Compressed domain video processing of meetings for activity estimation in dominance classification and slide transition detection

Chuohao Yeo
Kannan Ramchandran

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-79
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-79.html

June 7, 2008

Acknowledgement

This research was funded in part by the US VACE program and the Singapore Agency for Science, Technology and Research (A*STAR). We thank Sileye Ba, Gerald Friedland, Daniel Gatica-Perez, Yan Huang, Hayley Hung, Dinesh Jayagopi, Jean-Marc Odobez and Oriol Vinyals for their helpful discussions about the work. We are especially grateful to Hayley Hung and Dinesh Jayagopi for parsing the dominance annotations and sharing the results and analysis with us.

Compressed domain video processing of meetings for activity estimation in dominance classification and slide transition detection

Chuohao Yeo and Kannan Ramchandran
Dept. of EECS, UC Berkeley
{zuohao,kannanr}@eecs.berkeley.edu

Abstract

Compressed domain processing of video has been a widely used tool in enabling computationally efficient video analysis in the last decade or so since the standardization of video compression in the form of MPEGx/H.26x. We consider the use of such features in the meeting domain to reduce the processing time of video analysis. We review the various compressed domain features that can be extracted easily from compressed videos. In this report, two applications of interest in meeting analysis are considered. First, we present work on activity level estimation which is used in dominance modeling of meeting participants. Second, we look at the problem of detecting slide transitions in meetings, which can be used as contextual cues for estimating the visual focus of attention of meeting participants. In both applications, our experimental results show that compressed-domain methods do as well as their corresponding pixel-domain methods, but only require a fraction of computational costs.

1 Introduction

Our research goal is to reduce computational complexity in the analysis and identification of events and trends in meetings, so as to reduce processing time for both on-line applications and for batch processing. Specifically, we study the task of automatically estimating activity levels of participants which are in turn used for estimating dominance in group interactions. The working hypothesis here is that the more active a participant is in the meeting, the more dominant he is.

In our work, we leverage the fact that meeting videos are already in compressed form to extract compressed-domain features at very low cost [11]. This reduces the computational complexity of feature extraction for use in the analysis of meetings. As a secondary objective, we want to provide a working implementation of software that can easily extract these features from MPEG-4 videos. This also allows us to explore a new video format for compressed domain processing, which has traditionally only used either MPEG-1 or MPEG-2 videos.

To provide additional features for estimating visual focus of attention (VFOA), which in turn can be used for dominance modeling, we also investigate the task of automatically detecting slide

changes. In the meeting dataset, the participants make use of the projection screen for discussion purposes, and it has been observed that participants tend to look at the projection screen when there is a slide transition. Thus, slide transition can be used as a contextual cue for improved VFOA performance. We propose a compressed-domain processing approach to detect slide transitions.

1.1 Contributions

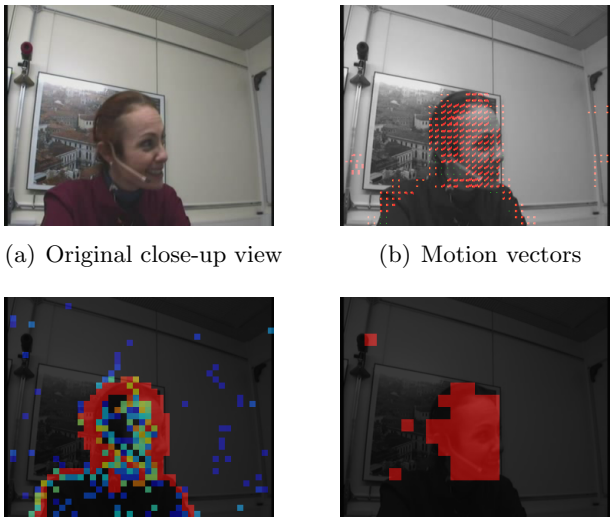
In this work, we investigate the use of computationally inexpensive compressed-domain features for mid-level tasks such as skin blob detection and activity level measurements. Rather than relying on highly complex person models, we develope features that are computationally simple while descriptive of salient activity patterns, both with respect to specific individual and across participants in a meeting. These features are generated from compressed-domain information such as motion vectors and block discrete-cosine transform coefficients that are accessible with almost zero cost from compressed video [11]. We evaluate these features in the task of estimating dominance level of meeting participants, and show that compressed-domain features perform almost as well as pixel-domain features, but only at a fraction of its computational cost.

We also implement a simple yet effective approach to detecting slide transitions in the compressed domain. Since the recording camera is stationary, the projection screen occupies a fixed area in the center view, and its coordinates and extent are determined manually and used in the feature extraction procedure. How to determine the location of the projection screen automatically would be a subject of future investigation. Our experimental results show that the compressed-domain method perform just as well as a pixel-domain method, but only at a fraction of its computational cost.

2 Compressed domain features

Compressed domain video processing has been developed over the last decade or so following the advent of compressed video standards like MPEG. However, a survey of this literature reveals that most work to date has been focused on (i) synthetic video analysis, such as video shot detection [14, 13, 10] and text caption extraction [15]; (ii) video indexing, querying and retrieval [8]; (iii) synthetic video manipulation, such as video resizing [4] and transcoding applications [12]; and (iv) optical flow estimation [2]. Here, we apply these techniques to a new problem domain, that of meetings video analysis.

In this section, we review some of the compressed domain techniques that are applicable for our work. Fig. 1 shows a preview of the various compressed domain features that can be extracted cheaply from compressed videos.



(a) Original close-up view

(b) Motion vectors

(c) Residual coding bit-rate

(d) Detected skin blocks

Figure 1: Example output from compressed domain feature extraction. *(Best viewed in color.)*

2.1 Motion vectors

Motion vectors, illustrated in Figure 1(b), are generated from motion compensation during video encoding; for each source block that is encoded in a predictive fashion, its motion vectors indicate which predictor block from the reference frame (in this case the previous frame for our compressed video data) is to be used. Typically, a predictor block is highly correlated with the source block and hence similar to the block to be encoded. Therefore, motion vectors are usually a good approximation of optical flow, which in turn is a proxy for the underlying motion of objects in the video [2]. However, motion vectors are computed for the sake of compression and not originally meant for video analysis. Therefore, we would need to post-process the extracted motion vectors by removing unreliable motion vectors. Following the approach of [2], we have implemented filtering for the extracted motion vectors. Specifically, by assigning to each motion vector a confidence measure based on local texture information, we remove the unreliable motion vectors, keeping only those with high confidence. The confidence measure can be computed using the DCT AC coefficients, which can be efficiently extracted from the MPEG video. Concretely, if $X(i, j)$ is the (i, j)th coefficient of the 8×8 DCT of a 8×8 block (where $0 \leq i, j \leq 7$), and λ is the confidence measure associated with that block, then:

$$\lambda = X(0, 1)^2 + X(1, 0)^2$$

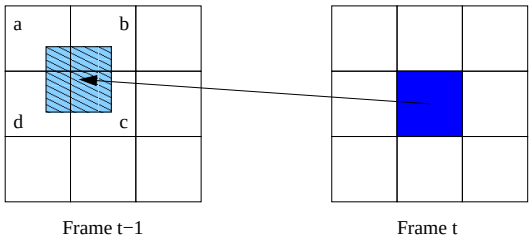


Figure 2: Illustration of how DCT DC terms are updated using motion vectors and residual. The block to be reconstructed in frame t, shown filled in blue, is predicted by a block in frame t-1, shown filled in blue with stripes. In general, the predictor overlaps with 4 blocks, labeled a-d here. The update is computed by considering the amount of overlap with each block, with an additional correction term due to the residue.

2.2 DCT coefficients

DCT coefficients are an alternate representation of the actual pixel values in an intra-encoded block, and of the prediction residual in an inter-encoded frames. In an intra-encoded block, the DC term of its DCT represents the average of the block of pixels; these can be utilized directly to build a spatially sub-sampled version of the frame [13]. However, in an inter-encoded block, the DC term of its DCT represents the average of the prediction residual, and gives limited information about the actual pixel values. We implement a first-order approximation approach described by [13] to build spatially sub-sampled frames (known as DC image sequence) even if they were inter-coded. Suppose that a block in the current frame overlaps with 4 blocks, $S = \{a, b, c, d\}$, as in Fig. 2, where $f_i, i \in S$ is the fraction of the block that overlaps with each of the blocks in S. Furthermore, let $\hat{Y}_i, i \in S$ be the reconstructed DC value of each of the blocks, $\hat{Y}_t$ be the DC value of the current block to be reconstructed, and ΔY_t be the DC value of the prediction residue. Then, $\hat{Y}_t$ is reconstructed as:

$$\hat{Y}_t = \sum_{i \in S} f_i \hat{Y}_i + \Delta Y_t$$

This gives reasonable results if the GOP (group-of-picture) size, or the interval of intra-encoded frames, is kept relatively small (about 9-15).

2.3 Residual coding bit-rate

We also investigate an additional feature: residual coding bit-rate. This is the number of bits used to encode the block residual following motion compensation at the video encoder. While the motion vector captures gross block translation, it often fails to fully account for non-rigid motion such as lips moving. On the other hand, the residual coding bit-rate is able to capture the level of such motion, since a temporal change that is not well-modeled by the block translational model will result in a residual with higher energy, and reflected in the bits needed to entropy code it. Hence, this is complementary to the extracted motion vectors. Visually, we have noticed that it also correlates well with high activity levels.

2.4 Skin-color blocks

By putting together some these compressed-domain features, we can then implement block-level skin detection. The knowledge of skin-color blocks will allow us to consider activity levels of the face and hand in dominance modeling, and ignore background clutter such as the motion of clothing. To do this, we implement a Gaussian Mixture Model (GMM) based skin-color block detector [9] that can detect head and hand regions. This works in the compressed domain with chrominance DCT DC coefficients and motion vector information, and produces detected *skin-color blocks* such as in Figure 1(d).

We use a GMM to model the distribution of chrominance coefficients [9] in the YUV colorspace. Specifically, we model the chrominance coefficients, (U, V), as a mixture of gaussians, where each gaussian component is assumed to have a diagonal covariance matrix. In other words, the probability density function (PDF) is given by:

$$p_{U,V|\text{skin}}(u, v|\text{skin}) = \sum_{k=1}^{K} \frac{1}{2\pi\sigma_{U,k}\sigma_{V,k}} \exp\left(-\frac{1}{2}\left[\frac{(u-\mu_{U,k})^2}{\sigma_{U,k}^2} + \frac{(v-\mu_{V,k})^2}{\sigma_{V,k}^2}\right]\right)$$

where K is the number of gaussian components, and $(\mu_{U,k}, \mu_{V,k})$ and $\begin{pmatrix} \sigma_{U,k}^2 & 0 \\ 0 & \sigma_{V,k}^2 \end{pmatrix}$ are respectively the mean vector and covariance matrix of the kth gaussian component. We then learn the parameters by applying Expectation Maximization [3] (EM) on a set of training face images. In our implementation, we chose $K = 5$.

In the Intra-frames, we compute the likelihood of observed chrominance DCT DC coefficients according to the trained GMM and threshold it to determine skin-color blocks. Specifically, when (u, v) are the actual chrominance DCT DC coefficients of a block, the block is declared to be a skin-color block if for some pre-determined threshold τ:

$$p_{U,V|\text{skin}}(u, v|\text{skin}) > \tau$$

Since MPEG-4 uses a YUV colorspace for encoding, there is no need for any additional steps to perform color-space conversion. Furthermore, since the chrominance DCT coefficients are quantized during video compression, we can use a look-up table (LUT) approach for increased computational efficiency.

Skin blocks in the Inter-frames are inferred by using motion vector information to propagate skin-color blocks through the duration of the GOP (group-of-picture). This is similar to an approach for object tracking in the compressed domain [5]. However, in the presence of long GOPs, such as in the AMI meeting videos, accumulated errors could lead to large areas of the frame being falsely detected as skin-color blocks. To prevent this, we add an additional verification step, performed in the pixel domain, to remove blocks that are erroneously tagged as skin-color blocks. This is done by thresholding the number of pixels in the block that are classified as skin, using the same

criterion as in (2.4). Note that this verification step only has to be done if a block is suspected to be a skin block.

We can also apply the GMM model to DCT DC coefficients estimated using the method in Section 2.2. However, as discussed earlier, the recovered DCT DC coefficients of predictively-coded blocks are fairly accurate only when the GOP size is small. For compressed videos with much larger GOP size, the method described here gives much better performance.

3 Activity level estimation for dominance classification

3.1 Technical description

To estimate individual activity level, we turn to the use of motion vector magnitude (see fig. 1(b)) and residual coding bit-rate (see fig. 1(c)) in estimating activity level. Specifically, we investigate the use of both motion vector magnitude and residual coding bit-rate, averaged over the detected skin blocks in each of the close-up camera views. Our rationale for using this is that these features capture the level of activity for each meeting participant by measuring the amount of movement they are exhibiting.

To detect when a participant is not in the close-up view, we threshold the number of skin-colored blocks in the close-up view. In this work, we used a threshold of 2% of the total number of blocks in one frame. Otherwise, if the participant is visible in the close-up view, we measure his motion activity by using either or both of motion vector magnitude and residual coding bit-rate. To compute a normalized motion activity from motion vector magnitude for participant i in frame t, we first calculate the average motion vector magnitude, $v_i(t)$, over the skin-colored blocks in each frame. For each participant in each meeting chunk, we then find the median of average motion vector magnitude over all frames where the participant is in the close-up view. Next, we compute the average of the medians, $\bar{v}$, of all the participants. The motion activity level from motion vector for participant i in frame t, $v_i^n(t)$, is then computed by normalizing as follows:

$$v_i^n(t) = \begin{cases} \frac{v_i(t)}{2\bar{v}} & v_i(t) < 2\bar{v} \\ 1 & v_i(t) \geq 2\bar{v} \end{cases}$$

The motion activity level from residual coding bit-rate is also normalized in a similar fashion. Note that if a participant is not detected in a frame of the close-up view, he is assumed to be presenting at the projection screen, and is assigned an activity level of 1 for that frame.

The features that we used for our dominance experiments were (i) motion activity level from motion vector; (ii) motion activity level from residual coding bit-rate; and (iii) average of motion activity level from motion vector and from residual coding bit-rate. We then sum up the computed activity levels over any desired segment of a meeting. The sum for each participant then quantifies how dominant he is; the higher the sum, the more dominant the participant.

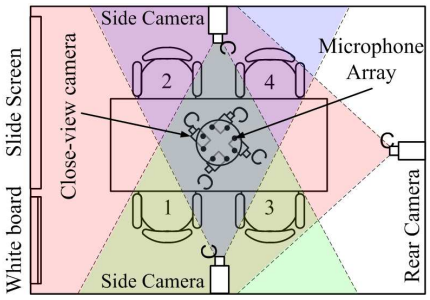


Figure 3: Floor plan of smart meeting room.

Figure 4: All available views in the data set.

3.2 Experiments

3.2.1 Data and annotation

We evaluate our proposed method on the publicly available AMI meeting corpus [1]. The meetings have been recorded in IDIAP's smart meeting room (see floor plan in Figure 3. In this dataset, there is a camera taking a close-up shot of each participant, for a total of four close-up camera views (shown in the bottom row of Figure 4). Each of these video streams has already been compressed by a MPEG-4 video encoder with a group-of-picture (GOP) size of 250 frames and a GOP structure of I-P-P-..., where the first frame in the GOP is Intra-coded, and the rest of the frames are predicted frames. Note that bi-directionally predicted frames (B-frames) can also be processed with appropriate re-scaling of the motion vectors [2].

In a recent work investigating automatic determination of dominance in meetings [7], a total of 59 five-minute meeting segments from 11 sessions were each annotated by 3 annotators for perceived dominance rankings of the participants. We target the task of automatically classifying the most dominant person in each meeting. To better understand the strengths and weaknesses of our method, we look at three sets of meetings: (a) 34 meeting chunks where *every* annotator agreed on the most dominant person; (b) 23 meeting chunks where *only 2* annotators agreed on the most dominant person; and (c) 57 meeting chunks where *at least 2* annotators agreed on the most dominant person. In addition, we also investigated the task of automatically classifying the least dominant person, but only for 29 meeting chunks where *every* annotator agreed on the least dominant person.

We use the percentage of meetings where there was agreement between automatic classification and annotators as the performance metric. We also consider the computational time of feature extraction.

3.2.2 Baseline comparison

For baseline comparison, we also implement a similar scheme that works in the pixel domain. For each frame, we compute its optical flow using the previous temporal frame as reference. We then warp the previous frame into the current frame using the computed optical flow, and compute the absolute difference between the two; we will refer to this as the pixel-domain warped residual. We also classify each pixel as a skin-color pixel or not, using the same trained skin-color GMM model.

We then process optical flow in the same way as we do motion vector, and process pixel-domain warped residual the same way as we do residual coding bit-rate. Of course, the averaging is performed over the skin-color pixels (instead of skin-color blocks in the compressed domain scheme).

The key differences between the baseline and the compressed domain scheme are that (a) in the compressed domain scheme, the motion field computation is already part of the video compression process; and (b) there is no need to compute the pixel-domain warped residual in the compressed domain scheme, since the residual coding bit-rate can be simply read off from the video bitstream,

3.2.3 Results

Tables 1 through 4 summarizes the results for the various tasks. For all of the tasks, both pixel-domain and compressed-domain schemes were able to provide discrimination (random guess would yield only 25% accuracy). It is pleasantly surprising to note that the compressed-domain features not only do not perform worse than the pixel-domain features, but in fact out-performs it in certain operating conditions. We speculate that this could be due to the fact that compressed-domain features are not as noisy as the pixel-domain features.

Comparing the performance for the task of identifying the most dominant person with varying degrees of annotator agreement, we see that in all cases, performance decrease when there is less annotator agreement. This is to be expected, since meeting chunks in which not all annotators agree

Table 1: Performance for most dominant person with 3 annotators agreement

Features	Pixel-domain	Compressed-domain	% Degradation
Motion	64.7	70.6	-9.1
Residual	70.6	70.6	0.0
Combo	73.5	73.5	0.0

Table 2: Performance for most dominant person with 2 annotators agreement

Features	Pixel-domain	Compressed-domain	% Degradation
Motion	47.8	47.8	0.0
Residual	47.8	47.8	0.0
Combo	47.8	47.8	0.0

on the most dominant participant are intrinsically more ambiguous and hence more challenging. Further analysis of the results reveals that in most of the meeting segments where the features fail to find the most dominant person, either the most active (in terms of body movement) participant is not the most dominant, or the participant who is at the projection screen the largest proportion of the time is not the most dominant. (Recall that a participant detected to be not seated is assumed to be at the projection screen, and given a high activity label) Furthermore, due to the position of the cameras, a person who is presenting at the projection screen is also often visible in other camera views, for example seat 1 in Figure 3. Thus, if a person in seat 1 gets up to present, he might still be visible from camera 1, and hence estimated as being 'seated'. Thus, the high activity label would not be automatically given to that person. There are also some cases where two participants exhibit almost equal lengths of visual activity in a meeting segment and the motion activity feature is unable to find the more active of the two.

We also find that performance in the task of identifying the least dominant participant was not as good as that for finding the most dominant participant. It was interesting to note that the average reported annotator confidence for this task was slightly lower than for the most dominant task.

We measured the computation run-time of extracting features from the 4 close-up view cameras from all the meetings. Each close-up video has a spatial dimension of 352x288 pixels, and a frame rate of 25 fps. Our compressed domain feature extraction routines run on top of a version of Xvid[1], an open source video decoder for MPEG-4, which we have modified to suit our purposes. No particular care has been taken to optimize it. The pixel domain baseline scheme is implemented

[1]Available at `http://www.xvid.org/`

Table 3: Performance for most dominant person with at least 2 annotators agreement

Features	**Pixel-domain**	**Compressed-domain**	**% Degradation**
Motion	57.9	61.4	-6.0
Residual	61.4	61.4	0.0
Combo	63.2	63.2	0.0

Table 4: Performance for least dominant person with 3 annotators agreement

Features	**Pixel-domain**	**Compressed-domain**	**% Degradation**
Motion	48.3	58.6	-21.3
Residual	44.8	48.3	-7.8
Combo	48.3	48.3	0.0

using OpenCV[2], a popular open source computer vision library. Both schemes were evaluated on a Xeon 2.4 GHz Intel processor with 4 GB of RAM. Table 5 shows that the computational time reduction of 94.2% can be achieved by using a compressed-domain approach instead of a pixel-domain scheme, yet with no degradation in dominance classification performance.

Table 5: Comparison between pixel-domain and compressed domain

Performance	Pixel-domain	Compressed-domain	% Reduction
Runtime	39612 s	2129 s	94.6
Average dominance classification	56.3%	58.3%	-3.4
Storage size	281280 MB	2624 MB	99.1

4 Slide transition detection

4.1 Technical description

Given that the location of the projection screen is known, the problem of determining slide transitions is very similar to the problem of shot boundary detection in video analysis. In fact, there has been work on performing shot boundary detection in the compressed domain [13, 6]. Considering

[2]Available at `http://sourceforge.net/projects/opencvlibrary/`

that the AMI meeting compressed videos have long group-of-picture (GOP) size, which results in estimated DCT DC coefficients exhibiting large drift, we have decided that the residual coding bit-rate would be more suitable for the task of detecting slide transitions [6].

The residual coding bit-rate, which is extracted easily from the compressed domain, captures the temporal changes which are not accounted for by the block translational model. In the case of slide transitions, there is no translational motion, yet there are very distinct frame differences. This difference is highly correlated with the residual coding bit-rate. We thus use the number of blocks with a sufficiently high residual coding bit-rate, $N_r(t)$, as the signal of interest in detecting slide transitions. In particular, if $r(x, y, t)$ is the residual coding bit-rate of the (x, y)th block at frame t, then we have:

$$N_r(t) = \sum_{(x,y) \in \text{projection screen ROC}} \mathbb{I}\left[r(x, y, t) > \tau_r\right]$$

for some threshold τ_r.

One key difference between slide transition detection and shot boundary detection is that here, we also have to deal with the fact that there might be people walking in front of the projection screen. Therefore, the image area associated with the projection screen might exhibit large temporal differences due to human motion, but yet there is no slide transition. We find that this can be overcome with the use of $N_r(t)$ in our proposed compressed-domain scheme. First, we can account for as much translational motion as possible with the use of block translational motion to capture human movement. By looking at the residual, the difference between each block and its predictor in the previous temporal frame, we will only consider blocks which cannot be well predicted in the previous frame. Second, by looking for sharp peaks in $N_r(t)$, we can further eliminate cases where large temporal differences are caused by human motion. This is because when there is a person walking in front of the projection screen, there will be a large number of blocks with significant residue over an extended period of time. In contrast, in a slide transition, there are a large number of such blocks over only 1-2 frames. This is clearly illustrated in Figure 5.

We found that thresholding the number of blocks which has a sufficiently high residual coding bit-rate gives reasonable performance in detecting slide transitions. In addition, we also performed non-maximal suppression with a 2 second window length. Using these heuristics, we declare there to be a slide transition at frame s if:

$$N_r(s) \geq \alpha \tag{1}$$

$$N_r(s) \geq N_r(s + v) \quad \forall \quad v \in [-T/2, T/2] \tag{2}$$

$$N_r(s) \geq \beta + \tfrac{1}{T/2} \textstyle\sum_{v=1}^{T/2} N_r(s - v) \tag{3}$$

$$N_r(s) \geq \beta + \tfrac{1}{T/2} \textstyle\sum_{v=1}^{T/2} N_r(s + v) \tag{4}$$

α and β are thresholds that determine what value of $N_r(t)$ is significant for a slide transition, and how much change it must have from its temporal neighbors to be a slide transition, respectively.

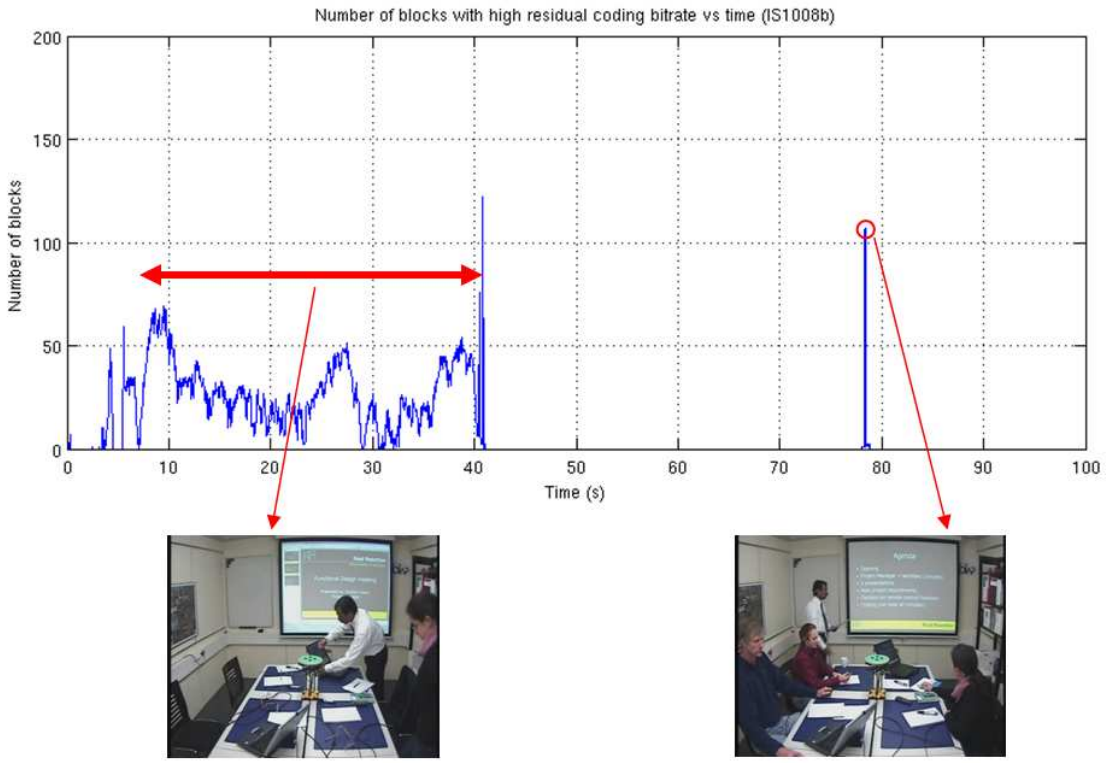


Figure 5: Plot of $N_r(t)$, the number of blocks with high residual coding bit-rate, in meeting session IS1008b. In the period around 10s-40s when a person is moving in front of the projection screen, note that while the number of blocks is moderately high, there is no sharp peak. On the other hand, a slide change at around 78s produces a very sharp peak.

T is the window size (in frames) we consider. In our experiments, we keep α and β the same, and vary them from 5 to 120, and set $T = 50$. We also use τ_r=48.

4.2 Experiments

4.2.1 Data

We carried out our evaluations on 12 meetings from the AMI meeting corpus [1], which contains 322 minutes (19343 seconds) of video data. The slide screen is only visible in the center camera view (see top row of Figure 4), so that is the only video stream we used in the experiments. To obtain ground truth for slide transitions, we look at the center view videos and record the times of slide transitions. There were a total of 401 slide transitions that we labeled, an average of about

1.2 slide transitions per minute. In our tests, we consider slide transitions to be correctly detected with a 0.5 second tolerance.

4.2.2 Baseline comparison

For baseline comparison, we also implement a similar scheme that works in the pixel domain. For each frame, we compute its optical flow using the previous temporal frame as reference. We then warp the previous frame into the current frame, and compute the absolute difference between the two. The result is then thresholded, and the number of pixels above the threshold is counted. The key differences between the baseline and the compressed domain scheme are that (a) in the compressed domain scheme, the motion field computation is already part of the video compression process; (b) there is no need to compute the residual in the compressed domain scheme, since the residual coding bit-rate can be simply read off from the video bitstream; and (c) the resolution of the difference is much finer in the pixel-domain scheme than the compressed domain scheme.

4.2.3 Results

The performance of these two schemes is shown as a ROC plot in Figure 6 below. The operating points are generated by varying the value of α and β as discussed earlier. Precision is the fraction of returned slide transitions that correspond to ground truth transitions, while recall is the fraction of ground truth transitions that were detected. The ROC plot shows us that neither schemes dominates the other in terms of slide transition detection performance. In fact, they seem to have relatively similar performance.

We also compute a single figure of merit, the balanced F-score, of the schemes. The balanced F-score is used in the information retrieval literature to measure how good a particular (precision,recall) operating point is, and is defined as:

$$F_1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$$

where Pr and Re are the precision and recall figures respectively. We find the maximum F_1 score over all the operating points for each scheme. The scores are listed in the Table 1. Surprisingly, there is no loss in performance going from the pixel domain baseline to the compressed domain scheme. In the compressed-domain scheme, the best F_1 score is obtained using $\alpha = \beta = 30$. Using leave-one-out full-fold cross-validation over the 12 meetings, we found that this choice of parameters consistently returns the best F_1 score.

The total computational time required for each scheme is also shown in Table 6. We also compute the speed-up factor, SUF, defined as:

$$SUF = \frac{SSD}{TPT}$$

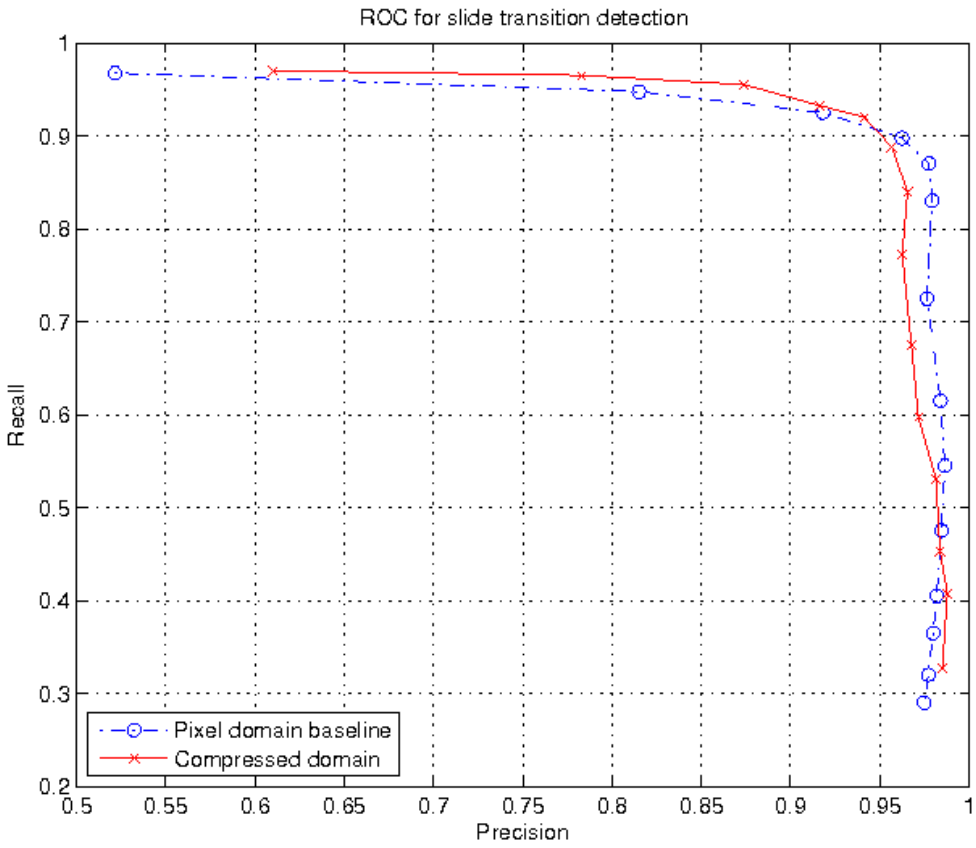


Figure 6: ROC plot for slide transition detection

where TPT is the total processing time and SSD is the source signal duration. Note that SUF has units of times-real-time, hence the larger SUF is, the faster processing is. Each video has a spatial dimension of 352x288 pixels, and runs at 25 fps. The 12 meeting videos have a total SSD of 19343 seconds. Our compressed domain feature extraction routines runs on top of Xvid[3], an open source video decoder for MPEG-4, but no particular care has been taken to optimize it. The pixel domain baseline scheme is implemented using OpenCV [4], a popular open source computer vision library. Both schemes were evaluated on a Xeon 2.4 GHz Intel processor with 4 GB of RAM.

As shown in table 6, we achieved an impressive SUF of 51.2 with the compressed domain scheme. In comparison, the baseline pixel domain scheme has a SUF of 3.7. With our compressed

[3]Available at `http://www.xvid.org/`

[4]Available at `http://sourceforge.net/projects/opencvlibrary/`

Table 6: Summary of performance figures for slide transition detection

Performance figures	**Pixel domain baseline**	**Compressed domain scheme**
F_1	0.93	0.93
Computation time (s)	5232	378
Speed-up Factor (times real-time)	3.7	51.2

domain scheme, we were able to achieve an impressive *93% decrease in run-time*, and still manage no loss in slide transition detection performance.

5 Conclusion

We have presented our work on extracting compressed domain features and combining them together to obtain activity level estimates. Our work indicates that for the task of dominance classification, compressed-domain video features were able to provide some discrimination. Furthermore, a computational time reduction of 94.2% can be achieved by using a compressed-domain approach instead of a pixel-domain scheme, but yet with no degradation in dominance classification performance. In the future, we can consider using the center camera view to obtain an estimate of the motion activity of a person who is presenting at a front of the meeting room. This way, we would not need to assign an arbitrary activity level.

We have also presented a simple and computationally efficient approach to detecting slide transitions in the compressed domain. The method makes use of residual coding bit-rate that can be easily extracted from the compressed video bit-stream without the need for full decoding. The experimental results on a subset of the AMI meeting corpus shows that the compressed domain scheme achieves a 93% decrease in run-time without any loss in slide transition detection performance with respect to the baseline pixel-domain scheme. In the future, it would be interesting to investigate how to determine the location of the projection screen automatically. A previous method used for detecting sub-windows in broadcast news videos might be a suitable starting point [14]. Furthermore, while we rely on heuristics in an unsupervised fashion to detect slide transitions, we can adopt a supervised approach using more powerful classifiers such as Support Vector Machines (SVM) to find such rules in a more principled fashion.

Acknowledgments

This research was funded in part by the US VACE program and the Singapore Agency for Science, Technology and Research (A*STAR). We thank Sileye Ba, Gerald Friedland, Daniel Gatica-Perez,

Yan Huang, Hayley Hung, Dinesh Jayagopi, Jean-Marc Odobez and Oriol Vinyals for their helpful discussions about the work. We are especially grateful to Hayley Hung and Dinesh Jayagopi for parsing the dominance annotations and sharing the results and analysis with us.

References

[1] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal *et al.*, "The AMI meeting corpus: A pre-announcement," *Proceedings of MLMI 2005, Edinburgh, UK*, 2005.

[2] M. T. Coimbra and M. Davies, "Approximating optical flow within the MPEG-2 compressed domain." *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 103–107, 2005.

[3] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[4] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, Apr 2001.

[5] L. Favalli, A. Mecocci, and F. Moschetti, "Object tracking for retrieval applications in MPEG-2," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 3, pp. 427–432, 2000.

[6] J. Feng, K.-T. Lo, and H. Mehrpour, "Scene change detecion for MPEG video sequence," in *Proc. of International Conference on Image Processing*, Sep 1996.

[7] H. Hung, D. Jayagopi, C. Yeo, G. Friedland, S. Ba, J. Odobez, K. Ramchandran, N. Mirghafori, and D. Gatica-Perez, "Using audio and video features to classify the most dominant person in a group meeting," *Proceedings of the 15th international conference on Multimedia*, pp. 835–838, 2007.

[8] V. Kobla, D. Doermann, and K. Lin, "Archiving, indexing and retrieval of video in the compressed domain," in *Proc. SPIE Conf. on Multimedia Storage and Archiving Systems*, vol. 2916, 1996, pp. 78–79.

[9] S. J. McKenna, S. Gong, and Y. Raja, "Modelling facial colour and identity with gaussian mixtures," *Pattern Recognition*, vol. 31, no. 12, pp. 1883–1892, 1998.

[10] J. Meng, Y. Juan, and S. Chang, "Scene change detection in a MPEG compressed video sequence," in *Proc. IS&T/SPIE Symposium*, vol. 2419, Feb 1995.

[11] H. Wang, A. Divakaran, A. Vetro, S. Chang, and H. Sun, "Survey of compressed-domain features used in audio-visual indexing and analysis," *Journal of Visual Communication and Image Representation*, vol. 14, no. 2, pp. 150–183, 2003.

[12] S. Wee, B. Shen, and J. Apostolopoulos, "Compressed-domain video processing," HP Labs, Tech. Rep., 2002.

[13] B. Yeo and B. Liu, "Rapid scene analysis on compressed video," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 5, no. 6, pp. 533–544, 1995.

[14] C. Yeo, Y. Zhu, Q. Sun, and S. Chang, "A Framework for Sub-Window Shot Detection," in *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*, 2005.

[15] Y. Zhong, H. Zhang, and A. Jain, "Automatic caption localization in compressed video," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 385–392, Apr 2000.