Privacy-Protecting Video Surveillance

Jehan Wickramasuriya, Mohanned Alhazzazi, Mahesh Datt,
Sharad Mehrotra and Nalini Venkatasubramanian

Department of Information & Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA
{jwickram,malhazza,mahesh,sharad,nalini}@ics.uci.edu

ABSTRACT

Forms of surveillance are very quickly becoming an integral part of crime control policy, crisis management, social control theory and community consciousness. In turn, it has been used as a simple and effective solution to many of these problems. However, privacy-related concerns have been expressed over the development and deployment of this technology. Used properly, video cameras help expose wrongdoing but typically come at the cost of privacy to those not involved in any maleficent activity. This work describes the design and implementation of a real-time, privacy-protecting video surveillance infrastructure that fuses additional sensor information (e.g. Radio-frequency Identification) with video streams and an access control framework in order to make decisions about how and when to display the individuals under surveillance. This video surveillance system is a particular instance of a more general paradigm of privacy-protecting data collection. In this paper we describe in detail the video processing techniques used in order to achieve real-time tracking of users in pervasive spaces while utilizing the additional sensor data provided by various instrumented sensors. In particular, we discuss background modeling techniques, object tracking and implementation techniques that pertain to the overall development of this system.

Keywords: Surveillance, Real-time, Object Tracking, Privacy, Access Control

1. INTRODUCTION

With the heightened consciousness among the public, private and government organizations for security, surveillance technologies (especially video surveillance) have recently received a lot of attention. Video surveillance systems are being considered/deployed in a variety of public spaces such as metro stations, airports, shipping docks, etc. As cameras go up in more places, so do the concerns about invasion of privacy.[1] Privacy advocates worry whether the potential abuses of video surveillance outweigh its benefits. A fundamental challenge is to design surveillance systems that serve the security needs while at the same time protect the privacy of the individuals.

In this paper, we describe the design of a privacy preserving video surveillance system that monitors subjects in an instrumented space only when they are involved in an access violation (e.g., unauthorized entry to a region). In our system, access control policies specify the access rights of individuals to different regions of the monitored space. Policy violations (detected via use of localization sensors such as RFID tags*, motion detection, etc) are used to trigger the video surveillance subsystem. The idea is that if a subject is authorized to be in a particular region (which is determined by their RFID tag) he/she may be hidden in the video (at various levels). This way, subjects' privacy is maintained until they violate their given rights (e.g. enter a region they are not authorized to be in). Video manipulation techniques such as masking are used to preserve the privacy of authorized subjects when the surveillance system is turned on. Our system is a fully implemented and achieves real-time video surveillance (approximately 25-30fps at our target resolution), as well as working with our masking techniques, incoming sensor data and access control infrastructure. This type of system is targeted at areas where there is heterogenous activity (e.g. hospitals, airports etc.) but also where there is a degree of regularity in the population (employees, law-enforcement etc.) with some level of adminstration. Using our framework, video surveillance

*A RFID (Radio-Frequency IDentification) tag is a tiny, relatively inexpensive device capable of transmitting a piece of static information across a distance. RFID tags are currently in use for mediating access to various regions, however it does not provide enough information to pinpoint the object being tracked within that space.

systems deployed in these areas can benefit from the extra sensor data, and the ability to specify and enforce spatio-temporal access control policies in a fairly transparent (to the users) manner.

In this work we build on our initial prototype of the system[2] which is a more generalized framework for data-collection, and focus on enhancements made to the video surveillance subsystem which better handle object detection, tracking and masking. The improvements we describe here allow the system to better handle more complex scenarios (i.e. dynamic lighting conditions, shadows, splitting and merging of subjects) and adapt better to changing environments. Accurately detecting moving objects in a scene with a low false alarm rate is very desirable in person-based surveillance systems. This becomes especially important when tracking in an outdoor setting, where motion from things like tree branches and bushes can add noise to traditional background subtraction-based methods. To tackle these issues, we utilize work on non-parametric background modeling[3] to better deal with both long and short-term changes in background conditions. Additionally, to better handle the disambiguation of subjects following merges we utilize color-template matching via a histogram-based intersection test.[4, 5]

To the best of our knowledge, the proposed system is the first of its kind that fuses information from various sensors with video information in implementing a privacy-protecting surveillance framework for pervasive spaces. The remainder of this paper is organized as follows; Section 2 describes our system architecture and outlines the role of RFID and access control in the overall framework. In Section 3 we discuss the techniques we have utilized for preprocessing and background modeling. Section 4 discusses our object tracker and accompanying privacy-protecting video processing techniques. In Section 5 we discuss related work and conclude with future work in Section 6.

2. SYSTEM ARCHITECTURE

Fig. 1 depicts a high-level outline of the system architecture.[2] The infrastructure comprises of the following components:

- **Sensing Module:** Processes data from incoming sensors. More specifically, a RFID control component deals with RF-related messaging arriving at the readers. Data from motion detection sensors are also processed here. A video input module handles the incoming video stream data from the various surveillance cameras.
- **Data Management Module:** Consists of a XML-based policy engine for access control. This policy engine interacts with a database subsystem consisting of profile and policy databases for the users.
- **Auxiliary Services:** A service library contains modules that provide auxiliary services on the sensed information (including the incoming video stream(s)). These include obfuscation, motion detection and object tracking modules. For example masking may be applied to the video stream before it is passed to the output module, if the subject has been authorized by the policy engine.
- **Output Module:** Handles customized reporting, logging and video rendering functionality.

Our system utilizes Radio Frequency Identification (RFID) and motion sensors to identify certain objects (people) in the video. We specify access control policies for individuals who are present in the space (on a regular basis) using an XML-based access control language (XACML).[6] When the system detects a RFID tag in a region, it contacts the access control subsystem to verify the policy for the corresponding tag, then passes the information to the video subsystem where the decision will be made to mask/show the corresponding object. For the purposes of this paper, an outline of this portion of the framework is provided but further details can be found in.[2]

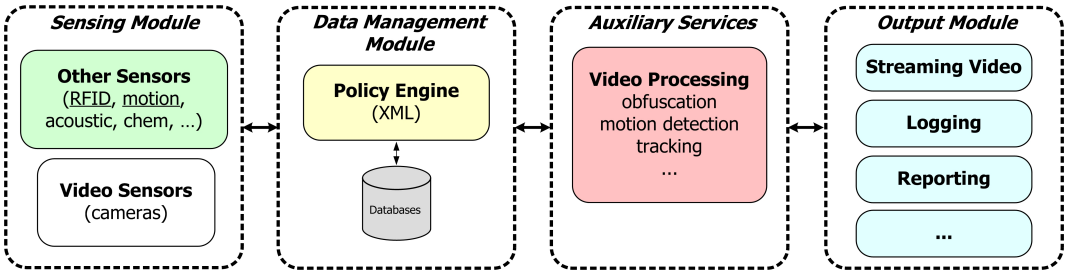

Figure 1. System architecture.

2.1. Radio Frequency Identification (RFID) Sensing

Radio frequency identification, or RFID, is a generic term for technologies that use radio waves to automatically identify people or objects. There are several methods of identification, but the most common is to store a serial number that identifies a person or object (and perhaps other information) on a microchip that is attached to an antenna (the chip and the antenna together are usually referred to as a RFID transponder). The antenna enables the chip to transmit the identification information to a reader. The reader converts the radio waves reflected back from the RFID tag, into digital information that can then be passed on to computers that can make use of it. A typical system consists of a tag (or a set of tags), a reader/receiver that can read and/or write data to these tags, and optionally a field generator. Our infrastructure equips each protected region with a field generator and motion detector. We place this equipment in the doorway to detect entry/exit to the region. When a person enters/exits this region, it will trigger the motion detector which in turn triggers the field generator. If there is no tag information associated with the motion, the signal sent to the reader is categorized as unauthorized and video surveillance of the region is triggered. If it contains valid tag information, then the policy engine will be contacted to evaluate the policy of the corresponding tag. This will be used by the video subsystem to render the corresponding object in the video.

2.2. Access Control Specification & Enforcement

This section gives a brief overview of the access control framework used by our system. We specify security policies for each tag using the eXtensible Access Control Markup Language (XACML).[6] These policies are processed by a policy engine which provides mediated access to a database containing the tag and policy information. The components of the access control model are the video-based objects, the potential users, and modes of access specified in the form $< s, o, m >$, where subject s (typically a person entering a region) is authorized to access object o (typically a region) under mode m, where the mode is associated with a particular privacy level. The privacy level indicates the type of masking technique used by the video subsystem. Currently our system supports four types of policies:

- Person without Tag: When a person without a tag enters a region, since there are no policies associated with him/her, a default policy of 'deny'(or the lowest privacy level) will be applied. So the person will be shown.
- Person with a valid tag: When a person enters with a valid tag, since there will be a policy which satisfies the spatio-temporal constraints, the system will return 'allow' (or the highest privacy level). So the person will be masked.
- Person with an invalid tag: When a person enters with an invalid tag, the system needs to consider two kinds of violations - spatial and temporal. The validation of the policies takes place in that order.
- Group-based authorization: This type of policy is useful for assigning inventory items to particular individual i.e. to specify that a certain object(e.g. laptop) can only be used by a certain person. The inventory items will have a different type of tag associated with it. We set a time threshold 't' for each event. If events occur within the threshold, it will be considered as a group authorization request. Even if a person has a valid tag, if he has an illegal inventory item, he or she will be shown in the video.

3. BACKGROUND MODELING

This section describes the functionality of a new background modeling technique utilized in our system, as well as other pre-processing carried out on the incoming video frames before they are passed to the object tracker. As motivation for choosing this new background modeling technique, we carried out analysis on the previous version of our video surveillance framework.[2] The prior iteration of the system utilized a pixel selection process to distinguish moving foreground pixels from the background. Here, each background pixel was modeled as one gaussian distribution. The sample scene used for this analysis is illustrated below in Fig. 2. Fig. 3 depicts the pixel distribution for both cases shown in the scene (labeled 1 and 2) with the pixels corresponding to the person entering the scene highlighted. The other set of pixels (the dominant peak) represent the background pixels. For point 2, the foreground pixels (i.e. the clothing worn by the person in the scene) are very close to the sample point (the window sill in this case). Therefore choosing the right threshold becomes very important in distinguishing the two. However this is a very empirical method and is prone to problems when the background changes or motion is introduced (e.g. shutter movement, shadows etc.) This simple model utilized in the prior version, while effective in most cases cannot handle any real dynamicity in the scene. We will describe a more advanced technique that overcomes a lot of the limitations of our initial model.

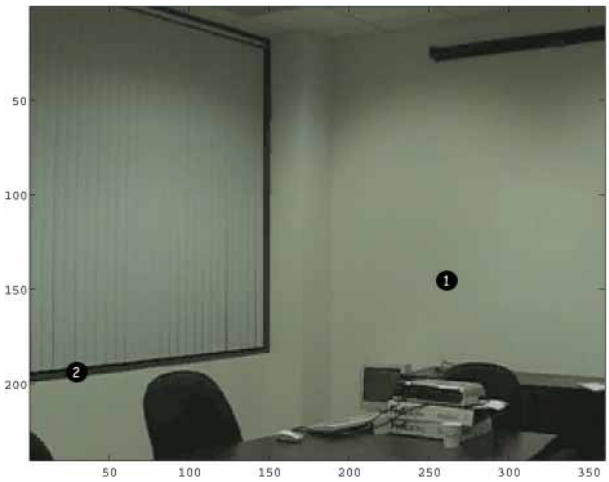

Figure 2. Example scene used for analysis.

3.1. Motion Detection

This module takes each new incoming frame and passes it to the background modeling component (described later) to differentiate foreground pixels from background. Once the foreground pixels are identified, they are passed to a simple 4-way connected component module which clusters foreground pixels into blobs. Blobs with small areas are considered noise and discarded from the incoming blob list. Finally, the blobs are passed to a blob-merging filter. In our testing we discovered that one person sometimes breaks into multiple blobs for reasons which may be attributed to camera noise, frame-rate jitter, or simply that the person's clothing may match very closely with the background. Regardless of the background model, this type of segmentation is possible and various techniques have been explored in the literature to handle it.[5] To overcome this problem, we implemented a filter which merges the broken blobs that lie directly on top of each other along the y-axis. This simple filter assumes that the camera is calibrated and pointed horizontally at the scene. By examining each blob's projection on the x-axis, blobs with overlapping projections can be identified and merged into one blob.

3.2. Background Model

As mentioned, our initial prototype[2] adopted a simple background model that detects moving foreground objects from a static background in an indoor setting using gaussian distributions to model background pixels. For the

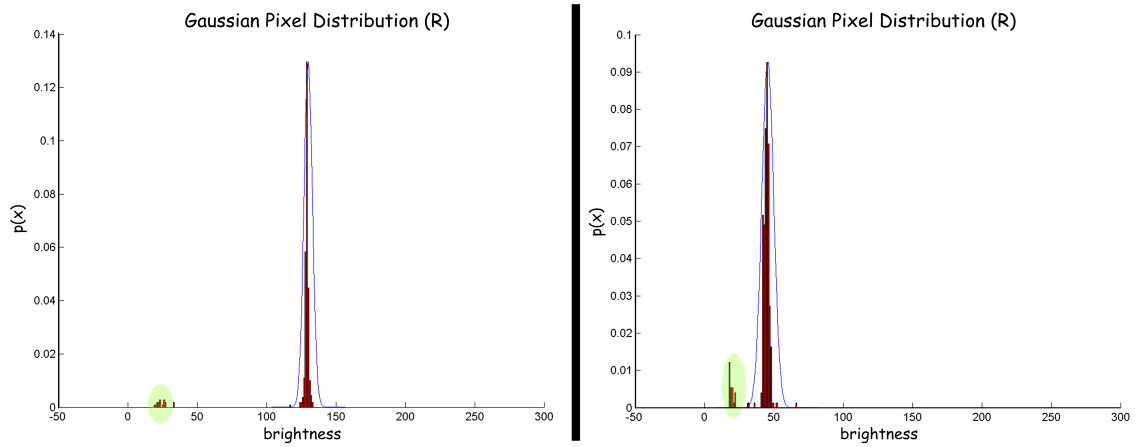

Figure 3. Gaussian pixel distribution (red channel only) for point 1 (left) and point 2 (right) for the scene shown in Fig. 2 Here a person enters the scene and moves from point 1 to 2. The pixel distribution for the two points in the video stream are shown.

current framework we adapted a non-parametric model for background detection, which was first proposed by Elgammal et. al.[3] The idea of the model is to capture the very recent history of a pixel in the image, and from this recent history calculate the probability of a new pixel intensity being a foreground pixel. The approach is based on kernel density estimation of the probability density function of the intensity of each pixel given a sample for this pixel. The idea of the model is to capture the very recent history of a pixel in the image, and from this recent history it calculates the probability of a new pixel intensity being a foreground pixel. For example, if $x_1, x_2, , x_n$ are the recent intensity samples for a particular pixel, the probability that this pixel has a value x_t at time t can be defined as (density estimation)[†]:

$$Prob(x_t) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{1}{2}\frac{(x_{t_j}-x_{i_j})^2}{\sigma_j^2}}$$

Once the $Prob(x_t)$ is estimated, the pixel is considered part of the foreground if the $Prob(x_t)$ is less than a probability threshold that is uniform over all the image pixels. This parameter is tunable in that it can be adjusted to achieve a desired percentage of false positives over the whole image.

We utilized this model because we needed a robust background model that handled camera noise, slow changes in illumination, and repetitive motion of background objects (e.g. shutters, which were a particular problem in our own test environment). The non-parametric background model is very robust and can run in real-time at a resolution of 360x240 and in testing we found that the memory usage was lower than comparable methods (e.g. mixture of gaussians[4]). Thus the model's features, real-time performance and implementation simplicity made it very attractive for use in the type of system we were targeting.

Features & Parameters: The background model includes solid detection of moving pixels in cluttered environments that have repetitive motion, such as moving tree branches, window shutters, flickering lights etc. The model can even learn and handle the noise introduced by cheaper cameras without the need to use any specific filters for noise removal. The model can also detect and filter out shadows of moving people and objects. A shadow suppression parameter which ranges between 0 and 1 (with 0 corresponding to no suppression and 1 to full suppression) needs to be set up.

The model also requires a probability threshold (between 0 and 1) that is uniform over all the pixels in the image; this threshold should be adjusted to achieve a desired percentage of false positives (a typical value

[†]The derivation of this formula can be found in[3]

is $10e^{-8}$). Furthermore, the model can adapt to slow changes in illumination and recently deposited/removed objects in the scene. For each frame, the model updates its background by using a binary mask where pixels with true values are masked out of the update. Two adaptation parameters, the number of samples, N and window size, W, are used to control how fast the model will adapt to changes in the scene. The model randomly selects N intensities from W to update a pixel selected for an update by the binary mask.

Design & Implementation: Our implementation of the model runs robustly in real-time (25-30fps) at a 360x240 RGB pixel resolution and requires less memory storage than other background models such as gaussian-based techniques[4] which model the variation of the scene over time and requires storing K Gaussian models in main memory. For the experiments described here we used Canon Optura 20 DV Cameras, capturing video at a resolution of 360x240. All processing was carried out on a uniprocessor Pentium 4 at 2.60 Ghz with 1 GB of RAM equipped with a Pinnacle firewire video capture card under Windows XP Professional. Microsoft's DirectX SDK (8.0) was utilized for interacting with the camera. The system was tested in an indoor office space. First, the background model was initialized as follows: the number of samples N was set to 20, window size W set to 100, shadow suppression to 0.2, and the probability threshold set to $10e^{-8}$. Also, the binary mask is initialized to zero so all the background pixels get updated during the modeling of the background. Then, we ran the model for 500 frames with no subjects in the scene to learn and build the background model. In each of the 500 iterations the model updates all the background pixels.

Updating the Background: There is a tradeoff when examining the rate at which to adapt the background to changing conditions.[3] If it is adapted too slowly, then the model tends to become inaccurate with low detection sensitivity. However, adapting too fast will lead to possibly adapting the subjects themselves and thus lead to inaccurate tracking. Once the background has been initialized and estimated, each incoming new frame is fed to the background model to perform background subtraction. This results in a binary mask where pixels with value 1 are considered foreground pixels and pixels with value 0 are considered background pixels. If the tracker module detected in the previous frames, that a new object had been deposited in the scene and it had been static for a predefined period of time, the pixels of this object in the binary mask are set 0 so it can be added to the background model. Note that once the object has been added to the background model, its pixels won't be recognized as foreground pixels. Yet the tracker will keep storing the object's information for a predefined period of time. Finally, the refined binary mask is passed back to the background model for updation.

Figure 4. An example of background adaptation using the non-parametric model

4. OBJECT TRACKING

The overall functionality of the system is illustrated in Fig. 5. Our tracker maintains three lists: temporary, object and person lists, and updates them for every frame. The temporary list holds subjects that have just entered the scene. The object list stores objects that have been deposited in the scene and have passed the silent motion detection test. The person list stores individuals who had entered the scene and failed the silent motion detection test (described later). Each entry in the lists have the following parameters which are updated for each frame; 1) minimum and maximum x and y pixel range to identify a bounding box around a blob. 2) the area a blob occupies in the frame (in pixels). 3) the blob's center of gravity (cog.x, cog.y). 4) the RGB pixel values for the blob. 5) privacy level information is obtained via the RFID subsystem which dictates how the entities are displayed in the scene. 6) previous blob information.

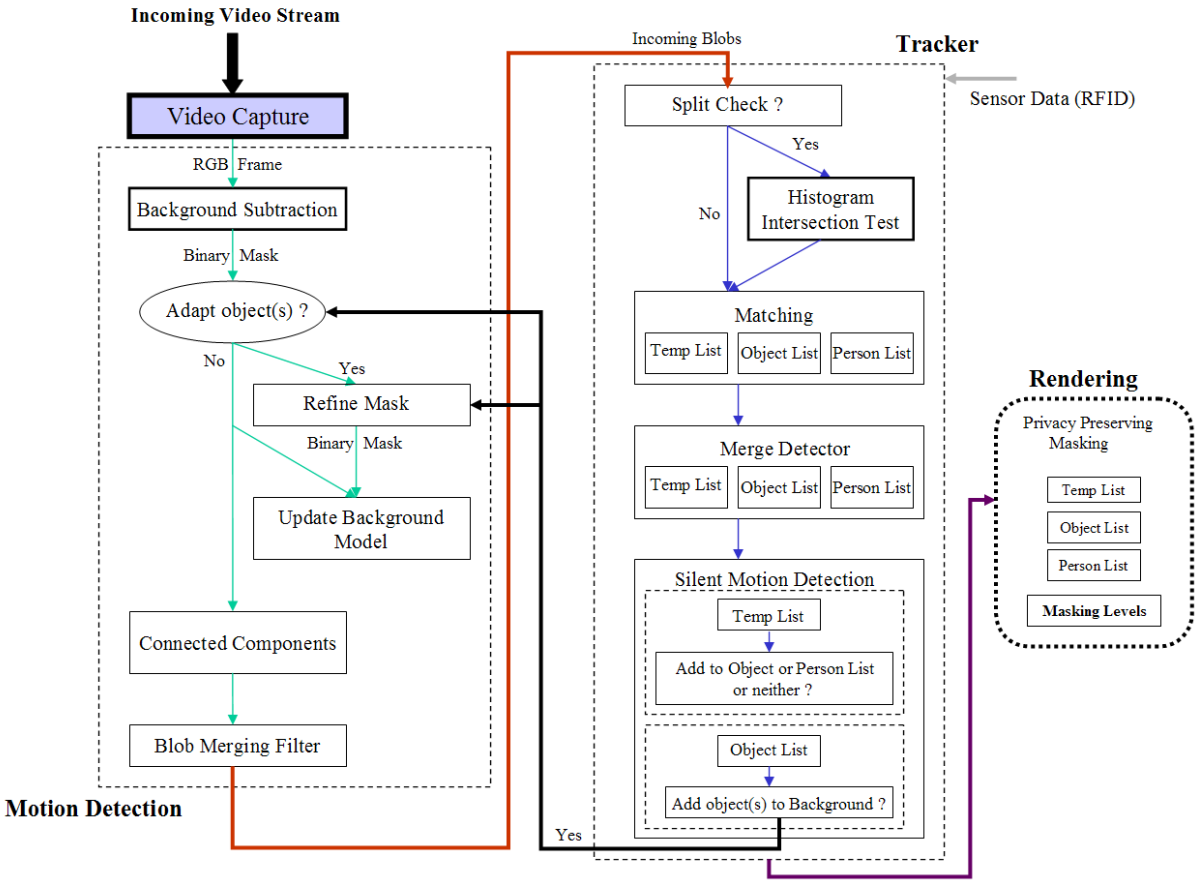

Figure 5. Flow diagram of the video architecture

```
bool Tracker :: doesInBlobMatchPerson(CBlob inBlob, Person inPerson) {

    if (((inPerson.info.cog.y <= inBlob.ymax) && (inPerson.info.cog.y >= inBlob.ymin)) &&
        ((inPerson.info.cog.x <= inBlob.xmax) && (inPerson.info.cog.x >=inBlob.xmin)))
        return true;
    else
        return false;
}
```

Figure 6. Function to check if a person's cog falls within one of the incoming blobs

For each frame the tracker tries to match entries in the list to new incoming blobs from the motion detector module. The matching is simply done by testing which incoming blob's bounding box encapsulates the center of gravity of the candidate blob (Fig. 6).

If a match is found, the entry's information is saved and updated with that of the new incoming blob. Otherwise, it is assumed that it left the scene and is thus removed from the list. It should be noted that only items in the temporary list or person list can leave the scene. Unmatched incoming blobs are added to the temporary list. If the merge detector detects that two entries are matched with the same new incoming blob, both entries get flagged as merged, and their information update process gets suspended until they split. The split is resolved by a simple histogram intersection test[5, 7] which is described later.

Merge Detection: After all the entries in the list have been matched to the new incoming blobs from the

Figure 7. The tracker correctly identifying a person (red) and an object (blue)

```
for(int i=1; i < numOfPerson; i++) {
    for( int j= i+1; j < numOfPerson; j++) {
        if ( personList[i].info.matchedIncomingBlobId == personList[j].info.matchIncomingBlobId ) {
            personList[i].info.merge = true;
            personList[j].info.merge = true;
        }
    }
}
```

Figure 8. Checking for a merge between entries.

motion and merge detectors, this module simply checks if any entries still match the incoming blob. If this is the case, these entries are flagged as merged entries. Fig. 8 shows a code fragment for checking merging between entries in the personList:

Silent Motion Detection: Entries in the temporary list are new subjects that have appeared recently in the scene and thus need to be examined (on a frame-by-frame basis) by the silent motion detector module to detect whether they are moving people or newly deposited objects. The module proceeds as follows; the motion of the entities in the temporary list is examined for K frames. If the item has moved J pixels in the previous K frames, the item is identified as a person and placed in the person list. However, if the item has moved less than J/4 pixels, then the item is placed in an object list and identified as an object that been deposited in the scene. The intuition here is that newly deposited objects in the scene must have moved a relatively small amount compared to other moving objects in the scene. Finally, if an entry in the object list stays static for some predefined time, then we consider this object as a candidate for adaptation to the background by setting its pixel location in the background model mask to zero. This way the background model will start adapting the object in next frame.

Histogram Intersection: In order to carry out this test, the prior color information of each subject in the previous frame need to be stored for each frame. When a split is detected, (e.g. a change in the number of blobs in the scene) we use the previous color information of the merged entries to determine a match with the incoming blobs. For example, if two entries had previously merged (in the person list) and we detect that they split, then we create a histogram of person 1 (A) and person 2 (B), and histograms of each of the incoming blobs from the motion detector. Then we do histogram intersection as defined by[5, 7] for person A and each new incoming blob. Let I be the histogram of one of the incoming blobs, and let A by the histogram for person 1. Then the intersection is defined as follows:

$$Hist(I, A) = \frac{\sum_{j=1}^{n} min(I_j, A_j)}{\sum_{j=1}^{n} A_j}$$

$Hist(I, A)$ will be a fraction between 0 and 1, and will be computed for all incoming blobs. We then pick the blob that has the highest histogram intersection with A. Then we assign this blob information to person 1, the same is done for person 2.

In our experiments we discovered that taking only the previous entry's color information before the merge happens, tends not to be robust. Therefore, a better solution is to store an average histogram of an entry's

color over K frames. Then use this averaged histogram to test against the new incoming blob when merging and splitting occurs. A drawback of this type of approach is that for the technique to work well, merged subjects must have somewhat distinguishable colors.

Figure 9. Merging and splitting using the histogram intersection test

4.1. Privacy-Protecting Techniques

When a new blob is detected by the tracker, it is also assigned authorization information (masking level) via the RFID subsystem. During the rendering process, the blob's corresponding masking level is applied to the outgoing video stream. In the case of a merge, the highest masking level among the persons involved in the merge is applied to the corresponding blob. For example if a person p_1, with privacy level L_1 merges with a person p_2, with privacy level L_2, then the merged blob is masked using L_2. Our current system supports 4 masking levels (illustrated in Fig. 10):

- No masking (Level 0).
- Blur-based pixel coloring: Here, pixels are colored as $pixel(x,y) = (backgroundMean(x,y) * t_1) + (currentPixel(x,y) * t_2) + (rgbColor * t_3)$, where $t_1 + t_2 + t_3 = 1$ and take values between [0,1] (Level 1).
- Pixel coloring: replaces the target's pixels with one (RGB) color that hides all the information contained within a persons outline (Level 2).
- Bounding box-based masking: In this case the bounding box around the target is colored with the average background frame values (Level 3).

5. RELATED WORK

Privacy concerns in video surveillance have not really been addressed in video processing research. Furthermore, these techniques require efficient implementations to process real-time video streams (usually MPEG-1 or MPEG-2). Variations of background subtraction has been used as a technique for foreground/background segmentation for long video sequences.[8] As a relatively simple method, it works fairly well in most cases but its performance depends heavily on the accuracy of the background estimation algorithms. We utilize a form of background subtraction for our video subsystem but it is not the focus of the work here. Issues dealing with illumination changes, shadows, dynamic backgrounds are not addressed here, but have been investigated in other work. The addition of these techniques will serve to make our video subsystem more robust. Relevant areas in real-time motion tracking, image/face recognition[9] and video data indexing have been studied but rarely[10] infused with techniques to preserve privacy.[11] proposed a quasi-automatic video surveillance approach based on event triggers to generate alarms and overcome the drawbacks of traditional systems. We are adopting a similar vision in providing surveillance only when activity is taking place within a region. A new approach known as experimental sampling was proposed in[12] which carries out analysis on the environment and selects data

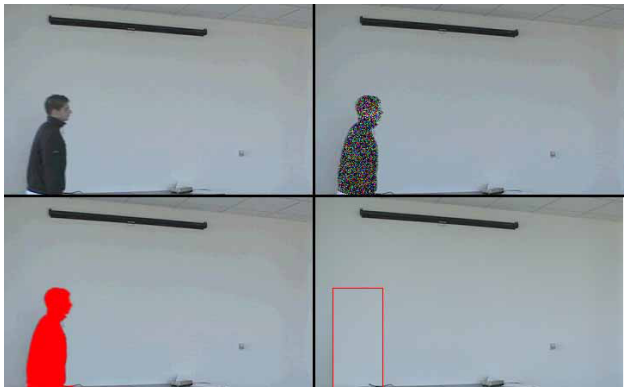

Figure 10. Privacy-protecting masking techniques for video utilized by our system. They represent different levels of user privacy. A frame of the original video is shown (top-left), followed by a noise/blur filter (top-right). A pixel-coloring approach is shown (bottom-left) followed by a bounding-box based technique (bottom-right) which hides details such as gender, race or even dimensions of the person in question.

of interest while discarding the irrelevant data. Our framework utilizes different modalities of sensor data in providing information that assist the video subsystem in detecting and classifying anomalies.

VideoIQ[13] by GE Interlogix is the latest state of the art commercially available surveillance system. However, *VideoIQ* does not focus on the privacy of subjects being monitored. The system learns the motion of the background environment that it is focused on, (such as moving tree branches and rippling water) clusters moving foreground pixel regions and then classifies them into humans/ non-humans. The system then tracks the classified regions from frame to frame and stores the information in a database. The system also allows alarms or alerts that system administrators can set up to detect various types of events. As documented, *VideoIQ* works with multiple cameras that track subjects captured by each camera independently. The system requirements are a PC with Pentium 2.8 GHz processor, 512 MB of memory and 40 GB of hard disk space. What we are developing with our system is something more fundamental, as well as utilizing the notion of sensor fusion and privacy preservation. Our goal is to have this framework be general and flexible enough to eventually be pushed into cameras themselves and allow policy specification that is powerful enough so that system administrators can define customized spatio-temporal policies and have them be carried out in an efficient manner while still preserving the privacy of the subjects as much as possible.

Another complete surveillance system that again does not focus on privacy at all is the tracker developed at the University of Reading in the United Kingdom.[14,15] The context for this tracker was an underground station with pedestrians as the subjects. The system uses a simple average image background subtraction algorithm for modeling the background. The highlight of this system is that it uses an active shape tracker algorithm that tries to track pedestrian outlines. The active shape tracker learns a space of pedestrian outlines via preprocessing and then uses this information for real-time tracking. A simple head-tracking algorithm is also used for disambiguation of subjects. The system does not provide any background adaptation for objects deposited/removed from the scene. It also cannot identify pedestrians wearing clothing that closely matches the background scene.

The area of work dealing with privacy preservation in media spaces is relatively new, and a lot of the related work is in the domain of computer-supported corporative work (CSCW).[10,16] Of particular interest is the work presented by Boyle et. al[17] which utilized blur and pixelization filters to mask sensitive details in video while still providing a low-fidelity overview useful for awareness. Specifically they analyze how blur and pixelize video filters impact both awareness and privacy in a media space. However, the limitation of these techniques are that the filters are not applied to the individual objects in the video but to the entire video frame, which makes enforcing separate policies and distinguishing between authorized and unauthorized personnel impossible. In our approach, we apply the processing techniques on a per-object basis and apply this effect in real-time as the object is tracked in the space. Previous work utilizing eigenspace filters[18] proposed a way to mask out potentially sensitive action associated with an individual by using a set of pre-defined base images to extract a representation

of the person (face) by taking an inner product of the video images with those base images stored in a database. This technique though useful, relies on capturing and storing base images of the potential subjects, which may be both infeasible as well as against the notion of trying to store as little identifiable information about individuals in the space as possible.

In regards to related work in the area of sensors and sensor-fusion, there has an increased interest in RFID-related research, both in the academic and commercial sectors.[19–23] Additionally, there exists a large body of work on on policy specification and access control in XML.[24, 25] We utilize these technologies in the design and implementation of our system architecture.

6. FUTURE WORK & CONCLUSIONS

Even though we have realized a fully-functional implementation of our framework, the deployment of a such a system should eventually be pushed to the camera level. In our implementation, we tightly coupled the processing capabilities (of the PC) to the camera and processed everything in real-time with no archival of the original video data (only the processed video stream is available). Ideally this processing capability should reside at the camera level to make the privacy preservation in media spaces more acceptable to the end-users. Optimization of the algorithms used here for object tracking and masking for privacy is a key component of such a realization as well as the possible use of MPEG-4 video. MPEG-4 has a superior compression efficiency, advanced error control, object based functionality and fine grain scalability making it highly suitable for streaming video applications. Recently MPEG-4 has emerged as a potential front runner for surveillance applications because of its layered representation, which is a natural fit for surveillance tasks as they are inherently object-based. Additionally, we would like to utilize existing work on multi-camera video surveillance in our framework which gives us the ability to monitor larger regions more extensively as well as provide some approximate tracking information for objects (or groups).

ACKNOWLEDGMENTS

Support of this research by the National Science Foundation under Award Numbers 0331707 and 0331690 is gratefully acknowledged. We would like to thank the members of RESCUE for their helpful suggestions and assistance in testing the infrastructure. We would also like to thank Professor. A. Elgammal for his suggestions and insight in helping us utilize his non-parametric background model implementation for our system. Finally, we would like to acknowledge Dr. Miguel Sainz for his suggestions and image processing libraries that were utilized in our implementation.

REFERENCES

1. Privacy International, "Privacy International: Video Surveillance." http://www.privacyinternational.org/issues/cctv/index.html.
2. J. Wickramasuriya, M. Datt, S. Mehrotra, and N. Venkatasubramanian, "Privacy Protecting Data Collection in Media Spaces," in *ACM Multimedia 2004*, 2004.
3. A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction.," in *ECCV (2)*, pp. 751–767, 2000.
4. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), pp. 747–757, 2000.
5. S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people," *Computer Vision and Image Understanding: CVIU* **80**(1), pp. 42–56, 2000.
6. S. Godik and T. M. (Eds), "eXtensible Access Control Markup Language (XACML) 1.0 Specification Set." OASIS Standard, 2003.
7. M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vision* **7**(1), pp. 11–32, 1991.
8. M. Harville, G. Gordon, and J. Woodfill, "Foreground Segmentation using Adaptive Mixture Models in Color and Depth," in *Proc. of the IEEE Workshop on Detection and Recognition of Events in Video*, 2001.
9. W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.* **35**(4), pp. 399–458, 2003.

10. Q. Zhao and J. Stasko, "Evaluating Image Filtering Based Techniques in Media Space Applications.," in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'98, Seattle)*, pp. 11–18, ACM Press, New York, NY, 1998.
11. L. Marchesotti, L. Marcenaro, and L. Regazzoni, "A Video Surveillance Architecture for Alarm Generation and Video Sequences Retrieval," in *Proc. of ICIP2002*, 2002.
12. J. Wang, M. S. Kankanhalli, W. Yan, and R. Jain, "Experiential Sampling for video surveillance," in *1st ACM SIGMM international workshop on Video surveillance*, pp. 77–86, ACM Press, 2003.
13. GE Industrial, "VideoIQ." http://www.geindustrial.com/ge-interlogix/kalatel/videoiq.html.
14. N. Siebel, *Design and Implementation of People Tracking Algorithms for Visual Surveillance Applications*. PhD thesis, Department of Computer Science, The University of Reading, Reading, UK, 2003.
15. N. Siebel and S. Maybank, "Fusion of Multiple Tracking Algorithms for Robust People Tracking," in *7th European Conference on Computer Vision (ECCV 2002)*, **4**, pp. 373–387, Springer Verlag, 2002.
16. P. Dourish and V. Bellotti, "Awareness and Coordination in Shared Workspaces.," in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'92, Toronto)*, pp. 107–114, ACM Press, New York, NY, 1992.
17. M. Boyle, C. Edwards, and S. Greenberg, "The Effects of Filtered Video on Awareness and Privacy," in *Proc. of CSCW*, pp. 1–10, 2000.
18. J. Crowley, J. Coutaz, and F. Berard, "Things That See," in *Communications of the ACM, 43:3 (March)*, pp. 54–64, ACM Press, New York, NY, 2000.
19. A. Juels, R. L. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy," in *Proc. of ACM CCS*, 2003.
20. A. Juels and R. Pappu, "Squeling Euros: Privacy Protection in RFID-Enabled bank notes," in *Financial Cryptography*, **LNCS 2742**, pp. 103–121, Springer-Verlag, 2003.
21. A. Juels, "Yoking-Proofs for RFID tags," in *First International Workshop on Pervasive Computing and Communication Security*, 2004.
22. K. S. M. Ohkubo and S. Kinoshita, "A Cryptographic Approach to a Privacy Friendly Tag," tech. rep., NTT Laboratories, 2003.
23. S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," in *Security in Pervasive Computing*, *Lecture Notes in Computer Science* **2802**, pp. 201–212, 2004.
24. E. Bertino, S. Castano, E. Ferrari, and M. Mesiti, "Controlled Access and Dissemination of XML Documents.," in *WIDM'99 ACM*, 1999.
25. E. Damiani and S. di Vimercati et. al, "A Fine-Grained Access Control System for XML Documents.," in *ACM Transactions on Information and System Security (TISSEC)*, **5, num 2**, pp. 169–200, 2002.