

# Improved Anomaly Detection in Crowded Scenes via Cell-based Analysis of Foreground Speed, Size and Texture

Vikas Reddy, Conrad Sanderson, Brian C. Lovell

NICTA, PO Box 6020, St Lucia, QLD 4067, Australia  
The University of Queensland, School of ITEE, QLD 4072, Australia

## Abstract

*A robust and efficient anomaly detection technique is proposed, capable of dealing with crowded scenes where traditional tracking based approaches tend to fail. Initial foreground segmentation of the input frames confines the analysis to foreground objects and effectively ignores irrelevant background dynamics. Input frames are split into non-overlapping cells, followed by extracting features based on motion, size and texture from each cell. Each feature type is independently analysed for the presence of an anomaly. Unlike most methods, a refined estimate of object motion is achieved by computing the optical flow of only the foreground pixels. The motion and size features are modelled by an approximated version of kernel density estimation, which is computationally efficient even for large training datasets. Texture features are modelled by an adaptively grown codebook, with the number of entries in the codebook selected in an online fashion. Experiments on the recently published UCSD Anomaly Detection dataset show that the proposed method obtains considerably better results than three recent approaches: MPPCA, social force, and mixture of dynamic textures (MDT). The proposed method is also several orders of magnitude faster than MDT, the next best performing method.*

## 1. Introduction

Automated detection of anomalous events in video feeds has the potential to provide more vigilant surveillance, possibly in lieu of, or as an assistance to, human operators who have limited attention spans when faced with tedious tasks [13]. Qualifying an event as anomalous is subjective and depends on the intended application as well as context. However, without being application or context specific, an anomalous event can be defined as *any event that is different from what has been observed beforehand*.

Detection of anomalous events can be hence viewed as a binary classification problem, where there are training examples only for one class, generally. Typical algorithms model the dynamics of ‘normal’ activity or ‘expected’ behaviour and compare new observations with an existing model. Any outliers are labelled as anomalous. An ideal

system is expected not only to detect anomalous events accurately, but also to adapt itself to the changes witnessed in the environment over time.

Several anomaly detection techniques have been proposed in various research fields. Chandola et al. [10] discuss them in detail in their survey, while Saligrama et al. [26] examine video based anomaly detections approaches in the context of surveillance. Existing methods in the literature can be roughly placed into two categories: **(i)** analysis by tracking, where trajectories of individual objects are maintained; **(ii)** analysis without tracking, where other features such as motion and texture are employed to model activity patterns of a given scene.

In the first category, almost all approaches use tracking information directly to gather object speed and direction, and indirectly as an aid in determining features such as the size and aspect ratio of objects [4, 14, 21, 24, 31]. While trajectory based approaches are suitable for cases where the scene is comprised of only a few objects, in crowded environments it is difficult to reliably maintain tracks due to occlusion and overlap of objects [16, 19].

In light of the above problems, in the second category the anomaly detection task is formulated while deliberately omitting the tracking of specific objects. Most approaches in this category largely rely on motion or motion-related features. For example, Mehran et al. [20] model crowd behaviour using a “social force” model, where the interaction forces are computed using optical flow. Adam et al. [1] model optical flow at a set of fixed spatial locations using probabilistic histograms. Ermis et al. [11] propose using busy-idle rates of each pixel to detect abnormal behaviour.

As the above techniques solely rely on motion information, anomalies occurring due to object size or appearance may not be detected. To address this limitation, Mahadevan et al. [19] recently proposed to jointly model the appearance and dynamics of crowded scenes, using mixtures of dynamic textures (MDT) [8]. The method explicitly investigates both temporal and spatial anomalies. Though the reported comparative results show improvements over earlier techniques, the method’s main drawback is heavy computation. Evaluating a frame of size  $240 \times 160$  takes about 25 seconds (ie. 2.4 frames per minute).

In this paper, we present a robust anomaly detection algorithm with relatively low complexity, targeted primarily for crowded scenes where traditional tracking based approaches tend to fail. To suppress undesirable background dynamics, such as waving trees and illumination variations, we perform foreground segmentation and retain only foreground objects for further analysis. Each input frame is split into non-overlapping cells (small regions). Based on each frame’s foreground mask, the relevant cells are analysed for the presence of an anomaly. Unlike most methods, a refined estimate of motion is achieved by computing the optical flow only for the foreground pixels. In addition to motion, the proposed method analyses the size and texture of foreground objects at each cell location.

Motion, size and texture are modelled separately. Independent analysis helps to keep computation efficient, and allows for inferring the nature of the anomaly (eg. speed violation, lack of motion, size too large, etc). Each cell is labelled as either normal or anomalous after combining the outputs of multiple classifiers (one for each feature type).

We continue the paper as follows. In Section 2 the proposed algorithm is described in detail. Performance evaluation and comparison with three recent algorithms is given in Section 3. The main findings and possible future directions are presented in Section 4.

## 2. Proposed Algorithm

The proposed method has four main components:

1. Feature extraction, where input images are split into non-overlapping spatial regions, termed *cells*, and features are extracted based on motion, size and texture of the foreground objects contained in the cells.
2. Model estimation, which models the normal dynamics witnessed at each cell location. There are separate models for each feature type.
3. Classification of each cell as anomalous or normal, where each cell is sequentially checked for normality by up to two classifiers. As soon as the first classifier deems that the cell is anomalous, the second classifier is not consulted. In order of processing, the two classifiers are:
  - (a) Speed check, where the likelihood of the magnitude of motion of any foreground objects is evaluated.
  - (b) Size and texture check, where first the likelihood of the size of a foreground object is evaluated. Cells with low likelihoods (suggesting an anomaly) are further analysed according to their texture, in order to validate the presence of the anomaly.
4. Spatio-temporal post-processing, to minimise isolated random noise present in the generated anomaly masks.

Each of the components is explained in detail in the following sections.

### 2.1. Feature Extraction

Let the resolution of the greyscale image sequence  $\mathcal{I}$  be  $\mathcal{W} \times \mathcal{H}$ . Each image is split into non-overlapping cells (regions) of size  $N \times N$ , with the cell located at  $i$  and  $j$  denoted by  $\mathcal{C}(i, j)$ . The cell co-ordinates have the range of  $i = 1, 2, \dots, (\mathcal{W}/N)$  and  $j = 1, 2, \dots, (\mathcal{H}/N)$ . Let  $\mathcal{I}_t$  be the frame at time instant  $t$  and let its corresponding cells be denoted by  $\mathcal{C}_t(i, j)$ .

In order to restrict the analysis to regions of interest and to filter out distractions (eg. waving trees, illumination changes, etc), we perform foreground segmentation on each incoming frame. We have used the method proposed in [23], due to its robustness, high-quality foreground masks and the ability to estimate the background even in the presence of multiple moving foreground objects. Alternative techniques for estimating the background in crowded scenes include [3, 22].

For each cell, we extract features based on motion, size and texture. The foreground masks are referenced while computing the features. The details of the three features are given below.

#### 2.1.1 Motion

To estimate the motion associated with cell  $\mathcal{C}_t(i, j)$ , we compute the optical flow of only the foreground pixels. The iterative Lucas-Kanade algorithm [7, 18] is employed to compute the displacement of pixels between two consecutive frames, with a fixed search window around each pixel. We first calculate the average motion associated with cell  $\mathcal{C}_t(i, j)$  using:

$$\widehat{\text{mot}}_t(i, j) = \frac{1}{N_f} \sum_{n=0}^{N_f} \left\| \left[ v_x^{(n)}, v_y^{(n)} \right] \right\|_1 \quad (1)$$

where, for foreground pixel  $n$ ,  $v_x^{(n)}$  and  $v_y^{(n)}$  are the optical flows in the  $x$  and  $y$  directions, respectively, while  $N_f$  is the total number of foreground pixels within the cell.

The motion feature for cell  $\mathcal{C}_t(i, j)$  is taken to be the smoothed (noise-reduced) version of the cell’s average motion, calculated using straightforward temporal averaging:

$$\text{mot}_t(i, j) = \frac{1}{3} \sum_{u=t-1}^{t+1} \widehat{\text{mot}}_u(i, j) \quad (2)$$

#### 2.1.2 Size

Relying on motion alone for anomaly detection might be insufficient, as certain anomalies may exhibit motion that is considered as normal (eg. a slow moving vehicle on a path designated for pedestrians). Furthermore, motion estimation techniques can suffer from the aperture problem [30]. To increase the sensitivity of anomaly detection, the size of foreground objects can be analysed.

A common technique to measure object size is via connected component analysis on the foreground masks. However, in crowded environments it becomes ineffective due

to object overlap and occlusion. Instead, an approximate size of an object contained within a cell can be obtained by considering its foreground occupation along with that of its neighbouring cells (as the object may occupy more than one cell). An example is shown in Fig. 1.

Specifically, let us denote the foreground occupancy (number of foreground pixels) for cell  $C_t(i, j)$  by  $o_t(i, j)$ . We define the size feature for cell  $C_t(i, j)$  as a weighted combination of the foreground occupancy values of the cell and its immediate neighbours:

$$\text{size}_t(i, j) = \sum_{a=i-1}^{i+1} \sum_{b=j-1}^{j+1} G(a-i+1, b-j+1) o_t(a, b) \quad (3)$$

where  $G$  is a  $3 \times 3$  Gaussian mask [12]. The mask is used for placing prominence on the center cell and hence reducing the impact of neighbouring cells that, in crowded scenarios, may contain foreground pixels belonging to other objects (in addition to the object of interest).

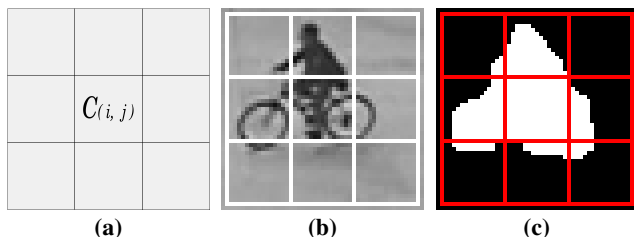
### 2.1.3 Texture

While the size feature can be useful for increasing the sensitivity of anomaly detection, using it without qualification may also increase the false alarm rate. For example, in crowded environments the foreground masks of people walking close to each other could resemble a large foreground object.

To address this problem, the texture present within the cell can be used for increasing selectivity. To this end, we filter a given image using 2D Gabor wavelets [17] at four orientations: 0, 45, 90 and 135 degrees. The texture descriptor for cell  $C_t(i, j)$  is hence a 4D vector:

$$\text{txt}_t(i, j) = [m_0 \ m_{45} \ m_{90} \ m_{135}]' \quad (4)$$

where  $m_\theta$  is the sum of the response magnitudes of the wavelet oriented at  $\theta$  degrees, over the pixels contained within the cell. The texture vectors are only collected for cells that have at least one foreground pixel, in order to minimise modelling of the background.



**Figure 1.** (a) the approximate size of the foreground object contained in the center cell  $C(i, j)$  is computed by considering foreground pixels in the cell as well as its neighbouring cells; (b) an example of a foreground object appearing in the center cell and its neighbours; (c) the corresponding foreground mask, found via an automatic foreground segmentation algorithm.

## 2.2. Scalable Semi-Parametric Model Estimation

Surveillance scenarios include platforms at train/bus stations, buildings (both indoors/outdoors) as well as road/walkway traffic. In all these scenarios, even normal day-to-day events have inherent variations that are random in nature. For example, the speed of vehicles on a road can vary arbitrarily due to traffic light signals and congestion. Furthermore, the dynamics of the scene can keep changing over time. As such, parametric approaches are unlikely to be effective for modelling distributions of features in these scenarios, as the number of modes is unlikely to be reliably known beforehand. With this in mind, we employed a semi-parametric modelling approach, which can be used for modelling arbitrary distributions without using any assumptions on the forms of the underlying densities.

We model each cell by considering only the features extracted from that particular cell location, along the temporal axis. As described in Section 2.1, there are three feature types: motion (a scalar), size (also a scalar) and a texture descriptor (a 4D vector). In tasks such as object detection/recognition, it is often argued that joint modelling of features yields better results than modelling them independently [2, 28]. However, in our context such an approach may fail to detect outliers (ie. the anomalies) accurately, due to the implicit mutual influence exhibited by the features in the decision process. Furthermore, the dynamics of a crowded scene are inherently arbitrary in nature, which may render joint modelling ineffective. We will hence model these features separately. Independent analysis keeps computation efficient, examines each feature precisely for anomalies, which in turn allows for inferring the nature of the anomaly (eg. speed violation, lack of motion, size too large, etc).

For modelling the motion and size features, a straightforward and efficient semi-parametric approach involves constructing normalised histograms. The training data can be discarded once the histograms are built. However, a major problem with this approach is the presence of sharp discontinuities in the estimated densities due to binning, rather than that of the underlying distribution that generated the data [6]. To overcome the above limitation, it is possible to use kernel density estimation techniques that result in smoother probability density functions [6, 25]. Their training phase only involves storing of all the data samples. However, their drawback is increased computational cost and memory requirements as the dataset becomes larger [6]. As such, these techniques can suffer from scalability issues.

To achieve a better trade-off between accuracy and computational requirements, we create a smoothed histogram by temporarily storing all the training samples and performing Gaussian kernel based density estimation to compute the probability of the continuous variable (ie. motion or size) only at discrete points over its entire permissible range.

In effect, we assume the random variable to be discrete and compute its probability at fixed points using:

$$f(s\Delta x) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h\sqrt{2\pi}} \exp \left\{ -\frac{\|s\Delta x - x_n\|^2}{2h^2} \right\} \quad (5)$$

where  $\Delta x$  is the resolution of the step size (eg. 0.25),  $s = \{0, 1, 2, 3, \dots, S\}$ , with  $S\Delta x$  being the valid upper limit of the variable in consideration.  $N$  is the number of samples in the training dataset and  $h$  is the bandwidth of the Gaussian kernel. The probability values are normalised to obtain a probability mass function (pmf). As in the histogram approach, the training data is discarded once the pmf is computed. The resultant pmf is denoted by  $\hat{p}(\cdot)$ .

### 2.2.1 Adaptive Modelling of Texture Descriptors

While the above approach is effective for modelling the distribution of scalars (motion and size in our case), using it to model the distribution of the 4D texture descriptors is impractical, as the number of resulting discrete samples required to cover the entire feature space (for just one cell location) would be quite large. For example, having only 20 equally spaced points in each dimension would generate  $20^4$  points in 4D space. As there is a non-trivial number of cell locations, the total storage costs would be hence prohibitive.

Furthermore, the above density estimation approach implicitly relies on an Euclidean based distance, which will be affected by variations in texture contrasts, rather than purely measuring the differences between texture patterns. For example, the texture descriptor will exhibit low magnitude responses when the intensity of a pedestrian's clothing is similar to that of the background, and high magnitude responses when the intensity is contrasting to the background.

To address the storage problem, for each cell location we model the distribution of the texture descriptors using a codebook that is trained in an online fashion (adaptively grown), inspired by [16]. To address the distance measure problem, we employ Pearson's correlation coefficient [29] for measuring the similarity of two descriptors:

$$\rho(\mathbf{a}, \mathbf{b}) = \frac{(\mathbf{a} - \mu_{\mathbf{a}})'(\mathbf{b} - \mu_{\mathbf{b}})}{\|\mathbf{a} - \mu_{\mathbf{a}}\| \|\mathbf{b} - \mu_{\mathbf{b}}\|} \quad (6)$$

where  $\mu_{\mathbf{x}}$  is the mean of the elements of vector  $\mathbf{x}$ , and  $\rho(\mathbf{a}, \mathbf{b}) \in [-1, +1]$ .

The codebook is built as follows. Initially, the first training vector is taken to be the first entry in the codebook. Each of the remaining vectors is sequentially treated as a new observation,  $\mathbf{x}$ , which is compared to each entry in the codebook,  $\mathbf{c}_k$ , using Eqn. (6). If, for the best matching  $\mathbf{c}_k$ ,  $\rho(\mathbf{x}, \mathbf{c}_k) > 0.9$ , the  $k$ -th entry is updated using [9]:

$$\mathbf{c}_k^{\text{new}} = \mathbf{c}_k^{\text{old}} + \frac{1}{W_k + 1} (\mathbf{x} - \mathbf{c}_k^{\text{old}}) \quad (7)$$

where  $W_k$  is the number of texture vectors associated so far with entry  $k$ . If  $\forall k \rho(\mathbf{x}, \mathbf{c}_k) \leq 0.9$ , vector  $\mathbf{x}$  is appended to the codebook, thereby expanding it.

## 2.3. Cell Classification

Each cell is sequentially checked whether it is anomalous by up to two classifiers. As soon as the first classifier deems that the cell is anomalous, the second classifier is not consulted. Specifically, given a decision threshold  $T$ , cell  $C_t(i, j)$  is classified as anomalous if either of the following two conditions are satisfied:

- (a)  $\hat{p}_{\text{mot}}(\text{mot}_t(i, j)) < T$
- (b)  $\hat{p}_{\text{size}}(\text{size}_t(i, j)) < T$  and  $\rho_{\text{max}}(\text{txt}_t(i, j)) < 0.9$

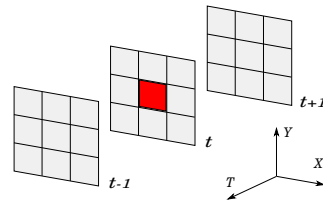
where  $\hat{p}_{\text{mot}}(\cdot)$  and  $\hat{p}_{\text{size}}(\cdot)$  are the pmfs calculated in Section 2.2, while  $\rho_{\text{max}}(\text{txt}_t(i, j)) = \max_k \rho(\text{txt}_t(i, j), \mathbf{c}_k)$ , i.e. the correlation coefficient of the closest matching codebook entry.

Condition (a) is effectively a speed check, where speeds that are either slower or faster than 'normal' are detected (note that  $\hat{p}_{\text{mot}}(\cdot)$  can define several 'normal' speeds). In condition (b) both the size and texture are checked. As the size feature alone is not be able to distinguish between a large object and a collection of small objects (eg. a crowd of people), the texture feature is in effect used to verify the presence of an anomaly indicated by the size feature.

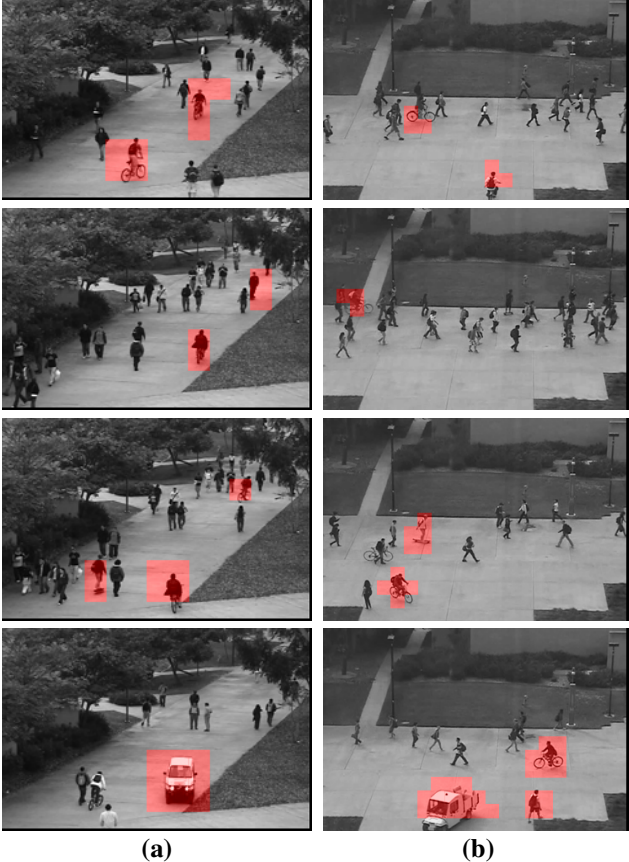
The texture feature is only calculated for cells that contain foreground pixels, in order to avoid modelling the background (which might be dynamic). As such, the texture feature is suitable for distinguishing among foreground objects. However, as the feature may end up capturing irrelevant textures when the cell contains only thin edges of the foreground, it is used in combination with the size feature rather than being used alone.

## 2.4. Spatio-Temporal Post-Processing

To minimise spurious and intermittent false alarms, spatio-temporal post-processing is performed on the anomaly masks generated by the classification procedure in Section 2.3. If a cell at time  $t$  was initially classified as anomalous, we consider its immediate neighbours along both the spatial and temporal axes (see Fig. 2). If at least two cells in each plane (ie.  $t-1, t, t+1$ ) were classified as anomalous, we assume that the cell in question was correctly classified as anomalous. Otherwise, it is re-classified as being normal (ie. non-anomalous).



**Figure 2.** For a cell initially classified as anomalous (marked in red), its immediate neighbours along both the spatial and temporal axes are consulted to verify whether the cell was classified as anomalous due to noise.



**Figure 3.** Examples of anomaly detection and localisation via the proposed method (highlighted in red). Results are shown on the (a) Ped1 and (b) Ped2 subsets of the UCSD Anomaly dataset.

### 3. Experiments

To appraise the performance of the proposed approach, we performed experiments on the recently released UCSD Anomaly Detection dataset [19]. The dataset contains multiple surveillance videos of two scenes (Ped1 and Ped2), both with considerable crowds. Anomalies present in the dataset include: skateboarders, bikers, motor vehicles, people pushing carts as well as walking on the lawn. The image size in Ped1 is  $238 \times 158$  pixels, while on Ped2 it is  $360 \times 240$ . Ped1 has 34 training and 36 test image sequences, while Ped2 has 16 training and 12 test image sequences. Examples are shown in Fig. 3.

The UCSD dataset has a prescribed evaluation protocol [19], involving two types of evaluations: (i) frame-level anomaly detection, and (ii) within-frame anomaly localisation. For frame-level anomaly detection, all test sequences have annotated groundtruth at frame-level in the form of a binary flag indicating the presence or absence of anomaly in each frame. For within-frame anomaly localisation, a subset of test sequences (10 in Ped1 and 9 in Ped2) has the anomalous regions within each frame marked. If at least 40% of detected pixels (belonging to a detected anomaly) match the

Approach	Social Force [20]	MPPCA [15]	MDT [19]	Proposed method
<b>Ped1</b>	31%	40%	25%	<b>22.5%</b>
<b>Ped2</b>	42%	30%	25%	<b>20.0%</b>
<b>Average</b>	37%	35%	25%	<b>21.25%</b>

**Table 1.** Equal error rates (EERs) for *frame-level* anomaly detection, obtained on the Ped1 and Ped2 subsets on the UCSD dataset.

Approach	Social Force [20]	MPPCA [15]	MDT [19]	Proposed method
<b>Ped1</b>	79%	82%	55%	<b>32.0%</b>

**Table 2.** EERs for *pixel-level* anomaly localisation.

ground-truth pixels, it is presumed the anomaly has been localised correctly; otherwise it is treated as a ‘miss’.

The proposed algorithm was compared with methods based on social force [20], MPPCA [15], and mixture of dynamic textures (MDT) [19]. The first two methods rely on features obtained from optical flow while the last approach employs features based on appearance and scene dynamics. The quantitative results of the above three algorithms were adapted from [19]. To aid the interpretation of the results, we have reported the false negative rate [5] instead of the true positive rate used in [19].

Based on preliminary experiments, the cell size was set to  $16 \times 16$ , while the search window size in the optical flow computation (Sec. 2.1.1) was set to  $15 \times 15$  (odd-sized to ensure a symmetrical search area around a given pixel). The experiments were implemented with the aid of the Armadillo C++ library [27].

The quantitative results for the frame-level evaluation are shown in Table 1 and in Fig. 4(a)-(b). The results for the within-frame evaluation are shown in Table 2 and in Fig. 4(c). Some of the qualitative results obtained by the proposed method are shown in Fig. 3. In Tables 1 and 2, the equal error rate (EER) is the point where the false negative rate is equal to the false positive rate. At the EER, the proposed method outperforms the other methods at both the frame-level and within-frame evaluations, most notably on the anomaly localisation task.

An experimental implementation of the proposed algorithm in C++ yielded 12 fps (720 frames per minute) on a standard 3 GHz PC, for sequences of images with a size of  $240 \times 160$  (ie. processing the Ped1 subset). We note that this is several orders of magnitude faster than the MDT method, which takes 25 seconds to process each frame (ie. 2.4 frames per minute) [19].

The proposed method has the ability to pick up anomalies (eg. skateboarder, bike) present even at the far end of the scene (eg. 2nd and 3rd images in column (a)). However, the last image in column (a) contains a ‘miss’: the biker was not detected. The cyclist was riding slowly and matching the pace of the neighbouring pedestrian (bottom-left

corner). The texture in this context has strong vertical gradients making the biker appear as a pedestrian. Using a more detailed texture descriptor may help in such cases.

We also note a false positive (a pedestrian being detected as anomaly) in the last image of column (b). Upon further investigation, this false positive was due to the fact that the cells in the region of the pedestrian had minimal or no activity during the training phase. Consequently any foreground object entering this zone during testing was considered as anomalous, irrespective of the observed features.

#### 4. Main Findings and Future Directions

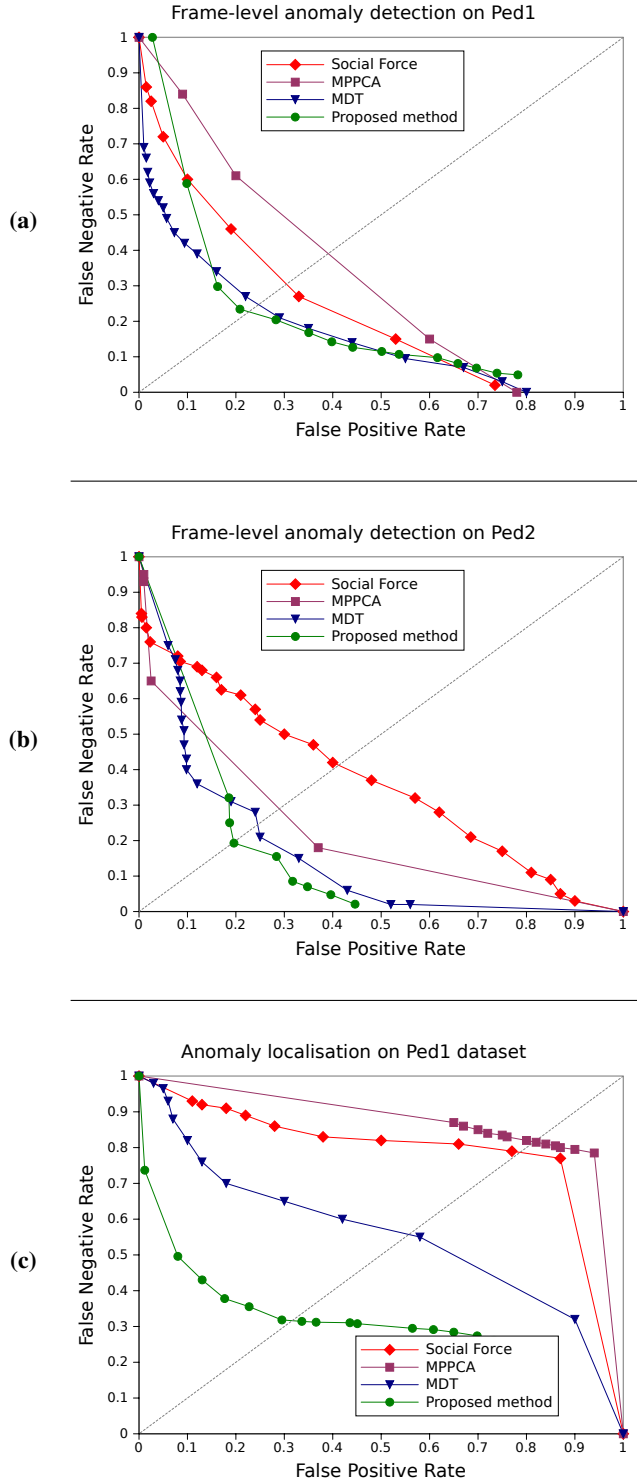
In this paper we have proposed an anomaly detection algorithm targeted towards crowded scenes. In addition to detecting anomalies based on motion, it inspects for anomalies occurring due to size and texture. Video images are initially segmented into foreground regions to confine the analysis to regions of interest, ignoring the background (which might be dynamic). Unlike most methods which compute the optical flow for all pixels or a fixed set of pixels, the flow is computed only for the foreground pixels, thereby achieving a more precise estimate of object motion.

Features based on motion, size and texture are extracted at cell level (small fixed-size regions) and are modelled independently for precise anomaly detection. Motion and size features are modelled by an approximated kernel density estimation technique, which is computationally efficient even on large training datasets. The texture features are represented by an adaptively grown codebook, which is generated in an online fashion.

Experiments on the recently published UCSD Anomaly dataset (containing annotated surveillance videos) show that the proposed method obtains better results than several recent methods: MPPCA, social force, mixture of dynamic textures (MDT). The proposed method attained considerably more accurate anomaly localisation than the next best performing method, MDT, while at the same time being several orders of magnitude faster than MDT.

As part of future work, we aim to investigate the use of more descriptive features such as the orientation of motion [15], which would allow the detection of events such as wrong-way traffic. It would also be useful to adaptively update the models over long periods of time, allowing for context changes (eg. dense traffic might be usual during the day, but it can be unusual at night).

The effect of the cell size should be analysed in the presence of object variations due to factors such as image resolution, perspective changes, as well as view angle. The optimal cell size might be scene dependant and vary across the scene. For example, in Fig 3(a), a larger size might be more appropriate for the bottom-left corner (where objects appear relatively large), while a smaller size might be more effective in the top-right corner (where objects appear relatively small).



**Figure 4.** ROC curves obtained on the UCSD Anomaly Detection dataset, with the bottom-left corner representing ideal performance: (a) frame-level anomaly detection on the Ped1 subset; (b) frame-level anomaly detection on Ped2; (c) within-frame anomaly localisation on Ped1. In all cases, the proposed method outperforms the other approaches at the equal error rate (EER) level, most notably on the anomaly localisation experiment.

## Acknowledgements

NICTA is funded by the Australian Government as represented by the *Department of Broadband, Communications and the Digital Economy*, as well as the Australian Research Council through the *ICT Centre of Excellence* program. We thank Dr Mehrtash Harandi for useful discussions.

## References

- [1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(3):555–560, 2008.
- [2] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *IEEE Intelligent Vehicles Symposium*, pages 255–260, 2005.
- [3] D. Baltieri, R. Vezzani, and R. Cucchiara. Fast background initialization with recursive Hadamard transform. In *Advanced Video and Signal Based Surveillance (AVSS)*, pages 165–171, 2010.
- [4] A. Basharat, A. Gritai, and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
- [5] S. Bengio, J. Mariétoz, and M. Keller. The expected performance curve. In *Int. Conf. Machine Learning (ICML), Workshop on ROC Analysis in Machine Learning*, 2005.
- [6] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [7] J. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm. *Microprocessor Research Labs, Intel Corporation*, 1999.
- [8] A. B. Chan and N. Vasconcelos. Modeling, Clustering, and Segmenting Video with Mixtures of Dynamic Textures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(5):909–926, 2008.
- [9] T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. Technical Report STAN-CS-79-773, Department of Computer Science, Stanford University, 1979.
- [10] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 2009.
- [11] E. Ermiş, V. Saligrama, P. Jodoin, and J. Konrad. Motion segmentation and abnormal behavior detection via behavior clustering. In *Int. Conf. Image Processing (ICIP)*, pages 769–772, 2008.
- [12] R. Gonzalez and R. Woods. *Digital Image Processing*. Pearson Prentice Hall, 3rd edition, 2008.
- [13] N. Haering, P. Venetianer, and A. Lipton. The evolution of video surveillance: an overview. *Machine Vision and Applications*, 19(5):279–290, 2008.
- [14] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, 2006.
- [15] J. Kim and K. Grauman. Observe locally, infer globally: A space-time MRF for detecting abnormal activities with incremental updates. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2928, 2009.
- [16] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1446–1453, 2009.
- [17] T. Lee. Image representation using 2D Gabor wavelets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.
- [18] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. Artificial Intelligence*, volume 3, pages 674–679, 1981.
- [19] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1975–1981, 2010.
- [20] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition (CVPR)*, pages 935–942, 2009.
- [21] C. Piciarelli, C. Micheloni, and G. Foresti. Trajectory-based anomalous event detection. *IEEE Trans. Circuits and Systems for Video Technology*, 18(11):1544–1554, 2008.
- [22] V. Reddy, C. Sanderson, and B. C. Lovell. A low complexity algorithm for static background estimation from cluttered image sequences in surveillance contexts. *EURASIP Journal on Image and Video Processing*, 2011. DOI: 10.1155/2011/164956.
- [23] V. Reddy, C. Sanderson, A. Sanin, and B. C. Lovell. Adaptive patch-based background modelling for improved foreground object segmentation and tracking. In *Advanced Video and Signal Based Surveillance (AVSS)*, pages 172–179, 2010. DOI: 10.1109/AVSS.2010.84.
- [24] P. Remagnino and G. Jones. Classifying Surveillance Events from Attributes and Behaviour. In *British Machine Vision Conference (BMVC)*, 2001.
- [25] I. Saleemi, K. Shafique, and M. Shah. Probabilistic modeling of scene dynamics for applications in visual surveillance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 1472–1485, 2008.
- [26] V. Saligrama, J. Konrad, and P. Jodoin. Video Anomaly Identification. *IEEE Signal Processing Magazine*, 27(5):18–33, 2010.
- [27] C. Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, 2010. <http://arma.sourceforge.net>.
- [28] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV), Lecture Notes in Computer Science (LNCS)*, volume 3953, pages 1–15. Springer, 2006.
- [29] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 3rd edition, 2006.
- [30] E. Trucco and A. Verri. *Introductory Techniques for 3D Computer Vision*. Prentice Hall, 1998.
- [31] X. Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *European Conference on Computer Vision (ECCV), Lecture Notes in Computer Science (LNCS)*, volume 3953, pages 110–123, 2006.