nair and J. J. Clark, "An unsupervised, online learning framework for moving object detection", *IEEE Computer Society CVPR 2004*, Washington, DC, USA, 2004: 317-324.

[11] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates", *IEEE Computer Society CVPR 1997*, San Juan, Puerto Rico, 1997: 193-199.

[12] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Computer Society CVPR 1999*, Fort Collins, CO, USA, 1999: 246-252.

[13] G. P. Stein, "Tracking from multiple view points: self-calibration of space and time", *IEEE Computer Society CVPR 1999*, Fort Collins, CO, USA, 1999: 521-527.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", *IEEE Computer Society CVPR 2001*, Kauai, HI, USA, 2001: 511-519.

[15] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7): 780-785, 1997.

[16] M. Xu and T. Ellis, "Illumination-invariant motion detection using colour mixture models", *BMVC 2001*, Manchester, UK, 2001: 163-172.