Video Surveillance using a Multi-Camera Tracking and Fusion System

Zhong Zhang, Andrew Scanlon, Weihong Yin, Li Yu, Péter L. Venetianer

Video Surveillance using a Multi-Camera Tracking and Fusion System

Zhong Zhang, Andrew Scanlon, Weihong Yin, Li Yu, Péter L. Venetianer
ObjectVideo Inc.
{zzhang, ascanlon, wyin, liyu, pvenetianer}@ObjectVideo.com

Abstract. Usage of intelligent video surveillance (IVS) systems is spreading rapidly. These systems are being utilized in a wide range of applications. In most cases, even in multi-camera installations, the video is processed independently in each feed. This paper describes a system that fuses tracking information from multiple cameras, thus vastly expanding its capabilities. The fusion relies on all cameras being calibrated to a site map, while the individual sensors remain largely unchanged. We present a new method to quickly and efficiently calibrate all the cameras to the site map, making the system viable for large scale commercial deployments. The method uses line feature correspondences, which enable easy feature selection and provide a built-in precision metric to improve calibration accuracy.

1 Introduction

The usage of IVS systems is spreading rapidly. Based on user defined rules or policies, IVS systems can automatically detect potential threats or collect business intelligence information by detecting, tracking and analyzing the targets in the scene. In large, multi-camera installations, a central management console provides unified access to all systems, allowing centralized configuration and quick access to all rules, alerts and results. The user interface may display all results together on a map, or a counting application may aggregate the counts from different feeds. But processing of the camera feeds, the rules and the alerts are still independent. While this setup is sufficient for some scenarios, its effectiveness is limited by detecting only local events. More complex events spanning multiple cameras cannot be detected, thus potentially missing important events. The system described in this paper fuses information from multiple cameras, thus providing much better awareness of what is going on in the whole area. Different from the majority of the previous works, which mainly address computer vision or data fusion topics such as object appearance matching [1,2,3], camera topological relationship estimation [4,5] and statistical data association [6,7], the objective of the present work is to develop a commercial viable system that has real-time performance, low bandwidth requirement and in particular, easy installation so that an ordinary security personnel can configure and operate it easily.

The paper is organized as follows: Section 2 describes the architecture of a typical single camera surveillance system. Section 3 explains how this architecture is expanded into a multi-camera system. Section 4 provides some real-life applications of the system. Section 5 lists potential extensions for future work, before concluding remarks in Section 6.

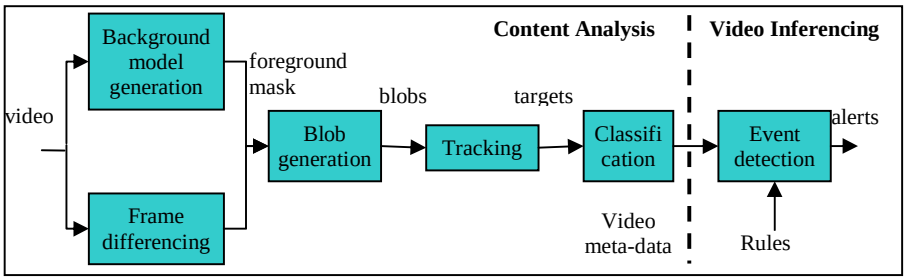

Figure 1: Flow-chart of typical IVS system

2 Single Camera Surveillance System Architecture

A typical IVS system is illustrated in Figure 1. A dynamic background model is continuously being built and updated from the incoming video frames. In each video frame, pixels that are statistically different from the background are marked as foreground. These foreground pixels are spatially grouped into blobs, which are tracked over time to form spatio-temporal targets, e.g. using a Kalman filter. Next, these targets are classified based on various features. Finally the events of interest (rules) specified by the user are detected on the targets. For example, the user may want to detect when people enter an area by defining a virtual tripwire.

The first part of the above processing pipeline up to and including the classification is very generic, largely independent of the details of the application and the user defined events of interest. These steps, marked as content analysis in Figure 1, all deal with the actual video frames and generate a high level meta-data description of what is happening in the video. This meta-data contains all target information (location, velocity, classification, color, shape, etc.), and potentially the description of the scene, including static (water, sky, etc.) and dynamic (lighting change) descriptors. The end of the processing pipeline, the event detection uses this meta-data description as its input instead of the video, and compares that with the user defined rules. This mode of operation means that only the meta-data has to be stored, instead of high quality video suitable for automated processing, and events can be detected very quickly, simply by analyzing the meta-data, instead of the much slower video analysis. And this meta-data enables the multi-camera surveillance system described in more detail in the next section.

3 Multi-Camera Surveillance System Architecture

The IVS system, as described in the above section, can provide an adequate solution for many applications. But by analyzing only a single video feed at a time, it offers a somewhat myopic view into the world, with all its associated limitations. For example the goal for the IVS system may be to detect suspicious activities around a facility with several cameras daisy-chained around its fence line. A vehicle parking near that fence line can easily be detected by the camera covering the area where the vehicle parks. But a vehicle circling around the facility multiple times cannot be detected by the single camera system. A multi-camera surveillance system tracking targets from

one camera to the next can overcome all these limitations. This section describes the key challenges of such a system and a solution that has been demonstrated to work well in several real life deployments.

3.1 Data Sharing

One of the key questions when designing the cross-camera surveillance system is to decide at which stage in the pipeline of Figure 1 should the single camera units share their information. Performing fusion before the foreground detection or blob generation steps requires building a mosaic, which is very expensive on cpu, memory and bandwidth usages. In addition, it usually requires the cameras having overlapped field of views and similar illumination and image resolution, which may not always be satisfied in real applications.

Fusing at the video meta-data level requires merging all the meta-data from the cameras onto a full representation of the environment. This approach distributes the most time consuming processing between the different sensors, eliminates the need for a mosaic, and minimizes communication, since only the meta-data needs to be transmitted, no video or imagery. Given these advantages, our system communicates only the video meta-data for fusion. For example, the video meta-data from a single camera unit for each video frame may include the following information: the camera time stamp, list of targets with their ids and image properties such as bounding box, centroid, footprint and classification label.

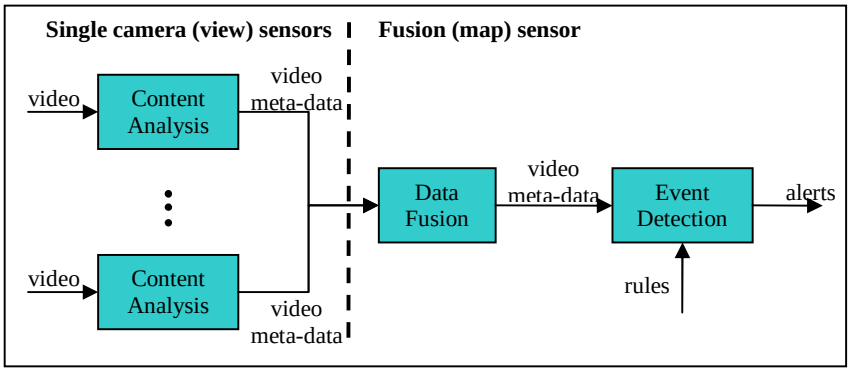

Figure 2: Cross-camera fusion system diagram

3.2 System Design

The cross-camera fusion system is illustrated in Figure 2. The video from each camera is initially processed the same way as in a single camera system: the content analysis module translates video into video meta-data, which is then sent from all sensors to the centralized data fusion module. Fusion combines the video meta-data from all sensors into a common coordinate system, but still maintaining the video meta-data format, so that the fused meta-data can be fed to the event detection module. This event detection module is identical to the one used in the single sensor

system of Figure 1. The rules and meta-data are all represented as relative coordinates. For a single camera sensor the coordinates are relative to a single frame, while for the fusion sensor they are relative to the global map is used. This means that the meta-data is the same whether it is generated by a view or a map sensor.

This design has many benefits. The time consuming video processing is distributed among the single camera sensors, communication bandwidth requirements are low due to transmitting only the video meta-data. The sensor architecture is simple: the system running on the individual sensors is almost identical to the single camera system. Content analysis turns the video into video meta-data. It is still possible to have single camera event detection running on the individual sensors, if required. The only difference is that the video meta-data is streamed out to the fusion sensor to enable multi-camera event detection. The fusion sensor is more different, but still has a lot in common to single camera sensors. The main difference is the front end: it ingests multiple video meta-data streams instead of video, and uses the data fusion module to convert it into fused meta-data. This similarity between the different modes means that our system has only a single main executable, which can be configured at installation to act as a stand-alone single camera sensor, as a single camera sensor used for fusion, or as a fusion sensor. More and more IVS systems are moving towards performing computations on the edge, embedded in a camera. This architecture works well for that approach as well. The embedded system processes the video and generates the meta-data, which is then sent to a centralized fusion sensor.

This approach also seamlessly supports the forensic applications described earlier. The video meta-data can be stored in the individual sensors, performing fusion and event detection at the time of forensic processing, or the fused meta-data can be stored, in which case forensics is the same as with the single camera forensics. Moreover it is also possible to later convert a standard installation into a cross-camera system for forensic analysis. If the single camera video meta-data has been stored, even calibration can be performed later, and forensics executed on the previously stored data.

3.3 Cross-camera calibration

The major challenge of a cross-camera tracking system is how to associate the targets detected and tracked in different individual cameras. The data fusion process illustrated in Figure 2 requires the single camera sensors to be calibrated in such manner that the targets in different cameras have a common coordinate system.

Traditional camera calibration approaches [8] rely on using a 3D reference object with a known Euclidean structure. However, setting up the 3D reference object with great accuracy is difficult, requires special equipment, and doesn't scale well. To overcome some of these problems, [9] and [10] introduced a simple and practical camera calibration technique using a model plane with a known 2D reference pattern. In this technique, the user places the model plane or the camera at two or more locations and captures images of the reference points. Camera parameters are recovered from the model plane to image plane homographies computed from correspondences between the reference points and their projections. Although this algorithm is simpler, it yields good results mainly for indoor and/or close range applications, where the object is big enough that its features can be easily and

accurately detected and measured. To make this approach viable for large area outdoor applications, the reference object would have to be very large to provide the necessary accuracy.

In the proposed system, we introduce a cross-camera calibration approach called map-view mapping, which maps each ground point in the image ("view") to its corresponding point on a global site map ("map"). The global site map here may be a fine resolution satellite image for an outdoor application or a blueprint drawing for an indoor application. In this approach, we assume that in each view the targets are on the same plane, called image ground plane; and the global map also represents a single plane in the world, called map ground plane. Thus for each camera, the mapping between point x in the view and the corresponding point X in the map is fully defined by a homography H [11,12]:

$$X = Hx \tag{1}$$

where H is a 3x3 homogeneous matrix. The map and view points are represented by homogeneous 3-vectors as $X = (X, Y, 1)'$ and $x = (x, y, 1)'$, respectively. The scale of the matrix does not affect the equation, so only the eight degrees of freedom corresponding to the ratio of the matrix elements are significant. The camera model is completely specified once the matrix is determined. H can be computed from a set of map-view correspondence points. From equation (1), each pair of correspondence points provides two equations. Given the 8 unknowns in H, $n \geq 4$ point pairs are needed. Writing H in vector form as $h = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})'$, (1) for n points becomes $Ah = 0$, where A is a $2n \times 9$ matrix:

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 & -X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 & -Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 & -X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nX_n & -y_nX_n & -X_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nY_n & -y_nY_n & -Y_n \end{bmatrix} \tag{2}$$

The goal is to find the unit vector h that minimizes $|Ah|$, which is given by the eigenvector of the smallest eigenvalue of $A'A$. This eigenvector can be obtained directly from the SVD of A.

The key element of the process becomes finding correspondence points between the map and each camera view. These correspondence points are also called control points in image registration. They provide unambiguous matching pairs between the map and the camera view. However, there are two potential problems with using only matching points for calibration. The first problem is that it may be difficult to find the precise corresponding point locations in some environments due to limited resolution, visibility, or the angle of view. As an example, looking at an overhead view of a road, the corner points of the broken lane dividing lines theoretically provide good calibration targets. However, it may be difficult to reliably determine which lane dividing line segment of the map view corresponds to which line segment in the camera view.

The second problem is that the precision of the pairs of matching points is usually unknown. The precision of a point measures the sensitivity of the accuracy of the map matching location with respect to the accuracy of the view location. For example, at one location in the camera image plane, one pixel movement away from that location may cause 100 pixels of movement away from its original corresponding location on the map. This means the precision of this pair of matching points is low. When we calibrate the camera view onto the map, we minimize the distance between these pairs of matching points. Assigning higher weight to points with higher precision improves calibration performance.

To overcome the above two problems, we introduce matching line features in conjunction with the matching points. A line feature is typically specified by two points, as a line segment, but for a matching line feature only the corresponding lines have to match, not the segments. For example when viewing a road, it might be difficult to find point correspondences, but the dividing line and the edges of the road define good line features. Hence the line features help to overcome the first limitation above. Figure 3 illustrates matching line segments.

Figure 3: Selecting matching features. In this corresponding pair of map (left) and view (right), it is much easier to find matching lines than points. Matching features are represented by the same color.

Matching line features also help overcome the second problem by providing a good precision metric, which helps to improve calibration accuracy by allowing the system to put increased weight on more precise control points. Line features can be directly specified by the user, or computed from pairs of user defined calibration control points. Additional control points are then computed from the intersection of line features. The precision of such a computed control point is determined as follows: First, use the point of intersection on the map as the reference point. Next, add small random Gaussian noise with zero mean and small standard deviation (e.g. 0.5) to the end points of all the related line segments on the map and recompute the point of intersection. Calculate the distance between the new and the reference point of intersection. Repeat the above process many times and compute the mean distance, which reflects the precision of the corresponding point of intersection. The point of intersection will be used as a control point only if its corresponding mean distance is less than a threshold, determined by the desired accuracy of the calibration.

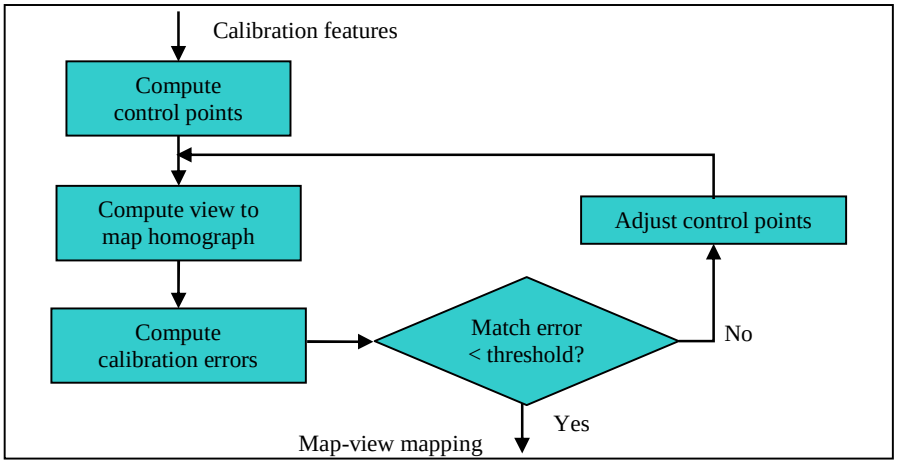

Figure 4: Camera calibration block diagram

Figure 4 illustrates the view-to-map camera calibration process. First, control points are computed as described above using the matching features selected by the operator on a GUI provided by the system. Next, the image plane to map plane homography is computed using the Direct Linear Transformation algorithm [12]. This least squares based method is very sensitive to the location error of the control points; especially if the number of the control points is small or the points are clustered in a small portion of the image. In the proposed approach, matching line features are used to iteratively improve the calibration. In each iteration, control points are added or adjusted, till the change in the error of feature matching falls below a threshold. Since a line segment is a more representative feature than a single point and its location is more reliable than that of a single point, this iterative refinement process very effectively reduces calibration errors and in our system it always rapidly converges to the optimal result.

In each iteration, the corresponding view and map features are used to estimate the homography. That homography is used to transform the view features onto the map. Then the calibration error is computed as the average distance on the map between the transformed view features and the corresponding map features. For a point feature, the distance is simply point to point distance. For a line feature, the distance is the enclosed area between the two line segments, as illustrated by the shaded area in Figure 5. In order to reduce this calibration error, we add new control points based on line features. In Figure 5, L_1 and l_1 represent a pair of matching line segments on the map and view, respectively. Note that their ending points are not required to be matching points, thus they may not be in the control point list initially. Once H is estimated, the view line segment l_1 is transformed into the map line segment L_1' with points P_1' and P_2'. The goal is to find an estimate of H that minimizes the distance between L_1' and the original matching line segment L_1 on the map. This is obtained by minimizing the shaded area between L_1 and L_1'. To achieve this, P_1' and P_2' are projected onto line L_1, yielding P_1 and P_2. Thus, point pairs (p_1,

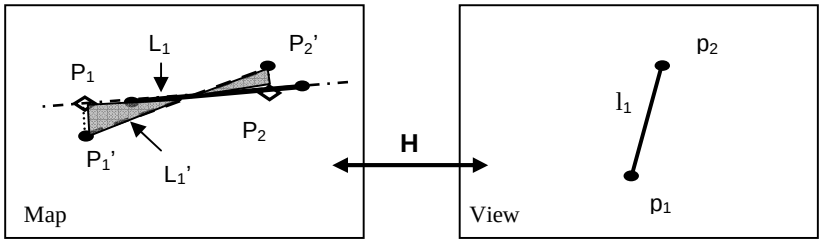

Figure 5: Adjusting control points

P_1) and (p_2, P_2) become matching pairs of points and are added to the list of control points for the next iteration. In subsequent iterations, these additional control points are further adjusted by projecting them on the line L_1 based on the newly estimated homography *H*, as long as these adjustments further improve the calibration accuracy. The above calibration processes are all performed automatically except the manual matching feature selection operation using a GUI.

Although the map-view mapping or the *H* matrix obtained above is not a full camera calibration, it provides the most valuable information for the cross camera target tracking. First, since all of the camera views are calibrated to the same map, the corresponding targets from multiple cameras can be naturally associated based on their map locations. Second, by using actual map scales, the physical size and velocity of the target can be estimated, providing useful new target information. Third, the map-view mapping can be used to estimate the effective field of view (EFOV) of each camera on the map. The EFOV defines the effective monitoring area of each camera, helping with planning camera placement, and performing cross camera target handoff. The EFOV of a camera includes the points where the view size of a human is above a threshold and the mapping error is low.

3.4 Data Fusion

The data fusion module of the fusion sensor continuously receives video meta-data of all targets from all individual sensors. As the first step, the fusion sensor projects each data onto the map using the calibration information of the sensor. After this step the fusion sensor introduces some delay (dependent on the typical network delay) to make sure that all target information for a given time instance is received from all sensors before processing. To achieve this it is crucial that all sensors are time synchronized with each other, ensuring that the fusion sensor can properly combine them. In the fusion sensor, we maintain a data buffer on the input meta-data from the view sensors and batch process them after every T seconds. The T is determined by the maximum network latency. Once the actual view sensor to fusion sensor delay is less than T, the input data will be properly synchronized. In our installations, the T is usually less than 1 second.

Figure 6 illustrates one iteration of the target data fusion process. Based on the incoming synchronized video meta-data, the update view target module builds its own representation of each view target, adding in additional data such as map location and physical size, computed from the map-view mapping. Next, the view target fusion module checks if any stable new view target matches an existing map target. If it

does, the map target is updated with the view target. Otherwise, the system may produce a new map target from the new stable view target. Here, a “stable” view target means the target has a consistent appearance and is tracked with high confidence. This requirement is used to temporarily ignore non-salient targets, partially occluded targets, and targets on the image boundaries, where both location and appearance may be inaccurate. The matching measure between two targets is the combination of the location, size and appearance matching probabilities. The location matching probability is estimated using the target map location from the map-view mapping. The size matching probability is computed from the relative physical size of each target. The appearance matching probability is obtained by comparing the appearance models of the targets under investigation. The appearance model used in our system is a distributed intensity histogram, which includes multiple histograms for different spatial partitions of the target. The appearance matching score is the average correlation between the corresponding spatially partitioned histograms.

One map target can represent multiple view targets from different views, e.g. when a target is in the overlapping area of two cameras. For each time stamp, the map target has a primary view target that provides the most reliable representation of the physical object at that time. The map target update process determines this primary view target and updates the general target properties such as map location, velocity, classification type, etc. It may also reclassify targets based on additional map target properties, such as physical size and speed.

Since target occlusion can cause significant map location estimation errors, the map target fusion module tests whether a map target corresponds to another existing map target when the stability status of the map target changes. Each target has an associated stability status based on how consistent its shape and size is in a temporal window. If a merge is needed, the map target with shorter history is merged into the other target.

4 Examples

The system described in the paper was successfully installed in a number of applications. This section describes two very different installations. Both installations are using Windows XP on Intel processors. A 2.8GHz dual-CPU machine can comfortably run up to four sensors each at around 10 fps.

4.1 Critical Infrastructure Protection

The most typical application of the camera fusion system is protecting a larger site with several cameras daisy chained around the perimeter, as illustrated in Figure 7. The system was installed around a military airfield, with 48 fixed and 16 PTZ cameras covering a 5 mile perimeter. The 48 fixed cameras were daisy chained, with overlapping fields-of-views, providing full fence coverage. As the first step of the installation all these cameras were calibrated to the site map by manually selecting corresponding point and line features. The PTZ cameras were installed to provide better resolution imagery in case of intrusion. The details of PTZ calibration and operation are beyond the scope of this paper.

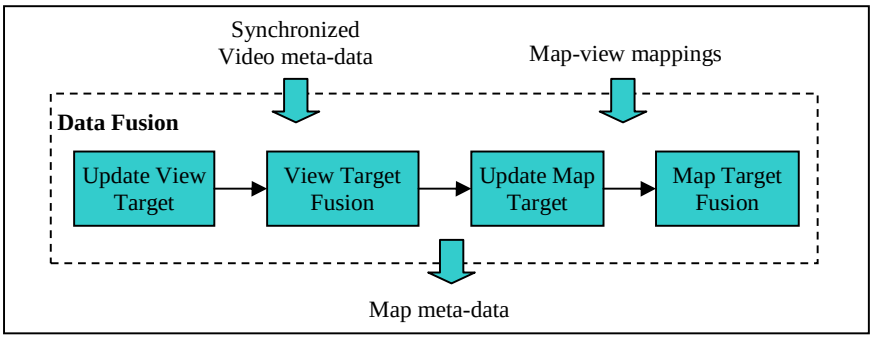

Figure 6: Block diagram of data fusion process

The system allows the user to setup rules either on individual camera views or on the map or both view and map. The most important rule for the user was a multi-segment tripwire following the fence-line around the whole perimeter, configured to detect when targets cross it from outside to in. The user interface for defining rules has several features to help the precise definition of rules. The rule can be drawn initially at a more zoomed out setting showing the full area. Then portions can be viewed and fine tuned in higher zoom levels. Besides seeing the rule on the map, the interface also allows projecting, visualizing and editing it in a camera view. This method typically provides the highest resolution, and it also allows fixing some inaccuracies resulting from potential calibration errors. In the far view of a camera, being just a pixel off may translate into meters on the map, and that discrepancy in turn can mean the difference between a rule aligned with the fence, or being on the public road around the fence, generating several false alarms. Such inaccuracies can easily be visualized and corrected by viewing the rules on the map. In addition to the multi-segment tripwire rule, some areas are protected with rules detecting enters and loiters in an area of interest.

All alerts contain the location of the detected event. By default, the location includes location in the camera with the best view of the target; and location on the map. If the map is just an image, then location is represented as an image coordinate. If the map has associated geographical information, such as a world file, than the location is expressed as lat/long coordinates as well.

The system is very flexible, can easily accommodate configuration changes. Additional cameras can be added quickly by calibrating them to the map and the PTZ cameras. The rest of the sensors and rules are completely unaffected by this addition.

4.2 Hazardous Lab Safety Verification

The same system was used in a very different application scenario by a major research university. The goal was to detect violations of the so called two-person rule in some infectious disease laboratories. The rule means that a person should never be alone in the lab, i.e. if there is anybody in the lab, there should be at least two people. The only exception is the few seconds around people entering and exiting, so if a person enters the empty lab, there should be no alert as long as another person enters within a few seconds.

The most straightforward solution would be to count the number of people entering and exiting the lab, and use the difference of the two as the person count. The big drawback of this approach is that it has no mechanism for correcting errors. For example if two people enter and are counted correctly, but they leave so close to each other that they are miscounted as a single person, the system will false alert, even though there is nobody left inside. Similarly the system could miss an event, which is an even greater problem. For this reason a robust system needs to monitor the whole lab, so that it can continuously keep track of the people inside, maintaining an up-to-date count. To obtain good coverage, minimizing occlusions which are the main reason for counting errors, the cameras were mounted on the ceiling. Some labs were small enough to be covered by a single wide angle camera, but others required more than a single camera. For these larger labs fusion was very important, otherwise people in the area where two cameras overlap would have been counted by both cameras.

System configuration was conceptually similar to the critical infrastructure example, but there were some differences. The map was replaced with a blueprint or sketch of the lab. The requirement is that the blueprint has to be of the correct scale and contain sufficient identifiable features on the floor (ground plane) for the calibration. The camera was calibrated to the blueprint using the manually selected feature correspondences. The cameras were running content analysis, reporting all human targets to the fusion sensor. The fusion sensor projected these targets onto the blueprint, fusing targets in the overlap area into a single human. The fusion sensor counted the number of people, and alerted if it saw only a single person for longer than a user defined minimum time, typically around 30sec.

5 Future work

We are looking at improving the overall system in several ways. The current fusion, as described in Section 3.4, uses location as the strongest cue for fusing targets, in

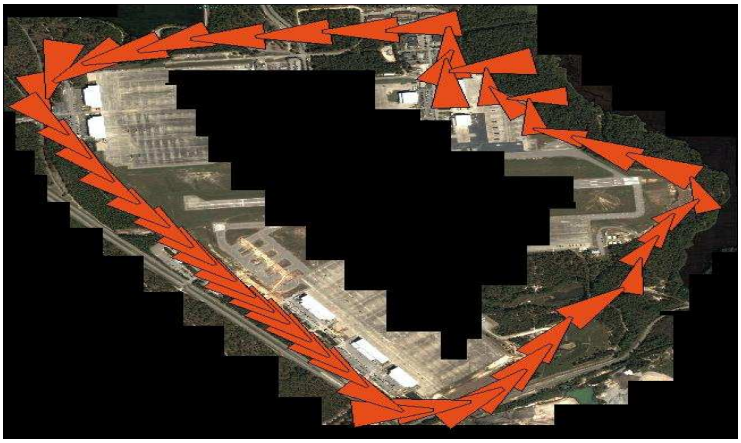

Figure 7: Wide area surveillance by daisy-chaining cameras (red cones) around the perimeter of the protected facility

conjunction with some other features. In more crowded environments, however, a more sophisticated method is required. We are looking into using additional features, such color, shape, etc. to properly resolve this challenge.

We are also looking into how to make the system more scalable. The current implementation was tested with over 50 individual sensors communicating with the fusion sensor, but an installation covering really large areas with hundreds of sensors would be problematic, with too much data flooding the fusion sensor and overwhelming its processing. A more scalable solution can use multiple layers of fusion: a fusion sensor handling a smaller cluster of sensors, and multiple such fusion sensors communicating to higher level fusion sensors. Fusion itself requires target information only from the overlapping areas of the cameras, but some of the event logic requires full target history. For example to detect a target circling around a facility, the whole target track is required, so the information has to be combined from all sensors to be able to generate the alert.

6 Conclusions

We presented a real-time cross-camera fusion system. It provides powerful new capabilities over single camera system, but with very little extra complexity both in terms of user interface and implementation. The system was successfully deployed in a variety of commercial applications.

References

1. Guo Y, Sawhney H S, Kumar R, et al. Robust Object Matching for Persistent Tracking with Heterogeneous Features. In Joint IEEE Int. Workshop VS-PETS. 2005, 81-88.
2. Shan Y, Sawhney H S, Kumar R. Unsupervised Learning of Discriminative Edge Measures for Vehicle Matching between Non-Overlapping Cameras. In CVPR. 2005, 894-901.
3. Javed O, Shafique K, Shah M. Appearance Modeling for Tracking in Multiple Non-overlapping Cameras. In CVPR. 2005, 26-33.
4. Ellis T J, Makris D, Black J. Learning a multi-camera topology. In Joint IEEE Int. Workshop VS-PETS. 2003, 165-171.
5. Tieu K, Dalley G, Grimson W E L. Inference of Non-Overlapping Camera Network Topology by Measuring Statistical Dependence. In ICCV. 2005, 1842-1849.
6. Huang T, Russell S. Object identification: A Bayesian analysis with application to traffic surveillance. Artific. Intell., 1998 (103): 1-17.
7. Kettnaker V, Zabih R. Bayesian Multi-camera Surveillance. In CVPR. 1999, 2253-2259.
8. Tsai R Y, A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses, IEEE Journal of Robotics and Automation, Aug 1987, 3(4):323–344.
9. Sturm P F, SMaybank S.J. On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications, Proc. Computer Vision and Pattern Recognition, 1999, volume 1, 432–437.
10. Zhang Z, Flexible Camera Calibration by Viewing a Plane from Unknown Orientations, Proc. 7th International Conference on Computer Vision, 1999, volume 1, 666–673.
11. Semple J, Kneebone G, Algebraic Projective Geometry. Oxford University Press, 1979.
12. Hartley R, Zisserman A, Multiple View Geometry in Computer Vision. Cambridge University Press, 2003.