Signal processing meets computer vision: Overcoming challenges in wireless camera networks

Chuohao Yeo

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-72
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-72.html

May 20, 2009

Acknowledgement

I would like to acknowledge financial support from the Singapore Agency for Science, Technology and Research (A*STAR).

Signal processing meets computer vision: Overcoming challenges in wireless camera networks

by

Chuohao Yeo

B.S. (Massachusetts Institute of Technology) 2002
M.Eng. (Massachusetts Institute of Technology) 2002

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

and the Designated Emphasis

in

Communication, Computation and Statistics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Kannan Ramchandran, Chair
Professor Ruzena Bajcsy
Professor Martin Banks

Spring 2009

The dissertation of Chuohao Yeo is approved.

Professor Kannan Ramchandran, Chair	Date
Professor Ruzena Bajcsy	Date
Professor Martin Banks	Date

University of California, Berkeley

Signal processing meets computer vision: Overcoming challenges in wireless camera networks

by

Chuohao Yeo

Abstract

Signal processing meets computer vision: Overcoming challenges in wireless camera networks

by

Chuohao Yeo

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

and the Designated Emphasis in Communication, Computation and Statistics

University of California, Berkeley

Professor Kannan Ramchandran, Chair

The availability of cheap wireless sensor motes with imaging capability has made possible wireless camera networks that can be cheaply deployed for applications such as environment monitoring, surveillance and 3DTV. However, the gaping disconnect between high-bandwidth image sensors and a combination of low bandwidth channels, lossy communications and low processing capabilities makes realizing such applications especially challenging and forces tradeoffs between rate, reliability, performance and complexity. In this dissertation, we attempt to bridge that disconnect and address those tradeoffs in a meaningful fashion by drawing from both the Signal Processing and Computer Vision fields.

First, we focus our attention on compression and transmission of video from multiple camera sensors in a robust and distributed fashion over wireless packet erasure channels. We develop a low encoding complexity, low latency and error resilient video streaming solution based on distributed source coding that effectively uses information from all available camera views. We also investigate two correlation models for modeling correlation between camera views; one is based on view synthesis and another is based on epipolar geometry.

Second, we examine the problem of establishing visual correspondences between multiple cameras under rate-constraints. This is a critical step in performing many computer vision tasks such as extrinsic calibration and multi-view object recognition, yet the wireless medium requires that any information exchange should be done in a rate-efficient manner. We pose this as a rate-constrained distributed distance testing problem and propose two novel and complementary solutions: one using distributed source coding to exploit statistical correlation between descriptors of corresponding features and another using random projections to construct low bit-rate and distance-preserving descriptors.

Third, we study the problem of video analysis for multiple video streams generated by the deployment of camera networks, where methods that can run in real-time at a back-end server are needed. We develop computer vision techniques that exploit information which can be efficiently extracted from compressed videos. We consider their application to the task of detecting occurrences of human actions at specific times and locations and study the effects of video compression parameters on recognition performance. We also consider their use in the analysis of meetings to perform tasks such as slide change detection and dominance modeling of participants.

Professor Kannan Ramchandran
Dissertation Committee Chair

To the pursuit of truth

Contents

Contents ii

List of Figures vi

List of Tables xiii

Acknowledgements xiv

1 Introduction 1

I Video transmission in camera networks 7

2 Multi-view video transmission 8

2.1 Video coding background 9

2.2 Distributed source coding 12

2.3 Epipolar geometry 15

2.4 Disparity estimation and compensation 16

3 Robust and distributed video transmission for camera networks 18

3.1 Contributions 19

3.2 Related work 20

3.3 Correlation models for multiple camera networks 23

3.3.1 View synthesis based correlation model 23

3.3.2 Disparity based correlation model 24

3.4 Proposed approach 26

3.4.1 Encoding 27

3.4.2 Decoding 29

3.4.3 Decoder view synthesis search . . . 31
3.4.4 Decoder disparity search . . . 31
3.4.5 Discussion . . . 32
3.5 Experimental results . . . 33
3.5.1 Empirical validation of correlation models . . . 33
3.5.2 Comparison with simulcast schemes . . . 34
3.5.3 Effect of decoder search range in view synthesis search . . . 36
3.5.4 Effect of redundancy of cameras used in disparity search . . . 37
3.5.5 Effect of distance of cameras used in view synthesis search and disparity search . . . 38
3.6 Recapitulation . . . 39

II Establishing visual correspondences under rate-constraints 41

4 Rate considerations in computer vision tasks 42
4.1 Key role of visual correspondences . . . 44
4.2 Background . . . 46
4.2.1 Feature detector . . . 46
4.2.2 Feature descriptor . . . 47

5 Strategies for rate-constrained distributed distance testing 49
5.1 Contributions . . . 51
5.2 Related work . . . 52
5.3 Distributed source coding of descriptors . . . 53
5.3.1 Descriptor coefficients de-correlation . . . 53
5.3.2 Correlation model for descriptors of corresponding features . . . 53
5.3.3 Descriptor encoding and decoding . . . 55
5.3.4 Algorithmic summary . . . 56
5.4 Distance preserving hashes using binarized random projections . . . 57
5.4.1 Analysis of binarized random projections . . . 58
5.4.2 Numerical demonstration of Theorem 1 . . . 62
5.4.3 Choosing the number of hash bits . . . 63
5.4.4 Using linear codes to reduce rate . . . 65
5.4.5 Algorithmic summary . . . 66

5.4.6 Simulations . . . 67
5.5 Experimental Evaluations . . . 69
5.5.1 Setup . . . 69
5.5.2 Results . . . 72
5.5.3 Effect on homography estimation . . . 75
5.6 Recapitulation . . . 80

III Efficient video analysis for multiple camera streams 83

6 Compressed domain video processing 84
6.1 Compressed domain features . . . 85
6.1.1 DCT coefficients . . . 86
6.1.2 Motion vectors . . . 87
6.1.3 Residual coding bit-rate . . . 88
6.1.4 Skin-color blocks . . . 88

7 Activity recognition and organization 91
7.1 Related work . . . 92
7.2 Contributions . . . 93
7.3 Approach . . . 94
7.3.1 Notation . . . 94
7.3.2 Estimation of coarse optical flow . . . 95
7.3.3 Computation of frame-to-frame motion similarity . . . 96
7.3.4 Aggregation of frame-to-frame similarities . . . 97
7.3.5 Space-time localization . . . 99
7.3.6 Video action similarity score . . . 100
7.4 Experimental results . . . 101
7.4.1 Classification performance . . . 102
7.4.2 Performance gain from thresholding optical flow confidence map . . . 103
7.4.3 Effect of α variation on classification performance . . . 104
7.4.4 Localization performance . . . 104
7.4.5 Computational costs . . . 105
7.5 Effects of video encoding options . . . 106
7.5.1 GOP size and structure . . . 106

7.5.2 Quarter-pel accuracy motion estimation . . . 108
7.5.3 Block size in motion compensation . . . 109
7.6 Organization of activity videos . . . 110
7.6.1 Method . . . 111
7.6.2 Results . . . 114
7.7 Recapitulation . . . 114

8 Video analysis of meetings **118**
8.1 AMI meeting data . . . 119
8.2 Activity level estimation for dominance classification . . . 119
8.2.1 Approach . . . 121
8.2.2 Experiments . . . 122
8.3 Slide transition detection - a contextual cue for VFOA . . . 126
8.3.1 Approach . . . 127
8.3.2 Experiments . . . 129
8.4 Recapitulation . . . 133

9 Conclusions **134**

Bibliography **137**

List of Figures

1.1 The "Big-Eye" vision 2

1.2 Intersection of technical issues in wireless camera network in achieving the "Big-Eye" vision 3

2.1 Illustration of color space conversion. A color image can be decomposed into some luminance/chrominance space such as YCbCr. Furthermore, the chrominance components, Cb and Cr, are often sub-sampled before video compression. 10

2.2 Illustration of inter block encoding. The predictor for a source block, X_i, is found by searching in the reference image. The motion vector, v_i, indicates which predictor is to be used, while the difference or residual, $N_i = X_i - Y_i$, is transform coded. 11

2.3 Illustration of Group-Of-Pictures (GOP) structure. Each GOP starts with an intra frame (I-frame). A predictive frame (P-frame) uses only forward prediction, *i.e.* from a reference frame in the past. A bi-directionally predictive frame (B-frame) uses both forward prediction and backward prediction, *i.e.* from a reference frame in the future. 12

2.4 Source coding models. (a) DSC model, where side-information $\vec{Y}$ is available only at the decoder; (b) MCPC model, where the same side-information $\vec{Y}$ is available at both encoder and decoder. 13

2.5 Scalar Wyner-Ziv example. The set of quantized codewords is divided into 3 cosets, corresponding to the black, grey and white colored circles. The encoder quantizes X to $\hat{X}$ with a scalar quantizer of step size Δ and transmits the coset index, grey, that contains the quantized codeword. The decoder then reconstructs $\hat{X}$ by looking for the codeword in the grey coset that is closest to the side-information Y. 14

2.6 Epipolar geometry [62]. Cameras 1 and 2 have camera centers at C and C' respectively. An epipole is the projected image of a camera center in the other view; e and e' are the epipoles in this diagram. A point x seen in the image plane of camera 1 (assuming a projective camera) could be the image of any point along the ray connecting C and x, such as X_1, X_2 or X_3. This ray projects to the epipolar line l' in camera 2; l' represents the set of all possible point correspondences for x. If x was the image of X_2, then the corresponding image point in Camera 2 would be x'_2. 15

2.7 Parallel cameras setup. Cameras 1 and 2 have camera centers at C and C' respectively, whose displacement is parallel to the x-axis. The image planes are parallel to the x-y plane, and the camera axis is parallel to the z-axis. In this case, the epipoles lie at infinity. A point at (u_1, v) in the image plane of camera 1 would have a corresponding image point at (u_2, v) in camera 2. . . 16

3.1 Problem setup of distributed video transmission. Each camera views a portion of the scene. Due to energy and computational limitations on the camera sensor platform and bandwidth constraints of the channel, the encoders are not allowed to communicate with each other. Therefore, the encoders have to work independently without knowledge of what other cameras are viewing. Furthermore, they have to encode under complexity constraints. In real-time applications such as surveillance, tight latency constraints would have to be satisfied. Each encoder then transmits the coded bitstream over a wireless channel, which we model as a packet erasure channel which can have bursty errors. The decoder receives packets from each encoder over the erasure channel, and performs joint decoding to reconstruct the video frames for each camera view. 19

3.2 View synthesis based correlation model. The dark shaded block in frame t of camera 2 is the source block, $\vec{X}$. In the view synthesis based correlation model, $\vec{X}$ is correlated through an additive innovations process with a predictor block, $\vec{Y}_{VS}$, located within a small range centered on the co-located block in the predicted view (denoted by the shaded region). The predicted view is generated by first estimating the scene depth map of camera 2 from disparity estimation between frame t of cameras 1 and 3, and subsequently synthesizing an interpolated view for camera 2. Note that prediction is done at the *decoder* instead of the encoder. 24

3.3 Disparity search correlation model. The dark shaded block in frame t of camera 2 is the source block, $\vec{X}$. In the disparity search correlation model, $\vec{X}$ is correlated through an additive innovations process with a predictor block, $\vec{Y}_{DS}$, located along the epipolar line in a neighboring view. In contrast to the view synthesis based correlation model, no attempt is made to first estimate the scene geometry. Note that prediction is also done at the *decoder* instead of the encoder. 25

3.4 System block diagrams. 26

3.5 Illustrative example of multilevel coset code [131]. The constellation at the top represents the set of quantized codewords, while x denotes the quantization index; here we consider a 3 level binary decomposition. Each bit specifies the coset for the next level, e.g. x_0 specifies the coset on the left. For a finite constellation, this decomposition continues until only a single signal point is specified, as in this 3-bit example; for an infinite constellation, this decomposition can continue indefinitely. The distance between codewords in the coset indicated at each level is used to line up bitplanes across different coefficients. 28

3.6 Computation of coset index. In this work, we quantize the coefficients, and line up their bits such that each level has the same coset codeword squared distance to innovations noise variance ratio. In other words, if for the kth coefficient, l_k, δ_k and σ_k^2 are the bitplane number at a particular level, the quantization step size and the innovations noise variance respectively, then $(2^{l_k}\delta_k)^2/\sigma_k^2$ is the same for all k at that level. 29

3.7 Multi-view test video sequences used in experiments. 33

3.8 Innovations noise statistics of various correlation models. In this graph, we show the innovations noise statistics when different correlation models are used. These statistics were obtained from the "Vassar" multi-view video sequences using the correlation models described earlier, as well as using temporal motion search. 34

3.9 System performance over different error rates. 36

3.10 System performance over frames at 8% packet outage 37

3.11 Visual results of "Ballroom" sequence at 8% average packet outage. Note the obvious blocking artifacts in MJPEG, and the obvious signs of drift in both H.263+FEC and H.263+IR. PRISM-DS and PRISM-VS produced reconstructions that are most visually pleasing. 38

4.1 Problem setup for distributed visual correspondences. A typical wireless camera network would have many cameras observing the scene. In many computer vision applications such as camera calibration, object recognition, novel view rendering and scene understanding, establishing visual correspondences between camera views is a key step. We are interested in the problem within the dashed ellipse: cameras A and B observe the same scene and camera B sends information to camera A such that camera A can determine a list of visual correspondences between cameras A and B. The objective of our work is to find a way to efficiently transmit such information. 43

4.2 Visual correspondences example. In this example, we show two views taken of the same scene ("Graf" [91]). In each view, we have marked out 3 feature points and a line is drawn between each pair of corresponding features. A pair of visual correspondence tells us that the image points are of the same physical point in the scene. 45

4.3 Examples of regions found by Hessian-Affine region detector. The detected interest points are plotted as red crosses, while the estimated affine region are shown as yellow ellipses. 47

4.4 Computation of SIFT descriptor. Each detected image region is first warped, scaled and rotated to achieve affine invariance, scale invariance and rotational invariance. The image patch in the warped region is then divided into 4×4 tiles of pixels. In each tile, the computed image gradients are binned into an 8-bin histogram based on the orientation of the gradient and weighted by its magnitude. The 16 8-bin histograms are then stacked into a 128-dimensional vector which is normalized. 48

5.1 Rate-constrained distributed distance testing setup. 49

5.2 De-correlating SIFT descriptors. Here, we show the covariance matrix of the SIFT descriptors before and after applying PCA to de-correlate the coefficients. For better visualization, we show the logarithm of the absolute values of the covariance matrix. A brighter value thus indicates greater correlation between coefficients. It is clear from (a) that coefficients of the SIFT descriptor are highly correlated. After applying PCA however, most of the correlation between coefficients have been removed, as can be seen in (b). . 54

5.3 Variance of descriptor coefficients. We show the variance of each coefficient of both descriptors and innovations noise after applying de-correlation. Note that variance is shown in dB in this plot for a better visual comparison. The innovation noise variance is clearly much smaller than that of the original coefficients and this enables the rate savings. 55

5.4 Graphical illustration of proof for Lemma 1. A general multi-dimensional case can always be reduced to a 2-D case, in the plane formed by $\vec{D}_i^A$, $\vec{D}_j^B$, and the origin. The angle subtended by the rays from the origin to $\vec{D}_i^A$ and $\vec{D}_j^B$ in this plane can be found using simple trigonometry to be $\theta = 2\sin^{-1}(\delta/2)$. If a hyperplane orientation is chosen uniformly at random, then the probability of the hyperplane separating $\vec{D}_i^A$ and $\vec{D}_j^B$ is just θ/π. 60

5.5 Simulation results demonstrating Theorem 1. We show the scatter plot of Euclidean distance between a pair of descriptors and the estimated probability of a randomly chosen hyperplane separating the pair for a randomly chosen subset of pairs of features. The x-axis is the actual Euclidean distance between the pair of descriptors, and the y-axis is the estimated probability of a randomly chosen hyperplane separating the descriptors. The blue circles represent pairs in correspondence, while green crosses represent pairs not in correspondence. The theoretical relationship between the two quantities is plotted in red. Note the close adherence to the theoretical result, and the good separation between corresponding and non-corresponding pairs. 63

5.6 ROC over different bit rates for proposed scheme 67

5.7 Comparison of ROC for various schemes. We show here the ROC for RP-LDPC, RP and SQ when 256 bits are used per vector. RP-LDPC has the best retrieval performance, followed by RP and SQ. RP-LDPC uses 512 projections; we show for reference the performance of RP which also uses 512 projections, but requiring double the rate. These two schemes have very similar performances. 68

5.8 Comparison of maximum F1 scores for various schemes over different bitrates. We use the maximum F1 score to capture the best trade-off between recall and precision for each scheme and choice of parameters. The results here show that at all bit rates, RP-LDPC out-performs RP, since it is able to use the LDPC layer to reduce rate. On the other hand, at low rates, RP-LDPC out-performs SQ, while at high rates, SQ does better. This suggests that the choice of scheme depends on the rate regime. 69

5.9 F1 scores vs threshold used for RP and RP-LDPC using different number of projections. We show how the F1 scores vary as the threshold used varies. Empirically, the results suggest that picking the threshold as $\gamma_M = M\rho(\tau)$ will give the best recall/precision trade-off. 70

5.10 Test dataset [91]. The data used for our tests are shown above: (a) "Graf"; and (b) "Wall". In "Graf", the different views are of a mostly planar scene, while in "Wall", the views are obtained by rotating the camera about its center. In both cases, the views are related by a homography [83]. 71

5.11 Rate-Recall tradeoff. The above plots show how the average number of correctly retrieved correspondences (C_{correct}) varies with rate. The results for "Graf" are shown in (a) and (c); that of "Wall" are shown in (b) and (d). In (a) and (b), a threshold of $\tau = 0.195$ is used, while in (c) and (d), a threshold of $\tau = 0.437$ is used. 73

5.12 Rate-F_1 tradeoff. The above plots show how the F_1 score, a measure that takes into account both recall and precision performance, varies with rate. The results for "Graf" are shown in (a) and (c); that of "Wall" are shown in (b) and (d). In (a) and (b), a threshold of $\tau = 0.195$ is used, while in (c) and (d), a threshold of $\tau = 0.437$ is used. 74

5.13 Rate-Performance tradeoff with dimensionality reduction. We can also apply the baseline and DSC schemes in conjunction with dimensionality reduction. Here, we keep only the first 64 coefficients after PCA. The above plots show how the average number of correctly retrieved correspondences (C_{correct}) varies with rate. The results for "Graf" are shown in (a) and (c); that of "Wall" are shown in (b) and (d). In (a) and (b), we show the rate-recall tradeoff, while in (c) and (d), we show the rate-F_1 tradeoff. A threshold of $\tau = 0.195$ is used. 76

5.14 Measure of homography difference. A point, p_1, is picked at random in image I_1. Its corresponding point, p_2 in image I_2, is computed according to homography H. Similarly, its estimated corresponding point, $\hat{p_2}$ in image I_2, is computed according to estimated homography $\hat{H}$. The estimated corresponding point of p_2 in image I_2, $\hat{p_1}$ in image I_1, is also computed using $\hat{H}$. The distances d_1 and d_2 are computed between point pairs $(p_1, \hat{p_1})$ and $(p_2, \hat{p_2})$ respectively. The measure of homography difference between H and $\hat{H}$ is then computed as the average of distances d_1 and d_2 over a large number of points. 78

5.15 Effect of visual correspondences on homography estimation (using $\tau = 0.195$). 79

5.16 Effect of visual correspondences on homography estimation (using $\tau = 0.437$). 81

6.1 Example output from compressed domain feature extraction 85

6.2 Illustration of how DCT DC terms are updated using motion vectors and residual. The block to be reconstructed in frame t, shown with a solid fill, is predicted by a block in frame $t-1$, shown with stripes. In general, the predictor overlaps with 4 blocks, labeled as $\{a, b, c, d\}$ here. The update is computed by considering the amount of overlap with each block, with an additional correction term due to the residue. 86

7.1 Flow chart of action recognition and localization method. Optical flow in the query and test videos are first estimated from motion vector information. Next, frame-to-frame motion similarity is computed between all frames of the query and test videos. The motion similarities are then aggregated over a series of frames to enforce temporal consistency. To localize, these steps are repeated over all possible space-time locations. If an overall similarity score between the query and test videos is desired, a final step is performed with the confidence scores. 95

7.2 An example similarity matrix and the effects of applying aggregation. In these graphical representations, bright areas indicate a high value. (a) Aggregation kernel, (b) Similarity matrix before aggregation, (c) Similarity matrix after aggregation. Notice that the aggregated similarity matrix is less noisy than the original similarity matrix. 98

7.3 Illustration of space-time localization. The query video space-time patch is shifted over the entire space-time volume of the input video, and the similarity, $C(n, m, i)$ is computed for each space-time location. 100

7.4 Snap-shot of frames from action videos in database [114]. From left to right: boxing, handclapping, handwaving, running, jogging, walking. From top to bottom: outdoors environment, outdoors with different clothing environment, indoors environment. The subjects performing each action is the same across the different environments. 101

7.5 Action localization results. The highlighting in (d) and (e) denotes detection responses, with bright areas indicating high responses. (a) A frame from the query video, (b) An input video frame with one person walking, (c) An input video frame with two people walking, (d) Detection of one person walking, (e) Detection of two people walking. 105

7.6 Effect of varying GOP size on classification performance and compression performance. In general, increasing GOP size results in decreasing classification performance. Also, having no B frames in the GOP structure offers a better compression-classification trade-off. The fairly constant performance of the scheme using I-P-P-P-... with no texture propagation error indicates that the main source of performance degradation with increasing GOP size is due to propagation errors in computing block texture. 107

7.7 Effect of quarter-pel accuracy motion estimation on classification performance and compression performance. There seems to be no significant improvement in the compression-classification trade-off by using motion estimation with quarter-pel accuracy instead of half-pel accuracy. 109

7.8 Effect of using different block sizes in motion compensation on classification performance and compression performance. Using a smaller block size results in a better compression-classification trade-off, but this has to be weighed against the resulting increase in computational time. 110

7.9 A qualitative example of an action hierarchy for the activity video collection Φ_X, with associated exemplars for the subtree under each node, shown up to 6 clusters. This was generated using our proposed approach with NCNC as the action similarity measure and Ward linkage as the neighbor-joining criterion. The 6 clusters from left to right: Jogging, Walking, Running, Boxing, Handclapping, Handwaving. See Section 7.6.2 for further discussion. 111

7.10 Data flow for our proposed approach. Given a set of videos Φ_X and a user-defined space-time scale for actions, we compute pair-wise action similarity scores between all pairs of videos, and then convert them to symmetric action distances, D_{SIM}. We use D_{SIM} in hierarchical agglomerative clustering to produce a dendrogram, which is a binary hierarchical tree representing the videos, and the pair-wise cophenetic distances D_{COPH}, which are distances computed from the constructed dendrogram. The cophenetic correlation coefficient, Θ, is the correlation coefficient between D_{SIM} and D_{COPH}, and can be used to evaluate the goodness of the hierarchy. 112

8.1 Floor plan of smart meeting room 119

8.2 All available views in the data set. The top row shows the right, center and left camera views. The bottom row shows each of the 4 close-up views. . . . 120

8.3 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. 128

8.4 ROC plot for slide transition detection 131

List of Tables

3.1 PSNR (dB) with different search ranges (pixels) in PRISM-VS 37

3.2 PSNR (dB) with different number of cameras used in PRISM-DS 38

3.3 PSNR (dB) with distance of neighboring cameras used in PRISM-DS and PRISM-VS 39

7.1 Confusion matrix using NZMS 102

7.2 Confusion matrix using normalized correlation [40] 103

7.3 Classification performance with and without thresholding confidence map . 103

7.4 Classification performance with varying α 104

8.1 Performance for most dominant person with 3 annotators agreement 124

8.2 Performance for most dominant person with 2 annotators agreement 124

8.3 Performance for most dominant person with at least 2 annotators agreement 124

8.4 Performance for least dominant person with 3 annotators agreement 125

8.5 Comparison between pixel-domain and compressed domain 126

8.6 Summary of performance figures for slide transition detection 132

Acknowledgements

I have had the great fortune of receiving the help and support of many people and organizations to get to this point in my graduate studies.

First, I wish to acknowledge my heart-felt gratitude to my research advisor and dissertation committee chair, Prof. Kannan Ramchandran, for his support, patience and advice over the course of my graduate study. Without his vision and encouragement, this research would not have been possible.

I am also extremely grateful to my other thesis committee members, Prof. Ruzena Bajcsy and Prof. Martin Banks, for their invaluable feedback on my dissertation. In addition, I would like to thank Prof. Martin Wainwright and Prof. Bruno Olshausen for their time and feedback while serving on my Qualifying Examination committee.

Graduate study at Berkeley would not have been possible without financial support from the Singapore Agency for Science, Technology and Research (A*STAR). I am deeply grateful to them for taking a chance on me.

Along various points in my graduate career, I have relied on the help and advice of many members of the Berkeley BASiCS group, including Daniel Schonberg, Jiajun Wang, Vinod Prabhakaran, Hao Zhang and Stark Draper. I enjoyed working with them and have found discussions with them extremely invigorating. I would also like to thank Wei Wang, Mark Johnson, Krishnan Eswaran, Ben Wild, Animesh Kumar, Dan Hazen, Abhik Majumdar and Rohit Puri for helping make my stay at Berkeley an enjoyable one.

Parvez Ahammad played a key role early in my research career and taught me much about the computer vision problems we tried to solve. The many hours we spent on discussions (and some pool playing) have been extremely productive. I also appreciate the help and advice from Ethan Johnson, Phoebus Chen, Edgar Lobaton, Songhwai Oh, Posu Yan, Allen Yang and Prof. S Shankar Sastry on various aspects of camera mote platform and operation. Ruth Gjerde has been extremely helpful in taking care of all the administrative details of graduate study.

I also have had the opportunity of working and interacting with talented individuals like Gerald Friedland, Yan Huang, Nikki Mirghafori and Nelson Morgan from ICSI and Hayley Hung, Dinesh Jayagopi, Silèye Ba, Jean-Marc Odobez and Daniel Gatica-Perez from IDIAP. I am especially grateful for a fulfilling research experience at IDIAP.

Kaitian Peng has been my pillar of strength ever since she walked into my life. I am very thankful for her being there for me and I appreciate her unrelenting support, devotion and patience. I am deeply indebted to my parents, Yek Seng Yeo and Chor Eng Tan, and sister, Kaiwen Yeo, who have been a constant source of support and encouragement. Nothing can sufficiently express my gratitude to them.

Chapter 1

Introduction

The fusion of wireless sensor motes and cheap cameras has resulted in a flurry of research into problems and applications of wireless camera networks. In particular, it holds great potential as a system that can be cheaply deployed for applications such as environment monitoring, scene reconnaissance and 3DTV recording. Our vision is that of emulating a single expensive high-end video camera with the clever and opportunistic use of numerous cheap low-quality cameras that are wirelessly networked and connected to a high-end backhaul server connected to the base station, as illustrated in Figure 1.1. We call this the "Big-Eye" vision. This architecture allows us to leverage both the high density of cheap cameras as well as the availability of increasingly inexpensive backend processing power riding Moore's law.

The density of these cameras and the interaction of these cameras with the wireless network both provide opportunities and pose challenges: How should the distributed cameras reliably transmit their captured video data to the server over the lossy wireless channel while leveraging the possible overlap between their views? If the cameras are constantly perturbed or mobile, how should we keep them calibrated in a rate-efficient manner? Given the large number of cameras, how can we quickly analyze videos from so many camera streams?

More broadly, these questions fall into various categories of technical issues, as illustrated

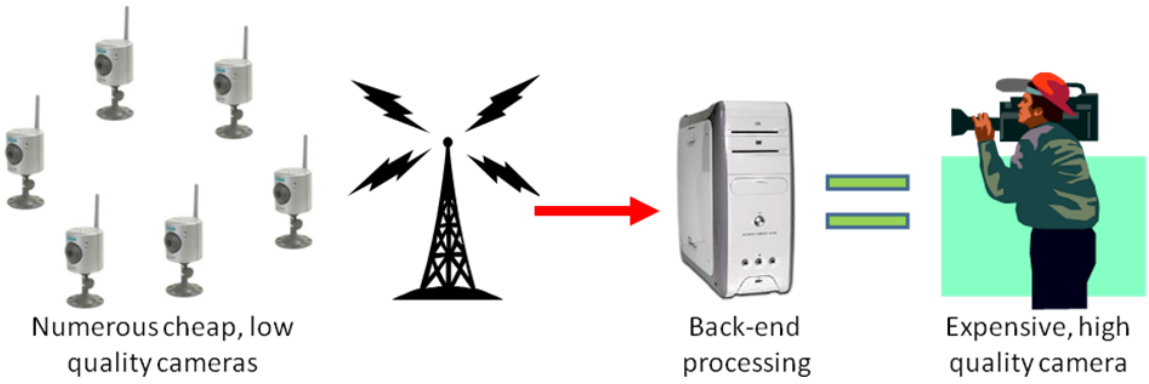


Figure 1.1. The "Big-Eye" vision

in Figure 1.2, that the "Big-Eye" vision requires to be considered. We aim to address each of these categories in this dissertation.

We address in Part I the problem of compressing and transmitting video from multiple camera sensors in a robust and distributed fashion over wireless packet erasure channels. This is a challenging problem that requires taking into account the error characteristics and bandwidth constraints of the wireless channel, the limitations of the sensor mote platform and the correlation between overlapping views of cameras. Furthermore, the real-time requirement of monitoring applications imposes stringent latency constraints on the system.

Current video codecs such as MPEG and H.264 are based on the motion compensated predictive coding (MCPC) framework, which can achieve high compression efficiency at the cost of high encoding complexity. However, in a wireless camera network, the MCPC framework is inadequate because each camera does not have access to other camera views. In addition, MCPC is not robust to errors in the video bitstream. Channel errors can cause loss of synchronization between the encoder and decoder that result in error propagation. This causes severe video quality degradation known as "drift". While tools such as forward error correction (FEC) codes or automatic-repeat-request (ARQ) protocols can be used to protect against channel errors, they do not satisfy stringent latency constraints.

We recognize that in cameras with overlapping views, there exists redundancy between views that can be exploited for robustness. Motivated by theoretical results in distributed

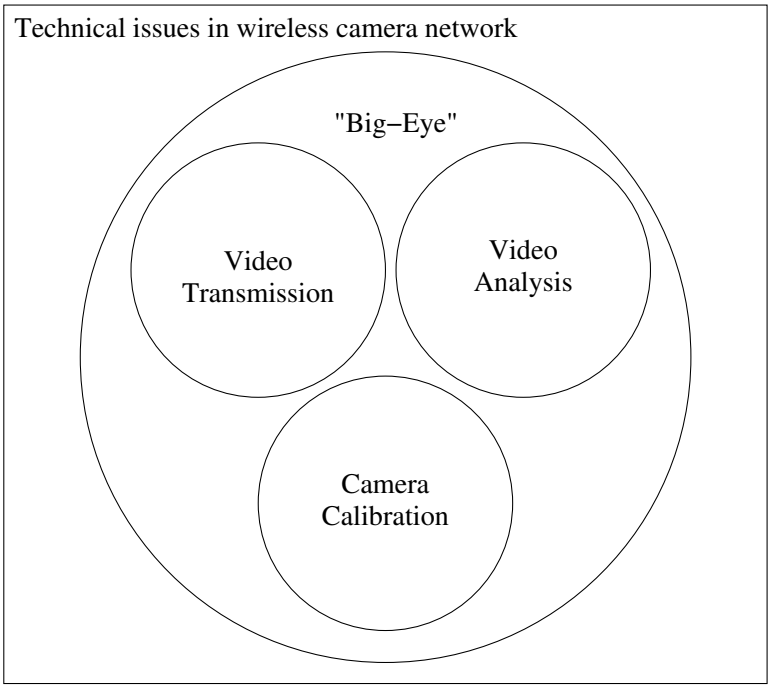


Figure 1.2. Intersection of technical issues in wireless camera network in achieving the "Big-Eye" vision

source coding [120, 137], we take the approach that information found in other camera views that have been correctly reconstructed can be used to help decode blocks that have been affected by erroneous transmission. Our main contributions are:

- We propose two correlation models that can be used to capture the statistical correlation of corresponding blocks in neighboring views. One is based on view synthesis and the other is based on epipolar geometry,
- We show how the correlation models can be used in a distributed source coding framework to exploit inter-camera redundancy effectively for robustness when transmitting videos from multiple cameras over lossy transmission channels. Furthermore, encoding is done independently, so there is no need for the wireless cameras to exchange information with each other. We present simulation results that demonstrate the robustness of our system compared to baseline methods such as using FEC and random intra-refresh.

Up to now, we have assumed that the external calibration parameters, *i.e.* location

and pose, of the cameras in the network are known. In a fixed camera network, such an assumption might seem reasonable because calibration can be performed with a one-time cost at the beginning of deployment. In a mobile camera network, this is certainly not the case. Furthermore, even if the camera network is designed to be fixed, environmental factors such as wind can perturb the location and pose of deployed cameras. Thus, if a *bandwidth-constrained* network of wireless cameras needs to be calibrated continuously, the communications cost of exchanging information for performing calibration needs to be accounted for.

In Part II of this dissertation, we address the above concern by investigating the problem of establishing visual correspondences between multiple cameras in a rate-efficient manner. Visual correspondences are not only used for calibration but also for other vision tasks such as multi-view object recognition. Our main contributions are:

- We propose a solution based on distributed source coding that exploits the statistical correlation between descriptors of corresponding features for rate savings. We show through simulations that our proposed method yields significant rate savings in practice.

- We propose a complementary solution based on constructing distance-preserving hashes using binarized random projections. We analyze its distance-preserving properties and verify it through simulations. We then show how this can be applied effectively in conjunction with distributed source coding by using linear codes.

- We describe a general class of problems that we term "rate-constrained distributed distance testing" that includes not just establishing visual correspondences but also video hashing for video file synchronization across remote users. While we have proposed practical methods for performing such tests, we believe that a theoretical study of this problem would be a fruitful area of future research and would shed light on how close to optimal our methods are.

Finally, in Part III, we consider the problem of efficient video processing for camera networks. Given the expected influx of large amounts of video data from deployment of

multiple cameras, we need efficient methods for analyzing video that can run in real-time. Our approach to efficient video processing is to reuse video processing already performed for video compression to minimize the amount of computation needed to compute features for video analysis. This technique is known in the literature as compressed domain processing [20].

Concretely, we first consider the task of human action recognition and localization. In surveillance applications, it is useful to perform rudimentary action recognition that can alert human operators when an activity of interest occurs. Current methods for action recognition in the pixel domain are slow [117], require difficult segmentation [40], or involve expensive computation of local features [114]. While computationally efficient, related work in performing action recognition in the compressed domain have shortcomings such as requiring segmentation of body parts [100] or the inability to localize actions [11, 10]. Our proposed method uses motion vector information to capture the salient and appearance-invariant features of actions. We then turn to analysis of meetings where an instrumented meeting room would have many camera views in order to capture movements of all participants. We consider the tasks of dominance estimation and slide change detection and propose features for these tasks that can be efficiently computed from compressed videos. Our main contributions are:

- We show how motion vectors computed for video compression can also be used for action recognition and localization in videos and propose a novel similarity measure, *Non-Zero Motion block Similarity* (NZMS), for this purpose. We show experimentally that this has a performance comparable to state-of-the-art vision techniques at a fraction of their computational costs. We also give insight into how video compression parameters affect recognition performance. Finally, we show how NZMS can be used to perform unsupervised organization of a collection of videos based on the similarity of actions in those videos.

- We show how data such as residual coding bitrate, motion vectors and transform coefficients in compressed videos can be used to perform meeting analysis tasks such

as slide change detection and dominance modeling. We show experimentally that such features achieve performance that matches or is superior to their pixel-domain counterparts with much lower computational cost.

We conclude in Chapter 9 and discuss possible future research directions and extensions to the work presented in this dissertation.

Part I

Video transmission in camera networks

Chapter 2

Multi-view video transmission

The practical deployment of wireless sensor networks [84] and the availability of small CMOS camera chips has held out the possibility of populating the world with networked wireless video camera sensors. Such a setup can be used for a wide variety of applications, ranging from surveillance to entertainment. For instance, a system endowed with multiple views can improve tracking performance by being able to disambiguate the effects of occlusion [34]. Free viewpoint TV and 3-D TV [87, 74] and tele-immersive applications can also benefit from the easy deployment of dense networks of wireless cameras.

The applications described above, as well as the "Big-Eye" vision proposed in Chapter 1, need to rely on a robust infrastructure which is capable of delivering accurate video streams from the wireless cameras. Unfortunately, this is a rather challenging task. The wireless environment poses bandwidth constraints and channel loss, while the sensor mote platform has limited processing capability and limited battery life [84]. In applications such as real-time surveillance, there are very stringent end-to-end delay requirements, which impose tight latency constraints on the system.

Traditional hybrid video encoders such as MPEGx and H.26x, while achieving high compression, have high encoder complexity due in part to the use of motion compensation, and are susceptible to prediction mismatch, or "drift", in the presence of data loss. Such drift causes visually disturbing artifacts and is made particularly worse in wireless channels

where packet losses are bursty and more frequent than in wired networks. On the other hand, Motion JPEG[1] (MJPEG) is computationally light-weight and robust to channel loss, but has poor compression performance. Recent work on low-complexity video codecs using joint source-channel coding ideas based on distributed source coding (DSC) principles provide a promising middle-ground between the robustness and low encoding complexity of MJPEG and the compression efficiency of full-search motion-compensated MPEG/H.26x [106, 1].

We will discuss our proposed approach to robust and distributed multi-view video compression in the next chapter (Chapter 3). In the rest of this chapter, we review the relevant background topics that are important in presenting our approach: video coding, distributed source coding (DSC), epipolar geometry and disparity estimation and compensation.

2.1 Video coding background

Hybrid video encoding technology, such as MPEGx and H.26x, uses a combination of motion compensation and transform coding and has been very successful at improving the rate-distortion (RD) performance of video compression [28, 136]. In this section, we describe the key features of such video encoding technology.

The input to a video encoder is typically raw video data that is a sequence of image frames. Each image frame commonly consists of either 3 matrices of tri-stimulus color pixels, *e.g.* RGB or XYZ, or 1 matrix of luminance pixels and 2 matrices of chrominance pixels, *e.g.* YUV or YCbCr. Since the tri-stimulus color components are highly correlated with each other, RGB video data is typically first converted to some luminance/chrominance space to decorrelate them [78]. Chrominance sub-sampling is also often applied to reduce the amount of data, as illustrated in Figure 2.1, because the human visual system is more sensitive to luminance than chrominance components [95] and high-frequency image details exist mainly in the luminance component [78]. For simplicity in the remaining discussion, we assume that only luminance data is to be compressed, although much of what we describe is applicable to chrominance data; see for example [28, 136] for further details.

[1] MJPEG simply codes each frame independently with JPEG without exploiting any temporal redundancy.

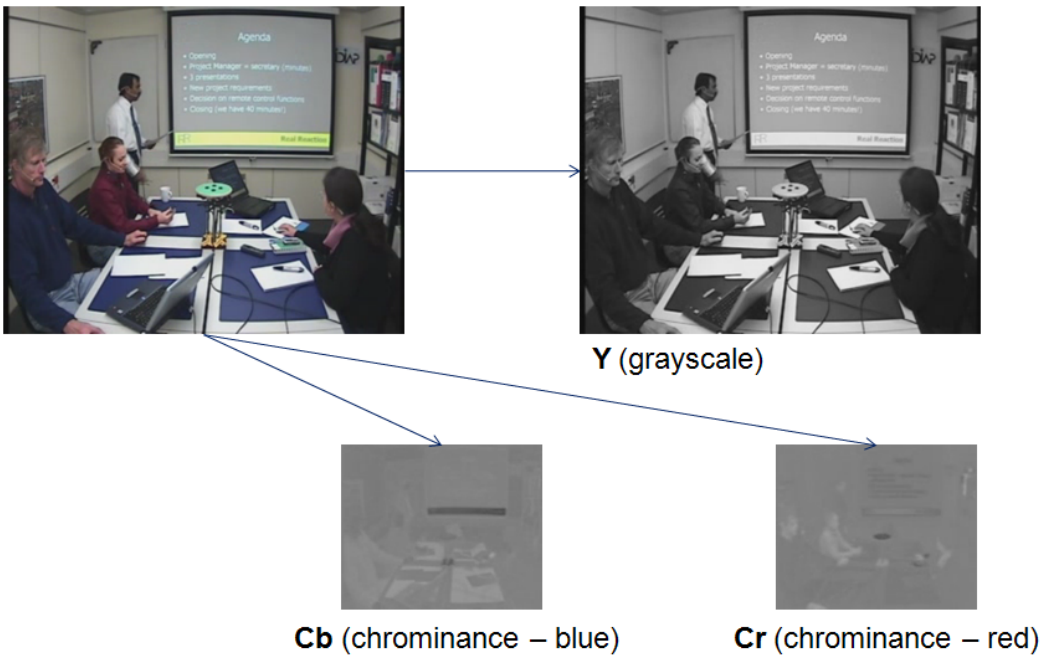


Figure 2.1. Illustration of color space conversion. A color image can be decomposed into some luminance/chrominance space such as YCbCr. Furthermore, the chrominance components, Cb and Cr, are often sub-sampled before video compression.

Uncompressed luminance data typically has both high spatial redundancy within a frame and high temporal redundancy between frames. Video encoding aims to reduce both sources of redundancy to achieve compression. Spatial redundancy is usually reduced through transform coding, *e.g.* Discrete Cosine Transform (DCT), while temporal redundancy is usually reduced using motion compensation. In the encoding process, each frame is divided into non-overlapping square blocks of equal size. Each block can be encoded in one of the following ways:

- **Intra block encoding.** Intra block encoding aims to remove spatial redundancy between pixels in the block. This is done by applying a 2-D transform such as the DCT to approximately decorrelate the pixels. Each transform coefficient is then quantized, zig-zag scanned, and run-length and entropy coded. The choice of quantization parameter allows one to make a rate-distortion tradeoff.

- **Inter block encoding.** Inter block encoding aims to reduce both temporal redundancy and spatial redundancy. As shown in Figure 2.2, for the ith block to be encoded,

X_i, motion search is first performed in a reference frame, typically a temporally neighboring frame that has already been encoded, for the best predictor block, Y_i. The residual, $N_i = X_i - Y_i$, which collected from all the blocks in the source frame forms the displaced frame difference (DFD), is then encoded using transform encoding as described above. The motion vector, $\vec{v}_i$, is also entropy coded and stored.

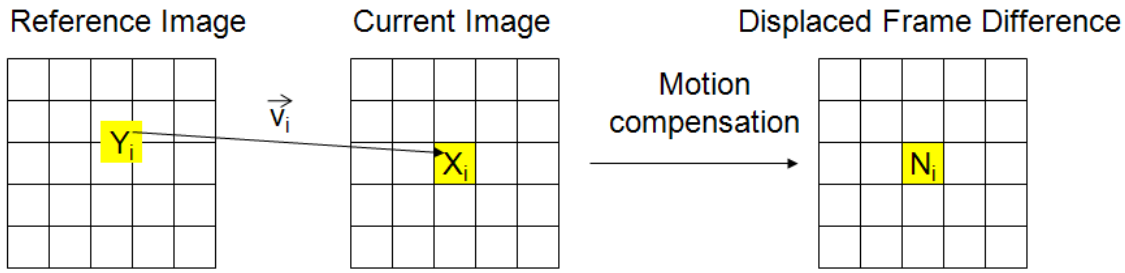


Figure 2.2. Illustration of inter block encoding. The predictor for a source block, X_i, is found by searching in the reference image. The motion vector, v_i, indicates which predictor is to be used, while the difference or residual, $N_i = X_i - Y_i$, is transform coded.

For random access and robustness reasons, video data is typically grouped into multiple Group of Pictures (GOP), illustrated in Figure 2.3. The first frame of each GOP is an intra frame (I-frame), in which every block in the frame is intra-coded. The remaining frames can either be a predictive frame (P-frame) or a bi-directionally predictive frame (B-frame), in which most blocks in the frame are inter-coded with the remaining blocks being intra-coded. They differ in how source blocks are predicted: an inter P-block uses only one predictor from a frame in the past (forward prediction) while an inter B-block uses two predictors, one in a frame in the past (forward prediction) and one in a frame in the future (backward prediction)[2]. Generally, a smaller GOP size leads to better random access capability and greater robustness to errors, while a larger GOP size leads to better compression efficiency.

[2]In H.264, this has been extended such that any combination of two predictors can be used [136]

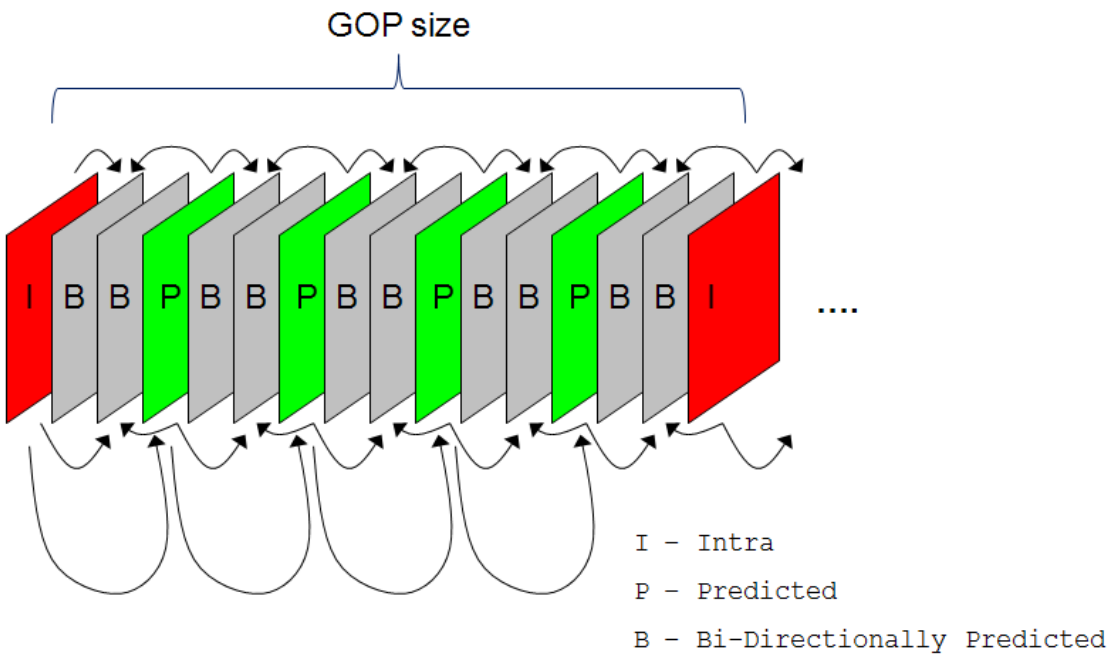
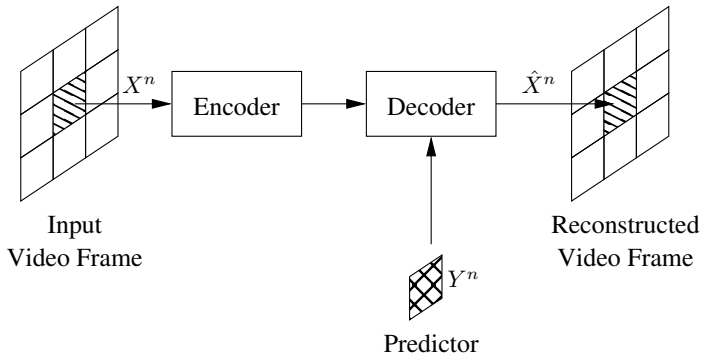


Figure 2.3. Illustration of Group-Of-Pictures (GOP) structure. Each GOP starts with an intra frame (I-frame). A predictive frame (P-frame) uses only forward prediction, *i.e.* from a reference frame in the past. A bi-directionally predictive frame (B-frame) uses both forward prediction and backward prediction, *i.e.* from a reference frame in the future.

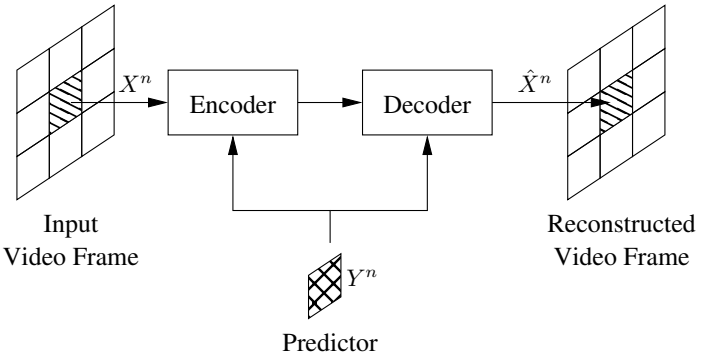
2.2 Distributed source coding

To enable distributed coding of physically separated sources, we rely on and are inspired by both information-theoretic and practical results in a particular setup of distributed source coding: *lossy source coding with side-information*, illustrated in Figure 2.4(a). In a video coding context, X^n is the current video block to be encoded, and Y^n is the best predictor for X^n from reconstructions of reference frames such as temporally neighboring frames or spatially neighboring camera views. $\{X_i, Y_i\}_{i=1}^n$ are i.i.d. with known joint probability distribution $p(x, y)$, and $\hat{X}^n$ is the decoder reconstruction of X^n. The objective is to recover $\hat{X}^n$ to within distortion D for some per-letter distortion $d(x, \hat{x})$. Note that in the set-up, Y^n is only available at the decoder. In contrast, in Figure 2.4(b), the side-information Y^n is available at both encoder and decoder; this setup is exemplified by conventional motion-

compensated predictive coding (MCPC) schemes such as MPEGx and H.26x (see [28] for example).



(a) Distributed source coding (DSC) model



(b) Motion-compensated predictive coding (MCPC) model

Figure 2.4. Source coding models. (a) DSC model, where side-information $\vec{Y}$ is available only at the decoder; (b) MCPC model, where the same side-information $\vec{Y}$ is available at both encoder and decoder.

In the case when X and Y are jointly Gaussian and the distortion measure is the mean square error (MSE), it can be shown using the Wyner-Ziv theorem [137] that the rate-distortion performance of coding X^n is the same whether or not Y^n is available at the encoder. This is also true when $X^n = Y^n + N^n$, with N^n being i.i.d. Gaussian and the distortion measure being the MSE [102]. However, in general, there is a small loss in rate-distortion performance, termed the *Wyner-Ziv rate loss*, when correlated side-information is not available at the encoder [153].

While the above results are non-constructive and asymptotic in nature, a practical approach was proposed by Pradhan and Ramchandran [104] and subsequently applied to

video coding [106, 1]. We will illustrate some of the main terms and concepts in lossy source coding with side-information using the following scalar example of source coding with side-information [70].

Suppose X is a real-valued random variable that the encoder wishes to transmit to the decoder, with a maximum distortion of $\frac{\Delta}{2}$, *i.e.* $|X - \hat{X}| < \frac{\Delta}{2}$, where $\hat{X}$ is the estimate computed by the decoder. Furthermore, the decoder has access to side-information Y, where X and Y are correlated such that $|X - Y| < \Delta$. To satisfy the distortion constraint, the encoder quantizes X to $\hat{X}$ using a uniform scalar quantizer with step size Δ. Instead of sending the identity of the quantized codeword, $\hat{X}$, the encoder divides all possible quantized codewords into 3 *cosets*, as shown in Figure 2.5, and transmits the *coset index* of the coset containing the codeword, thus requiring only $\log_2 3$ bits. The decoder has access to the received coset index (or syndrome) as well as the side-information Y. Due to the correlation structure of (X, Y) and the quantizer used, we have, by using the triangular inequality, that $|\hat{X} - Y| < \frac{3\Delta}{2}$. Thus, the decoder only has to look for the closest codeword to Y in the coset indicated by the coset index since there is only one codeword in each coset that is within $\pm\frac{3\Delta}{2}$ of Y.

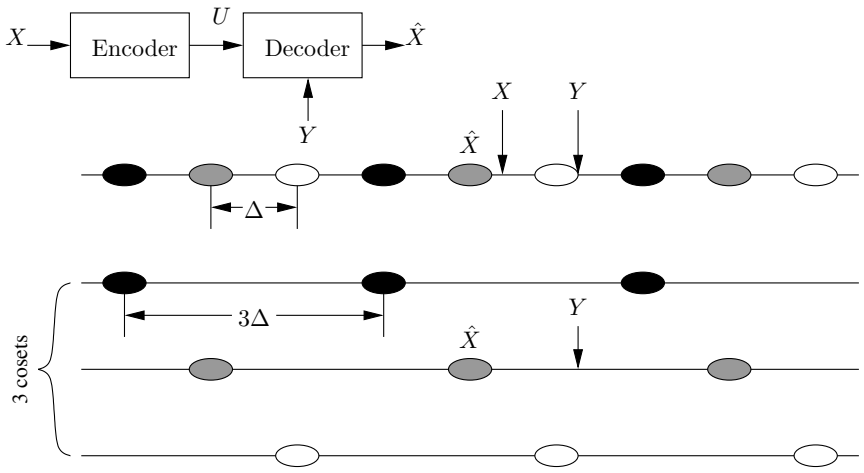


Figure 2.5. Scalar Wyner-Ziv example. The set of quantized codewords is divided into 3 cosets, corresponding to the black, grey and white colored circles. The encoder quantizes X to $\hat{X}$ with a scalar quantizer of step size Δ and transmits the coset index, grey, that contains the quantized codeword. The decoder then reconstructs $\hat{X}$ by looking for the codeword in the grey coset that is closest to the side-information Y.

In the above example, as well as in general distributed source coding, two pieces of

information are needed to design the quantizer and the coset. First, the targeted distortion constraint between the source and decoder reconstruction needs to be specified. Second, the correlation structure (model) between X and Y needs to be known or estimated.

2.3 Epipolar geometry

The geometric constraint of a single point imaged in two views, using the projective camera (pinhole camera) model, is governed by epipolar geometry [62]. As shown in Figure 2.6, given an image point in the first view, the corresponding point in the second view can be found on the epipolar line if it is not occluded in the second view. Furthermore, the epipolar line can be computed from the position of the point in the first view and the projection matrices of the cameras, and is independent of the scene geometry. Therefore, if the cameras are assumed to be stationary, then it is only necessary to calibrate the cameras once at the beginning to obtain the fundamental matrix necessary for computation of the epipolar line between the two views [62].

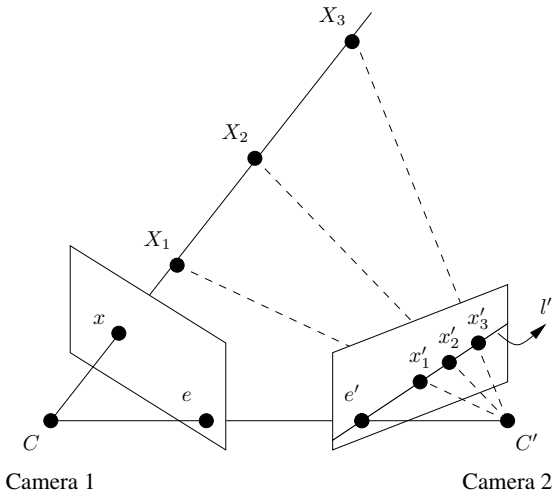


Figure 2.6. Epipolar geometry [62]. Cameras 1 and 2 have camera centers at C and C' respectively. An epipole is the projected image of a camera center in the other view; e and e' are the epipoles in this diagram. A point x seen in the image plane of camera 1 (assuming a projective camera) could be the image of any point along the ray connecting C and x, such as X_1, X_2 or X_3. This ray projects to the epipolar line l' in camera 2; l' represents the set of all possible point correspondences for x. If x was the image of X_2, then the corresponding image point in Camera 2 would be x'_2.

Figure 2.7 illustrates the parallel cameras setup that we use for ease of discussion in Chapter 3. There is no loss of generality since image rectification[3] can be applied as a pre-processing step [62]. In this setup, the key observation is that given a point in one camera view, the epipolar line in the other rectified view is just the same scan-line.

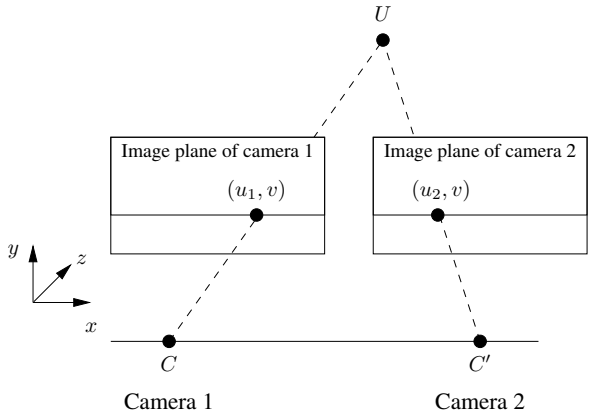


Figure 2.7. Parallel cameras setup. Cameras 1 and 2 have camera centers at C and C' respectively, whose displacement is parallel to the x-axis. The image planes are parallel to the x-y plane, and the camera axis is parallel to the z-axis. In this case, the epipoles lie at infinity. A point at (u_1, v) in the image plane of camera 1 would have a corresponding image point at (u_2, v) in camera 2.

Epipolar geometry allows us to constrain a search for corresponding points in a different view to a 1-D search. If there are constraints on the minimum and maximum scene depth, then the search can be further constrained to reduce decoder complexity [55].

2.4 Disparity estimation and compensation

Disparity refers to the shift in horizontal locations of a corresponding point imaged in two rectified views; in Figure 2.7, the disparity of point U imaged in camera 1 with respect to camera 2 is simply $u_1 - u_2$. The depth of a point is inversely proportional to its disparity; the smaller the disparity, the farther away the point. Disparity estimation, or stereo correspondence, is a problem in computer vision that is concerned with computing a dense disparity map from two rectified stereo images under known camera geometry. From

[3]Image rectification is a warping technique used in computer vision to project two or more views onto a common image plane, given the external (and intrinsic) calibration parameters of the set of cameras. By doing this, for any pixel in one rectified view, its corresponding pixel lies on the same scanline in other rectified views, hence simplifying stereo correspondence.

the disparity map, a relative depth map representing scene geometry can be computed. Among other things, depth maps can be used for disparity compensation as discussed later. Further discussion of disparity estimation is outside the scope of this dissertation, but a good survey and taxonomy of disparity estimation algorithms has been presented by Scharstein and Szeliski [113].

In computer graphics, both view synthesis and image based rendering involve solving the problem of using a set of captured images from calibrated cameras to generate an image that would have been captured by a camera at a desired viewpoint. For the synthesized image to be reasonably accurate, the desired viewpoint should be near the capturing viewpoints. If the camera views are rectified, then one can also use disparity compensation as a view synthesis approach to predict a desired view. There, assuming Lambertian surfaces[4] in the scene and no occlusions, a pixel is predicted by looking up its corresponding pixel, which is indicated by its disparity, in a captured view. For example, in the parallel camera setup shown in Figure 2.7, the pixel at (u_2, v) in camera 2 can be predicted through disparity compensation by the pixel at (u_1, v) in camera 1.

[4]Given an illumination source, the radiance of a Lambertian surface is the same from all viewing angles.

Chapter 3

Robust and distributed video transmission for camera networks

There is significant inter-view correlation between cameras with overlapping views. While exploiting this correlation for either compression or robustness is straight-forward in a centralized approach where all video streams are available at one encoder, such is not the case for a distributed wireless camera network where cameras are expected to work independently and where inter-camera communication is expensive. We seek solutions that can utilize inter-view correlation between cameras with overlapping views, even if the cameras are unable to communicate freely with each other. Specifically, as shown in Figure 3.1, our goal is to compress and transmit video frames from multiple wireless camera sensors in a robust and distributed fashion. Each encoder should have high compression performance to minimize transmission costs and low computational complexity to preserve battery life. We model the wireless links by packet erasure channels; the transmission scheme should be robust to packet losses. In addition, the overall system should have low end-to-end delay to satisfy tight latency constraints.

We do assume that the cameras have been calibrated and are fixed. However, even if this is not the case, there exist solutions for continuously calibrating cameras in a distributed and rate-efficient manner [26, 140], as we will discuss in Part II of this dissertation.

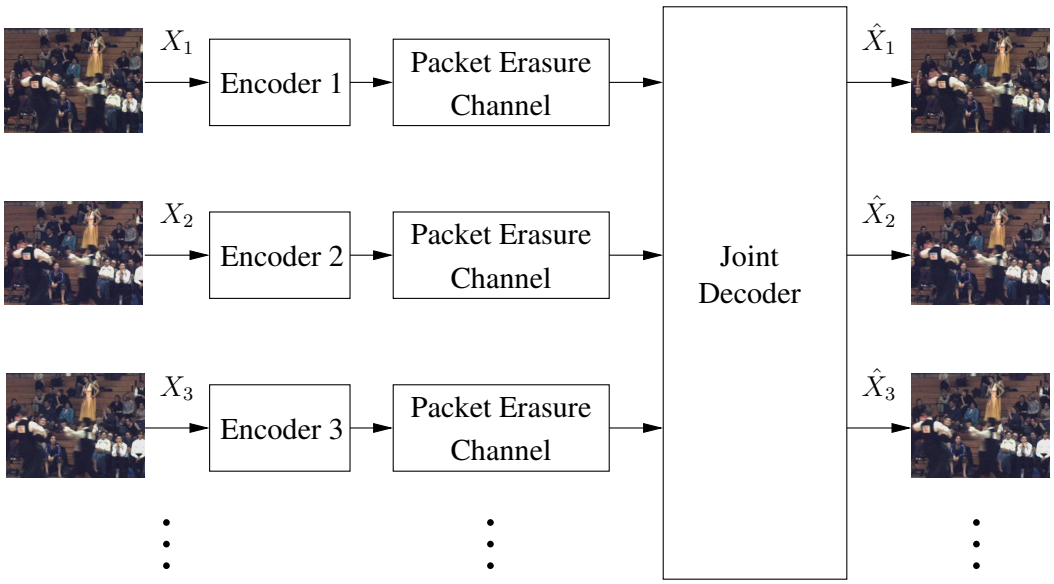


Figure 3.1. Problem setup of distributed video transmission. Each camera views a portion of the scene. Due to energy and computational limitations on the camera sensor platform and bandwidth constraints of the channel, the encoders are not allowed to communicate with each other. Therefore, the encoders have to work independently without knowledge of what other cameras are viewing. Furthermore, they have to encode under complexity constraints. In real-time applications such as surveillance, tight latency constraints would have to be satisfied. Each encoder then transmits the coded bitstream over a wireless channel, which we model as a packet erasure channel which can have bursty errors. The decoder receives packets from each encoder over the erasure channel, and performs joint decoding to reconstruct the video frames for each camera view.

The work presented in this chapter is joint work with Kannan Ramchandran, and has been presented in part in [146, 149, 148]. We also like to acknowledge the advice and assistance given by Jiajun Wang.

3.1 Contributions

Recognizing that cameras with overlapping views provide redundancy, our key contribution is the systematic development of principled approaches that can effectively harness this redundancy for robust video transmission with completely distributed encoders. In doing so, we jointly address two main issues by drawing from results in information theory and computer vision. First, the encoder at each camera does not have access to views observed from other cameras. Therefore, we propose a *distributed source coding* approach

based on the PRISM framework [105] that is able to make use of side-information (predictors) from other camera views for decoding even if that is not available at the encoder. Our approach also does not require explicit correspondence information between camera views to be known at the time of encoding. However, such a coding approach needs models which capture both the statistical relationships and the geometrical constraints between multiple camera views. In this work, we describe two such models. The first model requires two other camera views at the decoder and uses *disparity estimation* and *view interpolation* to generate side-information for decoding. The second model requires only one other camera view at the decoder and uses *epipolar constraints* to generate side-information for decoding. In our simulations, we show that with these two models, our proposed approaches are able to effectively exploit the redundancy in overlapping views for robustness.

The rest of this chapter is organized as follows. Section 3.2 discusses related work in multi-view video coding. The two different multi-camera correlation models we use for this work are presented in Section 3.3, and we describe the encoding and decoding procedures in Section 3.4. Experimental results on the performance of our proposed and baseline schemes using a realistic wireless channel simulator are presented in Section 3.5. Finally, concluding remarks and directions for future work are given in Section 3.6.

3.2 Related work

There has been work establishing significant compression gains in using block-based disparity compensation over independent coding for multi-view *image* compression [8, 119] and multi-view *video* compression [18]. Sophisticated methods using pixel-based disparity compensation and view synthesis for prediction have also been proposed [121, 85]. More recently, there has also been great amount of research interest in multi-view video coding methods due to ongoing standardization efforts [96, 47]. However, these approaches require knowledge of scene depth information at the encoder and hence assume joint encoding of the multi-view video.

In a wireless camera network, a more realistic approach is to perform compression in a

distributed fashion (as shown in Figure 3.1), in which encoders have no or low-bandwidth communications with each other. Wagner *et al.* proposed a scheme where compression gains are realized by down-sampling the image at each camera [132]. Reconstruction is then performed via application of a super-resolution procedure on the received images. This procedure requires each camera to perform a scene-dependent image warping before down-sampling the captured image for transmission. Hence, the scheme could be used if the depth map of the scene remains static; this may not be a suitable assumption in a surveillance scenario in which many objects of interest are moving about.

Zhu *et al.* divide cameras in a large array into conventional cameras and "Wyner-Ziv" cameras [158] . The image at each conventional cameras is coded independently using JPEG2000. The decoder first uses view synthesis to generate a prediction of the image at each "Wyner-Ziv" camera and then requests parity bits from each "Wyner-Ziv" camera, using the predicted image as side information to decode. Varodayan *et al.* proposed an interesting approach where disparity is learned in an unsupervised fashion and recovered jointly with the encoded image [130].

Gehrig and Dragotti model each scan-line of stereo images as piecewise polynomials [55]. Each camera encoder sends the locations of end-points of the polynomial pieces, as well as parameters of complementary polynomial pieces. The decoder attempts to match the discontinuities between the views and reconstructs each scan-line according to the polynomial pieces. Their method assumes no occlusions and that the same sequence of polynomial pieces will be generated for each scan-line. This is somewhat fragile since the reconstructed scan-line will be erroneous if the polynomials of each view were not correctly matched. This approach has recently been extended to the 2-D case, by using quad-tree decompositions [56]. In a similar geometric approach, Tosic and Froosard proposed using a sparse over-complete decomposition of images captured by omni-directional cameras and applying DSC to the location and shape parameters of the decomposed atoms [129].

Song *et al.* address distributed compression of multi-view video by first implementing a distributed algorithm that tracks block correspondences between two cameras views [122] . The corresponding blocks of the cameras are then encoded using distributed source coding.

Their experiments are performed with the *actual* block coefficients instead of the residual after temporal motion compensation; therefore, their implementation does not fully realize the potential for compression gains from exploiting both temporal and inter-view correlation. Using a similar approach, Yang *et al.* predict motion vectors for a stereo view which can be used as side-information when applying DSC to the motion vectors [138].

Others have used Wyner-Ziv video coding [57] with an appropriate fusion of side-information generated by both temporal and view interpolation [59, 99]. Inter-view correlation is modeled by the use of an affine scene model which is suitable in videos with simple scene geometry and with low temporal motion. Flierl and Girod exploit temporal correlation by using a motion-compensated lifted wavelet transform [45] and exploit inter-view correlation by applying disparity compensation to transform coefficients using disparity maps estimated from previously decoded frames [46].

The above works focus on compression performance by *removing* redundancies present in overlapping camera views and at the same time assume that lossless transmission of video data from individual cameras is possible. While they are interesting in their own right, here we take the view that packet drops are to be expected in wireless camera networks and choose to focus on robustness in video compression and transmission by *exploiting* redundancies present in overlapping camera views. Forward Error Correction (FEC) and Automatic-Repeat-Request (ARQ) are two popular approaches for providing protection, but they may not be suitable or adequate in a multi-camera video network operating in real-time and under channel loss. FEC requires long block lengths to work well and this could introduce intolerable latencies in a real-time surveillance scenario. While an ARQ system is an effective and simple way of dealing with erasure channels, it may require an arbitrary number of round-trips and this would not be suitable for systems with tight latency constraints. Furthermore, the decoder would have to scale its feedback responses with the number of camera sensors and this is not practical if the number of cameras grows large.

3.3 Correlation models for multiple camera networks

We now describe two alternative multi-view correlation models. The video frame to be encoded is divided into non-overlapping blocks of 8×8 pixels. We denote the DCT coefficients of the block to be encoded, the predictor block and the innovations process by $\vec{X}$, $\vec{Y}$ and $\vec{N}$ respectively. Furthermore, we assume that the scene contains only Lambertian surfaces and that cameras have identical photometric responses[1]. We defer discussion of the empirical performance of these models to Section 3.5.

3.3.1 View synthesis based correlation model

View synthesis using dense disparity maps has been used in the past for both joint and distributed multi-view video compression [85, 158]. In the view synthesis based correlation model, we use view synthesis to generate predictors for decoding when an estimate of scene depth can be obtained. As illustrated in Figure 3.2, if the current frame at camera 2 is to be encoded, and two neighboring views, corresponding to cameras 1 and 3, are available, it is possible to use those views to synthesize the frame at camera 2. To compensate for small errors in calibration, disparity estimation or view interpolation, the predictor block for $\vec{X}$, $\vec{Y}_{VS}$, is allowed to be one of the blocks from a small area around its location in the synthesized frame. The correlation model is thus $\vec{X} = \vec{Y}_{VS} + \vec{N}_{VS}$, where $\vec{N}_{VS}$ is the prediction error between $\vec{X}$ and $\vec{Y}_{VS}$ and is independent of $\vec{Y}_{VS}$. Compared to past distributed encoding approaches which simply use the synthesized frame without allowing for slight perturbations in the location of the predictor [158, 138, 46] *i.e.* a candidate set of size 1, this correlation model can choose a predictor from a candidate set that is a superset of the former and thus the prediction error between the source block and its predictor would be no larger than the former. This can be observed in our experimental results (see Section 3.5.3).

The view synthesis based correlation model can also be extended to non-rectified views

[1]This can be accounted for by calibrating the photometric responses of the cameras in advance.

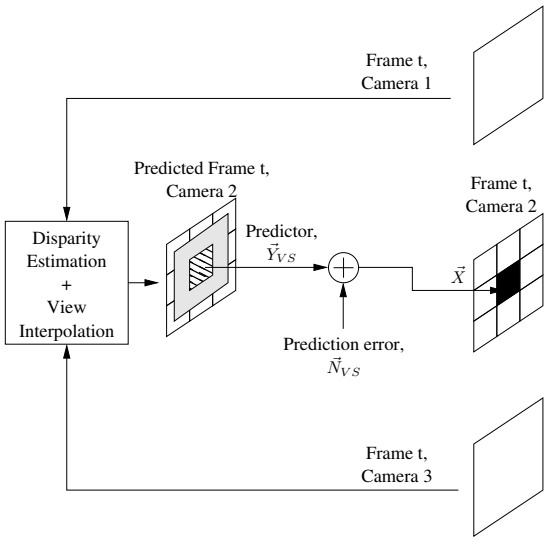


Figure 3.2. View synthesis based correlation model. The dark shaded block in frame t of camera 2 is the source block, $\vec{X}$. In the view synthesis based correlation model, $\vec{X}$ is correlated through an additive innovations process with a predictor block, $\vec{Y}_{VS}$, located within a small range centered on the co-located block in the predicted view (denoted by the shaded region). The predicted view is generated by first estimating the scene depth map of camera 2 from disparity estimation between frame t of cameras 1 and 3, and subsequently synthesizing an interpolated view for camera 2. Note that prediction is done at the *decoder* instead of the encoder.

by performing dense pixel correspondence instead of disparity estimation, and then applying view interpolation with the computed correspondence map.

3.3.2 Disparity based correlation model

While the view synthesis based correlation model is conceptually simple, there are some practical challenges. First, dense disparity estimation is a difficult problem in computer vision and remains an area of active research (for a recent survey and discussion, see [113]). The difficulty lies in the tension between locality, which requires a small image neighborhood, and robustness, which requires a large image neighborhood. Forced to compute dense correspondence, disparity estimation often returns estimated depth maps which are noisy. Furthermore, occlusions are also not easily handled. Second, view interpolation requires accurate disparity estimates and camera calibration for a high quality synthesis [25]. If disparity estimates are inaccurate, then the predictors will be degraded. Third, view synthesis search requires at least two other camera views. To circumvent these challenges, we consider

an alternative correlation model that can directly use one other camera view without any further processing.

Illustrated in Figure 3.3, the disparity based correlation model is based on exploiting the geometric constraints imposed by epipolar geometry on images captured by different cameras of the same scene. Assume that each 8x8 source block, $\vec{X}$, has negligible depth variation. Recall that depth is inversely proportional to disparity; hence, if the depths of all points within each 8x8 block are equal, then their disparities are also equal, and hence the corresponding block is simply a block on the epipolar line. Thus, the predictor block for $\vec{X}$, $\vec{Y}_{DS}$, is allowed to be one of the blocks along the epipolar line in other available camera views. To cope with small errors in camera calibration and image rectification, we also allow predictors to be a little above and below the epipolar line. The correlation model is then $\vec{X} = \vec{Y}_{DS} + \vec{N}_{DS}$, where $\vec{N}_{DS}$ is the prediction error between $\vec{X}$ and $\vec{Y}_{DS}$ and is independent of $\vec{Y}_{DS}$.

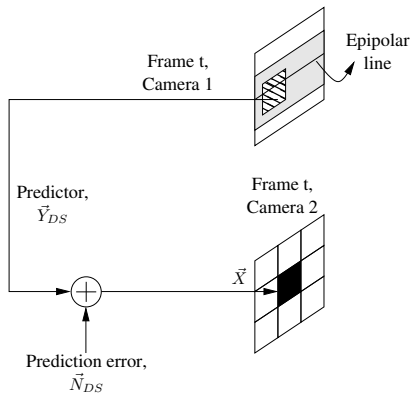


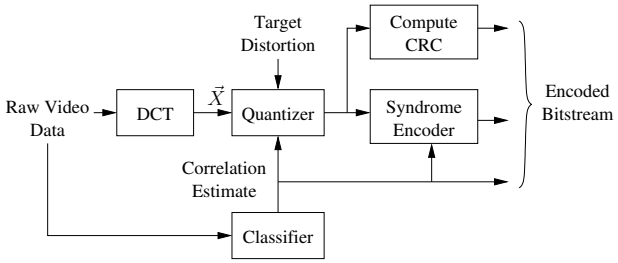
Figure 3.3. Disparity search correlation model. The dark shaded block in frame t of camera 2 is the source block, $\vec{X}$. In the disparity search correlation model, $\vec{X}$ is correlated through an additive innovations process with a predictor block, $\vec{Y}_{DS}$, located along the epipolar line in a neighboring view. In contrast to the view synthesis based correlation model, no attempt is made to first estimate the scene geometry. Note that prediction is also done at the *decoder* instead of the encoder.

This correlation model can be extended to the case when camera views are not rectified by making a constant depth assumption on the surface imaged by the 8x8 block, and then performing an appropriate re-sampling in the other camera view (i.e. camera 1 in Figure 3.3).

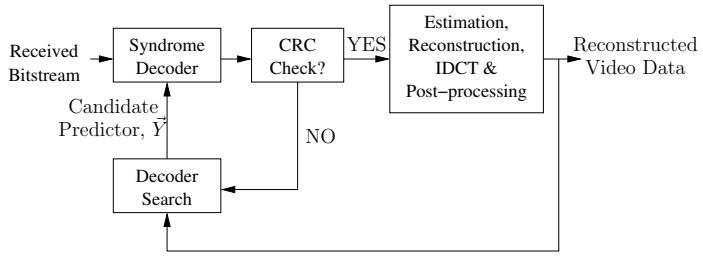
3.4 Proposed approach

The proposed approach is inspired in part by the PRISM framework, which is developed using the principles of distributed source coding [105]. Unlike conventional MCPC schemes (as modeled in Figure 2.4(b)) such as MPEGx and H.26x, in a DSC based video codec (as modeled in Figure 2.4(a)), there is no need for coding and decoding to depend on encoders and decoders having *"deterministic"* predictors [105, 57]. Therefore, there is an inherent robustness and flexibility in DSC based video codecs, since successful decoding can be performed as long as the decoder is able to find a suitable predictor to use as side-information. The encoder does not need to know block correspondence or the locations of other cameras; instead, the decoder performs motion search or correspondence search during the decoding process. Due to the use of DSC (rather than differential coding), our approach is also robust to transmission errors since drift can be mitigated even if the encoder and decoder do not have the same exact predictors.

The block diagrams of the encoder and decoder are shown in Figure 3.4. We will describe encoding and decoding in the following sub-sections.



(a) Encoder block diagram



(b) Decoder block diagram

Figure 3.4. System block diagrams.

3.4.1 Encoding

Figure 3.4(a) shows the block diagram of the encoder for the inter-frames [105]; intra-frames are coded conventionally using a H.263+ intra frame encoder (see, for example, the discussion in Section 2.1 of Chapter 6). Each video frame is divided into non-overlapping 8x8 blocks. The 2-D DCT is first applied to each source block to obtain its DCT coefficients, which are then arranged into a vector of 64 coefficients, denoted by $\vec{X}$, using a zig-zag scan. Next, the coefficients are quantized with a scalar quantizer with a step size chosen to achieve a user-specified reconstruction quality.

We assume a correlation structure of $\vec{X} = \vec{Y} + \vec{N}$, where $\vec{N}$ denotes the uncorrelated innovations process. Also, denote the innovations noise variance for the kth coefficient by σ_k^2 and the quantization step size to achieve a target distortion by δ. A suitable channel code that is matched to the statistics of $\vec{N}$ is used to partition the quantized codeword space of $\vec{X}$ into cosets [104]. The coset index of the quantized $\vec{X}$ is then transmitted. Note that no motion estimation is performed at the encoder; instead, a simple classifier based on the prediction error between $\vec{X}$ and its co-located block in the reference frame is used to determine the block mode and hence estimate the statistics of $\vec{N}$ and the appropriate channel code parameters to use. To aid decoding, the encoder also transmits a 16-bit cyclic redundancy check (CRC) hash of the quantized coefficients to the decoder. Thus, the bitstream for each block consists of the block mode, the syndrome bits, and the 16-bit CRC.

In this work, we use a multilevel block modulation code [131, 48, 80] as the channel code, with appropriately chosen binary BCH codes[2] as binary component codes for each level. More specifically, we use the quantization lattice as our signal set and consider a binary level decomposition of the quantization codeword index, as shown in Figure 3.5. Note that this decomposition is in reverse bit order. The bit at each level signals which coset to use at the next level. As shown in Figure 3.6, bitplanes of coefficients in a block are arranged such that at each level, the ratio of the squared distance between codewords in a coset

[2]BCH codes are used instead of more powerful codes such as LDPC or Turbo codes due to the short block lengths used.

at that level to the innovations noise variance is the same, thus ensuring that each level has the same signal-to-noise ratio (SNR). In the general multilevel block modulation code framework, bits from each level across coefficients can be coded with a binary BCH code. In our implementation, as illustrated in Figure 3.6, we use a non-trivial binary BCH code only for a single level. For bits above that level, we use a zero rate code, i.e. the bits are sent as is (corresponding to the parity bits of the code), since they are not very predictable from the side-information due to low SNR at those levels. For bits below that level, we use a rate-1 code, i.e. no bits are sent (corresponding to no parity bits), since given the lower-order bits and the side-information, they can be inferred with high probability due to high SNR at those levels. The coset index is then the concatenation of parity (syndrome) bits of the binary component code of each level.

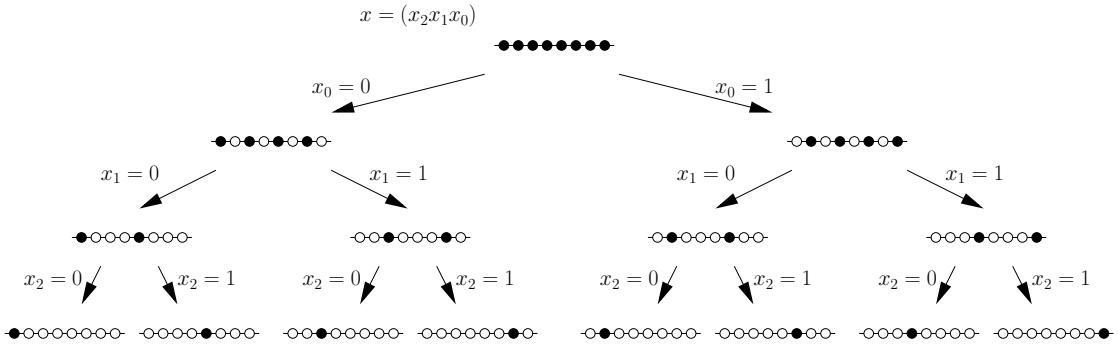


Figure 3.5. Illustrative example of multilevel coset code [131]. The constellation at the top represents the set of quantized codewords, while x denotes the quantization index; here we consider a 3 level binary decomposition. Each bit specifies the coset for the next level, e.g. x_0 specifies the coset on the left. For a finite constellation, this decomposition continues until only a single signal point is specified, as in this 3-bit example; for an infinite constellation, this decomposition can continue indefinitely. The distance between codewords in the coset indicated at each level is used to line up bitplanes across different coefficients.

More concretely, the number of levels to be coded for the kth coefficient is $(L_k + 1)$, where [48]:

$$L_k = \left\lceil \frac{1}{2} \log_2 \left(2\pi e \alpha^2 \frac{\sigma_k^2}{\delta^2} \right) \right\rceil \tag{3.1}$$

and α is a user parameter that determines the probability of decoding error on the highest uncoded bitplane, i.e. the $(L_k - 1)$th bitplane. In our implementation, we choose $\alpha^2 = 6.4$ dB. Note that in general, L_k is different for each coefficient k. For each coefficient, bitplanes 0 through $L_k - 1$ are sent uncoded. Bits from bitplane L_k of all coefficients are concatenated

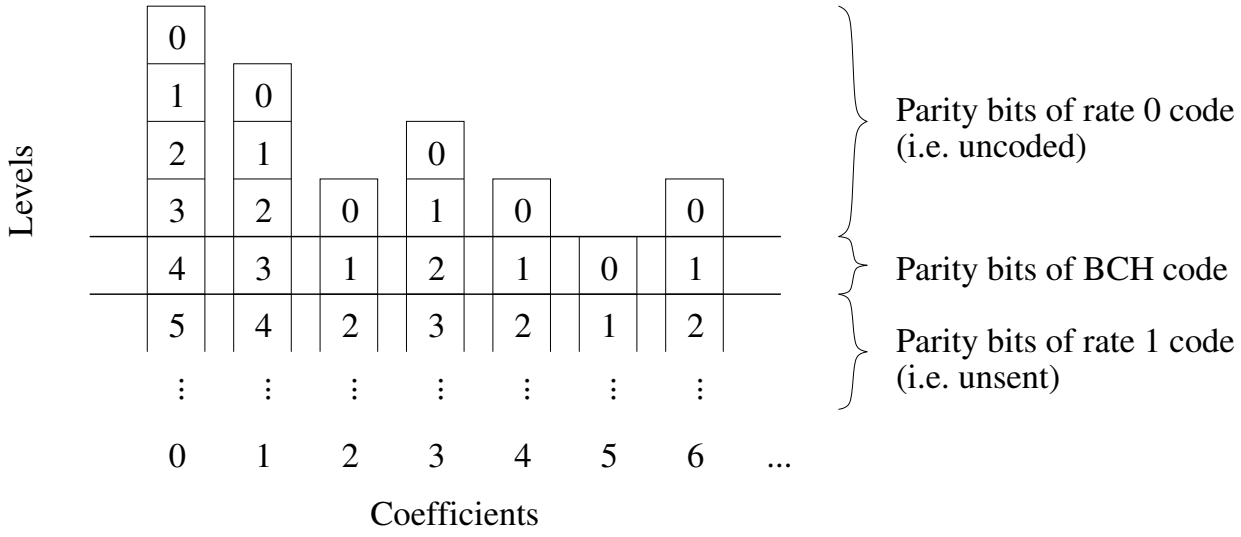


Figure 3.6. Computation of coset index. In this work, we quantize the coefficients, and line up their bits such that each level has the same coset codeword squared distance to innovations noise variance ratio. In other words, if for the kth coefficient, l_k, δ_k and σ_k^2 are the bitplane number at a particular level, the quantization step size and the innovations noise variance respectively, then $(2^{l_k}\delta_k)^2/\sigma_k^2$ is the same for all k at that level.

into a bit vector and its parity bits computed with an appropriate BCH code. The resulting syndrome bits are thus the uncoded bits from each coefficient and the computed BCH parity bits.

3.4.2 Decoding

The decoder operation is shown in Figure 3.4(b) [105]. For each block, the decoder receives the syndrome and CRC of the quantized coefficients. Unlike the classical Wyner-Ziv setup shown in Figure 2.4(a), the video decoder has many potential side-information candidates since it has not been indicated which is the "right" predictor to use as side-information for decoding. Decoder search is performed to generate a list of candidate predictors from previously decoded reference frames. In theory, the decoder should choose a predictor that is jointly typical with the quantized $\vec{X}$ [67]; this involves an exhaustive search through all combinations of possible predictors at the decoder and codewords in the coset indicated by the received syndrome to find a pair that is jointly typical. In

practice, due to practical complexity constraints, the decoder instead performs approximate maximum-likelihood syndrome decoding using multistage soft-decision decoding, where the soft-decision decoding is performed via ordered statistic decoding [80].

In multistage decoding, component BCH codes of each bitplane level are decoded one at a time, starting from the lowest level; the decoded information from each level is used in subsequent decoding of the next bitplane level [131]. In our implementation, the received uncoded bits corresponding to the first L_k bitplanes of each coefficient need not be further decoded. The received parity bits corresponding to the L_kth bitplane of all coefficients are decoded using ordered statistic decoding with soft-information provided by the candidate predictor (side-information) [80]. Together, all $(L_k + 1)$ decoded bits of the kth coefficient will signal a coset of quantized codewords. We then choose the quantized codeword in this coset that is closest to its side-information.

As a further verification step, the received CRC is used to check decoding success: if the CRC of the quantized coefficients of the decoded block checks out with the received hash, decoding is assumed to be successful; if not, the next candidate predictor is used and the process repeated.

The reconstruction of $\vec{X}$ is obtained by computing the minimum mean square error (MMSE) estimate of $\vec{X}$ given the decoded quantized coefficients, the candidate predictor and the assumed correlation model. Following that, the inverse DCT is performed to obtain the reconstructed pixels.

Typically, decoder search is performed by doing a temporal motion search to half-pel accuracy in a limited search range (± 15 pixels in both directions) of the co-located position in the reconstructed previous frame [105]. If predictors in the temporal reference frame are lost or badly corrupted due to packet drops, but the block to be reconstructed is visible from other camera views, then it might be possible to construct a good predictor from those views. As it turns out, we can use the correlation models described in Section 3.3 to guide this search for predictors.

3.4.3 Decoder view synthesis search

To perform decoder view synthesis search, we use the view synthesis based correlation model described in Section 3.3.1. As illustrated in Figure 3.2, suppose we want to decode a block from camera 2. We first use a relatively fast and simple stereo correspondence algorithm based on dynamic programming [49] to generate a dense disparity map using the current decoded frames from neighboring cameras (cameras 1 and 3 in Figure 3.2). Together with the decoded images from the neighboring cameras, the estimated disparity map is then given as input to a view interpolation routine [25] to synthesize a prediction of the current frame from camera 2. After performing disparity estimation and view interpolation, the decoder would sample blocks from a small area around its location in the synthesized frame to use as side-information in decoding the received syndrome.

If the camera calibration parameters are perfectly known, stereo correspondence is accurately performed and view interpolation is done correctly, then the co-located block in the synthesized frame would be the best side-information for syndrome decoding. However, in practice, this is not the case and each of the above steps introduces errors in the synthesized view. Therefore the best predictor might have a small offset which could be different for each source block. The decoder search and CRC check mechanisms in PRISM allow us to determine the appropriate offset independently for each source block. As our experimental results will show, this is helpful in letting the decoder tolerate small amounts of calibration, correspondence and interpolation errors inherently accumulated in the process of generating the predicted frame.

This will be referred to as PRISM-VS (PRISM view synthesis search).

3.4.4 Decoder disparity search

To perform decoder disparity search, we use the correlation model described in Section 3.3.2. This time, suppose we want to decode a block from camera 2 shown in Figure 3.3. From the epipolar constraint, the best predictor should lie along the epipolar line in the neighboring camera view. Therefore, as shown in Figure 3.3, the decoder would sample

blocks along the epipolar line from a neighboring view to use as side-information in decoding the received syndrome. In practice, we compensate for small amounts of calibration error by allowing the decoder to also search a little above and below the epipolar line.

The architecture of PRISM serves us well here. The small block size lets us assume that there is little depth variation within each block, therefore block sampling along the epipolar line would produce good side-information. Furthermore, the use of DSC allows us to decode the received syndrome using any suitable predictor as side-information. Finally, the use of CRC allows us to determine the success of disparity search and hence the location of the appropriate predictor.

This will be referred to as PRISM-DS (PRISM disparity search).

3.4.5 Discussion

In both PRISM-VS and PRISM-DS, the encoder at each of the video camera sensors does not need any knowledge about the relative positions of any other cameras. This is highly desirable since it reduces the computational and storage burdens on these sensor nodes which do not need to compute or store the camera parameters of its neighboring cameras.

Obviously, decoding has to be causal. For example, in the setup illustrated in Figure 3.3, when decoding the video from camera 1, we would need to make use of the current frame of camera 2 for decoder disparity search, and vice versa. Our solution is to first decode all the views using decoder temporal motion search, as in PRISM. For each block that is not successfully decoded, PRISM-DS and/or PRISM-VS is performed on the *currently available* reconstructions (possibly with error concealment). As it is possible that a successful decoding in one view can lead to a successful decoding in another view, we will attempt PRISM-DS/PRISM-VS across all cameras until either all blocks are successfully reconstructed or there are no further successful reconstructions. Since the number of blocks that fail to decode with temporal motion search is expected to be small, this iteration does

not pose much additional computational burden on the decoder over the original PRISM decoding.

3.5 Experimental results

In our experiments, we used multi-view test videos (cropped to 320 × 240, 30 fps) made publicly available by MERL [94], in which eight cameras were placed along a line, at an inter-camera distance of 19.5 cm, with optical axes that are perpendicular to camera displacement. The sequences are named "Ballroom" and "Vassar". Figure 3.7 shows two central neighboring views from each of the sequences to give an idea of the amount of overlap between cameras.

Each camera is assumed to be transmitting over a separate packet erasure channel. Our simulations used packet erasures generated using a two-state channel simulator to capture the bursty nature of lossy wireless channels, with a "good" state packet erasure rate of 0.5% and "bad" state packet erasure rate of 50%. All tests were carried out on a group-of-pictures (GOP) with 25 frames, and results shown are averaged over 100 trials of wireless channel simulation.

(a) Ballroom (b) Vassar

Figure 3.7. Multi-view test video sequences used in experiments.

3.5.1 Empirical validation of correlation models

To get an understanding of how well the correlation models introduced in Section 3.3 perform, we ran the following experiment. We used the MERL "Vassar" multi-view video sequence. For each source block, we find its mode using the classifier described in Sec-

tion 3.4.1. We then accumulate statistics of the best matching predictor using the view synthesis based correlation model and the disparity based correlation model, as well as using temporal motion search. In Figure 3.8, we show the variance of the innovations noise of the first 16 coefficients (using zig-zag scanning) for one particular mode[3]. We can make several interesting observations from the plot. First, the innovations noise statistics for temporal motion search and disparity search are relatively similar, which suggests that disparity search should provide reasonable side-information for decoding. Second, the disparity search model out-performs the view synthesis search model; this gap is also observed in all our other experimental results. Finally, there is a clear gap in performance between using the co-located block in the view predicted reference and doing a small search around that location.

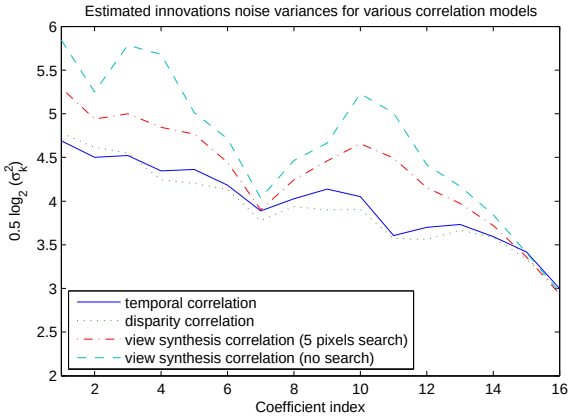


Figure 3.8. Innovations noise statistics of various correlation models. In this graph, we show the innovations noise statistics when different correlation models are used. These statistics were obtained from the "Vassar" multi-view video sequences using the correlation models described earlier, as well as using temporal motion search.

3.5.2 Comparison with simulcast schemes

We compare the performance of our proposed decoding schemes, PRISM-DS and PRISM-VS with the following: (a) PRISM, which uses only decoder motion search; (b) Motion JPEG[4] (MJPEG); (c) H.263+ with forward error correction (H.263+FEC); and

[3]MSE between source and previous co-located block is in the range $[650, 1030]$.

[4]Simulated by coding all frames as I-frames with a H.263+ encoder. We used a free version of H.263+ obtained from University of British Columbia for our simulations.

(d) H.263+ with random intra refresh (H.263+IR). These represent plausible simulcast solutions for multiple cameras.

All test systems used the same total rate of 960 Kbps per camera view, with a latency constraint of 1 frame. Each frame is transmitted with 15 packets, with an average packet size of 270 bytes. For H.263+FEC, we used an appropriate fraction of the rate for FEC, implemented with Reed-Solomon codes, such that the quality with no data loss matches that of PRISM. Similarly, we set the intra-refresh rate for H.263+IR such that the quality with no data loss matches that of PRISM.

Figure 3.9 shows the quality in PSNR of decoded video from all the cameras. In the "Ballroom" sequence, PRISM-DS and PRISM-VS achieved up to 0.9 dB and 0.4 dB gain in PSNR over PRISM respectively. Compared to H.263+FEC, PRISM-DS and PRISM-VS achieved up to 2.5 dB and 2.1 dB gain in PSNR respectively. In the "Vassar" sequences, both PRISM-DS and PRISM-VS demonstrated modest gains over PRISM. The main reason for this is that the "Vassar" sequence is largely static, and compared to the "Ballroom" sequence, a much smaller fraction of the frame consist of moving objects. Hence, the error concealment strategy of copying from the previous frame works very well, and there is very little further gain in using disparity search.

Figure 3.10 shows the recovery behaviors after catastrophic packet losses in frame 1 and again in frame 16, at 8% average packet drop rate. For both video sequences, the PRISM based systems are able to recover quickly after the loss event, with PRISM-DS and PRISM-VS doing better than PRISM. The difference in performance between the two was more significant in the "Ballroom" sequence due to the higher motion content. While H.263+IR demonstrated some error-resilience properties, it took a longer time to recover than PRISM, since it requires a few frames before being able to complete the intra refresh of the entire frame. As expected, the distortion in MJPEG is correlated with just the number of lost packets in that frame, since each frame is independently coded and no dependency exists between frames. However, because of its coding inefficiencies, it is unable to match the quality of the PRISM based systems.

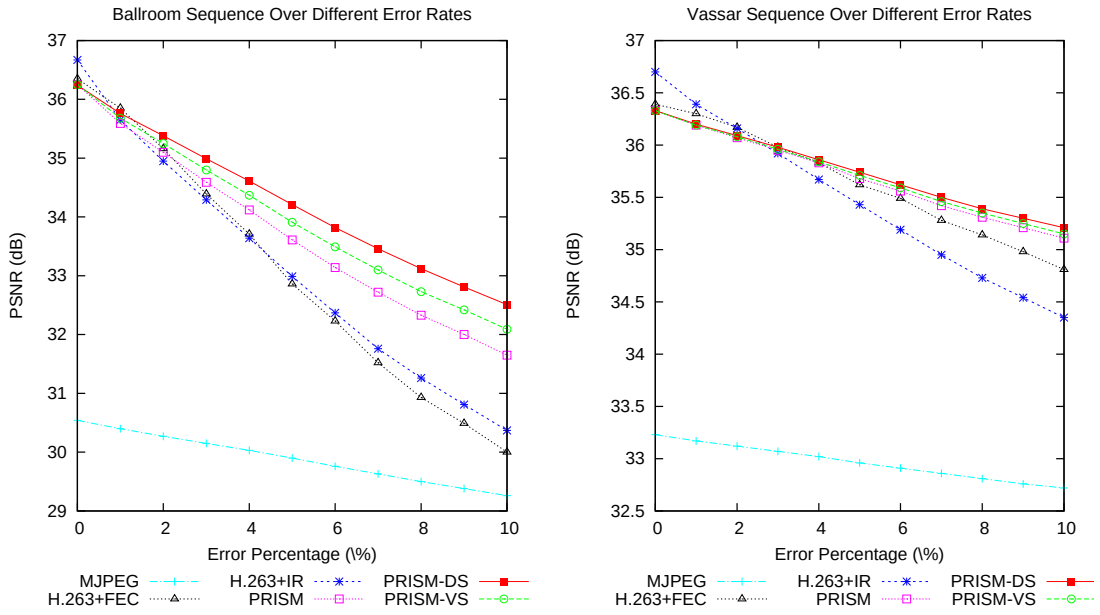
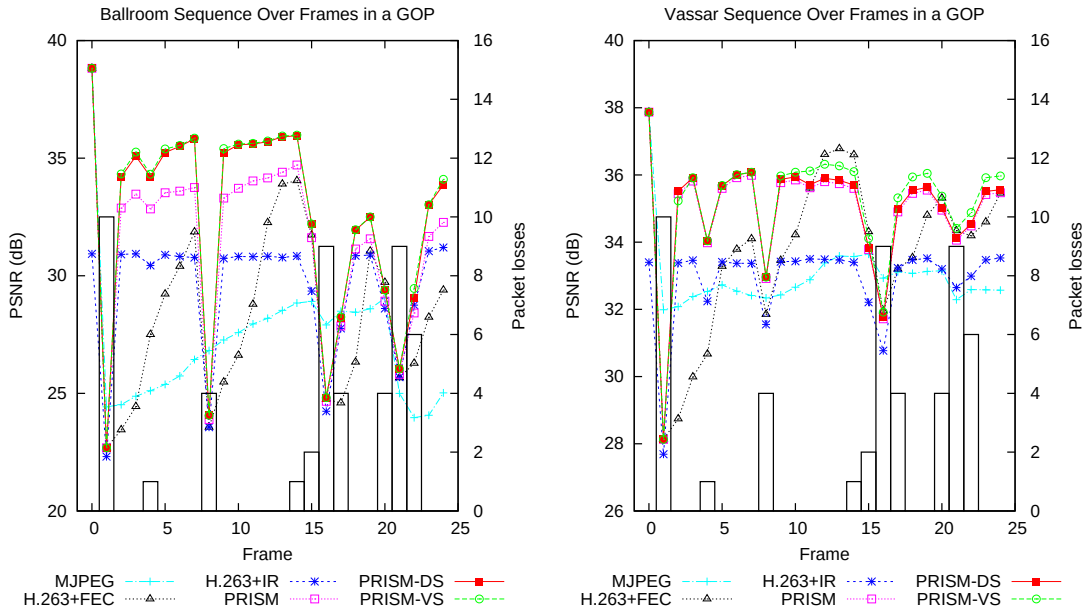


Figure 3.9. System performance over different error rates.

For visual comparison, Figure 3.11 shows a portion of the frame from the "Ballroom" sequence after a catastrophic loss event where 60% (reflecting the bursty nature of wireless packet drops) of the previous frame's packets were dropped. Both PRISM-DS and PRISM-VS produced more visually pleasing reconstruction than the other simulcast schemes.

3.5.3 Effect of decoder search range in view synthesis search

To investigate the effect that performing decoder search has on PRISM-VS, we varied the range of the search size (centered at the co-located block) at the decoder. The results shown in Table 3.1 are for 8% average packet drop rate. As evident, while decoder view synthesis search does help in providing error resilience, we see that its performance saturates at a search range of about ± 2 pixels. As suggested in our earlier discussion on correlation models (see Section 3.5.1), these results further reinforce the point that decoder search is helpful in effectively exploiting side-information generation via dense stereo correspondence and view synthesis. Other distributed video coding schemes[116, 57] code over the entire frame, and hence it would be intractable to try out all combinations of shifts of all blocks from the frame predicted by view synthesis.



(a) (b)

Figure 3.10. System performance over frames at 8% packet outage

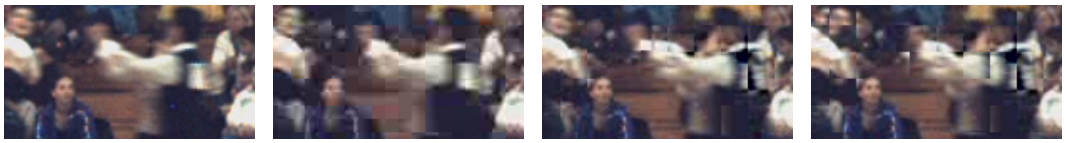
Table 3.1. PSNR (dB) with different search ranges (pixels) in PRISM-VS

PRISM refers to independent decoding without using any other camera views. The other columns refer to decoding with the specified search range in PRISM-VS.

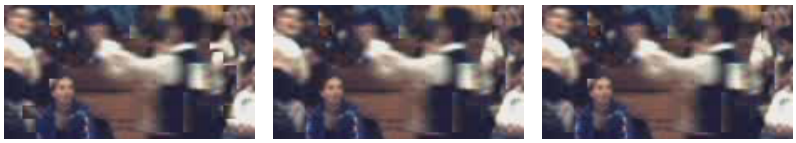
Sequence	PRISM	0	±1	±2	±3	±4
Ballroom	32.54	32.80	33.03	33.06	33.07	33.07
Vassar	35.37	35.37	35.41	35.41	35.41	35.42

3.5.4 Effect of redundancy of cameras used in disparity search

Recall that in PRISM-DS, for a given camera, decoder disparity search can be performed in any of its neighboring cameras. We would like to know how much effect increasing the number of neighboring cameras used would have on robustness. Thus, we performed experiments in which we vary the number of closest neighboring cameras used for PRISM-DS. Since we want to study the effect of having up to 6 neighboring cameras (3 on either side), we only perform this experiment for the center two cameras, and the results reported are their average. The results shown in Table 3.2 are for 8% average packet drop rate.



(a) Original (b) MJPEG (c) H.263+FEC (d) H.263+IR



(e) PRISM (f) PRISM-DS (g) PRISM-VS

Figure 3.11. Visual results of "Ballroom" sequence at 8% average packet outage. Note the obvious blocking artifacts in MJPEG, and the obvious signs of drift in both H.263+FEC and H.263+IR. PRISM-DS and PRISM-VS produced reconstructions that are most visually pleasing.

Table 3.2. PSNR (dB) with different number of cameras used in PRISM-DS

Sequence	PRISM	2	4	6
Ballroom	32.33	33.09	33.10	33.10
Vassar	35.25	35.35	35.36	35.36

Table 3.2 shows that as the number of cameras available for PRISM-DS increases, the quality of the reconstructed video also improves. Thus, PRISM-DS is able to take advantage of additional overlapping cameras for increased robustness. On the other hand, there clearly is a case of diminishing returns as the number of available cameras increases.

3.5.5 Effect of distance of cameras used in view synthesis search and disparity search

In PRISM-DS, for a given camera, decoder disparity search can be performed in any of its neighboring cameras. On the other hand, in PRISM-VS, disparity estimation and view synthesis are done with the closest available neighboring cameras. We would like to know how much effect increasing the distance of neighboring cameras used would have on robustness. Thus, we performed experiments in which we vary the distance of neighboring cameras used for both PRISM-VS and PRISM-DS. As above, we only perform this exper-

iment for the center two cameras, and the results reported are their average. The results shown in Table 3.3 are for 8% average packet drop rate.

Table 3.3. PSNR (dB) with distance of neighboring cameras used in PRISM-DS and PRISM-VS

Sequence	PRISM	1	2	3
Ballroom (DS)	32.33	33.09	33.01	32.95
Vassar (DS)	35.25	35.35	35.34	35.33
Ballroom (VS)	32.33	32.92	32.76	32.61
Vassar (VS)	35.25	35.29	35.28	35.27

Table 3.3 shows that as the distance of the available cameras increase, the reconstruction quality decreases, probably because the quality of side-information degrades. The decrease in reconstruction quality is more marked in the PRISM-VS scheme, due to the fact that when the distance of neighboring cameras increases, disparity estimation is operating on views with wider baseline, thus leading to poorer disparity estimates. This adversely affects performance of view synthesis.

3.6 Recapitulation

In deploying wireless camera networks, it is important to design video transmission systems that take into the account the lossy nature of wireless communications. We have presented a distributed video compression scheme for wireless camera networks that is not only robust to channel loss, but has independent encoders with low encoding complexity that are highly suitable for implementation on sensor mote platforms. While past works on distributed compression of multi-view videos have focused on achieving compression gain, we instead exploit inter-view redundancy to achieve error resilience in a distributed fashion. There is no need to perform correspondence tracking at the encoders and the encoding operation is truly distributed. Our simulation results indicate that PRISM with either view synthesis search or disparity search is able to exploit inter-view correlation for robustness under tight latency constraints. We also show results that demonstrate how our proposed approaches behave when physical camera network parameters, such as how far and

how dense the cameras are placed, are changed. In particular, as the number of available neighboring views increases, PRISM-DS becomes more robust, but with diminishing returns. Furthermore, as the distance of neighboring views increase, the performance of both PRISM-DS and PRISM-VS suffers, with PRISM-VS seeing a larger drop in reconstruction quality.

In future work, we would like to explore "smarter" encoders that are able to estimate inter-camera correlation based on intra-camera properties such as edge strength. This would require further research into (possibly distributed) inter-view correlation estimation. The regime of low frame rate video also promises to be an interesting area of research, since inter-camera correlation could possibly dominate intra-camera temporal correlation. While PRISM is built with H.263+ primitives and so a comparison with H.264 [136] would not provide any insight into the gains made possible by our approach, it would be worthwhile to consider an implementation built with H.264 features, such as in [93]. Specifically, the adoption of smaller block sizes and the integer transform makes it an interesting area of future investigation.

Part II

Establishing visual correspondences under rate-constraints

Chapter 4

Rate considerations in computer vision tasks

As motivated in this dissertation, the availability of cheap wireless sensor motes with imaging capability has inspired research on wireless camera networks that can be cheaply deployed for applications such as environment monitoring [126], surveillance [98] and 3DTV [87] (see Figure 4.1). Much progress has been made on developing suitable wireless camera mote platforms which are compact and self-powered while being able to capture images or videos, perform local processing and transmit information over wireless links [108, 127, 37, 24]. However, the gaping disconnect between high bandwidth image sensors (up to 1280×1024 pixels @ 15 fps [24]) and low bandwidth communications channels (a maximum of 250 kbps per IEEE 802.15.4 channel including overhead [24]) makes the exchange of all captured views impractical. Fortunately, depending on the task assigned to the wireless camera network, exchanging camera views may not be necessary, but there is still a need for intelligent processing that can satisfy bandwidth budgets [79].

Our primary application of interest in this part of the dissertation is camera calibration. Internal calibration, or the determination of camera parameters such as skew, aspect ratio and focal length, and external calibration, or the determination of camera parameters such as relative location and pose, have been the focus of much research in the computer vision

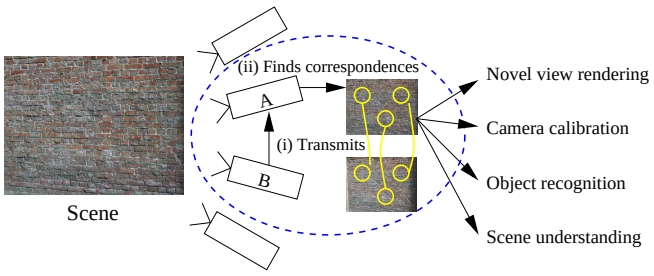


Figure 4.1. Problem setup for distributed visual correspondences. A typical wireless camera network would have many cameras observing the scene. In many computer vision applications such as camera calibration, object recognition, novel view rendering and scene understanding, establishing visual correspondences between camera views is a key step. We are interested in the problem within the dashed ellipse: cameras A and B observe the same scene and camera B sends information to camera A such that camera A can determine a list of visual correspondences between cameras A and B. The objective of our work is to find a way to efficiently transmit such information.

community [83, 62]. It is often reasonable to assume that internal calibration parameters are known, since in cheap cameras without zoom capability, internal calibration is a one-time procedure that can be performed prior to deployment. Then, given a list of correspondences between two camera views, the Essential matrix can be estimated using a variety of methods, while the relative location and orientation of the cameras can be easily extracted from the Essential matrix [83].

Traditionally, computer vision methods assume that images from all cameras are available at a central processor with an implicit one-time communications cost. In a *mobile* and *wireless* camera network, these assumptions are called into question — due to *changing camera states* and *bandwidth constraints*. For example, consider a calibration or localization task. If wireless camera motes are attached to the helmets of security personnel on patrol, it would be important to minimize the rate needed to continuously update the location and orientation of each camera relative to a reference frame. Even if the camera motes are designed to be static, environmental disturbance could affect their pose, thus requiring constant updating of calibration parameters. Furthermore, to avoid central coordination and long communication hops from sensor nodes to a backend server, the calibration procedure should ideally be distributed.

In part to address these practical concerns, there has been recent work on calibration

procedures which are more suitable for wireless camera networks [32, 77, 13]. Devarajan and Radke proposed a distributed algorithm that calibrates each camera's position, orientation and focal length but assumes that feature correspondences are known across cameras [32]. Lee and Aghajan assume the availability of a single moving target that is visible from the cameras that are to be calibrated [77], thus providing a temporal series of correspondences between cameras. Barton-Sweeney *et al.* assume the availability of beacon nodes that identify themselves by using LEDs to broadcast modulated light, hence allowing cameras to determine visual correspondences [13]. However, such constrained or controlled environments are not feasible in a practical deployment.

4.1 Key role of visual correspondences

Many computer vision tasks relevant to camera networks, such as calibration procedures [62, 83], localization [115], vision graph building [26], object recognition [43, 82, 14], novel view rendering [7, 118] and scene understanding [50, 112], typically require a list of visual correspondences between cameras. As illustrated in Figure 4.2, a visual correspondence refers to the set of image points, one from each camera, which are known to be projections of the same point in the observed scene. Partly due to the critical role that visual correspondences play in a wide variety of computer vision tasks that are relevant for wireless camera networks, we focus on the problem of finding visual correspondences between two cameras, denoted as camera A and camera B, communicating under rate constraints. Although we primarily use the two cameras problem as a way to illustrate our proposed approaches, the solutions presented in Chapter 5 can in fact be directly extended to a multiple cameras scenario.

In a centralized setup, one typical approach to finding visual correspondences is to make use of point *features* and *descriptors*. *Features*, or interest points, are first located in the images. *Descriptors* are then computed for each feature; these describe the image neighborhood around each feature and are usually high-dimensional vectors. Visual cor-

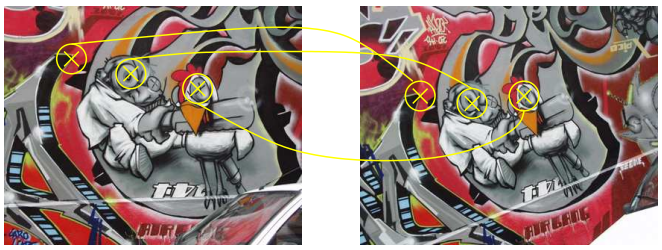


Figure 4.2. Visual correspondences example. In this example, we show two views taken of the same scene ("Graf" [91]). In each view, we have marked out 3 feature points and a line is drawn between each pair of corresponding features. A pair of visual correspondence tells us that the image points are of the same physical point in the scene.

respondences are then found by performing feature matching between all pairs of features between cameras A and B, based on some distance measure between descriptors.

In a distributed setting as shown in Figure 4.1, camera B should transmit information to camera A such that camera A can determine a list of point correspondences with camera B. A naïve approach would be for camera B to send either its entire image or a list of its features and descriptors to camera A for further processing [26]. A key observation is that in the feature matching process, the *Euclidean distance between descriptors* is often used as the matching criterion [82, 91]. Pairs of features that are estimated to be in correspondence would therefore have descriptors that are *highly correlated*. This observation suggests that a distributed source coding [120, 137] (DSC) approach can be used to exploit the correlation between corresponding features to reduce the rate needed for finding visual correspondences. We will discuss such an approach in detail in Chapter 5. Another approach that combines DSC with binarized random projections will also be discussed in Chapter 5.

One might question the choice of sending descriptors of features instead of the actual image itself. However, if the task is to establish visual correspondences, then as we will show in our results, sending descriptors is more bandwidth efficient than sending the entire image, even if lossless image compression is employed. While lossy compression schemes such as JPEG can be used, it would cause both feature localization and matching performance to degrade [91]. Furthermore, in a distributed setting, computing and transmitting descriptors at each camera offers the advantage of reducing redundant computational load. For example, in our setup, camera A only needs to compute descriptors for its own image, instead of

having to do so for both cameras A and B. This subtle point becomes even more important in the general case of where camera A might be receiving data from multiple cameras for calibration. Consider the example of a K cameras network sharing visibility in the region of observation. Each of the K cameras would have to compute descriptors for K observed images if the actual image was sent. However, if descriptors were transmitted, then each of the K camera only has to compute descriptors once for its own observed image.

4.2 Background

In this section, we discuss relevant background material on feature detectors and descriptors which are used in finding visual correspondences. Note that background material on distributed source coding is already covered in Chapter 2 (Section 2.2).

4.2.1 Feature detector

Feature detectors are used in computer vision applications as diverse as wide baseline matching, image retrieval, camera localization and object categorization [90]. Their goal is to detect and localize features, or interest points, that are invariant to rotations, scale changes and affine image transformations. Ideally, the same image patch can be reliably detected and accurately localized under such transforms.

We primarily use an off-the-shelf Hessian-Affine region detector [90], giving output like the example shown in Figure 4.3. In a comparison of affine region detectors, the Hessian-Affine region detector has been shown to have good repeatability and provide more features than other detectors [92].

We briefly describe the two step process used in the Hessian-Affine region detector. First, a Hessian-Laplace region detector is used to localize interest points. The determinant of the Hessian matrix of each pixel in the image is computed at various scales and candidate points are localized in space at local maximas of the computed determinant at each scale [92]. Then, candidate points which are also local maximas of the image Laplacian-

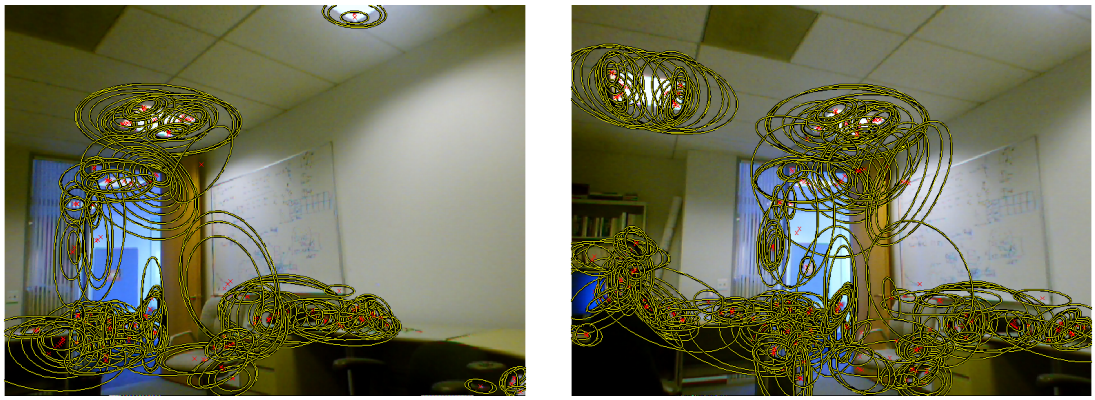


Figure 4.3. Examples of regions found by Hessian-Affine region detector. The detected interest points are plotted as red crosses, while the estimated affine region are shown as yellow ellipses.

of-Gaussians across scale are retained as interest points [90]. Therefore, they are simultaneously local maximas in space (in the determinant of the image Hessian) and in scale (in the Laplacian-of-Gaussians). Second, an affine adaptation step is carried out to estimate a feature neighborhood that is invariant to affine image transformations. This is a procedure that assumes the region is elliptical in shape and iterates between estimating the shape of the region and warping it into a circular region until the eigenvalues of the second moment matrix of the warped interest point are equal [90]. Such an affine adaptation is important when there are significant viewpoint changes between cameras.

4.2.2 Feature descriptor

Typically used in conjunction with feature detectors, feature descriptors are used to uniquely characterize a region of interest and are required to be robust to illumination and affine distortion. Descriptors can range from simple constructions, such as a vector of image pixels in the region of interest, to complex constructions, such as a histogram of gradients [91]. Ideally, the same image patch under different viewing conditions should yield descriptors that are close under some similarity measure, such as their euclidean distance.

We primarily use an off-the-shelf scale-invariant feature transform (SIFT) descriptor [82], which are 128-dimensional descriptors constructed to be invariant to scale and orientation changes and robust to illumination and affine distortions. SIFT has shown to

have good performance in practice and are widely used in computer vision [82, 91]. Briefly, the descriptors are computed as illustrated in Figure 4.4. First, the pixel neighborhood of the interest point, computed during Hessian-Affine region detection, is rotated, scaled and warped to achieve rotational, scale and affine invariance. Next, the area of pixels is divided into a total of 4×4 tiles. An 8-bin histogram of image gradients is constructed for each tile from the pixels in that tile, where each entry is weighted by the magnitude of each image gradient. The histograms are then stacked together to form a 128-dimensional vector. Finally, the vector is normalized to mitigate illumination induced effects.

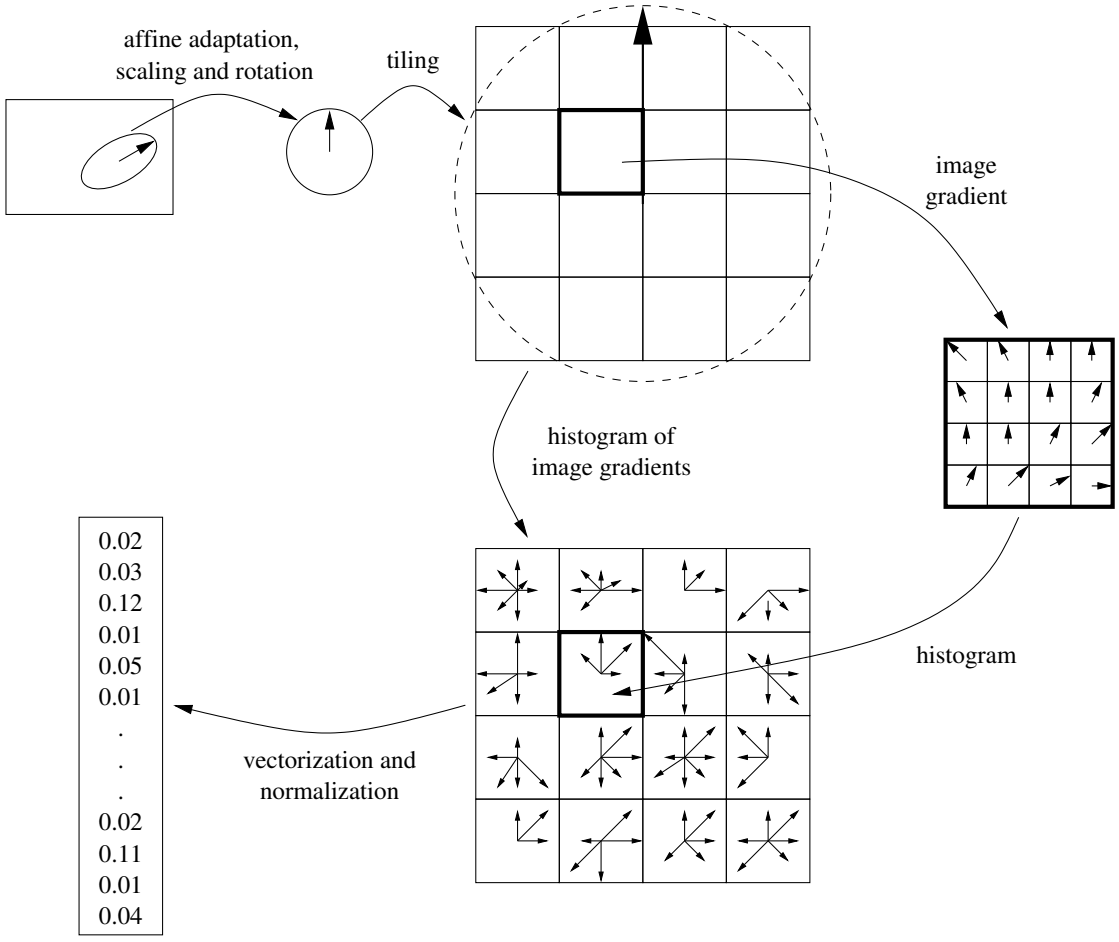


Figure 4.4. Computation of SIFT descriptor. Each detected image region is first warped, scaled and rotated to achieve affine invariance, scale invariance and rotational invariance. The image patch in the warped region is then divided into 4×4 tiles of pixels. In each tile, the computed image gradients are binned into an 8-bin histogram based on the orientation of the gradient and weighted by its magnitude. The 16 8-bin histograms are then stacked into a 128-dimensional vector which is normalized.

Chapter 5

Strategies for rate-constrained distributed distance testing

The distributed visual correspondences problem discussed in Chapter 4 is just one member of a general class of problems that we term *distributed distance testing under severe rate-constraints*, which is illustrated in Figure 5.1. Suppose there are two distributed sources, one that outputs $\vec{D}^A \in \mathbb{R}^N$, and another that outputs $\vec{D}^B \in \mathbb{R}^N$. Say Alice observes $\vec{D}^A$ and Bob observes $\vec{D}^B$. Under severe rate constraints in a distributed setup, Alice would like to know with high probability if $\|\vec{D}^A - \vec{D}^B\|_2 < \tau$.

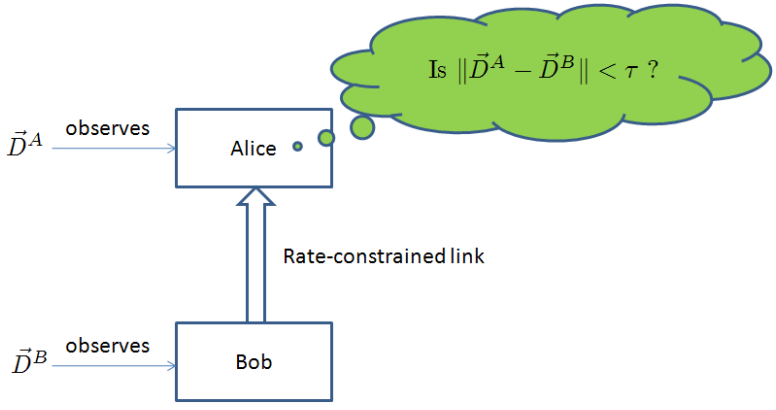


Figure 5.1. Rate-constrained distributed distance testing setup.

One other class of applications where such a problem arises is that of remote image

authentication (see for example [110, 81]) and video hashing. Suppose Alice wants to verify that her copy of an image (or video) is similar to what Bob has. To do so, they can compare perceptual hashes of their images. One possibility is to use the actual images such that the transmitted hash checks out if the images satisfy some mean square error (MSE) distortion constraint [81]. Alternatively, we can perform such a distance check in some image feature space [110].

A straight-forward solution is for Bob to send some suitably transformed and quantized version of $\vec{D}^B$ to Alice. However, under severe rate constraints in a distributed setup, this might not be the best approach. In this chapter, we present two alternative approaches. The first uses a distributed source coding [120, 137] (DSC) approach that exploits the correlation between $\vec{D}^A$ and $\vec{D}^B$ for rate reductions. This is discussed in Section 5.3. The second approach combines DSC with a distance-preserving binarized random projections hash and is discussed in Section 5.4.

To provide a concrete context for discussion in this chapter, we consider the problem of establishing visual correspondences in a distributed fashion between cameras operating under rate constraints, as illustrated in Figure 4.1. Cameras A and B have overlapping views of the same scene and camera A wishes to obtain a list of visual correspondences between the two cameras. Camera B should send information in a rate-efficient manner such that camera A can obtain this list and use it for any other down-stream computer vision task. We assume that both cameras A and B have already extracted a list of features and computed descriptors for each of the features from their respective image views. Let A_i denote the ith feature out of N_A features in camera A, with image coordinates (x_i^A, y_i^A) and descriptor $\vec{D}_i^A$, and B_j denote the jth feature out of N_B features in camera B, with image coordinates (x_j^B, y_j^B) and descriptor $\vec{D}_j^B$. In this chapter, we assume that camera A will determine that A_i corresponds with B_j if

$$\|\vec{D}_i^A - \vec{D}_j^B\|_2 < \tau \tag{5.1}$$

for some acceptance threshold τ. We denote this as the *Euclidean matching criterion*.

The work presented in this chapter is joint work with Parvez Ahammad and Kannan

Ramchandran, and has been presented in part in [141, 142, 145]. We also like to acknowledge the advice and assistance given by Hao Zhang.

5.1 Contributions

We make the following contributions in this chapter. First, we propose the novel use of DSC in the problem of establishing visual correspondences between cameras in a rate-efficient manner. We verify that descriptors of corresponding features are highly correlated, and describe a framework for applying DSC in feature matching given a particular matching constraint.

Next, we propose the use of coarsely quantized random projections of descriptors to build binary hashes and the use of Hamming distance between binary hashes as the matching criterion. We derive the analytic relationship of Hamming distance between the binary hashes to Euclidean distance between the original descriptors. We then show how a linear code can be applied to further reduce the rate needed. In particular, the rate to use for the code can be easily determined by the desired Euclidean distance threshold and a target probability of error.

Finally, we set up a systematic framework for performance evaluation of establishing visual correspondences by viewing it as a retrieval (of visual correspondences) problem under rate constraints. While Mikolajczyk and Schmid [90] consider the relative performance of various descriptors for correspondence, here we investigate an orthogonal direction in which rate constraints are imposed. Cheng *et al.* [26] considered the performance of vision graph building under rate constraints; however, establishing visual correspondence is a more fundamental task and we believe our approach and results are widely applicable to a variety of vision tasks. While we demonstrate our proposed method on a particular choice of feature detector and descriptor, namely the Hessian-Affine region detector [90] and Scale-Invariant Feature Transform (SIFT) descriptor [82], the framework is generally applicable to any other combination of feature detectors and descriptors.

5.2 Related work

Han and Amari presented a survey of work on statistical inference with consideration of communications costs [61]; while they presented theoretical and asymptotic results on achievable error-exponents, no constructive and practical scheme is given.

Cheng *et al.* studied the related problem of determining a vision graph that indicates which cameras in the network have significant overlap in their field of view [26]. A key component of their proposed approach is a rate-efficient feature digest constructed from features and their descriptors. They do this by applying Principal Components Analysis (PCA) to the descriptors and achieve dimensionality reduction by sending only the coefficients of the top principal components. However, they chose an arbitrary number of bytes (4) to represent each coefficient and ignored the correlation in descriptors between the matched features. Chandrasekhar *et al.* apply transform coding and arithmetic coding on descriptors to build compressed features for image matching and retrieval [19]. Tosic and Frossard studied the use of over-complete decompositions in establishing coarse correspondences between omni-directional cameras [128]. However, in these works, performance is evaluated on either the detection of overlapping views between cameras [26] or depth recovery of a small number of simple objects in a synthetic scene [128]. In particular, the performance of establishing visual correspondences is not evaluated directly.

Roy and Sun used binarized random projections to build a descriptor hash [110]; the Hamming distance between hash bits is then used to establish matching features. Martinian *et al.* proposed a way of storing biometrics securely using a syndrome code to encode the enrolled biometric bits [86], while Lin *et al.* proposed the use of syndrome codes on quantized projections for image authentication [81]. In both approaches, the syndrome is decoded using the test biometric or test image as side-information; a match is signaled by decoding success. However, the rate of the syndrome code has to be chosen by trial and error to balance security, false positive and false negative performance.

5.3 Distributed source coding of descriptors

Recall that camera B wishes to send information for camera A to determine correspondences between the two cameras (see Figure 4.1.) One possible approach is for camera B to send $\left\{(x_j^B, y_j^B), \vec{D}_j^B\right\}_{j=1}^{N_B}$ to camera A. However, camera A *does not* actually care to reconstruct the descriptors from camera B. Instead, camera A just wants to know whether each descriptor from camera B is of a point that matches a feature from its own camera view. In particular, if B_j is a feature that does not correspond with any features in camera A, then there is no need for camera A to reconstruct $\vec{D}_j^B$. This inspires us to use DSC to send *just enough bits* for each descriptor $\vec{D}_j^B$ such that it can be decoded using $\vec{D}_i^A$ as side-information if A_i corresponds with B_j.

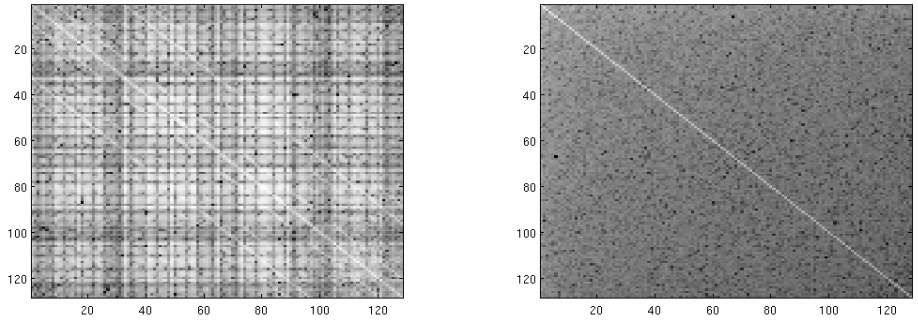
5.3.1 Descriptor coefficients de-correlation

Due to how SIFT descriptors are computed [82], their coefficients are highly correlated. This is clearly demonstrated in Figure 5.2(a), which visualizes the covariance matrix of a set of descriptors. We aim to de-correlate the coefficients by applying a linear transform (*i.e.* its discrete Karhunen-Loève Transform) to the descriptor prior to encoding. This transform is estimated by applying principal components analysis (PCA) to a set of 12514 descriptors computed over a collection of 6 training images. Figure 5.2(b) shows that applying the learned linear transform does a good job of de-correlation; the same linear transform will be used in our experiments. For notational convenience, we will assume in this section that $\vec{D}_i^A$ and $\vec{D}_j^B$ refer to the decorrelated descriptors.

5.3.2 Correlation model for descriptors of corresponding features

As discussed in the DSC background in Section 2.2, to apply DSC effectively, we need a reasonable correlation model for the descriptors of corresponding features. Suppose that A_i and B_j are corresponding features; we will assume that their descriptors satisfy the following correlation model:

$$\vec{D}_j^B = \vec{D}_i^A + \vec{N}_{ji}^{BA} \tag{5.2}$$



(a) Before PCA (b) After PCA

Figure 5.2. De-correlating SIFT descriptors. Here, we show the covariance matrix of the SIFT descriptors before and after applying PCA to de-correlate the coefficients. For better visualization, we show the logarithm of the absolute values of the covariance matrix. A brighter value thus indicates greater correlation between coefficients. It is clear from (a) that coefficients of the SIFT descriptor are highly correlated. After applying PCA however, most of the correlation between coefficients have been removed, as can be seen in (b).

Here, $\vec{N}_{ji}^{BA}$ denotes the innovation noise between $\vec{D}_i^A$ and $\vec{D}_j^B$, *i.e.* the side-information at camera A, $\vec{D}_i^A$, is a noisy version of $\vec{D}_j^B$. Since de-correlation has been performed (see Section 5.3.1), we assume $\vec{N}_{ji}^{BA}$ is also de-correlated. Furthermore, we will assume that the innovation noise is Gaussian. This enables bit allocation on each component through inverse waterfilling on the innovations noise components [29, 103].

We estimate the statistics of the innovation noise, $\vec{N}_{ji}^{BA}$, from descriptors of features belonging to known and detected correspondences in training image pairs. In other words, we apply the *Euclidean matching criterion* as in (5.1) to descriptors and restrict our attention to the estimated correspondences which are also correct. For this set of correspondences, we compute $\vec{N}_{ji}^{BA}$ as in (5.2) and then use this to estimate $\left\{\sigma_k^2\right\}_{k=1}^{128}$, where σ_k^2 is the variance of the kth element of innovation noise. We will defer discussion of how ground-truth correspondences are obtained to Section 5.5. Figure 5.3 shows both the source variance and innovation noise variance (for $\tau = 0.195$) of the descriptor coefficients in a log plot. It is clear that the innovation noise variance is much smaller than the source variance and that allows us to achieve rate savings.

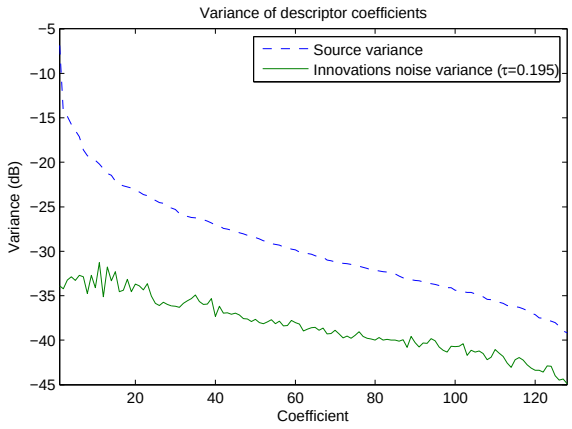


Figure 5.3. Variance of descriptor coefficients. We show the variance of each coefficient of both descriptors and innovations noise after applying de-correlation. Note that variance is shown in dB in this plot for a better visual comparison. The innovation noise variance is clearly much smaller than that of the original coefficients and this enables the rate savings.

5.3.3 Descriptor encoding and decoding

Instead of just sending $\vec{D}_j^B$ as is, we propose using DSC by dividing up the quantized descriptor space into cosets and sending the *coset index* of the quantized $\vec{D}_j^B$. The size of the coset would depend on the correlation strength between descriptors of corresponding features, using the correlation model described earlier in Section 5.3.2. In theory, we would find a channel code that is matched to $\vec{N}_{ji}^{BA}$ and use that to partition the quantized codeword space of $\vec{D}_j^B$ [104]. Furthermore, since the choice of side-information for B_j is unknown at camera A, the coset size needs to be reduced appropriately to account for that uncertainty [67]. Joint-typical decoding can then be used to recover $\vec{D}_j^B$ *if there is indeed a corresponding feature in camera A* [67]; this involves checking through all combinations of codewords in the coset and side-information at camera A and picking the codeword that is jointly-typical with some $\vec{D}_i^A$. In our work, we use cosets of a Multilevel code for encoding and decoding [131, 66]. Due to short block lengths and complexity constraints at the decoder, we first construct the Maximum-Likelihood (ML) estimate of $\vec{D}_j^B$ given the received coset index and side-information before using a Cyclic Redundancy Check (CRC) to verify decoding success.

Based on the estimated innovation noise statistics for each coefficient of the descriptor,

we compute the number of levels to use in the Multilevel code. In particular, we use L_k levels for the kth coefficient [48]:

$$L_k = \left\lceil \frac{1}{2} \log_2 \left(2\pi e \alpha^2 \frac{\sigma_k^2}{\delta^2} \right) \right\rceil \tag{5.3}$$

where δ is the desired quantization step size and α is a user parameter that determines the probability of decoding error. Recall that σ_k^2 is the variance of the kth element of $\vec{N}_{ji}^{BA}$. While further compression is possible by coding each of the levels across all coefficients as in the multilevel coset code framework, we transmit the coset indices uncoded in this work.

To encode a descriptor, the encoder will compute and transmit the coset index of each descriptor based on the Multilevel code determined by Equation (5.3); for the kth coefficient, the coset index is just the least significant L_k bits of the quantized coefficient. In addition, the encoder will compute and transmit a sufficiently strong CRC of the quantized descriptor. To ensure a low probability of collisions, a reasonable choice would be to use at least $\lceil 2 \log_2(N_A) \rceil$ CRC bits. This expression is obtained by by treating hash collision as a birthday attack [123] with N_A attempts and a uniform distribution assumption on the CRC hash that is computed. The image coordinates of the feature are also transmitted.

At the decoder, camera A will take each received coset index and perform multi-stage decoding using its descriptors, $\left\{ \vec{D}_i^A \right\}_{i=1}^{N_A}$, as side-information. Each candidate decoded descriptor will then be tested with the received CRC. If the CRC for received feature B_j checks out with feature A_i as side-information, then A_i is very likely to be the corresponding feature to B_j. A second pass can be performed to ensure that A_i is indeed the matching feature by using the *Euclidean matching criterion*. If the CRC for B_j does not check out with any feature A_i as side-information, then it is likely that B_j has no corresponding feature in camera A that satisfies the *Euclidean matching criterion*. This descriptor can then be discarded since camera A would not have been able to use this feature anyway.

5.3.4 Algorithmic summary

We summarize the encoder and decoder operations here. The encoder is described in Algorithm 1. For the jth descriptor, $\vec{S}_j^B$ is the vector of coset indices and C_j^B is the

computed CRC hash. The encoder takes as input the set of features and descriptors that are found by camera B and returns their coset indices and CRC hashes.

Algorithm 1 Encodes descriptors from camera B

Input: N_B, $\left\{\left(x_j^B, y_j^B\right), \vec{D}_j^B\right\}_{j=1}^{N_B}$

Output: $\left\{\left(x_j^B, y_j^B\right), \vec{S}_j^B, C_j^B\right\}_{j=1}^{N_B}$

for $j = 1$ to N_B **do**

Quantize each coefficient of $\vec{D}_j^B$ with step size δ

Compute $\vec{S}_j^B$ by keeping L_k (see Equation (5.3)) least significant bits of the kth coefficient of quantized $\vec{D}_j^B$

Compute C_j^B, the CRC of quantized $\vec{D}_j^B$

end for

The decoder is described in Algorithm 2. It takes as input the set of features and descriptors that are found by camera A and the received coset indices and CRC hashes of descriptors from camera B. The decoder then returns a list of visual correspondences between cameras A and B that are found.

5.4 Distance preserving hashes using binarized random projections

Inspired by work from Roy and Sun, we use coarsely quantized random projections to build a descriptor hash [110]; the Hamming distance between hash bits can then be used to determine if two features are in correspondence. For a feature point with descriptor $\vec{D} \in \mathbb{R}^n$, we construct a M-bit binary hash, $\vec{d} \in \{0,1\}^M$, from $\vec{D}$ using random projections as follows [110]. First, randomly generate a set of M hyperplanes that pass through the origin, $\mathcal{H} = \{H_1, H_2, \ldots, H_M\}$ and denote the normal vector of the kth hyperplane, H_k, by $\vec{h}_k \in \mathbb{R}^n$. Next, the kth bit of $\vec{d}$, $d(k) \in \{0,1\}$, is computed based on which side of the kth hyperplane $\vec{D}$ lies. In other words,

$$d(k) = \mathbb{I}\left[\vec{h}_k \cdot \vec{D} > 0\right] \tag{5.4}$$

Algorithm 2 Decode transmissions from camera B and find visual correspondences between camera A and camera B

Input: N_A, $\left\{\left(x_i^A, y_i^A\right), \vec{D}_i^A\right\}_{i=1}^{N_A}$

Input: N_B, $\left\{\left(x_j^B, y_j^B\right), \vec{S}_j^B, C_j^B\right\}_{j=1}^{N_B}$ {Received from camera B}

Output: List of visual correspondences between cameras A and B

```
for j = 1 to N_B do
  for i = 1 to N_A do
    Decode S⃗_j^B using D⃗_i^A as side-information
    if CRC of decoded codeword checks out with C_j^B then
      Dequantize decoded codeword to get D̂_j^B
      if ‖D̂_j^B − D⃗_i^A‖_2 < τ then
        Add (i, j) to the list of visual correspondences
      end if
    end if
  end for
end for
```

The intuition for using such a hash is that if two descriptors are close, then they will be on the same side of a large number of hyperplanes and hence have a large number of hash bits in agreement [110]. Therefore, to determine if two descriptors are in correspondence, we can simply threshold their *Hamming distance*. This also has the advantage that computing Hamming distances between descriptor hashes is computationally cheaper than computing Euclidean distances between descriptors.

5.4.1 Analysis of binarized random projections

To pick a suitable threshold, we need to understand how Hamming distances between descriptor hashes are related to Euclidean distances between descriptors. In this section, we assume that descriptors are normalized to unit length. This is not unreasonable; for example, SIFT descriptors are normalized in the last step of descriptor computation [82]

(see Section 4.2.2). With this assumption, we can show the following theorem about how a single hash bit relates to the distance between two descriptors and then use it to show the relationship between Hamming distance between the binary hashes and the Euclidean distance between the descriptors. After performing this work, we subsequently found that a similar theorem was used in similarity estimation (Section 3, [22]) and in approximate maximum cuts computation (Lemma 3.2, [58]).

Theorem 1. *Suppose n-dimensional descriptors $\vec{D}_i^A$ and $\vec{D}_j^B$ are separated by Euclidean distance δ, i.e. $\|\vec{D}_i^A - \vec{D}_j^B\|_2 = \delta$. Then, the probability that a randomly (uniformly) generated hyperplane will separate the descriptors is $\frac{2}{\pi}\sin^{-1}\frac{\delta}{2}$.*

Corollary 1. *Suppose n-dimensional descriptors $\vec{D}_i^A$ and $\vec{D}_j^B$ are separated by Euclidean distance δ, i.e. $\|\vec{D}_i^A - \vec{D}_j^B\|_2 = \delta$. If we generate M-bit binary hashes, $\vec{d}_i^A$ and $\vec{d}_j^B$, from $\vec{D}_i^A$ and $\vec{D}_j^B$ respectively, then their Hamming distance, $d_H(\vec{d}_i^A, \vec{d}_j^B)$, has a binomial distribution, $Bi\left(M, p_{ij}^{AB}\right)$, where $p_{ij}^{AB} = \frac{2}{\pi}\sin^{-1}\frac{\delta}{2}$. Furthermore, the ML estimate of the Euclidean distance between descriptors is given by $\hat{\delta} = 2\sin\left(\frac{d_H(\vec{d}_i^A, \vec{d}_j^B)}{M} \cdot \frac{\pi}{2}\right)$.*

Proof of Corollary 1. $d_H(\vec{d}_i^A, \vec{d}_j^B)$ is just the number of times a randomly generated hyperplane separates the two descriptors. Since the hyperplanes are generated independently, the Hamming distance has a binomial distribution with the Bernoulli parameter given by Theorem 1. The ML estimate can then be found in a straightforward fashion. □

Notice that the ML estimate is independent of the dimensionality of the descriptor.

To prove Theorem 1, we need the following lemma.

Lemma 1. *Suppose 2-dimensional descriptors $\vec{D}_i^A$ and $\vec{D}_j^B$ are separated by Euclidean distance δ, i.e. $\|\vec{D}_i^A - \vec{D}_j^B\|_2 = \delta$. Then, the probability that a randomly (uniformly) generated hyperplane will separate the descriptors is $\frac{2\sin^{-1}\frac{\delta}{2}}{\pi}$.*

Proof. In the simple case of 2 dimensions as illustrated in Figure 5.4, $\vec{D}_i^A$ and $\vec{D}_j^B$ lies on a unit circle with center at the origin since descriptors have unit-norm. A randomly (uniformly) generated hyperplane in this case is just a line passing through the origin with

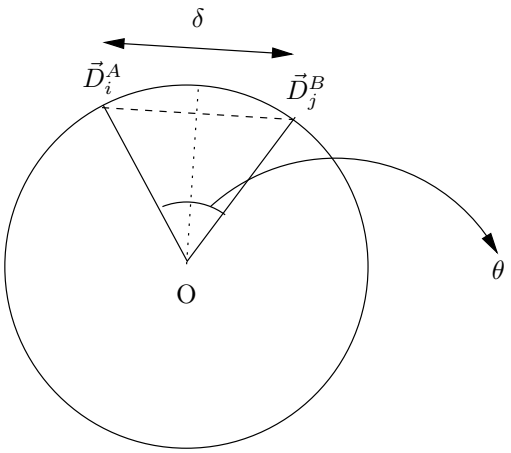


Figure 5.4. Graphical illustration of proof for Lemma 1. A general multi-dimensional case can always be reduced to a 2-D case, in the plane formed by $\vec{D}_i^A$, $\vec{D}_j^B$, and the origin. The angle subtended by the rays from the origin to $\vec{D}_i^A$ and $\vec{D}_j^B$ in this plane can be found using simple trigonometry to be $\theta = 2\sin^{-1}(\delta/2)$. If a hyperplane orientation is chosen uniformly at random, then the probability of the hyperplane separating $\vec{D}_i^A$ and $\vec{D}_j^B$ is just θ/π.

equal probability of being in any orientation. Observe that the hyperplane (line) separates the descriptors (denoted by event $\mathcal{E}$) *if and only if* it intersects the shorter of the arcs connecting $\vec{D}_i^A$ and $\vec{D}_j^B$. Hence, by simple trigonometry,

$$P(\mathcal{E}) = \frac{\text{Arc length between } \vec{D}_i^A \text{ and } \vec{D}_j^B}{\pi} = \frac{2\sin^{-1}\frac{\delta}{2}}{\pi}$$

□

We also need the following lemma to link the relationship between a general n-dimensional case and the 2-dimensional case.

Lemma 2. *Suppose we are given two points, D_1 and D_2, in n-dimensional space and a hyperplane H with normal vector $\vec{h}$. Consider the plane S defined by the origin (denoted by O), D_1 and D_2. Then, H separates the two points, D_1 and D_2,* if and only if *the line intersection between H and S also separates the projections of D_1 and D_2 on S.*

Proof. A point, X, lies in H if $\vec{h}^T\vec{OX} = 0$. Also, a point, X, in S can be parametrized as $\vec{OX} = \alpha\vec{OD}_1 + \beta\vec{OD}_2$, for some $\alpha, \beta \in \mathbb{R}$. Then, the line intersection between H and S can

be found by solving:

$$\begin{aligned} \vec{h}^T \left(\alpha O\vec{D}_1 + \beta O\vec{D}_2\right) &= 0 \\ \Rightarrow \alpha \vec{h}^T O\vec{D}_1 + \beta \vec{h}^T O\vec{D}_2 &= 0 \end{aligned} \tag{5.5}$$

Let us consider the first part of the lemma. If H separates the two points, then the projections of the points on $\vec{h}$ has opposite signs, *i.e.*

$$\left(\vec{h}^T O\vec{D}_1\right)\left(\vec{h}^T O\vec{D}_2\right) < 0 \tag{5.6}$$

Now, consider the following two exterior products[1]. The first is the exterior product of the vector representing the line intersection and the vector representing D_1.

$$\begin{aligned} O\vec{X} \wedge O\vec{D}_1 &= \left(\alpha O\vec{D}_1 + \beta O\vec{D}_2\right) \wedge O\vec{D}_1 \\ &= \beta O\vec{D}_2 \wedge O\vec{D}_1 \qquad (5.7) \\ &= -\beta O\vec{D}_1 \wedge O\vec{D}_2 \qquad (5.8) \end{aligned}$$

where (5.7) follows from the property that $\vec{v} \wedge \vec{v} = 0$, and (5.8) follows from the anti-symmetric property that $\vec{v} \wedge \vec{u} = -\vec{u} \wedge \vec{v}$. Similarly, we can compute

$$O\vec{X} \wedge O\vec{D}_2 = \alpha O\vec{D}_1 \wedge O\vec{D}_2 \tag{5.9}$$

Since X lies on the line intersection, from (5.5), we have that

$$\begin{aligned} \alpha \vec{h}^T O\vec{D}_1 &= -\beta \vec{h}^T O\vec{D}_2 \\ \Rightarrow \alpha\beta \left(\vec{h}^T O\vec{D}_1\right)^2 &= \beta^2 \left[-\left(\vec{h}^T O\vec{D}_1\right)\left(\vec{h}^T O\vec{D}_2\right)\right] \qquad (5.10) \\ \Rightarrow \alpha\beta &> 0 \qquad (5.11) \end{aligned}$$

where (5.10) follows by multiplying $\beta\left(\vec{h}^T O\vec{D}_1\right)$ on both sides, and (5.11) follows from (5.6). Finally, from (5.8), (5.9) and (5.11), we conclude that the line $O\vec{X}$ separates the point D_1 and D_2 on S since the bi-vectors $O\vec{X} \wedge O\vec{D}_1$ and $O\vec{X} \wedge O\vec{D}_2$ have opposite orientations. Thus, if H separates the two points D_1 and D_2 then the line intersection between H and S also separates the projections of D_1 and D_2 on S.

[1] One can think of it as the analog of cross-product in high (> 3) dimensional spaces.

Now, for the reverse direction, suppose that H does not separate the two points D_1 and D_2. Following the above argument, we can show that since $\left(\vec{h}^T O\vec{D}_1\right)\left(\vec{h}^T O\vec{D}_2\right) > 0$, $\alpha\beta < 0$, and so the line $O\vec{X}$ does not separate the point D_1 and D_2 on S, since the bi-vectors $O\vec{X} \wedge O\vec{D}_1$ and $O\vec{X} \wedge O\vec{D}_2$ have the same orientation. Thus, if H does not separate the two points D_1 and D_2 then the line intersection between H and S also does not separate the projections of D_1 and D_2 on S. □

Now, we can easily prove Theorem 1.

Proof of Theorem 1. We will show the result by reducing to the 2-D case as in Lemma 1. $\vec{D}_i^A$, $\vec{D}_j^B$ and the origin defines a plane, $\mathcal{S}$. From Lemma 2, a hyperplane H passing through the origin separates the descriptors *if and only if* the line intersection between H and $\mathcal{S}$ also separates the projections of $\vec{D}_i^A$ and $\vec{D}_j^B$ on $\mathcal{S}$ *(almost surely)*. Since this line has equal probability of being in any orientation, the result follows by applying Lemma 1. □

Using Theorem 1, we convert the distance testing problem from a deterministic and continuous-valued problem to a probabilistic and binary-valued one. Specifically, we can model $d_i^A(k)$ and $d_j^B(k)$ as being related by a binary symmetric channel (BSC) with parameter $\rho(\delta)$ given by:

$$\rho(\delta) = \frac{2}{\pi}\sin^{-1}\frac{\delta}{2} \tag{5.12}$$

when $\|\vec{D}_i^A - \vec{D}_j^B\|_2 = \delta$.

5.4.2 Numerical demonstration of Theorem 1

To demonstrate Theorem 1, we ran the following experiment on descriptors obtained from a separate set of training image pairs. We consider the set of all possible pairs of descriptors, and pick at random equal number of corresponding and non-corresponding pairs. We then compute the Euclidean distance between the pair, and estimate the probability that a randomly generated hyperplane separates the two points by performing a Monte-Carlo simulation with 5×10^4 trials.

A scatter plot of the estimated probability vs Euclidean distance is shown in Figure 5.5. We also plot the theoretical probabilities as derived in Theorem 1. Figure 5.5 shows that the simulation results agree with our analysis as expected. Furthermore, the plot also verifies that good separation between corresponding and non-corresponding pairs can be obtained with an appropriately chosen Euclidean distance threshold.

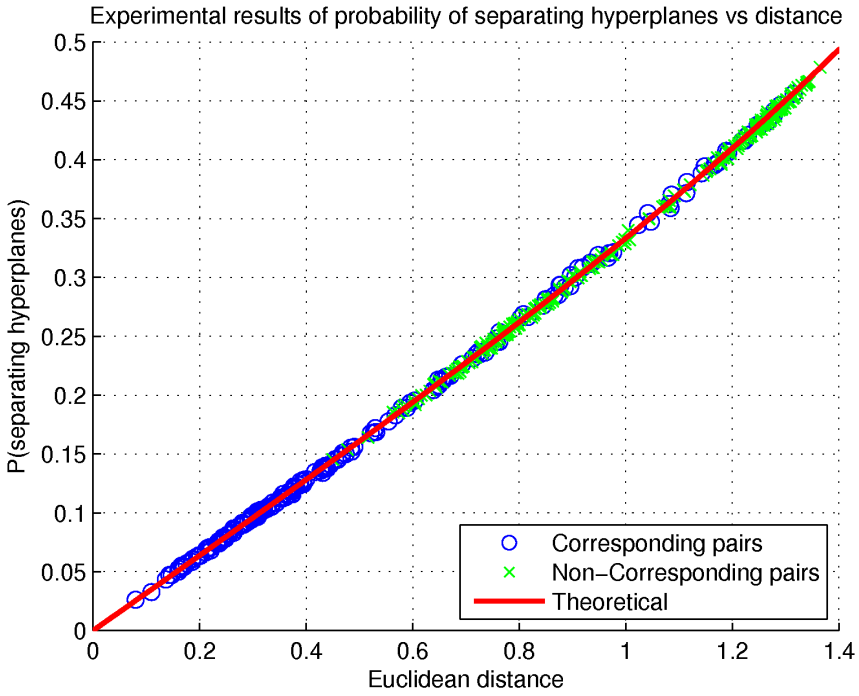


Figure 5.5. Simulation results demonstrating Theorem 1. We show the scatter plot of Euclidean distance between a pair of descriptors and the estimated probability of a randomly chosen hyperplane separating the pair for a randomly chosen subset of pairs of features. The x-axis is the actual Euclidean distance between the pair of descriptors, and the y-axis is the estimated probability of a randomly chosen hyperplane separating the descriptors. The blue circles represent pairs in correspondence, while green crosses represent pairs not in correspondence. The theoretical relationship between the two quantities is plotted in red. Note the close adherence to the theoretical result, and the good separation between corresponding and non-corresponding pairs.

5.4.3 Choosing the number of hash bits

Denote $\vec{d}^A$ and $\vec{d}^B$ to be binary-valued M-tuples formed by taking the M-bit binarized random projections hash of $\vec{D}^A$ and $\vec{D}^B$ respectively. Note that we have dropped the subscripts for clarity but we will use it when it is necessary to distinguish between various

features. From Corollary 1, the hamming distance between $\vec{d}^A$ and $\vec{d}^B$, $d_H(\vec{d}^A, \vec{d}^B)$, follows the binomial distribution and can be used as a test statistic in a hypothesis testing framework to decide if $\vec{D}^A$ and $\vec{D}^B$ satisfy the distance criterion.

Let p denote the probability of a randomly generated hyperplane separating $\vec{D}^A$ and $\vec{D}^B$ and let $p_\tau = \rho(\tau)$ (see Equation (5.12)). The hypotheses are:

$$H_0: \quad p > p_\tau + \mu/2 \quad (\text{i.e. } \|\vec{D}^A - \vec{D}^B\| > \tau)$$
$$H_1: \quad p < p_\tau - \mu/2 \quad (\text{i.e. } \|\vec{D}^A - \vec{D}^B\| < \tau)$$

where μ specifies an "insensitive" region around p_τ for which we would not measure performance. Since $d_H(\vec{d}^A, \vec{d}^B)$ has a binomial distribution, it is a monotone likelihood ratio (MLR) statistic [15]. Therefore, we can construct a uniformly most powerful (UMP) test of level α based on thresholding $d_H(\vec{d}^A, \vec{d}^B)$ with the following properties: the probability of falsely declaring a pair satisfying the distance criterion is always less than α while the probability of missing a pair satisfying the distance criterion is not more than any other tests of level α [15]. One reasonable choice for the threshold is:

$$\gamma_M = M \cdot p_\tau = \frac{2M}{\pi} \sin^{-1} \frac{\tau}{2} \tag{5.13}$$

To understand how many projections are needed for a test to satisfy a given error bound, we apply a Chernoff bound on the probability of false detection (declaring H_1 given H_0) and missed detection (declaring H_0 given H_1) of the hypothesis test. For example, given that $p > p_\tau + \mu/2$ (*i.e.* H_0),

$$P(\hat{H}_1|p, H_0) \leq \exp\left(-MD(p_\tau||p)\right) \tag{5.14}$$
$$\leq \exp\left(-MD(p_\tau||p_\tau + \mu/2)\right) \tag{5.15}$$

where $D(p||q)$ is the Kullback-Leibler divergence between two Bernoulli sources with parameter p and q, (5.14) follows from applying Chernoff bound and (5.15) follows from considering the worst case in H_0, which is when $p = p_\tau + \mu/2$. In this analysis, we assume the choice of threshold $\gamma_M = Mp_\tau$. A similar analysis also shows that $P(\hat{H}_0|H_1) \leq \exp\left(-MD(p_\tau||p_\tau - \mu/2)\right)$. These bounds can then be used to determine a suitable number of projections to use given a desired error bound.

Qualitatively, the above bounds tell us that the less stringent the matching criteria, *i.e.* the larger τ and hence p_τ is, the larger the number of projections needed to satisfy a target error, given the same absolute size of the "insensitive" region.

5.4.4 Using linear codes to reduce rate

In a related work, Körner and Marton [73] showed that if $\vec{d}^A$ and $\vec{d}^B$ are generated by binary symmetric sources related by a BSC with *known* cross-over probability p, then to recover the flip pattern, $\vec{Z} = \vec{d}^A \oplus \vec{d}^B$, with probability of failure less than ϵ, both Alice and Bob need to use a rate of at least $H(p)$ bits respectively (asymptotically). The achievable strategy uses a linear code and is as follows [73]: Let $f(\vec{Z})$ be a linear encoding function of the binary vector $\vec{Z}$ that returns K output bits from M input bits. Let $\psi(\cdot)$ be the decoding function of this linear code such that $P\left(\psi(f(\vec{Z})) \neq \vec{Z}\right) < \epsilon$. Alice and Bob then construct and transmit $f(\vec{d}^A)$ and $f(\vec{d}^B)$ respectively. A receiver can then construct $f(\vec{d}^A) \oplus f(\vec{d}^B) = f(\vec{d}^A \oplus \vec{d}^B) = f(\vec{Z})$, since $f(\cdot)$ is a linear code, and reconstruct $\vec{Z}$ with probability of failure less than ϵ. Thus, we can use this scheme as a way to obtain rate savings, using a rate of $H(p)$ instead of 1.

While the above scheme recovers the flip pattern $\vec{Z}$, Ahlswede and Csiszár showed that the above rate region in fact holds even if only the hamming distance is desired [4]. This also suggests that if we want to recover the hamming distance only when $p < p_\tau$ (but p is otherwise unknown), the best we can hope to do in a one-shot scenario, *i.e.* Bob just sends one message to Alice with no other interaction, is to use a rate of $H(p_\tau)$ and the method described earlier is an achievable strategy. The optimality of this scheme when we just want to know if the hamming distance is smaller than some threshold is an open question.

For a practical implementation used in this work, we use the parity-check matrix of a low-density parity-check (LDPC) code [54] as the linear encoding function [81, 86]; thus, the output $f(\vec{d}^A)$ is just the LDPC syndrome of $\vec{d}^A$. To decode, we apply belief-propagation (BP) decoding [109] on the XOR sum of the syndromes of $\vec{d}^A$ and $\vec{d}^B$, *i.e.* $f(\vec{d}^A) \oplus f(\vec{d}^B)$. We choose a code with blocklength M and rate r such that it has a threshold corresponding

to $\frac{\gamma_M}{M}$ [109]. To determine if the distance criterion is satisfied, decoding must converge[2] and the hamming weight of $\vec{Z}$ is less than γ_M.

5.4.5 Algorithmic summary

To summarize, the procedure for performing distributed distance testing is as follows. The user parameters are: n, the dimensionality of the real-valued source; M, the number of projections desired; and τ, the euclidean distance threshold (or equivalently $\gamma_M = M\rho(\tau)$). From these parameters, we generate a suitable LDPC code with K syndrome bits, *i.e.* with rate $(1 - \frac{K}{M})$, such that it has threshold $\frac{\gamma_M}{M}$, and obtain its parity check matrix $H \in GF(2)^{M\times K}$. We also generate a random projection matrix $L \in \mathbb{R}^{n\times M}$ with the kth column denoted by $\vec{l}_k$. Both H and L are shared by the encoder and decoder.

The encoder takes a vector $\vec{D}^A \in \mathbb{R}^N$ as input and returns a binary vector $\vec{m}_A \in GF(2)^K$. It performs the following:

1. Compute the binary random projections, $\vec{d^A}$, with the kth element being $d^A(k) = \mathbb{I}\left[\vec{l}_k \cdot \vec{D}^A > 0\right]$.
2. Compute the syndrome of $\vec{d^A}$, $\vec{m}_A = H^T\vec{d^A}$.

The decoder takes two binary vectors, $\vec{m}_A, \vec{m}_B \in GF(2)^K$ ($\vec{m}_B$ is obtained from $\vec{D}^B$ using the same encoder as described above) and returns H_1 if the distance criterion is satisfied by $\vec{D}^A$ and $\vec{D}^B$. Otherwise, it returns H_0. The decoding process is:

1. Compute $\vec{m}_z = \vec{m}_A \oplus \vec{m}_B$.
2. Perform BP decoding on the syndrome $\vec{m}_z$ to obtain reconstruction $\hat{Z} \in GF(2)^M$.
3. If BP decoding converges and $d_H(\hat{Z}) \leq \gamma_M$ then return H_1; else return H_0.

[2] We determine that it converges if the reconstruction satisfies the parity check matrix within 50 iterations.

5.4.6 Simulations

We now present results from a Monte-Carlo simulation of the following scenario to demonstrate the proposed approach. For each trial, we generate a vector $\vec{D}^A \in \left\{\vec{D} \in \mathbb{R}^{128} | \|\vec{D}\| = 1\right\}$ uniformly at random and perturb it by a random amount ($\sim$ unif$[0, 0.5]$) in a random direction to obtain $\vec{D}^B$ (normalized such that $\|\vec{D}^B\| = 1$). The distance criterion we are interested is whether $\|\vec{D}^A - \vec{D}^B\| < \tau = 0.2$. The corresponding probability of a separating hyperplane is $\rho(\tau) = 0.0638$. We evaluate the performance of three schemes in sending descriptors to determine if the vectors satisfy the given distance criterion: (i) Random projections (RP); (ii) Random projections with LDPC (RP-LDPC); and (iii) Scalar quantization (SQ). For each scheme, we perform 10000 trials and compute the precision, which is the fraction of retrieved pairs that satisfy the distance criterion, and recall, which is the fraction of all generated pairs satisfying the distance criterion that are retrieved, over various thresholds used. We also measure the rate used to transmit each vector. In the RP-LDPC scheme, we used a rate $\frac{1}{2}$ LDPC code which has an asymptotic threshold of 0.11.

Fig. 5.6 shows the ROC curves (precision vs recall) of RP-LDPC using different number of projections. As expected, as the number of projections increases, the retrieval performance of RP-LDPC increases.

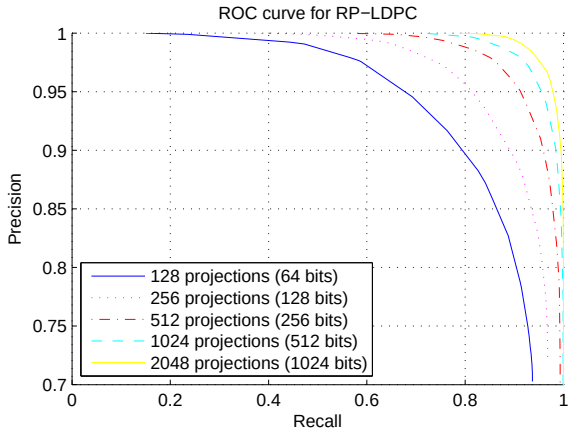


Figure 5.6. ROC over different bit rates for proposed scheme

Fig. 5.7 shows the ROC curves for RP-LDPC, RP and SQ using 256 bits per vector. Clearly, at this rate, RP-LDPC has the best performance, followed by RP and SQ. We also show the ROC curve for RP using 512 projections. Comparing it with RP-LDPC, there is almost no loss in performance in using RP-LDPC even though the rate required is halved.

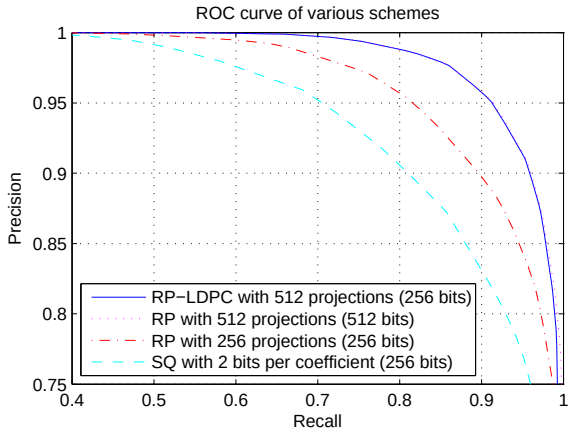


Figure 5.7. Comparison of ROC for various schemes. We show here the ROC for RP-LDPC, RP and SQ when 256 bits are used per vector. RP-LDPC has the best retrieval performance, followed by RP and SQ. RP-LDPC uses 512 projections; we show for reference the performance of RP which also uses 512 projections, but requiring double the rate. These two schemes have very similar performances.

Fig. 5.8 shows the maximum F1 score over all possible thresholds vs rate for the three schemes. This plot shows very clearly that at low rates, RP-LDPC and RP are preferable, while at high rates, SQ would be the right thing to do.

Finally, Fig. 5.9 shows the F1 score for RP and RP-LDPC using different number of projections when different thresholds are used. The thresholds shown are normalized by the number of projections used. It is clear that choosing γ_M as given by Equation (5.13) is a reasonable thing to do, particularly if we are interested in achieving the highest F1 score.

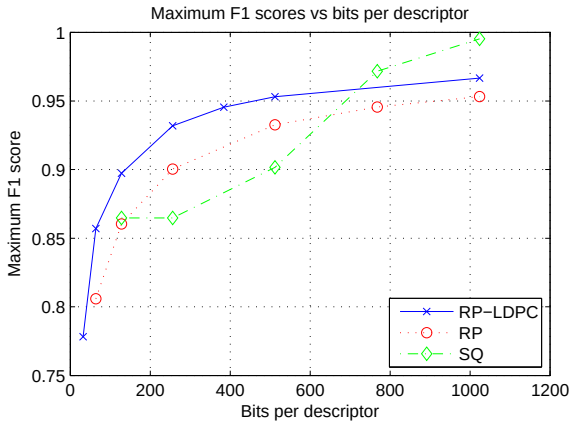


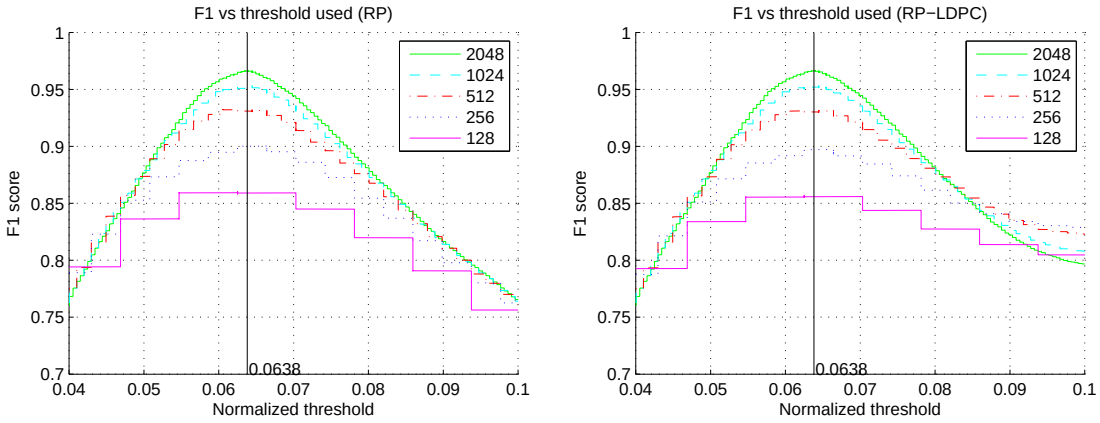
Figure 5.8. Comparison of maximum F1 scores for various schemes over different bitrates. We use the maximum F1 score to capture the best trade-off between recall and precision for each scheme and choice of parameters. The results here show that at all bit rates, RP-LDPC out-performs RP, since it is able to use the LDPC layer to reduce rate. On the other hand, at low rates, RP-LDPC out-performs SQ, while at high rates, SQ does better. This suggests that the choice of scheme depends on the rate regime.

5.5 Experimental Evaluations

5.5.1 Setup

We evaluate our proposed approaches on a standard benchmark dataset made publicly available[3] by Mikolajczyk and Schmid [91]. In particular, we consider the most challenging case of viewpoint changes where shots are taken of the same scene from different viewing angles with a viewpoint change of about 20 degrees between neighboring camera views. These are the "Graf" and "Wall" scenes, shown in Figure 5.10. Each image has dimensions of about 840×660. In "Graf", the images are taken of a planar scene, while in "Wall", the images are taken by a camera undergoing pure rotation. Due to geometric constraints in each of these cases, the image views are related by a homography [83]. The dataset also includes computed ground-truth homography which allows for ground-truth correspondence pairs to be extracted based on overlap error in the regions of detected features [91]. This leads naturally to a systematic performance evaluation of the task of establishing visual correspondences.

[3] `http://www.robots.ox.ac.uk/~vgg/research/affine`



(a) RP

(b) RP-LDPC

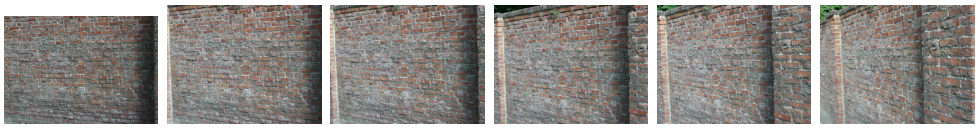
Figure 5.9. F1 scores vs threshold used for RP and RP-LDPC using different number of projections. We show how the F1 scores vary as the threshold used varies. Empirically, the results suggest that picking the threshold as $\gamma_M = M\rho(\tau)$ will give the best recall/precision trade-off.

Our evaluation procedure is as follows. We first run the Hessian-Affine feature detector to obtain a list of features in each image and then compute the SIFT descriptor for each feature. We note here that SIFT descriptors are normalized in the last step of computation to be robust to illumination changes and thus satisfy the unit-norm assumption used in the binarized random projections based schemes. We set the feature detector threshold such that it returns a maximum of 2000 features per image. Using the ground-truth homography and given the list of detected features in each image, we find the list of C_{total} ground-truth correspondences between those features. We encode and decode the descriptors from camera B using the following four procedures:

- **Baseline** – This consists of using a linear transform to de-correlate the descriptor, as discussed in Section 5.3.1, and then applying entropy coding on the quantized coefficients using an arithmetic coder. Decoding simply consists of undoing the above steps. Matches are found using the target *Euclidean matching criterion*. Different rate constraints can be satisfied by varying the quantization step size used.

- **DSC** – Descriptors are encoded using the encoding procedure outlined in Section 5.3.4. The received messages are decoded using descriptors from camera A as

(a) Graf



(b) Wall

Figure 5.10. Test dataset [91]. The data used for our tests are shown above: (a) "Graf"; and (b) "Wall". In "Graf", the different views are of a mostly planar scene, while in "Wall", the views are obtained by rotating the camera about its center. In both cases, the views are related by a homography [83].

side-information. Recall that matches are found when decoding is successful and meets the target *Euclidean matching criterion*. As in the baseline scheme, different rate constraints can be satisfied by varying the quantization step size used.

- **RP** – Descriptors are encoded using the binarized random projections discussed in Section 5.4 but without applying the linear code, *i.e.* the random projection bits are sent as is. Matches are found using a hamming distance threshold computed from the target *Euclidean matching criterion* using Equation (5.13). Different rate constraints can be satisfied by varying the number of projections used.

- **RP-LDPC** – Descriptors are encoded and decoded using the procedure described in Section 5.4.5. The received messages are decoded using the hashed descriptors from camera A as side-information. Recall that matches are found when BP decoding is successful and satisfies the target hamming distance threshold. As in the RP scheme, different rate constraints can be satisfied by varying the number of projections used.

In all cases, we note the rate, R, that is used. Each approach would return a list of C_{retrieve} retrieved correspondences and we compute C_{correct}, the number of correctly retrieved correspondences, using the ground-truth correspondence pairs obtained earlier. From these, we compute both the recall value, $Re = C_{\text{correct}}/C_{\text{truth}}$, and the precision value, $Pr = C_{\text{correct}}/C_{\text{total}}$, of the scheme. The recall indicates how many of the correspon-

dences present (given the list of detected features) can be found and the precision indicates how good the retrieved correspondences are. For example, when performing calibration, it is important to maintain high precision of the retrieved correspondences to ensure that outliers do not break the calibration procedure. To jointly quantify recall and precision, we use the balanced F-score, $F_1 = \frac{2 \times Re \times Pr}{Re + Pr}$, which is commonly used in the information retrieval literature [76].

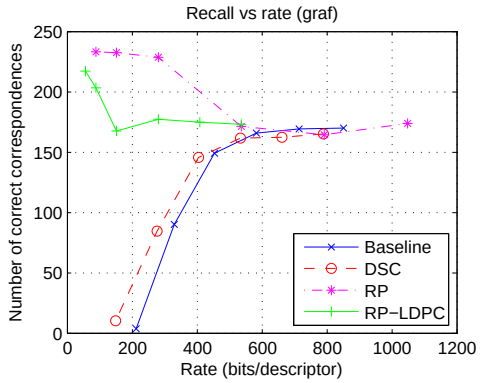
In our experiments, we consider both $\tau = 0.195$ ($\rho(\tau) = 0.0623$) and $\tau = 0.437$ ($\rho(\tau) = 0.1401$). For both the baseline and DSC schemes, we consider quantization step sizes ranging from 1.95×10^{-3} to 6.25×10^{-2}. In the DSC scheme, we use $\alpha = 1.718$ and a 24-bit CRC. For both the RP and RP-LDPC schemes, we vary the number of random projection used from 64 to 1024 (per descriptor). In the RP-LDPC scheme, we use a rate $(1 - 0.50)$ LDPC code when $\tau = 0.195$ and a rate $(1 - 0.73)$ LDPC code when $\tau = 0.437$.

5.5.2 Results

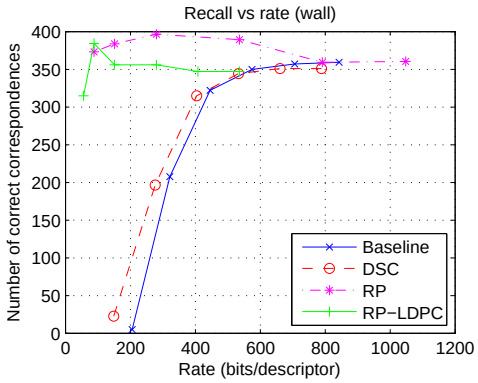
We present results averaged over all 5 pairs of neighboring views for each scene type. Figure 5.11 shows the rate-recall tradeoffs of the various schemes under consideration for an Euclidean Distance Criterion of $\tau = 0.195$ and $\tau = 0.437$ respectively. From Figures 5.11(a) and 5.11(b), at a lower threshold of $\tau = 0.195$, we see that in the baseline and DSC schemes, the number of correct correspondences retrieved increases with the amount of rate used. Furthermore, the DSC scheme always require less rate than the baseline scheme to obtain the same performance since it requires less rate to describe each descriptor. On the other hand, the number of correctly retrieved correspondences stay relatively stable over a wide range of rates in the RP and RP-LDPC schemes.

At a larger threshold of $\tau = 0.437$, Figures 5.11(c) and 5.11(d) shows that the baseline scheme now requires less rate than the DSC scheme. This is due to corresponding descriptors satisfying this larger threshold being less correlated. RP-LDPC still requires slightly less rate than RP due to the use of the linear code to further compress the binarized random projections. However, the baseline scheme outperforms both RP and RP-LDPC. As

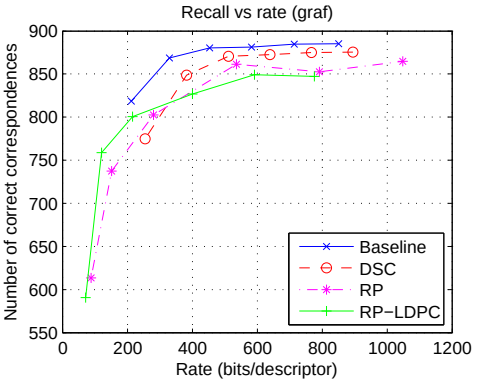
suggested by our analysis in Section 5.4.3, with a larger threshold, we would expect that more hash bits are needed to satisfy the same error bound.



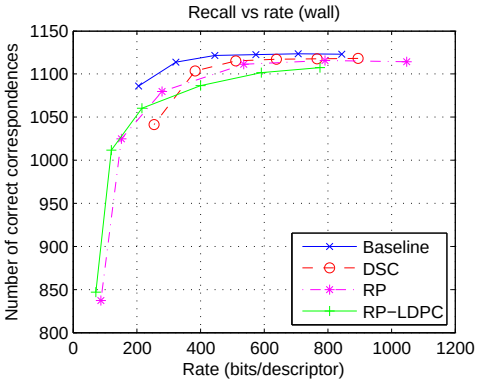
(a) C_{correct} vs rate - Graf ($\tau = 0.195$)



(b) C_{correct} vs rate - Wall ($\tau = 0.195$)



(c) C_{correct} vs rate - Graf ($\tau = 0.437$)

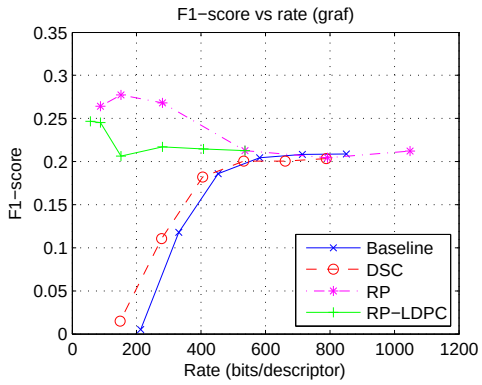


(d) C_{correct} vs rate - Wall ($\tau = 0.437$)

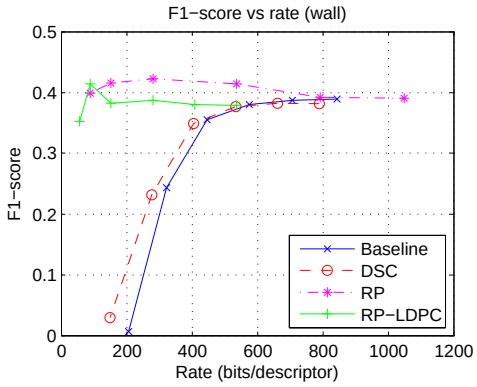
Figure 5.11. Rate-Recall tradeoff. The above plots show how the average number of correctly retrieved correspondences (C_{correct}) varies with rate. The results for "Graf" are shown in (a) and (c); that of "Wall" are shown in (b) and (d). In (a) and (b), a threshold of $\tau = 0.195$ is used, while in (c) and (d), a threshold of $\tau = 0.437$ is used.

Figure 5.12 shows how the F_1 score, a joint measure of recall and precision, varies with rate. At a low threshold, the DSC scheme performs better than the baseline scheme in requiring smaller rate for the same performance but this reverses at a higher threshold. In addition, the F_1 score is relatively stable over a range of rates for both the RP and RP-LDPC schemes at a low threshold – this implies that when a stricter criterion is necessary, one can get by with spending as little as 64 bits per descriptor. With a larger threshold, however, all the schemes appear to have a relatively similar F_1 performance over a wide

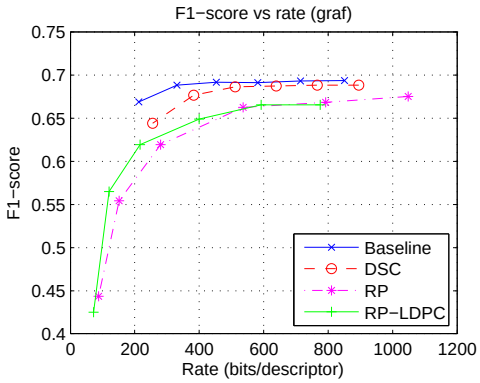
range of rates. At very low rates, RP-LDPC still requires slightly less rate than RP for the same performance.



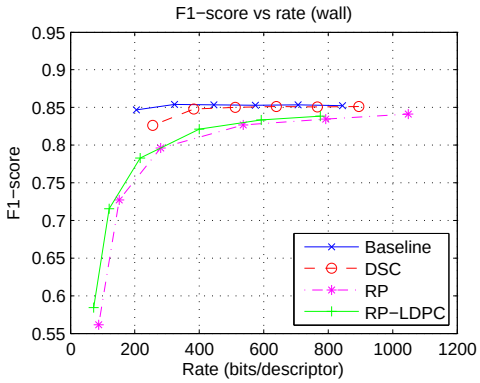
(a) F_1 vs rate - Graf ($\tau = 0.195$)



(b) F_1 vs rate - Wall ($\tau = 0.195$)



(c) F_1 vs rate - Graf ($\tau = 0.437$)



(d) F_1 vs rate - Wall ($\tau = 0.437$)

Figure 5.12. Rate-F_1 tradeoff. The above plots show how the F_1 score, a measure that takes into account both recall and precision performance, varies with rate. The results for "Graf" are shown in (a) and (c); that of "Wall" are shown in (b) and (d). In (a) and (b), a threshold of $\tau = 0.195$ is used, while in (c) and (d), a threshold of $\tau = 0.437$ is used.

We have also experimented with using the Portable Network Graphics (PNG) image format to compress the entire image losslessly prior to sending it. However, the rate used is much more (about an order of magnitude) than any of our proposed approaches and we do not show it in our above plots. Thus, all of our proposed approaches do better at utilizing bandwidth to establish correspondences than simply sending a lossless compressed version of the captured image.

In addition, recall that feature descriptors are usually high-dimensional. For example,

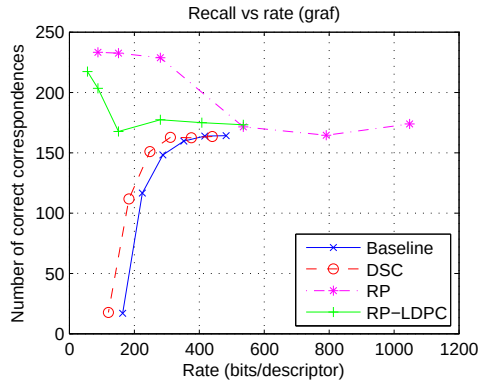
the SIFT descriptors used in our experiments are 128-dimensional. Since we use PCA to estimate the linear decorrelating transform needed in both the baseline and DSC schemes, the coefficients are already ordered according to their variances. Therefore, a possible way of further reducing rate is to perform dimensionality reduction by discarding the transformed descriptors coefficients with lower variance [26]. Since the number of dimensions is changed, there is a need to adjust the threshold as well. Here, we adjust the threshold proportionally to the fraction of remaining noise variances, *i.e.* $(\tau')^2 = \frac{\sum_{i=1}^{D'} \sigma_i^2}{\sum_{i=1}^{D} \sigma_i^2} \tau^2$, where τ' is the adjusted threshold, $D = 128$ is the original dimensionality of the descriptor and D' is the new dimensionality of the dimensionality reduced descriptor. Figure 5.13 shows results when we keep only the most dominant 64 coefficients of the transformed descriptor for the case when $\tau = 0.195$. Using DSC still gives significant performance gains over the baseline encoding. This suggests that the DSC framework can be successfully used in conjunction with dimensionality reduction via PCA.

Overall, in retrieving visual correspondences, all our proposed schemes outperform the baseline approach when a stringent matching criterion is used. Depending on the quantization used, the DSC scheme achieves a 6% to 30% rate savings over the baseline scheme with almost the same retrieval performance. Furthermore, the RP and RP-LDPC schemes respectively use up to 10× and 15× less rate than the baseline scheme. On the other hand, when a less stringent matching criterion is desired, our experimental results indicate that the baseline scheme would be the method of choice.

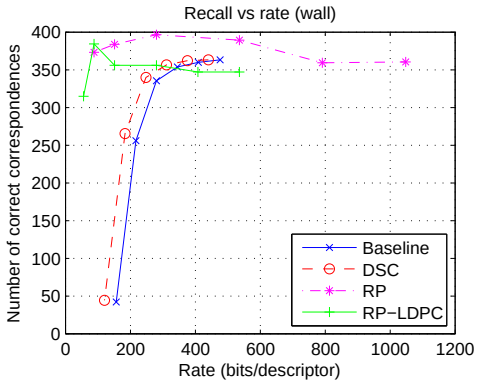
We note here that in comparing the two datasets, "Wall" seems to do better than "Graf", probably due to the richer scene texture. However, the relative performances of the different schemes remain the same. This suggests that regardless of the underlying scene statistics, the various proposed approaches can be applied successfully.

5.5.3 Effect on homography estimation

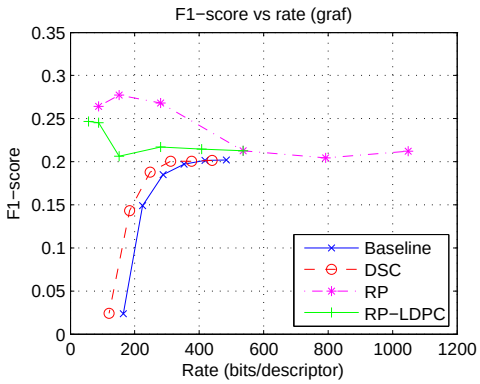
While a performance evaluation of visual correspondences retrieval is interesting in its own right, the retrieved list is typically used for some higher-level computer vision task



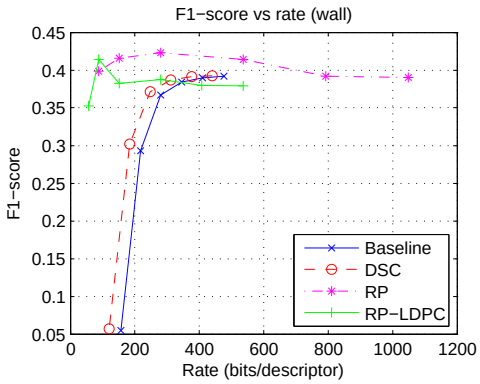
(a) C_{correct} vs rate - Graf ($\tau = 0.195$)



(b) C_{correct} vs rate - Wall ($\tau = 0.195$)



(c) F_1 vs rate - Graf ($\tau = 0.195$)



(d) F_1 vs rate - Wall ($\tau = 0.195$)

Figure 5.13. Rate-Performance tradeoff with dimensionality reduction. We can also apply the baseline and DSC schemes in conjunction with dimensionality reduction. Here, we keep only the first 64 coefficients after PCA. The above plots show how the average number of correctly retrieved correspondences (C_{correct}) varies with rate. The results for "Graf" are shown in (a) and (c); that of "Wall" are shown in (b) and (d). In (a) and (b), we show the rate-recall tradeoff, while in (c) and (d), we show the rate-F_1 tradeoff. A threshold of $\tau = 0.195$ is used.

such as camera calibration. We now briefly consider the performance of various schemes in homography estimation for two camera views.

The setup is almost the same as above. For each pair of neighboring views, we first find the list of correspondences between them using each of the methods under consideration. We then attempt to robustly fit a homography matrix by applying RANSAC[4] on the list of putative correspondences [62]. Using the final list of "good" matches, we first find a linear minimum mean square error estimate of the homography, followed by a non-linear optimization of the Sampson distance to arrive at the final estimate [62].

To quantify how good the homography estimate is, one could use the Frobenius norm of the difference between the estimate and the groundtruth. However, in our preliminary experiments, we found that it is not always a good indication of the goodness of the homography estimate. In particular, it does not quite capture how different the mapping of points between two images is. Instead, we use a measure that is inspired from the comparison of Fundamental matrices [156] and is aimed at capturing the difference between the homography mappings.

Assume two images, I_1 and I_2. Denote the groundtruth homography by H and the estimated homography by $\hat{H}$. The homography is the mapping of points in I_1 to points in I_2, i.e. if $\vec{p_1}$ and $\vec{p_2}$ are homogeneous coordinates of corresponding points in I_1 and I_2 respectively, then $\vec{p_2} \sim H\vec{p_1}$. The measure between H and $\hat{H}$, $d_{\text{proj}}(H, \hat{H})$, is computed as follows (see Figure 5.14 for an illustration) [156]:

1. Choose a random point p_1 in I_1.

2. Find the corresponding point p_2 in I_2 of p_1 in I_1 based on H. If the point is outside of the domain of I_2, go back to step 1

3. Find the estimated corresponding point $\hat{p_2}$ in I_2 of p_1 in I_1 based on $\hat{H}$ and compute the pixel distance, d_2, between p_2 and $\hat{p_2}$.

[4]RANSAC stands for "RANdom SAmple Consensus", which is an iterative procedure used to robustly estimate model parameters from a set of observed data that contains outliers [44].

4. Find the estimated corresponding point $\hat{p_1}$ in I_1 of p_2 in I_2 based on $\hat{H}$ and compute the pixel distance, d_1, between p_1 and $\hat{p_1}$.

5. Repeat steps 1 to 4 for T times.

6. Compute the measure as the average of the distances (d_1 and d_2) found above.

Note that the measure has a physical meaning in indicating on average how far apart mapped points would be using H vs $\hat{H}$. In our experiments, we choose $T = 50000$.

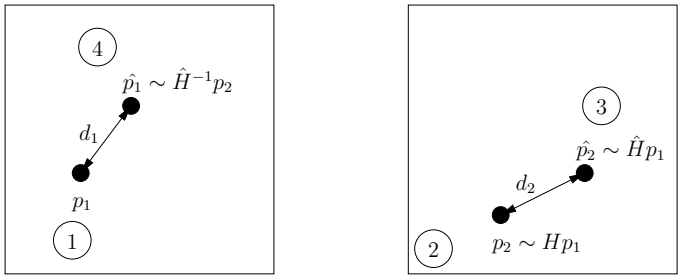
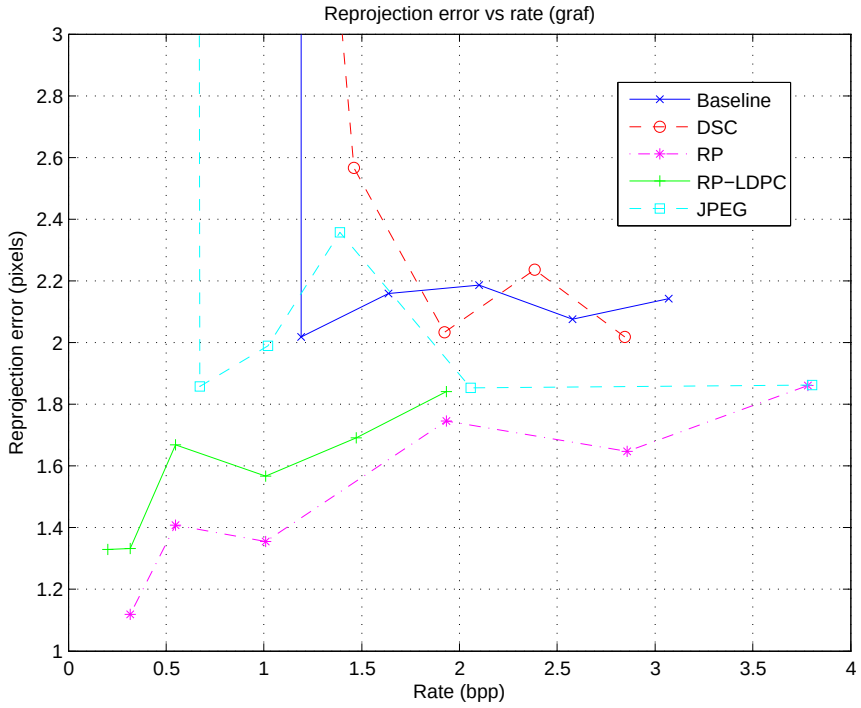


Figure 5.14. Measure of homography difference. A point, p_1, is picked at random in image I_1. Its corresponding point, p_2 in image I_2, is computed according to homography H. Similarly, its estimated corresponding point, $\hat{p_2}$ in image I_2, is computed according to estimated homography $\hat{H}$. The estimated corresponding point of p_2 in image I_2, $\hat{p_1}$ in image I_1, is also computed using $\hat{H}$. The distances d_1 and d_2 are computed between point pairs $(p_1, \hat{p_1})$ and $(p_2, \hat{p_2})$ respectively. The measure of homography difference between H and $\hat{H}$ is then computed as the average of distances d_1 and d_2 over a large number of points.

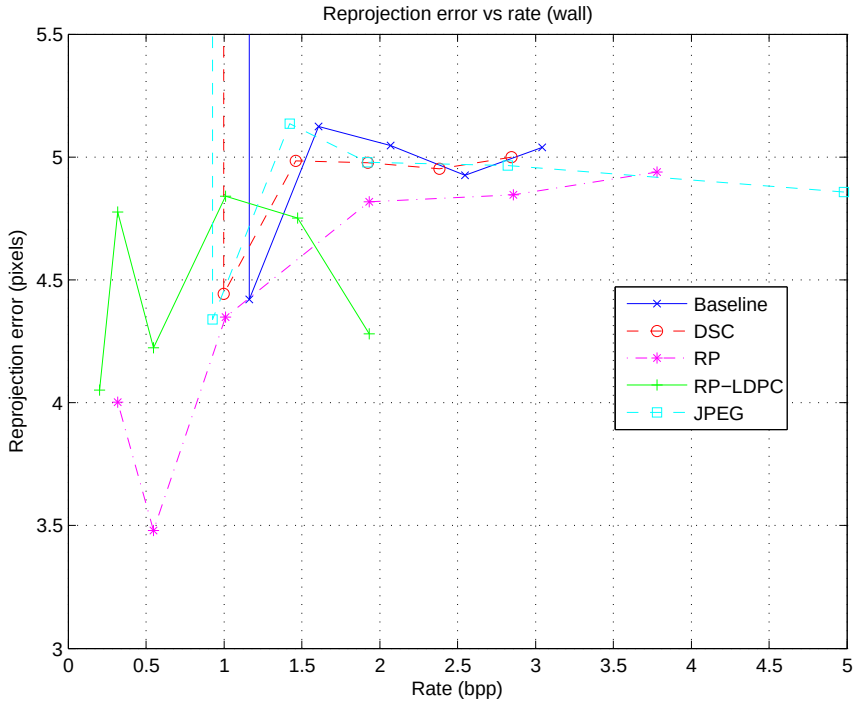
We measure $d_{\text{proj}}(H, \hat{H})$ for all schemes listed in Section 5.5.1. For comparison, we also use JPEG compression to reduce the rate of images before sending it, where rates are varied by changing the quality factor of the compression. All schemes use 2000 features with the highest "corneredness" score to first find visual correspondences before estimating homography.

Figure 5.15 shows the results when a stringent threshold of $\tau = 0.195$ is used. We see that both the RP and RP-LDPC schemes achieve smaller projection errors than the other schemes. In addition, RP-LDPC achieves the same projection errors using a lower rate than RP. On the other hand, using JPEG, the baseline scheme or the DSC scheme gives similar homography estimation performance, although at low rates, JPEG does a little worse.

Figure 5.16 shows the results when a threshold of $\tau = 0.437$ is used. In part because



(a) Graf



(b) Wall

Figure 5.15. Effect of visual correspondences on homography estimation (using $\tau = 0.195$).

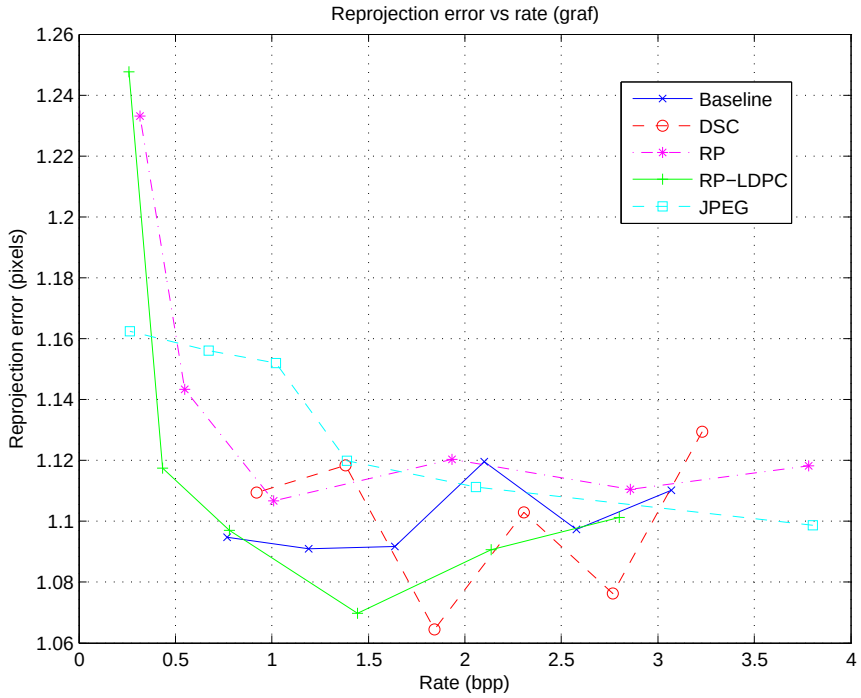
more correspondences are retrieved, the reprojection errors are on average smaller than when a more stringent threshold is used. While the JPEG scheme shows significantly worse performance at very low rates, all other schemes seem to have very similar performance. It appears that the effectiveness of RANSAC at eliminating outliers has leveled the field for all the schemes.

5.6 Recapitulation

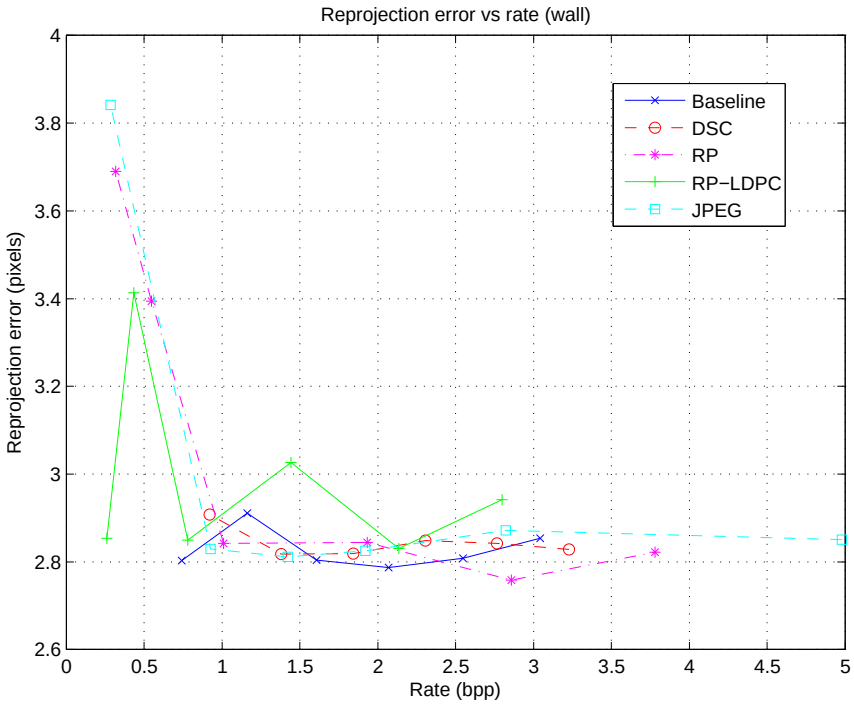
We have presented two constructive solutions for determining in a distributed fashion and under severe rate constraints if two normalized real vectors satisfy a given Euclidean distance criterion. This is an important step in performing camera calibration in a wireless camera network where communication costs are significant. The transmission of descriptors instead of compressed images in a distributed setting also prevents redundant computations since each camera only needs to perform feature extraction for the images that it captures. While we use a two terminal setup for sake of discussion, both proposed frameworks can be easily extended to a multiple cameras scenario. Furthermore, they can be generally used with any combination of feature detector and descriptor.

One approach applies DSC on feature descriptors to determine visual correspondences between two cameras in a distributed and rate-efficient fashion. Our results are encouraging; to encode each descriptor, the proposed DSC approach is able to achieve a bit-rate reduction of 6% to 30%, depending on the target quantization step size used, compared to a baseline scheme that simply entropy codes the descriptors. To retrieve the same number of correct correspondences, the proposed DSC scheme also requires less rate than the baseline encoding scheme.

Another scheme uses binarized random projections to convert the problem into a binary hypothesis testing problem and then obtain rate savings by applying a linear code to the computed bits. The rate to use for the code can be easily determined by the desired Euclidean distance threshold. Our experiments show that in determining visual correspondences, the binarized random projections approach often gives a better rate-performance



(a) Graf



(b) Wall

Figure 5.16. Effect of visual correspondences on homography estimation (using $\tau = 0.437$).

tradeoff than the baseline or DSC scheme. The same also holds when we consider the task of homography estimation.

Building hashes from binarized random projections has also been used in a video file synchronization application. Video hashing can be used to first determine which group of pictures (GOP) are in common between the source and destination videos [154, 155]. For example, Alice has a video which she gives to Bob who compresses it for storage. Later, Alice updates her copy of the video and Bob wishes to synchronize his copy. To avoid sending frames that Bob already has, Alice wishes to know which frames of Bob are within a target distortion of video frames of her copy — these frames need not be re-transmitted.

Along the lines of the binarized random projections approach, in future work, we would like to remove the same norm constraints and consider other useful source vector distributions and distance measures. We have not explored any security properties of our scheme, but we think that the proposed scheme offers some inherent security, due to the data obfuscation performed by both the binarized random projections and the syndrome coding [86].

Part III

Efficient video analysis for multiple camera streams

Chapter 6

Compressed domain video processing

The use of video cameras has become increasingly common as their costs decrease. In personal applications, it is common for people to record and store personal videos that comprise various actions, in part due to the widespread availability of phone cameras and cheap cameras with video recording capabilities. In security applications, multiple video cameras record video data across a designated surveillance area. A good example of this is the large network of surveillance cameras installed in London. Such proliferation of video data naturally leads to information overload. It would not only be incredibly helpful but also necessary to be able to perform rudimentary action recognition in order to assist the users in focusing their attention on actions of interest as well as allowing them to catalog their recorded videos easily.

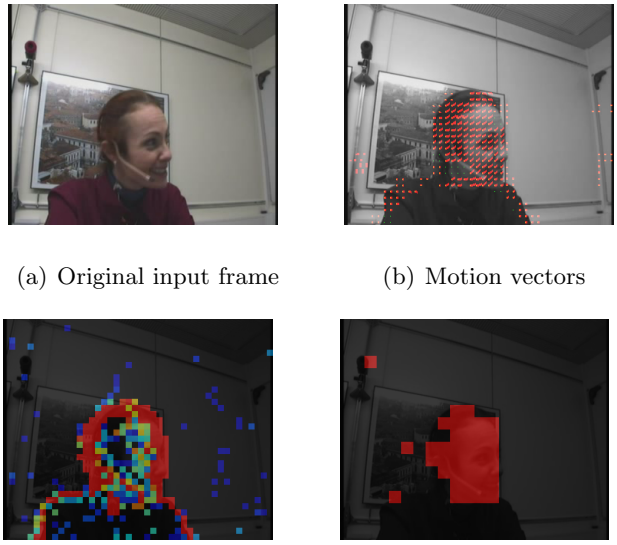
In addition, there has been a trend towards instrumenting meeting rooms with multiple microphones and cameras. Such a setup not only lends itself easily to teleconferencing applications, but also makes it easy to record meetings for analysis, evaluation and archived retrieval. It would be desirable to 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 batch processing.

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 focused on (i) synthetic video analysis, such as video shot detection [150, 139, 89] and text caption extraction [157]; (ii) video indexing, querying and retrieval [72]; (iii) synthetic video manipulation, such as video resizing [38] and transcoding applications [135]; and (iv) optical flow estimation [27]. In this part of the dissertation, we investigate the application of these techniques to new problem domains such as action recognition and organization and video analysis of meetings.

In the remainder of this chapter, we discuss some of the compressed domain video features that can be efficiently extracted from compressed videos. This of course relies on video coding, the background of which we discussed in Chapter 2 (Section 2.1).

6.1 Compressed domain features

Fig. 6.1 shows a preview of the various compressed domain features that can be extracted cheaply from compressed videos.



(a) Original input frame (b) Motion vectors

(c) Residual coding bit-rate (d) Detected skin blocks

Figure 6.1. Example output from compressed domain feature extraction

6.1.1 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 block. 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 [139]. 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 proposed by Yeo and Liu to build spatially sub-sampled frames from inter-coded frames, forming a DC image sequence [139]. Suppose that a block in the current frame overlaps with 4 blocks in the reference frame, $S = \{a, b, c, d\}$, as in Figure 6.2. Let $f_i, i \in S$, be 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 estimated as:

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

This gives reasonable results if the GOP size is kept relatively small (about 9-15).

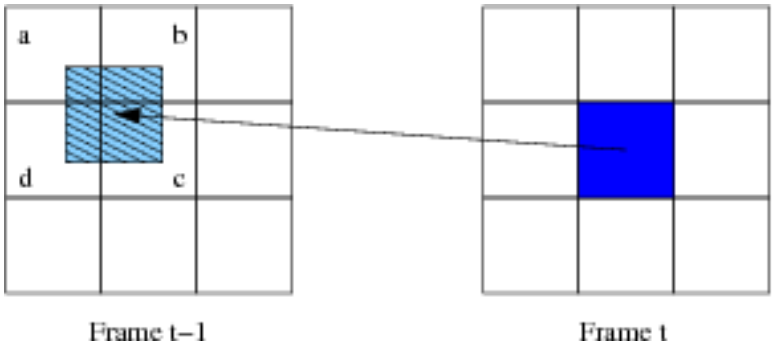


Figure 6.2. Illustration of how DCT DC terms are updated using motion vectors and residual. The block to be reconstructed in frame t, shown with a solid fill, is predicted by a block in frame $t-1$, shown with stripes. In general, the predictor overlaps with 4 blocks, labeled as $\{a, b, c, d\}$ here. The update is computed by considering the amount of overlap with each block, with an additional correction term due to the residue.

6.1.2 Motion vectors

Motion vectors, shown in Figure 6.1(b), are generated from motion compensation during video encoding. As explained earlier in Section 2.1 and illustrated in Figure 2.2, for each source block that is encoded in a predictive fashion, its motion vectors indicate which predictor block from the reference frame, typically the previous frame in time, is to be used. Presumably, a predictor block is highly similar to the source block. Therefore, motion vectors are a good, albeit coarse, approximation of optical flow, which in turn is a proxy for the underlying motion of objects in the video [27]. 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.

We follow the approach outlined by Coimbra and Davies [27] for computing a coarse estimate and a confidence map of the optical flow. To generate the optical flow estimate, we use the following rules [27]:

1. Motion vectors are normalized by the temporal distance of the predicted frame to the reference frame, and their directions are reversed if the motion vectors are forward-referencing.

2. Macroblocks with no motion vector information (e.g. macroblocks in I-frames and intra-coded macroblocks) retain the same optical flow estimate as in the previous temporal frame.

3. Macroblocks with more than one motion vector (e.g. bi-directionally predicted macroblocks in B-frames) take as the estimate a weighted average of the motion vectors, where the weights are determined by their temporal distance to the respective reference frames.

It has been recognized that optical flow estimation performance at each image location depends on the amount of texture in its local neighborhood [12]. In particular, if the local neighborhood suffers from the aperture problem, then it is likely to have an unreliable optical flow estimate. By thresholding a confidence measure derived from the DCT AC

coefficients that measures the amount of texture in the block [27], we can filter out optical flow estimates that are likely to be unreliable. To compute the confidence measure for intra-coded blocks, we use [27]:

$$\lambda = F(0,1)^2 + F(1,0)^2 \tag{6.1}$$

where λ is the confidence measure, and $F(u,v)$ is the 2D DCT of the block of pixel luminance values, $f(x,y)$. Coimbra and Davies have shown that $F(1,0)$ and $F(0,1)$ can be interpreted as a weighted average of spatial gradient in the x and y direction respectively [27]. For predicted macroblocks, we update the confidence map by taking a weighted average of the confidence map in the reference frame(s) as indicated by motion vector information; this is similar to how DC images are formed as described in Section 6.1.1.

By thresholding λ, we then decide whether to keep the optical flow estimate for the block or to set it to zero.

6.1.3 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, because a temporal change that is not well-modeled by the block translational model will result in a residual with higher energy, which in turn requires a larger number of bits when entropy coded. Hence, this is complementary to the extracted motion vectors.

6.1.4 Skin-color blocks

By putting together some of the above 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 hands in analysis of meetings, 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 [88] 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 6.1(d).

We use a GMM to model the distribution of chrominance coefficients [88] 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 [31] (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:

$$p_{U,V|\text{skin}}(u, v|\text{skin}) > \tau \tag{6.2}$$

Since MPEG-4 uses a YUV colorspace for encoding, there is no need for any additional steps to perform color-space conversion. Furthermore, because the chrominance DCT coefficients are quantized during video compression, we can use a look-up table (LUT) approach to increase 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. This is similar to an approach for object tracking in the compressed domain [41]. However, in the presence of long GOPs, 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 having skin-color, using the same criterion as in (6.2). Note that this verification step only has to be done if a block is suspected to have skin-color.

We can also apply the GMM model to chrominance DCT DC coefficients estimated using the method described in Section 6.1.1. 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 just described gives much better performance.

Chapter 7

Activity recognition and organization

In this chapter, we present a compressed domain scheme that is able to recognize and localize actions at high speeds. We formulate the problem of action recognition and localization as follows: given a query video sequence of a particular action, we would like to detect all occurrences of it in a test video, thereby recognizing an action as taking place at some specific time and location in the video. The approach should be person independent, so we want our method to be appearance invariant. In a surveillance setting, it is critical to be able to respond to events as they happen. Even in a consumer application, it is desirable to minimize processing time. Therefore, we want a solution that is fast so it can operate in real-time.

Any practical system that records and stores digital video is likely to employ video compression such as H.263+ [28] or H.264 [136]. It has long been recognized that some of the video processing for compression can be reused in video analysis or transcoding; this has been an area of active research (see for example [20, 135]) in the last decade or so. Our approach exploits this insight to attain a speed advantage.

It is reasonable to assume that a surveillance application would consist of a front-end system that records, compresses, stores and transmits videos, as well as a back-end system

that processes the transmitted video to accomplish various tasks. One focus in this paper is on the action recognition task that would presumably be performed at the back-end. However, we recognize that various engineering choices, such as the choice of video coding method, made at the front-end can have an impact on the action recognition performance in the back-end. In particular, we would like to understand how various video coding choices impact the action recognition performance of our approach.

The work presented in this chapter is joint work with Parvez Ahammad, Kannan Ramchandran and S Shankar Sastry, and has been presented in part in [143, 144, 3].

7.1 Related work

There has been much prior work in human action recognition; an excellent review of such methods has been presented by Aggarwal and Cai [2]. We are interested in approaches that work on video without relying on capturing or labeling body landmark points (see [152, 101] for recent examples of the latter approach). Efros *et al.* [40] require the extraction of a stabilized image sequence before using a rectified optical flow based normalized correlation measure for measuring similarity. This stabilization step required by [40] is a very challenging pre-processing step and affects the end result significantly. Shechtman and Irani [117] exhaustively test motion-consistency between small space-time (ST) image intensity patches to compute a correlation measure between a query video and a test video. While their method is highly computationally intensive, they are able to detect multiple actions (similar or different) in the test video and also perform localization in both space and time. Ke *et al.* [71] also use an image intensity based approach, but apply a trained cascade of classifiers to ST volumetric features computed from image intensity. Schüldt *et al.* [114] propose an approach based on local ST features [75] in which Support Vector Machines (SVM) are used to classify actions in a large database of action videos that they collected. Dollar *et al.* [35] adopt a similar approach, but introduce a different spatio-temporal feature detector which they claim can find more feature points.

There has also been prior work in performing action recognition in the compressed do-

main. Ozer *et al.* [100] applied Principal Component Analysis (PCA) on motion vectors from *segmented* body parts for dimensionality reduction before classification. They require that the sequences must have a fixed number of frames and be *temporally aligned*. Babu *et al.* [10] trained a Hidden Markov Model (HMM) to classify each action, where the emission is a codeword based on the histogram of motion vector components of the *whole* frame. In later work [11], they extracted Motion History Image (MHI) and Motion Flow History (MFH) [30] from compressed domain features, before computing global measures for classification. In [10, 11], the use of global features precludes the possibility of localizing actions with these compressed domain methods.

7.2 Contributions

Our proposed method makes use of motion vector information to capture the salient features of actions which are appearance invariant. It then computes frame-to-frame motion similarity with a *novel* measure that takes into account differences in both orientation and magnitude of motion vectors. The scores for each space-time candidate are then aggregated over time using a method similar to [40]. Our approach is able to localize actions in space and time by checking all possible ST candidates, much like in [117], except that it is more computationally tractable since the search space is greatly reduced from the use of compressed domain features. Our innovation lies in the ability of the proposed method to perform *real-time* localization of actions in space and time using a novel combination of signal processing and computer vision techniques. This approach requires no prior segmentation, no temporal or spatial alignment (unlike [40, 100]) and minimal training. Unlike in [40, 114, 71, 35], we also do not need to compute features explicitly; features are readily available in the compressed video data. We have to emphasize the fact that our action similarity computation is much faster than methods such as in [117], making possible applications such as content-based video organization for large-scale video databases (see Section 7.6).

We also study how various encoding options affect the performance of our proposed approach. This aspect is often overlooked in most other compressed domain video analysis

work, in which results are typically presented only on a single choice of encoding parameters. However, we recognize that different encoding options not only affect compression performance but also influence the performance of compressed domain processing. Hence, in this work, we undertake a systematic investigation to determine the trade-offs between compression performance and classification performance. This would be useful in understanding how best to choose encoding options to strike a good balance between compression and classification, and between speed and accuracy.

The rest of the chapter is organized as follows. Section 7.3 outlines our proposed method and describes each step in detail. The experimental setup and results are discussed in Section 7.4, and we discuss the effects of different video encoding options in Section 7.5. We show in Section 7.6 how the action similarity measure that is introduced can be used in the application of organizing activity videos. We then present concluding remarks in Section 7.7.

7.3 Approach

Given a query video template and a test video sequence, we propose a compressed domain procedure to compute a score for how confident we are that the action presented in the query video template is happening at each space-time location (to the nearest macroblock and frame) in the test video. Our working assumption is that similar actions will induce similar motion fields.

The steps of the algorithm are summarized in the flow chart shown in Figure 7.1. We will elaborate on each of these steps in the following subsections.

7.3.1 Notation

In this chapter, X^p denotes a video, with $p \in \{\text{test}, \text{query}\}$ referring to either the test video or query video. Each video X^p has T^p frames, with each frame containing $N^p \times M^p$ macroblocks. We assume that an *action* induces a motion field that can be observed as

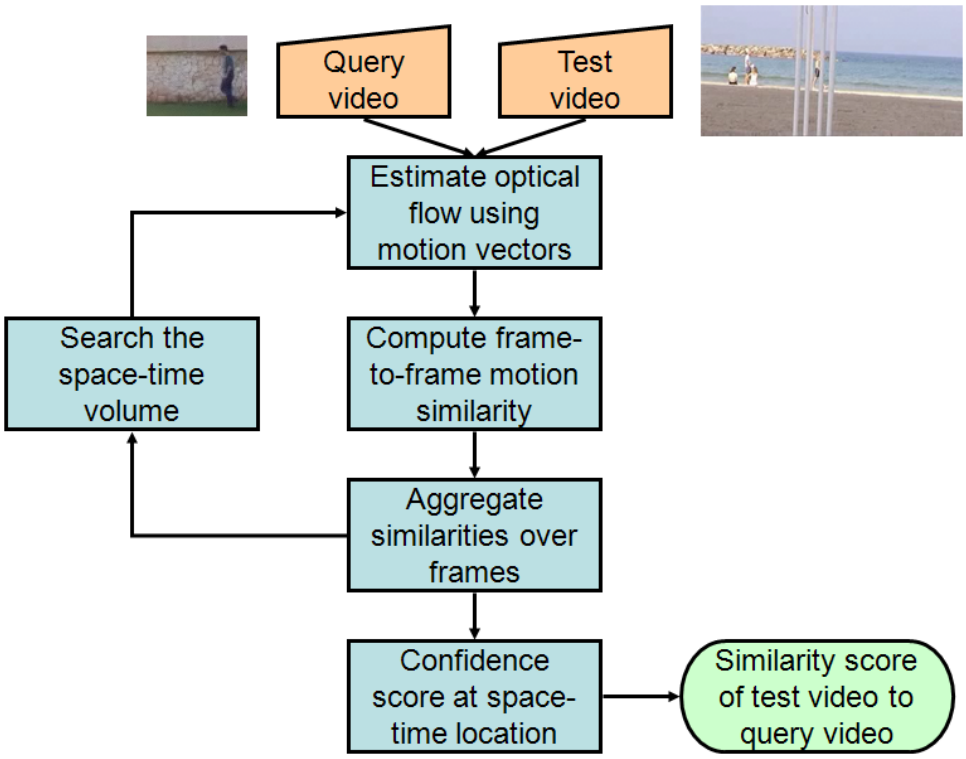


Figure 7.1. Flow chart of action recognition and localization method. Optical flow in the query and test videos are first estimated from motion vector information. Next, frame-to-frame motion similarity is computed between all frames of the query and test videos. The motion similarities are then aggregated over a series of frames to enforce temporal consistency. To localize, these steps are repeated over all possible space-time locations. If an overall similarity score between the query and test videos is desired, a final step is performed with the confidence scores.

a spatio-temporal pattern; let $\vec{V}^p$ be the spatio-temporal pattern (motion field) associated with video X^p. Furthermore, $\vec{V}^p_{n,m}(i) = [V^{p,u}_{n,m}(i) \quad V^{p,v}_{n,m}(i)]$ denotes the motion vector at location (n, m) in frame i of X^p. We will use $(\boldsymbol{u})_+$ as a shorthand for $\max(0, \boldsymbol{u})$.

7.3.2 Estimation of coarse optical flow

We use the method described in Section 6.1.2 of Chapter 6 to obtain $\vec{V}^p$ from the encoded motion vectors in the compressed video. In particular, we also threshold the confidence measure, which is given by Equation (6.1), of the optical flow estimate for each macroblock to decide whether to keep it or to set it to zero. In our experiments, we use a threshold of

4096. As we will show later in Section 7.4.2, this thresholding removes unreliable estimates and greatly improves the classification performance of our proposed algorithm.

7.3.3 Computation of frame-to-frame motion similarity

For the purpose of discussion in this section, both the test frame and query frame are assumed to have a spatial dimension of $N \times M$ macroblocks (the equal size restriction will be lifted later). We would like to measure the motion similarity between the motion field of the ith test frame, $\vec{V}^{\text{test}}_{n,m}(i)$, and that of jth query frame, $\vec{V}^{\text{query}}_{n,m}(j)$.

One way of measuring similarity is to follow the approach taken by Efros *et al.* [40]. Each motion field is first split into non-negative motion channels, *e.g.* $(V^{p,u}_{n,m}(i))_+$, $(-V^{p,u}_{n,m}(i))_+$, $(V^{p,v}_{n,m}(i))_+$ and $(-V^{p,v}_{n,m}(i))_+$ using the notation described in Section 7.3.1. We can then vectorize these channels and stack them into a single vector $\vec{U}^p(i)$. The similarity between frame i of the test frame and frame j of the query frame, $\tilde{S}(i,j)$, is then computed as a normalized correlation:

$$\tilde{S}(i,j) = \frac{\langle \vec{U}^{\text{test}}(i), \vec{U}^{\text{query}}(j) \rangle}{\|\vec{U}^{\text{test}}(i)\| \|\vec{U}^{\text{query}}(j)\|} \tag{7.1}$$

We will refer to this similarity measure as *Non-negative Channels Normalized Correlation* (NCNC).

NCNC does not take into account the differences in magnitudes of individual motion vectors. To address this, we propose a novel measure of similarity:

$$\tilde{S}(i,j) = \frac{1}{Z(i,j)} \sum_{n=1}^{N} \sum_{m=1}^{M} d(\vec{V}^{\text{test}}_{n,m}(i), \vec{V}^{\text{query}}_{n,m}(j)) \tag{7.2}$$

where if $\|\vec{V_1}\| > 0$ and $\|\vec{V_2}\| > 0$,

$$\begin{aligned} d(\vec{V_1}, \vec{V_2}) &= \frac{\left(\langle \vec{V_1}, \vec{V_2} \rangle\right)_+}{\|\vec{V_1}\| \|\vec{V_2}\|} \cdot \min\left(\frac{\|\vec{V_1}\|}{\|\vec{V_2}\|}, \frac{\|\vec{V_2}\|}{\|\vec{V_1}\|}\right) \qquad (7.3) \\ &= \frac{\left(\langle \vec{V_1}, \vec{V_2} \rangle\right)_+}{\max\left(\|\vec{V_2}\|^2, \|\vec{V_1}\|^2\right)} \end{aligned}$$

and $d(\vec{V_1}, \vec{V_2}) = 0$ otherwise. In line (7.3), the first and second terms measure the similarity in direction and magnitude of corresponding motion vectors respectively. The normalizing

factor, $Z(i,j)$, in Equation (7.2) is:

$$Z(i,j) = \sum_{n=1}^{N}\sum_{m=1}^{M} \mathbb{I}[\|\vec{V}^{\text{test}}_{n,m}(i)\| > 0 \text{ or } \|\vec{V}^{\text{query}}_{n,m}(j)\| > 0]$$

In other words, we want to ignore macroblocks in both the query and test video which agree on having no motion. This has the effect of not penalizing corresponding zero-motion regions in both the query and test video. We term this novel measure *Non-Zero Motion block Similarity* (NZMS).

7.3.4 Aggregation of frame-to-frame similarities

Section 7.3.3 describes how to compute $\tilde{S}(i,j)$, which tells us how similar the motion fields of frame i of the test frame and frame j of the query frame are. To take temporal dependencies into account, we need to perform an aggregation step. We do this by convolving $\tilde{S}(i,j)$ with a $T \times T$ filter parametrized by α, $H_\alpha(i,j)$, to get an aggregated similarity matrix $S(i,j) = (\tilde{S} * H_\alpha)(i,j)$ [40]. $S(i,j)$ tells us how similar a T-length sequence centered at frame i of the test video is to a T-length sequence centered at frame j of the query video. $H_\alpha(i,j)$ can be interpreted as a bandpass filter that "passes" actions in the test video that occur at approximately the same rate as in the query video. We use the following filter [40]:

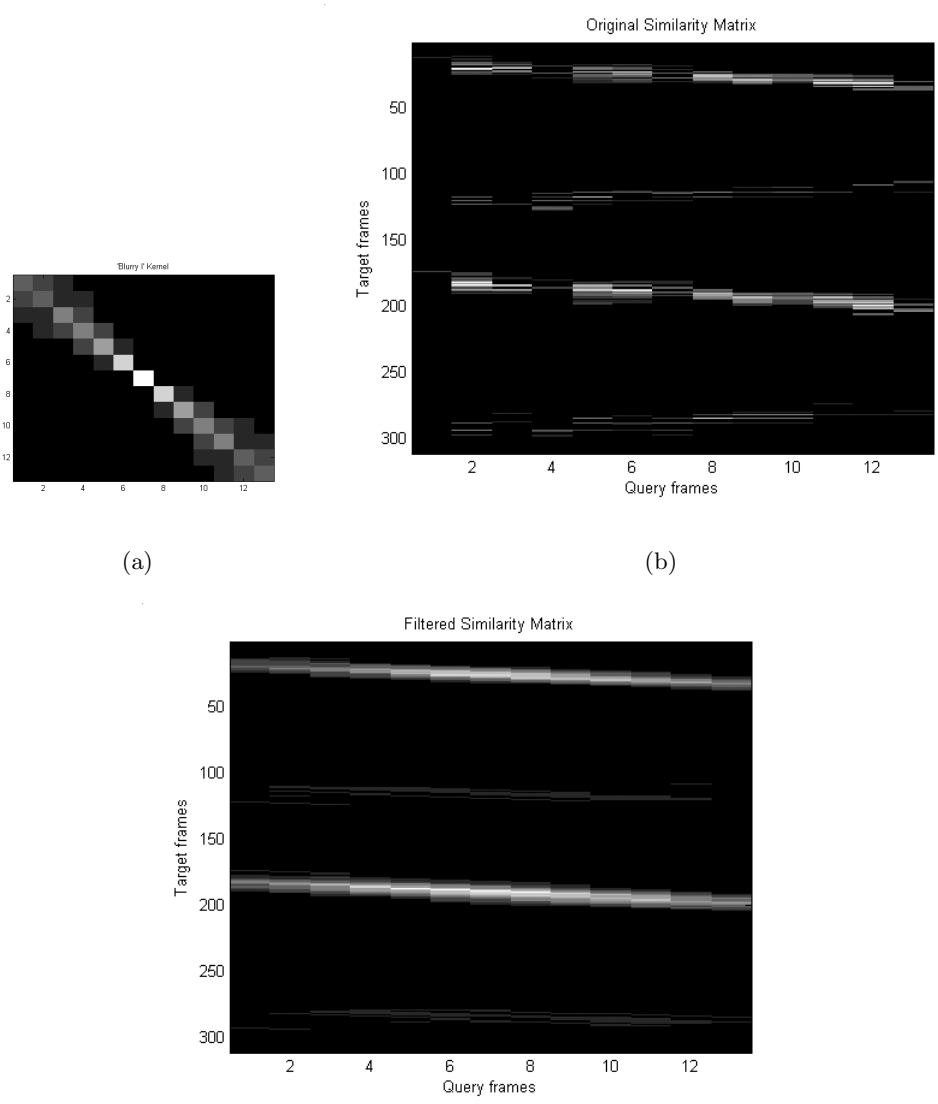
$$H_\alpha(i,j) = \sum_{r \in R} e^{-\alpha(r-1)} \left(\chi(i,rj) + \chi(j,ri)\right) \text{ for } -T/2 \leq i,j \leq T/2$$

where

$$\chi(u,v) = \begin{cases} 1 & \text{if } u = \text{sign}(v) \cdot \lfloor |v| \rfloor \\ 0 & \text{otherwise} \end{cases}$$

R is the set of rates (which has to be greater than one) to allow for and α ($\alpha \geq 1$) allows us to control how tolerant we are to slight differences in rates; the higher α is, the less tolerant it is to changes in the rates of actions. Figure 7.2(a) shows this kernel graphically for $\alpha = 2.0$.

Figure 7.2(b) shows a pre-aggregation similarity matrix, $\tilde{S}(i,j)$. Note the presence of near-diagonal bands, which is a clear indication that the queried action is taking place in



(a) (b)

(c)

Figure 7.2. An example similarity matrix and the effects of applying aggregation. In these graphical representations, bright areas indicate a high value. (a) Aggregation kernel, (b) Similarity matrix before aggregation, (c) Similarity matrix after aggregation. Notice that the aggregated similarity matrix is less noisy than the original similarity matrix.

those frames. Figure 7.2(c) shows the post-aggregation similarity matrix, $S(i, j)$, which has much smoother diagonal bands.

We will show later in Section 7.4.3 that this aggregation step is crucial in performing action classification. However, the choice of α is not that important; experimental results show that performance is relatively stable over a range of α.

7.3.5 Space-time localization

Sections 7.3.3 and 7.3.4 tell us how to compute an aggregated similarity between each frame of a T^{test}-frames test sequence and each frame of a T^{query}-frames query sequence, both of which are $N \times M$ macroblocks in spatial dimensions. To compute an overall score on how confident we are that frame i of the test frame is from the query sequence, we use:

$$C(i) = \max_{\substack{\max(i-\frac{T}{2},1) \le k \le \min(i+\frac{T}{2},T^{\text{test}}) \\ 1 \le j \le T^{\text{query}}}} S(k, j) \tag{7.4}$$

Maximizing $S(k, j)$ over j of the query video allows us to pick up the best response that a particular frame of the test video has to the corresponding frame in the query video. We also maximize $S(k, j)$ over k in a T-length temporal window centered at i. The rationale is that if a T-length sequence centered at frame k of the test video matches well with the query video, then all frames in that T-length sequence should also have at least the same score.

The above steps can be easily extended to the case where the test video and query video do not have the same spatial dimensions. In that case, as proposed by Shechtman and Irani [117], we simply slide the query video template over all possible spatial-temporal locations (illustrated in Figure 7.3), and compute a score for each space-time location using Equation (7.4). This results in a action confidence volume, $C(n, m, i)$, which represents the score for the (n, m) location of the ith frame of the test video. A high value of $C(n, m, i)$ can then be interpreted as the query action being likely to be occurring at spatial location (n, m) in the ith frame.

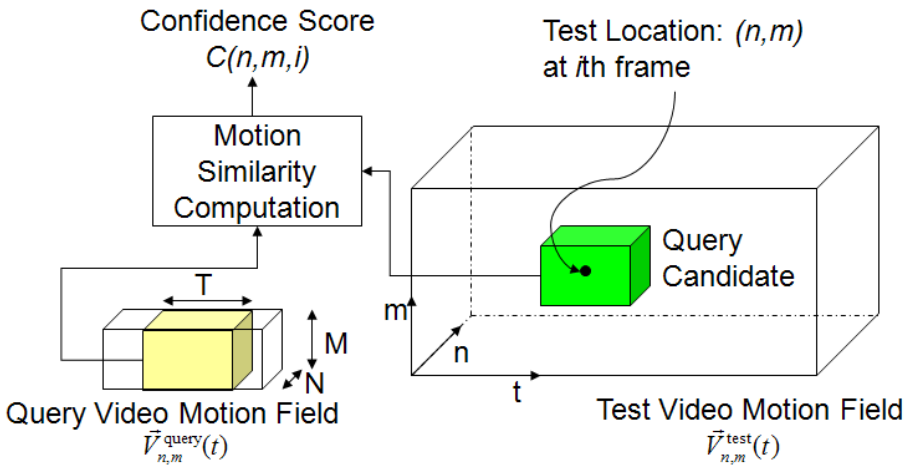


Figure 7.3. Illustration of space-time localization. The query video space-time patch is shifted over the entire space-time volume of the input video, and the similarity, $C(n, m, i)$ is computed for each space-time location.

While this exhaustive search seems to be computationally intensive, operating in the compressed domain allows for a real-time implementation.

7.3.6 Video action similarity score

Given $C(n, m, i)$, we can compute a non-symmetric similarity, $\rho(X^{\text{test}}, X^{\text{query}})$, of the test video to the query video by using:

$$\rho(X^{\text{test}}, X^{\text{query}}) = \frac{1}{L} \sum_{i=1}^{T_{\text{test}}} \eta(i) \left(\max_{n,m} C(n, m, i) \right) \tag{7.5}$$

where the normalization factor L is given by:

$$L = \sum_{i=1}^{T_{\text{test}}} \eta(i)$$

and $\eta(i)$ is an indicator function which returns one if at least T frames in the $(2T + 1)$-length temporal neighborhood centered at frame i have significant motion and returns zero otherwise:

$$\eta(i) = \mathbb{I} \left[\sum_{j=i-T}^{i+T} \mathbb{I} \left[Q(j) \geq \delta \right] \geq T \right]$$

and the fraction of significant motion vectors in frame j, $Q(j)$, is given by:

$$Q(j) = \frac{\sum_{n=0}^{N^{\text{test}}-1} \sum_{m=0}^{M^{\text{test}}-1} \mathbb{I} \left[\| \vec{V}_{n,m}^{\text{test}}(j) \| > \epsilon \right]}{N^{\text{test}} \cdot M^{\text{test}}}$$

Figure 7.4. Snap-shot of frames from action videos in database [114]. From left to right: boxing, handclapping, handwaving, running, jogging, walking. From top to bottom: outdoors environment, outdoors with different clothing environment, indoors environment. The subjects performing each action is the same across the different environments.

A frame is asserted to have significant motion if at least δ proportion of the macroblocks have reliable motion vectors (reliable in the sense defined in Section 7.3.2) of magnitude greater than ϵ, *i.e.* $Q(j) \geq \delta$.

7.4 Experimental results

We evaluate our proposed algorithm on a comprehensive database compiled by Schüldt *et al.* [114][1]. As illustrated in Figure 7.4, their database captures 6 different actions (boxing, handclapping, handwaving, running, jogging and walking), performed by 25 people, over 4 different environments (outdoors, outdoors with scale variations, outdoors with different clothes and indoors). Since our system was not designed to handle scale-varying actions, we considered only the three environments that do not have significant scale variations.

To evaluate performance, we perform a leave-one-out full-fold cross-validation within each environment, *i.e.* to classify each video in the dataset, we use the remaining videos that are not of the same human subject as the training set. This will improve the statistical significance of our results given the limited number of videos in the dataset. To perform

[1]Available for download at `http://www.nada.kth.se/cvap/actions/`

classification, we simply use Nearest Neighbor Classification (NNC) by evaluating the video action similarity score (see Section 7.3.6) with each of the videos in the training set.

In our experiments, we used $\delta = \frac{1}{30}$, $\epsilon = 0.5$ pels/frame, $\alpha = 2.0$ and $T = 17$. For comparison, we also tested both NCNC (Equation (7.1)) and NZMS (Equation (7.2)) when computing frame-to-frame motion similarity.

7.4.1 Classification performance

The action classification confusion matrix for our algorithm when using NZMS is shown in Table 7.1, while that using NCNC [40] is shown in Table 7.2. Each entry of the matrix gives the fraction of videos of the action corresponding to its row that were classified as an action corresponding to the column. Using the proposed NZMS, our overall percentage of correct classification is 90%. As a comparison against state-of-the-art methods that work in the pixel domain, we note here for reference that Schüldt *et al.* [114], Dollar *et al.* [35] and Ke *et al.* [71] report classification accuracies of 72%, 81% and 63% respectively on the same dataset. While the methodology and classification methods used in these works differ, our results compare very favorably, even though we use compressed domain features and a very simple classifier.

Table 7.1. Confusion matrix using NZMS

	Box	Hc	Hw	Run	Jog	Walk
Boxing	0.86	0.07	0.05	0.00	0.00	0.01
Hand**c**lapping	0.03	0.89	0.08	0.00	0.00	0.00
Hand**w**aving	0.00	0.04	0.96	0.00	0.00	0.00
Running	0.00	0.00	0.00	0.79	0.21	0.00
Jogging	0.00	0.00	0.00	0.01	0.97	0.01
Walking	0.00	0.00	0.00	0.00	0.07	0.93

Looking at the confusion matrices, we see that our proposed NZMS measure vastly outperforms NCNC. This is due to the fact that our measure looks at each corresponding pair of macroblocks separately instead of looking across all of them. NZMS also considers

Table 7.2. Confusion matrix using normalized correlation [40]

	Box	Hc	Hw	Run	Jog	Walk
Boxing	0.86	0.00	0.01	0.00	0.00	0.12
Hand**c**lapping	0.43	0.32	0.24	0.00	0.00	0.00
Hand**w**aving	0.01	0.01	0.97	0.00	0.00	0.00
Running	0.00	0.00	0.00	0.97	0.03	0.00
Jogging	0.00	0.00	0.00	0.21	0.79	0.00
Walking	0.00	0.00	0.00	0.00	0.61	0.39

both differences in motion vector orientations and norms, and ignores matching zero-motion macroblocks.

Using NZMS, most of the confusion is between "Running" and "Jogging", with a significant proportion of "Jogging" videos being erroneously classified as "Running". Looking at the actual videos visually, we found it hard to distinguish between some "Running" and "Jogging" actions. In fact, there are certain cases where the speed of one subject in a "Jogging" video is faster than the speed of another subject in a "Running" video.

7.4.2 Performance gain from thresholding optical flow confidence map

Table 7.3 shows the effects of thresholding on action classification performance using our proposed approach. By removing noisy estimates of the optical flow, we are able to achieve a 10% gain in classification performance when using NZMS as the motion similarity measure.

Table 7.3. Classification performance with and without thresholding confidence map

Method	With thresholding	Without thresholding
NZMS	90.0%	81.2%
NCNC	71.7%	72.5%

7.4.3 Effect of α variation on classification performance

To understand the effect of α on classification, we ran an experiment using NZMS with varying values of α. Table 7.4 shows the results of this experiment. We see that the classification performance is relatively stable over a range of α. More importantly, it is also clear that the aggregation step described in Section 7.3.4 is critical for action classification.

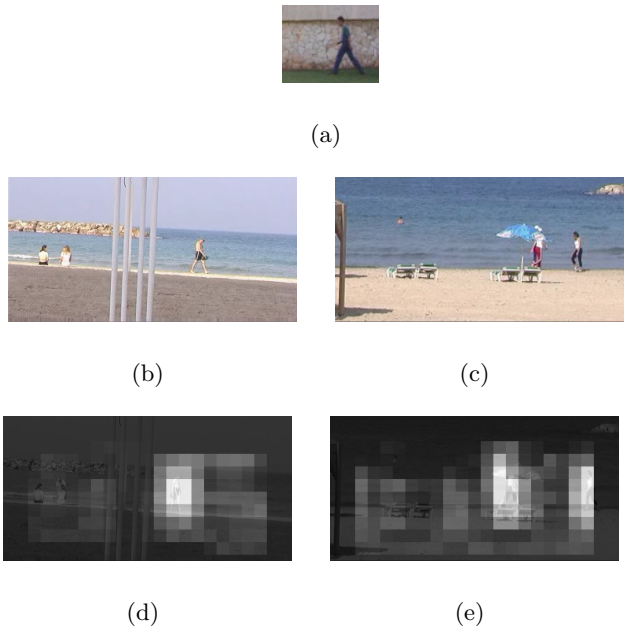
Table 7.4. Classification performance with varying α

α	Classification performance
1.0	88.2%
2.0	90.0%
3.0	91.0%
4.0	90.8%
No aggregation	62.5%

7.4.4 Localization performance

Unlike most other methods, with the notable exception of [117, 71], we are able to localize an action in space and time as well as detect multiple and simultaneous occurring activities in the test video. Figure 7.5 shows an example (the "beach" test sequence and walking query sequence from Shechtman and Irani [117]) which demonstrates our algorithm's ability to detect multiple people walking in the test video. We emphasize that we only use a single template video of a person walking to localize walking actions in the test video. Since our algorithm is not appearance based, there is no problem with using a query video of one person on a test video containing other people.

In the test sequence, there are both static background clutter, such as people sitting and standing on the beach, and dynamic background clutter, such as sea waves and a fluttering umbrella. This background is very different from that in the query sequence. Since the spatio-temporal motion field of background motion such as sea waves is different from that of walking, it is not picked up by our algorithm. No special handling of the background motion is necessary.



(a)

(b) (c)

(d) (e)

Figure 7.5. Action localization results. The highlighting in (d) and (e) denotes detection responses, with bright areas indicating high responses. (a) A frame from the query video, (b) An input video frame with one person walking, (c) An input video frame with two people walking, (d) Detection of one person walking, (e) Detection of two people walking.

7.4.5 Computational costs

On a Pentium-4 2.6 GHz machine with 1 GB of RAM, it took just under 11 seconds to process a test video of 368×184 pixels with 835 frames on a query video that is of 80×64 pixels with 23 frames. We extrapolated the timing reported in [117] to this case; it would have taken about 11 hours. If their multi-grid search was adopted, it would still have taken about 22 minutes. Our method is able to perform the localization, albeit with a coarser spatial resolution, up to *3 orders of magnitude faster*. On the database compiled in [114], each video has a spatial resolution of 160×120 pixels, and has an average of about 480 frames. For each environment, we would need to perform 22500 cross-comparisons. Yet, each run took an average of about 8 hours. In contrast, [117] would have taken an extrapolated run time of 3 years.

7.5 Effects of video encoding options

In the experiments described in the previous section, we have used input video compressed with MPEG [53], with a group-of-pictures (GOP) size of 15 frames, and a GOP structure of I-B-B-P-B-B-, where 'I' refers to an Intra-frame, 'P' refers to a Predicted-frame, and 'B' refers to a Bi-directionally predicted-frame. It would be interesting to see if there is any discernible difference when different encoding options, such as GOP size, GOP structure and the use of half-pel or quarter-pel motion estimation, are used. In addition, while MPEG uses 16×16 pixels macroblock as the basis of motion compensation, newer encoding standards such as H.263+ and H.264 allow the use of smaller block sizes [28, 136].

These experiments would be useful for a systems engineer in choosing a video encoder and its encoding options. While storage space and video quality are important considerations, it would be helpful to know if sacrificing a little compression performance would yield large performance gains in surveillance tasks such as action detection.

In the experiments below, we have used the publicly available "FFMPEG" video encoder[2]. When applicable, we will describe the encoder options and specify the actual flags used with FFMPEG in parentheses. Unless otherwise mentioned, the encoding options used are that the MPEG-4 video codec is used ("-vcodec mpeg4"), the output video is of similar quality to the input video ("-sameq"), and the "AVI" container format is used.

7.5.1 GOP size and structure

We first look at how varying GOP size and structure affects classification performance. We consider two commonly used GOP structure, I-B-B-P-B-B- ("-bf 2") and I-P-P-P-P-P-. We also look at a variety of GOP sizes $\{9, 12, 15, 18, 30, 60, 120, 240\}$ ("-g [GOP size]"). By looking at how classification performance varies with compression performance, we can get an idea of what trade-offs are possible by varying GOP parameters when performing video encoding. In these experiments, the output video quality is kept relatively similar over all GOP size and structure.

[2]Available at `http://ffmpeg.mplayerhq.hu/`

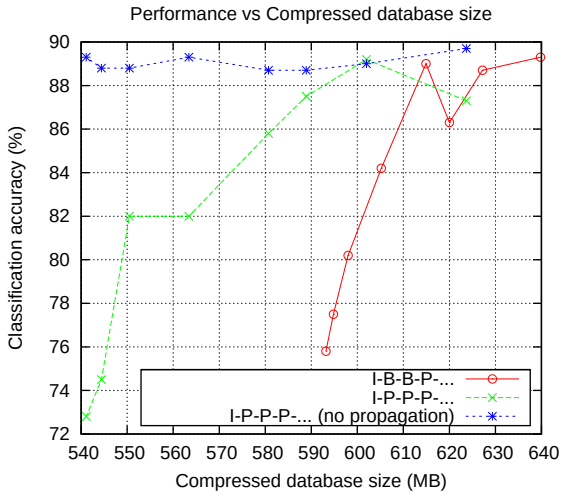


Figure 7.6. Effect of varying GOP size on classification performance and compression performance. In general, increasing GOP size results in decreasing classification performance. Also, having no B frames in the GOP structure offers a better compression-classification trade-off. The fairly constant performance of the scheme using I-P-P-P-... with no texture propagation error indicates that the main source of performance degradation with increasing GOP size is due to propagation errors in computing block texture.

It should be expected, and is in fact the case, that the larger the GOP size, the smaller the compressed videos, since predicted frames such as P and B frames can be more efficiently compressed than I frames. The results in Figure 7.6 further shows that in general, increasing GOP size also results in decreasing classification performance. This could be due to the fact that the update of the confidence measure computed as in Section 7.3.2 suffers from error propagation with each P frame. To test out this hypothesis, we also ran experiments where the confidence measure is computed from the *DCT of the actual decoded frame pixels* instead. Looking at the curve for the I-P-P-P-... GOP structure with no texture propagation error, we see that the classification accuracy is indeed fairly constant over a wide range of GOP sizes. This confirms that the main source of performance degradation with increasing GOP size is due to the propagation errors in computing the confidence measure.

Figure 7.6 also shows that for the most part, the I-P-P-P-... GOP structure offers a better classification-compression trade-off than the I-B-B-P... GOP structure. There are two possible reasons for this. First, because of the complexity of articulated motion, B-frames are unable to provide any substantial compression gains over P-frames, while

suffering from overhead. Hence, the I-B-B-P-... structure, for the same GOP size, actually performs worse in terms of compression performance. Second, the I-B-B-P-... structure introduces inaccuracy into the optical flow estimation process. The P frames are spaced 3 frames apart, and hence its estimated motion is actually over 3 temporal frames and not over 1 frame.

The experiments in this section seem to suggest that if action classification is an important factor in determining encoding options, then no B frames should be used in the encoding. This also has other advantages such as simpler encoders and decoders requiring less frame buffer memory. Further, if we used the confidence measure as computed by Equation 6.1 in Section 6.1.2, the GOP size should not be too large. A GOP size of 12, 15 or 18 seems to offer a good balance between compression and action classification. There might also be other factors in determining GOP size however, such as ease of random access and error resilience.

7.5.2 Quarter-pel accuracy motion estimation

In MPEG, motion estimation was carried out to half-pel accuracy. It was found that better motion compensation is possible with a further increase in accuracy to quarter-pel [136, 134]. This motivates us to investigate if an increase in motion estimation accuracy ("-qpel 1") would also translate into better action classification performance.

Figure 7.7 shows that using quarter-pel accuracy in motion estimation does not actually improve the classification-compression trade-off. There are two main reasons for this. First, we observe that on this set of action videos, for the same GOP size, using quarter-pel accuracy actually performs worse than half-pel accuracy in terms of compression performance. This could be due to the storage overhead of motion vectors with increased accuracy. Second, quarter-pel accuracy does not translate into better action classification performance.

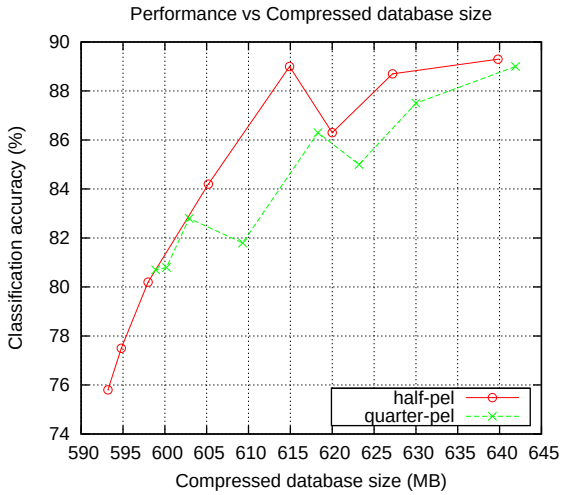


Figure 7.7. Effect of quarter-pel accuracy motion estimation on classification performance and compression performance. There seems to be no significant improvement in the compression-classification trade-off by using motion estimation with quarter-pel accuracy instead of half-pel accuracy.

7.5.3 Block size in motion compensation

As mentioned earlier, newer encoding standards have the option of allowing smaller block sizes to be used in motion compensation [28, 136]. We compare the effect of forcing smaller blocks in motion compensation ("-mv4 1") on both action classification performance and compression performance. In this set of experiments, we used a GOP structure of I-B-B-P-...

Figure 7.8 shows that using smaller blocks in motion compensation does result in a better performance vs compression trade-off. Smaller blocks allows for a more refined motion compensation and prediction, hence resulting in better compression performance. At the same time, with higher resolution motion vectors, action classification performance also improves. Of course, while using smaller blocks for motion compensation improves the trade-off, it has to be weighted by the increase in computation time. In our experiments, increasing the motion estimation resolution by 2 in each dimension resulted in about 5 times increase in run-time.

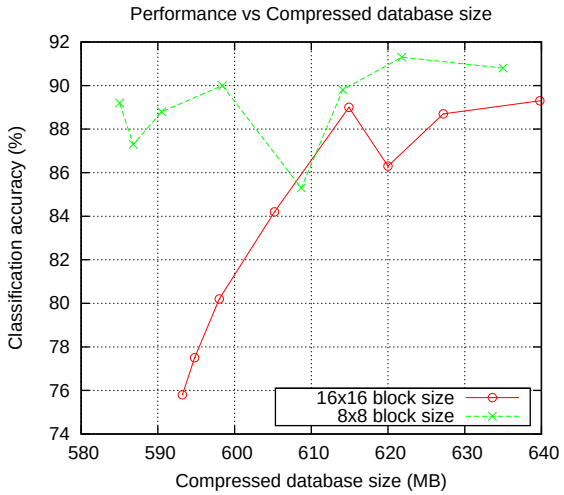


Figure 7.8. Effect of using different block sizes in motion compensation on classification performance and compression performance. Using a smaller block size results in a better compression-classification trade-off, but this has to be weighed against the resulting increase in computational time.

7.6 Organization of activity videos

So far, we have only considered measuring similarities between activity videos using Equation (7.5). However, this notion of action similarity induces a perceptual hierarchy on a collection of videos (see Figure 7.9 for example). A system that can efficiently generate such a hierarchy of the videos based on action similarity would be very useful in facilitating efficient navigation of the database thus improving its utility. Building such a system is very challenging if we consider videos containing actions of articulated structures like humans and animals moving in the visual scenes. It is preferable to assume *no metadata* (*e.g.* labels), *no segmentation* and *no prior alignment* for the video collections.

Specifically, given a set of videos and a user-defined *space-time scale* of actions, we would like the system to: (a) automatically and efficiently organize the videos into a hierarchy based on action similarity; (b) estimate clusters; and (c) select one representative exemplar for each cluster.

There has been some prior works on organizing large databases of videos using techniques that operate directly on compressed domain features to offer a significant speed-up in processing time. Chang *et al.* assume that objects can be segmented and tracked easily

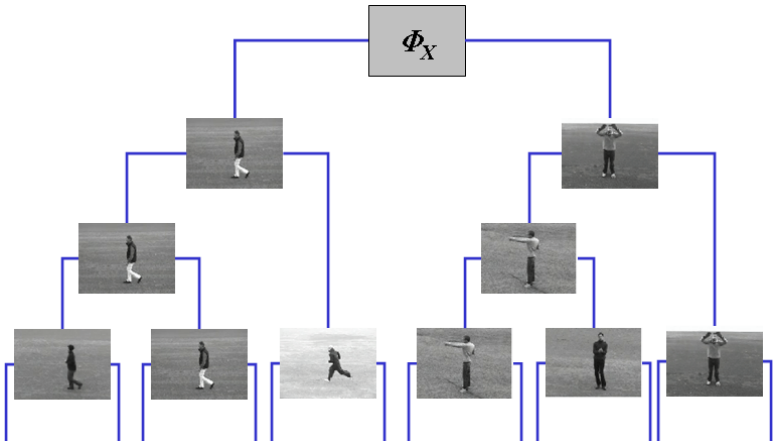


Figure 7.9. A qualitative example of an action hierarchy for the activity video collection Φ_X, with associated exemplars for the subtree under each node, shown up to 6 clusters. This was generated using our proposed approach with NCNC as the action similarity measure and Ward linkage as the neighbor-joining criterion. The 6 clusters from left to right: Jogging, Walking, Running, Boxing, Handclapping, Handwaving. See Section 7.6.2 for further discussion.

in order to compute features [21]. Some approaches segment a single video into shots and organize neighboring shots into a hierarchy for browsing the video but they do not build action based hierarchies across a large collection of videos [151, 97]. Dimitrova *et al.* make use of motion vectors to estimate object trajectories and then use the estimated object trajectories to reason about actions [33].

7.6.1 Method

Let $\Phi_X \doteq \{X^p\}_{p=1}^P$ be the given set of videos, where $P \in \mathbb{Z}_+$ is the cardinality of the set, and let $\tilde{N} \times \tilde{M} \times \tilde{T}$ be the user-specified space-time scale of interest. Each video X^p has an action label $y^p \in \{1, .., K\}$, where K is the number of actions in the collection. Reusing the notation described in Section 7.3.1, X^p is a video with T^p frames, with each frame containing $N^p \times M^p$ macroblocks. $\vec{V}^p$ is the spatio-temporal pattern (motion field) associated with video X^p. We again assume that an *action* induces a motion field that can be observed as a spatio-temporal pattern.

Figure 7.10 shows the flow of our algorithm for organizing the videos (Φ_X) with minimal user input. Using the similarity scores computed using Equation (7.5), we compute the pair-

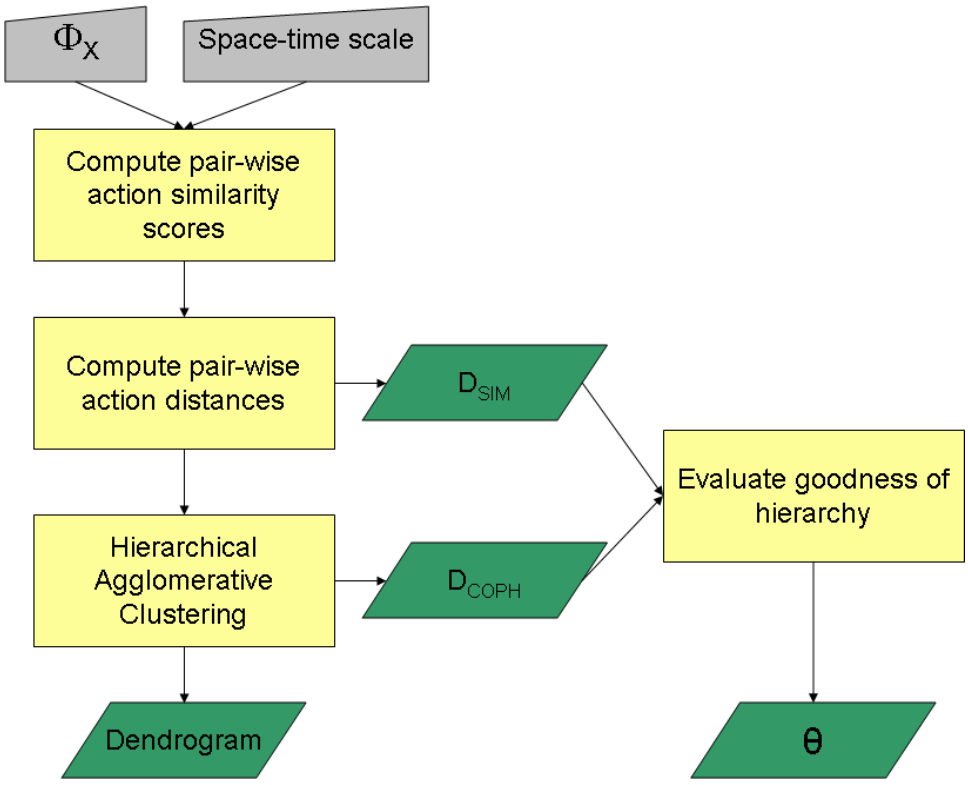


Figure 7.10. Data flow for our proposed approach. Given a set of videos Φ_X and a user-defined space-time scale for actions, we compute pair-wise action similarity scores between all pairs of videos, and then convert them to symmetric action distances, D_{SIM}. We use D_{SIM} in hierarchical agglomerative clustering to produce a dendrogram, which is a binary hierarchical tree representing the videos, and the pair-wise cophenetic distances D_{COPH}, which are distances computed from the constructed dendrogram. The cophenetic correlation coefficient, Θ, is the correlation coefficient between D_{SIM} and D_{COPH}, and can be used to evaluate the goodness of the hierarchy.

wise symmetric action distances for videos X^p and X^q as follows:

$$D_{\text{SIM}}(X^p, X^q) = \frac{1}{\max\left(\frac{1}{2}\left(\rho(X^p, X^q) + \rho(X^q, X^p)\right), \beta\right)} \tag{7.6}$$

where β represents the smallest value of $\rho(.,.)$ admissible. In our experiments, we choose $\beta = 0.01$.

We then apply hierarchical agglomerative clustering (HAC) [133] to construct a binary tree (also called *dendrogram*) containing all the elements of Φ_X as leaf nodes. Divisive methods (*e.g.* K-means, K-medoids) for constructing dendrogram are usually sensitive to initialization [133]. To address this sensitivity with divisive methods, typically one needs to perform many randomly initialized trials in order to obtain a good clustering solution, thus

resulting in loss of computational efficiency. In contrast, HAC constructs the dendrogram in a sequential and *deterministic* fashion using a neighbor-joining (also called linkage) criterion. We use four different linkage criteria in our experiments:

- **Single linkage.** This method uses minimum distance between the clusters as the merging criterion, where distance between clusters is defined as the distance between closest pair of elements(one element drawn from each cluster) [60]. Pairs consisting of one element from each cluster are used in the calculation. The first cluster is formed by merging the two groups with the shortest distance. Then the next smallest distance is found between all of the clusters. The two clusters corresponding to the smallest distance are then merged.

- **Complete linkage.** The merging process for this method is similar to single linkage, but the merging criterion is different: the distance between clusters is defined as the distance between most distant pair of elements(one element drawn from each cluster) [60].

- **Average linkage.** The merging process for this method is similar to single or complete linkage, but the merging criterion is the average distance between all pairs, where one element of the pair comes from each cluster [60].

- **Ward's linkage.** The distance between two clusters in this method is defined as the incremental sum of the squares between two clusters [60].

The user defines a stopping condition for the agglomeration, L^{STOP}, which is the farthest allowable merging distance between clusters. L^{STOP} is used to cut the dendrogram at an appropriate level and obtain the clusters. After computing the matrix of pair-wise action distances $D_{\text{SIM}} \in \mathbb{R}^{P \times P}$ as described in Equation (7.6), we apply HAC to obtain the hierarchy. The *cophenetic distance* between videos X^p and X^q, $D_{\text{COPH}}(X^p, X^q)$, is their linkage distance when first merged into the same cluster in the HAC procedure [133].

7.6.2 Results

We use the same dataset shown in Figure 7.4 [114] to perform our evaluations. From each action video, we create a query video by cropping out a space-time volume in an automatic fashion. Since automatic determination of space-time scale is very hard, we let the user specify the size of an approximate space-time bounding box, $\tilde{N} \times \tilde{M}$ macroblocks by $\tilde{T}$ frames, for the entire collection of videos. This implicitly constrains the system to consider actions of approximately similar space-time scale. The system then looks in each action video for a $\tilde{M} \times \tilde{N} \times \tilde{T}$ space-time volume that contains the most number of significant motion vectors, where $\vec{V}$ is significant if $\|\vec{V}\| > \epsilon$ (as defined in Section 7.3.6).

In each cluster, an exemplar is defined as the element that has the minimum pair-wise distance with respect to all the other elements in the cluster. A meaningful hierarchy would organize the videos in a way such that each cluster contains elements that are homogeneous and the exemplar from each cluster would represent a distinct action from the dataset. In Figure 7.9, we show the estimated action hierarchy constructed using NCNC action similarity measure with Ward linkage neighbor-joining criterion. Notice that the actions such as running, walking and jogging were grouped separately compared to actions such as boxing, handwaving or handclapping. Intuitively, this fits well with what a human operator would do given the same task. Among the 4 linkage criteria we used, we found qualitatively that the combination of NCNC and Ward linkage gives the best inference for exemplars of actions in the database.

7.7 Recapitulation

We have designed, implemented and tested a system for performing action recognition and localization by making use of compressed domain features such as motion vectors and DCT coefficients which can be obtained with minimal decoding. The low computational complexity of feature extraction and the inherent reduction in search space makes real-time operation feasible. We combined existing tools in a novel way in the compressed domain for this purpose and also proposed NZMS, a novel frame-to-frame motion similarity measure.

Our classification results compare favorably with existing techniques [114, 35, 71] on a publicly available database and the computational efficiency of our approach is significantly better than existing action localization methods [117].

Our experimental results provide justification for the engineering choices made in our approach. In particular, we showed the value of filtering motion vectors with low texture and of aggregating frame-to-frame similarities. We also systematically investigated the effects of various encoding options on the action classification performance of our proposed approach. The results showed that for action videos, using a GOP structure with only P frames results in a better compression-classification trade-off. We also found that while a larger GOP size might result in a lower classification performance, it is mostly due to the effects of drift in computing block texturedness. Thus, a simple extension for improving classification performance in videos with large GOP size, if memory constraints permit, is to perform full decoding of every frame, and to use the decoded pixels at shorter regular intervals to update the confidence map. We found that quarter-pel accuracy in motion estimation does not appear to provide any benefits. While using smaller blocks in motion compensation does lead to better action classification and compression performance, the increased computational time of both encoding and action classification should be taken into account.

In this work, we have used a very simple classifier, *i.e.* Nearest Neighbor Classification (NNC), which has given very good performance. For further improvement in classification, we can use more sophisticated classifiers such as Support Vector Machines (SVM); on the same dataset, Dollar *et al.* have shown that using SVMs results in a slight improvement over NNC [35].

For future work, we plan to extend our system to adopt a hierarchical approach which would allow us to approach the spatial resolution of existing pixel-domain methods at lower computational cost. By leveraging the ability of state-of-the-art encoders such as H.264 to use smaller blocks in motion compensation, motion vectors at resolutions of up to 4×4 pixels block can be obtained. The algorithm can first perform action recognition at the coarsest level, *i.e.* 16×16 pixels macroblock, and then perform a progressively finer level

search in promising regions. Furthermore, using the motion vectors of 4×4 pixels block as an initial estimate also allows the computation of dense optical flow at lower cost, hence enabling the progressive search to proceed to pixel level granularity.

One current limitation of our approach is that while it is robust to small variations in spatial scale, it is not designed to handle large spatial scale variations or differences in spatial scales between the query and test videos. We would like to explore a truly scale-invariant approach in future work. A possibility is to apply our method at different resolutions in parallel; this can be done naturally with the hierarchical extension described earlier. Parallelizing this scale-space search could lead to significant gains in performance while being scale-invariant.

While we present results on a benchmark dataset widely used for evaluating activity recognition algorithms [114, 35, 71], it would be interesting to consider data with other actions and containing more varied backgrounds as part of future work. For example, the BEHAVE project, which has the objective of automatically detecting anomalous or criminal behavior from surveillance videos, has publicly available datasets[3]. One interesting approach uses optical flow information to identify such behavior [6]; it would be useful to see how our method, which uses only motion vectors, compares with the former, which uses optical flow. While we consider single person actions, detecting multi-party activities such as greeting or fighting is also a potential area of further investigation [111, 6].

Another interesting angle to consider is the type of motion estimation used at the encoder. Rate-Distortion (RD) optimization is commonly performed in sophisticated video encoders to seek an optimum trade-off between compression and reconstruction quality [125]. It has also been used in the motion compensation process to reduce the rate used for coding motion vectors [124, 23]. This has the effect of smoothing the motion vector field which can be interpreted as a de-noising process. We hypothesize that this has a positive influence on the compression-classification trade-off, but this would have to be verified.

We have also demonstrated an efficient unsupervised approach for organizing large collections of videos into a meaningful hierarchy based on the similarity of *actions* embedded

[3] `http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/`

in the videos. This facilitates quick navigation of the database. Using the derived hierarchy, we showed how to select representative videos (exemplars) from a dataset. The database can be quickly indexed by assigning a unique action tag to each cluster. For example, a user can easily label a cluster simply by identifying the cluster exemplar. These derived action tags can then be combined with other features, such as color and texture, to build more complex queries or to develop organizational principles for managing video databases.

Chapter 8

Video analysis of meetings

In this chapter, we present work that aims 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 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 Section 8.2, we briefly describe the concept of dominance before presenting our method and results.

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, participants make use of a projection screen for discussion purposes. It has been observed that participants tend to look at the projection screen when there is a slide transition. Thus, the presence of a slide transition can be used as a contextual cue for improved VFOA performance. In Section 8.3, we propose a compressed-domain processing approach to detect slide transitions.

The work presented in this chapter is joint work with Dinesh Jayagopi, Hayley Hung, Kannan Ramchandran and Daniel Gatica-Perez, and has been presented in part in [147, 63, 69]. We also like to acknowledge the advice and assistance given by Silèye Ba and Jean-Marc Odobez.

8.1 AMI meeting data

We use meetings from the publicly available AMI meeting corpus [17]. The meetings have been recorded in IDIAP's smart meeting room (see floor plan in Figure 8.1), which also has a table, a slide screen and a white board. In this dataset, there is a camera taking a close-up shot of each participant, for a total of four close-up camera views as shown in the bottom row of Figure 8.2. There are also three other camera views capturing side-views and a global view, as shown in the top row of Figure 8.2. 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.

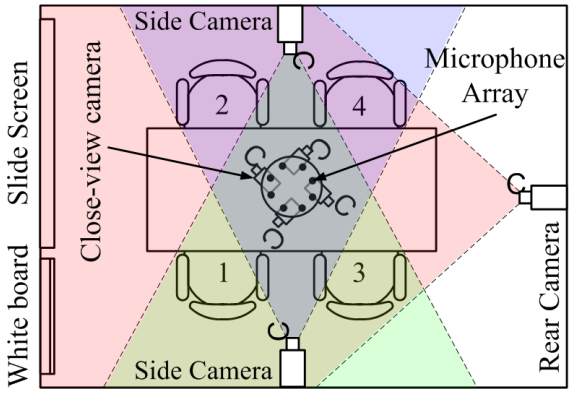


Figure 8.1. Floor plan of smart meeting room

In each meeting, 4 participants went about the task of designing a remote control. Each participant was assigned distinct roles, namely "Project Manager", "User Interface Specialist", "Marketing Expert" and "Industrial Designer". To encourage natural behavior, the meetings were not scripted. However, teams were required to carry out general tasks such as presentations and discussions.

8.2 Activity level estimation for dominance classification

A concept that is well studied in social psychology, dominance is one of the basic mechanisms of social interaction and has fundamental implications for communications both

Figure 8.2. All available views in the data set. The top row shows the right, center and left camera views. The bottom row shows each of the 4 close-up views.

among individuals and within organizations [16]. A good way to understand this concept is by distinguishing it from power. While power is the "capacity to produce intended effects, and in particular, the ability to influence the behavior of another person", dominance is the set of "expressive, relationally based communicative acts by which power is exerted and influence achieved" and hence "necessarily manifest" [39].

When there is recorded video data available, for example, in an instrumented meeting room, automatic dominance estimators using recorded data could be useful in applications such as self-assessment, training or group collaboration. Studies of dominance in the social psychology literature has suggested that such an enterprise is worth pursuing. First, vocalic cues, such as speaking length, speaking energy and vocal control, and kinesic cues, such as body movement, posture and gestures, have been found to be correlated with dominance [39]. Of particular interest is the finding that dominant people are normally more active than non-dominant people [16]. Second, both active participants and passive observers are known to be able to decode dominance [36]. This suggests that reliable data annotation, which is necessary, and it for evaluating automatic dominance estimators, is possible and that there is a possibility of designing such estimators.

In this section, we study a set of visual features that can be efficiently extracted from

compressed video and are justified by the observation that dominant people are normally more active than non-dominant people [16]. We focus on an unsupervised approach for dominance modeling and evaluate it on video from the AMI meeting dataset.

8.2.1 Approach

To estimate individual activity level, we turn to the use of motion vector magnitude (see Figure 6.1(b)) and residual coding bit-rate (see Figure 6.1(c)) as described in Section 6.1. 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 (shown in the bottom row of Figure 8.2). 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. In particular, we have noticed visually that residual coding bit-rate also correlates well with high activity levels.

To detect when a participant is not in the close-up view, we threshold the number of skin-colored blocks (see Figure 6.1(d)) in the close-up view, obtained as described in Section 6.1.4. 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 use 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. This can be done in an unsupervised manner.

8.2.2 Experiments

Annotation

A total of 59 five-minute meeting segments from 11 sessions of the AMI meeting corpus were each annotated by 3 annotators for perceived dominance rankings of the participants [69]. The segments were chosen to be 5 minutes long to provide more data points for testing. At the same time, there is evidence to suggest that people need a relatively small amount of time to make accurate judgments about the behavior of others [5].

For each meeting segment, annotators were asked to rank the participants from 1 (highest) to 4 (lowest) according to their level of perceived dominance. Annotators were also asked to state their confidence in their rankings on a seven-point scale. Note that annotators were given neither a prior definition of dominance nor were told what cues to look out for.

Evaluation criteria

We target the task of automatically classifying the most dominant person in each meeting segment. To better understand the strengths and weaknesses of our method, we look at three sets of meetings: (a) 34 meeting segments where *every* annotator agreed on the most dominant person; (b) 23 meeting segments where *only 2* annotators agreed on the most

dominant person; and (c) 57 meeting segments 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 segments where *every* annotator agreed on the least dominant person.

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

Baseline comparison

For baseline comparison, we implement a similar scheme that works in the pixel domain. For each frame, we compute its optical flow[1] 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 as in Section 6.1.4.

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. Averaging is performed over the skin-color pixels instead of skin-color blocks as is done 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.

[1]We used the OpenCV library implementation of the Lucas-Kanade optical flow algorithm.

Results

Tables 8.1 through 8.4 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 outperforms them under some 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.

Table 8.1. 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 8.2. 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

Table 8.3. 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

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

Table 8.4. 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

segments where the features fail to find the most dominant person, either the most active participant, in terms of body movement, 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 8.1. 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 is not as good as that for finding the most dominant participant. It is interesting to note that the average reported annotator confidence for this task is slightly lower than for the most dominant task.

We next consider 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[2], an open source video decoder for MPEG-4, which we have modified for our purposes. No particular care has been taken to optimize it. The pixel domain baseline scheme is implemented using OpenCV[3], a popular open source computer vision library. Both schemes were evaluated on a Xeon 2.4 GHz Intel processor with 4 GB

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

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

of RAM. Table 8.5 shows that a computational time reduction of 94.2% can be achieved by using the compressed-domain approach instead of the pixel-domain scheme, yet with no degradation in dominance classification performance.

Table 8.5. 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

8.3 Slide transition detection - a contextual cue for VFOA

The goal in estimating visual focus of attention (VFOA) is to determine the visual target that each participant is looking at [9]. Acting as a proxy for eye gaze, VFOA is of great importance in meetings analysis tasks such as identifying addressees in dialogue acts, inferring turn taking and modeling conversation structures.

In the AMI meeting corpus, possible visual targets are unfocused, the other meeting participants and objects in the meeting room (see Figure 8.1) such as the table and the slide screen [9]. It is possible that the same head pose can be used to focus at different visual targets; for example, in Figure 8.1, there could be some ambiguity in determining if the participant in seat 4 is looking at the participant in seat 2 or at the slide screen. Hence, contextual cues, in addition to estimated head pose, could be used to resolve such ambiguities [9].

One such contextual cue is the presence of slide transition. When a new slide is displayed, it is likely that meeting participants would look at the slide screen instead of other meeting participants [9]. In this section, we focus on the fast detection of slide transitions in compressed videos which can be used as a contextual cue for VFOA.

8.3.1 Approach

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 [139, 42]. Considering 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 [42].

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. Specifically, 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 even when 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, which is 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 8.3.

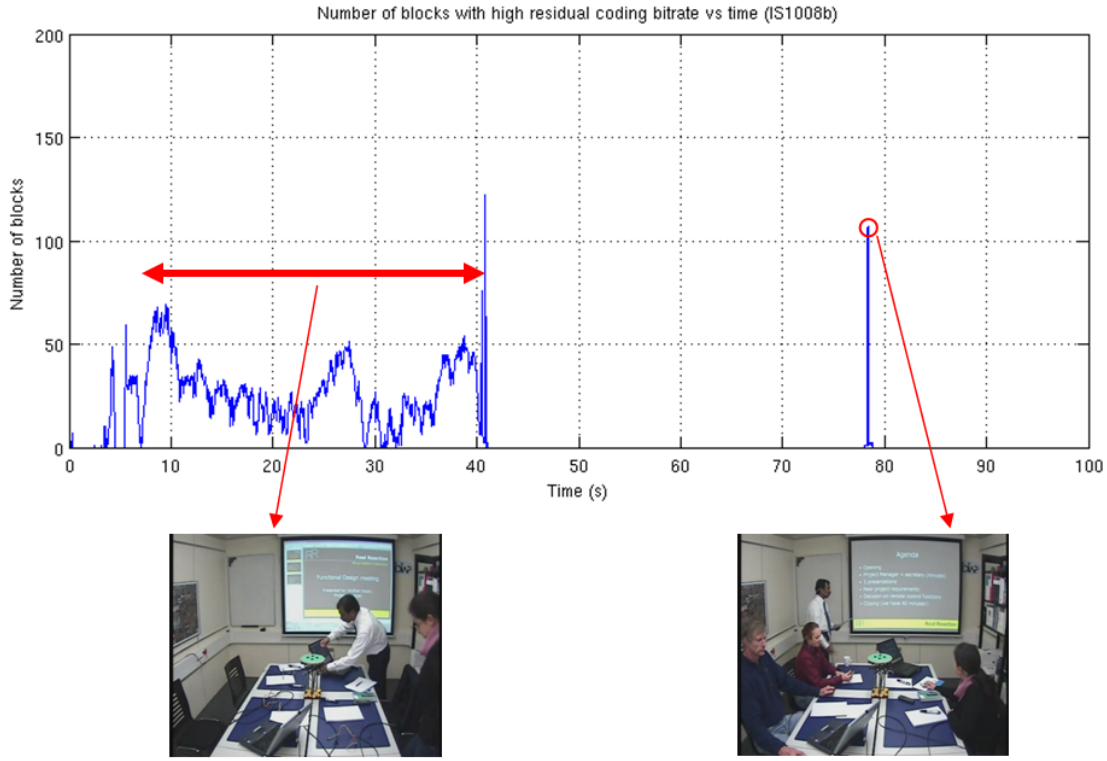


Figure 8.3. Plot of $N_r(t)$, the number of blocks with high residual coding bit-rate, in meeting session IS1008b. In the period around 10s-40s when a person is moving in front of the projection screen, note that while the number of blocks is moderately high, there is no sharp peak. On the other hand, a slide change at around 78s produces a very sharp peak.

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:

$$N_r(s) \geq \alpha \tag{8.1}$$

$$N_r(s) \geq N_r(s+v) \quad \forall \quad v \in [-T/2, T/2] \tag{8.2}$$

$$N_r(s) \geq \beta + \tfrac{1}{T/2} \textstyle\sum_{v=1}^{T/2} N_r(s-v) \tag{8.3}$$

$$N_r(s) \geq \beta + \tfrac{1}{T/2} \textstyle\sum_{v=1}^{T/2} N_r(s+v) \tag{8.4}$$

α and β are thresholds that determine respectively 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. 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.

8.3.2 Experiments

Evaluation criteria

We carried out our evaluations on 12 meetings from the AMI meeting corpus [17], 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 8.2), 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.

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 residual is much finer in the pixel-domain scheme than the compressed domain scheme.

Results

The performance of these two schemes is shown as a ROC plot in Figure 8.4. 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 [76], 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 and list those scores in Table 8.6. 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 8.6. We also compute the speed-up factor, SUF, defined as:

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

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

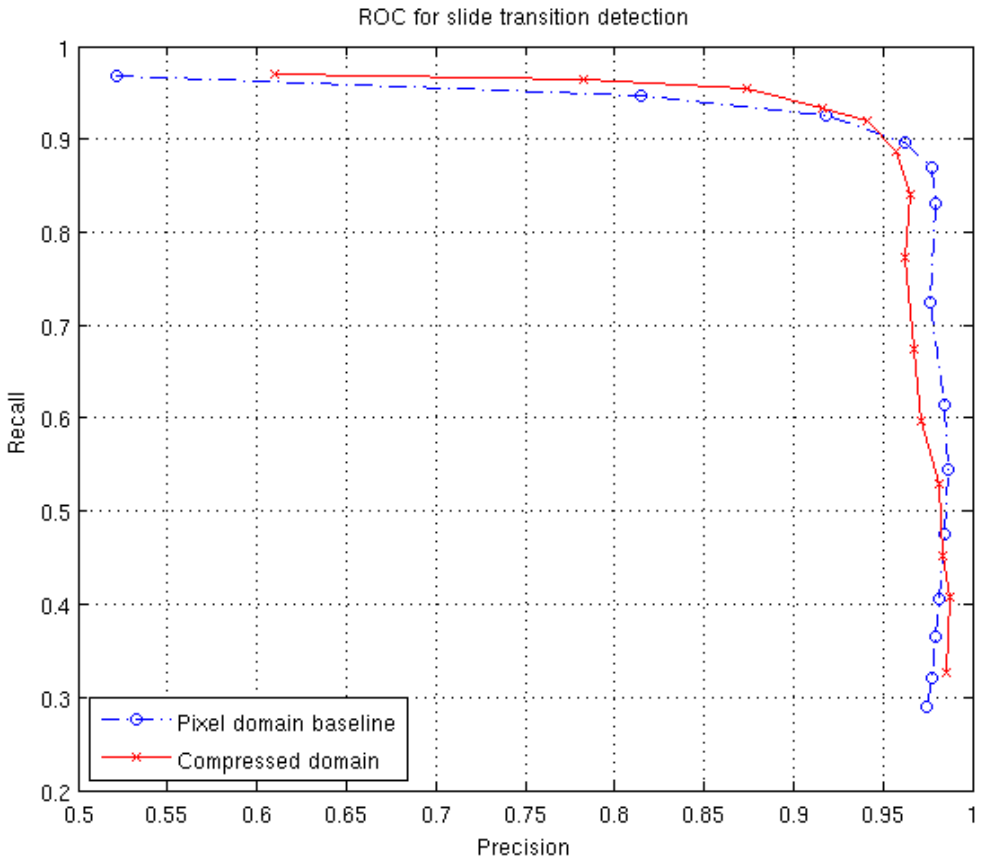


Figure 8.4. ROC plot for slide transition detection

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[4], 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 [5], 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 8.6, we achieve 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

[4]Available at `http://www.xvid.org/`
[5]Available at `http://sourceforge.net/projects/opencvlibrary/`

Table 8.6. 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

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

Effect on VFOA estimation

Our slide transition detector has been used in a state-of-the-art VFOA algorithm proposed by Ba and Odobez that relies on both head pose estimation and contextual cues [9]. A full description of their method is outside the scope of this dissertation; instead, we refer interested readers to their excellent description. Briefly, they use a Dynamic Bayesian Network to jointly infer VFOA, conversational events and model parameters based on observed head pose, speaking activity and slide transition events. In particular, the time since the last slide transition is used to modify the priors that a participant is looking at the projection screen, the table or other participants.

It has been found that using the slide transition contextual cue helps to partly resolve the ambiguities for participants in seat 3 and 4 when they are looking at the participants in seat 1 or 2 or at the projection screen. Specifically, by just using the slide context, there is a 5% improvements in VFOA recognition rate [9]. Furthermore, when combined with additional conversation context, there is a 15% improvement over a baseline that does not use contextual cues (a 10% improvement was observed when just conversation context is used) [9].

8.4 Recapitulation

In this chapter, 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 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.

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 a 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 [150]. 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.

The compressed domain video features used for the two meetings analysis tasks described in this chapter can also be used for a variety of other meetings analysis tasks as well. For example, it has also been used in multi-party dominance modeling [68], audio-visual association [64, 65] and audio-visual speaker diarization [52].

Chapter 9

Conclusions

In this dissertation, we have considered a collection of technical issues, such as video transmission, camera calibration and video analysis, that needs to be resolved in order to realize our "Big-Eye" vision – that of emulating a single expensive high-end video camera with a dense network of cheap low-quality cameras. We will now recapitulate the work that has been done and summarize avenues for future work.

In Part I of the dissertation, we have considered the problem of compressing and transmitting video from multiple camera sensors in a robust and distributed fashion. To exploit redundancy between camera views for robustness, we use a distributed source coding based approach that relies on geometrical constraints in multiple views. Furthermore, our proposed scheme does not require any cooperation between encoders and is suitable for platforms with low computational capabilities.

Some possible directions for future work include:

- Investigating how encoders can independently estimate inter-camera correlation based on intra-camera properties such as edge strength.
- Exploration of low frame rate operating regimes, where inter-camera correlation could possible dominate intra-camera temporal correlation.
- The possible use of feedback from the back-end server to improve compression perfor-

mance. In particular, we believe that a hybrid approach combining motion vector feedback from the decoder [107] and distributed source coding based video coding [105, 57] is very promising.

In Part II of this dissertation, we have investigated the problem of establishing visual correspondences in a distributed and rate-efficient manner. This is especially important in a network of mobile cameras which need to be calibrated continuously. We pose this problem as one of "rate-constrained distributed distance testing" and propose two solutions: one based on distributed source coding exploiting statistical correlation between descriptors of corresponding features and another based on binarized random projections.

We believe this to be a fruitful area of future research. Possible future directions include:

- The proposed binarized random projections scheme shows a relationship between euclidean distance and hamming distance but require that descriptors be of unit norm. One straightforward extension is to consider the relationship between angle and hamming distance, in which case there is no need for the unit norm assumption. It would be interesting to extend this to other metrics or to remove the unit norm assumption.
- The exploration of security properties of the binarized random projections scheme.
- Alshwede and Csiszár showed that the Slepian-Wolf rate region is needed even if only the hamming distance between two correlated binary vectors is desired [4]. It will be interesting to determine the rate region in the case where we only want to know if *the hamming distance is less than a threshold*.

Finally, in Part III of this dissertation, we have focused on the problem of efficient video processing for multiple camera networks. Our approach to efficient video analysis is the use of compressed domain processing to minimize the amount of computations in feature extraction. We have demonstrated its use and effectiveness in the tasks of human action recognition, localization and organization. We have also explored its use in meetings analysis tasks such as dominance modeling and slide change detection.

Possible directions for future work in this part of the dissertation include:

- The use of more sophisticated classifiers such as Support Vector Machines for action classification.
- The use of an hierarchical approach to effectively and efficiently use both compressed domain techniques and pixel domain techniques.
- Extending our action classification scheme to handle larger variations in spatio-temporal scales.
- An exhaustive study and evaluation of organization techniques, in addition to hierarchical agglomerative clustering, to perform automatic or minimally supervised organization of action videos. In particular, affinity propagation [51] seems like a very attractive approach since it performs clustering based on user-defined similarity between data points.
- Further exploration of compressed domain features for various meetings analysis tasks such as audio-visual association [64, 65] and audio-visual speaker diarization [52].

Clearly, there remains much work to be done to make the "Big-Eye" vision a reality. In this dissertation, we hope to have laid down some of the groundwork and to have provided ideas for future investigations.

Bibliography

[1] A. Aaron, R. Zhang, and B. Girod, "Wyner-Ziv coding of motion video," in *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, vol. 1, 2002.

[2] J. K. Aggarwal and Q. Cai, "Human motion analysis: a review," in *Proc. IEEE Nonrigid and Articulated Motion Workshop*, 1997, pp. 90–102.

[3] P. Ahammad, C. Yeo, K. Ramchandran, and S. S. Sastry, "Unsupervised discovery of action hierarchies in large collections of activity videos," in *Proc. IEEE International Workshop on Multimedia Signal Processing*, 2007.

[4] R. Ahlswede and I. Csiszár, "To get a bit of information may be as hard as to get full information," *IEEE Transactions on Information Theory*, vol. 27, no. 4, pp. 398–408, 1981.

[5] N. Ambady, F. J. Bernieri, and J. A. Richeson, "Toward a histology of social behavior: Judgmental accuracy from thin slices of the behavioral stream," *Advances in Experimental Social Psychology*, vol. 32, pp. 201–257, 2000.

[6] E. L. Andrade, S. Blunsden, and R. B. Fisher, "Hidden markov models for optical flow analysis in crowds," in *Proc. International Conference on Pattern Recognition*. IEEE Computer Society Washington, DC, USA, 2006, pp. 460–463.

[7] S. Avidan and A. Shashua, "Novel view synthesis by cascading trilinear tensors," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 4, pp. 293–306, 1998.

[8] H. Aydinoglu and M. H. Hayes, "Compression of multi-view images," in *Proc. IEEE International Conference on Image Processing*, 1994.

[9] S. O. Ba and J.-M. Odobez, "Multi-person visual focus of attention from head pose and meeting contextual cues," IDIAP, IDIAP-RR 47, August 2008, iDIAP-RR 08-47.

[10] R. V. Babu, B. Anantharaman, K. R. Ramakrishnan, and S. H. Srinivasan, "Compressed domain action classification using HMM," *Pattern Recognition Letters*, vol. 23, no. 10, pp. 1203–1213, Aug. 2002.

[11] R. V. Babu and K. R. Ramakrishnan, "Compressed domain human motion recognition using motion history information," in *Proc. IEEE International Conference on Image Processing*, Barcelona, Spain, Sept. 2003, pp. 321–324.

[12] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[13] A. Barton-Sweeney, D. Lymberopoulos, and A. Savvides, "Sensor localization and camera calibration in distributed camera sensor networks," in *Proc. IEEE Basenets*, October 2006.

[14] A. Berg, T. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 26–33.

[15] P. Bickel and K. Doksum, *Mathematical statistics: basic ideas and selected topics. Vol. 1*, 2nd ed. Prentice Hall, 2000.

[16] J. K. Burgoon and N. E. Dunbar, "Nonverbal expressions of dominance and power in human relationships," *The Sage Handbook of Nonverbal Communication*, pp. 279–297, 2006.

[17] 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," in *Proc. Machine Learning for Multimodal Interaction*, 2005.

[18] S.-C. Chan, K.-T. Ng, Z.-F. Gan, K.-L. Chan, and H.-Y. Shum, "The compression of simplified dynamic light fields," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003.

[19] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, J. Singh, and B. Girod, "Transform Coding of Image Feature Descriptors," in *Proc. SPIE Visual Communication and Image Processing*, Jan 2009.

[20] S.-F. Chang, "Compressed-domain techniques for image/video indexing and manipulation," in *Proc. IEEE International Conference on Image Processing*, 1995, pp. 314–317.

[21] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "A fully automated content-based video search engine supporting spatiotemporal queries," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 602 – 615, Sept. 1998.

[22] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. ACM symposium on Theory of Computing*, 2002, pp. 380–388.

[23] M. Chen and A. Willson Jr, "Rate-Distortion Optimal Motion Estimation Algorithms for Motion-Compensated Transform Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 2, p. 147, 1998.

[24] P. W.-C. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. J. Lobaton, M. L. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, D. Tygar, and S. S. Sastry, "Citric: A low-bandwidth wireless camera network platform," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-50, May 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-50.html

[25] S. E. Chen and L. Williams, "View interpolation for image synthesis," in *Proc. International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 1993, pp. 279–288.

[26] Z. Cheng, D. Devarajan, and R. J. Radke, "Determining vision graphs for distributed camera networks using feature digests," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. Article ID 57 034, 11 pages, 2007.

[27] 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.

[28] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H. 263+: video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849–866, 1998.

[29] T. Cover and J. Thomas, *Elements of information theory*. Wiley New York, 1991.

[30] J. Davis and A. Bobick, "The representation and recognition of action using temporal templates," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 928–934.

[31] A. P. Dempster, N. M. Laird, and D. B. 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.

[32] D. Devarajan and R. Radke, "Distributed metric calibration of large camera networks," in *Proc. Workshop on Broadband Advanced Sensor Networks*, 2004.

[33] N. Dimitrova and F. Golshani, "Rx for semantic video database retrieval," in *Proc. ACM international Conference on Multimedia*. New York, NY, USA: ACM Press, 1994, pp. 219–226.

[34] S. L. Dockstader and A. M. Tekalp, "Multiple camera tracking of interacting and occluded human motion," *Proc. of the IEEE*, vol. 89, no. 10, pp. 1441–1455, Oct 2001.

[35] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proc. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005, pp. 65–72.

[36] J. F. Dovidio and S. L. Ellyson, "Decoding visual dominance: Attributions of power based on relative percentages of looking while speaking and looking while listening," *Social Psychology Quarterly*, vol. 45, no. 2, pp. 106–113, 1982.

[37] I. Downes, L. B. Rad, and H. Aghajan, "Development of a mote for wireless image sensor networks," in *Proc. COGnitive systems with Interactive Sensors (COGIS)*, March 2006.

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

[39] N. E. Dunbar and J. K. Burgoon, "Perceptions of power and interactional dominance in interpersonal relationships," *Journal of Social and Personal Relationships*, vol. 22, no. 2, pp. 207–233, 2005.

[40] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Proc. IEEE International Conference on Computer Vision*, Nice, France, Oct. 2003.

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

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

[43] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Simultaneous object recognition and segmentation by image exploration," in *Proc. European Conference on Computer Vision*, vol. 1. Springer, 2004, pp. 40–54.

[44] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[45] M. Flierl and B. Girod, "Video coding with motion-compensated lifted wavelet transforms," *Signal Processing: Image Communication*, vol. 19, no. 7, pp. 561–575, 2004.

[46] ——, "Coding of multi-view image sequences with video sensors," in *Proc. IEEE International Conference on Image Processing*, 2006, pp. 609–612.

[47] ——, "Multiview video compression," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 66–76, Nov. 2007.

[48] G. D. Forney Jr and M. D. Trott, "Sphere-bound-achieving coset codes and multilevel coset codes," *IEEE Transactions on Information Theory*, vol. 46, no. 3, pp. 820–850, 2000.

[49] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt, "Real-time stereo by using dynamic programming," in *Proc. CVPR Workshop on Real-time 3D Sensors and Their Use*, 2004.

[50] U. Franke and A. Joos, "Real-time stereo vision for urban traffic scene understanding," in *Proc. IEEE Intelligent Vehicles Symposium*, 2000, pp. 273–278.

[51] B. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, p. 972, 2007.

[52] G. Friedland, H. Hung, and C. Yeo, "Multi-modal speaker diarization of real-world meetings using compressed-domain video features," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009.

[53] D. L. Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[54] R. G. Gallager, "Low-density parity-check codes," *MIT Press*, 1963.

[55] N. Gehrig and P. L. Dragotti, "DIFFERENT - Distributed and fully flexible image encoders for camera sensor networks," in *Proc. IEEE International Conference on Image Processing*, vol. 2, 2005, pp. 690–693.

[56] ——, "Distributed compression of multi-view images using a geometrical coding approach," in *Proc. IEEE International Conference on Image Processing*, Sep 2007.

[57] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. of the IEEE*, vol. 93, no. 1, pp. 71–83, Jan 2005.

[58] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[59] X. Guo, Y. Lu, F. Wu, W. Gao, and S. Li, "Distributed multi-view video coding," in *Proc. SPIE Visual Communications and Image Processing*, Jan 2006.

[60] J. Hair, R. Anderson, R. Tatham, and W. Black, *Multivariate Data Analysis*, 4th ed. New York, NY: Prentice Hall, 1995.

[61] T. Han and S. Amari, "Statistical inference under multiterminal data compression," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2300–2324, 1998.

[62] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[63] H. Hung, D. Jayagopi, C. Yeo, G. Friedland, S. Ba, J. M. Odobez, K. Ramchandran, N. Mirghafori, and D. Gatica-Perez, "Using audio and video features to classify the most dominant person in a group meeting," in *Proc. ACM International Conference on Multimedia*. ACM Press New York, NY, USA, 2007, pp. 835–838.

[64] H. Hung, Y. Huang, C. Yeo, and D. Gatica-Perez, "Correlating audio-visual cues in a dominance estimation framework," in *Proc. CVPR Workshop on Human Communicative Behavior Analysis*, 2008.

[65] H. Hung, C. Yeo, and G. Friedland, "Approaching on-line audio-visual association through aspects of human discourse," *Computer Vision and Image Understanding*, under review.

[66] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 371–376, May 1977.

[67] P. Ishwar, V. M. Prabhakaran, and K. Ramchandran, "Towards a theory for video coding using distributed compression principles," in *Proc. IEEE International Conference on Image Processing*, Sep 2003.

[68] D. Jayagopi, H. Hung, C. Yeo, and D. Gatica-Perez, "Predicting the dominant clique in meetings through fusion of nonverbal cues," in *Proc. ACM International Conference on Multimedia*, 2008.

[69] ——, "Modeling dominance in group conversations using nonverbal activity cues," *IEEE Transactions on Audio, Speech and Language Processing*, 2009.

[70] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2992–3006, Oct 2004.

[71] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient Visual Event Detection Using Volumetric Features," in *Proc. IEEE International Conference on Computer Vision*, vol. 1, 2005.

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

[73] J. Körner and K. Marton, "How to encode the modulo-two sum of binary sources," *IEEE Transactions on Information Theory*, vol. 25, no. 2, pp. 219–221, 1979.

[74] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview imaging and 3DTV," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10–21, Nov. 2007.

[75] I. Laptev and T. Lindeberg, "Space-time interest points," in *Proc. IEEE International Conference on Computer Vision*, Nice, France, Oct. 2003.

[76] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM Press, 1999, pp. 16–22.

[77] H. Lee and H. Aghajan, "Collaborative node localization in surveillance networks using opportunistic target observations," in *Proc. ACM International Workshop on Video Surveillance and Sensor Networks*. ACM Press New York, NY, USA, 2006, pp. 9–18.

[78] J. S. Lim, *Two-dimensional signal and image processing*. Prentice Hall, 1990, ch. 10.

[79] S. Lim, L. Davis, and A. Mittal, "Task Scheduling in Large Camera Networks," *Lecture Notes In Computer Science*, vol. 4843, p. 397, 2007.

[80] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Pearson Prentice Hall, 2004.

[81] Y. C. Lin, D. Varodayan, and B. Girod, "Image authentication based on distributed source coding," in *Proc. IEEE International Conference on Image Processing*, Sep 2007.

[82] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[83] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2004.

[84] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. ACM International Workshop on Wireless sensor networks and applications*, 2002, pp. 88–97.

[85] E. Martinian, A. Behrens, J. Xin, and A. Vetro, "View synthesis for multiview video compression," in *Proc. Picture Coding Symposium*, Apr 2006.

[86] E. Martinian, S. Yekhanin, and J. S. Yedidia, "Secure biometrics via syndromes," in *Proc. Allerton Conference on Communications, Control and Computing*, Sep 2005.

[87] W. Matusik and H. Pfister, "3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 814–824, Aug 2004.

[88] 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.

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

[90] K. Mikolajczyk and C. Schmid, "Scale and Affine Invariant Interest Point Detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[91] ——, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[92] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool, "A Comparison of Affine Region Detectors," *International Journal of Computer Vision*, vol. 65, no. 1, pp. 43–72, 2005.

[93] S. Milani, J. Wang, and K. Ramchandran, "Achieving H.264-like compression efficiency with distributed video coding," in *Proc. SPIE Visual Communications and Image Processing*. SPIE, 2007.

[94] Mitsubishi Electric Research Laboratories, "MERL multiview video sequences," ftp://ftp.merl.com/pub/avetro/mvc-testseq.

[95] K. T. Mullen, "The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings," *The Journal of Physiology*, vol. 359, no. 1, pp. 381–400, 1985.

[96] K. Muller, P. Merkle, and T. Wiegand, "Compressing time-varying visual content," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 58–65, Nov. 2007.

[97] C. W. Ngo, T. C. Pong, and H. J. Zhang, "On clustering and retrieval of video shots through temporal slices analysis," *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 446–458, 2002.

[98] S. Oh, L. Schenato, P. Chen, and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: System design, algorithms and experiments," *Proc. of the IEEE*, vol. 95, pp. 234–254, 2007.

[99] M. Ouaret, F. Dufaux, and T. Ebrahimi, "Fusion-based multiview distributed video coding," in *Proc. ACM International Workshop on Video Surveillance and Sensor Networks*, Oct 2006.

[100] B. Ozer, W. Wolf, and A. N. Akansu, "Human activity detection in MPEG sequences," in *Proc. IEEE Workshop on Human Motion*, Austin, USA, Dec. 2000, pp. 61–66.

[101] V. Parameswaran and R. Chellappa, "Human action-recognition using mutual invariants," *Computer Vision and Image Understanding*, vol. 98, no. 2, pp. 294–324, 2005.

[102] S. S. Pradhan, J. Chou, and K. Ramchandran, "Duality between source coding and channel coding and its extension to the side information case," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1181–2003, July 2003.

[103] S. S. Pradhan and K. Ramchandran, "Enhancing analog image transmission systems using digital sideinformation: a new wavelet-based image coding paradigm," in *Proc. Data Compression Conference*, 2001, pp. 63–72.

[104] ——, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, Mar 2003.

[105] R. Puri, A. Majumdar, and K. Ramchandran, "PRISM: A video coding paradigm with motion estimation at the decoder," *IEEE Transactions on Image Processing*, vol. 16, no. 10, pp. 2436–2448, 2007.

[106] R. Puri and K. Ramchandran, "PRISM: A new robust video coding architecture based on distributed compression principles," in *Proc. Allerton Conference on Communication, Control and Computing*, 2002.

[107] W. Rabiner and A. Chandrakasan, "Network-driven motion estimation for wireless video terminals," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 644–653, 1997.

[108] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation," in *Proc. ACM Conference on Embedded Networked Sensor Systems*, November 2-4 2005.

[109] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[110] S. Roy and Q. Sun, "Robust hash for detecting and localizing image tampering," in *Proc. IEEE International Conference on Image Processing*, Sep 2007.

[111] M. S. Ryoo and J. K. Aggarwal, "Recognition of composite human activities through context-free grammar based representation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006.

[112] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?"," in *Proc. European Conference on Computer Vision*, vol. 1. Springer, 2002, pp. 414–431.

[113] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[114] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach." in *Proc. International Conference on Pattern Recognition*, Cambridge, UK, Aug. 2004, pp. 32–36.

[115] S. Se, D. Lowe, and J. Little, "Global localization using distinctive visual features," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and System*, vol. 1, 2002.

[116] A. Sehgal, A. Jagmohan, and N. Ahuja, "Wyner-Ziv coding of video: an error-resilient compression framework," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 249–258, Apr 2004.

[117] E. Shechtman and M. Irani, "Space-time behavior based correlation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, USA, June 2005, pp. 405–412.

[118] H. Shum and S. B. Kang, "A review of image-based rendering techniques," in *Proc. SPIE Visual Communications and Image Processing*, vol. 4067, no. 1. SPIE, 2000, pp. 2–13.

[119] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1020–1037, Nov 2003.

[120] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[121] A. Smolic, K. Mueller, P. Merkle, T. Rein, M. Kautzner, P. Eisert, and T. Wiegand, "Free viewpoint video extraction, representation, coding and rendering," in *Proc. IEEE International Conference on Image Processing*, Oct 2004.

[122] B. Song, O. Bursalioglu, A. K. Roy-Chowdhury, and E. Tuncel, "Towards a multi-terminal video compression algorithm using epipolar geometry," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006.

[123] D. Stinson, *Cryptography: Theory and Practice*. CRC Press, Inc. Boca Raton, FL, USA, 1995.

[124] G. J. Sullivan and R. L. Baker, "Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks," in *Proc. IEEE Global Telecommunications Conference*, vol. 3, 1991, pp. 85–90.

[125] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[126] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. M. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communication of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[127] T. Teixeira, D. Lymberopoulos, E. Culurciello, Y. Aloimonos, and A. Savvides, "A lightweight camera sensor network operating on symbolic information," in *Proc. Workshop on Distributed Smart Cameras*, Boulder, Colorado, September 2006.

[128] I. Tosic and P. Frossard, "Coarse scene geometry estimation from sparse approximations of multi-view omnidirectional images," in *Proc. European Signal Processing Conference*, 2007.

[129] ——, "Wyner-ziv coding of multi-view omnidirectiona limages with overcomplete decompositions," in *Proc. IEEE International Conference on Image Processing*, Sep 2007.

[130] D. Varodayan, A. Mavlankar, M. Flierl, and B. Girod, "Distributed grayscale stereo image coding with unsupervised learning of disparity," in *Proc. Data Compression Conference*, 2007, pp. 143–152.

[131] U. Wachsmann, R. F. H. Fischer, and J. B. Huber, "Multilevel codes: theoretical concepts and practical design rules," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1361–1391, 1999.

[132] R. Wagner, R. Nowak, and R. Baraniuk, "Distributed compression for sensor networks using correspondence analysis and super-resolution," in *Proc. IEEE International Conference on Image Processing*, vol. 1, 2003, pp. 597–600.

[133] A. Webb, *Statistical Pattern Recognition.* Oxford: Oxford University Press, 1999.

[134] T. Wedi and H. G. Musmann, "Motion-and aliasing-compensated prediction for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 577–586, 2003.

[135] S. Wee, B. Shen, and J. Apostolopoulos, "Compressed-domain video processing," Hewlett-Packard, Tech. Rep. HPL-2002-282, 2002.

[136] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[137] A. D. Wyner and J. Ziv, "The rate distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, Jan 1976.

[138] Y. Yang, V. Stankovic, W. Zhao, and Z. Xiong, "Multiterminal Video Coding," in *Proc. IEEE International Conference on Image Processing*, vol. 3, 2007.

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

[140] C. Yeo, P. Ahammad, and K. Ramchandran, "A rate-efficient approach for establishing visual correspondences via distributed source coding," in *Proc. SPIE Visual Communications and Image Processing*, Jan 2008.

[141] ——, "A rate-efficient approach for establishing visual correspondences via distributed source coding," in *Proc. SPIE Visual Communications and Image Processing*, Jan 2008.

[142] ——, "Rate-efficient visual correspondences using random projections," in *Proc. IEEE International Conference on Image Processing*, Oct 2008.

[143] C. Yeo, P. Ahammad, K. Ramchandran, and S. S. Sastry, "Compressed domain real-time action recognition," in *Proc. IEEE Workshop on Multimedial Signal Processing*, Victoria, BC, Canada, Oct. 2006.

[144] ——, "High-speed action recognition and localization in compressed domain videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1006–1015, 2008.

[145] C. Yeo, P. Ahammad, H. Zhang, and K. Ramchandran, "Rate-constrained distributed distance testing and its applications," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr 2009.

[146] C. Yeo and K. Ramchandran, "Robust distributed multi-view video compression for wireless camera networks," in *Proc. SPIE Visual Communications and Image Processing*, Jan 2007.

[147] ——, "Compressed domain video processing of meetings for activity estimation in dominance classification and slide transition detection," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-79, Jun 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-79.html

[148] ——, "Robust distributed multi-view video compression for wireless camera networks," *IEEE Transactions on Image Processing*, under review.

[149] C. Yeo, J. Wang, and K. Ramchandran, "View synthesis for robust distributed video compression in wireless camera networks," in *Proc. IEEE International Conference on Image Processing*, Sep 2007.

[150] C. Yeo, Y.-W. Zhu, Q. Sun, and S.-F. Chang, "A Framework for Sub-Window Shot Detection," in *Proc. 11th International Multimedia Modelling Conference*, 2005.

[151] M. M. Yeung and B. Liu, "Efficient matching and clustering of video shots," in *Proc. IEEE International Conference on Image Processing*, vol. 1, 1995, pp. 338–341.

[152] A. Yilmaz and M. Shah, "Recognizing human actions in videos acquired by uncalibrated moving cameras," in *Proc. IEEE International Conference on Computer Vision*, vol. 1, 2005.

[153] R. Zamir, "The rate-loss in the Wyner-Ziv problem," *IEEE Transactions on Information Theory*, vol. 42, no. 11, pp. 2073–2084, Nov 1996.

[154] H. Zhang, C. Yeo, and K. Ramchandran, "Vsync — a novel video file synchronization protocol," in *Proc. ACM International Conference on Multimedia*, Oct 2008.

[155] ——, "Rate efficient remote video file synchronization," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr 2009.

[156] Z. Zhang, "Determining the Epipolar Geometry and its Uncertainty: A Review," *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998.

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

[158] X. Zhu, A. Aaron, and B. Girod, "Distributed compression for large camera arrays," in *Proc. IEEE Workshop on Statistical Signal Processing*, 2003, pp. 30–33.