Presented at the Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009) from 30th August to 2nd September, 2009 at Como in Italy

Autonomous Real-time Surveillance System with Distributed IP Cameras

Kofi Appiah, Andrew Hunter
and Jonathan Owens
Department of Computing and Informatics
University of Lincoln
Lincoln, UK
Email: ahunter@lincoln.ac.uk
Tel: +44 (1522) 886456

Philip Aiken and Katrina Lewis
SecuraCorp
Kirkland
WA 98083-0643
USA
Email: phila@securacorp.com
Tel: +1 (425) 822-3636

***Abstract*—An autonomous Internet Protocol (IP) camera based object tracking and behaviour identification system, capable of running in real-time on an embedded system with limited memory and processing power is presented in this paper. The main contribution of this work is the integration of processor intensive image processing algorithms on an embedded platform capable of running at real-time for monitoring the behaviour of pedestrians. The Algorithm Based Object Recognition and Tracking (ABORAT) system architecture presented here was developed on an Intel PXA270-based development board clocked at 520 MHz. The platform was connected to a commercial stationary IP-based camera in a remote monitoring station for intelligent image processing. The system is capable of detecting moving objects and their shadows in a complex environment with varying lighting intensity and moving foliage. Objects moving close to each other are also detected to extract their trajectories which are then fed into an unsupervised neural network for autonomous classification. The novel intelligent video system presented is also capable of performing simple analytic functions such as tracking and generating alerts when objects enter/leave regions or cross tripwires superimposed on live video by the operator.**

I. Introduction

Video surveillance systems have since the 1970s consisted of National Television System Committee (NTSC) or Phase Alternating Line (PAL) analogue cameras connected over a coaxial cable network to VHS tape recorders or digital video recorders (DVRs) in a monitoring station. Such surveillance systems are often comprised of black and white, poor quality analogue videos with little or no signal processing, recorded on the same cassette. Most of the recorded images are of insufficient quality to hold as evidence in a law court. It is also expensive to have human operators monitoring real-time camera footage 24/7. The effectiveness and response of the operator is largely dependant on his/her vigilance rather than the technological capabilities of the surveillance system [1]. Events and activities can be missed, should the concentration level of the operator drop; attentional levels drop significantly after 15 minutes of inactivity in the scene.

The advent of high resolution digital IP surveillance cameras, connected via the internet to a remote security monitoring station, enables a new approach that draws attention to events identified in the camera scene. IP (Internet Protocol) cameras coupled with the introduction of video content analysis or video analytics promise to extend the reach of video beyond security in a local area into a wide area surveillance system. Such automation and wider coverage will significantly reduce the drudgery workload on law enforcement agencies, thus making it possible for them to concentrate on the thing they do best: responding to suspicious events [2].

The Algorithm Based Object Recognition and Tracking (ABORAT) system presented in this paper is a vision-based intelligent surveillance system, capable of analyzing video streams. These streams are continuously monitored in specific situations for several days (even weeks), learning to characterize the actions taking place there. This system also infers whether events present a threat that should be signalled to a human operator. However, the implementation of advanced computer vision algorithms on embedded systems with battery life is a non-trivial task as such platforms have limited computing power and memory [3]. The concept of the ABORAT system is to apply intelligent vision algorithms on images acquired at the system's edge (the camera), thus reducing the workload of the processor at the monitoring station and the network traffic for transferring high resolution images to the monitoring station.

II. Related Systems

Increasing the number of video sources or channels for a single human observer to monitor and identify critical situation is becoming a norm in today's surveillance systems. This not only increases the burden on the human observer, but implies that critical situations in the scene are easily missed. A class of new technologies referred to as Intelligent Video Surveillance (IVS) makes it possible for computers to monitor video feeds in real-time. The system signals the human operator when an event which poses a threat develops [2]. An IVS systems should be able to keep track of objects in a camera view (*identity tracking*), and determine where they are (*location tracking*) and what they are doing in the scene (*activity tracking*) [4].

There are a number of commercially available IVS systems, in addition to those in the research literature. *ActivEye* [5], offers an Intelligent Video Management (IVM) software for security, traffic management and business operations. The system is capable of detecting up to 35 events; which include the differentiation between humans, automobiles and environmental noise in real-time under varying weather conditions. The system also promises the ability to detect normal behaviour patterns of objects, even though details have not been given.

Perceptrak, video analytics software from Cernium [6] provides behaviour-recognition for the security industry. The intelligent video surveillance technology is capable of identifying up to 16 events, which includes the detection of people, vehicle and other objects. Similar to ActivEye, it also offers an advanced recording facility as well as issuing automatic alerts when specific threats are identified.

VideoIQ from GE Security [7] also offers an intrusion detecting system capable of accurately detection human actions. *FenceWATCH*, from Guardian Solutions [8] is also an intelligent surveillance system capable of learning normal and abnormal activities within the camera's field of view. Other commercially available software-based video analytic solutions includes *Nextiva Analytics* [9], *SmartCatch* [10], *VisionAlert* [11], and *Smart IQ* [12], capable of detecting and counting people/vehicles, performing behavioural analysis and tracking as well as generating alerts.

A machine vision system with an image sensor alongside an integrated circuit with some computational power for image processing is referred to as a *smart camera* [13]. A commercially available smart camera solution for security surveillance is presented by ObjectVideo [14] in the form of software *ObjectVideo VEW* running on a DSP-based hardware architecture. The ObjectVideo OnBoard is used in conjunction with the VEW to pro-actively analyze video and produce alerts and other actionable information on the basis of user-defined rules. The intelligence offered by the ObjectVideo systems is similar to the commercially available Video Motion Anomaly Detection (VMAD) [15], capable of learning normal scene behaviour. Trigger [16], a product from *Mate-Media Access technologies* has a processor placed next/near to the video camera for intelligent video analysis to spot and pass to the central control only events that require the attention of the operator.

In this paper, we present a smart camera system (ABORAT), with an intelligent processing architecture (ABORGuard) Video Processing Unit (VPU) placed next to an IP camera for processing real-time images, which will then generate and send alerts to the control/monitoring station (*ABORGuard Server*). The ABORAT system detects and tracks moving objects such as persons/automobiles, collects their trajectories and classifies the behaviour using an autonomous behavioural identifier. For such an "Event-Based" IVS, the network bandwidth is significantly reduced, as images are only transmitted when useful to the operator. The system also records live video footage for review purposes.

The rest of this paper is organised as follows. Section III gives details of the ABORAT system architecture and platform integration. Section IV gives details of the image processing and algorithmic design of the ABORAT system; Section V presents experimental results. Conclusions and proposals for future work are presented in Section VI.

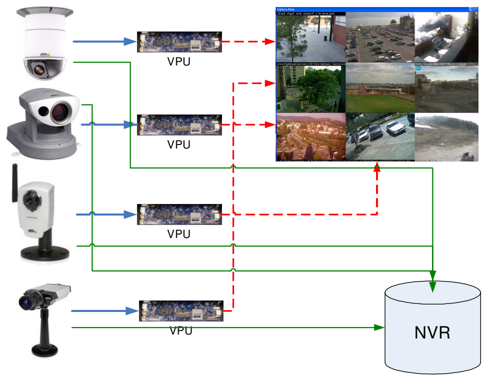


Fig. 1. The distributed nature of the ABORAT system, showing four different IP cameras, each with it's embedded processing unit capable of processing the video and identifying events to be transmitted.

III. Systems Architecture

There are two broad classes of IVS: centralized and distributed [17]. Centralized IVS processes video and other sensor information on a central server. Distributed IVS have "intelligent" cameras/sensors, capable of processing the video and extracting relevant information to a server. The ABORAT systems is an example of a distributed IVS, which processes sensor data as they are collected. The ABORAT system comprises of multiple sensor units, each associated with a video processor unit (VPU), a communication unit and a monitoring unit. The various components, their functions and the requirements for video surveillance are presented in this section.

A. Overview

There is a proliferatino of surveillance cameras around the world, and of recorded video footage from these, but very few cameras get watched or videos reviewed due to cost considerations. As a result, events and activities are missed, and suspicious behavior remains unnoticed. The ABORAT system consists of a distributed network of fully bi-directional IP cameras capable of communicating with a server via the VPU placed "next to" each camera to perform tasks, such as motion detection, object tracking and behavior detection. The system is based on a distributed client-server architecture offering true convergence of surveillance over local or wide area networks (LANs/WANs). As shown in Figure 1, the system is capable of dealing with large numbers of cameras from different camera manufacturers, distributed over a very large, wide network, making it possible for authorized users to access real-time video data at any time.

The ABORAT hardware platform uses an image sensor as the primary source of input, and hence appropriate image quality is essential to the performance of the entire system.

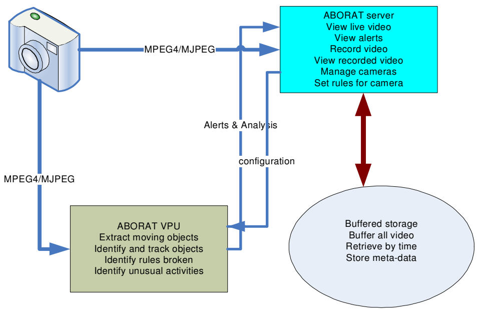


Fig. 2. Block Diagram of ABORAT System

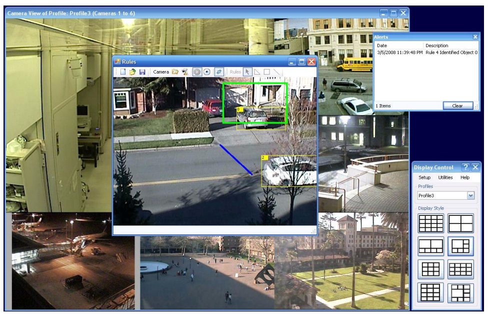


Fig. 3. A view of the ABORAT server software

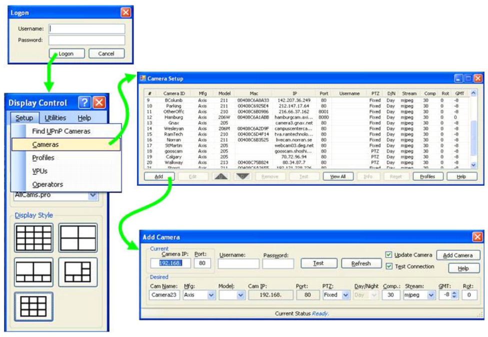


Fig. 4. A step-by-step demonstration on how to add an IP camera

A distributed image sensor with high dynamic range and little blur, capable of transmitting over an internet protocol has been chosen. The digital image from the IP camera is uncompressed by a dedicated Digital Signal Processor (DSP) embedded in the VPU for image processing. The VPU also has a general purpose microprocessor running at a maximum speed of 520MHz for the intelligent image processing. Generated alerts are transmitted over internet to the monitoring unit. The monitoring unit has Network Video Recorder (NVR) software, which runs continuously to record incidents across multiple cameras. Figure 2 is an overview of the entire ABORAT system.

B. Platform Integration

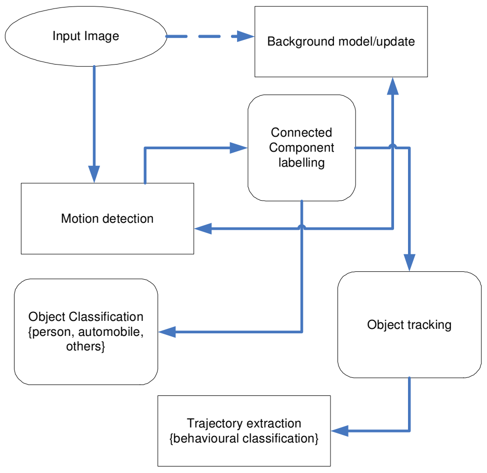
The ABORAT system is based on off-the-shelf components including a DSP, a low-power general-purpose processor, network peripherals, and efficient storage. All components are interfaced using custom developed ABORAT software, which allows security personnel to specify his/her preferred analytics algorithms. The system works with any mix of local and remote legacy analogue cameras and/or the latest IP-based digital cameras, both available from multiple manufacturers. The installation and setup of IP cameras is automated using network Plug-and-Play, thus the server application automatically detects and add IP cameras within the network. Power consumption, a major design constraint on every embedded system governs the choice of an Intel Xscale technology, the PXA270 clocked at 520MHz with low power consumption and heat dissipation.

The compressed video stream from the IP camera is sent to the VPU's dedicated DSP for decompression. The result is then placed in First-In First-Out (FIFO) buffer memory for access by the PXA270 with 104MHz memory bus. The raw image data is processed to generate alerts for every single frame. This processing includes the extraction of every moving object from a modelled background, followed by the extraction of parameters of the objects. Analysis of the behaviour of each object is then performed, and if an object poses a threat an XML alert-log file is generated and sent to the monitoring station. For flexibility and fault-tolerance, the communication link between the ABORAT VPU and the server can be wireless-LAN or Ethernet. The transmission of visual data over camera-based wireless sensor networks (WSNs) is challenging due to the high computational and bandwidth requirements [18].

Live video feeds from all of the IP-cameras are sent to the custom ABORAT server as well as to the NVR for display and recording respectively. The display screen on the monitoring server changes to highlight a video stream for which an anomalous condition has been identified. Every camera has an associated VPU. The server also offers On-Screen Rules Definition (OSRD) for drawing of polygons, rectangles, or tripwires over top of live video on the Operator Console. The resulting rules are transmitted to the VPU for execution, along with autonomous analytics. Figure 3 is a snapshot of the ABORAT server software showing that the automobile with *label 0* has violated rule *number 4*, which restricts moving objects from entering the area marked in green. The user friendliness of the ABORAT server software is demonstrated in Figure 4, showing how easily a local or remote IP-based network cameras can be added. Local cameras are those that the administrator, given appropriate permissions, may manage on their local subnet. The remainder will be remotely accessed cameras (located at another site or building) whose parameters may not be changed.

IV. Algorithmic Design

The detection, matching and classification of human appearance is a challenging problem [19] A further weakness of video detection is the limitation of conventional camera systems to operate under wide dynamic range lighting, which is typical for outdoor applications. Therefore, real-time video-based tracking application are mostly constrained with limited



Input Image
Background model/update
Connected Component labelling
Motion detection
Object Classification {person, automobile, others}
Object tracking
Trajectory extraction {behavioural classification}

Fig. 5. The operational modules of the ABORAT system.

resources at the price of the optimal performance [20]. Detail of the algorithms used in the ABORAT systems is given in this section. An outline of the various modules implemented is depicted in Figure 5.

A. *Motion Detection*

The first stage in processing for many image applications is the segmentation of (usually) moving objects with significant difference in colour and shape from the background, to determine their location in the image scene. Where the camera is stationary, a natural approach is to model the background and detect foreground objects by differencing the current frame with the background. All pixels that are not similar to the modelled background are referred to as the "foreground" [21]. There are situations in which some background objects (like water ripples or moving foliage) are not perfectly static and induce local noise. Many authors have thus proposed modelling each background pixel with a probability density function (PDF) learned over a series of training frames [22]. The ability to extract moving objects in real time from live video data using an embedded processor is our primary aim.

Two simple algorithms have been designed to match the output from the camera. The first version relies on RGB raw data and the second on YUV. Shadows can easily be detected in YUV rather than RGB, but it requires a little more processing to convert the RGB camera data from RGB to YUV. Fortunately, the dedicated DSP on the VPU for video decompression can easily generate YUV data at no extra cost to the image processor (PXA270), making it possible to use the second approach.

Following Grimson [23], we maintain a number of clusters, each with weight w_k, where $1 \leq k \leq K$, for K clusters [24]. Rather than modelling a Gaussian distribution, we maintain a model with a central value, c_k of 11-bits (8 bits integer part and 3 bits fractional part). We use an implied global range, $[c_k - 15, c_k + 15]$, rather than explicitly modelling a range for each pixel based on its variance as in [23]. The weights and central values of all the clusters are initialised to 0.

A pixel $X = I(i,j)$ (where X is 11-bit fixed-point) from an image I is said to match a cluster, k, if $X \geq c_k - 15$ and $X \leq c_k + 15$. The highest weight matching cluster is updated, if and only if its weight after the update will not exceed the maximum allowed value (i.e. $w_k \leq 64$, given the data width of the weight as 6 bits). The update for the weight is as follows:

$$w_{k,t} = \begin{cases} \frac{63}{64} w_{k,t-1} + \frac{1}{64} & \text{for the matching cluster} \\ \frac{63}{64} w_{k,t-1} & \text{otherwise} \end{cases} \tag{1}$$

The central values of all the clusters are also updated as follows:

$$c_{k,t,i,j} = \begin{cases} \frac{7}{8} c_{k,t-1,i,j} + \frac{1}{8} X_{i,j} & \text{matching cluster} \\ c_{k,t-1,i,j} & \text{otherwise} \end{cases} \tag{2}$$

where $c_{k,t,i,j}$ is the central value for cluster k at time t for pixel (i,j).

If no matching cluster is found, then the least weighted cluster's central value, c_K is replaced with X; its weight is reset to zero. The way we construct and maintain clusters make our approach gradually incorporate new background objects. This is similar to [25] and hence the insertion delay is $2^3 = 8$ frames in our case.

The K distributions are ordered by weight, with the most likely background distribution on top. Similar to [23], the first B clusters are chosen as the background model, where

$$B = arg_b \; min(\sum_{k=1}^{b} \omega_i > T). \tag{3}$$

The threshold T is a measure of the minimum portion of the data that should be accounted for by the background. The choice of T is very important, as a small T usually models a unimodal background while a higher T models a multi-modal background.

For the YUV version, we use a 32 bit fixed-point data representation. We use three reference values per pixel to represent the background data: $refY, refU, refV$. Again, two threshold values tY, tUV are used to classify the input pixels as background or foreground in the YUV plane. The YUV background data is represented as a point in a 3D space. If the difference between the input Y and $refY$ exceeds the threshold tY or the difference between the input UV and $refUV$ (calculated using the Manhattan distance) is greater than the threshold tUV, the pixel is considered as foreground. The choice of YUV and RGB background differencing method is dependent on the scene (indoor, outdoor and lighting intensity).

B. *Object Tracking*

Object segmentation gives a single frame snapshot of the current state of the world as expressed in the simple dichotomy of pixels as foreground or background. The job of the object tracker is to establish and maintain the temporal correspondence of an object in the world, given its frame-by-frame representation. The object tracker must be able to handle the uncertainty manifest by the object segmentation algorithm. A sub-threshold colour difference between the background

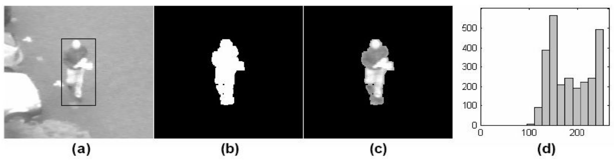


Fig. 6. (a) Object delineated by minimum bounding rectangle. (b) Area, height and width calculated from binary silhouette. (c) Segmented grey-scale silhouette used to calculate grey-scale histogram (d), with grey values along the abscissa and number of pixels on the ordinate.

and parts of the moving object will result in incomplete segmentation. Similarly, partial segmentation will also result from occlusion of the object by static scene elements, such as parked vehicles. Multiple objects may be segmented as single connected-components if they are in close proximity in the image plane, requiring disambiguation of the merged objects. Thus, there may be multiple connected-components associated with each object, or multiple objects may be associated with a single connected component.

Connected-component labelling is normally used to convert the binary image generated from the motion detection unit into a symbolic one with each connected component having a unique numeric label. A highly optimised version of the conventional connected-component labelling algorithm has been implemented. The use of individual pixels which create a very large equivalence table has been replaced with blocks of contiguous pixels in a single row (run-length encoded format). This gives a significant improvement in processing time for an image with medium to low noise level. The complexity approaches that of the pixel-based algorithm for very noisy images.

The system includes a robust tracker that is capable of handling partial occlusion [26]. The system makes use of all available visual information to successfully track moving objects. Objects are tracked from frame to frame using a feature vector, $f_i = [a, h, w, g]$, consisting of the area, height, width and grey-scale histogram as well as direction of movement, as illustrated in Figure 6. The system keeps track of the total number of objects in the previous scene to identify objects entering and leaving the scene. It is also able to detect when a single object splits into parts, mainly due to over-segmentation. Similarly, when two objects merge as a single object the tracker is able to detect this under normal conditions. The implementation outperforms other trackers solely based on Kalman filter and extended Kalman filter [27].

Central to the tracking algorithm is the concept of *difference* between object and silhouette (connected component) feature vectors. The difference between the feature vectors of object Q and silhouette S is defined as in equation 4, which is a four element vector comprised of the absolute differences of the area, height and width of the object and silhouette and the scalar length of the grey-scale histogram difference vector.

$$\mathbf{d}(Q,S) = \big[|a_Q - a_S|, |h_Q - h_S|, |w_Q - w_S|, \sqrt{(\mathbf{g}_Q - \mathbf{g}_S)\cdot(\mathbf{g}_Q - \mathbf{g}_S)}\big] \tag{4}$$

The distances in the image plane between the expected centroid of every object Q (calculated from the centroid and velocity of object Q at time $t-1$), and the centroids of the segmented silhouettes, S, are calculated. This initial measurement is used to form a valid-match matrix, V, based on an object's expected location and an arbitrary search radius around that position. The search radius, r, establishes a limit on the number of possible matches that can be evaluated by the object-to-silhouette assignment algorithm. The initial match assignment for an object may only be made with a silhouette within the valid match radius, and when dealing with silhouette fragmentation, the algorithm restricts the search to silhouette fragments lying within the match radius. V is a matrix of dimension $\{n, m\}$, where n is the number of tracked objects and m is the number of segmented silhouettes.

$$c(Q,S) = \sum_k \frac{d_k(Q,S)}{f_k(Q)} \tag{5}$$

The *cost* of every object-to-silhouette assignment having a non-zero entry in matrix V is given by the scalar value $c(Q,S)$ as in equation 5, where $d_k(Q,S)$ is the k^{th} element of the difference vector calculated between object Q and silhouette S, and $f_k(Q)$ is the k^{th} element of the feature vector of object Q. The histogram element in the feature vector in the denominator of expression (4.7) is transformed into a scalar value by calculating the Euclidean length of the vector. The elements of the object feature vector in the denominator have the effect of scaling the values of the difference vector, assuming that the within population coefficients of variation are roughly equal for the separate feature vector elements.

C. Trajectory Classification

Abnormal activity detection has been divided into two categories – parametric and non-parametric – by Zhou et. al [28]. The parametric approach models normal and abnormal activities using visual features like position, speed and appearance, while the non-parametric learns the normal and abnormal patterns from the statistical properties of the observed data. In this paper we further divide the non-parametric into two sub-groups; the on-line and the batch approach. The batch approach trains and detects normal and abnormal activities using complete trajectories. The on-line approach may or may not train the system using complete trajectories, yet it is able to detect normal/abnormal activities using incomplete trajectories; hence the ability to detect abnormalities as they happen. The centroids of the tracked objects in Section IV-B are used as input to the trajectory classifier. Generally, the trajectory data of tracked objects are recorded as a set of (x, y) locations of the tracked object's centre of mass from frame to frame. In [29], they used flow vectors $f = \{x, y, \delta x, \delta y\}$ rather than sequence of positions to describe an object's movement.

Thus if an object i appears in n frames it can be represented by a set Q_i of n flow vectors all lying within a unit hypercube in 4D phase space: $Q_i = \{f_1, f_2, \ldots, f_{n-1}, f_n\}$. Owens *et al.* in [27], used a hierarchical neural network as a novelty detector. Normal trajectories are used during training, and the experiments conducted show a high detection rate. Humphreys *et al.* [30] has extensively use cost functions based on a Self Organising Feature Map (SOFM) to detect, track and classify object trajectories. The paper also demonstrates improved performance by using three SOFMs dedicated to different sub cost functions.

To efficiently implement a trajectory discriminator on a low powered processor using SOFM and Gaussian distribution, we have conducted two basic analyses. First, we analyse the minimal dimension that can be used to represent the point-to-point trajectory data (x_t, y_t) without losing any behavioral information. Intuitively, the minimum dimension is 2D, yet in [31] the (x_t, y_t) coordinate information has been reduced to a single value $\dot{\gamma_t}$ encoding the local curvature and velocity information. The penalty for the model is the high dimensional vector used in the HMM. Secondly, we analyse the most efficient way to represent the trajectory data in the SOFM. By reducing the dimension of the trajectory data we have been able to implement the SOFM based classifier on the PXA270 running at a reasonable speed.

Similar to [32], our system monitors trajectories as they are generated, in contrsat to other systems [28], [33] which need the entire trajectory to make a decision. Hence the trajectory encoding used here converts both full and sub trajectories into a fixed length feature vector $F = (x, y, s\delta x, s\delta y)$, where $s\delta x$ and $s\delta y$ are the moving averages for the change in x and y respectively. As the feature vector generated for each individual point is of fixed length, a SOFM has been used for classification.

We have designed our SOFM with 100 network nodes, each with four weights representing the 4-input feature vector $(x, y, \delta x, \delta y)$. During training, we maintain four extra parameters for each node in the network: the total number of training samples that get associated with each node T_i, the maximum distance between the node and all associated inputs, M_i, the mean μ_i and variance σ_i^2 of the distances. A Gaussian distribution of all distances associated with every node is also maintained.

The training data is made up of both normal and abnormal trajectories, unlabelled, yet our implementation is able to distinguish between normal and abnormal trajectories after training. Trajectory data (x, y) is collected over a period of time from a stationary camera and converted into a 4D feature vector F for training the SOFM. During training, the 100 network nodes are randomly initialized, then for every input vector (feature vector), the Manhattan distance between the input vector and every network node is computed to estimate the winner. For a winner w_t and input vector x, all the weights i of the winning node are updated as follows $w_{i,t+1} = w_{i,t} + \beta(x - w_{i,t})$ to reflect the input data. If the Manhattan distance $m_{w,x}$ between w_t and x is the maximum for node $w_t, M_w = m_{w,x}$. Similarly, the total distance for the winner T_w is increased by one.

The training of the SOFM is repeated for a number of epochs with the same input data. The Gaussian distribution for each node is generated for a random iteration $t \leq (epoch - 1)$ during training. The network is ready for use after the training phase. During the test phase, point-to-point trajectory data (x, y) is converted into a 4D vector and used as input to the SOFM. The winning node is identified as the node with the minimum Manhattan distance to the input vector. In the test phase the network is not subject to any further modification, but rather is used to make a decision on the input vector or trajectory.

D. Alerts

The ABORAT system tests for alerts for every image frame. There are three different alerts generated: the object type, zone and tripwire violation and the behavioural alert.

1) Object Type: This is used to identify Humans, Vehicles, group of people and any other object. Humans are easily distinguished from vehicles using the aspect ratio. The camera view can make it difficult to distinguish between other objects and humans. Thus a bird very close to the camera might have the same aspect ratio as a person walking at a distance from the camera. A position-wise aspect ratio has been used to resolve such ambiguity. Groups of people may also have the same aspect ratio as a vehicle. Shape variation is used to distinguish between vehicles and groups of people. After object identification, the object type is sent to the monitoring station if there is a suspicious behaviour in the scene or a tripwire rule is violated.

2) Tripwire and Zones: This is used to identify objects going over a virtual line (e.g. level crossing) or entering or leaving restricted areas. The centroid (rather than the entire body) of moving objects is used to determine their position.

A line is defined by the two end points $\mathbf{a}$ and $\mathbf{b}$. If the line extends across the entire viewing area then a line-crossing can simply be detected by a change in the sign of the scalar product of the line's orthogonal vector with the moving object, because if $\mathbf{n}$ is orthogonal to $\mathbf{b} - \mathbf{a}$ then $(\mathbf{c} - \mathbf{a}) \cdot \mathbf{n} > 0$ for all points $\mathbf{c}$ on the side of the line where $\mathbf{n}$ points to, < 0 for all points on the opposite site, and $= 0$ for all points on the line. If the line is shorter and only covers part of the viewing area we need to test whether it intersects with the line drawn by two consecutive points (centroids) of the object tracked. Two lines $\mathbf{a} \rightarrow \mathbf{b}$ and $\mathbf{c} \rightarrow \mathbf{d}$ intersect if we can solve

$$\mathbf{a} + \alpha(\mathbf{b} - \mathbf{a}) = \mathbf{c} + \beta(\mathbf{d} - \mathbf{c}) \qquad (6)$$

for α and β and find $0 \leq \alpha, \beta \leq 1$.

For a restricted area specified by its endpoints $\mathbf{p_1}, \ldots, \mathbf{p_k}$, the procedure for testing whether a point $\mathbf{c}$ is inside or outside is similar. We need to specify any arbitrary line starting at $\mathbf{c}$ and going into one direction towards infinity. If the number of lines it crosses out of $\mathbf{p_1} \rightarrow \mathbf{p_2}, \ldots, \mathbf{p_{k-1}} \rightarrow \mathbf{p_k}, \mathbf{p_k} \rightarrow \mathbf{p_1}$ is even then $\mathbf{c}$ is outside the restricted area and inside if the number is odd.

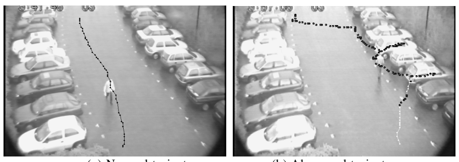


Fig. 7. Images showing (a) normal and (b) abnormal trajectories. In (b), abnormal points are labelled black.

3) Behavioural Detection: The system is fully autonomous and capable of using the trajectory of moving object to classify the behaviour. An input trajectory data for tracked objects is identified as abnormal (suspicious) if any of the following conditions is true:

1) If the Manhattan distance $m_{w,x}$ between the input vector x and the winner w is greater than the maximum allowable distance for the winner M_w.
2) If T_w (the total number of input vectors associated with the winner during training) is less than a gobal threshold Th set as $0.01\% * total\ train\ points$.
3) If the Manhattan distance $m_{w,x}$ is outside 2.5 standard deviation of the Gaussian distribution for the winner.

A score ranking is used to generate alerts for different violations. The penalty for option 1 is the highest, followed by options 2 and 3 respectively. An input node whose Manhattan distance is greater than M_w is abnormal on the assumption that such a point is new to the SOFM. Since the system is trained with both normal and abnormal trajectories, it is possible for a node in the network to represent only abnormal trajectory points. Since unusual trajectories are rare, an assumption that no more than $Th = 0.01\%$ of the entire trajectory points are abnormal is made. Hence, any network node with less than the global threshold value Th of points, is labelled as an abnormal network node n_{ab}. Thus any input vector whose winner is $n_a b$ is also considered abnormal.

It is also possible to associate an abnormal point to a normal network node n_{nor} during training. If this happens, we expect the Manhattan distance between the abnormal point x_{ab} and the network node n to be much greater than all other points associated with n_{nor}. The Gaussian distribution maintained for n_{nor} is then used to identify such abnormal trajectory points. Figure 7 shows two images with normal and abnormal trajectory points. If the trajectory of a moving object is classified as abnormal, the level of abnormality as well as the object type is sent to the monitoring station.

V. Experimental Results

To evaluate the training time of the autonomous trajectory classifier, three different datasets have been used in testing the implementation on a PC with a general purpose processor clocked at 2.8GHz, and on the ABORAT VPU with PXA270 clocked at 520MHz. All the images have been obtained using a stationary camera. The input image is sent to an object tracker and the trajectory fed into the SOFM for training. Two of the image sequences have been acquired on a normal day while the last of the three has been collected on a rainy day. They have all been collected over a period of 3 hours. The datasets are made of 34713 and 21867 trajectory points taken from various trajectories for the normal day, and 12636 trajectory points for the rainy day. Table I presents a summary of the test conducted on the PXA270 and PC with the same input data and epoch.

TABLE I
Timing results for training the SOFM on PXA270 and PC

Day	Points	PC(min.)	PXA270(min.)	epoch
Normal	34713	45	190	346
Normal	21867	27	110	218
Rainy	12636	10	65	126

TABLE II
Timing results of the algorithm using YUV background differencing on PXA270 and PC

Application Type	Processing time (ms)	
	320x240	640x480
PC	$\sim$ 40	$\sim$ 175
PX_{BMP}	$\sim$ 95	$\sim$ 351
PX_{IP}	-	$550 - 750$

A test has also been conducted on the number of trajectory points correctly classified with the implementation. For 520 trajectory points collected on a normal day, 421 were correctly classified as normal, 76 correctly classified as abnormal and 23 were incorrectly classified as normal, representing approximately 4.4% error. A similar test conducted on the same scene, on a rainy day with a total of 151 trajectory points gave 97 correctly classified as normal, 32 correctly classified as abnormal, 19 incorrectly classified as normal with 3 classified incorrectly as abnormal. This represents a total of 14.5% error.

Timing analysis has also been conducted on various components with different frame sizes on both the PC and the PXA270. Again, the test is conducted using Bitmap images on the PC and on the PXA270 (PX_{BMP}). Timing analysis with IP data has also been conducted on the PXA270 (PX_{IP}). Table II is a summary of the processing time for image sizes of 320×240 and 640×480 using the YUV background differencing algorithm on both the PC and PXA270.

Using the grey-scale intensity value to generate the background, the processing time reduces from $750ms$ to $350ms$, making it possible to process 3 frames of VGA sized image every second. Processing 3 frames per second is enough for tracking pedestrians in real-time. The PXA270 is capable of processing 10 frames per second of a QVGA image size, which is sufficient resolution for most surveillance applications. The total processing time for the PXA270 and a PC implementation of the entire ABORAT algorithm using the grey-scale background modelling is given in table III

VI. Conclusion

This paper demonstrates a distributed smart camera architecture using an IP camera as an image sensor, a low power processor as an image processor, a neural network

TABLE III
TIMING RESULTS OF THE ALGORITHM USING GREY-SCALE INTENSITY BACKGROUND DIFFERENCING ON PXA270 AND PC

Application Type	Processing time (ms)	
	320x240	640x480
PC	~ 40	~ 175
PX_{BMP}	~ 95	~ 351
PX_{IP}	-	$290 - 350$

(SOFM) as an autonomous trajectory classifier and a general purpose PC as a monitoring unit. Compared to a CCTV camera which can only view a local area, an IP camera can view over a wider geographical area. An object tracker developed purposely for an embedded platform without the use of floating point numbers has also been presented. The on-line classifier based on the point-to-point trajectory of moving objects makes this architecture more usable for today's embedded security surveillance systems. The novel system presented here is autonomous and does not require any human intervention before or after training. Hence a camera is deployed, autonomously collects data over a period of time, trains the SOFM and detects suspicious behaviour after training. A possible extension is to incorporate the VPU into a single smart camera unit. The tradeoff for future applications between enhancing the intelligence and performance of the VPU or the client PC (or both) using software/hardware architecture like Intel's Quick Assist [34] will depend upon silicon cost, the complexity of the video analytics required, and the bandwidth constraints of the reader's network.

ACKNOWLEDGMENT

The authors would like to give special thanks to the following ABORAT team members: Eastern Kentucky University's Justice & Safety Center who generously financed part of this project with 3 grants from the U.S. Department of Homeland Security; Shawn von Benzon; Dennis O'Brien and John Johnston.

REFERENCES

[1] M. Shah, O. Javed, and K. Shafique, "Automated visual surveillance in realistic scenarios," vol. 14, no. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 30–39.

[2] A. J. Lipton, "Keynote: intelligent video as a force multiplier for crime detection and prevention." The IEE International Symposium on Imaging for Crime Detection and Prevention, 2005, pp. 151–156.

[3] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl, "Autonomous multicamera tracking on embedded smart cameras," *EURASIP Journal on Embedded Systems*, vol. 2007. [Online]. Available: http://www.hindawi.com/GetArticle.aspx?doi=10.1155/2007/92827

[4] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti, "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking." *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 38 – 51, March 2005.

[5] ActiveEye, "Active alert," 2005. [Online]. Available: http://www.activeye.com/press_rel_forensics.htm

[6] Cernium, "Perceptrak," 2009. [Online]. Available: http://www.cernium.com/perceptrak.asp

[7] G. Security, "Video iq," 2007. [Online]. Available: http://www.gesecurity.com

[8] G. Solutions, "Fencewatch," 2007. [Online]. Available: http://www.guardiansolutions.com

[9] Verint, "Nextiva integrated video analytics," 2007. [Online]. Available: http://www.verint.com

[10] Vidient, "Smartcatch," 2003. [Online]. Available: http://www.vidient.com

[11] L-3Communications, "Visionalert," 2007. [Online]. Available: http://www.l-3com.com

[12] iOmniscient, "Smart iq," 2007. [Online]. Available: http://www.iOmniscient.com

[13] A. Bigdele, B. C. Lovel, and T. Shan, "Smart cameras: Enabling technology for proactive intelligent cctv." The Research Network Secure Australia Security Technology Conference, 2006, pp. 151–156.

[14] ObjectVideo, "Objectvideo vew and objectvideo onboard," 2003. [Online]. Available: http://www.objectvideo.com

[15] Roke, Manor, Research, and Limited, "Video motion anomaly detection," 2009. [Online]. Available: http://www.roke.co.uk

[16] Mate-Media, Access, and Technologies, "Trigger," 2007. [Online]. Available: http://www.mate.co.il

[17] Axis, "Axis and intelligent video (iv)," Axis Communications, 2007.

[18] Y. Charfi, N. Wakamiya, and M. Murata, "Network-adaptive image and video transmission in camera-based wireless sensor networks." First ACM/IEEE international conference on distributed smart cameras, 2007, pp. 336–343.

[19] T. Pham, M. Worring, and A. Smeulders, "A multi-camera visual surveillance system for tracking of reoccurrence of people." First ACM/IEEE international conference on distributed smart cameras, 2007, pp. 164–169.

[20] M. Litzenberger, A. Belbachir, P. Schon, and C. Posch, "Embedded smart camera for high speed vision." First ACM/IEEE international conference on distributed smart cameras, 2007, pp. 81–86.

[21] M. Daniels, K. Muldawer, J. Schlessman, B. Ozer, and W. Wolf, "Real-time human motion detection with distributed smart cameras." First ACM/IEEE international conference on distributed smart cameras, 2007, pp. 187–194.

[22] P. Jodoin, M. Mignotte, and J. Konrad, "Statistical background subtraction using spatial cues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 12, pp. 1758–1763, December 2007.

[23] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 246–252.

[24] K. Appiah and A. Hunter, "A single-chip fpga implementation of real-time adaptive background model," *IEEE International Conference on Field-Programmable Technology*, pp. 95–102, December 2005.

[25] A. Makarov, "Comparison of background extraction based intrusion detection algorithms." IEEE Int. Conference on Image Processing ICIP96, 1996, p. section 16P3.

[26] J. Owens, A. Hunter, and E. Fletcher, "A fast model-free morphology-based object tracking algorithm," in *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2002.

[27] ——, "Novelty detection in video surveillance using hierarchical neural networks," *International Conference on Artificial Neural Networks (ICANN 2002).*

[28] Y. Zhou, Y. Yan, and S. Huang, "Detecting anomaly in videos from trajectory similarity analysis," *IEEE International Conference on Multimedia and Expo*, 2007.

[29] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," *Proceedings of the 6th British Conference on Machine Vision*, vol. 2, pp. 583–592, 1995.

[30] J. Humphreys and A. Hunter, "Multiple object tracking using a neural cost function," *Image and Vision Computing, June 2008.*

[31] A. Hervieu, P. Bouthemy, and J.-P. L. Cadre., "A statistical video content recognition method using invariant features on object trajectories," *IEEE Trans. on CSVT (Special Issue on 'Event Analysis in Videos')*, 2008.

[32] J. Owens and A. Hunter, "Application of the self-organizing map to trajectory classification," in *VS '00: Proceedings of the Third IEEE International Workshop on Visual Surveillance (VS'2000).* Washington, DC, USA: IEEE Computer Society, 2000, p. 77.

[33] F. Jiang, Y. Wu, and K. A. Katsaggelos, "Abnormal event detection from surveillance video by dynamic hierarchical clustering." *in Proc. IEEE Int'l Conf. on Image Processing (ICIP'07), San Antonio, TX*, Sept. 2007.

[34] Intel, "Quickassist technology," 2008. [Online]. Available: http://developer.intel.com/technology/platforms/quickassist/index.htm