Dublin City University at TRECVID 2008

Peter Wilkins, Philip Kelly, Ciarán Ó Conaire,
Thomas Foures, Alan F. Smeaton, and Noel E. O'Connor
Centre for Digital Video Processing, CLARITY Centre
Dublin City University, Glasnevin, Dublin 9, Ireland
Alan.Smeaton@computing.dcu.ie

Abstract

In this paper we describe our system and experiments performed for both the automatic search task and the event detection task in TRECVid 2008. For the automatic search task for 2008 we submitted 3 runs utilizing only visual retrieval experts, continuing our previous work in examining techniques for query-time weight generation for data-fusion and determining what we can get from global visual only experts. For the event detection task we submitted results for 5 required events (*ElevatorNoEntry*, *OpposingFlow*, *PeopleMeet*, *Embrace* and *PersonRuns*) and 1 optional event (*DoorOpenClose*).

1 Introduction

This year the participation of Dublin City University in TRECVid 2008 contains submissions to both the automatic search task and the event detection task.

Our participation in automatic search is a continuation of our activities from 2007. We concentrated on different approaches to query-time weight generation for data fusion. The retrieval experts utilized were visual only.

For the event detection task we developed our system that contained three basic hierarchical levels, each built on top of each other. These 3 levels were a low-level detector layer, a pedestrian tracking sub-systems and an event detection level. We submitted results for 5 required events (*ElevatorNoEntry*, *OpposingFlow*, *PeopleMeet*, *Embrace* and *PersonRuns*) and 1 optional event (*DoorOpenClose*).

The rest of the paper is organized as follows. The automatic search task is descried in retrieval system itself and the features used in these experiments are described in Section 2, with the results of the proposed approach outlined in Section 2.3. For approach for the event detection task is presented in Section 3.4, where by each of the 3 levels of the system are first outlined. The results for the proposed approach are then outlined in Section 3.4.

2 Automatic Search Activity

Our automatic search experiments continue our exploration of query-time data fusion techniques to determine if it is possible to successfully combine multiple sources of evidence at query time to enhance retrieval without any prior information. This is an extension of our previous work in the area and leverages our previous investigations in the area [7]. To focus our investigation we've restricted our retrieval system to making use only of global low-level visual features. The remainder of this section is organized as follows. We will first discuss the visual features used in retrieval followed with a brief description of our data fusion approaches and conclude with results.

2.1 Visual Features

To facilitate content-based search of query images against shot keyframes our retrieval system made use of six different low-level visual features. Our features are MPEG-7 features and were extracted using the aceToolBox, developed as part of our collaboration in the aceMedia project [1]. We made use of the following features:

- **Scalable Color (SC):** derived from a color histogram defined in the HSV color space which uses a Haar transform coefficient encoding, allowing scalable representation.
- **Color Structure (CS):** based on color histograms, represents an image by both the color distribution (similar to a color histogram) and the local spatial structure of the color.
- **Color Layout (CL):** compact descriptor which captures the spatial layout of the representative colors on a grid superimposed on an image.
- **Color Moments (CM):** similar to Color Layout, this descriptor divides an image into 4x4 subimages and for each subimage the mean and the variance on each LUV color space component is computed.
- **Edge Histogram (EH):** represents the spatial distribution of edges in an image, edges being categorized into either vertical, horizontal, 45 °C diagonal, 135 °C diagonal and non-directional.
- **Homogeneous Texture (HT):** provides a quantitative representation using 62 numbers, consisting of the mean energy and the energy deviation from a set of frequency channels.

Further details on these visual features can be found in [6]. Our visual features when queried were ranked using variations on Euclidean distance as specified by Mpeg-7. Earlier work within DCU [5] highlights our reasons for employing this metric.

2.2 Result Fusion

Our fusion approach is to treat all components of a query as equal and to not perform hierarchical weighting. An example of hierarchical weighting would be that if we have 6 retrieval experts with 2 query images, that first the queries are sent to each expert, where the results are weighted and combined to form a retrieval result for each expert. Then these six lists (one for each expert which is the combination of the results from that expert for the two query images) and weighted and combined. This produces a weighting hierarchy. The data fusion and weighting approaches we use do not apply any implicit hierarchical ordering. Using the above example of 6 experts and 2 query images, we would directly combine the 12 ranked lists, rather than a multiple staging process.

For data fusion itself we only experimented with the use of CombSUM [2] for combining results lists. Our combinations were always normalized first, and for normalization we employed MinMax normalization, formally given by Equation 1.

$$Norm_{score(x)} = \frac{Score_x - Score_{min}}{Score_{max} - Score_{min}} \tag{1}$$

Our system generates query-time weights for the fusion of different information sources based upon the comparison of the score distribution differences of one retrieval expert as compared to another. When these retrieval expert scores for a given topic are normalized and plotted, we observe that a correlation appears to exist between a retrieval expert whose top ranked documents undergo a rapid change in score and average precision performance for that expert. In other words, the best performing expert was generally the expert which exhibited this rapid change when compared to other experts. This observation however is not universal and we need further investigations to examine why it occurs, if it is chance and if it can be reliably exploited for all query types or only certain types of query. For a complete description of these observations, and how we derive weights from them, refer to [7].

In general terms, our weighting approach is that for any given retrieval expert we calculate the gradient of the scores of the first fifty results, then calculate the gradient of the top 950 results. Using these two numbers we determine how much the gradient changed from the top 50 results to the top 950 results. Using these two figures we can determine which Experts exhibited a greater change are assigned a higher weight (which we refer to in previous paper as variable SC). We can then calculate a weight for each expert by summing all values of SC and then dividing each experts SC score by the sum of all SC. For our experiments we ran the above technique with two variants. These three runs were:

RUN	MAP	Recall	P10	P100
SAW	0.0100	0.1500	0.0750	0.0473
AW1	0.0048	0.0932	0.0583	0.0327
AW2	0.0049	0.1039	0.0417	0.0352

Table 1: Results of automatic retrieval with visual experts only, median MAP 0.0074

- **Standard Auto Weights (SAW)** Our standard query-time weighting as described previously.
- **Aggressive Weighting, variant 1 (AW1)** Similar to above, except that only the top weighted 10 experts were assigned weights as determined by method described. The remainder were assigned a very small weight. This is to determine how effectively our weighting approach is in determining which experts require the most weight.
- **Aggressive Weighting, variant 2 (AW2)** As above, except that the penalty weight is lower again than what was used in AW1.

2.3 Results and Conclusions

Our results are presented in Table 1, which in the caption also presents the median MAP score. We can see from this table that using visual only experts, our runs performed around the median. Our standard approach performed decently, scoring above the median result, whilst our two variations performed worse than the median result. Overall the median result is very low, indicating the difficulty of this years task. We would take some heart from our standard approach to automatic weighting that it performed above median whilst only using visual experts.

3 Event Detection Activity

At a high level of abstraction, our system for event detection contains three basic sub-systems. The first level obtains a number of low-level detectors, including foreground region and person detectors. The second sub-system layer applies the low-level information within a framework to track pedestrians temporally through a scene. Each tracked person has an associated colour model, 2D image location and 3D positioning and velocity information, which has been inferred with the use of a manually obtained image-plane to ground-plane homography. In the third, and final, level of our system events are detected from the movements and interactions between people and/or their interactions with specific areas that have been manually annotated by the user. We will now discuss the features of each of these three basic sub-systems and then conclude with results.

3.1 Sub-system 1: Low-level Data Extraction

Each video frame was processed to extract the foreground regions and the location of possible persons within the scene. Each technique will now be discussed in turn.

3.1.1 Background Model

In our framework, we employed an N layered background model to obtain moving foreground regions within each video frame. Within this model, each pixel is described by N layers. In our experiments, we set $N = 4$. In each layer the pixel has an associated colour, a *weight* and the frame number of when the colour was last observed. When a new colour is observed, the model is updated as follows:

- If the colour exists in one of the layers already (using a user defined threshold), then the weight of that colour is incremented by one. The position of this layer is swapped with the layer directly above it if its incremented weight is greater, or if the higher colour has not been observed for C frames. In our experiments, we used $C = 1500$.

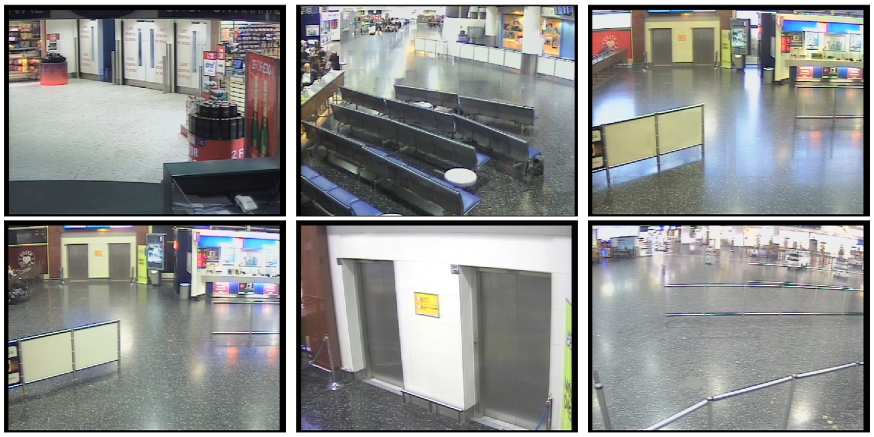
Figure 1: Background images used for each camera: a day-time and night-time image was generated for camera 3.

- If the colour *does not* exist in any of the layers, then lowest layer is reset (i.e. the weight is set to 1) and initialised with the detected colour.

Using this technique, a pixel's background colour is modelled by each of the colours that appear within the top layers that make 75% up at least 75% of the total weight sum within the layered background model. A pixel is detected as foreground if its colour is not found in these top colours. In our system, the layered background models were initialised using a manually generated background image of the particular scene in question – figure 1 shows the background images applied for each camera view.

Shadow removal Shadows and other lighting-changes are a frequent occurrence within the Gatwick dataset. However, using the proposed technique of background subtraction, shadows tend to be erroneously detected as foreground pixels. These shadows may have a negative impact upon a particular systems reliability as shadows can result in increased false detections of pedestrians, or lost tracks. In order to counter this issue, the foreground data obtained was post-processed using a shadow suppression technique. For all foreground pixels, we remove shadow pixels by computing the change in luminance and chrominance. Using the normalised BG colourspace, as outlined in Equation 2:

$$L = R + G + B, g = G/L, b = B/L \tag{2}$$

We compute the change in luminance and the change in chrominance as follows:

$$L_{dif} = |log(\frac{I_L}{B_L})| \tag{3}$$

$$C_{dif} = sqrt((I_g - B_g)^2 + (I_b - B_b)^2) \tag{4}$$

If the following inequality holds, then the pixel is set as shadow (i.e. removed as a foreground pixel):

$$L_{dif} + 18 \times C_{dif} < 0.5 \tag{5}$$

These parameters were trained using manually labelled foreground and shadow pixels.

3.1.2 Person Detection

A low-level person-detector was employed using the OpenCV object detector framework. For each of the 5 camera views, a person-detector was trained using manually segmented positive and negative example data from the development video datasets. Initially, the detector was trained using only

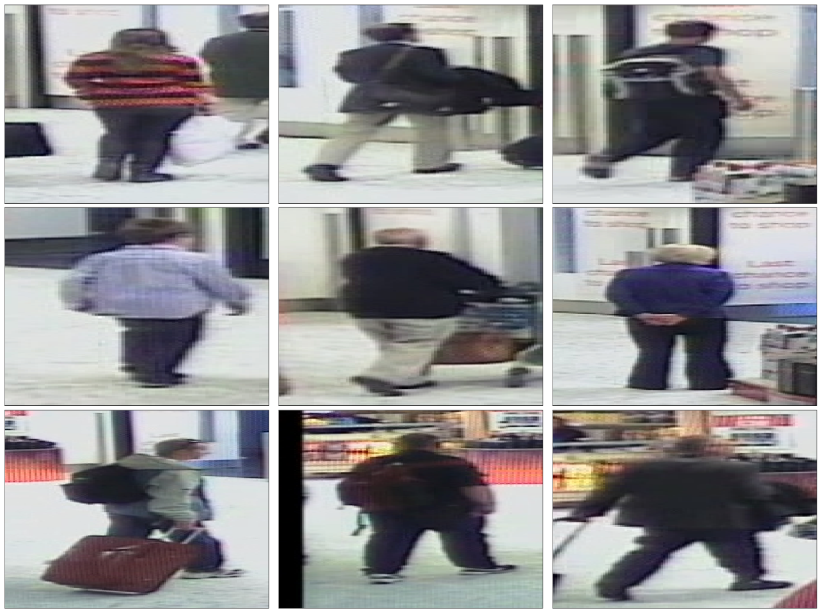

Figure 2: Examples of people used for training our person detector.

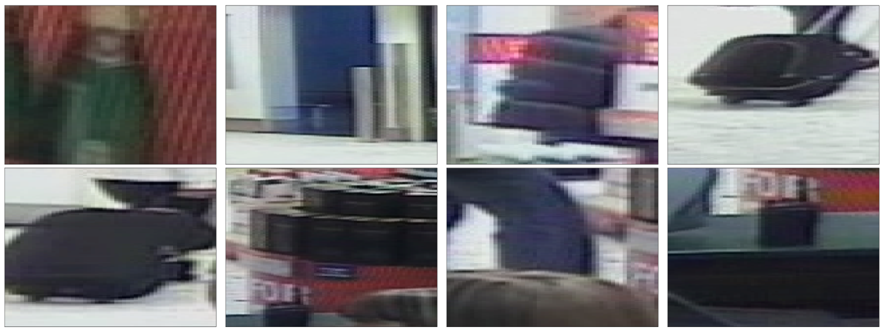

Figure 3: Examples of non-people used for re-training our person detectors.

positive examples – see figure 2 for examples of the people used to train our detector for camera view 1.

We then ran one iteration of active-learning which worked as follows. After the first version of the detectors were trained, these were then run on a large batch of images from the development collection. All detections registered by the detectors were then manually classified as true-positives or false-positives. The detectors were then retrained using these new samples. Figure 3 shows some examples of non-people used in the retraining stage. These detectors were released to the rest of the TRECVID community. Figure 4 shows examples of the results of our trained detector on data from the test video sequences.

3.2 Sub-system 2: Person Detection and Tracking

The second sub-system layer within our framework applies the previously outlined low-level information to track pedestrians temporally through a scene. Tracking is performed in a depth-ordered way, so that people closer to the camera are tracked first. This allows us to infer occlusion for the people further back in the scene. Each person is represented by a 6-colour model. These colours are obtained by drawing a 2×3 grid over them and splitting the person into 6 parts. Each part is represented by the average colour of the foreground pixels within it. We use this model for speed and efficiency, as it

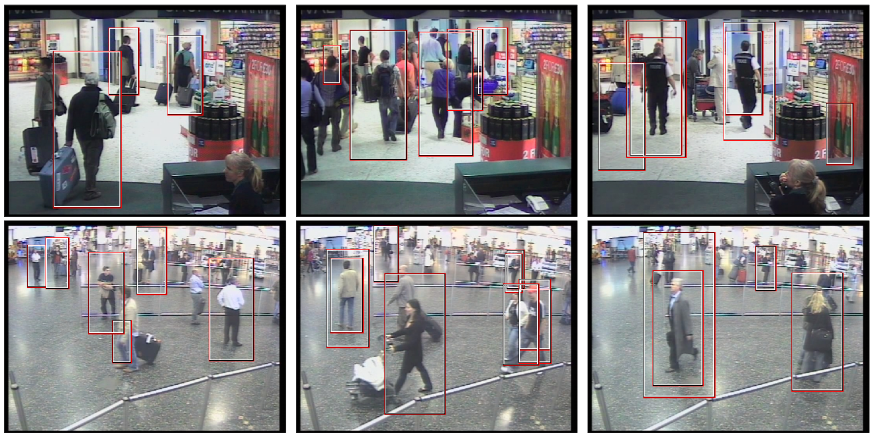
Figure 4: People detected by our person detector for camera 1 (top row) and camera 5 (bottom row).

is very fast to compute using integral images. Tracking is done by exhaustive search in a 21×11 pixel window centred on the last known location of the person.

The next step in this stage of the system is to filter our false-positives obtained from the low-level person detector. This is achieved using two techniques; (1) by removing detections that contain less than 25% of foreground, as detected by our layer-based background model; and (2) by filtering out detections that are too small or too large to be human. This second technique is achieved by employing the use of a linear model of a person's 2D image foot location and their image height. In addition to these techniques, all detections which overlap significantly with a currently tracked pedestrians are removed.

Each tracked person has an associated confidence value, this tends to zero over time if it's existence is not supported in the data. This confidence can be increased updated by significant foreground pixel data within a persons bounding box, or a significant overlap with a detection from the low-level person detector, with a greater overlap causing a greater increase in confidence. However, if the confidence becomes too low (< 0.05), then the tracked person is removed from the system.

Finally, in addition to a tracked persons location in the 2D image, we infer their 3D location using their 2D image foot location and a manually obtained image-plane to ground-plane homography. This information is valuable for some event detectors where 3D positioning and velocity information (obtained temporally).

3.3 Sub-system 3: Event Detectors

In total, we wrote detectors for 5 required events (*ElevatorNoEntry*, *OpposingFlow*, *PeopleMeet*, *Embrace* and *PersonRuns*) and 1 optional event (*DoorOpenClose*). These events were all detected within the third layer of our system and are inferred by the 2D/3D movements of the people, and/or their interactions between other people and 2D/3D areas which have been manually annotated by the user. The 2D areas include regions of interest within an image, and areas of the image where, according to annotated training data, events tend to occur. The 3D areas include regions of interest called hotspots [4, 3], which are 3D regions of interest that have been created from 2D plan-view (or birds-eye) images obtained using the image-plane to ground-plane homography. These hotspot regions can be then used to filter pedestrians based on their 3D statistics such as velocity, location, direction of movement, etc. The confidence for each event was score based upon some function of person tracking confidence, plus the confidence of other factors – such as how likely an event tended to occur within a specific area of the scene.

3.3.1 DoorOpenClose

For each camera view, we found a rectangular region on each door that significantly changed colour when the door opened / closed. In the case of the lift doors in camera 4, we used a colour differential

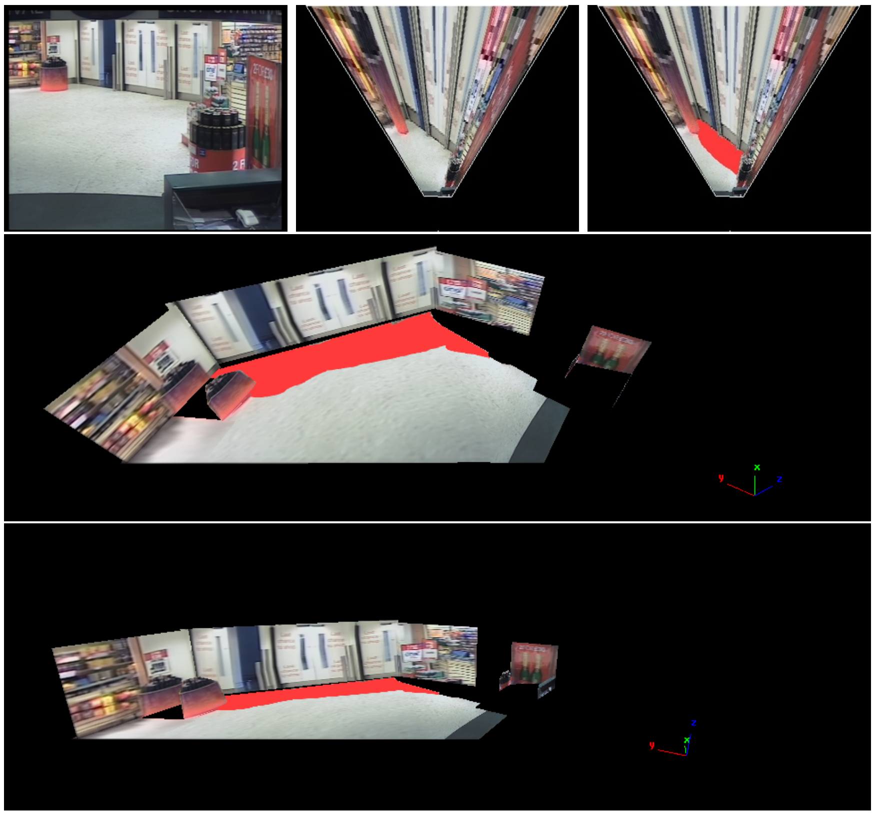

Figure 5: Top Row (left To right): Background Image, 2D plan-view image, 2D plan-view image with hotspot. Middle and Bottom Rows: 3D scene rendering with hotspot.

(the difference of two rectangular region colours). The regions were chosen so as to be high up on the door, so that were not easily occluded by people. We then simply applied a threshold to determine the state of the door (open or closed) and triggered an event when the door closed. For speed and efficiency, we used integral images to quickly detect these events.

3.3.2 ElevatorNoEntry

In camera 4, our person detector was not very robust due to the low number of training samples in this view. We therefore triggered an *ElevatorNoEntry* event each time the elevator doors closed, as detected by our *DoorOpenClose* detector. The confidence of the event was conditioned on any detected people in the locality of the lift doors.

3.3.3 OpposingFlow

In camera 1, a hotspot was created from the 2D plan-view image of the scene, see the area in red in 5. This hotspot was used to filter out all persons in the scene who were on the hotspot for a minimum of 3 frames and who were travelling left to right in the scene. The system then triggered an event when a filtered person either stepped off the hotspot, or stopped travelling in the opposing direction. The confidence of the event was based on the person's tracking confidence plus the length of time they were moving in the wrong direction across the hotspot.

3.3.4 PeopleMeet

This event was detected in each camera using the 3D location of tracked persons. If the Euclidean distance between two people's 3D locations was greater than a threshold, t_1, and over time this

Figure 6: Embrace: Background Image, Confidence map.

distance drops below a second threshold, t_2, then an event is immediately triggered. The confidence of the event is a function of each of the people's tracking confidence over the period of time between the two thresholds being broken.

3.3.5 Embrace

This event was similar to *PeopleMeet*, except the final confidence was then multiplied by a value, α. The value of α is obtained using a confidence map, see 6, which in turn is created using annotated events within then training data. Within this image, the brighter an area is the more likely it is that *Embrace* event will occur. Using this probability image (which is scaled between 0 and 1), a tracked persons probability of taking part in an *Embrace* event in a given frame is defined as the average confidence map value, β, within the person's bounding box at that frame. This can be calculated efficiently using integral images. Finally, α is set to the highest value of β for each of the two persons in the scene over the period of time between the two thresholds, t_1 and t_2 (as defined in the *PeopleMeet* event) being broken.

3.3.6 PersonRuns

This event was detected in each camera using the 3D velocity of tracked persons. If a person is travelling at a velocity of over 150cm per second for 3 consecutive frames, then the system then triggered an event when the person either stopped travelling at that velocity or stepped out of the scene. The confidence of the event was based on the person's tracking confidence plus the length of time they were travelling at a high velocity.

3.4 Results and Conclusions

Unfortunately, as *DoorOpenClose* was not a required event, it was not annotated within the test dataset. As such, there are no results to present for *DoorOpenClose*. In a similar vein, although *ElevatorNoEntry* was a required event, no such events were annotated within the test dataset, leading to no Detection Cost Rate (DCR) results for the event (although our system did produce 189 false positive events). The actual and minimum DCRs for each of the events can be seen in table 2. We can see from this table that our system did not perform very well, this tended to occur for the following reasons outside of the difficulty of this years task; (a) our pedestrian tracking approach was not hugely reliable, thus leading to lost and erroneous tracks; (b) our pedestrian tracker was always initiated using the low-level pedestrian detector, which tended to miss pedestrians in cluttered crowds; and (c) the event detections were inferred by the movements / interactions of the people, if the tracker was not reliable, then the event detections are subsequently limited by the success of the underlying flawed tracking algorithm.

Overall, this year for the event detection submission we focused on developing a re-usable and flexible system framework for use in this, and other, tasks. As a result, we spent significantly less

Event	Act. DCR	Min. DCR
PersonRuns	1.2070	0.9955
Embrace	1.2712	0.9920
PeopleMeet	1.3648	0.9998
OpposingFlow	0.8794	0.7702

Table 2: Event detection results

time developing, training and testing robust and flexible system algorithms. We hope to balance this relationship out in TrecVid 2009.

Acknowledgements

This work is supported by Science Foundation Ireland under grant 07/CE/I1147.

References

[1] AceMedia. The AceMedia Project, available at http://www.acemedia.org.

[2] E. A. Fox and J. A. Shaw. Combination of Multiple Searches. In *Proceedings of the 2nd Text REtrieval Conference*, 1994.

[3] P. Kelly and N. O'Connor. Vision-based analysis of pedestrian traffic data. In *CBMI - 6th International Workshop on Content-Based Multimedia Indexing*, pages 133–140, 2008.

[4] P. Kelly, N. O'Connor, and A. F. Smeaton. Event detection in pedestrian detection and tracking applications. In *SAMT - 2nd International Conference on Semantic and Digital Media Technologies*, pages 296–299, 2007.

[5] K. McDonald and A. F. Smeaton. A comparison of score, rank and probability-based fusion methods for video shot retrieval. In *Proceedings of CIVR 2005*, 2005.

[6] N. O'Connor, E. Cooke, H. le Borgne, M. Blighe, and T. Adamek. The AceToolbox: Low-Level Audiovisual Feature Extraction for Retrieval and Classification. In *2nd IEE European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies*, 2005.

[7] P. Wilkins, P. Ferguson, and A. F. Smeaton. Using score distributions for querytime fusion in multimedia retrieval. In *MIR 2006 - 8th ACM SIGMM International Workshop on Multimedia Information Retrieval*, 2006.