University of Central Florida at TRECVID 2008 Content Based Copy Detection and Surveillance Event Detection

O. Bilal Orhan, Jingen Liu, Jason Hochreiter, Jonathan Poock,
Qinfeng Chen, Ajay Chabra, Mubarak Shah
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, Florida 32816, USA

ABSTRACT

In this paper, we describe our approaches and experiments in content-based copy detection (CBCD) and surveillance event detection pilot (SEDP) tasks of TRECVID 2008. We have participated in the video-only CBCD task and four of the SEDP events. The CBCD method relies on sequences of invariant global image features and efficiently matching and ranking of those sequences. The normalized Hu-moments are proven to be invariant to many transformations, as well as certain level of noise, and thus are the basis of our system. The most crucial property of proposed CBCD system is that it relies on the sequence matching rather than independent frame correspondences. The experiments have shown that this approach is quite useful for matching videos under extensive and strong transformations which make single frame matching a challenging task. This methodology is proven to be fast and produce high F1 detection scores in the TRECVID 2008 task evaluation.

We also submitted four individual surveillance event detection systems. "Person-Runs", "Object-Put", "Opposing-Flow" and "Take-Picture" are the four selected events. The systems rely on low level vision properties such as optical flow and image intensity as well as heuristics based on a given event and context.

1. INTRODUCTION

The Computer Vision Lab team at University of Central Florida (vision@ucf) participated in the video-only content based copy detection and surveillance event detection pilot tasks. We submitted one base run for content based copy detection and one base run for the surveillance events of "Person-Runs", "Opposing-Flow", "Take-Picture" and "Object-Put". We used C++ and the OpenCv library to implement our systems. Based on the evaluation results, our copy detection approach performed very well. In terms of F1 scores, we achieved the highest results for nine out of the ten transformation types. Comparatively, our event detection results were not so significant, but still reasonable. For the last transformation, even though we could achieve high recall, we chose not to incorporate it to our base run due to low precision in detection.

1. Content Based Copy Detection (Video-Only)

Our CBCD solution is an efficient computer vision system composed of 3 main parts: feature extraction & indexing, matching & retrieval and picture in picture detection & extraction. Our system is implemented using C++, OpenCv and a Java platform. Thus, it is significantly faster than comparable Matlab implementations. We have submitted one base run:

VisionUCF.v.base: Result of the proposed computer vision system. The last transformation type is not included in this run because of its low precision. We rely on global image features densely extracted from the target videos for indexing. More specifically, we used Normalized Hu-Moment invariants (NHMI) taken from every frame. Since we are extracting minimal information from each frame, an exact frame match becomes irrelevant. While we lose this capability, the overall sequence matching becomes more robust and efficient. This is the main novelty of our approach, since the frame matching itself is complex under a variety of different transformations and trying to match a sequence of frames makes the problem more complicated. Our approach reduces the dimension of the search space by sacrificing the information of individual frames but keeping the video sequence data. Because we have near 20 million frames in our search corpus, we needed to use serialized Java objects to efficiently save and maintain the indexed data.

The matching and retrieval part makes use of the knowledge that the target query has been transformed and will not match exactly. This is the reason we are not matching the frames in the first place. Rather, we look for a shifted sequence of features that behave similarly with the untransformed data. We match the target sequence through the corpus in a moving window fashion, where we move our search window frame by frame for each video in the database. It is clear that for a good sequence match, the matching error should first go down and go up again; the minimal error frame is the actual match. This second verification of the matching eliminates the non-matching sequences that happened to behave similarly, thus increasing the accuracy of the matching.

The picture-in-picture (a video clip inserted into another one) transformation required us to first localize the inserted video clip. Following that, we can extract the features (which are scale invariant) from the localized clip. If the inserted clip is not a reference clip, the case becomes one that is very similar to insertion of a pattern. Since the features we use are invariant to noise, this case becomes irrelevant. If, however, the inserted clip is a reference video, then the features extracted from it will allow matching regardless of the reduced resolution and other possible noise types. We achieved over 70% accuracy in localizing the inserted clips (picture-in-picture).

For a given query the steps are:

- Convert the MPEG query file to an AVI format that we can use OpenCv.
- Extract the features (NHMIs) for the video itself and for any extant picture-in-picture detector output.
- Match the sequence(s) to whole indexed corpus using our matching algorithm.
- Look for a consistent match that has fluctuating matching error.

As it is necessary to convert the videos to appropriate format and then try to acquire two possible feature sequences, a fair amount of time is required to run the system. The actual matching algorithm takes less than 20% of the time overall time; since the required time for the whole system achieved better performance than the median, we can state our vision system is efficient.

We have learned experimentally that by reducing the problem to sequence matching, we gain efficiency and some degree of robustness against many transformations, but lose the ability to distinguish the frames and match short clips that does not have enough discriminative information.

The results of our base submission are significant; we achieved the high mean F1 scores for 9 of the transformations. The mean processing times are below median for all transformations and the minimum NDCRs are around and slightly above the medians.

2. Surveillance Event Detection Pilot

We assembled our event detection system from individual event detectors created specifically for each event. We have submitted one base run for the event detection pilot (UCF_1) which is composed of four different event detector outputs. The brief description of each system is as follows:

PersonRuns: PersonRuns detection code relies heavily on the optical flow concept to track feature points throughout the scene. The tracked feature points are grouped into clusters. Location, direction and speed of the feature points are used to create clusters; the main tracking component then becomes the clusters rather than the individual feature points. The final step is determining whether or not the tracked cluster is running or not, achieved by using trained motion maps of the scene. The decision is then made by analyzing cluster flow over the scene map.

OpposingFlow: A similar approach to that which was used in PersonRuns was adapted. The decision process is simpler than that of PersonRuns. Only the clusters that move in a pre-defined opposing direction will be detected.

ObjectPut: The algorithm relies on the optical flow of the gridded scene. The flow is averaged within the grid and stored over a number of frames. Then, based on the temporal behavior, a conclusion is drawn for an ObjectPut event.

TakePicture: The approach for detecting TakePicture is based on the assumption that a camera flash is used. Under this assumption, a very efficient system can easily be implemented. Our algorithm searches for an intensity spike in the video and then localizes it to the source to detect a camera flash.

We have used basic computer vision concepts and heuristics to solve these complex problems. Even thought we did not introduce much novelty to the existing research our students greatly benefitted from the pilot study itself.

2. Content Based Video Copy Detection Using Sequences of Normalized Hu-Moment Invariants

Due to the increasing number of digital multimedia content, content-based copy detection is becoming a strong alternative to watermarking. Copy detection is important for copyright control, business intelligence and advertisement tracking, law enforcement investigations and many other areas where content of the media plays an important role. The CBCD systems generally consist of similar parts; feature extraction, indexing and retrieval. One can crudely classify features into two main categories: global image features and local features. For the rest of the system, there are key frame based approaches, learning models, histogram based matching and/or some combination of these methods. We will explain our choices building this system in detail in the following 3 sections. As mentioned in introduction, our system is composed of 3 main parts: feature extraction & indexing, matching & retrieval and picture-in-picture detector. We will be explaining the system components in detail and then presenting the TRECVID evaluation results of our system followed by a discussion.

2.1. Feature Extraction & Indexing

This section presents the choice of the image signatures that will remedy the challenges of the task. Considering the large target dataset, the choice of global features versus local features becomes a straightforward decision. The computational cost of tracking and matching local features over the large dataset is large enough to render it an infeasible option. The global feature selected to represent the images under various transformations is the backbone around which the entire vision system is designed; as such, the signatures to be extracted should be invariant to as many aspects of change as possible.

The well-known Hu moment invariants [HuMoments] are attractive global features for this task because of their invariance under translation, scale, and rotation. Unfortunately, they are sensitive to noise and are not invariant to a gamma or a quality change. As reducing the effect of noise on the moments has been addressed in the literature by various authors, the main issue we were faced with was finding a method to make the moments invariant to gamma and quality changes. For this, we chose to use the normalization of the moments to the powers of the first moment, presented by Hupkens et. al.[HM Normalization]. For each of the seven moments normalization is performed by dividing the moment to the powers of the first moment as shown in Table 2.1.

$$\omega_1 = \mu_1 / \mu_1 = 1$$

$\omega_2 = \mu_2 / \mu_1^{\,2} * 1000$	$\omega_3 = \mu_3 / \mu_1^{\,3} * 10000$
$\omega_4 = \mu_4 / \mu_1^{\,3} * 10000$	$\omega_5 = \mu_5 / \mu_1^{\,6} * 100000$
$\omega_6 = \mu_6 / \mu_1^{\,4} * 10000$	$\omega_7 = \mu_7 / \mu_1^{\,6} * 100000$

Table 2.1. Normalized Hu moment invariant normalization [HM Normalization]

The moment invariants have been extensively used for single image analysis tasks, but have historically not been applied to video-based applications as commonly. Completely different images may have close values for the image moments, which does not necessarily suggest that they may be similar; in fact, completely unrelated images can have the same exact moment values. To summarize: the invariant moments map one image to multiple different images, one of which may be a copy. When the temporal order of a video is also incorporated, it becomes very unlikely that this method will match a sequence of unrelated images. This is the reason that invariant moments are not good enough for single image analysis, but are still suitable for video analysis.

The formulas in Table 1 show the normalization of Hu invariants μ_i to obtain the normalized integer moments ω_i. For our experiments, we used ω_k (where $6 \geq k \geq 2$) , as the 1st moment is always 1 and the 7th moment has a very wide value range and, as such, is not suitable for our system.

As discussed previously, our image signatures are not robust for single image matching; rather, they require a sequence of matching frames. To increase the robustness of the system, we processed and indexed all the frames from each of the 438 reference videos, before correlating them with the corresponding frame and video id for 6 integer signatures per frame.

There are roughly 20 million frames in the dataset; therefore, we had an approximate total of 120 million features to index and search. In order to be able to store and process this amount of data, we used data synchronization using custom-made Java objects. This allowed us to handle the data of this quantity in an efficient way. Using these Java objects, each video is indexed as a long sequence of extracted features.

2.2. Matching & Retrieval

The matching and retrieval component is one of the most sensitive parts in the whole system. Under extensive transformations, the query videos (and, by extension, data extracted from the query videos) may not resemble their parent videos at all. Additionally, as a result of certain transformations (e.g. frame dropping), the query frame sequence may have naturally non-matching features from dropped frames. Considering possible changes and the size of the data, an adaptive and efficient feature matching algorithm is necessary.

Empirical studies have shown that the effect of the noise is proportional to the magnitude of image moment response of the frame. Using this observation, we propose an adaptive difference sliding window search for matching the target queries. We allowed for a 10% error in matching, where error is defined as the ratio of mismatched frames. The frame matching is decided with an adaptive allowed difference $\nabla_{i\,k}$ (1) (for the i^{th} moment invariant for the k^{th} frame) between the frame in the query and the frame in the dataset.

$$\nabla_{i\,k} = 10 * l * \{1 + \mathrm{mod}(\omega_{i\,k}, 1000)\} \qquad (1)$$

The frame matching by itself does not mean anything, as the moment features are not very descriptive. The second important step of the matching function is the sliding window search of the whole query, where the query feature sequence is matched to the target data as a whole by temporally sliding through the entire video. The last part of the matching and retrieval algorithm consists solely of determining the strength of the match. If the query does not contain enough information to be distinguishable, the matching function may match the query to a long list of random videos; this can happen if the query is a steady, generic scene. In that case, we conclude that the query is a false alarm and flag it as a non-match.

For a query to be considered as a good match, it should match a video with more than one window. The matching error will be decreasing when window gets closer to the exact matching portion and increasing again when the window is moving away. A matching log sample from the system is shown in Table 2.2.

MATCH at time : 9:32:4 missedFrames:9
MATCH at time : 9:32:6 missedFrames:7
MATCH at time : 9:32:8 missedFrames:5
MATCH at time : 9:32:12 missedFrames:0
MATCH at time : 9:32:16 missedFrames:2
MATCH at time : 9:32:18 missedFrames:10
MATCH at time : 9:32:20 missedFrames:15

Table 2.2. Sample run output of query matching algorithm.

Actual matching data is shown in Figure 2.1. The figure represents the matching of the query number 1 to reference video BG_36090. The transformations applied to the query are gamma change and text insertion, both of

which can be considered as a type of noise.

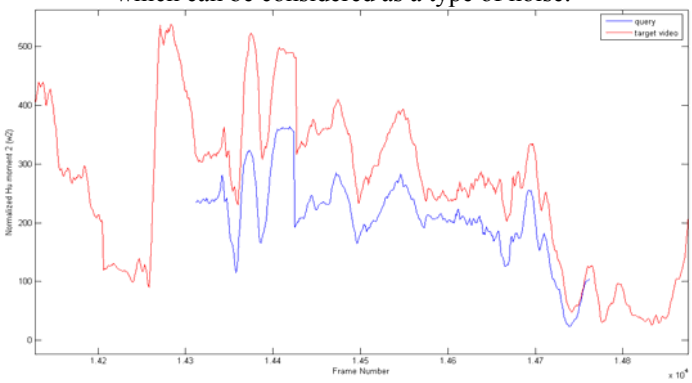


Figure 2.1. Feature Matching result for Query 1.

2.3. Picture-in-Picture EXTRACTION

The task description allows for two types of picture-in-picture transformation in the challenge. The first type includes the reference video in the background; therefore, the reference video occupies most of the scene. For this type of transformation we do not need to take extra measures, as this degenerates into an insertion-of-patterns case and therefore our system can match this transform as it is (see Figure 2.2.a and 2.2.b.).

Figure 2.2.a. A frame from original video (right), query with picture in picture inserted and flipped (left)

On the other hand, if the reference video we are trying to match is the inserted video, we need to localize it to be able to get its global shape characteristics (Normalized Hu-Moment Invariants). To solve this kind of transformation, we need to localize the frames inserted and then extract global features on that local window. Since the normalized moment invariants we use are scale invariant, the localized frame will match the original video.

The algorithm we use takes the image derivatives from + and - x–axis and tries to find persistent strong horizontal lines using Hough lines [Hough Lines], where $\sum_i^N I_{+x} + I_{-x}$ is consistent over the whole query. Then, we connect the horizontal parallel lines of a given allowed ratio to a window and localize that picture. Figure 2 shows the persistent lines and the localization output of the algorithm over that result.

The algorithm worked with 74% overall accuracy, missing some of the positive picture-in-pictures and finding non-existing windows within frames. Since this step is followed by an actual matching algorithm (explained in matching & retrieval), incorrect windows will not be matched to the reference videos and will not affect the overall system accuracy. On the other hand false negatives, will reduce the accuracy of the system, as there is no other way of localizing the frames inserted. Figure 2.3 displays some of the good detections, where the detected picture portion is highlighted with a blue frame.

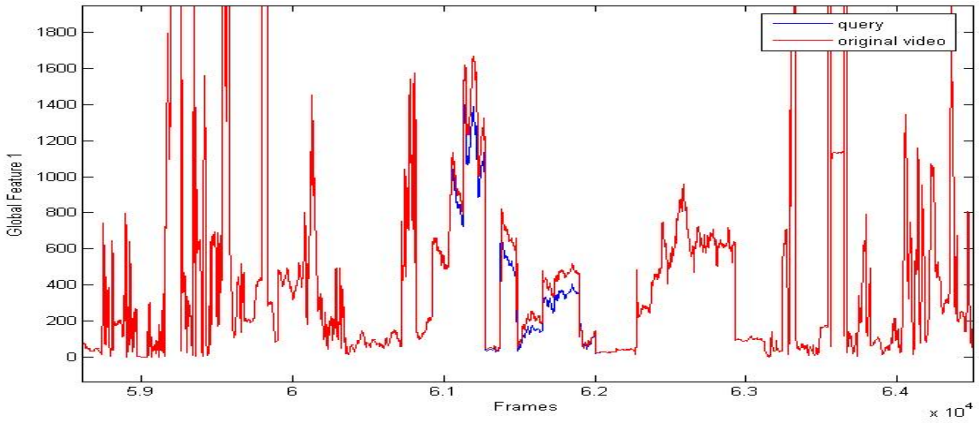


Figure 2.2.b.Actual query matching output for Figure 2.2.a.

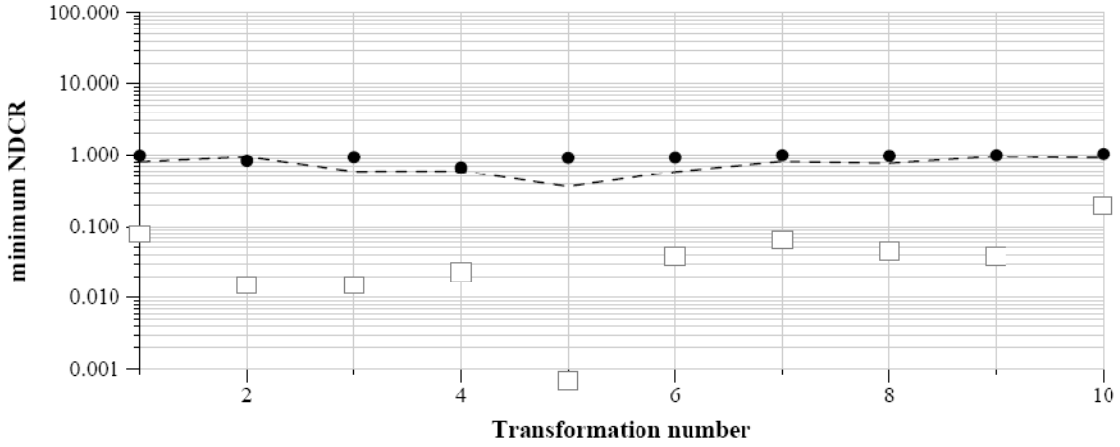
Figure 2.3. Finding the picture inserted using persistent horizontal lines

Figure 2.4. Picture in picture finding successful samples

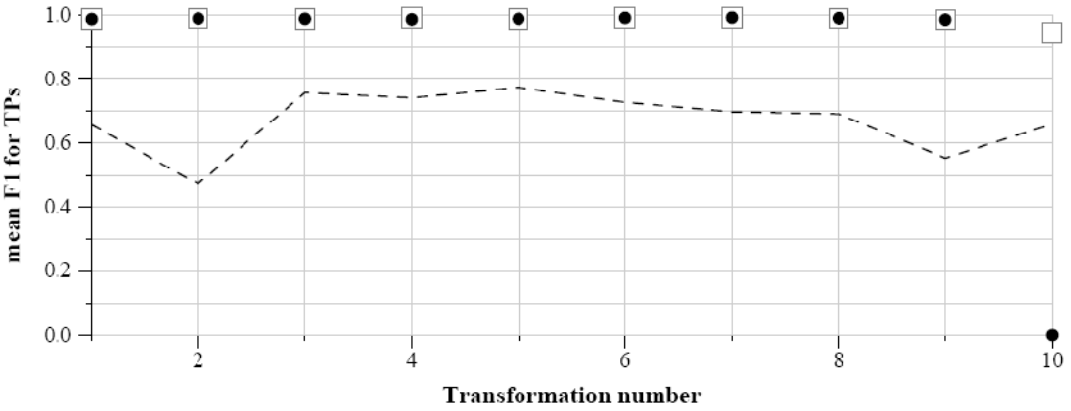
2.4. Results and Discussions

TRECVID 2008: copy detection results

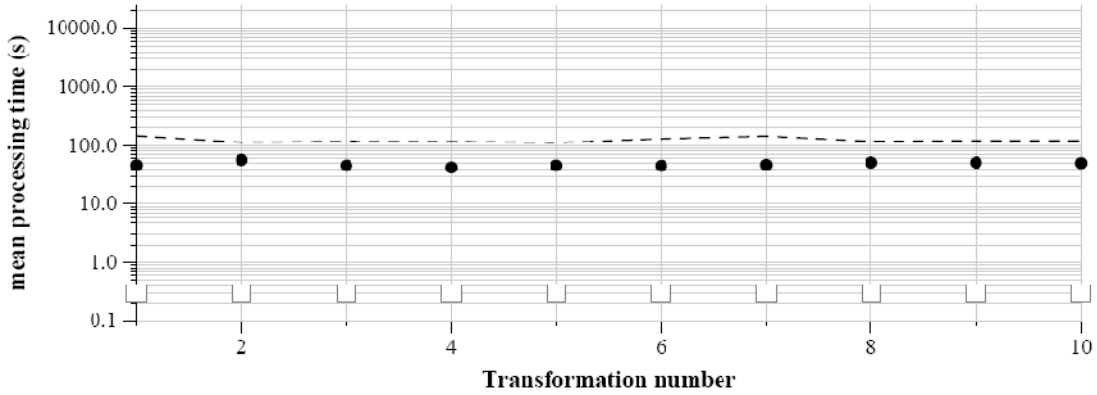
Run name: VisionUCF.v.base
Run type: video-only



Run score (dot) versus median (---) versus best (box) by transformation



Run score (dot) versus median (---) versus best (box) by transformation



Run score (dot) versus median (---) versus best (box) by transformation

3. Surveillance Event Detection Pilot

3.1. PersonRuns

Our PersonRuns detection code relies heavily on the optical flow concept to track feature points for each frame of a video. The feature points in each frame are then grouped into clusters (called local clusters) using a clustering algorithm. The properties of the local clusters can only be held for a length of one frame before being updated in the next frame. Another group of clusters (global clusters) are used to store useful local clusters. The properties (in this case, the x and y coordinates and associated derivatives thereof) of the global clusters in the current frame are then used to predict those of the next frame. Finally, the algorithm makes comparisons between the predicted global clusters' x, y, dx, and dy and local clusters' location and displacement information and assigns the local clusters to matching global ones if the matching error is small.

The clustering algorithm incorporates some other features to improve the accuracy of the PersonRuns event. Among others, this involves the previously mentioned position prediction check. In addition to that, we also used cyclic motion analysis, tracking history analysis and cluster combination in order to increase the accuracy of the tracker. Furthermore, in order to actually make the decision, we had to compute the average speed of a scene.

We observed that a person makes slight cyclic movements while running, so cyclic motion analysis is applied to allow the system to distinguish and better predict the movement of a running person. Therefore, the chance of tracking PersonRuns is increased. To reduce lost tracks caused by optical occlusion, we implemented a tracking history. We allowed the prediction of the cluster movement throughout several frames even if there is no matching movement; this allows the system to recover from occlusion by assuming the similar motion carried through the matchless frames. In order to reduce noise from the background (and, by extension, produce better results), the averaged speed of each region (in this case, a region being defined as one square in a grid of overlapping squares, designed to reduce issues arising from discontinuous areas) in the videos is calculated as a reference to filter out people walking and other noise. The method of combining clusters is used to merge the movement of the upper and lower body of the same person. It is also used to prevent helpful local clusters from being discarded and incorrect local clusters from being assigned to the matching global clusters. Figure 3.1 shows a successful tracking and detection output of our system.

The proposed system detects only 45% of the person run events with a low precision (4%). These results are the combined results of 5 different camera views that have been tested with generic system settings. Using individual cameras and changing the system settings, we could achive much better results; for the purposes of this challenge, however, we used the same set of parameters for each camera - which, in turn, greatly reduced our system accuracy.

3.2. ObjectPut

The ObjectPut event is generally characterized by a smooth downward motion in the camera. The system processes the motion history images and filters out the downward motion which is consistent with the ObjectPut criteria. Essentially, the algorithm first splits the video to be processed into a grid of squares. Following that, the optical flow is computed over the entire frame, but averaged within each square. From these averages (which are stored over a period of frames), the system attempts to draw conclusions. In particular, if certain grid regions have reasonably large average optical flow in a downward direction over several frames, the system decides that there is an ObjectPut event. Figures 3.2. and 3.3. show intermediate system output for such events.

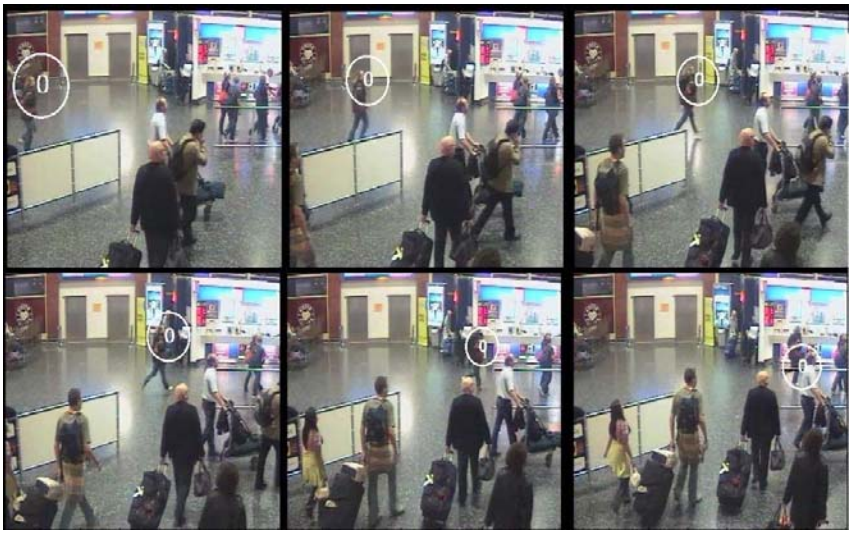


Figure 3.1. PersonRuns event detector output.

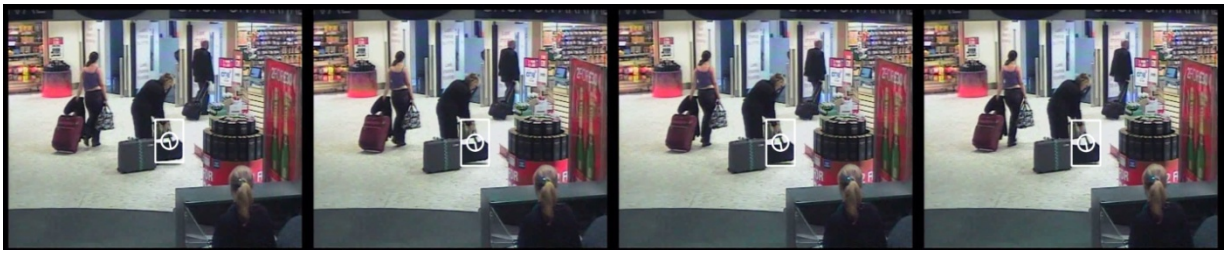


Figure 3.2. Object put event detected

The system is designed to detect the ObjectPut event for large objects such as bags, a smaller subset of the problem that is outlined in the pilot annotation, which also counts a person releasing their hold on a small object (e.g. bottle of water) as an ObjectPut event. In cases such as those, with small objects and no downward motion, our system will be useless as it works solely off this particular motion.

One of the major flaws inherent in this approach is that the system will occasionally detect a person sitting down as an ObjectPut event. Conversely, the downward motion from a person walking towards the camera in a given scene is not detected by the system as a positive event, as this type of downward motion is ultimately different than the one expected from an ObjectPut event.

This simple approach (as with all of our systems) works in real-time and detects a considerable amount of ObjectPut events in the test set (roughly ~30% correct detection). Even though this system does not fully work within the boundaries of the original task's specification, it produces significant results under the assumption that it detects ObjectPut events for large objects.

Fig 3.3. Object put event detected

3.3. Opposing Flow

This task is restricted to only one camera in the system, as the direction of the opposing flow is only defined for that given camera.We adapted the clustering algorithm we used for the PersonRuns event for use here by making some minor changes. We no longer needed some of the properties of the running clustering, such as cyclic motion analysis and the average speed decision computation. Our algorithm can easily be adapted to detect the person opposing the general flow of the scene, even without a predefined direction of opposing flow. Even though we achieved a high recall (75%) , the system precision is unacceptably low (less than 1%).

3.4. Take Picture

We started this event by trying to detect the camera and associated motion, which proved to be very difficult. Under the rules covering an occluded event, the object detection could be nigh-impossible due to the nature of occlusion and the size of the object. In an attempt to make the problem easier and to allow us to operate in real-time, we accepted a large assumption – all TakePicture events would be preceded by a camera flash. Based on the annotation data for the development set, this seemed to be a fairly reasonable assumption.

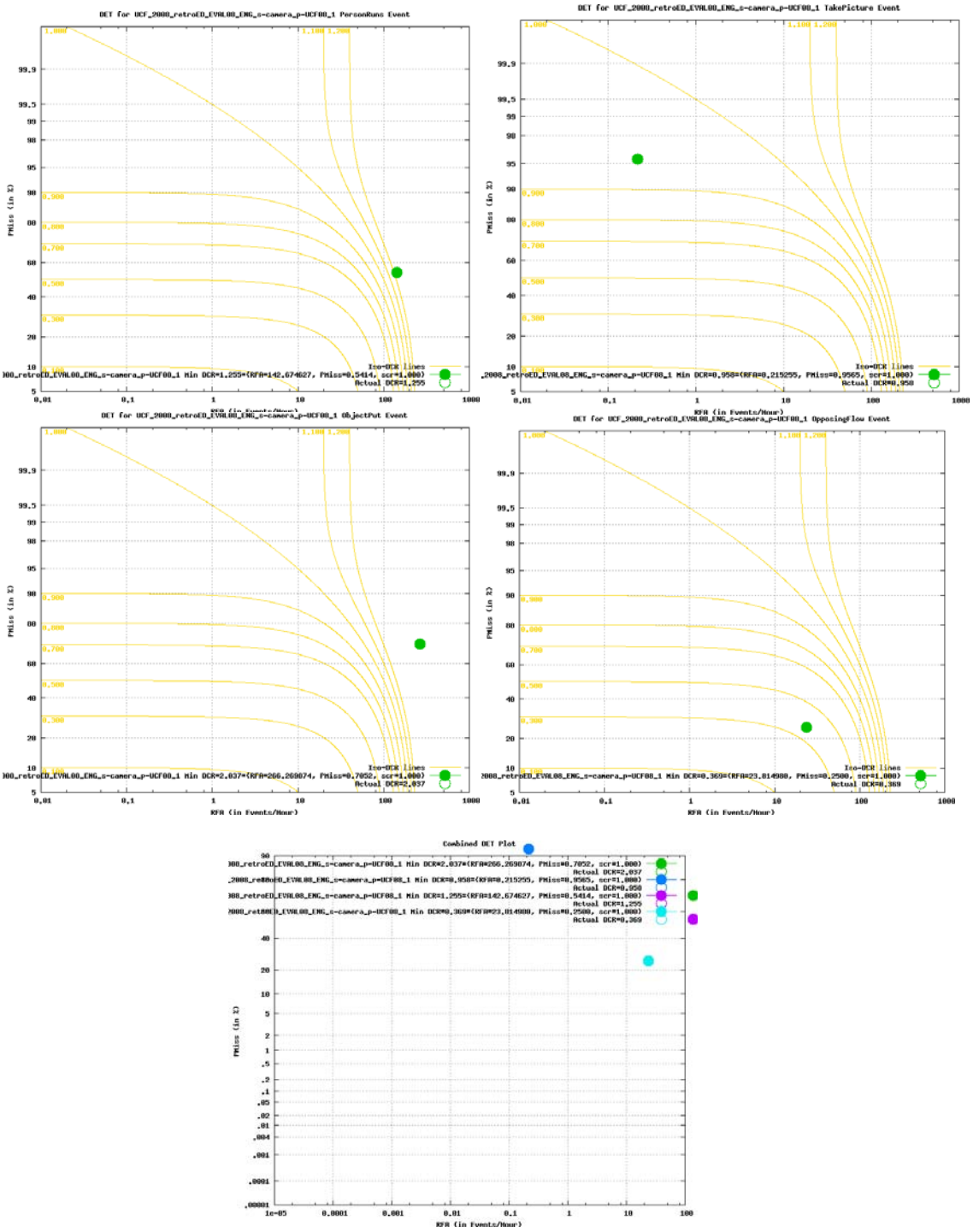
The approach we used simply finds a camera flash in a scene. The vision system searches for a peak in the color intensities and, upon detecting an intensity spike, distinguishes the camera flash. Once the flash is found, the event can easily be located spatially and temporally.

Figure 3.4. Camera flash detected

This approach works even in the case where the camera itself is not visible, as the flash is generally apparent throughout the entire scene. A particular weakness of this system is that it is very subject to making false detections based on other intensity spikes that may occur (e.g. the reflection of a bolt of lightning). The system did not operate as we anticipated, as the TRECVID evaluation results showed that our detector has low accuracy.

3.5. Results and Discussions

	#Ref	#Sys	#CorDet	#FA	#Miss	Act. RFA	Act. PMiss	Act. DCR	Min RFA	Min PMiss	Min DCR	Det. Score
ObjectPut	1944	573	573	13607	1371	266.3	0.7052	2.0366	266.3	0.7052	2.037	1
TakePicture	23	1	1	11	22	0.215	0.9565	0.9576	0.215	0.9565	0.958	1
PersonRuns	314	144	144	7291	170	142.7	0.5414	1.2548	142.7	0.5414	1.255	1
OpposingFlow	12	9	9	1217	3	23.82	0.25	0.3691	23.82	0.25	0.369	1



ACKNOWLEDGEMENTS

We thank our colleague Syed Zain Masood smasood@cs.ucf.edu for his invaluable contribution to our Content-Based Copy Detection System and his Picture-in-Picture Detection algorithm.

REFERENCES

[*TRECVID*] Smeaton, A. F., Over, P., and Kraaij, W. 2006. Evaluation campaigns and TRECVid. In Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval
(Santa Barbara, California, USA, October 26 - 27, 2006). MIR '06. ACM Press, New York, NY, 321-330. DOI= http://doi.acm.org/10.1145/1178677.1178722

[*OpenCv*] The Open Computer Vision Library (http://sourceforge.net/projects/opencvlibrary/)

[*HuMoments*] M.K. Hu, "Pattern Recognition by Moment Invariants," Proc. IRE Trans. Information Theory, vol. 8, pp. 179-187, 1962.

[*HM Normalization*] Th. M. Hupkens, J. de Clippeleir "Noise and intensity invariant moments" Pattern Recognition Letters, Volume 16, Issue 4, April 1995, Pages 371-376

[*Comparative Study*] Law-To, J., Chen, L., Joly, A., Laptev, I., Buisson, O., Gouet-Brunet, V., Boujemaa, N., and Stentiford, F. 2007. Video copy detection: a comparative study. In Proceedings of the 6th ACM international Conference on Image and Video Retrieval (Amsterdam, The Netherlands, July 09 - 11, 2007). CIVR '07. ACM, New York, NY, 371-378. DOI= http://doi.acm.org/10.1145/1282280.1282336

[*H.Lines*] Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11–15 (January, 1972)