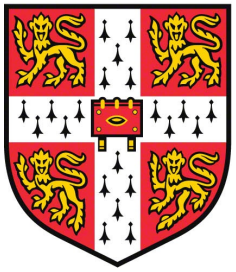
Compressive sensing of images and video: towards low-complexity, real-time operation



Joseph Zammit

Computer Laboratory
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Christ's College December 2020

Declaration

This dissertation is submitted for the degree of Doctor of Philosophy. As required by the University Statues, I hereby declare that this dissertation is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text and Acknowledgements.

This dissertation contains less than the limits of 150 figures and 60,000 words.

Joseph Zammit
December 2020

Acknowledgements

I would first like to thank my PhD supervisor Dr. Ian J. Wassell for his constant support throughout my PhD, and for the opportunity to be able to embark on this journey. He has been very kind and understanding, and I'll be forever in debt for his support throughout the birth of my daughter and the COVID-19 pandemic.

I would also like to thank my wife, my parents and family. Needless to say, completing this long journey without their support would have been impossible.

Special thanks to the Cambridge Trust and the Endeavour Scholarship Scheme (Malta) for funding my studies.

Joseph Zammit
December 2020

Abstract

Compressive Sensing (CS) acquires sparse signals with far fewer measurements than samples required by the classical Nyquist sampling theorem, at the cost of more computationally intensive reconstruction. Video Block Compressive Sensing (VBCS), using a Multi Pixel Camera (MPC), divides the sensed image into blocks reducing storage requirements, and allowing lower latency capture in parallel at the sensor. These approaches suffer from low performance or long reconstruction times. Distributed Video Coding (DVC) techniques emerged at the turn of the millennium for compressing video for transmission from resource constrained devices. The rationale behind this thesis is to combine VBCS and DVC to develop a video transmission system having a low power implementation. The focus is on three areas: the development of real-time, computationally efficient sensor-side CS video frame compression and decoder-side reconstruction techniques, the quantization and coding of image and video measurements for transmission, and the exploitation of temporal correlation between frames to improve performance.

With respect to the compression of independent video frames, this work proposes two novel classes of techniques: Adaptive Block Compressive Sensing (ABCS) and re-purposed single image super resolution. In the first class, two novel algorithms are proposed that adapt the number of measurements in each block using initial measurements in the spatial or discrete cosine transform domains. These achieve State of the Art (SOTA) performance amongst ABCS algorithms. A SOTA Iterative Denoising Algorithm (IDA) is also developed that can deblock and improve the rate-distortion quality, with a time penalty. In the second class, an algorithm (Shrink) is developed that combines a bicubic interpolator with an antialiasing filter to perform downsampling on an MPC. GPU-based reconstruction algorithms are used to super resolve the downsampled images at the decoder in real time improving quality at high compression.

Two algorithms are developed to quantize and code ABCS sensed frames for digital transmission: a JPEG-like algorithm and a novel, low-complexity table based algorithm: TAB. At low compression factors, the best performance is achieved by the JPEG-like algorithm that allows the ABCS techniques to match the JPEG rate-distortion performance even though the frames are compressively sensed. TAB performs better at higher compression factors. Shrink is encoded with a low-complexity algorithm based on Golomb coding and achieves better rate-distortion than the end-to-end ABCS algorithms at high compression.

Temporal correlation is exploited at the encoder to better adapt the number of measurements per block. At the decoder, it increases the quality of the highly compressed non-key frames to match that of key frames that are periodically coded with higher quality. ABCS techniques are found to exploit temporal correlation significantly better than Shrink based techniques. A proposed VBCS algorithm using differential pulse code modulation and video frame interpolation is shown to achieve SOTA results from a dimensionality reduction perspective. Finally, the TAB algorithm is improved to better compress encoder measurements for end-to-end transmission.

Table of contents

List of figures **10**

List of tables **16**

List of algorithms **19**

List of acronyms **20**

Notation **23**

1 Introduction **24**
1.1 Aims and objectives 26
1.2 Contributions 26
1.2.1 Chapter 3 26
1.2.2 Chapter 4 27
1.2.3 Chapter 5 27
1.2.4 Chapter 6 27
1.2.5 Chapter 7 28
1.2.6 Chapter 8 28
1.3 Publications 28
1.4 Dissertation structure 29

2 Preliminaries **30**
2.1 Compressive image sensing 30
2.2 Reconstruction algorithms 32
2.2.1 Approximate message passing 33
2.2.2 Denoising Approximate Message Passing (AMP) 33
2.2.3 Accelerating reconstruction using deep neural networks 34
2.3 Video compressive sensing 37
2.4 Image and video Compressive Sensing (CS) cameras 38
2.4.1 Focal plane processing sensors 38
2.4.2 Single pixel camera 39
2.4.3 CS camera taxonomy 39
2.4.4 Multiple pixel cameras 40

2.4.5 Applications . . . 40
2.4.6 Super resolution . . . 41
2.5 Chapter summary . . . 41

3 Image and video transmission system analysis framework 43
3.1 Introduction . . . 43
3.2 Framework for comparing low-energy sensors . . . 44
3.2.1 Transmission system models . . . 44
3.2.2 Framework parameters . . . 45
3.2.3 Sensors . . . 46
3.2.4 SPC characteristics for CS and SR . . . 46
3.2.5 MPC characteristics for CS schemes . . . 47
3.2.6 vMPC characteristics for CS and super resolution . . . 47
3.2.7 Multi Pixel Camera (MPC) characteristics for super resolution . . . 48
3.2.8 Number of Analogue to Digital Converter (ADC) conversion cycles for Bicubic Convolutional Filter (BCF) on an MPC . . . 48
3.2.9 Verification by simulation . . . 51
3.3 Evaluating CS and Super Resolution (SR) sensors using the framework . . . 52
3.4 Chapter summary . . . 55

4 Adaptive block compressive sensing of images 56
4.1 Introduction . . . 56
4.2 Related work . . . 58
4.3 Image Block-based CS (BCS) using deterministic 2D-DCT sensing matrices . . . 59
4.3.1 L-DCT-ZZ . . . 60
4.3.2 AL-DCT-THB and AL-DCT-THI . . . 60
4.4 Adaptive image BCS . . . 61
4.4.1 Adaptive L-DCT-ZZ using block boundary variation . . . 63
4.4.2 Adaptive L-DCT-ZZ in the Discrete Cosine Transform (DCT) domain . . . 65
4.4.3 Analysis of the Block Boundary Variation (BBV) and DCT Domain (DD) adaptive techniques . . . 68
4.4.4 Real-time deblocking . . . 70
4.5 Reconstructing deterministic CS using iterative denoising-based algorithms . . . 72
4.6 Empirical investigation . . . 76
4.6.1 Comparing the deterministic algorithms using direct and CS reconstruction . . . 79
4.6.2 Comparison with adaptive CS results in the literature . . . 81
4.6.3 Comparison with two state-of-the-art, non-adaptive algorithms . . . 83
4.6.4 Complexity analysis . . . 84
4.7 Chapter summary . . . 85

5 Super resolution of images for compressive sensors 86
5.1 Introduction . . . 86
5.2 Related work . . . 87
5.3 Methodology . . . 90
5.3.1 Implementing the bicubic filter on compressive sensors . . . 90
5.3.2 The Shrink algorithm . . . 91
5.4 Dimensionality reduction results . . . 93
5.4.1 PSNR . . . 93
5.4.2 SSIM . . . 94
5.4.3 Execution time . . . 95
5.5 Discussion of results . . . 95
5.6 Conclusions . . . 96

6 End-to-end transmission of compressively sensed images 97
6.1 Introduction . . . 97
6.2 Related work . . . 99
6.3 End-to-end transmission of adaptive block-based compressively sensed images 99
6.3.1 The linear 2D-DCT algorithms with JPEG-style coding . . . 100
6.3.2 The linear 2D-DCT with low complexity table-based encoding . . . 100
6.4 End-to-End transmission of downsampled images . . . 102
6.4.1 Low-complexity lossless coding . . . 102
6.5 Results: comparison of techniques . . . 106
6.5.1 End-to-end transmission of compressively sensed images with JPEG-style entropy coding . . . 107
6.5.2 End-to-end transmission of compressively sensed images using the TAB algorithm . . . 112
6.5.3 End-to-end ISR: Shrink-LLC-SR . . . 113
6.5.4 Quantization and entropy coding of state-of-the-art CS CNNs . . . 115
6.5.5 Comparison with CNN-based, image CS algorithms . . . 115
6.6 Comparison using the low-energy sensor framework . . . 117
6.6.1 Rate/Distortion performance . . . 117
6.6.2 Energy efficiency . . . 120
6.6.3 Frame capture time . . . 122
6.7 Conclusions . . . 122

7 Distributed video compressive sensing and video super resolution algorithms 125
7.1 Introduction . . . 125
7.2 Related work . . . 128
7.3 Distributed video compressive sensing . . . 129
7.3.1 Overview . . . 129
7.3.2 VAL-MCP system block diagram . . . 130

7.3.3 Adaptive block compressive sensing of video frames . . . 131
7.3.4 IDA reconstruction . . . 133
7.3.5 Motion compensated prediction of non-key frames . . . 133
7.3.6 Differential pulse code modulation . . . 134
7.3.7 Real-time reconstruction . . . 136
7.3.8 Best pixel discriminator . . . 137
7.3.9 VAL-MCP parameters . . . 137
7.4 VAL-MCP experimental results . . . 138
7.5 Improving VAL-MCP using video frame interpolation . . . 143
7.5.1 VAL-VFI system block diagram . . . 144
7.5.2 VAL-VFI results . . . 144
7.6 Distributed video super resolution . . . 146
7.6.1 VSR-MCP system block diagram . . . 146
7.6.2 VSR-MCP results . . . 146
7.7 Improving VSR-MCP with VFI . . . 150
7.7.1 VSR-VFI system block diagram . . . 150
7.7.2 VSR-VFI results . . . 151
7.8 Execution time . . . 151
7.9 Chapter summary . . . 152

8 End-to-end distributed video compressive sensing 154
8.1 Introduction . . . 155
8.2 Related work . . . 156
8.3 End-to-end distributed VCS . . . 158
8.3.1 System diagram . . . 158
8.3.2 Quantization . . . 158
8.3.3 Low-complexity entropy coding . . . 159
8.4 Results . . . 160
8.4.1 Rate-distortion performance of key frames . . . 160
8.4.2 Rate-distortion performance when exploiting temporal correlation . . . 163
8.4.3 Ablation study . . . 164
8.5 Sensor analysis . . . 166
8.5.1 PSNR quality performance . . . 166
8.5.2 Energy to modulate mirrors . . . 166
8.5.3 Energy to perform analogue to digital conversion . . . 167
8.5.4 Encoder complexity . . . 168
8.5.5 Energy for entropy encoding . . . 168
8.5.6 Energy to transmit a video frame . . . 168
8.5.7 Time to capture one frame . . . 169
8.6 Chapter summary . . . 169

9 Conclusions and further work **171**
9.1 Conclusions 171
9.2 Future work 175

References **177**

List of figures

2.1 The spectrum of signal recovery algorithms, from [16] 36
2.2 Single Pixel Camera [57] 39

3.1 Classical and compressive sensing image transmission system models. 44
3.2 BCF on an MPC: examples when the convolutional kernel (green) falls within one block N_1, two blocks N_2 and four blocks N_4. In this example $B = 48 \times 48$, $S_F = 4$ and the kernel is $4.S_F \times 4.S_F = 16 \times 16$. The centre $S_F \times S_F$ kernel coefficients are shown in gray. All the kernels have the kernel coefficients. . . . 49
3.3 Simulation of BCF on an MPC showing kernel cycles with convolutions contained in (a) one MPC block (b) and (c) two blocks, and (d) four blocks. 50

4.1 Adaptive AL-DCT-BBV and AL-DCT-DD system architecture. I_i is the input image, I_o is the image output in real time by the inverse 2D-DCT and $\widehat{I_o}$ is the higher quality image output by the CS reconstruction, for example, using the Iterative Denoising Algorithm (IDA) algorithm. AL-DCT-{BBV|DD} acquires m_i^2 adaptive CS measurements per block where each measurement is a 2D-DCT Transform Coefficient (TC). 62
4.2 Adaptive AL-DCT-DD phase 1 and 2 measurements. 62
4.3 Block boundary variation measurements for a 32x32 block. The shaded measurements are imported from adjacent blocks. 63
4.4 Set256: 256×256 image set. 68
4.5 Set512: 512×512 image set. 68
4.6 Generic block diagram for iterative denoising reconstruction algorithms: Denoising AMP (D-AMP), D-AMP Damped (DAMP-D) and IDA. This depicts equations 2.13, 2.14 and 2.15 in the text. The delay blocks Δ impart a one-iteration delay such that the output is the vector in the previous iteration. The Onsager term is $\alpha^t \mathbf{z^{t-1}}$ with α^t given by equation 4.9 and is derived from the divergence of the $\mathbf{x^t}$ vector from the previous iteration. 72
4.7 The 17-layer Denoising CNN (DnCNN) [27] used to reconstruct grayscale (one colour component, c=1) images. Convolution (Conv), rectified linear unit (ReLU), batch normalization (BN). 73

4.8 Reconstructing AL-DCT-DD using (a) D-AMP (when $D_F = 1.00$) and IDA (when $D_F > 1.00$) with a Block-matching and 3D filtering (BM3D) denoiser and (b) using D-AMP and IDA with a DnCNN denoiser. The results shown for the 256×256 Monarch image sensed at $C_R = 0.1$. 74
4.9 Comparison of the PSNR for the 256×256 images reconstructed using AL-DCT-DD-IDA with BM3D and DnCNN denoisers at $C_R = 0.1$. 75
4.10 PSNR versus D_F for [L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA for the 256×256 image set, with compression ratio $C_R = 0.1$. 76
4.11 Visual Quality of the [A]L-DCT-{ZZ|BBV|DD}-[ZZF|THF|BM3DF|IDA] on the *Lena* image in Set256 at a compression ratio of $C_R = 0.1$. Blue PSNR and SSIM values are worse than those of ZZ-IDCT in the top left corner. Red values are better. Decoding time is reported in seconds. 81
4.12 Details from the corresponding images in figure 4.11. 81
4.13 Set12: The first seven images from the left have a resolution of 256×256 and the last five are 512×512. The fourth and fifth images from the left are *Starfish* and *Monarch*. 82
4.14 Set11: The images have a resolution of 256×256, except for the fourth and fifth from the left, *Fingerprint* and *Flintstones*, that have a resolution of 512×512 pixels. 82

5.1 Block diagram of the proposed JPEG-based interpolative scheme from [100] . . 87
5.2 The Shrink algorithm implements bicubic interpolation at the compressive sensor to compress the image that is then super resolved using state-of-the-art super resolution algorithms such as Information Multi-Distillation Network (IMDN) [109]. 90
5.3 1D imresize bicubic function kernels for $S_F = 2, 3 \ldots 6$. 92
5.4 2D Shrink cubic convolutional interpolation kernels: N_{NZ} is the number of non-zero coefficients and U is the number of unique values quantized to 8 Bits including sign bit. 92
5.5 (a) PSNR versus M/N for Set14 (b) SSIM versus M/N for Set14, from M/N = 0.01 to 0.25. 93
5.6 (a) PSNR versus M/N for Set14 (b) SSIM versus M/N for Set14, from M/N = 0.01 to 0.50. 94
5.7 Execution Time versus M/N 95

6.1 Super resolution with Low-Complexity Lossless Coding (LC-LLC) 102
6.2 Histogram (blue) of the pixel difference count and the histogram bit profile (orange) for baboon.bmp in Set14 [107]. The plateaus of the histogram bit profile equal 2^b where b is the number of bits used to encode the histogram values below the plateaus. 103

6.3 Efficiency achieved by the five parameter m estimators; (1) FS, (2) EE1, (3) EE2, (4) MHE, (5) PSS, for seven image sets at six subsampling rates $S_F = 1 \cdots 6$, applied in the horizontal and vertical dimensions. S_F=1 means no subsampling - use Differential Pulse Code Modulation (DPCM) on original image. 104
6.4 Execution time for the Golomb parameter m Estimators; (1) FS, (2) EE1, (3) EE2, (4) MHE, (5) PSS, for seven image sets and sub-sampling rates $S_F = 1 \cdots 6$, applied in the horizontal and vertical dimensions. S_F=1 means no subsampling - use DPCM on original image. 105
6.5 (a) PSNR and (b) SSIM performance of Shrink-Golomb-Bicubic algorithm with the five different Golomb parameter m estimators for Set14. Downsampling using the Shrink algorithm followed by bicubic upsampling. The plots show that the PSNR and SSIM performance of the five algorithms are practically the same and it is possible to use the least complex algorithm, that is EE1. 107
6.6 Comparison of (a) PSNR and (b) SSIM for L-DCT-ZZ-JPEG quantized and coded using the JPEG style algorithm at different $C_R = M/N$ ratios, with standard JPEG, on Set14. The plots show that L-DCT-ZZ-JPEG can achieve the performance of JPEG but measuring a fraction - C_R - of all the transform coefficients. 108
6.7 Comparison of (a) PSNR and (b) SSIM for L-DCT-ZZ-JPEG-IDA quantized and coded using the JPEG style algorithm at different $C_R = M/N$ ratios, against standard JPEG, on Set14. The plots show that L-DCT-ZZ-JPEG-IDA can substantially exceed the performance of JPEG even when measuring a fraction - C_R - of all the transform coefficients. 109
6.8 (a) PSNR and (b) SSIM vs bpp for Better Portable Graphics (BPG) [123], JPEG 2000 [55], JPEG [54], L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD with $B = 8$ on Set14. The plots show that L-DCT-ZZ-JPEG-RD matches the performance of JPEG, whereas L-DCT-ZZ-JPEG-IDA-RD exceeds it significantly. L-DCT-ZZ-JPEG-IDA-RD matches the PSNR performance of BPG and JPEG between C_R = 0.6 and C_R = 0.7 whereas its SSIM exceeds that of all other algorithms above C_R = 0.3. 110
6.9 (a) PSNR and (b) SSIM versus Rate in bpp for L-DCT-ZZ-TAB with blocksize B=32 and various compression ratios, on Set14. The compression ratio points on the TAB curves were generated using $C_F = \{1.5, 2, 3, 4, 6, 12, 17, 35, 45, 65, 106\}$. The plots shows that the PSNR performance of AL-DCT-ZZ-TAB is better than JPEG below C_R = 0.3. 111
6.10 (a) PSNR and (b) SSIM versus Rate in bpp for L-DCT-ZZ-TAB-IDA with blocksize B=32 and various compression ratios, on Set14. The compression ratio points on the TAB curves were generated using $C_F = \{1.5, 2, 3, 4, 6, 12, 17, 35, 45, 65, 106\}$. The plots shows that the PSNR performance of AL-DCT-ZZ-TAB-IDA is better than JPEG below C_R = 0.4. 112

6.11 (a) PSNR versus bpp and (b) SSIM versus bpp for the TAB algorithm variants with B=32 on Set14. The plots show that the ZZ variants perform better than the DD variants and significantly better than the BBV variants. 113
6.12 (a) PSNR versus Bpp for (b) SSIM versus Bpp for the L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-IDA-RD, L-DCT-ZZ-TAB-RD, JPEG 2000 and JPEG on Set14. The plots show that there is a rate above which it is better to use L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD algorithms, but below which it is better to use L-DCT-ZZ-TAB-RD and L-DCT-ZZ-TAB-IDA-RD. 114
6.13 PSNR for Set14 - JPEG 2000 (JP2), JPEG, L-DCT-ZZ-JPEG, Shrink-LLC-IMDN, Shrink-LLC-VDSR. The plots show that Very Deep Super Resolution (VDSR) upsampling results in a significant improvement in PSNR over bicubic upsampling, for the losslessly coded (LLC) variants following downsampling using Shrink. The Lossless compression schemes investigated are JPEG 2000 (JP2), PNG, Lossless JPEG (LS-JPG) and our DPCM algorithms that compress the difference with Huffman coding (HUFF), Arithmetic Coding (AC) and Golomb coding with m parameter estimator EE1. Below $C_R = 0.2$ the performance of the Shrink-LLC-VDSR algorithm is better than L-DCT-ZZ-JPEG-RD and JPEG. IMDN has the potential to improve the VDSR results. 115
6.14 SSIM for Set14 - JPEG 2000 (JP2), JPEG, L-DCT-ZZ-JPEG, Shrink-LLC-IMDN, Shrink-LLC-VDSR. The plots show that VDSR upsampling results in a significant improvement in SSIM over bicubic upsampling, for the losslessly coded (LLC) variants following downsampling using Shrink. The Lossless compression schemes investigated are JPEG 2000 (JP2), PNG, Lossless JPEG (LS-JPG) and our DPCM algorithms that compress the difference with Huffman coding (HUFF), Arithmetic Coding (AC) and Golomb coding with m parameter estimator EE1. Below $C_R = 0.2$ the performance of Shrink-LLC-VDSR algorithm is better than L-DCT-ZZ-JPEG-RD and JPEG. IMDN has the potential to improve the VDSR results. 116
6.15 (a) PSNR vs bpp and (b) SSIM vs bpp for uniformly quantized CS schemes. The plots show that CS-NET performs best above a rate of 1.6 bpp. Below 1.6 bpp, DR2-Net performs better than ISTA-NET+ and CS-NET. 117
6.16 Comparison of (a) PSNR and (b) SSIM for L-DCT-ZZ-{JPEG|TAB}-[IDA]-RD, Shrink-PNG-{Bicubic|IMDN|VDSR}, JPEG 2000, JPEG, and CS CNN-based schemes on Set14 (a) PSNR (b) SSIM. The plots show that our L-DCT-ZZ-based and Shrink-based schemes perform better than the uniformly quantized CS CNN-based schemes, even the real-time schemes that do not employ IDA reconstruction. 118

6.17 Comparison of JPEG 2000, JPEG, L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-IDA-RD, L-DCT-ZZ-TAB-RD, Shrink-Golomb-IMDN, Shrink-Golomb-VDSR and Shrink-Golomb-Bicubic on Set480. The plots investigate the performance at low bit rates of interest for both real-time and IDA reconstruction. Below 0.2 bpp, the best real-time performance is provided by Shrink-VDSR. However, above 0.2 bpp, L-DCT-ZZ-JPEG-RD provides the best performance. The non-real time L-DCT-ZZ-JPEG-IDA-RD returns the best performance by far, exceeding the JPEG 2000 PSNR between 0.15 bpp and 0.55 bpp, and its SSIM above 0.1 bpp. 119

7.1 Block diagram of the distributed video CS and video SR schemes proposed and studied in this chapter 7. End-to-End systems are studied in chapter 8. 126
7.2 Block diagram of the distributed VAL-MCP algorithm. The AL-DCT-DD decoder has optional IDA reconstruction. 131
7.3 GOP Structure. n is the Group of Pictures (GOP) number. 131
7.4 Non-distributed, non-compressive DPCM. Key frame blocks $\mathbf{K}$ are 2D DCT transformed to generate transform coefficients $\mathbf{T}_K$, that are lowpass filtered by Hadamard multiplication with lowpass matrix $(\mathbf{L}+\mathbf{M})$, to output lowpass transform coefficients $\mathbf{C}_K$. Non-key frame blocks $\bar{\mathbf{K}}$ and the locally reconstructed key reference blocks $\hat{\mathbf{K}}$ are subtracted to form the difference blocks Δ that when 2D-DCT transformed produce the transform coefficients $\mathbf{T}_\Delta$ of the difference blocks. These are lowpass filtered by Hadamard multiplication with the lowpass filter matrix $\mathbf{L}$ to produce lowpass difference coefficients $\mathbf{C}_\Delta$. At the decoder, $\mathbf{C}_K$ and $\mathbf{C}_\Delta$ are inverse 2D DCT transformed to produce $\hat{\mathbf{K}}$ and $\hat{\Delta}$ that added together produce the reconstructed non-key blocks $\hat{\bar{\mathbf{K}}}$. 135
7.5 Distributed compressive DPCM. $\mathbf{CS}_K$ and $\mathbf{CS}_{\bar{K}}$ represent the CS operations on the key and non-key blocks. $\mathbf{K}$ and $\bar{\mathbf{K}}$ are the reference and non-reference blocks, $\mathbf{C}_K$ are the filtered transform coefficients of the key frame, $\mathbf{C}_{\bar{K}}$ are the filtered transform coefficients of the non-key block, $\mathbf{M}$ is a mid-band filter, $\mathbf{C_k} \odot \mathbf{M}$ are the mid-band transform coefficients of the key block, $\mathbf{D^{-1}}$ is the inverse DCT and $\hat{\bar{\mathbf{K}}}$ is the reconstructed non-key block. 136
7.6 Comparison of Peak Signal-to-Noise Ratio (PSNR) per frame for the six algorithms. All sequences were measured using $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ with a GOP size of 8, except for VAL-IDA-MCP* for which the respective values are 0.4 and 0.1 and GOP size is 4. 141
7.7 Comparison of Multiscale SSIM (MS-SSIM) per frame for the six algorithms. All sequences were measured using $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ with a GOP size of 8, except for VAL-IDA-MCP* for which the respective values are 0.4 and 0.1 and GOP size is 4. 142

7.8 Frame 6 of the Foreman sequence (a) Original (b) Motion Estimation/Motion Compensation (ME/MC) (c) Reweighted Residual Sparsitys (RRSs) (d) VAL-MCP (e) VAL-IDA-MCP* (f) VAL-IDA-MCP. 143
7.9 Block diagram of the distributed VAL-VFI algorithm. The Video Frame Interpolation (VFI) block replaces the bi-directional Motion Compensated Prediction (MCP) in figure 7.2. 144
7.10 Block diagram of the distributed VSR-MCP algorithm. 146
7.11 Block diagram of the distributed VSR-VFI algorithm. 151

8.1 Block diagram of the distributed VAL-[IDA]-{MCP|VFI}-TAB algorithm . . . 158
8.2 VAL-TP and VAL-TP-IDA compared with the intra mode of H263+ and H264/AVC when coding 144 frames of the four Quarter CIF (QCIF) video sequences Coastguard, Foreman, Hall and Soccer, sub-sampled to 15 frames per second. The plotted VAL-TP and VAL-TP-IDA points were measured with a compression ratio $\delta_k = \{0.10..0.50\}$. 161
8.3 VAL-TP-[IDA]-{MCP|VFI} compared to DIStributed COding for Video sERvices (DISCOVER) and the intra mode of H264/AVC on the Coastguard and Hall QCIF sequences. 162
8.4 VAL-TP-[IDA]-{MCP|VFI} compared to DISCOVER and the intra mode of H264/AVC on the Foreman and Soccer QCIF sequences. 163
8.5 Improvement in VAL PSNR performance with MCP-DPCM and IDA 165
8.6 Improvement in VAL Structural Similarity Index Measure (SSIM) performance with MCP-DPCM and IDA 166
8.7 Comparison between VAL-TP-IDA-VFI and VLZ-TP-IDA-VFI. 167

List of tables

1 List of main algorithms in each chapter. Components in square brackets are optional and only one acronym in curly brackets, separated by an OR | symbol, is to be included. For example, AL-DCT-{BBV|DD}-[IDA] refers to the four adaptive algorithms that may optionally be reconstructed using the IDA algorithm, namely, AL-DCT-BBV, AL-DCT-DD, AL-DCT-BBV-IDA and AL-DCT-DD-IDA. 19

3.1 Comparing N_C calculated using equation 3.12 and measured by simulation, when $H = 480$ and $W = 480$. 52
3.2 Calculating the framework parameters for CS and SR sensors. 53
3.3 Framework symbols used in this chapter and table 3.2. 54

4.1 Optimizing the DCT Domain (DD) adaptivity algorithm threshold T empirically to maximize PSNR. Largest values in bold. 67
4.2 Values of T based on image height H and C_F. The dynamic range of the pixel values is [0 255]. 68
4.3 Correlation coefficient $\rho_{m_i,\widehat{m}_i}$ for 256×256 and 512×512 image sets for both BBV and DD adaptive algorithms with varying C_R. Correlation values below 0.7 are shown in bold. 69
4.4 Comparison of IDCT reconstruction and ZZF, THF and BM3DF deblocking on the Set256 image set, with best PSNR and SSIM in bold. 70
4.5 Comparison of IDCT reconstruction and ZZF, THF and BM3D deblocking on the 512×512 Image Set, with best PSNR and SSIM in bold. 71
4.6 L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD with IDCT, D-AMP and IDA reconstruction on the 256×256 image set, using the DnCNN denoiser. Maximum PSNR and SSIM values for all compression ratios are in bold. 77
4.7 L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD with IDCT, D-AMP and IDA reconstruction on the 512×512 image set, using the DnCNN denoiser. Maximum PSNR and SSIM values for all compression ratios are in bold. 78
4.8 Comparison of L-DCT-ZZ, AL-DCT-DD and AL-DCT-BBV with ABCS results in the literature derived on the image sets in the literature, using the compression ratios in the referenced publication. Best results are highlighted in bold. The underlined results are the only inferior L-DCT-ZZ results. 79

4.9 Comparison of AL-DCT-BBV-IDA and AL-DCT-DD-IDA using the DnCNN denoiser with adaptive CS algorithms in the literature using the indicated reconstruction algorithm. The results in each PSNR column were derived using the image set in the cited publication, using the compression ratios in the referenced publication. 80
4.10 Comparing CREAM PSNR and SSIM with L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA with the DnCNN denoiser, with $D_F = 2.0$ and 15 iterations, on Set12. Maximum PSNR/SSIM values are in bold and second largest underlined. 82
4.11 Comparing BCS-Net PSNR and SSIM with L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA with the DnCNN denoiser, with $D_F = 2$ and 15 iterations, on Set11. Maximum PSNR values are in bold and second largest underlined. 83
4.12 Comparison of reconstruction times for L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD using 2D-IDCT and IDA reconstruction with the DnCNN denoiser on the 256×256 image set, $D_F = 2$ and 20 iterations. 84

6.1 TAB algorithm table structure. Values in the top three rows are coded and transmitted using the number of bits in the first column. AC and DC values are bit-encoded using the number of bits calculated using equation 6.2. 101
6.2 L-DCT-ZZ-JPEG-[IDA]-RD Rate/Distortion values, on Set14. C_R is the compression ratio, Q_F is the quantization matrix multiplication factor, and Bpp the rate in bits per pixel. This table shows the quality factors Q_F of the JPEG algorithm that when combined with the compression ratio C_R of the L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD algorithms, generate the rate distortion points plotted in figure 6.8. 111
6.3 Comparison of PSNR (dB) and SSIM of L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-RD, and Shrink-GOLOMB-EE1 on Set480 at different rates in bpp generated by different compression ratios $CR = 1/S_F^2$. These points are plotted in figure 6.17. 120
6.4 Comparison of parameters for L-DCT-ZZ-JPEG, L-DCT-ZZ-TAB and Shrink algorithms. 121
6.5 Comparison of CS (e.g. L-DCT-ZZ) and SR (e.g. Shrink) on a 480×480 image with varying block size B and sub-sampling factor S_F. C_F is the measured compression factor, $n_S = B/S_F$ is the number of Shrink sub-samples in one-dimension of a block of size B at shrink factor S_F, M is the number of compressed measurements, K is the one-dimensional filter size in Shrink, M_B is the number of measurements in a block, n_B is the number of blocks, N_{MEM} is the memory size to store the measurement matrix, N_M is the number of DMD mirror modulations, N_{ADC} is the number of ADC conversions and N_C is the number of ADC conversion cycles to capture one image. 123

7.1 Comparison of ME/MC, RRS, SGS-OF, VAL-MCP and VAL-IDA-MCP. VAL-MCP executes in real-time. All algorithms use the L-DCT-ZZ measurement matrix, GOP = 8, $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ except VAL-IDA-MCP* for which GOP = 4, and $\delta_K = 0.4$. Best results in bold, second best underlined. 139
7.2 Comparison of VAL-MCP for $\delta_K = 0.7 \ldots 0.175$ and $\delta_{\bar{K}} = 0.1 \ldots 0.175$ with GOP = 8. No post-reconstruction filtering is employed. Best results in bold. . . 140
7.3 Comparison of VAL-IDA-MCP for $\delta_K = 0.7..0.175$ and $\delta_{\bar{K}} = 0.1..0.175$ with GOP = 8. No post-reconstruction filtering is employed. Best results in bold. . . 140
7.4 Comparison of reconstruction time for the various algorithms executed in our test environment. 143
7.5 Ablation study on the VAL-MCP algorithm. 144
7.6 VAL-VFI and VAL-IDA-VFI algorithms compared with SGS-OF, VAL-MCP and VAL-IDA-MCP with GOP = 8, $\delta_k = 0.7$ and $\delta_{\bar{K}} = 0.1$. Best results in bold, second best underlined. 145
7.7 VAL-VFI and VAL-IDA-VFI algorithms compared with SGS-OF and VAL-IDA-MCP with $\bar{\delta} = 0.175$. Best results in bold, second best underlined. 145
7.8 Comparing Shrink-{Bicubic|SRCNN|VDSR|IMDN}, L-DCT-ZZ, AL-DCT-DD, VAL-[IDA]-MCP and VSR-{Bicubic|VDSR|IMDN} with algorithms in the literature that exploit motion implicitly - 3DSRNet, VSRResNet and VSRResNetFeatGAN - and explicitly - VSRnet, VESPCN and SPMC-VSR on VidSet4. VAL and VSR algorithms have $GOP = 4$, and $\delta_K = \delta_{\bar{K}}$. Best results in bold. $\times 2$, $\times 3$ and $\times 4$ refer to 2, 3 and 4 times super resolution in both vertical and horizontal dimensions. The corresponding δ values are 0.25, 0.1111, 0.0625. . 148
7.9 Comparing our Shrink, VAL and VSR algorithms with those in the literature that exploit motion implicitly - 3DSRNet, VSRResNet and VSRResNetFeatGAN - and explicitly - VSRnet, VESPCN and SPMC-VSR on VidSet4. VAL and VSR algorithms have $GOP = 4$, $\delta_K = 0.25$ and $\delta_{\bar{K}} = 0.0625$. Best results in bold, second best underlined. $\times 3$ refers to 3 times super resolution in both vertical and horizontal dimensions. The corresponding δ value is 0.1111. . . . 150

8.1 The parameters used to derive the rate distortion points in the figures in this chapter with the top row pertaining to the worst performance and the bottom row equivalent to the best performance. 164

List of algorithms

Algorithm	Type	Chapter
L-DCT-ZZ-[IDA] AL-DCT-{THB|THI} AL-DCT-{BBV|DD}-[IDA]	Dimensionality Reduction	4
Shrink-{Bicubic|VDSR|IMDN}	Dimensionality Reduction	5
AL-DCT-{BBV|DD}-{JPEG|TAB}-[IDA]-[RD] L-DCT-ZZ-{JPEG|TAB}-[IDA]-[RD] Shrink-LLC-SR LLC := {DPCM-ENTROPY | DPCM-Golomb | DPCM-HUFF | DPCM-AC | JPEG2000 | LS-JPEG | PNG } SR := {Bicubic | VDSR | IMDN}	End-to-End	6
VAL-[IDA]-[{MCP|VFI}] VSR-{Bicubic|VDSR|IMDN}-[{MCP|VFI}]	Dimensionality Reduction	7
VAL-TP-[IDA]-[{MCP|VFI}] VLZ-TP-IDA-VFI	End-to-End	8

Table 1: List of main algorithms in each chapter. Components in square brackets are optional and only one acronym in curly brackets, separated by an OR | symbol, is to be included. For example, AL-DCT-{BBV|DD}-[IDA] refers to the four adaptive algorithms that may optionally be reconstructed using the IDA algorithm, namely, AL-DCT-BBV, AL-DCT-DD, AL-DCT-BBV-IDA and AL-DCT-DD-IDA.

List of acronyms

ABCS	Adaptive Block-based Compressive Sensing
ADC	Analogue to Digital Converter
AMP	Approximate Message Passing
ARPS	Adaptive Rood Pattern Search
BBV	Block Boundary Variation
BCF	Bicubic Convolutional Filter
BCS	Block-based CS
BM3D	Block-matching and 3D filtering
BM3DF	BM3D filtering
BPD	Best Pixel Discriminator
BPG	Better Portable Graphics
CIF	Common Intermediate Format
CMOS	Complimentary Metal Oxide Silicon
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CS	Compressive Sensing
DAIN	Depth-aware Video Frame Interpolation
D-AMP	Denoising AMP
DAMP-D	D-AMP Damped
DCT	Discrete Cosine Transform
DD	DCT Domain
DISCOS	Distributed Compressed Video Sensing
DISCOVER	DIStributed COding for Video sERvices
DMD	Digital Micromirror Device

DnCNN Denoising CNN

DNN Deep Neural Network

DPCM Differential Pulse Code Modulation

DS-CS Decoder-Side CS

DVC Distributed Video Coding

DWHT Discrete Walsh Hadamard Transform

ES-CS Encoder-Side CS

ES-FS Encoder-Side Full Sensing

FPPS Focal Plane Processing Sensors

FS Full Search

GOP Group of Pictures

GPU Graphics Processing Unit

HEVC High Efficiency Video Coding

IDA Iterative Denoising Algorithm

IDCT Inverse 2D DCT

IDN Information Distillation Network

IMDN Information Multi-Distillation Network

ISTA Iterative soft-thresholding algorithm

JP2 JPEG 2000

LC-LLC Low-Complexity Lossless Coding

LDAMP Learned D-AMP

MCP Motion Compensated Prediction

ME Motion Estimation

ME/MC Motion Estimation/Motion Compensation

ME/MCP Motion Estimation/Motion Compensation Prediction

MPC Multi Pixel Camera

MPCs Multi Pixel Cameras

MS-SSIM Multiscale SSIM

PDWZ Pixel Domain Wyner-Ziv

PSNR Peak Signal-to-Noise Ratio

PSS Power Step Search

QCIF Quarter CIF

RIP Restricted Isometry Property

RRS Reweighted Residual Sparsity

SDA Stacked Denoising Autoencoder

SGS Structural Group Sparsity

SGS-OF Structural Group Sparsity and Optical Flow estimation

SISR Single Image Super Resolution

SOTA State-of-the-Art

SPC Single Pixel Camera

SR Super Resolution

SRCNN Super Resolution CNN

SSIM Structural Similarity Index Measure

TC Transform Coefficient

TDWZ Transform Domain Wyner-Ziv

THB Threshold per Block

THF Threshold Filter

THI Threshold from the whole Image

THI2 Threshold over all Image Blocks

TV Total Variation

VCS Video Compressive Sensing

VDSR Very Deep Super Resolution

VFI Video Frame Interpolation

vMPC Virtual MPC

VSR Video Super Resolution

ZZF Zig-Zag Scan Filter

Notation

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$LCM(.)$	Least common multiple operator
$\min(.)$	Minimum function
$\max(.)$	Maximum function
$\|\mathbf{x}\|_0$	Number of nonzero elements; L_0 norm
$\|\mathbf{x}\|_p$	L_p norm
$\lfloor x \rfloor$	Floor operator, rounds x to the nearest integer less than or equal to x
$\lceil x \rceil$	Ceil operator, rounds x to the nearest integer greater or equal to x
$\mathbf{X}^+$	Positive matrix i.e. a matrix with elements that are greater or equal to zero
$\mathbf{I}$	Identity matrix with ones on the diagonal, zeroes elsewhere
$\mathbf{J}$	Unit matrix with all $M \times N$ elements equal to one
$\bar{x}$	The complement of x
$DCT(.)$	Discrete 2D cosine transform
$IDCT(.)$	Inverse 2D discrete cosine transform
$\odot$	Element-wise multiplication
$\|x\|$	Modulus, i.e. the absolute value, of x
$diag(.)$	Function returning elements on matrix diagonal
$\mathbf{1}$	Vector with all ones

Chapter 1

Introduction

Transmitting video from resource constrained wireless sensors is challenging because of the high bit rate associated with the video signal and the computational complexity of source coding algorithms required for its compression [1]. Classical Distributed Video Coding (DVC) techniques are based on the Slepian-Wolf distributed source coding theory [2] that allows a decoder to benefit from correlation at the source without explicit exploitation at the encoder, for example using Motion Estimation (ME) and Motion Compensated Prediction (MCP). DVC assumes implicitly that capturing video is power-efficient, and focuses on compressing video as efficiently as possible, so that it can be transmitted viably from the sensor. This means that computationally intensive operations, such as ME and MCP, have to be moved to the decoder where they can exploit the correlation between frames to increase reconstruction quality. This would lead to a rise in transmission bitrate, and is countered by transmitting some of the frames at a lower bit rate, normally resulting in lower visual quality at the decoder. However, it is possible to leverage the correlation between frames to restore video quality. The distributed source coding theories of Slepian and Wolf [2] and Wyner and Ziv [3], show that we can transmit correlated data without exploiting correlation at the encoder, at a lower rate than that required if we assume the data is uncorrelated, and still recover the full information at the decoder.

Compressive Sensing (CS) [4], [5] represents a departure from the normal source coding paradigm, of sampling an analogue signal at the Nyquist rate, digitizing the samples at the highest possible signal-to-quantization-noise-ratio and then using complex source coding algorithms to compress the data as much as possible. In CS, the capture and compress processes are combined to obviate the need for computationally intensive source coding algorithms. However, Goyal [6] cautions, using information theoretic arguments, that the compression factors achievable with CS alone, are lower than those that can be achieved by classical source coding paradigms. This means that some form of lower-complexity source coding is still required prior to transmission.

CS senses a multidimensional signal $\mathbf{x} \in \mathbb{R}^N$ by performing the inner product between all the dimensions of $\mathbf{x}$ and rows of a measurement matrix $\mathbf{\Phi}^{M \times N}$. The CS operation generates an M dimensional compressed signal $\boldsymbol{y} = \mathbf{\Phi}\boldsymbol{x}$, that is $\mathbf{y} \in \mathbb{R}^M$. This reduces the dimensionality of $\mathbf{x}$ from N to the dimensionality of $\mathbf{y}$, achieving a compression ratio, $\delta = M/N$. Note that a higher

compression ratio means more compressive measurements.

Applying CS to video frames requires the storage of the measurement matrix $\mathbf{\Phi}^{M\times N}$ at the encoder. Contemporary video CS techniques operate on a video signal with a Group of Pictures (GOP) structure. The leading key frame in the GOP is coded with a higher quality than the remaining non-key frames. The key frame is typically coded with a compression ratio δ in the range 0.4 to 0.7 in the literature. This means that even for a modest resolution, for example that of the classical Common Intermediate Format (CIF) video frame (352×288 pixels), the storage requirement of the measurement matrix $N_s = \delta \cdot N^2$ is of the order of gigabytes when representing each matrix coefficient by an integer (e.g., 16 bits) and is impractical for resource constrained sensors. Generating the matrix on the fly would be too power inefficient. The solution to this problem, proposed in the literature [7], [8], is to break the video frames into sub-image blocks of size $B \times B$ pixels, where B is typically 4, 8, 16 or 32. The same measurement matrix being used to compressively sense each block.

However, block-based CS of video frames comes with two problems. The first arises from the fact that the number of measurements M required by CS theory is proportional to the sparsity of the signal S, that is $M = kS\log(N/S)$ [9], where k is a constant with typical values in the range $\{2\ldots6\}$. The sparsity is the number of non-zero coefficients, in some domain Ψ that can represent the signal with the desired quality. If the video frame is processed as one block, there is one sparsity value for the frame. With block-based processing, each block has its own sparsity level, that is changing from frame to frame. In adaptive block-based image CS, the sparsity of each block is estimated prior to compressively sensing it [10]. The challenge is to estimate the sparsity with as low a complexity and overhead as possible.

The second problem caused by block-based coding, is that of blocking artefacts appearing in the reconstructed image, if the CS reconstruction is also block-by-block. This can be solved by applying deblocking filters or by sensing the image block-by-block but reconstructing it as a whole, using a full-image sensing matrix that is composed of the block sensing matrices on its diagonal, that is $\mathbf{\Phi_I} = diag(\mathbf{\Phi_B})$ [7].

In this dissertation, bold capital letters represent matrices, bold lower case letters represent vectors, and normal case letters, scalar values. Consider a multidimensional signal $\mathbf{x} \in \mathbb{R}^N$ with a sparse representation $\boldsymbol{f} \in \mathbb{R}^M$ in some domain Ψ, that is $\boldsymbol{f} = \mathbf{\Psi}\boldsymbol{x}$. Then the compressive samples of $\mathbf{y}$, produced by the measurement matrix $\mathbf{\Phi}$ are given by $\boldsymbol{y} = \mathbf{\Phi}\mathbf{\Psi}\boldsymbol{x} + \boldsymbol{n}$ where $\boldsymbol{n}$ is the measurement noise. The reconstruction of $\mathbf{x}$ requires the solution of this equation. However, this is an ill-posed problem because the sensing matrix $\boldsymbol{A} := \mathbf{\Phi}\mathbf{\Psi} \in \mathbb{R}^{M\times N}$ and $M << N$. Yet the equation does have a sparse solution because $\mathbf{x}$ is S-sparse, and $M = kS\log_2(N/S) > S$. A tractable solution can be pursued by first casting it as a convex linear program:

$$\min_{\mathbf{x}} \quad \|\mathbf{x}\|_1, \quad \mathbf{s.t.} \ \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 < \varepsilon, \tag{1.1}$$

where ε is a measure of the noise level, and solving it using state-of-the-art Linear Programming (LP) solvers. This is nontrivial and time consuming, and a significant number of reconstruction techniques have been proposed to accelerate the solution, such as matching pursuit

[11], Bayesian [12], Approximate Message Passing (AMP) [13] and Denoising AMP (D-AMP) [14].

Recently, Convolutional Neural Networks (CNNs) have been applied to solve equation (1.1) [15], [16], [17], [10]. A number of authors have also proposed CNNs that can reconstruct compressively sensed images in 10's to 100's of milliseconds, such as ReconNet [18], CS-Net [19], ISTA-Net and ISTA-Net+ [20], and SCSNet [21]. The reconstruction times of these CNNs allows CS video to be transmitted at tens of frames per second, but without exploiting temporal correlation between frames, generating large bit-rates.

Video Compressive Sensing (VCS) techniques have been proposed that exploit temporal correlation by multihypothesis prediction, motion compensated prediction or optical flow at the decoder such as ME/MC [22], RRS [23], and SGS-OF [24]. Recently VCS has been reconstructed using CNNs, for example VCSNet[25].

1.1 Aims and objectives

The main aim in this dissertation is to develop video compressive sensing techniques that maximise frame-rate and quality in dimensionality reduction applications, and end-to-end distributed video compressive coding systems that maximise energy efficiency, for example in autonomous wireless sensor networks.

Increasing the frame rate requires the use of block-based video compressive sensing and this has the added benefit of decreasing the measurement matrix size by orders of magnitude. Block-based CS (BCS) leads to the problem of variable sparsity in the blocks. Adaptive block-based CS techniques and deblocking filters will be investigated to solve this problem.

The use of deterministic measurement matrices with a fast inversion algorithm will be investigated. Quality improvements will be pursued through the use of fast CS reconstruction algorithms that leverage CNN-based algorithms implemented on Graphics Processing Units (GPUs).

Another objective is to adopt and adapt distributed video coding methods to distributed video CS, in order to exploit temporal correlation between frames, whilst implementing energy efficient encoders.

1.2 Contributions

1.2.1 Chapter 3

A framework is proposed for comparing energy-efficient compressive image and video sensors. Nine parameters are identified, two pertaining to quality, four to energy efficiency, one to frame-rate, one to signal to noise ratio and another to computational complexity. The framework is used to compare the end-to-end image and video transmission systems developed in chapters 6 and 8.

1.2.2 Chapter 4

The deterministic partial Discrete Cosine Transform (DCT) measurement matrix is employed in the non-adaptive, block-based L-DCT-ZZ algorithm that can be reconstructed with good quality in under 10ms for a 256×256 image. Two deblocking algorithms are developed that deblock the reconstructed images in the DCT domain.

Two novel, Adaptive Block-based Compressive Sensing (ABCS) algorithms are developed, that employ the deterministic, low-pass, 2D-DCT sensing matrix, namely, (i) adaptive linear 2D-DCT with block boundary variation estimation (AL-DCT-BBV), and (ii) adaptive linear 2D-DCT with DCT Domain estimation (AL-DCT-DD). These algorithms estimate block sparsity in the spatial and transform domains respectively, and vary the number of compressive measurements in each block. The algorithms can be reconstructed using the inverse 2D DCT transform in under 8 ms and 30 ms for 256×256 and 512×512 images respectively.

A further contribution is the development of the Iterative Denoising Algorithm (IDA), derived from the D-AMP algorithm [14]. IDA uses a CNN-based denoiser and can be used to post-process the output of the AL-DCT-BBV and AL-DCT-DD algorithms. IDA reconstruction achieves State-of-the-Art (SOTA) Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) results amongst ABCS results published in the literature. IDA reconstructs a 256×256 image in around 5.5 seconds.

1.2.3 Chapter 5

The Shrink algorithm is proposed that can implement the Matlab imresize bicubic interpolation and anti-aliasing function as a 2D convolution on Single Pixel Cameras (SPCs) or Multi Pixel Cameras (MPCs). When combined with a state-of-the-art CNN-based single image super resolution algorithm, Shrink-IMDN exceeds the PSNR performance of the best IDA-reconstructed ABCS algorithms in chapter 4 and CNN-based image CS algorithms in the literature, below a compression ratio of 0.15. Above this value, the IDA reconstructed AL-DCT-DD-IDA algorithm performs best.

1.2.4 Chapter 6

Two low-complexity algorithms, a JPEG adaptation and TAB, are proposed that can quantize and encode the ABCS algorithms developed in chapter 4. The JPEG-style algorithm renders the rate distortion performance of the ABCS algorithms equivalent to the classical JPEG algorithm but with significantly less compressively sensed Transform Coefficients (TCs). The TAB algorithm uses 32×32 blocks and achieves better performance than JPEG below 0.3 bits per pixel (bpp).

When IDA is used to reconstruct the JPEG encoded images, the rate-distortion performance exceeds that of JPEG by a considerable amount and matches the JPEG2000 PSNR between 0.4 and 0.6 bpp. The SSIM performance is substantially above JPEG2000 above 0.25 bpp.

A low-complexity algorithm is developed to encode the output of the Shrink algorithm based on spatial Differential Pulse Code Modulation (DPCM) followed by entropy coding algorithms. Alternatively, the subsampled Shrink images can be coded using standard compression algorithms such as JPEG2000 or PNG. When super resolved using state-of-the-art CNN-based algorithms, performance is better than JPEG below 0.2 bpp.

1.2.5 Chapter 7

Distributed video adaptive compressive sensing algorithms are developed that exploit temporal correlation at the decoder. A GOP structure with a leading key frame is adopted. Block MCP or Video Frame Interpolation (VFI) is used to predict non-key frames from the key frames. The predicted frames are also used in a DCT-domain DPCM operation with the reconstructed non-key frames to improve the quality. Finally, three versions of the non-key frames; 2D IDCT reconstructed, MCP or VFI predicted, and DPCM predicted, are processed by a best pixel discriminator function that selects best pixels for the output image. The VFI version of the Video AL-DCT-DD (VAL) algorithm, VAL-VFI, achieves state-of-the-art performance from a dimensionality reduction perspective. IDA reconstruction improves the performance of VAL-IDA-VFI above that of VAL-VFI.

When the Shrink algorithm replaces AL-DCT-DD to sub-sample key and non-key frames, the video sSuper Resolution (SR) algorithm exceeds the performance of state-of-the-art CNN-based video super resolution algorithms. However, the VAL-VFI and VAL-IDA-VFI algorithms have superior performance.

1.2.6 Chapter 8

The dimensionality reduction algorithms are modified to quantize and entropy code the transform coefficients. A modified version of the TAB algorithm developed in chapter 6 is used to encode the VAL coefficients. The best performing algorithm, VAL-TP-IDA-VFI approaches the performance of the well-known DISCOVER DVC algorithm [26], whilst being more energy efficient due to compressive sensing.

1.3 Publications

These contributions have led to the following publications:

- J. Zammit and I. J. Wassell, "Adaptive Block Compressive Sensing: Toward a Real-Time and Low-Complexity Implementation," in IEEE Access Journal, vol. 8, pp. 120999-121013, 2020, doi: 10.1109/ACCESS.2020.3006861.
- J. Zammit and I. J. Wassell, "Distributed Video Adaptive Block Compressive Sensing" pre-print, https://arxiv.org/a/zammit_j_1.html

1.4 Dissertation structure

Chapter 2 introduces the theory of compressive sensing and the AMP and D-AMP iterative reconstruction techniques that have been applied successfully to reconstruct compressively sensed images. CNN-based techniques that have been proposed to accelerate reconstruction are then reviewed, followed by a brief introduction to video compressive sensing. CS camera technology is identified that guarantees the feasibility of the algorithms developed in this dissertation. The chapter identifies deterministic sensing matrices, adaptive block-based image CS, distributed video coding techniques and super resolution as having potential to deliver the objectives of this dissertation.

A framework for comparing the performance and energy efficiency of compressive image and video sensors is presented in chapter 3. The implementation of convolutional operations on a multipixel camera is investigated and the number of analogue to digital conversions per compressively sensed image is calculated and integrated with the comparative framework.

Chapter 4 presents two novel ABCS algorithms using the partial 2D DCT transform as a measurement matrix and two techniques to deblock the 2D-IDCT reconstructed ABCS algorithms in real-time. The D-AMP algorithm is interpreted as a feedback system and modified to implement the IDA that can reliably reconstruct adaptively sensed images. The IDA algorithm is shown to deliver state of the art performance especially when combined with the CNN-based DnCNN [27] denoiser.

Chapter 5 shows how the Matlab bicubic interpolation and anti-aliasing function can be implemented as a convolution on compressive sensing cameras. This leads to the definition of the Shrink compressive, subsampling algorithm that produces subsampled images that can be reconstructed using two GPU-accelerated, super resolution CNNs. A comparison with the image CS techniques developed in chapter 4 and state-of-the-art algorithms shows the potential of the Shrink-based algorithms for video CS.

The image CS and SR algorithms developed in chapters 4 and 5 are quantized and low-complexity encoded in chapter 6. Two low-complexity encoding techniques are proposed for the image CS algorithms, and one for the image SR algorithm. The end-to-end techniques are compared and the super resolution algorithms are shown to produce competitive results and are shown to be worthy of further study.

Two distributed video CS algorithms and one distributed video SR algorithm are designed in chapter 7 and studied from a dimensionality reduction perspective. The distributed video CS algorithms exploit temporal correlation using DPCM at the encoder. The three algorithms exploit DPCM and motion compensated prediction or video frame interpolation at the decoder. The distributed video CS algorithms perform better than the distributed video SR counterpart and achieve state-of-the-art results, even in real-time reconstruction.

Chapter 8 shows how the CS measurements of the distributed video CS algorithms developed in chapter 7, can be quantized and encoded for end-to-end digital transmission. The techniques are compared with a classical distributed video coding algorithm.

Conclusions are drawn in chapter 9. Suggestions for further work are also presented.

Chapter 2

Preliminaries

Compressive Sensing (CS) is predicated on sensing a signal $\mathbf{f} \in \mathbb{R}^N$ using $M = \mathcal{O}(S \cdot log(N/S))$ non-adaptive measurements with a measurement dictionary $\mathbf{\Phi}$ that is incoherent with the basis $\mathbf{\Psi}$ in which the signal of interest is sparse [9]. The sparsity S is the number of non-zero components of signal $\mathbf{x}$ such that $\mathbf{f} = \mathbf{\Psi} \cdot \mathbf{x}$. The development of appropriate measurement dictionaries has been the subject of intense research with random, structurally random and deterministic dictionaries proven to be suitable for CS. CS theory is overviewed in section 2.1.

Image and video CS have proven to be very challenging because of the large volumes of data associated with these sources. Reconstruction is particularly challenging and a large number of techniques have been developed to accelerate reconstruction. CS reconstruction techniques are overviewed in section 2.2, with a focus on the Approximate Message Passing (AMP) and Denoising AMP (D-AMP) iterative techniques, as examples of fast reconstruction algorithms, in sections 2.2.1 and 2.2.2. Recent work on the use of Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs) to accelerate image CS reconstruction is then reviewed in section 2.2.3.

The development of CS theory led to the development of the Single Pixel Camera (SPC), initially as a proof of concept tool. However, sensing of sparse signals predates the developments in CS. Focal Plane Processing Sensors (FPPS) have been proposed, with some systems performing transforms and pixel operations adjacent to the pixel. It is possible to re-purpose these systems to capture samples based on the measurement matrices developed using CS techniques. Camera systems for image and video CS are overviewed in section 2.4. This chapter concludes by proposing the research directions for the work in the rest of this dissertation.

2.1 Compressive image sensing

In sensing applications, it is often the case that information needs to be extracted from a set of measured data points. Taking $\mathbf{f} \in \mathbb{R}^N$ as the signal of interest and $\mathbf{y} \in \mathbb{R}^M$ as the measured signal, the relationship between the two can be written as:

$$\mathbf{y} = \mathbf{\Phi}\mathbf{f}, \tag{2.1}$$

where $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ is the measurement matrix sampling signal $\mathbf{f}$ to give $\mathbf{y}$. If linear measurements of $\mathbf{f}$ are taken, then the reconstruction problem is reduced to solving a set of linear equations.

The classical Nyquist-Shannon sampling theorem sets the sampling rate to twice the highest frequency for $\mathbf{f}$ to be reconstructed. CS theory shows that it is possible to represent $\mathbf{f}$ with only $M = \mathcal{O}(S \cdot log(N/S))$ non-adaptive measurements [9].

In a CS framework, the signal of interest $\mathbf{f}$ has a discrete representation $\mathbf{x} \in \mathbb{R}^N$ in some transform domain with basis $\mathbf{\Psi} \in \mathbb{R}^{N \times N}$ such that $\mathbf{f} = \mathbf{\Psi x}$. Vector $\mathbf{x}$ is defined as sparse if it contains S non-zero elements such that $S \ll N$. The measured signal $\mathbf{y}$ is therefore given by:

$$\mathbf{y} = \mathbf{\Phi \Psi x}. \tag{2.2}$$

Grouping the measurement matrix and sparsifying basis matrix gives:

$$\mathbf{y} = \mathbf{Ax}, \tag{2.3}$$

where $\mathbf{A} = \mathbf{\Phi \Psi} \in \mathbb{R}^{M \times N}$ is defined as the sensing matrix. The goal in CS is thus to represent the original signal using $M < N$ samples, substantially below the Nyquist rate. As the locations of the S sparse non-zero values of $\mathbf{x}$ are not known in advance, reconstruction involves solving the undetermined set of equations represented by (2.3) [28].

Two main questions arise from equation (2.3): (i) how should the measurement and sensing matrices be chosen; and (ii) how does one solve for $\mathbf{x}$? As the solution of (2.3) involves locating the non-zero components in vector $\mathbf{x}$, the most natural approach is an L_0 minimization, which can be written as:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \qquad s.t.\ \mathbf{Ax} = \mathbf{y}. \tag{2.4}$$

As L_0 minimization is NP-hard and thus computationally intractable, a common approach is to rewrite the problem as an L_1 minimization:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \qquad s.t.\ \mathbf{Ax} = \mathbf{y}. \tag{2.5}$$

Solving equation (2.5) can now be achieved via linear programming, which is tractable [28], and is commonly referred to as basis pursuit [28]. Faster reconstruction methods have been proposed in the literature, such as matching pursuit [11], AMP [13], D-AMP [14] and neural networks [15].

In real-world applications, noise is always present, and rewriting (2.3) to take this into account gives:

$$\mathbf{y} = \mathbf{Ax} + \mathbf{n}, \tag{2.6}$$

where $\mathbf{n} \in \mathbb{R}^M$ represents Gaussian noise. The convex relation in (2.5) does not hold in this instance. However, it can be replaced by Basis Pursuit Denoising (BPDN) [28] giving:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \qquad s.t.\ \|\mathbf{Ax} - \mathbf{y}\|_2^2 < \varepsilon, \tag{2.7}$$

where the noise estimate ε depends on $\mathbf{n}$.

An important feature of the sensing matrix is that it should preserve the information pertinent to the sparse signal $\mathbf{x}$. There are several conditions that can determine whether a sensing matrix possesses this feature. Three well-known properties are (i) the Restricted Isometry Property (RIP), (ii) the null space property, and (iii) mutual coherence. The sensing matrix that satisfies any of these three properties possesses the required information preservation feature [28].

It has been shown that when $\mathbf{A}$ is a random Gaussian matrix, it satisfies the RIPproperty [28] with high probability. Gaussian matrices are often used as the measurement matrices in the image CS literature. When the measurement matrix $\mathbf{\Phi}$ is Gaussian, the resulting sensing matrix $\mathbf{A} = \mathbf{\Phi\Psi}$ is also Gaussian (because of the affine property of multivariate distributions) and benefits from satisfying the RIPwith high probability. Simpler matrices, such as random (scrambled) Hadamard matrices, are also RIPcompliant (studied as structured random sampling matrices in [28]) and have added implementation advantages, especially on Digital Micromirror Device (DMD) cameras [29]. However, optimal performance can be obtained using Gaussian or Bernoulli i.i.d matrices [4].

2.2 Reconstruction algorithms

Many compressive sensing reconstruction algorithms exist to recover sparse signals from compressive samples. Recently, Pilastri and Tavares presented a taxonomy of reconstruction algorithms in compressive sensing [30]. In this taxonomy, the algorithms are grouped into six clusters: convex relaxation, greedy, non-convex, iterative, Bregman iterative, and combinatorial.

Reconstruction algorithms are characterized by their inherent complexity and the minimum number of compressive samples from which they can reliably recover the sparse coefficients and hence reconstruct the signal.

In image and video coding, we encounter signals whose transform in some domain is only approximately sparse; hence, we need reconstruction algorithms that can recover the predominant S sparse components in the presence of "noise" consisting of a long tail of small, non-zero components. If the sorted coefficients x_i decay with a power law, such that:

$$|x_i| \leq ci^{-p}, \tag{2.8}$$

where c is a constant and $p \geq 1$, then, x_i is called p-compressible and can be approximated by a sparse signal (definition 2.2 in [28]). The transform coefficients of the images and video fit this model [31].

2.2.1 Approximate message passing

In 2009, Donoho et al. [13] proposed AMP as a faster alternative to LP as a reconstruction technique for CS. AMP adds a correction term to Iterative soft-thresholding algorithm (ISTA) [32] to vastly improve convergence.

In ISTA , the inverse noisy problem $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$ is solved for $\mathbf{x}$ iteratively. Let $\mathbf{A}$ be the $M \times N$ sensing matrix with $M < N$, $\mathbf{y}$ a vector with M measurements, and $\mathbf{x^t} \in \mathbb{R}^N$ the current estimate of the sparse solution. Then, starting from an initial guess $\mathbf{x^0} = 0$, $\mathbf{x^t}$ is computed by solving equations (2.9) and (2.10) iteratively:

$$\mathbf{x^{t+1}} = \eta_t(\mathbf{A}^*\mathbf{z^t} + \mathbf{x^t}), \tag{2.9}$$

$$\mathbf{z^t} = \mathbf{y} - \mathbf{A}\mathbf{x^t}, \tag{2.10}$$

where $\mathbf{z^t}$ is the current error, $\mathbf{A}^*$ is the transpose of $\mathbf{A}$, and η_t is a scalar threshold function that is applied component-wise. This simple iterative scheme can be applied to larger-scale applications than standard LP solvers because of the very low cost per iteration and low storage requirements [13].

It is shown that adding the Onsager correction term $1/\delta \mathbf{z^{t-1}} \langle \eta'_{t-1}(\mathbf{A}^*\mathbf{z^{t-1}} + \mathbf{x^{t-1}}) \rangle$ to (2.10), derived from belief propagation in graphical models, improves the convergence of the ISTA solution. The operator $\langle \mathbf{u} \rangle$ performs the component-wise mean of the vector $\mathbf{u}$, that is, $\langle \mathbf{u} \rangle \equiv \sum_{i=1}^{N} u_i/N$, η'_t is the derivative of η_t and $\delta = M/N$ (equivalent to C_R in this dissertation).

The AMP algorithm then finds $\mathbf{x}$ iteratively by solving (2.11) and (2.12) below:

$$\mathbf{x^{t+1}} = \eta_{\mathbf{t}}(\mathbf{A}^*\mathbf{z^t} + \mathbf{x^t}), \tag{2.11}$$

$$\mathbf{z^t} = \mathbf{y} - \mathbf{A}\mathbf{x^t} + \frac{1}{\delta}\mathbf{z^{t-1}} \langle \eta'_{t-1}(\mathbf{A}^*\mathbf{z^{t-1}} + \mathbf{x^{t-1}}) \rangle, \tag{2.12}$$

The authors in [13] further show that the AMP solution has the same theoretical sparsity-undersampling trade-off as the LP-based reconstruction.

2.2.2 Denoising AMP

Metzler et al. [14] extended the AMP framework by replacing the thresholding nonlinearity η_t with a denoiser D_σ, that can be applied to reduce the Gaussian noise in the estimate $\mathbf{x_0} + \sigma\mathbf{z}$, where $\mathbf{z} = \mathbf{N}(\mathbf{0}, \mathbf{1})$ is Gaussian with zero mean and unit variance. They do this to leverage the rich literature on image denoisers, which has led to sophisticated algorithms that can remove Gaussian noise with large variance σ^2 from images. Their D-AMP iterations are given by equations (2.13) to (2.15):

$$\mathbf{x^{t+1}} = D_{\hat{\sigma}^t}(\mathbf{x^t} + \mathbf{A}^*\mathbf{z^t}), \tag{2.13}$$

$$\mathbf{z^t} = \mathbf{y} - \mathbf{A}\mathbf{x^t} + \frac{1}{\delta}\mathbf{z^{t-1}}\frac{div[D_{\hat{\sigma}^{t-1}}(\mathbf{x^{t-1}} + \mathbf{A}^*\mathbf{z^{t-1}})]}{m}, \tag{2.14}$$

$$(\hat{\sigma}^t)^2 = \frac{\|\mathbf{z^t}\|_2^2}{m}, \tag{2.15}$$

where $div[D(\mathbf{x})] = \sum_{i=1}^{n} \frac{\partial D(\mathbf{x})}{\partial x_i}$, x_i is the i^{th} element of $\mathbf{x}$, n is the size of $\mathbf{x}$, and m is the number of measurements.

2.2.3 Accelerating reconstruction using deep neural networks

Reconstruction using D-AMP takes tens of seconds to reconstruct images, which is still to long for real-time video CS reconstruction, where it is required to reconstruct images in tens of milliseconds. However, accelerating CS reconstruction is possible using the parallel processing capabilities of Graphics Processing Units (GPUs). The path to achieve this is through the use of GPUs to accelerate DNNs, of which CNNs are the most pertinent for image processing tasks. This section reviews how CNNs have evolved to reconstruct CS images in tens of milliseconds.

In 2006, Hinton and Salakhutdinov [33] reignited interest in neural networks by showing that deep neural networks can be built by first pretraining two layers at a time, then unrolling – or concatenating – them into deep networks, and finally fine-tuning the weights using backpropagation [34]. In their seminal work, they designed one autoencoder with $2000 \times 1000 \times 500 \times 30$ nodes in the encoder, by first pretraining a stack of Restricted Boltzmann Machines (RBMs), each with one layer of feature detectors. They then fine-tuned the cascaded encoder, and a transposed decoder, using backpropagation. The designed autoencoder outperformed Principal Component Analysis (PCA) on a 25×25 pixel image test set.

Gregor and LeCun first studied learning fast approximations of sparse coding [35]. They considered the solution of the LASSO program using the ISTA algorithm, time-unfolded it, and truncated it to a fixed number of iterations, to provide an approximate solution. They then proposed using backpropagation to learn the Learned ISTA parameters. They showed that Learned ISTA is about twenty times faster than Fast ISTA.

Burger et al. first studied the use of neural networks to denoise images [36]. Xie et al. were the first to employ deep neural network to denoise and inpaint corrupted images [37]. These two papers showed that neural networks can match state-of-the-art denoisers for quality and beat them on speed.

In 2015, Angshul Majumdar proposed reconstructing dynamic MRI in real-time using a Stacked Denoising Autoencoder (SDA) [38]. The author sates *"The SDA is not related to CS. But CS is a non-linear inversion operator; we hope that SDA will be a good tool to learn the non-linear inversion operation and yield good quality images"*. An SDA is then trained on 100×100 pixel images, and achieves poorer performance than non-real-time, offline MRI video reconstruction techniques, but close to Kalman filter and differential CS real-time, online, techniques. The author reports that reconstruction speeds using Matlab on a 3.1 GHz Core i7

CPU, were 0.33 seconds for Kalman filter, 0.13 seconds for differential CS and 0.0274 and 0.03 seconds for 2-layer and 3-layer SDA s.

Motivated by the work of Hinton on autoencoders, Mousavi, Patel and Baraniuk conducted, what they claim to be, the first study to propose a deep learning approach to recover compressively sensed images [15]. The proposed stacked denoising autoencoder either reconstructs the compressive samples from a conventional linear measurement encoder (3-layer Linear SDA), or includes an additional first layer which will perform non-linear measurements (4-layer nonliner SDA). This additional layer, is trained together with the three decoding layers, and can therefore adapt to the class of training signals. The feed-forward, block-based, deep neural nets are trained on the ILSVRC 2014 image database [39], converted to 256×256 greyscale images, and normalized to [0,1]. The study focused on a subrate M/N of 0.25. The authors compared the trained DNNs with D-AMP , Tiled D-AMP and Total Variation (TV). Whereas the conventional reconstruction schemes took tens of seconds to reconstruct an image, the trained DNNs reconstructed the image in 2 milliseconds, with better quality than tiled D-AMP . We note that this reconstruction time is very suitable for real-time video play-back.

Kulkarni et al. proposed ReconNet, a novel CNN to non-iteratively reconstruct compressively sensed images [18]. The authors also include results from a single pixel camera they built, and a visual tracking algorithm. Kulkarni et al. also claim theirs to be the first peer-reviewed work on non-iterative recovery of CS images using neural networks, although they acknowledge the prior non-peer-reviewed work of Mousavi et al. Reconnect is a six-layer CNN, an operates on 33×33 pixel image blocks. The CNN is trained on 21,760 patches obtained from a database of 91 training images. ReconNet was compared with four algorithms: D-AMP ; min TV; SDA ; and NLR-CS, at subrates of 0.25, 0.10, 0.04 and 0.01. ReconNet returned the best PSNR at the 0.1, 0.04 and 0.01 sub-rates, and reconstructed images in approximately 20 ms whereas SDA took less than 3 ms. Post-processing the reconstructed images using BM3D was found to increase PSNR, at the expense of time.

Iliadis et al. acknowledged the prior work on SDAs [15] and CNNs [18] and proposed the first DNNs to reconstruct 3D temporal video CS [40]. Temporal video CS encodes a sequence of frames into one frame, by multiplying each frame with a binary mask and adding the resulting coded frames together. CS reconstruction techniques then recover the frames with high spatial and temporal resolution [41]. The authors consider $8 \times 8 \times 16$ spatio-temporal blocks, with 16 stacked 8×8 blocks, from 16 video frames. The measurement matrix is of the same size, and repeats in both vertical and horizontal directions. The fully connected DNN thus has an $8 \times 8 = 64$ input layer, k hidden layers, and one $8 \times 8 \times 16 = 1024$ output layer. Training was conducted using 10 million $8 \times 8 \times 16$ video blocks from 400 high definition YouTube videos. A seven layer DNN achieves state of the art performance but requires 10 s to reconstruct 16 frames.

Adler et al. were also pioneers in this area, claiming the first fully-connected DNN approach to reconstructing block-based compressively sensed images [42]. The authors were inspired by the work of Burger [36] who pioneered the use of DNNs in image denoising. A three-layer DNN

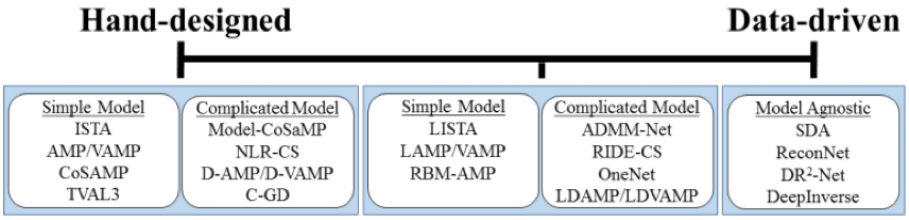


Figure 2.1: The spectrum of signal recovery algorithms, from [16]

was found to perform best, with a fully connected first layer performing blocked-based sensing, and two reconstruction layers followed by rectified linear units. The best block size was found to be 16×16 pixels, whereas the two reconstruction layers had $8 \times B^2$ fully-connected nodes. At a sensing rate of 0.25, Matlab reconstruction took 0.8 s, whereas the compared reconstruction schemes, based around Fowler et al.'s BCS schemes, varied from 6.39 to 207.32 seconds. In contrast, full image min TV took 1675.09 seconds. The fully connected DNN also performed 1 dB to 2 dB better than all the other compared schemes at sub-rates between 0.1 and 0.3.

Borgerding and Schniter unfolded the AMP algorithm and trained a deep neural network to solve sparse linear inverse problems [43]. Learned AMP is shown to be an improvement on Gregor and LeCun's Learned ISTA DNN, for the same reason that AMP is better than ISTA. The authors also unfolded the VAMP algorithm, and the Learned VAMP (LVAMP) network was found to be an improvement over Learned AMP.

Inspired by the work of Borgerding and Schniter [43], Metzler, Mousavi and Baraniuk unroll Denoising-based Iterative Thresholding (D-IT), D-AMP and Denoising-based VAMP (DVAMP) to produce CNN-based Learned D-IT (LDIT), Learned D-AMP (LDAMP) and Learned DVAMP (LDVAMP) [16]. The authors opine that by unrolling algorithms with a strong theoretical framework, and implement them in otherwise 'unprincipled' deep neural networks, the resulting learnt DNNs will become 'principled-CNNs'. The paper also presents a categorization of signal recovery algorithms which is reproduced from their figure 1 in figure 2.1.

The authors found that LDAMP and LDVAMP delivered equal performance, which was superior to LDIT. Hence, we only outline LDAMP here. LDAMP was implemented as a 10-layer unfolded D-AMP with a Denoising CNN (DnCNN) denoiser [27] which is itself a 20 layer CNN [16]. The 382,464, overlapping, 50×50 pixel training image blocks were pulled from the Berkley BSD-500 dataset [44]. LDAMP was found experimentally to surpass the state-of-the-art BM3D AMP and NLR-CS algorithms, and is more than 60 times faster. At a sampling rate of 0.10, LDAMP reconstructs an image in 0.4 s, BM3D-AMP in 4.8 s and NLR-CS in 85.91 s.

Mousavi and Baraniuk also experimented with non-block-based, full-image, CS reconstruction using a deep neural network called DeepInverse [45]. The input signal proxy is obtained by multiplying the measurement vector with the transpose of the measurement matrix. The output signal is then reconstructed using three convolutional layers. The PSNR performance for DeepInverse was better than TV minimization and P-AMP at all tested sub-rates (0.01, 0.1 and 0.2). D-AMP PSNR was significantly better at 0.1 and 0.2, however DeepInverse was significantly better at 0.01. Reconstruction time with DeepInverse was a constant 10ms, whereas the other algorithms required more than one second to reconstruct a 64×64 pixel image.

Yao et al. recently proposed a novel DNN, called deep residual reconstruction network – DR2-Net, based on a fully-connected linear mapping unit, followed by four residual learning blocks [46]. The block size adopted was 33×33 pixels and the residual training blocks each consists of three convolutional layers: the first using an 11×11 kernel with 64 feature maps, the second with 1×1 kernels and 32 feature maps, and the third generates 1 feature map with a 7×7 kernel. The training was based on 86,656, 33×33 pixel patches extracted from the same 91 images used by [18]. DR2-Net was compared with TVAL3, NLR-CS, D-AMP , SDA and ReconNet, at sampling rates of 0.25, 0.1, 0.04 and 0.01, with and without a post-processing BM3D filter. DR2-Net was found to be significantly better than all other techniques at sampling rates of 0.1, 0.04 and 0.01, on eleven 256×256, monochrome test images. Execution time was 56.5 ms to reconstruct one 256×256 monochrome image.

Compressed sensing algorithms are developed in this thesis, based on deterministic CS dictionaries, that can be reconstructed in real-time with and without using GPUs and that perform better than DNN -based techniques surveyed in this section. The literature also inspired the development of the GPU-accelerated, Iterative Denoising Algorithm (IDA) that produces state-of-the-art results when reconstructing images produced by the compressed sensing algorithms, though not in real-time.

2.3 Video compressive sensing

Initial work on video encoding and transmission from resource constrained sensors, concentrated on moving resource intensive operations, for example motion estimation and compensation, from the encoder to the decoder. The work was motivated by the side information coding theorems of Slepian and Wolf [2] and Wyner and Ziv [3] and led to a significant number of DVC algorithms, for example [47], [48], [49].

Building on the work of Gan et al [7], Chen, Trammel and Fowler developed algorithms for block-based compressed sensing and reconstruction of images and video including ME/MC and MH [50].

The theory of Distributed Compressed Sensing (DCS) [12] inspired Do et al. to propose the novel Distributed Compressed Video Sensing (DISCOS) framework [51]. The authors argued that the inherent analog processing nature of CS made DISCOS suitable in situations where analog to digital compression is expensive. The proposed applications included low-power video cameras for visual sensor networks.

More recently, Zhao et al. proposed reconstructing Video Compressive Sensing (VCS) using a Reweighted Residual Sparsity (RRS) technique [23].

Zhang et al. described a high-performance distributed compressive video sensing algorithm that generates accurate side information $\mathbf{w}$ using the HEVC [52] motion estimation algorithm and jointly minimize the ℓ_1-norm of the sparse signal $\mathbf{x}$ and the difference signal $(\mathbf{x} - \mathbf{w})$ [53]. Chen et al. proposed VCS reconstruction based on Structural Group Sparsity (SGS) and a coarse-to-fine, Optical flow (OF) based successive approximation estimation method [24].

Their Structural Group Sparsity and Optical Flow estimation (SGS-OF) algorithm was shown, by simulation, to achieve state-of-the-art performance when compared to ME/MC, MH and RRS.

Video compressive sensing algorithms usually sense the video as a sequence of Group of Pictures (GOP) with each GOP consisting of one key frame sensed at high quality and a number of non-key frames captured with higher compression but lower quality. The key and non-key frames are "compressed" by the compressive sensing operation performed in the optical domain, in order to reduce the computational complexity at the encoder. An underlying assumption is that it is inefficient, or too expensive, to capture all the pixels using one sensor per pixel, convert the samples to digital using an Analogue to Digital Converter (ADC) and then compress the digitized pixels using some image compression algorithm, for example JPEG [54], JPEG 2000 [55] or JPEG-XS [56].

JPEG-XS is a recent image coding standard, targeting near-lossless coding of images, having a lightweight design that can be implemented efficiently on CPUs, GPUs and Field Programmable Gate Arrays (FPGAs). It introduces another alternative to implement energy efficient video transmission, alongside distributed video coding, when capturing video frames using a conventional camera; a Complimentary Metal Oxide Silicon (CMOS) camera, for example.

2.4 Image and video CS cameras

The implementation of image and video CS theory requires sensors that sense images compressively, that is with less measurements than pixels. The single pixel camera [57] presented in 2.4.2 was the first device designed to exploit CS theory. Previously available CMOS devices are also capable of compressing an image by processing it on the pixel array itself as shown below.

The CS literature usually assumes that there is no non-linearity introduced in the camera system. The work in this thesis follows this trend. CS cameras do not capture pixels directly, but measurements from which the values of pixels have to be reconstructed off-camera. If it is necessary to introduce a non-linearity in the image/video transmission chain, this can be inserted in the camera itself or during reconstruction at the decoder. In the former case, the non-linearity can be integrated with the analogue to digital function shown in figure 2.2.

2.4.1 Focal plane processing sensors

FPPS [58] combine an active CMOS pixel with processing in the same area and are suitable for compressive sensing. Bandyopadhyay et al. [59] presented an example of a CMOS device capable of performing programmable matrix operations on 16×16 pixel blocks. They presented data showing their CMOS block matrix transform imager architecture performing convolutions and transforms, including the JPEG block Discrete Cosine Transform (DCT) transform. Their architecture processes images with a power consumption of only 80 μW per frame, demonstrating the very low power consumption of devices that perform mathematical operations in

the analogue domain.

Robucci et al. [60] were amongst the first to apply a computational focal-plane image sensor, capable of performing 2-D separable transforms, to compressively sense images. Their 256×256 CMOS sensor can perform 16×16 separable transforms and is demonstrated to be sufficient to perform non-adaptive compressive sensing measurements.

2.4.2 Single pixel camera

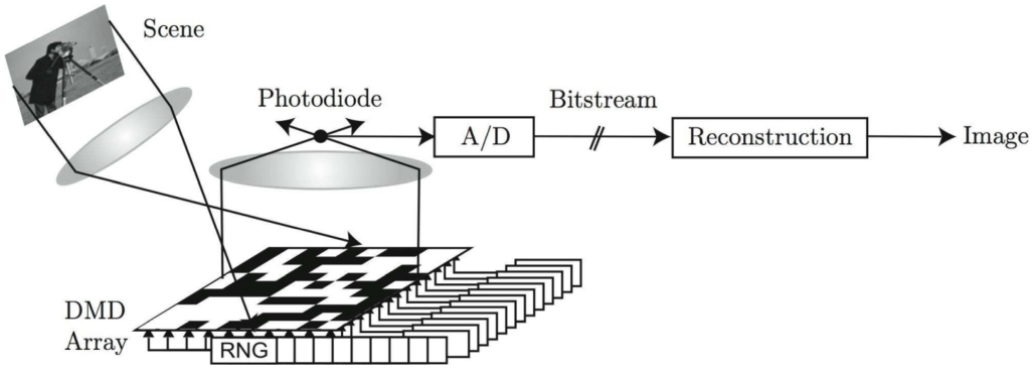


Figure 2.2: Single Pixel Camera [57]

The single pixel camera was proposed by Takhar et al. [57] in 2006 to exploit the new field of compressive sensing. The SPC shown in figure 2.2 consists of a digital micro-mirror device onto which a scene is focused by an objective lens. The light pixels from the scene are modulated by the positions of the DMD mirrors towards or away from a relay lens that focuses the light on a single photodiode. The positions of the mirror form a measurement basis vector and the light output by the photodiode for one particular pattern is the inner product of the image pixel array with the basis vector. The ADC converts the photodiode signal to a digital value. A collection of measurements, less than the number of pixels in the image, can be reconstructed into an image using CS reconstruction techniques.

2.4.3 CS camera taxonomy

Recently, Baraniuk et al. [61] identified three classes of CS video imaging architectures, namely spatial multiplexing cameras, temporal multiplexing cameras and spectral and angular multiplexing cameras.

1. Spatial multiplexing cameras boost the resolution of a low resolution sensor to the size of the compressive sensor. These include single SPC as well as multipixel detectors or Multi Pixel Cameras (MPCs). The focus in this thesis is on spatial multiplexing cameras.

2. Temporal multiplexing cameras modulate the light falling on a low-rate, full spatial resolution sensor and integrate the modulated light over a number of frames. The one captured frame is then super resolved into the same number of frames as the light modulations during the integration period. Thus temporal multiplexing cameras super resolve the temporal resolution of a camera that is limited by the analogue to digital conversion rate. Coded

Aperture Compressive Temporal Imaging (CACTI) is an example of temporal multiplexing cameras using a translating mask for high speed imaging [61].

3. Spectral and angular multiplexing cameras increase the resolution in the spectral or angular domain.

The authors observe that these camera systems aim to improve spatial, temporal or spectral resolution above the Nyquist sampling limit.

2.4.4 Multiple pixel cameras

Spatial multiplexing multipixel cameras use a DMD array to focus light onto a low resolution 2D or 1D array of photodetectors, rather than just one sensor in the SPC. The light from an image is focused on a DMD array and the DMDs pixellate the image. DMD motions modulate the light from the pixels and a particular DMD orientation pattern forms a Hadamard product with the pixellated image. Light from rectangular or square regions, or blocks, of the DMD array falls on one of the sensors and is integrated into one signal formed by the inner product of the image pixels and the mirror pattern in each square region. Each sensor can be equipped with its own ADC, or one ADC may be shared by the whole low-resolution array. The measurement rate for a spatial multiplexing camera with F photodetectors sharing one ADC with conversion rate R_{ADC} is given by [61]:

$$min\{F \times R_{DMD}, R_{ADC}\}, \tag{2.16}$$

where R_{DMD} is the rate at which the DMD patterns change. Under balanced conditions, $F_{min} \times R_{DMD} = R_{ADC}$ and $F_{min} = R_{ADC}/R_{DMD}$. The authors in [61] give an example with R_{DMD} = 10 kHz, R_{ADC} = 10 MHz so that $F_{min} = 1,000$ pixels and the spatial multiplexing camera can sustain 10 million measurements per second and remark that an SPC would only be able to collect 10,000 measurements per second with one photodetector. Therefore, spatial multiplexing cameras are essential for video compressive sensing as otherwise, an SPC will not be able to collect enough measurements per second. For example, the LiSens camera [62] has been proposed for video compressive sensing using visible light.

The DLP9500 is a 1920×1080 pixel DMD from Texas Instruments [63]. The device can support 1 bit binary patterns at 23,148 Hz and 8 bit binary patterns at 2,893 Hz.

2.4.5 Applications

It is thus clear that MPCs and FPPS devices exist that can compressively sense video in real-time. CMOS FPPS devices in particular, have very low power consumption that make them eminently suitable for autonomous video sensors in wireless sensor networks, for example. MPCs can be used in super resolution applications at wavelengths that are not suitable for implementation using CMOS technology and where alternative technology is very expensive at high resolutions [62].

2.4.6 Super resolution

The basic application of an MPC is in the super resolution of low resolution imaging sensor arrays. Together with the fast developments in CNN-based image and video super-resolution techniques, this has inspired the work on the implementation of super resolution techniques on the DMD array itself, or on focal plane processing sensors, in this dissertation.

2.5 Chapter summary

Distributed video coding is the classical technique for transmitting video from resource constrained devices. Authors have proposed extending DVC with compressive sensing. Both DVC and the CS extensions acquire and transmit video frames in a group of picture structure and postpone computationally intensive motion estimation to the decoder. Each GOP usually has one leading key frame that is compressed as efficiently as possible using the latest compression standards that are invariably block-based. Non-key frames are usually compressed using block-based 2D DCT transforms. This framework is adopted in the work presented in this dissertation and requires a focus on the energy efficient compressive sensing and transmission of video frames.

The classical formulation of CS requires a measurement matrix that is incoherent with the basis in which the image or video frame is sparse. The measurements are non-adaptive as long as the sparsity is known and enough measurements are sensed to allow successful reconstruction with high probability. However, authors have (1) studied measurement matrices that are random, deterministic or structurally random for compressively sensing images and videos, (2) proposed adaptive techniques to cater for reconstruction problems (refer to section 4.2), for example blocking, that occur with block-based CS. Using deterministic measurements, such as partial DCT matrices, allows the images to be reconstructed in real-time on commodity hardware. It is therefore proposed to study the use of a linear 2D DCT measurement matrix, selecting low-pass transform coefficients scanned in JPEG zigzag order.

Real-time processing of images and videos require block-based MPCs to sense the images with an adequate frame rate. However, this may lead to blocking artefacts as the sub-image blocks contain differing amounts of sparsity. It is therefore required to study deblocking filters and adaptive measurement techniques to combat blocking artefacts, in conjunction with full-image CS-reconstruction.

Several authors have studied feedback techniques to estimate the number of measurement blocks at the decoder from partial measurements and then feed them back to the encoder [64] [65] [66]. The work in this dissertation avoids this feedback loop as it requires a continuously powered receiver at the resource constrained sensor. For the same reason, the distributed video coding techniques developed in this dissertation are without the feedback associated with the classical Stanford architecture [67].

To decode images and video transmitted from a CS sensor in real-time requires fast CS reconstruction techniques. Currently, the fastest techniques are iterative denoising based tech-

niques, that leverage denoisers implemented on GPUs to accelerate them. However, the current reconstruction techniques e.g. D-AMP [14] were not designed with deterministic matrices in mind. It is therefore required to study how to reliably extend iterative denoising to partial 2D-DCT matrices.

GPUs can be leveraged to accelerate part or all of the decoding process [16]. Rather than train end-to-end deep neural networks and try to explain their behaviour, the approach in this dissertation is to use GPU-accelerated modules, such as CNN-accelerated denoisers, and hand-craft decoders that can still reconstruct images and video in real-time.

Finally, the literature review has identified the large, and very-active, body of work in the area of image and video super resolution. The work in this dissertation will study whether the down-sampling process can be implemented energy-efficiently on a CS image sensor, and will be studied to determine whether this lightweight technique can compete with CS techniques.

Chapter 3

Image and video transmission system analysis framework

3.1 Introduction

This chapter investigates theoretical energy efficiency considerations determined by the sensor structures. Generic results are generated that are applicable to as broad a range of different compressive sensors as possible, rather than focusing on one particular implementation. The algorithms developed in this dissertation can be implemented on all classes of compressive image sensors introduced in this chapter. Therefore, the analysis framework presented here, will be used in chapters 6 and 8 to compare end-to-end image and video transmission systems.

The literature review identified distributed video coding combined with Compressive Sensing (CS) as an appropriate strategy for transmitting video power-efficiently. Distributed Video Codings (DVCs) depends on the efficient transmission of key and non-key frames in a Group of Pictures (GOP) as individual frames, migrating computationally intensive operations, such as motion estimation and compensation, to the decoder.

Following the literature review, three competing techniques have been proposed for real-time transmission of images from low-power sensors: CS supported by CNN-based reconstruction; Adaptive Block-based Compressive Sensings (ABCSs) using deterministic matrices; and Bicubic Convolutional Filter (BCF) down-sampling followed by Super Resolution (SR) using Convolutional Neural Networks (CNNs).

In this chapter, a framework is proposed for comparing the energy efficiency of these three techniques when implemented on SPC, Multi Pixel Camera (MPC) and Focal Plane Processing Sensors (FPPS). It is also possible to study compressively sensing images on SPCs using a subimage block structure, a scheme that is referred to as a Virtual MPC (vMPC) in this dissertation. The framework proposed consists of nine parameters, covering image quality, energy efficiency, frame capture time and complexity.

In conventional block-based CS schemes, one MPC block size allows the implementation of all compression ratios required, simply by collecting more or less measurements of size $B \times B$. This is not the case with sub-sampling (and subsequent super resolution) where a MPC with

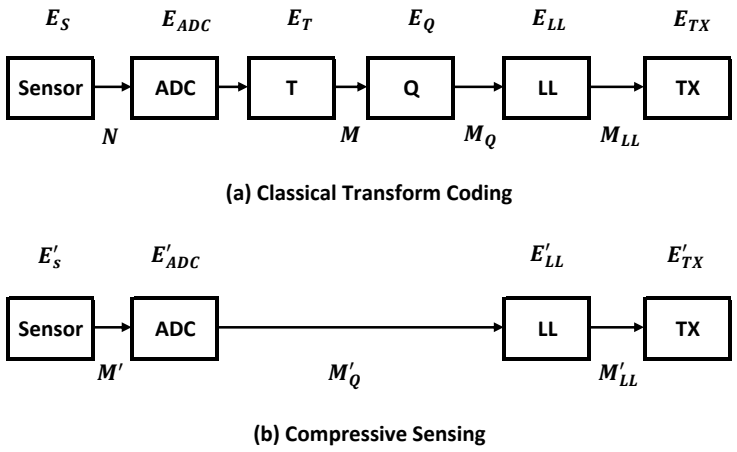


Figure 3.1: Classical and compressive sensing image transmission system models.

blocks of size $B \times B$ is required to implement BCF kernels of various sizes corresponding to the compression required. Sections 3.2.7 and 3.2.8 therefore show how a convolutional kernel of size $K \times K$ can be implemented efficiently on an MPC of size $B \times B$ where $K \neq B$ and derive equations for the number of Analogue to Digital Converter (ADC) conversions and conversion cycles required. The theoretical results derived are verified by simulation in section 3.2.9.

The symbols and equations introduced in this section are summarized in tables 3.2 and 3.3.

3.2 Framework for comparing low-energy sensors

3.2.1 Transmission system models

The image transmission system models pertinent to the work in this paper are the classical transform compression model in figure 3.1(a) and the compressive sensing model in figure 3.1(b). In the classical model, the analogue image sensor captures N analogue pixels and transforms them into a digital representation using an ADC. The number of pixels N is determined by the Nyquist criterion to avoid aliasing. The 2D pixel array is then 2D transform coded into N transform coefficients to compact the energy into a few of the lower frequency components, such that quantization, represented by block Q, will result in a sparse representation with M_Q non-zero quantized components where $M_Q \ll N$. These non-zero components are further compressed using a lossless compression scheme represented by block LL. The resulting compressed bitstream is characterised by a rate R_b measured in bits per pixel (bpp) and transmitted through transmission block TX at R_t bits per second (bps).

In CS, the analogue sensor captures $M^{'} \ll N$ analogue measurements at well below the Nyquist rate, where $M^{'} = k \cdot S.\log(S/N)$, k is a constant and S is the sparsity of the transformed image. Each measurement is an inner product between the image matrix and a measurement matrix.

The promise of CS is that the compressive sensor computes $M^{'} \ll N$ analogue measurements such that the transmission chain in figure 3.1(b) consumes less energy than the classical transmission chain in figure 3.1(a). The total energy consumed in a classical image transmission

system E_{TOT} is given by:

$$E_{TOT} = E_S + E_{ADC} + E_T + E_Q + E_{LL} + E_{TX}, \tag{3.1}$$

where E_S is the sensing energy using classical sensors, for example Charge Coupled Device (CCD) or Complimentary Metal Oxide Silicon (CMOS) sensors, E_{ADC} is the energy consumed in analogue to digital conversion, E_T is the energy consumed to transform the image (e.g. using a 2D Discrete Cosine Transform), E_Q is the energy required to quantize the transform coefficients, E_{LL} is the energy consumed in lossless coding of quantized transform coefficients, for example using {run, value} coding followed by Huffman coding, and E_{TX} is the energy required to transmit the compressed bit stream.

The total energy E'_{TOT} consumed in a compressive system, on the other hand, is given by:

$$E'_{TOT} = E'_S + E'_{ADC} + E'_{LL} + E'_{TX}, \tag{3.2}$$

where E'_S is the energy consumed by the compressive sensor (e.g. Single Pixel Camera (SPC)), E'_{ADC} is the energy consumed to convert the M' analogue measurements to digital, E'_{LL} is the energy consumed in further compressing the quantized measurements losslessly, and the E'_{TX} is the energy required to transmit the resulting compressed image. In the classical image transmission model quantization is required at the sensor output and following the transform block. In compressive sensing, quantization can happen solely in the ADC.

The classical image transmission model in figure 3.1(a) can also be used to represent a super resolution image transmission scheme. In this case, the transform block T, is replaced by a downsampling filter that can also be implemented as a 2D convolutional filter.

In a compressive sensing system the time T'_F required to capture and transmit a frame is given by:

$$T'_F = T'_M + T'_{ADC} + T'_{LL} + T'_{TX}, \tag{3.3}$$

where T'_M is the time required to modulate the mirrors, T'_{ADC} is the time required to perform ADC conversion on the whole image, T'_{LL} is the time required to encode the quantized samples with low-complexity and T'_{TX} is the time required to transmit the image or frame. T'_F represents the latency, whereas the rate at which frames can be captured by the pipeline shown in figure 3.1b F'_R is given by:

$$F'_R = \frac{1}{min(T'_M, T'_{ADC}, T'_{LL}, T'_{TX})}. \tag{3.4}$$

3.2.2 Framework parameters

To allow the comparison of low-energy image sensors using SPC, MPC or FPPS, a framework is proposed that measures and compares the following parameters:

1. The energy E_M to modulate the mirrors or mask;

2. The energy E_{ADC} to perform ADC conversion of the analogue measurements;

3. The energy E_{LL} to encode the CS measurements with low complexity;
4. The energy E_{TX} to transmit the encoded measurements;
5. The time T_F to capture and transmit one image or video frame;
6. The complexity of the sensor;
7. The signal to noise ratio of the compressive image sensor.
8. The Peak Signal-to-Noise Ratio (PSNR) at a given compression ratio;
9. The Structural Similarity Index Measure (SSIM) or Multiscale SSIM (MS-SSIM) at a given compression ratio;

These parameters are expanded upon in the sections below.

3.2.3 Sensors

Three different types of sensors are of interest, that can be used to perform both compressive sensing as well as sub-sampling followed by super resolution. The three sensors are:

1. An SPC which has one mirror per pixel, i.e. N mirrors and one light sensor for the whole image;
2. An MPC (or FPPS) which has one mirror (or mask) per pixel, that is N mirrors (or masks), but which are organized as n_B, $B \times B$ pixel blocks, and have n_B light sensors, one per block;
3. An MPC simulated on an SPC - a Virtual MPC (vMPC).

3.2.4 SPC characteristics for CS and SR

When an SPC is used to compressively sense an $H \times W$ pixel image, $H \times W$ mirrors focus light from the image onto one sensor. M measurements are collected independently and used to reconstruct $N = H \times W$ pixels. The compression ratio $C_R = M \,/\, N$. Each measurement is an inner product between the N pixels and a row of an $M \times N$ measurement matrix, which means that $N_M = M \times N$ mirror modulations are required to sense the whole image. The measurement matrix requires $N_{MEM} = M \times N$ storage locations to store the matrix coefficients.

To implement sub-sampling on an SPC using a BCF kernel of size $K \times K$, each sample requires the modulation of at most $N_K = K \times K$ mirrors, although some of the kernel coefficients may be quantized to zero. Hence the total number of mirror modulations is $N_M = M \times N_K = M \cdot K^2$. In SR, there is only one kernel to store for every subsampling rate and this requires $N_{MEM} = K^2$ storage locations.

When collecting CS measurements on an SPC, the measurement matrix $\mathbf{\Phi}$ contains both positive and negative elements, in general. However, the SPC cannot process negative values

directly. However, if one of the measurements collects the sum of all pixels σ, and the measurement matrix elements are made positive by adding the modulus of its minimum component μ, than the SPC can still collect the measurements as follows:

$$\mathbf{y} = \mathbf{\Phi x} = (\mathbf{\Phi} + \mu \mathbf{J})\mathbf{x} - \mu \mathbf{J x} = (\mathbf{\Phi} + \mu \mathbf{J})\mathbf{x} - \mu\sigma\mathbf{1} = \mathbf{\Phi}^{+}\mathbf{x} - \mu\sigma\mathbf{1}, \tag{3.5}$$

where $\mathbf{J}$ is an $M \times N$ unit matrix, $\mathbf{1}$ is an $M \times 1$ vector with all ones and $\mathbf{\Phi}^{+}$ contains only elements greater or equal to zero. Hence the number of measurements and ADC conversions is M. If each ADC conversion takes a time Δ_{ADC}, then the total conversion time for the M measurements is $M \cdot \Delta_{ADC}$.

In the case of subsampling implemented as a 2D convolution, each measurement is composed of two sub-measurements that are added together, one of which computes the inner product between the pixels and the positive kernel coefficients and the other the inner product between the pixels and the negative kernel coefficients. Hence, the number of ADC conversions is $2 \cdot M$. The total conversion time for the $2 \cdot M$ measurements is $2 \cdot M \cdot \Delta_{ADC}$.

3.2.5 MPC characteristics for CS schemes

In an MPC, the image is partitioned into non-overlapping blocks such that the number of pixels in the block $N_B = B^2$. It is assumed that the height and width of the image are exactly divisible by B. The number of blocks in an image is given by $n_B = N \,/\, B^2$. If the desired compression ratio is C_R, the number of measurements per block M_B is given by $M_B = \lfloor C_R.B^2 \rfloor$. The total number of mirror modulations to capture one image is now substantially less than in the case of an SPC and is given $N_M = n_B \cdot M_B \cdot B^2$. The sensing matrix is shared by all the blocks and requires $N_{MEM} = M_B \cdot B^2$ memory locations.

The CS schemes capture M measurements distributed over all the blocks. Therefore, the total number of ADC conversions $N_{ADC} = M_B \cdot n_B = M$, the same as in the case of the SPC. However there are only $N_C = M_B$ ADC conversion cycles since the measurements in the blocks happen in parallel. This is the case for both adaptive and non-adaptive CS schemes.

The conversion time for the three CS schemes is given by $M \cdot T_{ADC} \,/\, n_B$ which is the conversion time for the SPC case divided by the number of blocks n_B, because the measurements are captured in parallel by the n_B blocks.

3.2.6 vMPC characteristics for CS and super resolution

It is also possible to use an SPC to capture block-based image measurements. We refer to this use of the SPC as a virtual MPC. Each measurement only modulates the mirrors in a sub-image block rather than the whole image, and would therefore result in less energy expended to move the mirrors, and a lower acquisition time because of the requirement to only move a small subset of the mirrors. The measurements in each sub-image block would not be captured in parallel though, and hence the acquisition time in a vMPC is less than in an SPC but greater than in an

actual MPC. Therefore N_{MEM}, N_M and N_{ADC} are the same for the an MPC but the number of ADC conversion cycles N_C is the same as for an SPC, that is $N_C = M_B \cdot n_B = M$.

The implementation of subsampling on an SPC modulates $K \times K$ mirrors at a time, and hence the normal implementation of the BCF uses the SPC as a vMPC. The storage requirements for the kernel are $N_{MEM} = K^2$ locations. The total number of mirror modulations is $N_M = M.K^2$.

The number of ADC conversions considering a kernel with both positive and negative coefficients is given by $N^{\pm}_{ADC} = 2.M$. Although the measurements are not accelerated, the kernel can always be processed in one ADC conversion cycle so that the number of ADC conversion cycles $N^{\pm}_C = 2.M$.

3.2.7 MPC characteristics for super resolution

In the literature, the best super resolution results [68] are obtained by first downsampling the image using the Matlab *imresize* function which can be implemented as a BCF as shown in section 5.3.1. The implementation of the filter on the MPC requires convolving the pixels with a kernel. This is non-trivial because the filter kernel is of size $K \times K$ whereas the MPC block size is $B \times B$ with $K \neq B$ in general.

In this case, the total number of mirror modulations remains the same $M \cdot N_K$ since the same number of samples must be captured, although distributed over the MPC subimage blocks. The number of ADC conversions and the conversion time for convolutions with kernels of size $K \times K$ and stride S_F, will now be derived when computed on an MPC with the convolutions at the edge of the blocks overflowing into adjacent blocks, for example as shown in figure 3.2.

3.2.8 Number of ADC conversion cycles for BCF on an MPC

Consider an MPC with blocks of side B. One such block is shown in figure 3.2. If the subsampling factor is S_F, the number of blocks of size $S_F \times S_F$ that fit into the MPC block is given by $(B \,/\, S_F)^2$. This is also equal to the number of samples that are collected from the block. B is chosen such that it is exactly divisible by S_F. If it is necessary for the MPC to cater for several S_F, then the block size must be chosen to be a multiple of all S_F. This ensures that the sampling patterns are the same in all the blocks in the MPC.

The subsampling operation replaces each $S_F \times S_F$ block by one sample whose value is calculated by placing a convolutional kernel centred on the $S_F \times S_F$ block and performing one convolution with the pixels it covers. The kernels of interest are of size $K \times K$ where $K = 4.S_F$ if S_F is even and $K = (4.S_F - 1)$ if S_F is odd. This means that for samples at the edges of the MPC blocks, the kernel will extend into adjacent blocks (e.g. blocks marked N_2 and N_4 in figure 3.2). This happens for two rows and two columns of $S_F \times S_F$ blocks adjacent to other MPC blocks. In these cases, the sub-samples cannot all be captured in parallel because when a convolutional operation requires pixels from adjacent blocks, it blocks it from being used by other kernels. However, a block can independently perform a convolution operation if it does

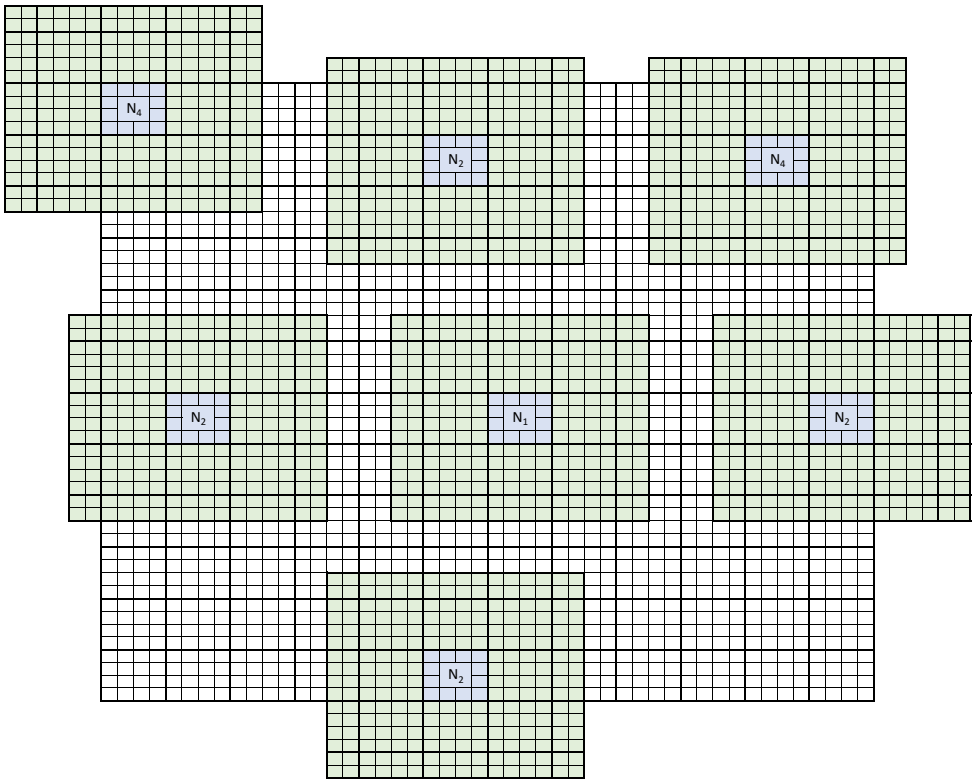


Figure 3.2: BCF on an MPC: examples when the convolutional kernel (green) falls within one block N_1, two blocks N_2 and four blocks N_4. In this example $B = 48 \times 48$, $S_F = 4$ and the kernel is $4.S_F \times 4.S_F = 16 \times 16$. The centre $S_F \times S_F$ kernel coefficients are shown in gray. All the kernels have the kernel coefficients.

not overflow into adjacent blocks (e.g. N_1). At other instances, one subsampling convolution operation requires the use of two vertically or horizontally adjacent blocks (e.g. N_2), or four blocks for samples at the corners of the MPC block (e.g. N_4).

Number of ADC conversions

Let the size of a convolutional filter kernel be $K \times K$ and the stride between samples be S_F. The convolutional filter kernels we are interested in, collect samples centred on $S_F \times S_F$ blocks and are of size $K \times K$ with $K = 4.S_F$ for S_F even and $K = (4.S_F - 1)$ for S_F odd. This means that in a block of side B subdivided into blocks of side S_F:

1. 16, corner $S_F \times S_F$ blocks will extend into three adjacent blocks and require four ADC conversions, in four different blocks, to return one sample;

2. 2-column and 2-row deep edge $S_F \times S_F$ blocks will extend into one adjacent block and require two ADC conversions, in two different blocks, to return one sample;

3. the remaining inner $S_F \times S_F$ blocks will return one sample by a convolutional operation contained solely in one block.

Let

$$n_S = B \,/\, S_F, \tag{3.6}$$

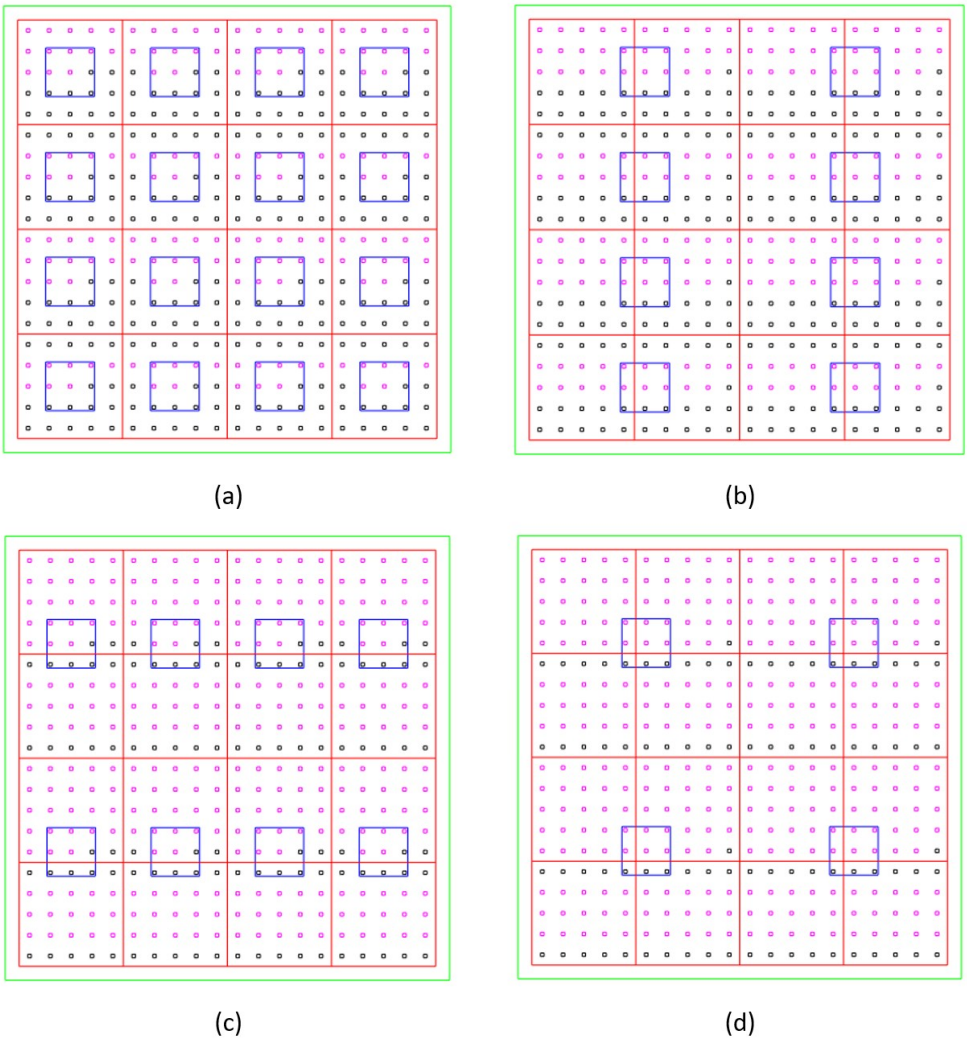


Figure 3.3: Simulation of BCF on an MPC showing kernel cycles with convolutions contained in (a) one MPC block (b) and (c) two blocks, and (d) four blocks.

then, the number of ADC conversions required to capture samples for the above three cases, denoted by N_4, N_2, and N_1, where the suffix represents the number of ADC conversions per sample, are given by:

$$N_4 = 16 \times 4 = 64, \tag{3.7}$$

$$N_2 = 8(n_s - 4) \times 2 = 16(n_s - 4), \tag{3.8}$$

$$N_1 = (n_S - 4)^2. \tag{3.9}$$

The number of ADC conversions per block is then given by:

$$N_{ADC}^{B} = N_1 + N_2 + N_4 = (n_S^2 + 8n_S + 16). \tag{3.10}$$

The number of ADC conversions for the whole image N_{ADC} is given by the number of ADC

conversions per block multiplied by the number of blocks:

$$N_{ADC} = N_{ADC}^{B} n_B = (n_S^2 + 8n_S + 16)n_B. \tag{3.11}$$

Number of conversion cycles

N_1 measurements fit exactly in 1 block as shown in figure 3.3(a). These then require one parallel measurement cycle to collect. There are $(n_s - 4)^2$ such blocks.

N_2 measurements extend over two adjacent blocks, either horizontally as shown in figure 3.3(b) or vertically as shown in figure 3.3(c). N_2 measurements require two parallel measurement cycles to collect, and there are $8.(n_s - 4)$ such blocks.

Finally, N_4 measurements extend over four adjacent blocks as shown in figure 3.3(d). Therefore these corner measurements require 4 parallel measurement cycles to collect. There are 16 such measurements per block.

In total,

$$N_C = (n_s - 4)^2 + 2(8)(n_s - 4) + 16(4) = n_s^2 + 8n_s + 16, \tag{3.12}$$

cycles are required to collect all the measurements.

Practical considerations

When computing convolutions with positive and negative values on an SPC or MPC, it is required to split the kernel into positive and negative values, i.e. $K = K_+ - K_-$, perform two separate convolutions with the two positive sub-kernels K_+ and K_-, and then subtract the convolution with K_- from the convolution with with K_+. Therefore two ADC conversions are required per convolution and equation 3.12 becomes:

$$N_{ADC}^{\pm} = 2N_{ADC}^{B} n_B = 2(n_S^2 + 8n_S + 16)n_B. \tag{3.13}$$

Similarly, the number of measurement cycles in equation 3.12 doubles to

$$N_C^{\pm} = 2N_C = 2(n_S^2 + 8n_S + 16). \tag{3.14}$$

This doubling of ADC conversions only happens when the kernel is used once to sample an image. In the case of compressive sensing, one of the measurements would be the average of the block and this allows subimage block measurements with the same number of ADC conversions as shown in section 3.2.4.

3.2.9 Verification by simulation

A Matlab simulation was coded to verify that the number of ADC conversions calculated by equations 3.11 and 3.12 are correct. Figure 3.3 depicts outputs from the simulation, showing a BCF kernel distributed over (a) one, (b) and (c) two, and (d) four blocks. The theoretical results from equation 3.12 and the simulation are compared in table 3.1, when $H = 480$ and $W = 480$.

S_F	B	n_s	N_C	
			Theory	Simulated
2	30	15	361	361
2	60	30	1156	1156
3	30	10	196	196
3	60	20	576	576
4	60	15	361	361
5	30	6	100	100
5	60	12	256	256
6	30	5	81	81
6	60	10	196	196

Table 3.1: Comparing N_C calculated using equation 3.12 and measured by simulation, when $H = 480$ and $W = 480$.

The simulation concerns the bicubic convolutional filters that are used to implement the *imresize* function described in section 5.3.1. The size of the kernel K when $S_F = \{2,3,4,5,6\}$ is $K = \{8,13,16,21,24\}$. When the kernel coefficients are quantized to ease implementation on the MPC, the quantized kernel may shrink in size to $K_Q \times K_Q$ because some of the coefficients may be quantized to zero. Refer to figure 5.4 for an example when the BCF kernels defined in section 5.3.1 are quantized to 8 bits (including the sign bit). In this case, $K_Q = \{8,11,14,19,22\}$.

If the side of the square block $B = LCM(2,3,5,6) = 30$, the subsampling pattern is the same in all blocks at the required subsampling rates except 4. Smaller block sizes allow more samples to be captured in parallel and hence require less capture cycles, which translates into a lower latency. Therefore, by using blocks of size 30×30 to build an MPC, regular patterns are achieved for $S_F = \{2,3,5,6\}$ with lowest latency. Combining 2×2 adjacent 30×30 MPC blocks allows $S_F = 4$ to be captured with a regular pattern but more latency. If $S_F = 4$ is required with lower latency, it is possible to use a block size of 32×32, which can cater for $S_F = \{2,4,8\}$ with a regular, low-latency pattern.

No differences were found with the unquantized kernel sizes $K = \{8,13,16,21,24\}$ and the quantized kernel sizes $K_Q = \{8,11,14,19,22\}$ indicating that the slightly smaller kernel sizes did not reduce the number of capture cycles required.

3.3 Evaluating CS and SR sensors using the framework

In this section, we highlight how the framework presented in this chapter can be used to compare different image and video sensor technologies in this dissertation.

Estimating E_M and E_{ADC}

The results in sections 3.2.4 to 3.2.8 can be used to estimate framework parameters E_M, E_{ADC} and T_F. The three sensors are compared in table 3.2, in terms of the number of mirror modulations, the number of ADC conversions and the number of conversion cycles for one image or intraframe in a video sequence, when using CS with deterministic Discrete Cosine Transform (DCT) measurement masks or downsampling using a bicubic convolutional filter for SR. The symbols used in this table are defined in table 3.3.

	CS	SR
SPC	$N_{MEM} = H.W.M$	$N_{MEM} = K^2$
	$N_M = H.W.M$	$N_M = M.K^2$
	$N_{ADC} = M$	$N^{\pm}_{ADC} = 2.M$
	$N_C = M$	$N^{\pm}_C = 2.M$
vMPC on SPC	$N_{MEM} = M_B.B^2$	$N_{MEM} = K^2$
	$N_M = M_B.n_B.B^2$	$N_M = M.K^2$
	$N_{ADC} = M_B.n_B$	$N^{\pm}_{ADC} = 2.M$
	$N_C = M_B.n_B$	$N^{\pm}_C = 2.M$
MPC	$N_{MEM} = M_B.B^2$	$N_{MEM} = K^2$
	$N_M = M_B.n_B.B^2$	$N_M = M_B.n_B.K^2$
	$N_{ADC} = M_B.n_B$	$N^{\pm}_{ADC} = 2.n_B.(n_S^2 + 8.n_S + 16)$
	$N_C = M_B$	$N^{\pm}_C = 2.(n_S^2 + 8n_S + 16)$

Table 3.2: Calculating the framework parameters for CS and SR sensors.

The parameters in table 3.2 are used to calculate the energy required to sense an image E_{CAP} as follows:

$$E_{CAP} = E_M + E_{ADC} = N_M \varepsilon_M + N_{ADC} \varepsilon_{ADC}, \tag{3.15}$$

where N_M is the number of total mirror motions to sense an image, ε_M is the energy required to move one mirror, N_{ADC} is the number of ADC conversions and ε_{ADC} is the energy to convert one compressive analogue measurement to digital.

Estimating T_F

The time to capture and transmit one image or frame is equal to the time T_{CAP} to capture and digitize the compressive image or frame, plus the time required to further compress it with low complexity prior to transmission. T_{CAP} on an SPC, MPC (or FPPS) and vMPC can be calculated as follows:

$$T_{CAP,SPC} = T_M + T_{ADC} = MN\Delta_M + M\Delta_{ADC}, \tag{3.16}$$

$$T_{CAP,MPC} = T_{MB} + T_{ADC} = N_{MB}\Delta_M + M_B\Delta_{ADC}, \tag{3.17}$$

B	MPC and SPC block dimension	E_{CAP}	Energy to capture one frame
bpp	Bits per pixel	E_M	Energy to move mirrors
bps	Bits per second	E_{ADC}	Energy to perform ADC
C_F	Compression Factor = N/M	ε_M	Energy to perform one mirror movement
C_R	Compression Ratio = $1/S_F^2 = M/N$	ε_{ADC}	Energy to perform one ADC
H	Height of Image	N_M	Number of mirror modulations in an image
K	CBF size $= 4.S_F + (1-(-1)^{S_F})/2$	N_{MB}	Number of Mirror modulations in a block
n_S	B/S_F	N_C	Number of ADC cycles
n_B	Number of blocks in image = $H.W/B^2$	$N_C^{\pm}$	Number of ADC cycles (K with $\pm$ values)
N	Number of Pixels	N_{ADC}	Number of ADC conversions
M	Number of Measurements	$N_{ADC}^{\pm}$	Number of ADC conversions (K with $\pm$ values)
M_B	Number of Samples in Block = $\lceil B^2/S_F^2 \rceil$	T_{CAP}	Time to capture one frame
PSNR	Peak Signal to Noise Ratio	T_M	Time to move mirrors in an image
SSIM	Structure Similarity Index	T_{MB}	Time to move mirrors in a block
S_F	Subsampling Factor in one dimension	T_{ADC}	Time to perform all ADC conversions
W	Width of Image	Δ_M	Time to move one mirror
N_{MEM}	Storage size for measurement matrix	Δ_{ADC}	Time to perform one ADC conversion

Table 3.3: Framework symbols used in this chapter and table 3.2.

$$T_{CAP,vMPC} = T_{MB}n_B + T_{ADC} = N_{MB}\Delta_M n_B + M\Delta_{ADC}, \tag{3.18}$$

where the various terms are defined in tables 3.2 and 3.3.

In a video system, T_{CAP} must be less than or equal to the execution time of the succeeding compression and transmission blocks. If T_{CAP} is the largest execution time in the transmission chain, then the frame rate is less than or equal to $1/T_{CAP}$, and the execution of the other blocks will be experienced as an added delay and not a reduction in frame rate.

Estimating E_{LL} and complexity

Parameter E_{LL} depends on the complexity of the algorithm used to encode quantized measurements prior to transmission and the energy characteristics of the sensor's computational platform. In this dissertation, as with other published literature, we compare computational complexity by comparing the execution time of the simulated algorithms. This requires the algorithms to be executed on the same platform, and to be equally optimized.

Estimating E_{TX}

The energy, and power, required to transmit images or video frames, is directly proportional to the bit rate that results from dimensionality reduction achieved on the compressive sensor and the ensuing compression achieved by the low-complexity encoding scheme prior to transmission. A comparison between systems is possible by assuming the same energy per bit expended and comparing the output bit-rate.

Estimating the Signal to Noise Ratio

The signal to noise ratio of the sensor increases linearly with the number of mirrors that focus light on the pixel sensor. In an SPC, all N mirrors focus light on the pixel sensor, whereas in an MPC or vMPC this reduces to B^2 mirrors. Therefore the signal to noise ratio in an SPC is n_B times higher than in an MPC or vMPC.

Estimating PSNR and SSIM

The remainder of this dissertation focuses on measuring the PSNR and SSIM [69] or MS-SSIM [70] performance of image and video compressive sensing schemes and relating them to the other framework parameters in our discussions.

3.4 Chapter summary

This chapter proposes an end-to-end transmission systems model and a framework for comparing low-energy sensor systems. The framework consists of the following parameters: PSNR, SSIM, sensor Signal to Noise Ratio (SNR), complexity of the sensor, $E_{CAP} = E_M + E_{ADC}$, E_{LL}, E_{TX}, and T_F.

The number of ADC conversion cycles required to capture convolutional samples on an MPC was derived and used to estimate E_{CAP} and T_{CAP} for these operations. Subsequent chapters on end-to-end image and video transmission systems will use this framework to compare the proposed methods.

Chapter 4

Adaptive block compressive sensing of images

4.1 Introduction

The work in this dissertation focuses on low-complexity, real-time image and video compressive sensing, using Single Pixel Camera (SPC), Multi Pixel Camera (MPC) or Focal Plane Processing Sensors (FPPS) as explained in chapter 3. Low complexity is required at the encoder to enable long-duration autonomous operation, for example in wireless video sensor networks.

In his tutorial-style paper on imaging via compressive sampling, Romberg proposed that 1000 linear low-pass 2D Discrete Cosine Transform (DCT) coefficients were required to capture the coarse details of the 256×256 Cameraman image and then switch to 20,000 additional pseudorandom measurements to acquire the detail [71]. In this chapter, we investigate using a sensing matrix comprised solely of linear low-pass 2D DCT coefficients in JPEG-style zigzag order. If this matrix provides good results, then reconstruction in real-time is achieved simply using an inverse 2D DCT.

As reported in the literature review in chapter 2, real-time compressive sensing of images and video benefits from using block-based techniques [7][72]. However, the sparsity in blocks varies greatly, and hence so does reconstruction quality, both when the reconstruction is block-by-block but even when the Compressive Sensing (CS) reconstruction is over the whole image. In order to improve quality, as reported in the related work section, authors have studied Adaptive Block-based Compressive Sensing (ABCS) of images, where the measurement rate is varied block-by-block depending on the sparsity. Therefore, we propose and study two adaptive algorithms namely, (i) Adaptive Linear 2D-DCT with Block Boundary Variation estimation (AL-DCT-BBV) and (ii) adaptive linear 2D-DCT with DCT Domain estimation (AL-DCT-DD). These algorithms estimate block sparsity in the 2D spatial and 2D-DCT transform domains respectively and vary the number of compressive measurements in each block. The number of measurements in each block is increased when the sparsity is low (where low sparsity means that the block has more non-zero transform coefficients in the 2D-DCT domain).

Work related to ABCS is reported in section 4.2. Non-adaptive and non-compressive linear

2D DCT-based algorithms are introduced in section 4.3. The proposed ABCS schemes are studied in section 4.4. Section 4.5 develops an Iterative Denoising Algorithm (IDA) suitable for our linear 2D DCT sensing matrices. In section 4.6 we investigate our ABCS algorithms empirically and show that the IDA algorithm achieves state-of-the-art reconstruction quality. Conclusions are drawn in section 4.7.

Naming Convention

In this work we refer succinctly to a number of algorithms by using the following convention. Components in square brackets are optional and only one acronym in curly brackets separated by an or | symbol is to be included. For example, AL-DCT-{BBV|DD}-[IDA] refers to the four adaptive algorithms that may optionally be reconstructed using the IDA algorithm, namely, AL-DCT-BBV, AL-DCT-DD, AL-DCT-BBV-IDA and AL-DCT-DD-IDA.

Algorithms

The following image compressed sensing algorithms are developed in this chapter.

- L-DCT-ZZ is the block-based, non-adaptive image CS algorithm using a deterministic dictionary composed of the low-pass 2D-DCT basis functions in zig-zag scan order. The image blocks are reconstructed using the inverse 2D DCT.
- AL-DCT-THB is a block-based, adaptive, non-compressive 2D-DCT image coding algorithm. The same number of 2D-DCT transform coefficients with the largest energy are retained in each block.
- AL-DCT-THI is a block-based adaptive, non-compressive 2D-DCT image algorithm. The M coefficients with the largest energy are retained from amongst all the blocks.
- The Block Boundary Variation (BBV) algorithm uses the sum of the variation of pixel pairs along the boundary of a block, to estimate the number of transform coefficients to be measured in each block.
- The DCT Domain (DD) algorithm uses partial low-pass, 2D-DCT measurements in zig-zag order, to estimate the total number of transform coefficients to measure in each of the image blocks.
- AL-DCT-BBV is an adaptive block-based image CS algorithm that uses the BBV algorithm to adapt the number of low-pass 2D-DCT measurements in each block whilst retaining the same number of transform coefficients over the whole image.
- AL-DCT-DD is an adaptive block-based image CS algorithm that uses the DD algorithm to adapt the number of low-pass 2D-DCT measurements in each block whilst retaining the same number of transform coefficients over the whole image.

- The Zig-Zag Scan Filter (ZZF) is a real-time algorithm that deblocks images produced by L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD. ZZF takes a full-image 2D DCT of the reconstructed image and retains M low-pass transform coefficients in zig-zag scan order.
- The Threshold Filter (THF) is a real-time algorithm that deblocks images produced by L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD. THF takes a full-image 2D DCT of the reconstructed image and retains the M transform coefficients with the largest energy.
- The IDA algorithm is an iterative denoising algorithm, derived from Denoising AMP (D-AMP), that can reconstruct images encoded using the L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD algorithms without blocking artefacts and that can use GPU-based full-image denoisers such as Denoising CNN (DnCNN).

4.2 Related work

Image Block-based CS (BCS) was first proposed by Lu Gan [7]. In his paper, the author proposed the development of spatially adaptive reconstruction algorithms as further work. Other seminal work on BCS of images and video was conducted by Fowler, Mun, Chen and Tramel in [8], [22], [50], and [73] respectively.

The non-adaptive, linear 2D-DCT CS technique, referred to here as L-DCT-ZZ, was inspired by the work of Romberg [71], who observed that CS of images is difficult and that a number of low-pass, linear 2D-DCT transform measurements, acquired in JPEG zigzag order [54], aid reconstruction significantly. In [71], Romberg compared low-pass, linear DCT acquisition in JPEG zigzag order with a scheme that combines some linear DCT measurements with noiselet measurements, concluded that the latter achieves better results, and consequently proposed that image CS was eminently suitable for further study.

Recently, Yuan and Haimi-Cohen showed that L-DCT-ZZ can be reconstructed using CS reconstruction techniques, and can achieve better SSIM results when compared with JPEG, especially at high compression ratios [74]. However they reported that PSNR was reduced by their technique.

A significant body of literature has concentrated on the development of ABCS algorithms for images. The techniques can be classified into Encoder-Side (ES) and Decoder-Side (DS-) techniques, depending on where the number of the adaptive measurements is estimated.

In some encoder-side papers, it is assumed that all the image pixels are already available at the encoder prior to compressive sensing and are required to estimate the number of measurements per block. We refer to these as Encoder-Side Full Sensing (ES-FS) techniques. These are not feasible from a CS perspective. We include these ES-FS schemes as benchmarks. Due to their inherent advantage in assuming that all pixels are already available before an image is adaptively sensed, they should achieve optimal results. A number of authors have published ES-FS techniques for example [75] [76] [77] [78] [79] [80] [81] and [82].

The AL-DCT-BBV and AL-DCT-DD algorithms proposed in this chapter do not assume that a full set of pixels or transform measurements is available and are referred to as Encoder-Side CS (ES-CS) techniques. Other authors have proposed ES-CS schemes, such as [65] [83] [84] and [85].

Averbuch et al. proposed a departure from using random dictionaries that are incoherent with the sparse basis as employed in classical CS [83]. Their dictionaries, instead, are nonrandom linear measurements from the sparsifying basis, for example, wavelets. They showed that these nonrandom measurements can still be captured on a Digital Micromirror Device (DMD). This scheme is not block-based, however.

An alternative to ES-CS is to first transmit a reduced resolution image using normal BCS techniques, reconstruct it at the decoder, perform the analysis there, and feed back the subrate information to the encoder. We refer to these Decoder-Side CS (DS-CS) techniques. [64] [65] and [66] proposed DS-CS ABCS techniques.

Several encoder-side papers ignore the measurements required to classify blocks prior to allocating the subrate to each block. Others include these measurements as part of the overall compressive measurements. Our AL-DCT-BBV and AL-DCT-DD algorithms include these estimation measurements as part of the block measurement allocation.

Donoho, Maleki and Montanari developed message passing algorithms for compressed sensing with reduced reconstruction time [13]. This led to the work on D-AMP by Metzler [14] [16], using Block-matching and 3D filtering (BM3D) and DnCNN denoisers. The BM3D and DnCNN denoisers have also been used in recent work on image CS reconstruction using the plug-and-play method [86] [87] [88].

In this chapter we propose and study adaptive block-based compressive sensing algorithms that estimate the total number of measurements per block from an initial set of measurements that form part of the overall set of measurements acquired per block. The number of measurements per block needs to be communicated to the decoder together with the measurements themselves. No feedback is required from the decoder.

The ABCS algorithms use deterministic 2D-DCT measurement matrices that can be reconstructed, block-by-block in real-time using the inverse 2D-DCT. We also study how to reconstruct the image using the full-image (not block-by-block) D-AMP technique developed by Metzler et al. using both BM3D and DnCNN denoisers [14]. It is proposed that the real-time reconstruction is used for monitoring purposes, whereas the non real-time, higher-quality video is used for inspection purposes, if required.

4.3 Image BCS using deterministic 2D-DCT sensing matrices

In this chapter, we focus on the use of deterministic, linear 2D transform sensing matrices such as the 2D-DCT. The Discrete Walsh Hadamard Transform (DWHT) can also be used and has a simpler implementation because the measurement matrix consists of {+1/-1} elements. However, the simulated DWHT PSNR and SSIM performance is inferior to 2D-DCT and is not

reported here.

4.3.1 L-DCT-ZZ

To accomplish compressive sensing, capitalizing on the energy compaction property of the 2D DCT transform, M low-pass transform coefficients are sensed and sorted in increasing order of sequency using the zigzag scanning procedure present in JPEG [54]. We refer to this as the Linear 2D-DCT with Zigzag scan (L-DCT-ZZ) algorithm that is described in algorithm 4.1. This technique has been proposed and studied by a number of authors, for example [71], [89]. The block-based algorithm generalises to sensing the whole image if the block is the same size as the image. L-DCT-ZZ is at the basis of our novel AL-DCT-BBV and AL-DCT-DD algorithms.

Algorithm 4.1 L-DCT-ZZ

Input: Image $\mathscr{I}$, compression ratio $C_R = M/N$, block size B, image width W, image height H.
Output: Image $\mathscr{O}$ reconstructed using the inverse 2D-DCT transform.

1. Crop and partition $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ non-overlapping blocks;
2. Calculate the number of non-adaptive measurements per block $m = \lfloor C_R \cdot B^2 \rfloor$;
3. Collect m lowpass DCT coefficients in zigzag order from each of the n_B blocks;
4. Transmit $M = m \times n_B$ measurements;
5. Reconstruct the Image from the m Transform Coefficients (TCs) per block using the inverse 2D-DCT transform.

We have chosen the 2D-DCT transform to collect linear measurements because it is close to optimal on image data [90], the measurements can be easily acquired on single-pixel cameras, multi-pixel cameras and focal plane processing image sensors, and allow the image to be reconstructed in real-time using the inverse 2D-DCT.

4.3.2 AL-DCT-THB and AL-DCT-THI

If the objective is not compressive sensing but compression, all the transform coefficients may be collected (full sensing), and then only those whose absolute value exceeds some threshold are retained for transmission. It is possible to set one threshold for the whole image - THI, or one threshold per block - THB. In the first case, the number of retained transform coefficients per block varies block-by-block so that the Threshold from the whole Image (THI) algorithm is inherently adaptive. The number of retained transform coefficients in the Threshold per Block (THB) algorithm is the same in each block, but the positions of the transform coefficients in the DCT domain are not predetermined. The THI algorithm sets an upper bound on the performance that can be achieved using adaptive 2D-DCT techniques. The two algorithms, AL-DCT-THB and AL-DCT-THI, are defined formally in algorithms 4.2 and 4.3.

Algorithm 4.2 AL-DCT-THB

Input: Image $\mathscr{I}$, compression ratio $C_R = M/N$, block size B, image width W, image height H.
Output: Image $\mathscr{O}$ sensed using AL-DCT-THB, reconstructed using the inverse 2D-DCT transform.

1. Crop and partition $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ non-overlapping blocks;
2. Collect all $B \times B$ L-DCT-ZZ coefficients from each of the n_B blocks;
3. Select the $m = \lfloor C_R \cdot B^2 \rfloor$ largest coefficients in each block;
4. Transmit $M = m \times n_B$ measurements;
5. Reconstruct the Image from the m TCs per block using the inverse 2D-DCT transform.

Algorithm 4.3 AL-DCT-THI

Input: Image $\mathscr{I}$, compression ratio $C_R = M/N$, block size B, image width W, image height H.
Output: Image $\mathscr{O}$ sensed using AL-DCT-THI, reconstructed using the inverse 2D-DCT transform.

1. Crop and partition $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ non-overlapping blocks;
2. Collect all 2D-DCT coefficients from all blocks;
3. Select the M largest coefficients from all blocks;
4. Transmit the M measurements;
5. Reconstruct the Image on a block-by-block basis using the inverse 2D-DCT transform.

4.4 Adaptive image BCS

The non-adaptive L-DCT-ZZ algorithm measures a fixed number of transform coefficients per block. In the adaptive schemes, the number of transform coefficients collected varies according to the sparsity of each block. Figure 4.1 shows a block diagram of the proposed adaptive system.

Two techniques for adapting the number of linear 2D-DCT transform coefficients in block compressive sensing are proposed, one in the spatial domain and the other in the 2D-DCT transform domain. Both techniques use two phases to encode an image adaptively. In the first phase, m_i^1 measurements are collected from each block i and used to determine the number of additional transform coefficients, m_i^2, to be collected from all blocks in the second phase.

In the case of the DD technique, the m_i^1 measurements are collected as the low-frequency components, as shown in figure 4.2, and determine the next m_i^2 measurements to be collected in phase 2.

All the measurements collected from both phases are considered for reconstruction purposes and hence the total number of measurements collected $M = \sum_{i=1}^{n_B}(m_i^1 + m_i^2)$, where n_B is the number of blocks in the image.

The methods used to estimate m_i^2 must not be computationally intensive because they are

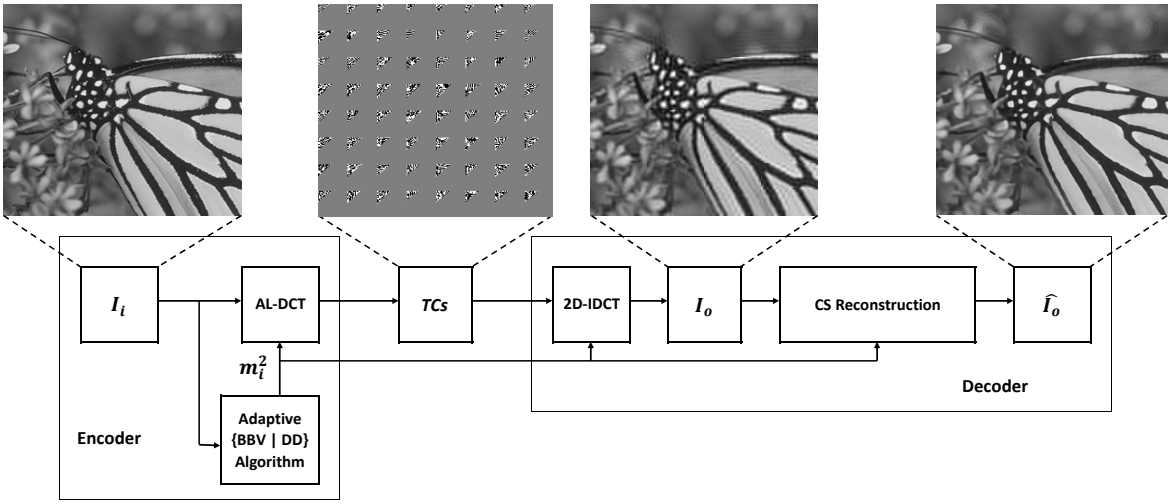


Figure 4.1: Adaptive AL-DCT-BBV and AL-DCT-DD system architecture. I_i is the input image, I_o is the image output in real time by the inverse 2D-DCT and $\widehat{I_o}$ is the higher quality image output by the CS reconstruction, for example, using the IDA algorithm. AL-DCT-{BBV|DD} acquires m_i^2 adaptive CS measurements per block where each measurement is a 2D-DCT TC.

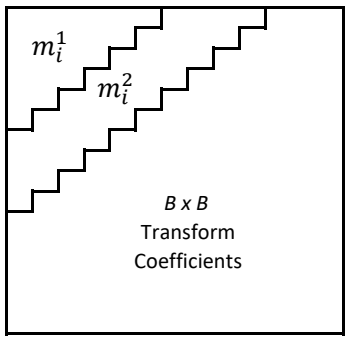


Figure 4.2: Adaptive AL-DCT-DD phase 1 and 2 measurements.

implemented at the encoder side, and a target application for the methods described in this dissertation is for autonomously powered wireless image and video sensors. However, it is also possible to transmit the m_i^1 measurements to a decoder and use more elaborate algorithms at the decoder to estimate m_i^2 per block. This information is then fed back to the encoder as has been proposed in the literature [64], [65], [66]. In power-constrained applications, it is necessary to consider whether the power required to operate a receiver module to receive the m_i^2 feedback information is greater than the power required to just compute m_i^2 at the encoder. The requirement to wait for the reception of feedback information will also add to the reconstruction time of the image, although the increase may be contained if the sensing node is close to the decoder and the data rate of the link is sufficiently high.

Figure 4.1 shows the block transform coefficients reconstructed in real time at the decoder using an inverse 2D-DCT transform and reconstituted into the output image I_o. The number of phase-two transform coefficients, m_i^2, is transmitted as side information. If better reconstruction quality is required, full-image CS reconstruction is invoked to output image $\widehat{I_o}$, albeit with a large reconstruction time penalty.

4.4.1 Adaptive L-DCT-ZZ using block boundary variation

The adaptive L-DCT-ZZ algorithm using BBV, i.e. AL-DCT-BBV, is described in algorithm 4.4. The idea behind this technique is to adapt the number of transform coefficients collected in proportion to the total variation of pixel values around the perimeter of a block. It is inspired by the observation that if a block includes significant texture features, then these will likely cross block-boundaries and can be detected by measuring the Total Variation (TV) at the block border. Since the number of pixels in the boundary is only $4B$, determining m_i^2 from just these pixels is more efficient than having to determine it from all B^2 pixels ($4B$ is the boundary constructed from efficient adjacent block measurements, as depicted in figure 4.3, it is not the $(4B-4)$ boundary of a block in isolation). When the compression factor $C_F = B^2/m_i$ in the block is high, the number of transform coefficients that can be collected is low, and the measurements required to calculate the BBV become a significant fraction of the whole budget for the block.

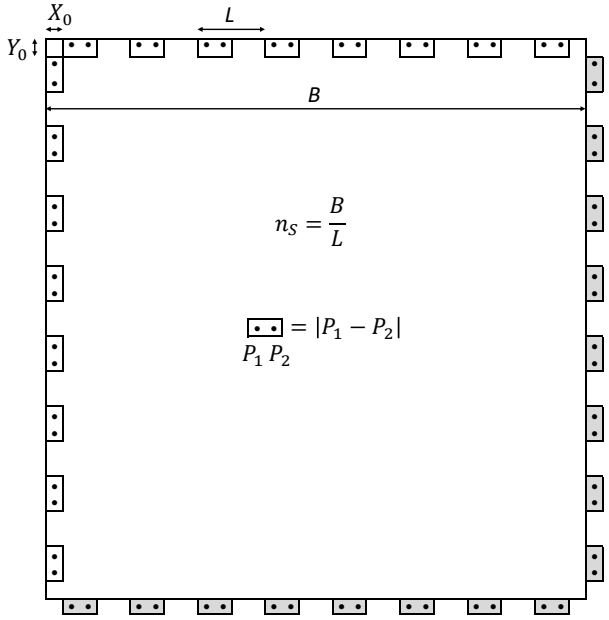


Figure 4.3: Block boundary variation measurements for a 32x32 block. The shaded measurements are imported from adjacent blocks.

The measurement of the BBV can either be treated as a necessary overhead, or it is accounted for by reducing the number of transform coefficients that can be collected. Our work takes the latter view and attempts to minimize the number of measurements to estimate the BBV. This is accomplished in three ways. First, only the TV in the top row and left-hand column borders are measured in each block. Since the blocks stack to form an image, the right-hand TV is obtained from the next block to the right, and the bottom row TV is obtained from the bock below the current row. This reduces the number of pixels that need to be measured per block to $(2B-1)$. Second, rather than collecting pixel values, it is possible to collect pixel difference measurements, which involve only two adjacent pixels at a time. Thus, it is possible to reduce the measurements to approximately B per block. Third, it is possible that not all adjacent pixel measurements are necessary and that the BBV can be estimated from fewer measurements. It is thus possible to collect a pixel difference measurement every L pixels around the block border.

Algorithm 4.4 AL-DCT-BBV

Input: Image $\mathscr{I}$, compression factor C_F, block size B, Reconstruction Algorithm = {IDCT2|D-AMP|DAMP-D|IDA}, image width W, image height H.
Output: Image $\mathscr{O}$ sensed using AL-DCT-BBV and reconstructed using a reconstruction algorithm.

1. Crop and partition image, $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ blocks;
2. Collect $m_i^1 = 2 \cdot n_S$ samples per block, where the $n_S = \lfloor B/C_F \rfloor$ measurements are equally spaced in the top and left-hand block borders;
3. Estimate the number m_i^2 of additional samples to collect from each block i, where m_i^2 is given by equation (4.7);
4. Collect the m_i^2 transform coefficients from each block i;
5. Transmit $M = \sum_{i=1}^{n_B} (m_i^1 + m_i^2)$ measurements;
6. Reconstruct $\mathscr{O}$ from the M received samples using the selected reconstruction algorithm.

The right-hand blocks and the bottom blocks do not have right-hand and bottom neighbours that can contribute to the missing edges of the boundary. In these cases, the right-hand and bottom TVs can be measured from the current block.

Figure 4.3 shows BBV measurements in a 32×32 pixel block. The variation measurements in the current block, which consist of the absolute differences $|P_1 - P_2|$ of two adjacent pixels P_1 and P_2, are shown as rectangles outlined in black. X_0 is the pixel offset of the first variation measurement in a row, Y_0 is the pixel offset of the first measurement in a column, and L is the stride between pixel variation measurements. In the figure, $X_0 = 2$, $Y_0 = 2$ and $L = 4$. We acquire phase 1 measurements that decrease as C_F increases. Hence, we set L empirically:

$$L = \lfloor C_F \rfloor, \tag{4.1}$$

and

$$X_0 = Y_0 = \lfloor L/2 \rfloor. \tag{4.2}$$

The number of measurements n_S per block side is given by:

$$n_S = \lfloor B/L \rfloor, \tag{4.3}$$

and the number of measurements used to calculate the BBV is $4 \cdot n_S$, with half of them in the current block and the other half in adjacent blocks, shaded in grey in figure 4.3. BBV_i is given by:

$$
\begin{aligned}
BBV_i = \sum_{j=0}^{n_S-1} \{ \quad & |P_{(1,X_0+jL+1)} - P_{(1,X_0+jL)}| \\
& + |P_{(Y_0+jL+1,1)} - P_{(Y_0+jL,1)}| \\
& + |P_{(B+1,X_0+jL+1)} - P_{(B+1,X_0+jL)}| \\
& + |P_{(Y_0+jL+1),B+1)} - P_{(Y_0+jL,B+1)}| \quad \}.
\end{aligned} \tag{4.4}
$$

If $C_F > B$, n_s is zero according to equations (4.3) and (4.1). In this case, the algorithm reverts to being non-adaptive.

The number of blocks n_B in an image of H by W pixels is given by:

$$n_B = \lfloor H/B \rfloor \lfloor W/B \rfloor, \tag{4.5}$$

where the floor operation indicates that the image dimensions are cropped to be multiples of block size B. Then, the number of BBV measurements M_{BBV} required in phase one is equal to the number of measurements in each block i.e. $2 \cdot n_S$, multiplied by the number of blocks n_B and adding the measurements at the right and bottom edges and is given by:

$$M_{BBV} = 2n_S n_B + (H+W)/L. \tag{4.6}$$

These BBV measurements, in addition to being used to calculate the number of measurements m_i^2 required in phase two, can also be transmitted to the decoder to serve as additional reconstruction measurements. Then, assuming the total number of measurements used in image reconstruction is equal to M, the number of measurements in block i during phase 2, m_i^2, is given by the number of remaining measurements $(M - M_{BBV})$ distributed amongst the blocks as a proportion of the block boundary variation measurement in block BBV_i to the total block boundary measurements from all blocks and is hence given by:

$$m_i^2 = \left\lfloor (M - M_{BBV}) \frac{BBV_i}{\sum_{i=1}^{n_B} BBV_i} \right\rfloor. \tag{4.7}$$

where BBV_i is the BBV of block i, which is the sum of the absolute difference of two adjacent pixels at each measurement point around the boundary of the block. Half of the measurements are collected from block i, and the other half are collected from the adjacent blocks to the right and below, as shown in figure 4.3. m_i^2 has to be capped at $B^2 - m_i^1$, and any remaining measurements can be collected from other non-fully measured blocks.

4.4.2 Adaptive L-DCT-ZZ in the DCT domain

The adaptive L-DCT-ZZ algorithm in the DCT domain, AL-DCT-DD, described in algorithm 4.5, uses m_i^1 linear 2D DCT measurements per block i, in zigzag scan order, to compute the number of measurements m_i^2 to collect in phase 2, as shown in figure 4.2. Several authors have

Algorithm 4.5 AL-DCT-DD

Input: Image $\mathscr{I}$, compression factor C_F, block size B, Reconstruction Algorithm = {IDCT2 | D-AMP|DAMP-D|IDA}, image width W, image height H.
Output: Image $\mathscr{O}$ sensed using AL-DCT-DD and reconstructed using reconstruction algorithm.

1. Crop and partition, $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ blocks;
2. Calculate the number of non-adaptive measurements per block $m_i^1 = \lfloor B^2/(2 \cdot C_F) \rfloor$;
3. Collect m_i^1 L-DCT-ZZ coefficients from each block;
4. Estimate the number of L-DCT-ZZ transform coefficients to collect per block in phase 2, m_i^2, from equation (4.8);
5. Collect m_i^2 additional coefficients as required;
6. Transmit $M = \sum_{i=1}^{n_B} (m_i^1 + m_i^2)$ measurements;
7. Reconstruct $\mathscr{O}$ from the M received samples using the selected reconstruction algorithm.

considered first capturing a full set of measurements ($M = N$) to compute the value of all the pixels at the encoder and then adapting the number of measurements to transmit per block, as reported in section 4.2. These full sensing techniques are feasible if the power required to sense the transform coefficients directly in the optical domain is less than the power required to capture pixels, convert them from analogue to digital, and then perform the transformation using digital computations. Zhu et al. [65] also consider a similar *significance-based allocation factor*, but this requires a whole block of transform coefficients. CS techniques, such as our AL-DCT-DD algorithm, collect fewer measurements ($M < N$) and hence require less sensing power than full sensing techniques.

The best PSNR performance by an adaptive algorithm in the DCT domain is achieved when using the full sensing algorithm 4.3 AL-DCT-THI that selects the highest valued transform coefficients from all the blocks. However, if compressive sensing is used, the strategy in the AL-DCT-DD algorithm is to assign m_i^2 in proportion to the number n_i of the m_i^1 measurements whose absolute value exceeds a threshold T with values given in table 4.2. We found empirically that collecting half the number of available measurements in phase 1, that is, $n_B \cdot m_i^1 = M/2$, allows us to distribute the remaining measurements in proportion to n_i such that m_i^2 is given by:

$$m_i^2 = \left\lfloor n_B m_i^1 \frac{n_i}{\sum_{j=1}^{n_B} n_j} \right\rfloor . \tag{4.8}$$

The selection of T is crucial to the performance of the algorithm. The best values were found empirically as shown in table 4.1 and are summarized in table 4.2 for different image heights H and compression factors C_F.

C_R	C_F	256×256 Image Set	512×512 Image Set
Non-Adaptive			
0.1	10.00	26.44	28.63
0.2	5.00	29.37	31.62
0.3	3.33	31.95	33.82
0.4	2.50	34.38	35.90
0.5	2.00	36.93	38.06
Adatpive $T = 15$			
0.1	10.00	26.82	29.10
0.2	5.00	30.10	32.39
0.3	3.33	33.14	34.85
0.4	2.50	35.89	37.05
0.5	2.00	**38.58**	**39.19**
Adatpive $T = 30$			
0.1	10.00	26.96	29.29
0.2	5.00	30.32	**32.50**
0.3	3.33	33.43	**34.94**
0.4	2.50	**36.10**	**37.08**
0.5	2.00	38.55	39.08
Adaptive $T = 60$			
0.1	10.00	**27.07**	**29.40**
0.2	5.00	**30.48**	32.41
0.3	3.33	**33.46**	34.79
0.4	2.50	35.98	36.75
0.5	2.00	38.24	38.64
Adaptive Optimized			
0.1	10.00	**27.07**	**29.40**
0.2	5.00	**30.48**	**32.50**
0.3	3.33	**33.46**	**34.94**
0.4	2.50	**36.10**	**37.08**
0.5	2.00	**38.58**	**39.19**
Average		**33.14**	**34.62**

Table 4.1: Optimizing the DCT Domain (DD) adaptivity algorithm threshold T empirically to maximize PSNR. Largest values in bold.

	H = 256	H = 512
T = 15	$C_F \leq 2$	$C_F \leq 2$
T = 30	$2 < C_F < 3.33$	$2 < C_F < 10$
T = 60	$C_F \geq 3.33$	$C_F \geq 10$

Table 4.2: Values of T based on image height H and C_F. The dynamic range of the pixel values is [0 255].

Figure 4.4: Set256: 256×256 image set.

Figure 4.5: Set512: 512×512 image set.

4.4.3 Analysis of the BBV and DD adaptive techniques

Using the 2D DCT as the sparsifying domain, the number of transform coefficients in each block, m_i, that maximize PSNR, can be found by measuring all transform coefficients in all the blocks and selecting the M largest transform coefficients, irrespective of the blocks in which they occur. This follows from the orthogonal property of the DCT [90], meaning that it preserves the inner product. Since the sum of the square of the image pixel values is equal to the sum of the square of the transform coefficients, if we select the highest valued transform coefficients, we are guaranteed to collect the highest signal energy.

The resulting algorithm is referred to as the THI algorithm. The higher the number of transform coefficients in a block i, the lower the sparsity in the block, and the more measurements that are required from that block. It is required to show that the number of measurements per block $\widehat{m}_i$ estimated by the BBV and DD algorithms are well correlated with the number of

measurements m_i that maximize PSNR.

ADAPT = DD						ADAPT = BBV					
Image Set	C_R					Image Set	C_R				
256×256	0.1	0.2	0.3	0.4	0.5	256×256	0.1	0.2	0.3	0.4	0.5
Barbara	**0.531**	**0.488**	**0.476**	**0.596**	0.807	Barbara	0.724	0.813	0.773	0.766	0.793
Boat	0.826	0.816	0.821	0.900	0.932	Boat	**0.638**	0.712	**0.677**	0.753	0.818
Cameraman	0.861	0.842	0.802	0.871	0.940	Cameraman	0.741	0.858	0.853	0.844	0.862
House	0.905	0.821	0.741	0.743	0.776	House	0.791	0.812	0.865	0.859	0.858
Leaves	0.767	0.910	0.905	0.923	0.932	Leaves	**0.515**	0.771	0.811	0.791	0.794
Lena	0.664	0.753	0.794	0.898	0.913	Lena	**0.671**	0.818	0.825	0.841	0.871
Monarch	0.843	0.905	0.887	0.872	0.914	Monarch	**0.542**	0.705	0.787	0.728	0.787
Parrots	0.847	0.929	0.882	0.902	0.924	Parrots	0.852	0.871	0.872	0.895	0.906
Peppers	0.767	0.782	0.720	0.812	0.841	Peppers	**0.330**	**0.512**	**0.643**	**0.501**	**0.613**
Average	0.779	0.805	0.781	0.835	0.887	Average	**0.645**	0.764	0.789	0.775	0.811
Image Set	C_R					Image Set	C_R				
512×512	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	512×512	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Barbara	**0.361**	**0.403**	**0.396**	**0.432**	**0.656**	Barbara	0.813	0.817	0.823	0.834	0.830
Barbara2	**0.537**	0.750	0.813	0.838	0.897	Barbara2	0.743	**0.680**	**0.648**	**0.676**	0.700
Boat	0.868	0.890	0.888	0.872	0.876	Boat	0.723	0.798	0.788	0.805	0.831
Cameraman	0.841	0.923	0.911	0.898	0.940	Cameraman	0.837	0.851	0.855	0.862	0.894
Girl	0.778	0.841	0.783	0.734	**0.695**	Girl	**0.509**	**0.598**	**0.637**	**0.611**	**0.615**
Goldhill	0.817	0.877	0.869	0.845	0.905	Goldhill	0.705	0.796	0.826	0.845	0.844
Lena	0.800	0.885	0.874	0.847	0.864	Lena	0.781	0.892	0.881	0.887	0.889
Living Room	0.886	0.922	0.916	0.893	0.923	Living Room	0.749	0.751	0.771	0.804	0.807
Man	0.698	0.808	0.842	0.840	0.906	Man	0.688	0.830	0.860	0.799	0.808
Mandrill	0.726	0.876	0.929	0.947	0.956	Mandrill	0.858	0.910	0.899	0.910	0.914
Peppers	0.844	0.828	0.778	0.731	0.750	Peppers	**0.503**	**0.699**	0.759	**0.612**	**0.695**
Average	0.741	0.818	0.818	0.807	0.852	Average	0.719	0.784	0.795	0.786	0.802

Table 4.3: Correlation coefficient $\rho_{m_i,\widehat{m}_i}$ for 256×256 and 512×512 image sets for both BBV and DD adaptive algorithms with varying C_R. Correlation values below 0.7 are shown in bold.

The correlation coefficient $\rho_{m_i,\widehat{m}_i}$ and the confidence level $(1-p)$, where p is the probability of the null hypothesis (that there is no correlation, i.e., $\rho_{m_i,\widehat{m}_i} = 0$), were measured for the Set256 and Set512 image sets in figures 4.4 and 4.5 for $C_R = \{0.1...0.5\}$. In all cases, the probability of the null hypothesis was always below 0.0001, indicating strong confidence in the correlation results. The average correlation results $\rho_{m_i,\widehat{m}_i}$ for the 256×256 and 512×512 image sets are tabulated in table 4.3 for both the BBV and DD algorithms (original image sets are available in [91]).

For both image sets, the best correlation was obtained with the DD algorithm. The correlation was strongest at $C_R = 0.5$. The lowest correlation occurs for BBV at $C_R = 0.1$ and may be due to the low level of measurements acquired to compute m_i, since the number of block

boundary measurements is proportional to C_R. This indicates that BBV should probably have a minimum number of boundary measurements from a correlation perspective, although this could impact PSNR and SSIM, because each phase-one measurement reduces the number of DCT measurements.

4.4.4 Real-time deblocking

The advantage of deterministic sensing in L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD is that the signals can be decoded very rapidly at the decoder, such that the collected images or frames can be used in real-time image and video transmission, for example. Block-based L-DCT-ZZ, similar to other block-based image compression algorithms, suffers from blocking artefacts at low C_R values, for example $C_R \leq 0.1$. Although the adaptive algorithms were found to attenuate blocking artefacts caused by the block-based measurements, some blocking is still visible when $C_R \leq 0.1$. Blocking can be removed in real time using the ZZF or the THF described in this section. Better quality deblocking and image quality enhancement can be achieved using the CS reconstruction algorithms proposed and studied in this chapter. These are derived from Metzler's D-AMP algorithm [14] and are described in section 4.5.

C_R	L-DCT-ZZ-IDCT			L-DCT-ZZ-ZZF			L-DCT-ZZ-THF			L-DCT-ZZ-BM3DF		
	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s
0.10	26.44	0.8096	0.0069	26.36	0.8036	0.0139	26.48	0.8042	0.0152	26.59	0.8224	0.1173
0.20	29.37	0.8889	0.0069	29.22	0.8836	0.0136	29.38	0.8840	0.0150	29.46	0.8930	0.1193
0.30	31.95	0.9308	0.0070	31.81	0.9275	0.0143	31.95	0.9270	0.0151	32.00	0.9315	0.1237
0.40	34.38	0.9562	0.0071	34.21	0.9539	0.0143	34.37	0.9536	0.0148	34.41	0.9557	0.1217
0.50	36.93	0.9728	0.0071	36.76	0.9710	0.0140	36.91	0.9710	0.0156	36.89	0.9704	0.1263
Average	31.81	0.9117	0.0070	31.67	0.9079	0.0140	31.82	0.9080	0.0151	31.87	0.9146	0.1217
C_R	AL-DCT-BBV-IDCT			AL-DCT-BBV-ZZF			AL-DCT-BBV-THF			AL-DCT-BBV-BM3DF		
	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s
0.10	26.70	0.7984	0.0067	25.75	0.7714	0.0134	26.40	0.7715	0.0147	26.84	0.8104	0.1193
0.20	30.49	0.8922	0.0066	28.79	0.8637	0.0133	30.08	0.8716	0.0148	**30.60**	0.8983	0.1173
0.30	33.37	0.9338	0.0066	31.38	0.9143	0.0139	32.88	0.9180	0.0147	33.45	**0.9366**	0.1220
0.40	35.94	0.9563	0.0067	33.67	0.9422	0.0142	35.38	0.9447	0.0150	35.99	0.9577	0.1213
0.50	38.29	0.9702	0.0068	35.99	0.9608	0.0147	37.71	0.9619	0.0153	38.25	0.9700	0.1250
Average	32.96	0.9102	0.0067	31.12	0.8905	0.0139	32.49	0.8935	0.0149	33.03	0.9146	0.1210
C_R	AL-DCT-DD-IDCT			AL-DCT-DD-ZZF			AL-DCT-DD-THF			AL-DCT-DD-BM3DF		
	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s
0.10	27.07	0.8173	0.0066	26.45	0.7974	0.0134	26.99	0.8047	0.0150	**27.22**	**0.8302**	0.1146
0.20	30.48	0.8953	0.0068	29.25	0.8765	0.0136	30.25	0.8807	0.0155	30.59	**0.9001**	0.1169
0.30	33.46	0.9350	0.0067	31.80	0.9207	0.0138	33.09	0.9230	0.0147	**33.52**	0.9364	0.1230
0.40	36.10	0.9588	0.0066	34.18	0.9483	0.0138	35.69	0.9507	0.0148	**36.12**	**0.9590**	0.1241
0.50	**38.58**	**0.9735**	0.0066	36.61	0.9661	0.0139	38.17	0.9686	0.0146	38.47	0.9717	0.1253
Average	33.14	0.9160	0.0067	31.66	0.9018	0.0137	32.84	0.9055	0.0149	**33.19**	**0.9195**	0.1208

Table 4.4: Comparison of IDCT reconstruction and ZZF, THF and BM3DF deblocking on the Set256 image set, with best PSNR and SSIM in bold.

The ZZF deblocks images reconstructed by the L-DCT-ZZ algorithm by executing the L-DCT-ZZ algorithm on the whole image and retaining only the first M low-pass transform coef-

C_R	L-DCT-ZZ-IDCT			L-DCT-ZZ-ZZF			L-DCT-ZZ-THF			L-DCT-ZZ-BM3DF		
	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s
0.10	28.63	0.7940	0.0261	28.57	0.7886	0.0649	28.67	0.7903	0.0696	28.69	0.7900	0.50
0.20	31.62	0.8767	0.0257	31.53	0.8719	0.0656	31.64	0.8733	0.0717	31.60	0.8701	0.52
0.30	33.82	0.9164	0.0260	33.72	0.9129	0.0663	33.82	0.9136	0.0722	33.73	0.9109	0.53
0.40	35.90	0.9411	0.0261	35.75	0.9378	0.0686	35.88	0.9389	0.0729	35.72	0.9371	0.54
0.50	38.06	0.9593	0.0263	37.79	0.9564	0.0656	38.02	0.9578	0.0731	37.54	0.9537	0.53
Average	33.61	0.8975	0.0260	33.47	0.8935	0.0662	33.61	0.8948	0.0719	33.45	0.8924	0.52
C_R	AL-DCT-BBV-IDCT			AL-DCT-BBV-ZZF			AL-DCT-BBV-THF			AL-DCT-BBV-BM3DF		
	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s
0.10	29.18	0.7922	0.0261	28.25	0.7666	0.0668	28.98	0.7778	0.0717	29.23	0.7911	0.52
0.20	**32.66**	0.8794	0.0262	31.25	0.8548	0.0664	32.38	0.8675	0.0723	32.65	0.8770	0.54
0.30	**35.30**	**0.9196**	0.0262	33.47	0.8986	0.0660	34.94	0.9106	0.0711	35.22	0.9172	0.55
0.40	**37.42**	**0.9425**	0.0266	35.37	0.9253	0.0649	37.04	0.9359	0.0722	37.29	0.9406	0.52
0.50	**39.37**	0.9571	0.0262	37.24	0.9450	0.0648	38.99	0.9524	0.0722	39.01	0.9538	0.52
Average	**34.79**	0.8982	0.0263	33.12	0.8781	0.0658	34.47	0.8888	0.0719	34.68	0.8959	0.53
C_R	AL-DCT-DD-IDCT			AL-DCT-DD-ZZF			AL-DCT-DD-THF			AL-DCT-DD-BM3DF		
	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s	PSNR dB	SSIM	Time s
0.10	29.40	**0.8000**	0.0260	28.65	0.7823	0.0648	29.28	0.7918	0.0728	**29.42**	0.7966	0.48
0.20	32.50	**0.8797**	0.0257	31.56	0.8669	0.0663	32.34	0.8737	0.0738	32.40	0.8741	0.50
0.30	34.94	0.9179	0.0256	33.77	0.9072	0.0675	34.72	0.9130	0.0731	34.73	0.9136	0.51
0.40	37.08	0.9408	0.0257	35.67	0.9314	0.0644	36.82	0.9365	0.0726	36.78	0.9377	0.52
0.50	39.19	**0.9595**	0.0260	37.70	0.9520	0.0643	38.98	0.9567	0.0709	38.52	0.9548	0.53
Average	34.62	**0.8996**	0.0258	33.47	0.8880	0.0655	34.43	0.8943	0.0726	34.37	0.8954	0.50

Table 4.5: Comparison of IDCT reconstruction and ZZF, THF and BM3D deblocking on the 512×512 Image Set, with best PSNR and SSIM in bold.

ficients in zigzag scan order, where $M = \lfloor (N/C_F) \rfloor$, and $\lfloor x \rfloor$ is the integer part of x.

If ZZF is applied to the AL-DCT-BBV and AL-DCT-DD algorithms, it filters out high frequency transform coefficients that are preserved by the adaptive process. The THF first computes a 2D-DCT of the whole image and then retains the M largest transform coefficients before re-inverting them to the spatial domain, thereby protecting the high frequencies preserved by AL-DCT-BBV and AL-DCT-DD.

If the image is non-square, then it is first split into two overlapping square sub-images, which are deblocked independently. The two deblocked sub-images are then reformed into the rectangular image by overlapping them and averaging the pixels in the overlap region.

Tables 4.4 and 4.5 show the PSNR, SSIM and execution time for L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD algorithms reconstructed with Inverse 2D DCT (IDCT) and then deblocked with ZZF, THF or the BM3D filtering (BM3DF) [92] (BM3D with the fast 'Lc' option set, and σ set to C_F) for the two image sets. Execution time was estimated using the Matlab tic and toc functions.

Deblocking is mostly required when C_R is low for example $C_R \leq 0.1$. With both image sets, BM3DF improved the reconstruction most for all tested C_R values, followed very closely by THF. Since the latter executes 4 to 22 times faster, it is the preferred solution for real-time operation.

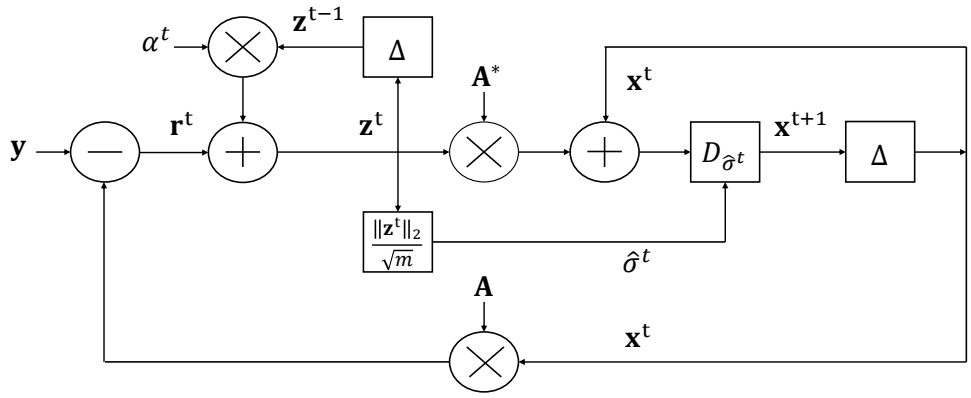


Figure 4.6: Generic block diagram for iterative denoising reconstruction algorithms: D-AMP, D-AMP Damped (DAMP-D) and IDA. This depicts equations 2.13, 2.14 and 2.15 in the text. The delay blocks Δ impart a one-iteration delay such that the output is the vector in the previous iteration. The Onsager term is $\alpha^t \mathbf{z^{t-1}}$ with α^t given by equation 4.9 and is derived from the divergence of the $\mathbf{x^t}$ vector from the previous iteration.

4.5 Reconstructing deterministic CS using iterative denoising-based algorithms

In this section, we build on the work of Metzler [14] on D-AMP, described in section 2.2.2, to design iterative algorithms that better reconstruct images compressively sensed using AL-DCT-BBV and AL-DCT-DD. The iterative denoising-based reconstruction algorithm is represented in block diagram form in figure 4.6. In D-AMP:

$$\alpha^t = \frac{1}{\delta} \frac{div[D_{\hat{\sigma}^{t-1}}(\mathbf{x^{t-1}} + \mathbf{A}^* \mathbf{z^{t-1}})]}{m}, \tag{4.9}$$

so that $\alpha^t \mathbf{z^{t-1}}$ is the Onsager term. The Δ block introduces a delay of one iteration, $D_{\hat{\sigma}_t}$ is the denoiser block, and $\mathbf{r^t}$ is given by:

$$\mathbf{r^t} = \mathbf{y} - \mathbf{A}\mathbf{x^t}. \tag{4.10}$$

The other variables are as defined for D-AMP in section 2.2.1. To use D-AMP on deterministic, block-based CS, the deterministic forward transform is cast as a sensing matrix $\mathbf{A}$ which has a sparse representation:

$$\mathbf{A} = \begin{bmatrix} \mathbf{B}_{1,1} & & & \\ & \ddots & & \\ & & \mathbf{B}_{i,i} & \\ & & & \ddots \\ & & & & \mathbf{B}_{n_B,n_B} \end{bmatrix}, \tag{4.11}$$

where $\mathbf{B}_{i,i}$ are $m \times n$ basis matrices with $n = B \times B$, $m = \lfloor n \times C_R \rceil$, C_R is the compression ratio M/N, B is the size of one side of the image block and n_B is the number of blocks in the image. In non-adaptive CS, m and n are constant since there are equal measurements per block. In

adaptive CS, m varies per block. $\mathbf{A}^*$ is the transpose of $\mathbf{A}$.

When $\mathbf{B}$ is a random Gaussian block matrix, we empirically found that D-AMP frequently failed to recover some of the blocks correctly, leading to very significant blocking artefacts. This never happens when the row vectors of $\mathbf{B}$ are taken from the 2D-DCT basis functions, in zigzag order, with the first basis function being that for the DC component.

BM3D and DnCNN Denoisers

Metzler et al. [16] implemented their D-AMP algorithm as an unrolled deep neural network, replacing the BM3D [92] with DnCNN [27], which is known to be a more accurate and faster denoiser. In this chapter, we compare the performance of iterative reconstruction using the DnCNN and BM3D denoisers.

The block diagram of the DnCNN denoiser is shown in figure 4.7. The version of DnCNN used consists of 17 layers. The first layer consists of 64, $3 \times 3 \times 1$ convolutional filters. The next fifteen layers consist of 64, $3 \times 3 \times 64$ convolutional filters, followed by batch normalization [93] and a Rectified Linear Unit (ReLU) [94]. The final layer consists of $3 \times 3 \times 64$ convolutional filters. The neural network was trained using 400 images from the Berkeley Segmentation Dataset (BSDS500) [44], cropped to 180×180 pixels.

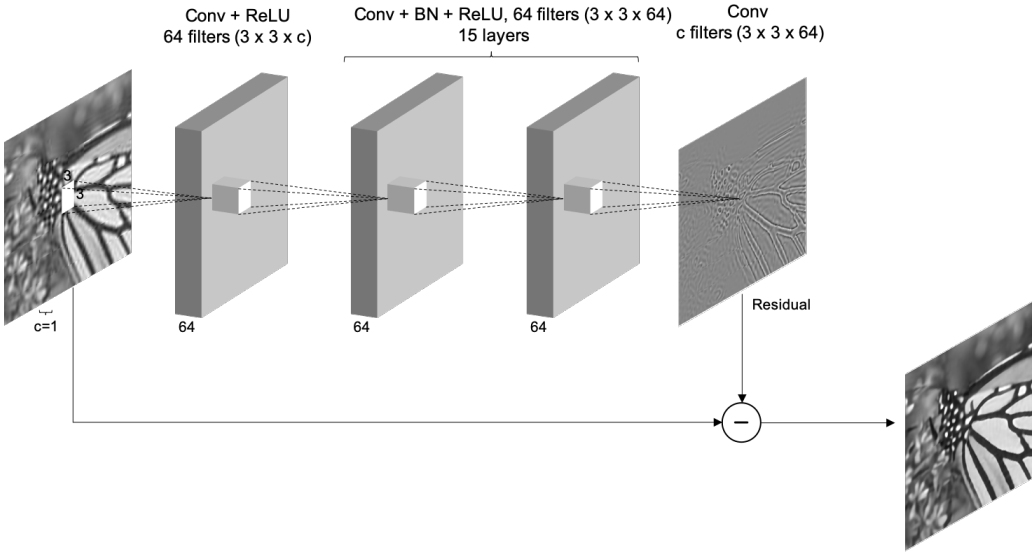


Figure 4.7: The 17-layer DnCNN [27] used to reconstruct grayscale (one colour component, c=1) images. Convolution (Conv), rectified linear unit (ReLU), batch normalization (BN).

Modified D-AMP - DAMP-D

D-AMP was derived based on the assumption of a Gaussian distribution of the residual reconstruction error at each iteration [14]. When the measurement matrix is a low-pass deterministic 2D-DCT matrix, this assumption no longer holds, and the reconstruction quality deteriorates. Figure 4.6 shows that the Onsager term implements an adaptive integration of the reconstruction error, akin to integral control in a closed-loop feedback system. This inspires us to vary

the integral control loop gain by introducing a damping factor D_F. Then, α^t in equation (4.9) is divided by D_F which acts as a damping factor in the integral loop in figure 4.6, and becomes:

$$\alpha^t = \frac{1}{\delta} \frac{div[D_{\hat{\sigma}^{t-1}}(\mathbf{x^{t-1}} + \mathbf{A}^* \mathbf{z^{t-1}})]}{m} \frac{1}{D_F}. \tag{4.12}$$

Suitable values of D_F are investigated in figure 4.10. The modified algorithm is referred to as DAMP-D with damping factor D_F.

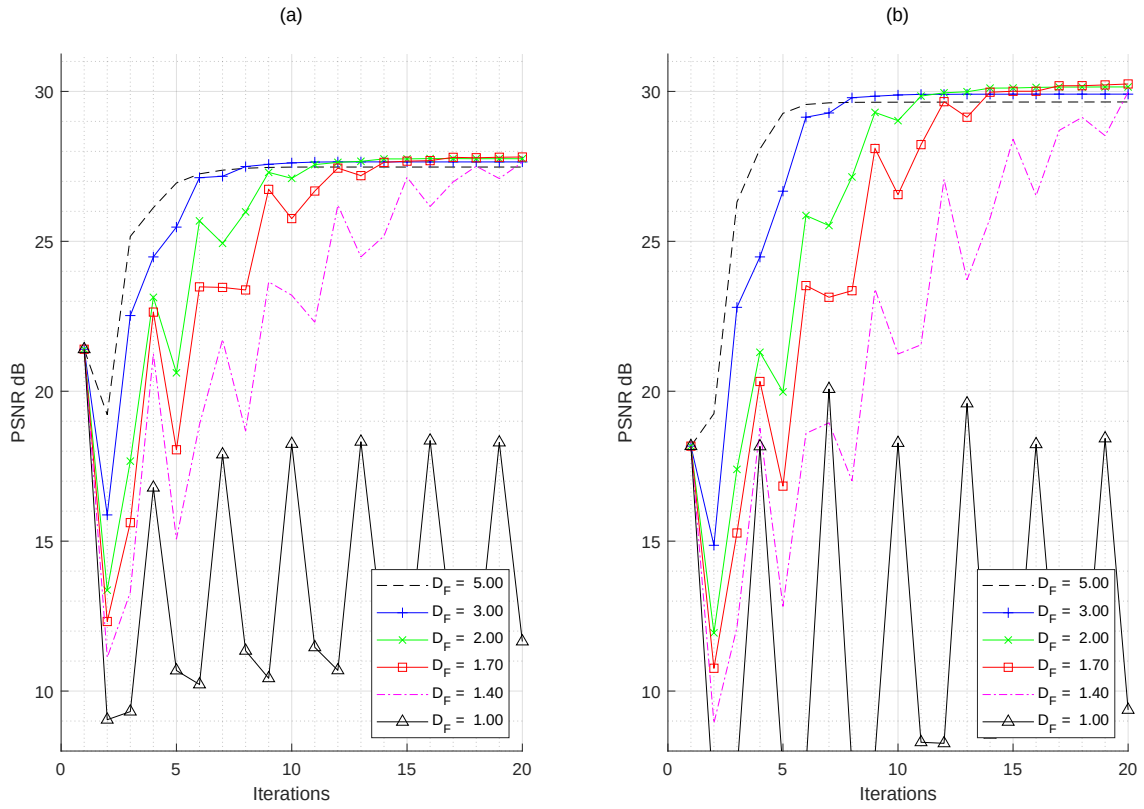


Figure 4.8: Reconstructing AL-DCT-DD using (a) D-AMP (when $D_F = 1.00$) and IDA (when $D_F > 1.00$) with a BM3D denoiser and (b) using D-AMP and IDA with a DnCNN denoiser. The results shown for the 256×256 Monarch image sensed at $C_R = 0.1$.

Simplified D-AMP - IDA

Since the Onsager term requires the residual error to be Gaussian, it was hypothesized that a simplified version of the iterative denoising algorithm might provide better performance. Indeed, simplifying α^t to:

$$\alpha^t = \frac{1}{D_F}, \tag{4.13}$$

was found to provide better performance than DAMP-D. The ensuing algorithm is called the IDA with damping factor D_F.

Figure 4.8 shows that IDA without damping, i.e., $D_F = 1$, does not reconstruct the AL-DCT-DD-sensed Monarch image from image Set256 satisfactorily using either of the denoisers. The PSNR decreases on the first iteration and then oscillates around a level that is 7.5 dB to 10 dB

lower. However, when D_F increases above 1, following the initial dip, the PSNR tends to increase. With $D_F \geq 1.4$, when employing the BM3D denoiser, IDA improves the initial AL-DCT-DD PSNR by approximately 2 dB, the maximum value after 20 iterations being reached with $D_F = 2$. When employing the DnCNN denoiser, IDA improves the AL-DCT-DD PSNR by 4.4 dB after 20 iterations.

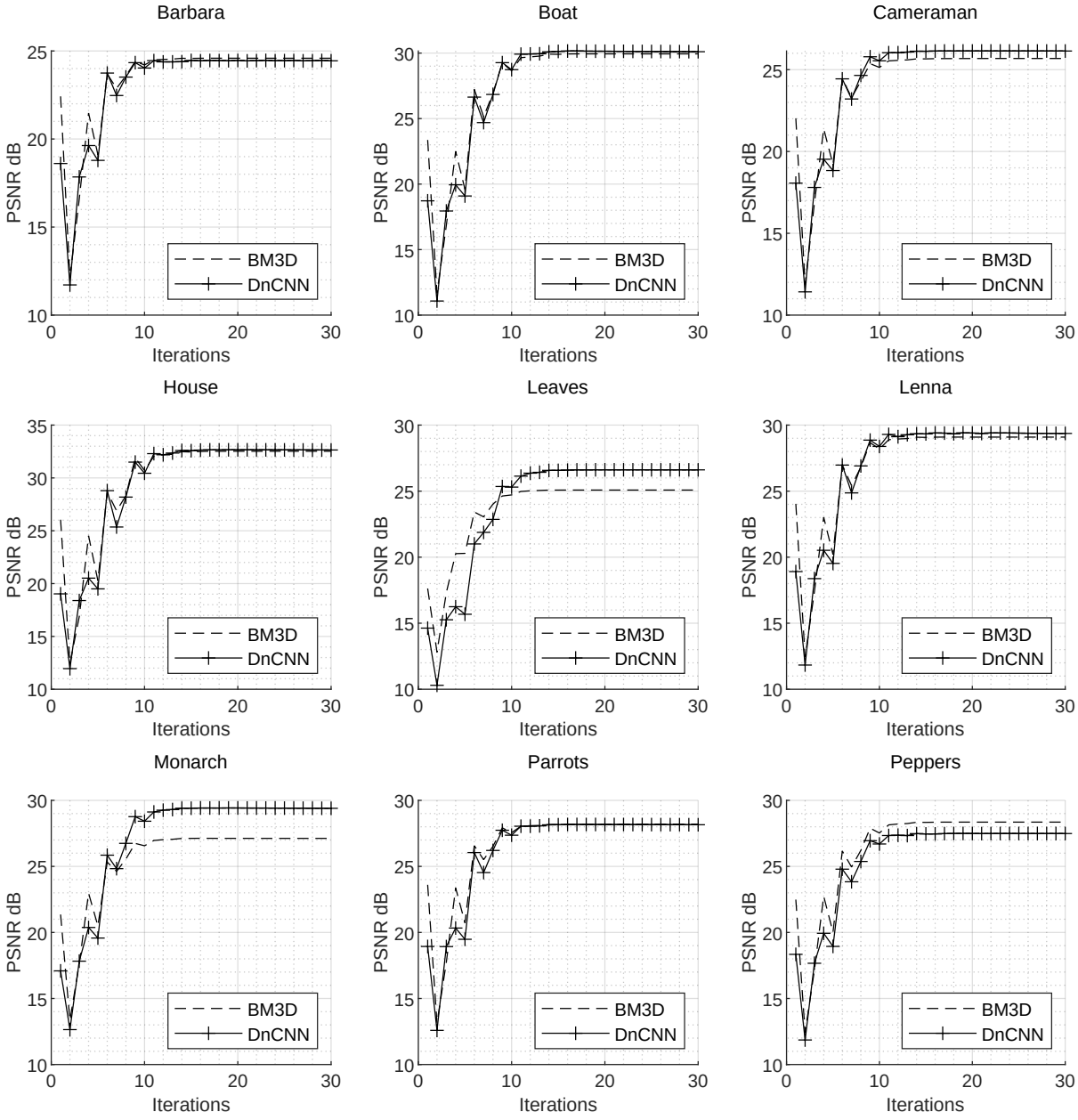


Figure 4.9: Comparison of the PSNR for the 256×256 images reconstructed using AL-DCT-DD-IDA with BM3D and DnCNN denoisers at $C_R = 0.1$.

The PSNR for the images in Set256 reconstructed using IDA with DnCNN and BM3D are shown in figure 4.9. The PSNR for Monarch and Leaves is substantially higher with DnCNN, and slightly better in all other images except for Peppers. Additionally, IDA using the DnCNN denoiser is 4.1 times to 17.8 times faster than with the BM3D denoiser (refer to table 9 in [91] for run-time comparison) and since it achieves superior performance, we use this denoiser in subsequent work.

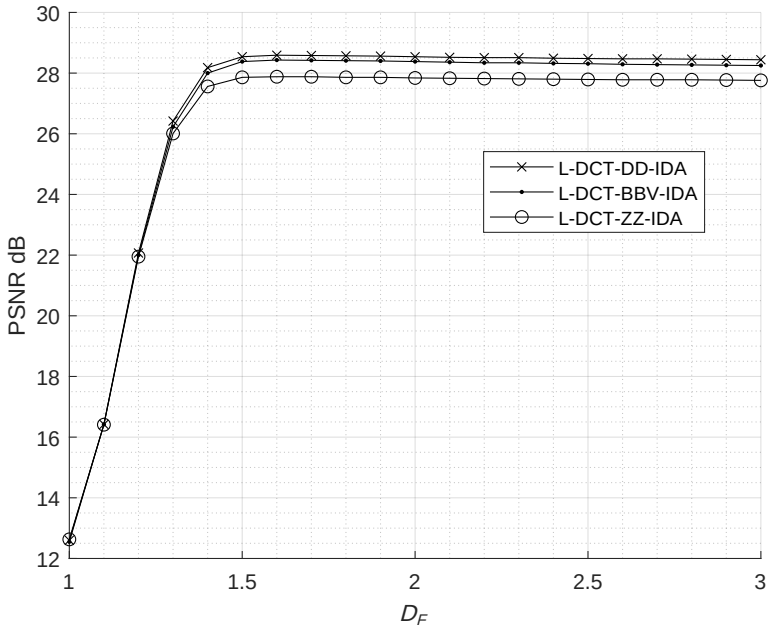


Figure 4.10: PSNR versus D_F for [L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA for the 256×256 image set, with compression ratio $C_R = 0.1$.

Tuning D_F

To find the optimal value of D_F, the PSNR was plotted for D_F between 1.0 and 3.0 for various algorithms, image sets and compression ratios. For example, figure 4.10 shows the PSNR versus D_F plotted for L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA for Set256, with $C_R = 0.1$. D_F = 2.0 was empirically found to be a good compromise value across the compression ratios of interest.

4.6 Empirical investigation

In this section, L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD are first compared empirically when reconstructed directly using the IDCT and using CS reconstruction with the D-AMP and IDA algorithms on the Set256 and Set512 image sets. The CS algorithms were implemented using a DnCNN denoiser as detailed in section 4.5. The two adaptive algorithms are then compared with other published encoder-side and decoder-side adaptive CS algorithms, as well as with full sensing techniques. Finally, IDA is compared with two state-of-the-art non-adaptive algorithms, CREAM [17] and BCS-Net [95]. Since none of the techniques published in the literature were accompanied by source code, the method we used to compare with these published results was to repeat the simulation on the same image test sets and at the same compression ratios.

The simulations were executed on a server equipped with an Nvidia GTX 1080 Ti Graphics Processing Unit (GPU) with 11GB of memory, an Intel Xeon CPU E5-160 v3 clocked at 3.50 GHz, with 32.00 GB of RAM, running MATLAB version 2019a on Windows 10. The D-AMP algorithm was downloaded together with the D-AMP toolbox from [96]. The DnCNN code and

C_R	L-DCT-ZZ		L-DCT-ZZ-D-AMP		L-DCT-ZZ-IDA	
	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
0.01	20.09	0.5187	4.82	0.0000	20.79	0.5831
0.02	21.68	0.5955	5.86	0.0000	22.65	0.6739
0.04	23.40	0.6806	6.71	0.1530	24.87	0.7687
0.10	26.44	0.8096	19.84	0.7432	28.42	0.8733
0.20	29.37	0.8889	28.89	0.8828	31.95	0.9281
0.30	31.95	0.9308	31.06	0.9208	34.85	0.9563
0.40	34.38	0.9562	31.96	0.9282	37.31	0.9725
0.50	36.93	0.9728	33.41	0.9568	39.69	**0.9823**
0.01 - 0.04	21.72	0.5983	5.80	0.0510	22.77	0.6752
0.10 - 0.50	31.81	0.9117	29.03	0.8864	34.45	0.9425
C_R	AL-DCT-BBV		AL-DCT-BBV-D-AMP		AL-DCT-BBV-IDA	
	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
0.01	20.09	0.5187	4.82	0.0000	20.79	0.5831
0.02	21.68	0.5955	5.86	0.0000	22.65	0.6739
0.04	23.00	0.6629	6.71	0.1661	24.38	0.7436
0.10	26.70	0.7982	15.63	0.6863	28.73	0.8672
0.20	30.54	0.8930	29.36	0.8891	**33.37**	**0.9359**
0.30	33.32	0.9334	30.93	0.9112	**36.42**	**0.9607**
0.40	35.90	0.9561	33.15	0.9403	**39.01**	**0.9733**
0.50	38.29	0.9702	33.88	0.9459	**41.19**	0.9810
0.01 - 0.04	21.59	0.5924	5.79	0.0554	22.61	0.6669
0.10 - 0.50	32.95	0.9102	28.59	0.8746	**35.74**	0.9436
C_R	AL-DCT-DD		AL-DCT-DD-D-AMP		AL-DCT-DD-IDA	
	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
0.01	20.32	0.5271	5.32	0.0000	**21.05**	**0.5918**
0.02	22.02	0.6085	5.64	0.0087	**23.09**	**0.6880**
0.04	23.99	0.6998	7.76	0.2146	**25.43**	**0.7810**
0.10	27.07	0.8173	20.38	0.7697	**29.13**	**0.8784**
0.20	30.48	0.8953	29.92	0.8944	33.14	0.9308
0.30	33.46	0.9350	31.19	0.9228	36.13	0.9563
0.40	36.10	0.9588	32.67	0.9324	38.78	0.9722
0.50	38.58	0.9735	33.72	0.9524	40.97	0.9811
0.01 - 0.04	22.11	0.6118	6.24	0.0744	**23.19**	**0.6869**
0.10 - 0.50	33.14	0.9160	29.57	0.8943	35.63	**0.9438**

Table 4.6: L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD with IDCT, D-AMP and IDA reconstruction on the 256×256 image set, using the DnCNN denoiser. Maximum PSNR and SSIM values for all compression ratios are in bold.

C_R	L-DCT-ZZ		L-DCT-D-AMP		L-DCT-IDA	
	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
0.01	22.85	0.5343	4.99	0.0000	23.53	0.5753
0.02	24.23	0.5925	5.71	0.0049	24.97	0.6327
0.04	25.70	0.6685	6.97	0.1738	26.63	0.7102
0.10	28.63	0.7940	19.67	0.6853	29.60	0.8204
0.20	31.62	0.8767	28.25	0.8162	32.63	0.8923
0.30	33.82	0.9164	29.64	0.8629	34.74	0.9273
0.40	35.90	0.9411	30.16	0.8680	36.68	0.9487
0.50	38.06	0.9593	32.31	0.9094	38.73	**0.9643**
0.01 - 0.04	24.26	0.5984	5.89	0.0596	25.04	0.6394
0.10 - 0.50	33.61	0.8975	28.01	0.8284	34.47	0.9106
C_R	AL-DCT-BBV		AL-DCT-BBV-D-AMP		AL-DCT-BBV-IDA	
	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
0.01	22.85	0.5343	4.99	0.0000	23.53	0.5753
0.02	24.23	0.5925	5.71	0.0049	24.97	0.6327
0.04	25.74	0.6636	7.10	0.1292	26.74	0.7035
0.10	29.18	0.7923	13.96	0.5735	30.30	0.8216
0.20	32.72	0.8805	22.77	0.8033	**33.85**	**0.8976**
0.30	35.26	0.9191	27.63	0.8533	**36.32**	**0.9303**
0.40	37.36	0.9419	31.03	0.8860	**38.25**	**0.9492**
0.50	39.36	0.9571	33.08	0.9080	**40.02**	0.9612
0.01 - 0.04	24.27	0.5968	5.93	0.0447	25.08	0.6372
0.10 - 0.50	34.78	0.8982	25.69	0.8048	**35.75**	**0.9120**
C_R	AL-DCT-DD		AL-DCT-DD-D-AMP		AL-DCT-DD-IDA	
	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
0.01	23.09	0.5417	5.06	0.0000	**23.81**	**0.5837**
0.02	24.60	0.6042	5.83	0.0019	**25.35**	**0.6434**
0.04	26.33	0.6826	7.36	0.1784	**27.13**	**0.7170**
0.10	29.40	0.8000	17.20	0.6771	**30.31**	**0.8227**
0.20	32.50	0.8797	27.14	0.8156	33.39	0.8938
0.30	34.94	0.9179	30.00	0.8556	35.67	0.9269
0.40	37.08	0.9408	30.68	0.8741	37.70	0.9467
0.50	39.19	0.9595	33.21	0.9083	39.77	0.9634
0.01 - 0.04	24.67	0.6095	6.08	0.0601	**25.43**	**0.6480**
0.10 - 0.50	34.62	0.8996	27.64	0.8261	35.37	0.9107

Table 4.7: L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD with IDCT, D-AMP and IDA reconstruction on the 512×512 image set, using the DnCNN denoiser. Maximum PSNR and SSIM values for all compression ratios are in bold.

models were downloaded from https://github.com/ricedsp/prDeep and require the MatConv-Net package from https://www.vlfeat.org/matconvnet/. DnCNN was trained using 400 images from the Berkeley Segmentation Dataset (BSDS500) [44], cropped to 180×180 pixels.

Algorithm	Type	PSNR dB	L-DCT-ZZ PSNR dB	AL-DCT-BBV PSNR dB	AL-DCT-DD PSNR dB
ABCS-TVAL3 [84]	ES-CS	30.82	37.38	**39.35**	38.68
JRW-BCS [65]	ES-CS	33.50	33.25	**34.26**	34.19
ABCS-Zhang [64]	DS-CS	29.21	30.06	**33.73**	31.21
Proposed in [66]	DS-CS	31.08	32.88	34.16	**34.44**
JRW-BCS-Sol2-EB [65]	DS-CS	34.08	33.25	34.19	**34.26**
JRW-BCS-Sol2-VB [65]	DS-CS	34.04	33.25	34.19	**34.26**
ABCS-Wang [75]	ES-FS	28.36	35.22	**36.85**	36.54
Var-reg OP3 [76]	ES-FS	29.18	30.39	31.02	**31.64**
Proposed [79]	ES-FS	27.08	31.48	**32.93**	32.41
ABCS-Canh [80]	ES-FS	32.16	34.37	35.59	**35.62**
ABCS-SF-D [81]	ES-FS	32.64	32.26	**33.84**	33.19
JND [82]	ES-FS	28.35	32.13	33.34	**33.36**

Table 4.8: Comparison of L-DCT-ZZ, AL-DCT-DD and AL-DCT-BBV with ABCS results in the literature derived on the image sets in the literature, using the compression ratios in the referenced publication. Best results are highlighted in bold. The underlined results are the only inferior L-DCT-ZZ results.

4.6.1 Comparing the deterministic algorithms using direct and CS reconstruction

Tables 4.6 and 4.7 present the reconstruction results of the baseline linear L-DCT-ZZ algorithm and the two adaptive algorithms presented here: AL-DCT-BBV and AL-DCT-DD. The results of the original D-AMP algorithm and the IDA algorithm used as a post-processing step, are also reported.

When no CS reconstruction post-processing was utilised, AL-DCT-DD achieved the best results at all compression ratios $(0.01 - 0.5)$ on Set256. The average PSNR performance improved by an average of 0.97 dB over the baseline L-DCT-ZZ algorithm across all compression ratios, and average SSIM improved from 0.7941 to 0.8019. This improvement suffered no penalty in reconstruction execution time, as shown in table 4.12. On Set512, average PSNR increased by 0.79 dB and the average SSIM increased from 0.7854 to 0.7908, respectively. However, AL-DCT-BBV achieved a marginally higher PSNR increase for compression ratios in the range $0.1 - 0.5$ than AL-DCT-DD.

The original D-AMP algorithm failed to reconstruct images at very low compression ratios $(0.01 - 0.04)$. This was expected, as the algorithm was not designed to handle images that have been sensed using a deterministic sensing matrix. Interestingly, D-AMP achieved some success in reconstructing images at higher compression ratios, but the quality degraded in all cases.

Algorithm	Type	Image Set	[37]	[39]	[36]
		Reconstruction algorithm	PSNR dB	PSNR dB	PSNR dB
StatACS [83]	ES-CS	IWT	31.57		
InVDS-WT [85]	ES-CS	DAMP		34.72	
InVDS-DCT [85]	ES-CS	DAMP		34.83	
InVDS-HT [85]	ES-CS	DAMP		34.40	
JRW-BCS-Sol1-EB [65]	ES-FS	NESTA			34.59
JRW-BCS-Sol1-VB [65]	ES-FS	NESTA			35.41
JRW-BCS-Sol1-SB [65]	ES-FS	NESTA			35.48
JRW-BCS-Sol2-SB [65]	DS-CS	NESTA			34.53
AL-DCT-BBV-IDA-DnCNN	ES-CS	IDA-DnCNN	32.12	35.06	35.12
AL-DCT-DD-IDA-DnCNN	ES-CS	IDA-DnCNN	32.30	34.41	35.07
L-DCT-THB	ES-FS	IDCT			39.03
L-DCT-THI	ES-FS	IDCT			40.87

Table 4.9: Comparison of AL-DCT-BBV-IDA and AL-DCT-DD-IDA using the DnCNN denoiser with adaptive CS algorithms in the literature using the indicated reconstruction algorithm. The results in each PSNR column were derived using the image set in the cited publication, using the compression ratios in the referenced publication.

IDA successfully improved the performances of all three algorithms. Average PSNR improved by 2.94 dB and 1.54 dB across all compression ratios for Set256 and Set512, respectively. Similarly, average SSIM improved from 0.7941 to 0.8475 for Set256 and from 0.7854 to 0.8122 for Set512.

It is interesting to note that AL-DCT-BBV-IDA achieved the best performance with the higher compression ratios ($0.10 - 0.50$), whilst AL-DCT-DD-IDA performed best with the low compression ratios ($0.01 - 0.04$). This has interesting implications when using these algorithms in video transmission systems. Such systems usually employ a group of picture structure [97] wherein key frames are sensed with a higher compression ratio than and non-key frames. Hence, key-frames could be sensed adaptively using AL-DCT-BBV whereas non-key-frames could use AL-DCT-DD.

Visual Quality

Figure 4.11 shows the quality improvements achieved with the real-time ZZF and THF deblocking, non-real-time BM3D deblocking and IDA CS reconstruction, on L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD, when compressively sensing and reconstructing the *Lena* image in Set256. Details from the same image are show in figure 4.12. Some blocking is still visible in the L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD images on the left of the two figures. ZZF, THF and BM3DF (the BM3D denoiser used as a deblocking filter) remove blocking, but leave some ringing artefacts, that arise because of the low-pass nature of the L-DCT-ZZ measurements. IDA reconstruction of the BBV algorithm for the Lena image in Set256, removes all ringing and increases PSNR by 2.52 dB. SSIM increases from 0.8274 to 0.8879.

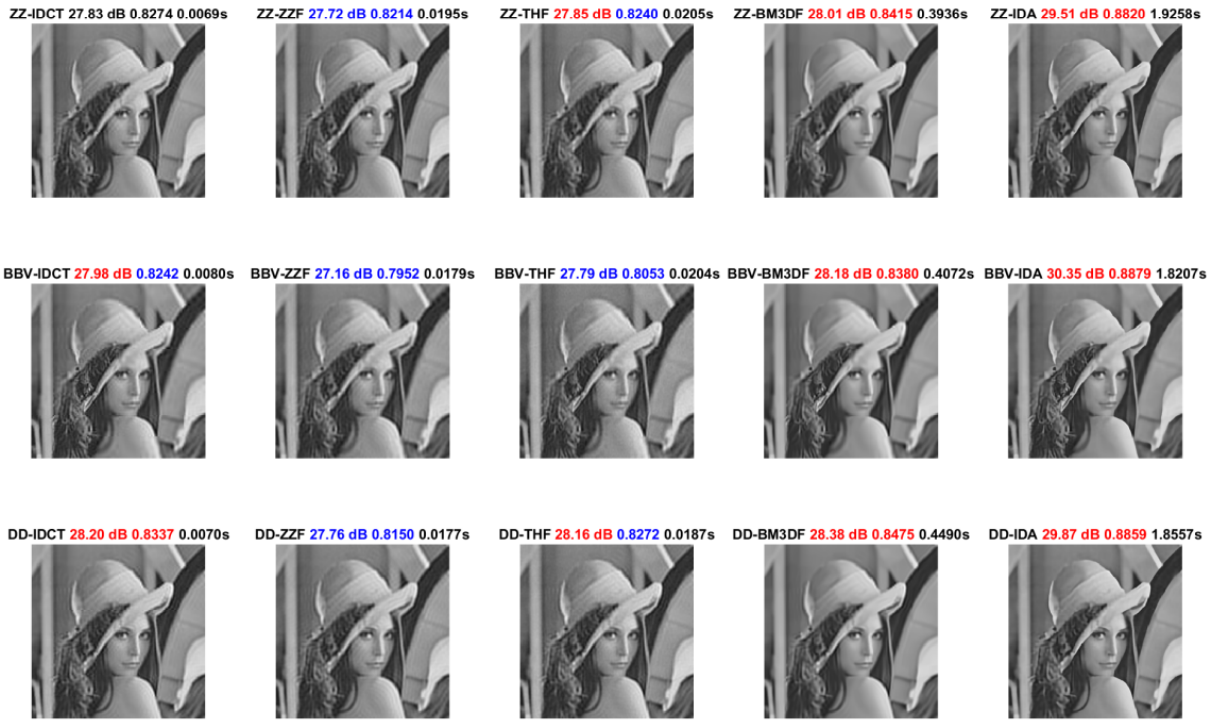


Figure 4.11: Visual Quality of the [A]L-DCT-{ZZ|BBV|DD}-[ZZF|THF|BM3DF|IDA] on the *Lena* image in Set256 at a compression ratio of $C_R = 0.1$. Blue PSNR and SSIM values are worse than those of ZZ-IDCT in the top left corner. Red values are better. Decoding time is reported in seconds.

Figure 4.12: Details from the corresponding images in figure 4.11.

4.6.2 Comparison with adaptive CS results in the literature

L-DCT-ZZ and the two adaptive algorithms achieved state-of-the art performance, in PSNR and SSIM terms, across many adaptive CS algorithms proposed in the literature. Table 4.8 summarizes the PSNR results, where our real-time algorithms perform better than the ES-CS,

DS-CS and ES-FS BCS algorithms in the literature. Each row in table 4.8 reports the PSNR result of a published ABCS scheme and those of our algorithms on the same image set.

Only JRW-BCS, JRW-BCS-Sol2-EB and JRW-BCS-Sol2-VB from [65] and ABCS-SF-D [81] achieve better results than L-DCT-ZZ, with our AL-DCT-BBV and AL-DCT-DD algorithms achieving the best performance in all cases.

Table 4.9 compares AL-DCT-BBV and AL-DCT-DD, reconstructed using IDA with the DnCNN denoiser, against seven adaptive algorithms whose performance cannot be matched by our real-time algorithms. All our CS reconstructed algorithms outperformed StatACS and JRW-BCS-Sol2-SB in terms of PSNR. StatACS is not block-based, although it can be reconstructed in real time. AL-DCT-BBV-IDA outperformed the three InVDS algorithms in [65].

None of our CS reconstructed algorithms exceed the PSNR performance of the full sensing, Sol1-VB and Sol1-SB variants presented in [65]. A comparison with the full sensing THB and THI algorithms is more appropriate. In this case, THB achieves a PSNR of 39.03 dB and THI 40.87 dB, exceeding the best Sol1 variant by 3.55 dB and 5.39 dB, respectively.

Figure 4.13: Set12: The first seven images from the left have a resolution of 256×256 and the last five are 512×512. The fourth and fifth images from the left are *Starfish* and *Monarch*.

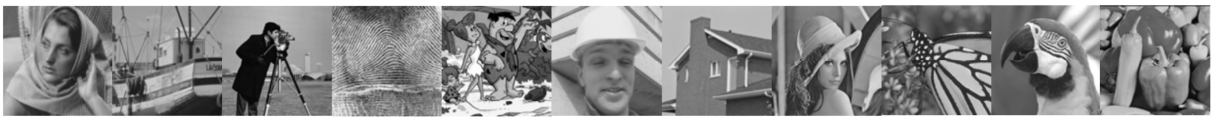


Figure 4.14: Set11: The images have a resolution of 256×256, except for the fourth and fifth from the left, *Fingerprint* and *Flintstones*, that have a resolution of 512×512 pixels.

C_R	0.1		0.2		0.3		0.4		Average	
Algorithm	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
CREAM	27.31	0.8206	31.67	0.9014	34.14	0.9318	36.11	0.9510	32.31	0.9012
L-DCT-ZZ	26.92	0.8030	29.67	0.8795	31.82	0.9182	33.71	0.9429	30.53	0.8859
AL-DCT-BBV	27.28	0.7982	30.64	0.8823	33.34	0.9226	35.57	0.9459	31.71	0.8873
AL-DCT-DD	27.67	0.8131	30.78	0.8865	33.19	0.9223	35.22	0.9446	31.72	0.8916
L-DCT-ZZ-IDA	28.41	<u>0.8506</u>	31.61	0.9105	33.82	<u>0.9385</u>	35.70	<u>0.9571</u>	32.39	0.9142
AL-DCT-BBV-IDA	<u>28.85</u>	0.8486	**32.73**	**0.9158**	**35.43**	**0.9433**	**37.48**	**0.9591**	**33.62**	**0.9167**
AL-DCT-DD-IDA	**29.28**	**0.8575**	<u>32.68</u>	<u>0.9128</u>	<u>34.89</u>	<u>0.9385</u>	<u>36.76</u>	0.9554	<u>33.40</u>	<u>0.9161</u>

Table 4.10: Comparing CREAM PSNR and SSIM with L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA with the DnCNN denoiser, with $D_F = 2.0$ and 15 iterations, on Set12. Maximum PSNR/SSIM values are in bold and second largest underlined.

C_R	0.01		0.03		0.05		0.1	
Algorithm	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
BCS-Net	20.88	0.5505	24.47	0.7087	26.04	0.7723	29.43	0.8676
L-DCT-ZZ	20.33	0.5072	22.82	0.6369	24.60	0.7260	27.06	0.8193
AL-DCT-BBV	20.33	0.5072	22.91	0.6424	24.17	0.7031	27.29	0.8112
AL-DCT-DD	20.56	0.5163	23.27	0.6519	25.03	0.7366	27.68	0.8272
L-DCT-ZZ-IDA	20.99	0.5615	24.09	0.7153	25.81	0.7864	28.78	0.8721
AL-DCT-BBV-IDA	20.99	0.5615	24.09	0.7153	25.58	0.7676	29.09	0.8676
AL-DCT-DD-IDA	**21.25**	**0.5717**	**24.48**	**0.7240**	**26.30**	**0.7934**	**29.45**	**0.8769**
C_R	0.2		0.3		0.4		Average	
Algorithm	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
BCS-Net	33.06	0.9283	35.60	0.9554	36.70	0.9662	29.45	0.8213
L-DCT-ZZ	30.17	0.8953	32.70	0.9340	35.01	0.9569	27.53	0.7822
AL-DCT-BBV	31.08	0.8980	33.90	0.9363	36.29	0.9563	28.00	0.7792
AL-DCT-DD	31.16	0.9013	34.06	0.9376	36.53	0.9591	28.33	0.7900
L-DCT-ZZ-IDA	32.29	0.9275	34.90	0.9544	37.05	**0.9698**	29.13	0.8267
AL-DCT-BBV-IDA	**33.29**	**0.9316**	**36.11**	**0.9567**	**38.35**	0.9694	29.64	0.8242
AL-DCT-DD-IDA	33.27	0.9296	35.95	0.9543	38.23	0.9694	**29.85**	**0.8313**

Table 4.11: Comparing BCS-Net PSNR and SSIM with L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD reconstructed using IDA with the DnCNN denoiser, with $D_F = 2$ and 15 iterations, on Set11. Maximum PSNR values are in bold and second largest underlined.

4.6.3 Comparison with two state-of-the-art, non-adaptive algorithms

In this section, we compare our algorithms with CREAM [17] and BCS-Net [95]. Both these algorithms are GPU assisted. Since the code of CREAM and BCS-Net are not available, we report our results on the same image sets used in both works respectively.

The comparison with CREAM was carried out using Set12 set shown in figure 4.13. The results in table 4.10 show that the real-time AL-DCT-DD algorithm achieved a better PSNR than CREAM at $C_R = 0.1$, but CREAM was superior to our real-time algorithms in all other cases. When our adaptive algorithms were reconstructed using IDA, the situation reversed, with both always performing better. The best PSNR and SSIM results were achieved by AL-DCT-BBV-IDA followed by AL-DCT-DD-IDA for $D_F = 2$ with 15 iterations.

L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD were reconstructed in less than 40 ms. AL-DCT-BBV-IDA reconstructs an image in 5.2 s on average, with 15 iterations at a compression ratio of 0.4. The reported reconstruction time of CREAM on two 256×256 images, *Starfish* and *Monarch* in Set12 (figure 4.13), is around 5.5 s on their platform.

Comparing our algorithms with BCS-Net on the images in Set11 (figure 4.14), table 4.11 shows that the best average PSNR and SSIM results were obtained by AL-DCT-DD-IDA with $D_F = 2$ and 15 iterations. AL-DCT-BBV-IDA achieves better results at higher compression ratios. The average reconstruction time over all compression ratios was approximately 3.5 seconds for 15 iterations. The reconstruction time for a compression ratio of 0.1 was approximately

Algorithm	L-DCT-ZZ	AL-DCT-		L-DCT-ZZ-IDA	AL-DCT-	
		BBV	DD		BBV-IDA	DD-IDA
C_R	Time s					
0.01	0.0069	0.0069	0.0067	0.69	0.69	0.69
0.02	0.0080	0.0074	0.0070	0.79	0.79	0.80
0.04	0.0067	0.0069	0.0066	1.05	1.02	1.07
0.10	0.0067	0.0071	0.0067	1.86	1.75	1.84
0.20	0.0067	0.0068	0.0066	3.20	2.89	3.11
0.30	0.0067	0.0069	0.0065	4.42	4.09	4.38
0.40	0.0068	0.0069	0.0067	5.89	5.38	5.72
0.50	0.0067	0.0067	0.0066	7.86	6.50	7.08

Table 4.12: Comparison of reconstruction times for L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD using 2D-IDCT and IDA reconstruction with the DnCNN denoiser on the 256×256 image set, $D_F = 2$ and 20 iterations.

2.37 seconds for 15 iterations.

4.6.4 Complexity analysis

Block-based L-DCT-ZZ and AL-DCT-DD acquire M measurements using $M \times B^2$ operations, whereas full image CS algorithms require $M \times (H \times W)$ operations. AL-DCT-BBV requires $M_{BBV} + (M - M_{BBV}) \times B^2$ operations. These analogue computations occur in the optical domain on single-pixel and multi-pixel cameras. The adaptive algorithms also need to compute the number of phase-2 measurements per block in the digital domain. These total $n_B \times n_S$ computations in the case of AL-DCT-BBV and $n_B \times B^2$ in the case of AL-DCT-DD.

The direct reconstruction of L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD using the inverse 2D DCT requires M inversions of a $B \times B$ matrix. Implemented as a separable inverse transform, 2D-IDCT requires $\mathcal{O}(B^2 \log B)$ operations. Therefore, IDCT reconstruction requires $\mathcal{O}(MB^2 \log B)$ operations.

With reference to figure 4.6 and equation (4.12), D-AMP has a computational complexity dominated by the three matrix multiplications and the denoiser computation and is of order $\mathcal{O}(MN^2 + 2NM^2) + \mathcal{O}(D_{\widehat{\sigma^t}})$, where $\mathcal{O}(D_{\widehat{\sigma^t}})$ is the complexity of the denoiser.

With reference to equation (4.12), DAMP-D has the same order of complexity because it merely divides the Onsager term by α every iteration. However, equation (4.13) shows that IDA avoids the computation of the Onsager term altogether and has complexity $\mathcal{O}(MN^2 + NM^2) + \mathcal{O}(D_{\widehat{\sigma^t}})$. Simulation results show that D-AMP and DAMP-D reconstruct a 256×256 image in approximately 2.4 s when using a DnCNN denoiser, whereas IDA-DnCNN requires 1.8 s. These results are in agreement with the theoretical complexity results.

On our platform, the L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD algorithms reconstruct 512×512 images in less than 30 ms. The improvement in performance with CS reconstruction comes at the expense of reconstruction time. Table 4.12 compares the reconstruction times

of these algorithms with and without IDA post-processing using the DnCNN denoiser on the 256×256 image set.

4.7 Chapter summary

Two adaptive algorithms, AL-DCT-BBV and AL-DCT-DD, have been described that adapt the number of deterministic 2D-DCT measurements collected from block-based compressive image sensors. Both techniques have been shown to estimate the number of transform coefficients per block that correlates well with the number of transform coefficients estimated by the full 2D DCT analysis. AL-DCT-BBV and AL-DCT-DD achieve state-of-the art performance in adaptive block CS of images that can be used in real-time applications without requiring any form of GPU support, such as image and video wireless sensor systems. The algorithms reconstruct the compressively sensed images in real time, for example, less than 8 ms and 30 ms for 256×256 and 512×512 images, respectively.

The original D-AMP algorithm frequently fails to reconstruct or improve our algorithms using deterministic sensing matrices. A block diagram representation of the D-AMP algorithm has been derived and used to interpret D-AMP as a feedback system with integral control. A modified version, DAMP-D, which damps the Onsager term with a factor D_F and a simplified version, called the IDA algorithm, has been derived that performs better when reconstructing our deterministic CS algorithms. The IDA algorithm simplifies the Onsager term to $\mathbf{z}^{\mathbf{t}}/D_F$ and achieves a considerable reduction in computational complexity and better results than D-AMP and DAMP-D. Both modified algorithms can be used as a post-processing technique on the AL-DCT-BBV and AL-DCT-DD algorithms. Setting $D_F = 2.0$ has been empirically found to be close to optimal for the 256×256 and 512×512 image sets used in our simulations.

The two adaptive algorithms reconstructed using IDA with a DnCNN denoiser outperform two state-of-the-art algorithms in terms of PSNR and SSIM. AL-DCT-BBV-IDA outperforms the PSNR performance of the GPU-assisted CREAM by 1.31 dB, and SSIM reaches 0.9167 to CREAM's 0.9012. AL-DCT-DD-IDA outperforms the DNN BCS-Net PSNR by 0.40 dB and its SSIM reaches 0.8313 to BCS-Net's 0.8213. Even the non-CS reconstructed, real-time algorithms perform close to CREAM and BCS-Net, with low to moderate PSNR penalties. For example, AL-DCT-DD comes within 0.59 dB of the CREAM PSNR, and its SSIM is 0.8916 to CREAM's 0.9012. It also comes within 1.12 dB of BCS-Net PSNR and its SSIM is 0.7900 to BCS-Net's 0.8676. This shows that AL-DCT-DD is suitable for real-time compressive image and video sensing.

We also note that by utilising DCT measurements, the proposed algorithms can leverage the well-established source and channel coding techniques developed for hybrid DPCM/DCT encoding standards.

Chapter 5

Super resolution of images for compressive sensors

5.1 Introduction

Advances in image display technology, increasing spatial and temporal resolution in particular, have driven the interest in super resolution techniques, to up-sample material originally created with lower resolution. We propose another use for super resolution. Inspired by the compressive sensing model which seeks to lower the sampling rate, our approach is to repurpose super resolution for image compression.

Techniques based around Deep Neural Networks (DNNs) have recently been very successful in Single Image Super Resolution (SISR) and Multiple Image Super Resolution (MISR), and can up-sample images and video in tens of milliseconds. In 2017, Timofte et al. [68] reported that *"most single image Super Resolution (SR) publications use the 91 train images of Yang et al. ... and a bicubic downscaling (imresize from Matlab) to simulate the HR to LR transformation*".

The imresize function in Matlab uses bicubic downscaling defined by Keys in [98], as explained in section 5.3.1, combined with an antialiasing filter. To use trained DNNs from the literature as plug-and-play solutions, without retraining the DNNs it is required to use the same bicubic downscaling operation, even though other downscaling methods, such as Lanczos resampling (section III.2 in [99]), exist.

The fact that the low resolution images are created artificially is seen as a limitation for creating super resolution techniques that up-sample original, low resolution material that was not produced at a higher resolution and then downsampled. However, this is not a limitation when creating compression algorithms that use down-sampling with the equivalent of the Matlab imresize function at the compressive sensor, and then up-sampling using super resolution algorithms at the decoder.

In this chapter, we first review the super resolution literature which is currently dominated by convolutional neural networks. With the aim of leveraging super resolution for compression. We then show how the Matlab imresize function can be implemented on single pixel and multi pixel cameras or focal plane processing sensors. The Shrink algorithm is developed allowing

the bicubic convolutional interpolation filter, used in the imresize function, to be implemented as a convolution operation on the Single Pixel Camera (SPC) and Multi Pixel Camera (MPC) or Focal Plane Processing Sensors (FPPS). We then use super resolution convolutional neural networks to up-sample the compressed images. Results are presented to compare dimensionality reduction performance with that of our super resolution techniques.

Algorithms

The following image compressed sensing algorithms are developed in this chapter.

- Shrink is a dimensionality reduction algorithm that implements anti-aliasing filtering and bicubic downsampling of an image as a 2D convolution operation on an SPC or MPC.
- Shrink-VDSR uses the GPU-accelerated Very Deep Super Resolution (VDSR) Convolutional Neural Network (CNN) to upsample images compressed using the Shrink algorithm.
- Shrink-IMDN uses the GPU-accelerated Information Multi-Distillation Network (IMDN) CNN to upsample images compressed using the Shrink algorithm.

5.2 Related work

Figure 5.1: Block diagram of the proposed JPEG-based interpolative scheme from [100]

In 1993, Zeng and Venetsanopoulos [100] observed that the JPEG standard [54] produced poor results at high compression ratios due to blocking, and proposed a scheme whereby an image is first decimated in the spatial domain, compressed and decompressed using the JPEG algorithm and then interpolated to its original resolution. Their scheme is shown in figure 5.1. They showed that decimating the image using a 2×2 pixel averaging low-pass filter, and interpolating it using a replication filter and a 5×5 Gaussian filter, achieves higher performance with no visible blocking at low compression ratios.

Bruckstein, Elad and Kimmel [101] observed that, since the publication of the JPEG standard, the compression industry had known that preceding transform encoding by down-scaling and then up-scaling the inverse transformed image, gives better reconstruction quality at low compression ratios. They made this observation because the hierarchical mode of the JPEG standard [54] creates a down-scaled image pyramid when transmitting images at low bit rates. The authors then proceed to show how to compute the amount of down-sampling prior to transform encoding to produce optimal performance.

The authors in [102] extended the concept of down-sampling followed by up-sampling to low bit rate video compression. They modified the MPEG-2 reference software encoder and

adaptively down-sample 16×16 sub-image blocks to two or one 8×8 sub-image blocks in the hybrid DPCM/DCT coder. They demonstrated coding gains of more than 1 dB on Common Intermediate Format (CIF) sequences below 1 bit/pixel.

Inspired by the above, we review the literature to assess whether existing super resolution techniques allow us to down-sample an image at a compressive sensor and up-sample it at the decoder in real-time, to compete with classical compressive sensing. The field of single-image and multiple-image super resolution has progressed rapidly recently, driven by convolutional neural networks.

SISR has been studied extensively in the literature. In 2014, Yang et al. [103] classified SISR algorithms according to the image priors that are used to generate high quality images from the low resolution input images. They identified four SISR methods: prediction models that use predefined mathematical formulas without training data; edge based methods that learn priors from edge features; image statistical methods; and patch based methods that learn mapping functions from sub-image patches.

In 2014, Dong et al. proposed a deep CNN for SISR [104]. The proposed Super Resolution CNN (SRCNN) consists of three operations: patch extraction and representation; non-linear mapping; and reconstruction. The $f_1 \times f_1$ patch extraction and representation convolutional filters, operating on the low resolution image upscaled to the desired size using bicubic interpolation, result in $n_1 = 64$ feature maps of the low resolution image. These maps are than processed by $f_2 \times f_2$ convolutional filters and result in $n_2 = 32$ feature maps of the high resolution image. The high resolution output is computed using $f_3 \times f_3$ convolutional filters operating on the high resolution image feature maps. SRCNN was shown to produce the best PSNR in all tested cases, when compared against the state-of-the art algorithms. The $(f_1, f_2, f_3) = (9-1-5)$ SRCNN out-performed the the closest performing technique in the literature and had a 60% lower average execution time.

SRCNN focused the attention of the SISR research community on techniques that use DNNs. A recent survey on the subject, published by Anwar et al. [105] classifies DNN-based SISR techniques based on their most distinguishing features, into nine classes: linear networks; residual networks; recursive networks; progressive reconstruction design; densely connected networks; multi-branch designs; attention based networks; multiple degradation handling networks; and generative adversarial networks.

VDSR, proposed by Kim et al. [106] in 2016, improves on SRCNN by increasing the receptive field, learning the residual image instead of the final image and training one CNN for multiple scale factors. In contrast with SRCNN, VDSR consists of 20 layers instead of 3, and the receptive field is 41×41 instead of 9×9. The CNN consists of a first layer that is $41 \times 41 \times 64$ and operates on the lower resolution image interpolated to the required higher resolution, $d-2$ convolutional layers consisting of 64 filters of size $3 \times 3 \times 64$ that operate on the 64 feature maps of the previous layer, and a final convolutional layer consisting of one filter of size $3 \times 3 \times 64$ that outputs the final residual image. The output image is produced by adding the residual image to the low resolution image interpolated to the higher resolution

using bicubic interpolation. VDSR achieves 0.87 dB better PSNR than SRCNN on Set5 [106] at a scale factor of $\times 2$.

SRCNN and VDSR upscale the low resolution image to the required high resolution using bicubic interpolation and then process the higher resolution frames. In 2016, Shi et. al [86] proposed an Efficient Sub-Pixel Convolution Network (ESPCN) that processes the low resolution frames until the final convolutional layer that reconstructs the high resolution image using sub-pixel convolution. Additionally, ESPCN is very shallow. An ESPCN with only 3 convolutional layers achieved state of the art performance, beating the previously SOTA PSNR result by 0.15 dB at scale factor $\times 3$ on Set14 [107], but executing an order of magnitude faster.

The Information Distillation Network (IDN) proposed by Hui et al. [108] adopts a similar strategy to ESPCN and computes the convolution operation on frames that are the same size as the low resolution image, before using transposed convolution in the final step to compute a residual image of the required higher resolution. The distinguishing feature is that IDN uses distillation blocks that divide the features of the previous layer into two, retains one half and further process the remaining features. When published in 2018, IDN claimed SOTA PSNR performance but its main advantage was its speed, executing five times faster than VDSR. In 2019, Hui et al. updated IDN, improving the image distillation blocks, to produce the IMDN [109].

The interest in image and video super resolution has stimulated the NTIRE super resolution challenge that has been held annually since 2017 [68], [110], [111], [112]. Though not mandated, most of the entries to the challenges use convolutional neural networks or generative adversarial networks.

The first NTIRE 2017 challenge had two tracks: bicubic downscaling and unknown downscaling [68]. Each track had three competitions, in x2, x3 and x4 downscaling. Twenty teams participated in the final with the winning entry achieving an improvement in PSNR over the bicubic interpolation baseline of 3.92 dB, 2.91 dB and 2.44 dB for x2, x3 and x4 bicubic upscaling respectively. The authors of the paper opine *"As expected, the unknown downscaling track is more challenging than the bicubic one and this is reflected by the relatively lower PSNR (up to 1 dB for the winners) of the results"*. In our application, it is important to reduce the challenge of upscaling as much as possible and it is therefore justifiable to focus on bicubic downsampling.

The NTIRE 2018 competition contained only one classical bicubic competition with x8 downscaling [110]. The wining entry improved the PSNR and SSIM of the DIV2K test set [68] by 1.752 dB and 0.0701 over the Bicubic baseline.

The third edition of the competition, NTIRE 2019, focused on video super resolution [111]. A new video dataset, consisting of 400 videos with 100 frames of 720 x 1280 resolution, was implemented. Of the two challenge tracks, the *Clean* track required entrants to upscale video downscaled x4 using the bicubic imresize function. The winning entry increased the PSNR and SSIM performance by 5.31 dB and 0.1163 over the baseline bicubic upscaling function.

NTIRE 2020 consisted of 12 image and 3 video challenges, of which two were image x4

super resolution challenges that address real-world super resolution without the availability of paired low and high resolution images for training [112]. The first challenge concerned x4 super resolution of the DIV2K dataset images that underwent an undisclosed degradation operator typical of low-end devices. The second, concerned the super resolution of iPhone3 images without the availability of the ground truth images, that were judged by a human perception study.

Whereas the previous work of Zeng et al. [100], Bruckstein et al. [101] and Nguyen et al. [102], inspire us to study low-complexity image and video sensors using downscaling, the extensive literature on single image super resolution gives us confidence that we can upscale the images in real-time and with good PSNR and SSIM performance.

5.3 Methodology

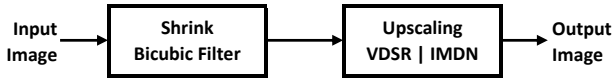


Figure 5.2: The Shrink algorithm implements bicubic interpolation at the compressive sensor to compress the image that is then super resolved using state-of-the-art super resolution algorithms such as IMDN [109].

We deploy downsampling and super resolution as an image compression scheme as show in figure 5.2. The SPC, MPC or FPPS simply downsamples the image which is then upsampled at the decoder using one of the many schemes that have been proposed in the super resolution literature.

Low-energy considerations require a simple procedure to generate the low resolution image. The simplest strategy is to collect all the photons falling on a square element, which is equivalent to summing the values of $n \times n$ pixels when studying SR compression schemes with super resolution factor $S_F = n$. This can be achieved by using a lower-resolution camera. If we assume the same optics and sensor size, the pixels are smaller in a high resolution camera, and bigger and less numerous in a low resolution camera.

However, when we studied super resolution with images down-sampled using the box interpolation filter to simulate this simplest downsampling scheme, the PSNR results suffered significant deterioration. In the literature [68], the best reported results require the low resolution image to be generated using bicubic interpolation and in section 5.3.1 we show how this can be achieved by collecting measurements from Compressive Sensing (CS) sensors using a 2D bicubic filter kernel implemented as a measurement matrix.

5.3.1 Implementing the bicubic filter on compressive sensors

The Matlab imresize function with bicubic interpolation is used in the literature to create low resolution images which are then super resolved using CNNs. In the default mode, the Matlab imresize 'antialiasing' parameter is set to true and the cubic interpolation kernel is modified to

become an antialiasing and interpolation kernel. Without aliasing, the one dimensional bicubic interpolation kernel consists of four components, as an off-grid pixel is interpolated from four surrounding on-grid pixels.

The cubic convolutional interpolation kernel was derived by Keys [98] for the one dimensional case, but can be applied separately in two dimensions, in which case it is usually referred to as bicubic interpolation. The cubic interpolation kernel is defined in the interval (-2,2) and is zero outside this interval. It is composed of cubic polynomial functions defined in four subintervals (-2,-1), (-1,0), (0,1) and (1,2). The kernel must obey the following conditions: (i) it is symmetric; (ii) it has a value of 0 or 1 at the four integer points; (iii) it is continuous; (iv) it has a continuous first derivative; and (v) deliver third-order precision. Keys showed that the cubic interpolation kernel that satisfies these conditions is given by:

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1 & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2 & 1 < |s| < 2 \\ 0 & 2 < |s| \end{cases}, \tag{5.1}$$

where $s = \frac{(x-x_k)}{h}$, x is the point to interpolate off the image grid, x_k are the four interpolation nodes on the image grid, and h is the sampling increment, which is equal to one when the samples are pixels.

If the Matlab imresize 'antialiasing' parameter is set to false, then Matlab generates an interpolation convolutional kernel with four components that are multiplied with the two pixels either side of the pixel to interpolate. In Matlab, the interpolation convolutional kernel is applied to the image separately, along rows and columns. If the 'antialiasing' parameter is set to true, then the number of kernel components generated increases to $4 \times S_F$ if S_F is even and to $(4 \times S_F - 1)$ if S_F is odd, where S_F is the super resolution factor. Then besides interpolating, the convolution with the surrounding pixels performs as an antialiasing filter as well.

The one-dimensional interpolation convolutional kernels generated for $S_F = 2,3\ldots6$ are shown in figure 5.3. When the Matlab imresize function is called with scale parameter $S = 1/S_F$, $S_F = 2,3,\ldots$, the pixel in the compressed image undergoes antialiase filtering and interpolation from pixels with positions $\mathbf{p_k}$ that are at an integer offset from the interpolated pixel position $\mathbf{p_0}$ if S_F is odd. If S_F is even, then the positions $\mathbf{p_k}$ are at offsets $\pm\frac{(n+1)}{2}$ where $n = 0,1,\ldots$. The one dimensional kernel values are then given by $\mathbf{k_1} = u(\mathbf{p_k})$ where $u()$ is given by equation 5.1.

5.3.2 The Shrink algorithm

To perform bicubic down-scaling on an SPC, MPC or acFPPS, we define the Shrink algorithm. We first calculate the two dimensional kernel $\mathbf{K} = \mathbf{k_1'} \cdot \mathbf{k_1}$ where $\mathbf{k_1'}$ is the transpose of column vector $\mathbf{k_1}$, for the required S_F. We then collect the M inner products of $\mathbf{K}$ with the projected pixels, centered at the compressed pixel positions, from the compressive sensor. The two dimensional kernels $\mathbf{K}$ for $S_F = \{2\ldots6\}$ are shown in figure 5.4. The kernel values are quantized to eight bits in this figure, and some of the smaller values at the edges of the two dimensional

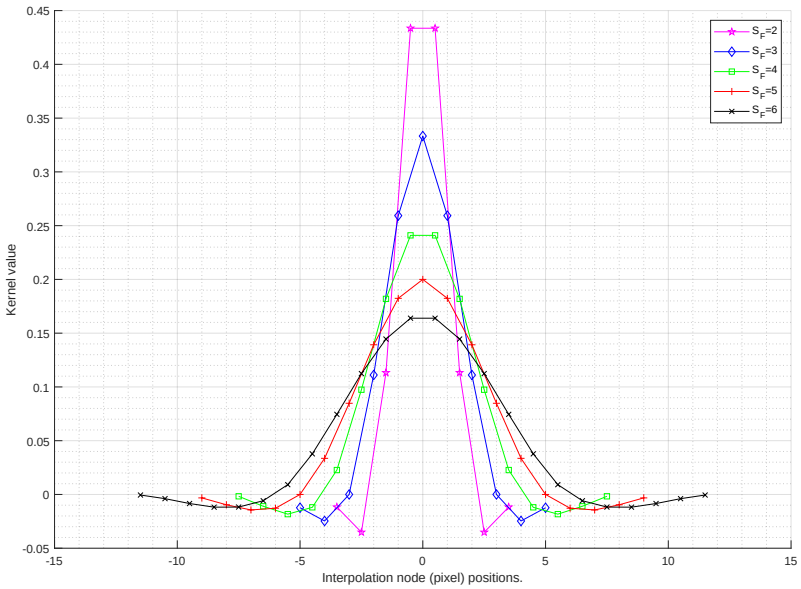


Figure 5.3: 1D imresize bicubic function kernels for $S_F = 2, 3 \ldots 6$.

kernel are quantized to zero. In these cases, the kernel is smaller than $K \times K$ where $K = 4 \times S_F$ if S_F is even and to $(4 \times S_F - 1)$ if S_F is odd, as expected for the non-quantized case.

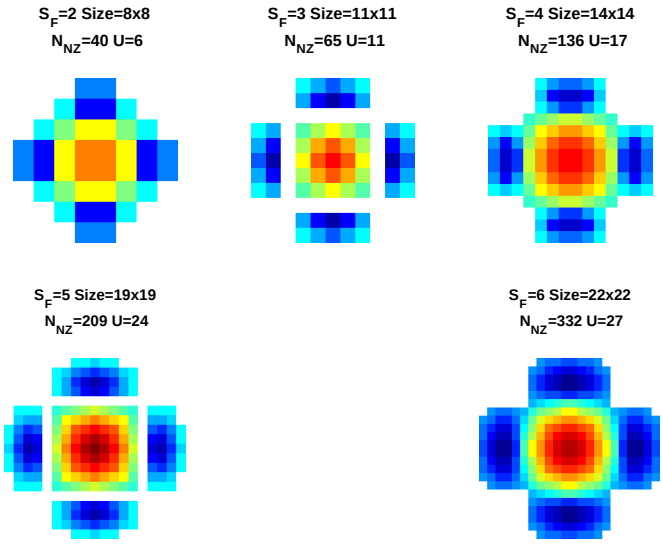


Figure 5.4: 2D Shrink cubic convolutional interpolation kernels: N_{NZ} is the number of non-zero coefficients and U is the number of unique values quantized to 8 Bits including sign bit.

When implemented on an SPC, the negative values are dealt with using the technique described in section 3.2.4. Each measurement requires the modulation of only N_{NZ} Digital Micromirror Devices (DMDs), where N_{NZ} is the number of non-zero-valued quantized coefficients in the cubic convolutional interpolation filter. This represents a very significant reduction in energy since each of the M full-image random CS measurements require modulating all N mirrors, that is $N \times M$ modulations for one image, whereas the bicubic measurements require only $N_{NZ} \times M$ modulations with N_{NZ} significantly lower than N. The bicubic filtering can be also implemented on an MPC or FPPS with the same reduction in energy, because the number of mirror motions is the same.

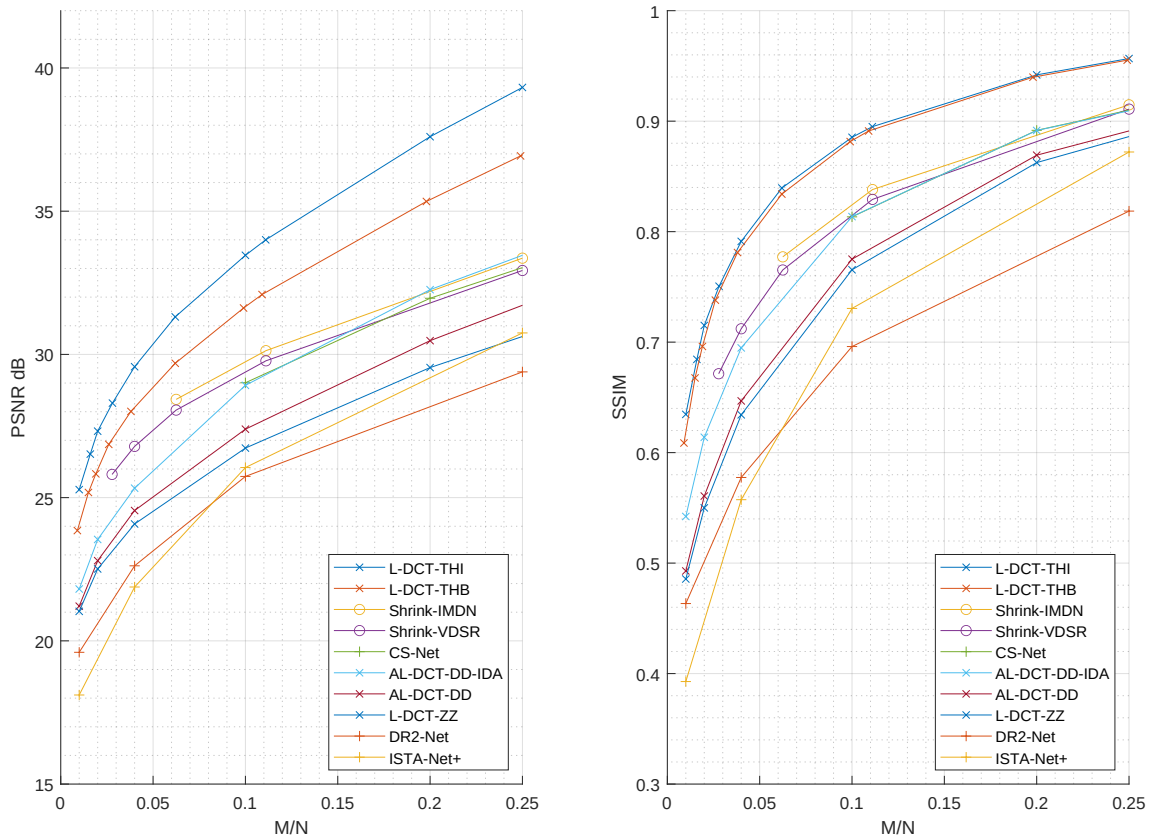


Figure 5.5: (a) PSNR versus M/N for Set14 (b) SSIM versus M/N for Set14, from M/N = 0.01 to 0.25.

5.4 Dimensionality reduction results

In this section we compare the Shrink algorithm super resolved using the CNN-based VDSR and IMDN algorithms and our real-time L-DCT-ZZ, AL-DCT-DD, AL-DCT-THI and AL-DCT-THB algorithms, with three Image CS CNNs; CS-NET [19], DR2-Net [46] and ISTA-Net+ [20]. We compare with these CNN algorithms, because they achieve state-of-the-art performance for real-time CS reconstruction. We also include the IDA reconstructed AL-DCT-DD-IDA algorithm for comparison although this does not execute in real time.

The IMDN [109], VDSR [106], CS-Net [19], DR2-Net [46], and ISTA-Net+ [27] results were obtained from code and trained models made available by the respective authors. The code required three deep neural network environments to be set up namely: Pytorch [113]; MatConvNet [114]; and Tensorflow [115]. All the simulations were executed on a Linux server equipped with an Nvidia GTX 1080 Ti Graphics Processing Unit (GPU). The linear L-DCT-ZZ, AL-DCT-DD, AL-DCT-THB and AL-DCT-THI algorithms were coded and run in Matlab but do not use GPU acceleration. The AL-DCT-DD-IDA algorithm uses the GPU-accelerated Denoising CNN (DnCNN) denoiser.

5.4.1 PSNR

The PSNR versus compression ratio C_R results, for the set of fourteen images (Set14) used in the literature [107], are compared in figure 5.5(a) for M/N = 0.01 to 0.25 and in figure

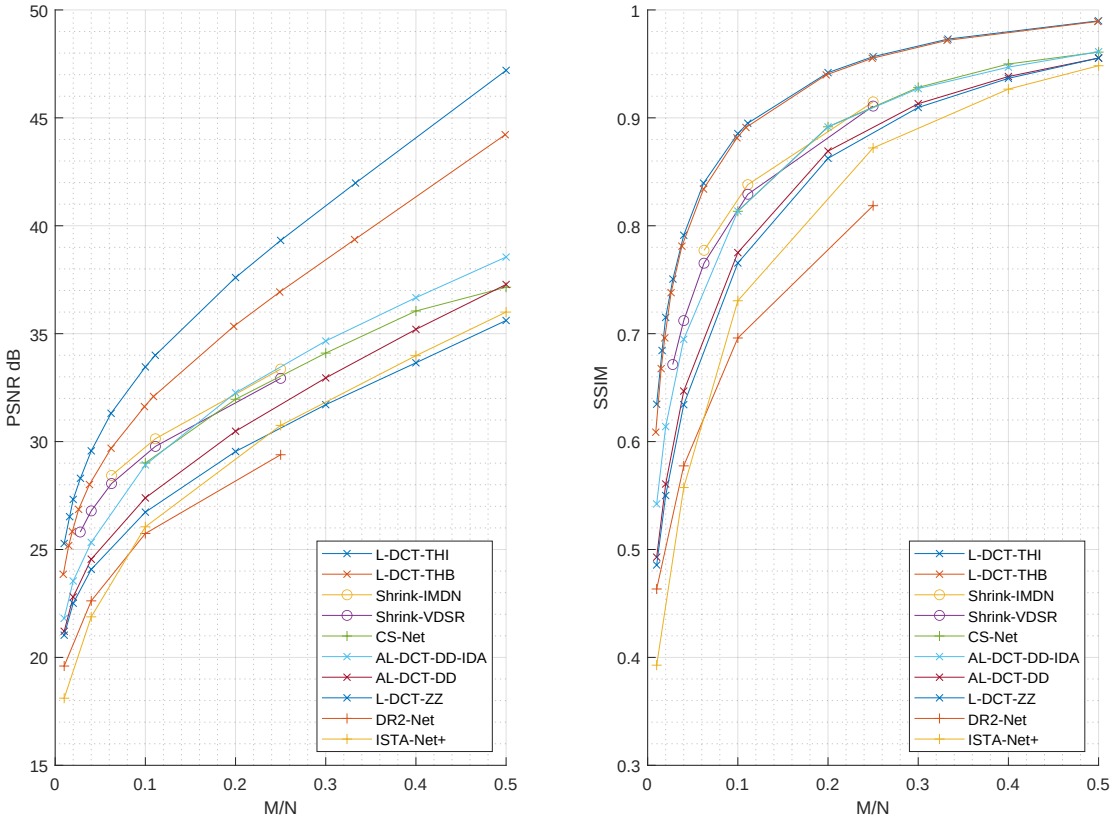


Figure 5.6: (a) PSNR versus M/N for Set14 (b) SSIM versus M/N for Set14, from M/N = 0.01 to 0.50.

5.6(a) for M/N = 0.01 to 0.50, where M is the number of measurements and N is the total number of image pixels. The best results are obtained by the AL-DCT-THI and AL-DCT-THB algorithms. As discussed above, these algorithms are not compressive, and initially require N measurements from which the largest M coefficients are selected. However, if implemented using an SPC, the 2D-DCT is computed in the analogue domain. The remaining algorithms acquire M compressive measurements.

For $M/N \leq 0.15$ the best results results are obtained by the Shrink-IMDN algorithm, followed closely by the Shrink-VDSR algorithm. The compressive sensing algorithm CS-Net performs close to IMDN and VDSR on this image set. The simple L-DCT-ZZ algorithm, which is implementable on an SPC and acquires only M measurements, performs better than ISTA-Net+ and DR2-Net even though these use CNNs that execute on GPUs. AL-DCT-DD performs better than L-DCT-ZZ and with only a small increase in complexity. The non-real-time AL-DCT-DD-IDA algorithm performs best above M/N = 0.2. At M/N = 0.5, it is 1.5 dB better than CS-Net, though the later executes in real-time on a GPU. Above M/N=0.50 our real-time AL-DCT-DD performs better than CS-Net, both executing in real-time.

5.4.2 SSIM

The SSIM versus C_R results are shown in figures 5.5(b) and 5.6(b). The SSIM results follow the same pattern observed in the PSNR results with AL-DCT-THI and AL-DCT-THB performing better than the rest of the algorithms, the super resolution algorithms performing slightly better

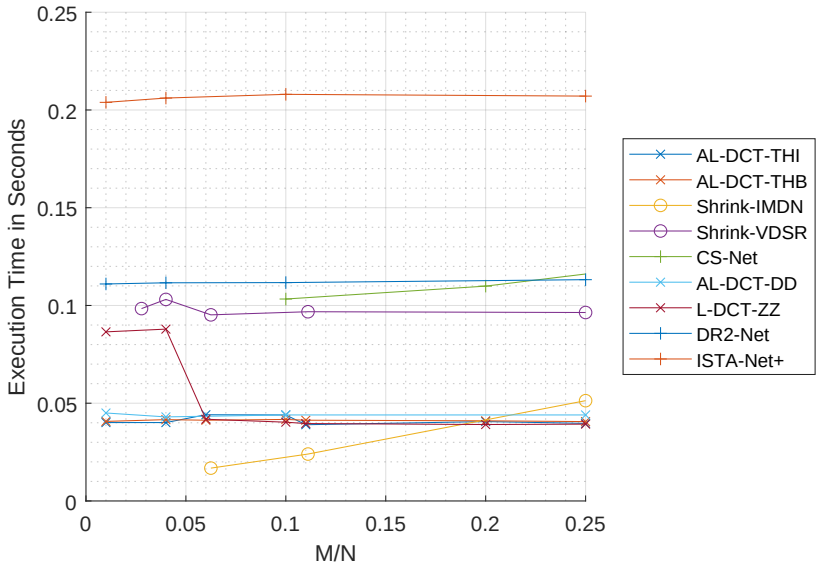


Figure 5.7: Execution Time versus M/N

than CS-Net, and AL-DCT-DD and L-DCT-ZZ exceeding the ISTA-Net+ and DR2-Net SSIM results. Our AL-DCT-DD-IDA algorithm matches the SSIM performance of CS-Net, but not in real-time.

5.4.3 Execution time

The execution time for the nine algorithms on Set14 is shown in figure 5.7. We note that apart from ISTA-Net+, all the algorithms run at less than 150 ms per image. For C_R ratios below 0.25, our [A]L-DCT algorithms and CS-Net require around 40 ms per image, on average. However, we note again the [A]L-DCT algorithms use the Central Processing Unit (CPU) not the GPU. The jump in execution time below $C_R = 0.05$ for the L-DCT-ZZ algorithm occurs because of the deblocking algorithm which runs without the benefit of the GPU. IMDN is the quickest algorithm below $C_R = 0.2$, but the execution time increases linearly with C_R.

5.5 Discussion of results

It is clear from the benchmarking exercise that AL-DCT-{THB|THI} algorithms exceed the state-of-the-art performance of the CNN CS algorithms CS-Net, ISTA-Net+ and DR2-Net, by a significant amount. However, these are non-compressive and are presented to benchmark the other algorithms.

Our Shrink-IMDN algorithm performs better than the best performing CS CNN, CS-Net, although the PSNR and SSIM performance is close at M/N = 0.25. AL-DCT-DD is our best performing CS algorithm and exceeds the performance of the GPU-implemented CS-Net at M/N = 0.5. Even the L-DCT-ZZ algorithm holds its own, exceeding the performance of DR2-Net and ISTA-Net+.

All the algorithms compared can execute on an SPC at the encoder. The Shrink-IMDN and

Shrink-VDSR super resolution algorithms require a simpler measurement matrix at the encoder and are therefore very suitable for low-energy applications.

Our real-time CS algorithms run on CPUs in less than 50 ms and compete with the Shrink-VDSR and Shrink-IMDN execution times though these require GPUs.

5.6 Conclusions

The Shrink-IMDN and Shrink-VDSR algorithms are worthy of further study as they produce the best PSNR and SSIM performance from a dimensionality perspective, whilst executing in real-time. The fact that the bicubic convolutional interpolation function, when used to down-sample images, results in better performance is very beneficial in our compressive sensing application.

We note that the output of the Shrink algorithm is a low-pass filtered image, that contains significant spatial correlation, so that it can be compressed further using low-complexity compression schemes, that respect the low-energy requirements of wireless sensor nodes.

At high compression ratios, the non-real-time AL-DCT-DD-IDA achieves better results than all other algorithms and is suitable as a post-processing function, should the quality of the real-time [A]L-DCT algorithms need to be improved.

Chapter 6

End-to-end transmission of compressively sensed images

6.1 Introduction

Chapters 4 and 5 introduced compressive sensing of images using deterministic 2D Discrete Cosine Transform (DCT) measurement matrices or downsampling using a bicubic convolutional interpolation filter. The techniques were investigated from a dimensionality reduction perspective, compressing the images from analogue measurements, and reconstructing them directly. This chapter investigates how these two techniques perform in an end-to-end transmission system setting that requires the analogue measurements to be quantized, encoded with low-complexity and transmitted digitally to a decoder.

Section 6.2 discusses some related work that has considered end-to-end transmission of compressively sensed images. Section 6.3 investigates how [A]L-DCT-{ZZ|BBV|DD} measurements can be encoded and transmitted with low complexity. The first technique studied adapts the JPEG post-quantization lossless encoding mechanism to quantized compressive measurements. The second technique is a lower-complexity tabular scheme proposed in this dissertation. These techniques are compatible with generic, GPU-accelerated reconstruction techniques implemented at the decoder.

Section 6.4 investigates how an image, downsampled using a bicubic convolutional interpolation filter on the compressive sensor, can be further compressed with low-complexity and digitally transmitted to the decoder. Once reconstructed at the decoder, the low resolution image can be up-scaled to a higher resolution using GPU-based Convolutional Neural Network (CNN) super resolution techniques.

Section 6.5 presents simulation results that compare the developed techniques with other state-of-the-art techniques from the literature. The Compressive Sensing (CS) techniques are compared using the framework introduced in chapter 3 in section 6.6 before summarising the chapter in section 6.7.

Algorithms

The following end-to-end image compressive sensing algorithms are developed in this chapter.

- L-DCT-ZZ-JPEG, AL-DCT-BBV-JPEG and AL-DCT-DD-JPEG use part of the JPEG standard to quantize and entropy encode the transform coefficients measured by the L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD algorithms developed in Chapter 4, prior to digital transmission.
- L-DCT-ZZ-JPEG-IDA uses the Iterative Denoising Algorithm (IDA) to improve the reconstruction quality of L-DCT-ZZ-JPEG.
- L-DCT-ZZ-JPEG-RD selects the compression ratio and JPEG quality factor points in the L-DCT-ZZ-JPEG algorithm, such that the compression ratio is the lowest possible and the PSNR is better than JPEG's.
- L-DCT-ZZ-JPEG-IDA-RD improves the quality of L-DCT-ZZ-JPEG-RD by using IDA to reconstruct images.
- TAB is a low-complexity table-based algorithm that quantizes and entropy encodes the transform coefficients produced by the L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD algorithms.
- AL-DCT-BBV-TAB and AL-DCT-DD-TAB use the TAB algorithm prior to digital transmission.
- AL-DCT-BBV-TAB-IDA and AL-DCT-DD-TAB-IDA use the IDA reconstruction algorithm to improve quality.
- Shrink-{JP2|PNG|LS-JPG} entropy encodes the compressed image produced by the Shrink algorithm using JPEG 2000 (JP2), PNG or LS-JPEG. The decoded image is then upsampled using bicubic interpolation.
- Shrink-{JP2|PNG|LS-JPG}-VDSR upsamples the decoded image using the GPU-based Very Deep Super Resolution (VDSR) CNN.
- Shrink-JP2-IMDN usamples the decoded image after compression by the JPEG 2000 algorithm, using the GPU-based Information Multi-Distillation Network (IMDN) CNN.
- GOLOMB-{FS|EE1|EE2|MHE|PSS} defines five algorithms for determining the GOLOMB *m* parameter: Full Search (FS), Entropy Estimator - 1 (EE1), Entropy Estimator - 2 (EE2), Mean Histogram Estimator (MHE), and Power Step Search (PSS).
- Shrink-DPCM-{HUFF|AC|GOLOMB-EE1} defines three algorithms for entropy coding the difference of pixels of the compressed image produced by the shrink algorithm: Huffman Coding (HUFF), Arithmetic coding (AC) and Golomb coding with the EE1 estimator (GOLOMB-EE1). The decoded images are then upsampled using bicubic interpolation or VDSR in algorithms Shrink-DPCM-{HUFF|AC|GOLOMB-EE1}-VDSR.

6.2 Related work

Mun and Fowler [116] noted that in block-based compressed sensing of images, the compressive measurements of adjacent blocks j at the same position m in the measurement vector $\mathbf{y}$, that is y_m^j, are highly correlated. They therefore proposed to encode y_m^j with reference to $\hat{y}_m^{(j-1)}$ using Differential Pulse Code Modulation (DPCM) with scalar quantization of the prediction error. The authors report that DPCM achieves 0.5 dB to 1 dB improvement over Block-based CS (BCS) reconstruction employing uniform scalar quantization alone. They also report that the performance of their MH-BCS-SPL algorithm with DPCM encoding is competitive when compared with JPEG and exceeds it when the bitrate is below 0.2 bpp. The authors also propose further lossless compression of the DPCM output signal, using arithmetic coding for example, but they do not present results for this technique.

Recently, Yuan and Haimi-Cohen [117] considered end-to-end transmission of compressively sensed images that quantizes and losslessly codes CS measurements prior to transmission. The authors consider deterministic matrices (DCT and DWHT) and Structurally Random Matrices (SRM) (SRM-DCT and SRM-DWHT) to sense the whole image. The DCT and Discrete Walsh Hadamard Transform (DWHT) coefficients are quantized using a uniform, mid-thread, scalar quantizer followed by a variable length lossless coder. In this work, the variable length coder is required because of the highly non-uniform quantized coefficient distribution. Reconstruction is studied using GAP-TV, D-AMP and NLR-CS. The authors report that simulation results show that the deterministic matrices performed consistently better than the structurally random counterparts.

6.3 End-to-end transmission of adaptive block-based compressively sensed images

Dimensionality reduction techniques based on the partial DCT measurement matrix result in analogue transform coefficients that have to be quantized prior to digital transmission. The quantized coefficients can still be compressed further by applying lossless entropy coding techniques such as Run-length Coding (RLC), Huffman coding and AC. This section describes techniques to quantize and code the output of the [A]L-DCT-{ZZ|BBV|DD} algorithms.

The linear 2D-DCT algorithms share the 2D-DCT transform with JPEG. An Single Pixel Camera (SPC) or Multi Pixel Camera (MPC) can be used to compute the transform coefficients in the analogue domain, that are then converted to digital prior to quantization and lossless coding. Two end-to-end algorithms are studied. The first, collects 2D-DCT 8×8 block measurements and encodes them using the efficient quantization and lossless coding algorithm used in JPEG as described in section 6.3.1. The second, collects 32×32 block measurements and encodes them for transmission using the TAB algorithm; a low complexity, lossless coding algorithm described in section 6.3.2.

6.3.1 The linear 2D-DCT algorithms with JPEG-style coding

To apply JPEG-style coding [54], the DC component is removed, and the AC Transform Coefficients (TCs) are divided by a quantization matrix and rounded as per equation 6.1, which is equivalent to applying a dead-zone quantizer to each TC. Following this step, a number of TCs are quantized to zero, with the probability of this happening increasing with frequency.

Lossless run-value coding is then used, by zigzag scanning the quantized TCs into a linear vector and coding the AC coefficients as (R,V) pairs with R being the number of preceding zeros and V, the quantized value of the coefficient. A triple Huffman code (H_R,H_N,H_V) is then generated from tables for the number R of preceding zeros, the number of bits N used to represent the V element and the codeword for the value V. The DC values are grouped together and coded using a double Huffman code (H_N,H_V) since there are no preceding zeros before each DC coefficient.

In the case of L-DCT-ZZ, when JPEG quantization and lossless coding are applied, some transform coefficients computed by the SPC or MPC are quantized to zero. As the compression ratio M/N increases to 1, quantized and lossless coded L-DCT-ZZ approaches JPEG compression performance.

To compute AL-DCT-THI and AL-DCT-THB it is necessary to first measure all the TCs and then select the largest coefficients to retain. When JPEG style quantization and lossless coding is applied to these threshold algorithms, the selection of coefficients becomes unnecessary as the quantization step does this automatically. Therefore, the performance of quantized and lossless encoded linear 2D-DCT is equivalent to JPEG.

6.3.2 The linear 2D-DCT with low complexity table-based encoding

The L-DCT-ZZ algorithm operates on grayscale images. If the image is colour, then it is converted to (Y,C_B,C_R) representation and the algorithm operates on the Y component. The grayscale component is partitioned into 32×32 pixel blocks and M, 2D-DCT measurements per block are computed starting from the DC component and proceeding through the AC components in JPEG zigzag order. The measurements would be collected from an SPC and MPC in analogue form and converted to digital using a uniform quantizer with a variable step size so that the higher the sequency of the transform coefficient, the less energy is expended in the analogue to digital conversion. In our simulation, the 2D-DCT results in double precision TCs. The i-th transform coefficient $\mathscr{T}_i$ is quantized by dividing it by a quantization matrix element Q_i and rounding the value, such that:

$$\mathscr{T}_i^q = \lfloor \mathscr{T}_i / Q_i \rceil, \tag{6.1}$$

where $\lfloor x \rceil$ rounds x to the nearest integer.

The quantized 2D-DCT measurements are then stored in an n_b row by M_b column table as shown in table 6.1, one block per row, with the DC component of each block occupying the leftmost column and the AC components stored with sequency increasing towards the rightmost

column. Therefore, each column of the transform coefficient table contains TCs from the same position in the 2D transform coefficient matrix.

For each column i, the maximum TC value $\mathscr{T}_{max,i}$ and minimum TC value $\mathscr{T}_{min,i}$ are calculated and stored as 8-bit integer values. From these values, the number of bits b_i required to store and transmit each of the quantized transform coefficients in a column as a fixed-length integer offset from the $\mathscr{T}_{max,i}$ or $\mathscr{T}_{min,i}$ extrema, is calculated from:

$$b_i = \lceil \log_2(\mathscr{T}_{max,i} - \mathscr{T}_{min,i}) \rceil. \tag{6.2}$$

The 2D-DCT algorithm compacts the energy towards the top left-hand corner of the 2D transform coefficient matrix. Therefore, the magnitude of the TCs decreases away from the DC component in zigzag scan order and many of the high-sequency coefficients are quantized to zero. In the proposed scheme, a zero AC value is coded as a 0 bit and the remaining AC values are prefixed with a 1 bit. This is only employed if a reduction in bits in a TC column is possible and a one-bit flag f_i is used to indicate whether this scheme is being employed in the column.

8 bits	$\mathscr{T}_{max,1}$	$\mathscr{T}_{max,2}$	...	$\mathscr{T}_{max,M_b}$
8 bits	$\mathscr{T}_{min,1}$	$\mathscr{T}_{min,2}$	...	$\mathscr{T}_{min,M_b}$
1 bit	f_1	f_2	...	f_{M_b}
$Block_1$	DC_1	$AC_{1,2}$	...	AC_{1,M_B}
$Block_2$	DC_2	$AC_{2,2}$	...	AC_{2,M_B}
...	...	...	...	...
$Block_{n_B}$	DC_{n_B}	$AC_{n_B,2}$	...	AC_{n_B,M_B}

Table 6.1: TAB algorithm table structure. Values in the top three rows are coded and transmitted using the number of bits in the first column. AC and DC values are bit-encoded using the number of bits calculated using equation 6.2.

It is possible for all the TCs in one column of the TC table to be quantized to zero. In this case $\mathscr{T}_{max,i}$, $\mathscr{T}_{min,i}$, and f_i are zero and the column values (all zeros) are not transmitted. The first three rows are transmitted in their entirety to allow the decoder to compute b_i, and determine which columns contain all-zero coefficients.

If the number of zero values in each column of the TC table in figure 6.1 is Z_i, then the number of bits N_b required to code an N-pixel image which has been compressively sensed by collecting M_b measurements per block is given by:

$$N_b = \sum_{i=1}^{M_b} \{[Z_i + (n_B - Z_i)(b_i + 1)] f_i + (n_B b_i \bar{f}_i)\} + 17M_b, \tag{6.3}$$

where n_B is the number of rows in the TAB table ($n_B = (H \times W) / (B \times B)$ is the number of blocks of size $B \times B$ in an image of height $H \times W$ pixels), and summation is over columns $i = 1 \ldots M_b$.

When the image is reconstructed at the receiver, the quantized transform coefficients are multiplied by the quantization matrix Q which is assumed available to both the transmitter and the receiver. Then each block is reconstructed by evaluating the inverse DCT on the TCs. This

is unlike CS, where the whole image is reconstructed by considering the received measurements from all blocks simultaneously. Block-by-block reconstruction allows the whole image to be reconstructed in a few tens of milliseconds. However, at high compression rates, the image will appear blocky. The Zig-Zag Scan Filter (ZZF) and Threshold Filter (THF) techniques can be used to deblock the image in real time as explained in chapter 4. Alternatively, IDA-DnCNN can be used to reconstruct the image as a whole to deblock and improve quality, but not in real time.

6.4 End-to-End transmission of downsampled images

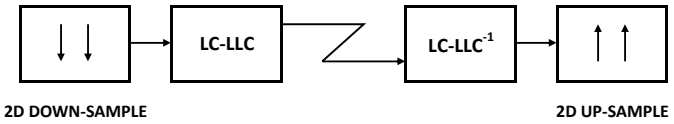


Figure 6.1: Super resolution with Low-Complexity Lossless Coding (LC-LLC)

Down-sampling followed by super resolution has been proposed as a simple and efficient scheme that can be implemented using compressive sensors in chapter 5. The low-resolution image can be further compressed using lossless or lossy techniques as shown in figure 6.1, but the computational effort required is reduced by a significant factor since there are $S_f \times S_f$ fewer pixels on which to operate at scale factor S_F. Lossless compression improves the rate-distortion performance of Super Resolution (SR) by decreasing the rate without altering the distortion.

LC-LLC can be deployed to increase compression further. The trade-off here is between the additional energy required at the encoder to compress the low-resolution image further, and that saved to transmit the compressed image. However, it is also possible to compress the low-resolution image using established algorithms such as JPEG 2000 [55], JPEG, Lossless-JPEG [54] and PNG [118]. Section 6.5.3 compares three low-complexity algorithms proposed in this chapter, based on spatial DPCM, against these more complex ones.

6.4.1 Low-complexity lossless coding

This section describes three low-complexity algorithms for compressing the low-resolution image produced by downsampling, using a 2D bicubic filter on a compressive sensor. DPCM is used to exploit spatial correlation between adjacent pixels. The pixel differences are then encoded using Huffman (HUFF), Arithmetic (AC) or Golomb (GOLOMB) entropy coders, resulting in three classes of algorithms; DPCM-HUFF, DPCM-AC, and DPCM-GOLOMB. DPCM-HUFF and DPCM-AC need to transmit the histogram values of the pixel differences to the decoder so that the decoder can calculate the Huffman code or probability (for arithmetic coding) of each transmitted pixel-difference symbol. A method for the efficient transmission of the pixel-difference histogram to the decoder is therefore developed in this section. Golomb coding is fully specified by one parameter m and the histogram is not transmitted in this case.

Pixel difference histogram for arithmetic and Huffman coding

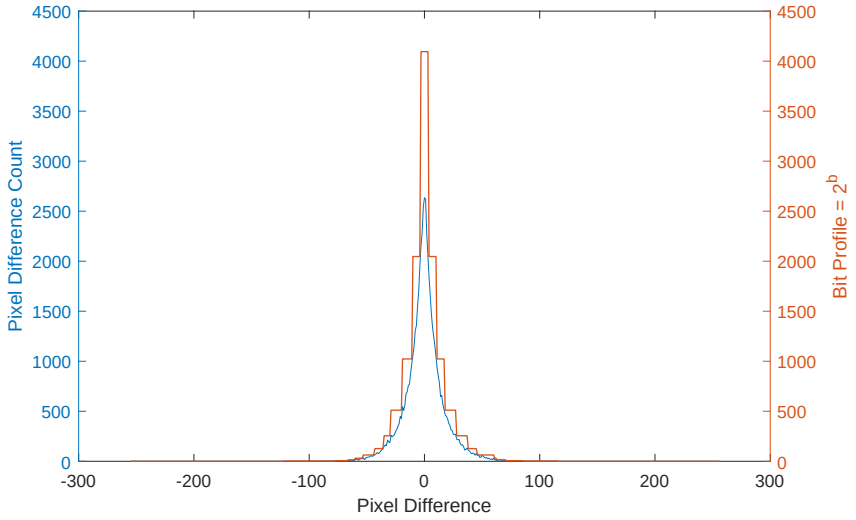


Figure 6.2: Histogram (blue) of the pixel difference count and the histogram bit profile (orange) for baboon.bmp in Set14 [107]. The plateaus of the histogram bit profile equal 2^b where b is the number of bits used to encode the histogram values below the plateaus.

DPCM generates the difference between the current pixel and the previous pixel in the horizontal direction, except for the first pixels in each row for which the difference is derived from the first pixel in the row above the current pixel. The first pixel on the first row is coded in absolute terms. If the dynamic range of the input image is $[0 \dots V]$, the dynamic range of the pixel difference is $[-V \dots V]$.

The histogram of the pixel difference is collected in $2V+1$ counters and is then used to calculate the symbol probabilities. Golomb coding uses the histogram to find the optimal value for the Golomb code parameter m. In DPCM-AC, the symbol probabilities are then used in arithmetic coding of the DPCM pixel differences. In the DPCM-HUFF algorithm, the symbol probabilities are used to build a Huffman code for each image at the encoder. In both DPCM-AC and DPCM-HUFF algorithms, successful decoding depends on the transmission of the histogram to the decoder since the arithmetic decoder needs the symbol probabilities to decode the image and the DPCM-HUFF algorithm needs the symbol probabilities to build the Huffman decoder.

The statistics of the pixel differences generated by the DPCM process, have been described by Laplace, Generalized Gaussian or Mixture distributions, amongst others [119]. If complexity were not an issue, it would be possible to model the distribution of the pixel differences and only transmit the model parameters to the decoder. Because of complexity issues, in this case, the histogram counts are transmitted directly using an efficient scheme.

Figure 6.2 shows the histogram of pixel differences in blue and the histogram bit profile in orange for the luminance component of the baboon.bmp image in Set14 [107]. The histogram bit profile is equal to 2^b where b is the integer number of bits required to code the counts in the histogram using a fixed length code. Scanning the histogram bit profile from left to right, step-increases occur at the first histogram value that requires an increased integer number of bits

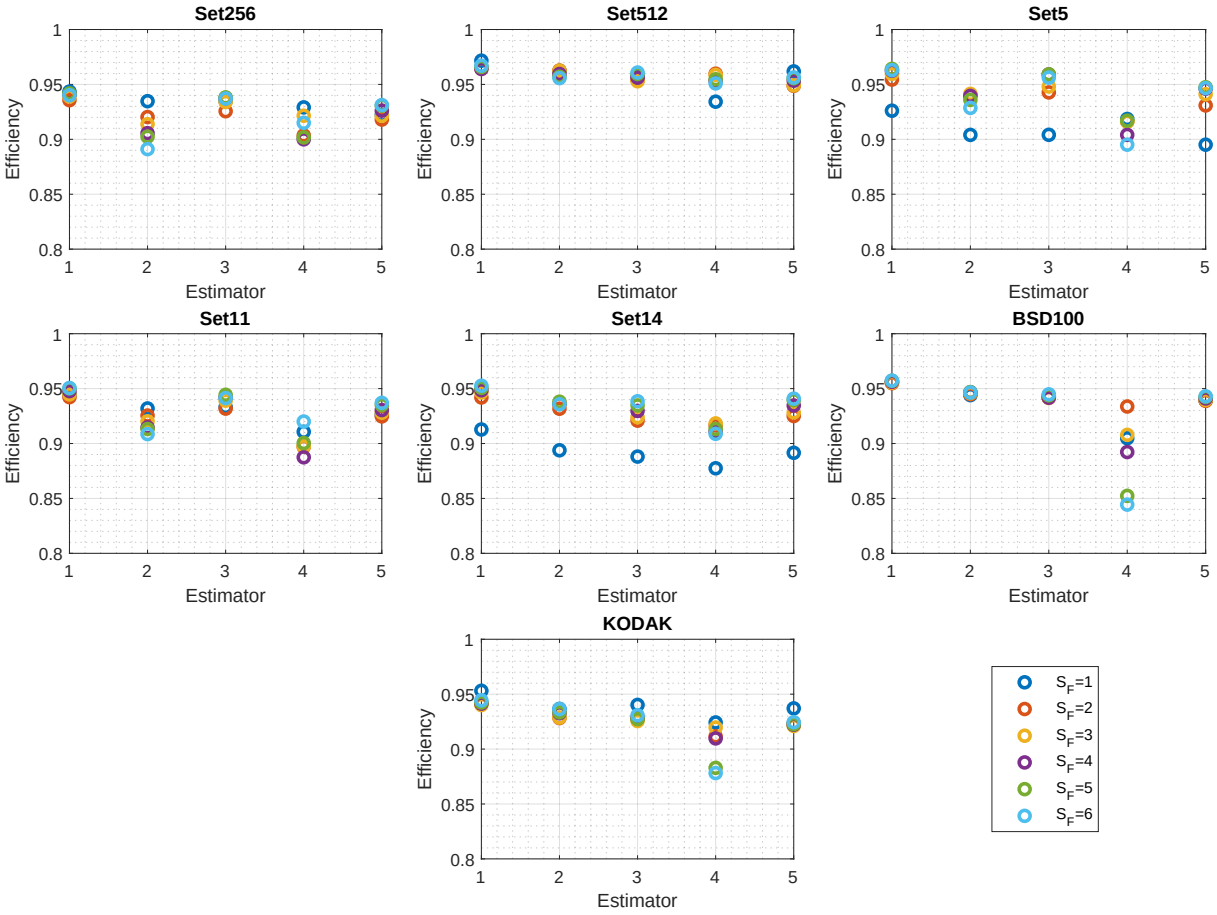


Figure 6.3: Efficiency achieved by the five parameter m estimators; (1) FS, (2) EE1, (3) EE2, (4) MHE, (5) PSS, for seven image sets at six subsampling rates $S_F = 1 \cdots 6$, applied in the horizontal and vertical dimensions. S_F=1 means no subsampling - use DPCM on original image.

and decreases when there are no more histogram values that require the current integer number of bits.

The number of bits required to encode the histogram bit profile in figure 6.2 is 2119 bits. This comprises 511 bits which encode the location of non-zero counts in the Histogram, the bytes which encode the transitions in the profile and the counts coded in a fixed length code with the number of bits b indicated by the log_2 of the histogram bit profile. The histogram counts are geometrically distributed around 0. Therefore, the counts are coded with a decreasing number of bits away from 0.

Golomb coding

The lowest-complexity is achieved using a Golomb coder which is coded into algorithm DPCM-GOLOMB. The Golomb coder is characterised by a parameter m. This parameter is used to code a symbol n by computing a quotient q, a remainder r and a value L as follows [120]:

$$q = \left\lfloor \frac{n}{m} \right\rfloor, \quad r = (n - qm), \quad L = 2^{\lceil \log_2(m) \rceil} - m \,, \tag{6.4}$$

where $\lfloor x \rfloor$ is the floor function – the integer part of x; and $\lceil x \rceil$ is ceiling function – next integer larger or equal to x. Since pixel differences are positive and negative, these are pre-processed

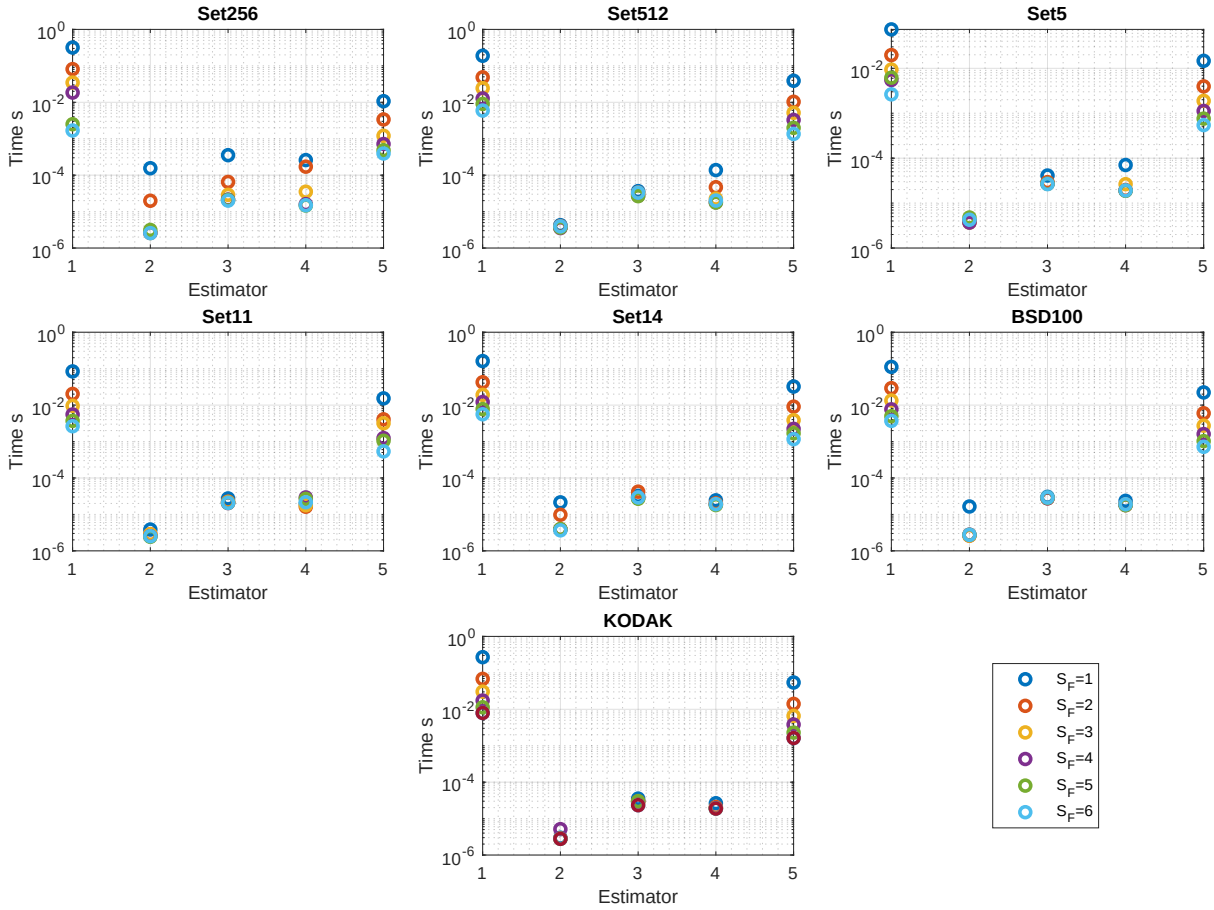


Figure 6.4: Execution time for the Golomb parameter m Estimators; (1) FS, (2) EE1, (3) EE2, (4) MHE, (5) PSS, for seven image sets and sub-sampling rates $S_F = 1 \cdots 6$, applied in the horizontal and vertical dimensions. S_F=1 means no subsampling - use DPCM on original image.

such that 0 and positive values become even, whilst negative values become odd.

Then integer quotient q is coded using a unary code, that is q 1's followed by one 0. If m is a power of two, the remainder r is coded using a fixed length code with $\log_2(m)$ bits. If it is not a power of two and it is less than L, then it is coded using $\lfloor \log_2(m) \rfloor$ bits. Otherwise, it is coded as $(r + 2^{\lceil \log_2(m) \rceil} - m)$ using $\lceil \log_2(m) \rceil$ bits [120].

The optimal value of this parameter can be found by searching the value of m that minimizes the Average Codeword Length (ACWL) $\bar{\ell}$. The efficiency η of the code is then given by $\eta = E / \bar{\ell}$ where E is the entropy calculated from the histogram. For the image sets under study, to optimise m, it was enough to conduct a *full search* (FS), for all values of m between 1 and 32. Because complexity is an issue, four other parameter-m estimation algorithms were developed and compared. The *power step search* (PSS) algorithm, searches for the best η for $m = \{1, 2, 4, 8, 16, 32\}$, cutting the computation by more than five times.

FS and PSS compute the codeword lengths for all pixel differences for each value of m and are thus computationally intensive. Three estimators for m were developed empirically which operate on the image Entropy and the mean and standard deviation (std) of the image Histogram, namely:

- Entropy estimator - 1 (EE1)

$$m = \lfloor 2(Entropy) \rfloor (Entropy < 4) + (\lceil 2(Entropy) \rceil + 1)(Entropy \geq 4), \tag{6.5}$$

- Entropy estimator - 2 (EE2)

$$m = \left\lceil \frac{sum(Histogram)}{128000} \left\lceil \frac{3000(Entropy)}{std(Histogram)} \right\rfloor \right\rceil, \tag{6.6}$$

- Mean histogram estimator (MHE)

$$m = \left\lceil \log_2 \left[\frac{mean(Histogram)}{2} \right] \right\rceil. \tag{6.7}$$

The efficiency results obtained on the seven image sets; Set256 in figure 4.4, Set512 in figure 4.5, Set5 [121], Set11 [18], Set14 [107], BSD100 [44] and KODAK [122], at 2D sub-sampling factors (S_F) from 2 to 6, are shown in figure 6.3. The time taken for the algorithms to execute in Matlab is shown in figure 6.4.

The results in figure 6.3 show that the full search finds the best m parameter with the simple Entropy Estimators EE1 and EE2 performing very well. Figure 6.4 shows that Entropy Estimator EE1 is the least complex estimator and is around two to three orders of magnitude faster than the full search. Figure 6.5 compares the PSNR and SSIM achieved using the five estimators with SR-DPCM-GOLOMB on set14 images (downsampled using the SHRINK algorithm). There is little difference between the estimators and hence EE1 can be chosen as the most computationally efficient.

6.5 Results: comparison of techniques

Sections 6.5.1 and 6.5.2 report the PSNR and SSIM performance of our image compressive sensing algorithms when entropy coded using the JPEG algorithm and the TAB algorithm using fixed length coding. Section 6.5.3 presents the end-to-end PSNR and SSIM performance of our image super resolution algorithms namely Shrink-LLC-VDSR and Shrink-LLC-IMDN.

To compare the end-to-end image CS and image SR algorithms developed in this chapter with the literature, three state-of-the-art GPU-accelerated algorithms, namely DR2-Net [46], ISTA-Net+ [20] and CS-Net [19] are quantized and fixed-length coded as shown in section 6.5.4. The three end-to-end CS CNNs are then compared with the image CS and image SR algorithms proposed in this chapter, in section 6.5.5.

Most of the tests are conducted on image Set14 [107] that contains fourteen images, twelve of them in colour, and in various resolutions from 250×356 (Comic) to 768×512 (Butterfly). The colour RGB images are converted to YCbCr using the Matlab *rgb2ycbcr*() function. Set14 was selected because it is widely used in the image super resolution literature and it contains resolutions that would be typical of the target applications in this dissertation. All the results

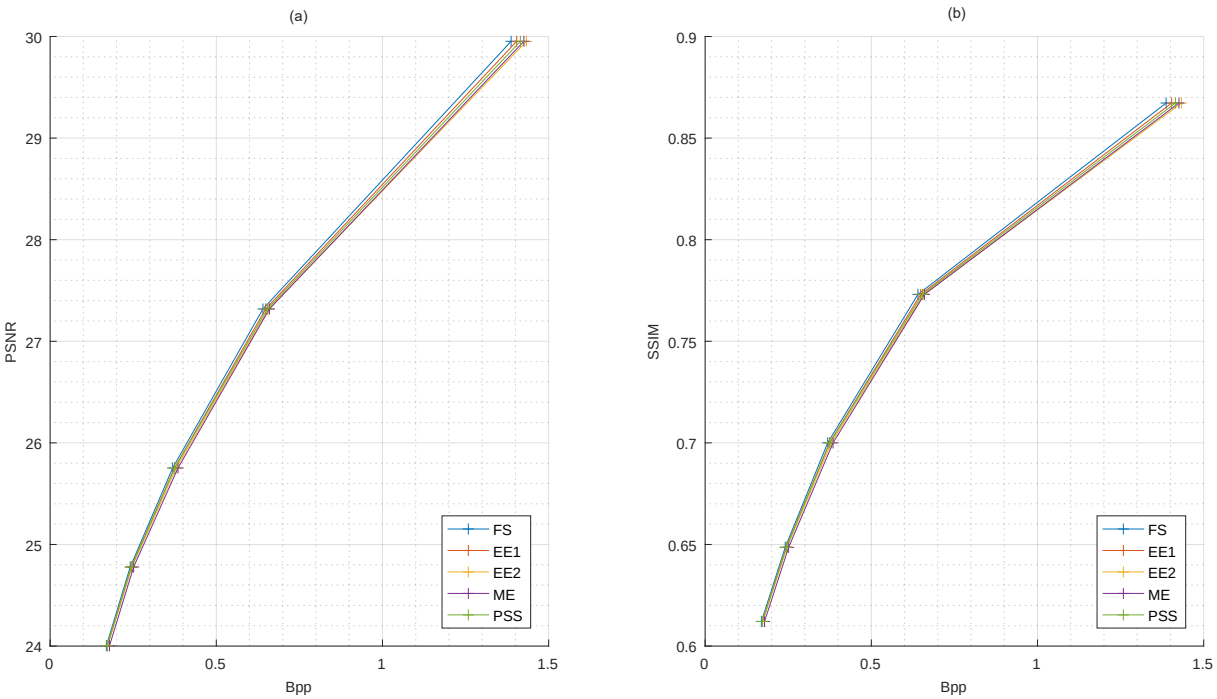


Figure 6.5: (a) PSNR and (b) SSIM performance of Shrink-Golomb-Bicubic algorithm with the five different Golomb parameter *m* estimators for Set14. Downsampling using the Shrink algorithm followed by bicubic upsampling. The plots show that the PSNR and SSIM performance of the five algorithms are practically the same and it is possible to use the least complex algorithm, that is EE1.

in this section are reported on the luminance Y component only. The hardware setup was the same as in section 4.6.

6.5.1 End-to-end transmission of compressively sensed images with JPEG-style entropy coding

The block-based [A]L-DCT-{ZZ|BBV|DD} image CS algorithms are set to compressively sense images using 8×8 blocks. The quantization matrix Q_{MPEG} is that from the MPEG standard [124], and achieves better PSNR and SSIM results than the standard JPEG algorithm with the default JPEG matrix. The JPEG code used to compress the quantized coefficients is based on [125] and follows the JPEG algorithm closely.

To generate the results in figure 6.6, Set14 is compressed using the three algorithms, L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD, at the seven compression ratios shown. Then the quantization matrix is multiplied with a quality factor Q_F to obtain six rate distortion points for each CS compression ratio.

The ensuing rate distortion curves saturate towards the higher rates. The first three rate distortion points of the compression ratios, $0.234, 0.328$ and 0.438, are marginally better than JPEG applied to the whole image with all the pixels available. The next three compression ratios, $0.563, 0.672$ and 0.766, perform better over the first four rate distortion points. At $C_R = 0.844$ five of the six rate distortion points are above JPEG.

Therefore, L-DCT-ZZ-JPEG reaches the performance of JPEG, with less computations than

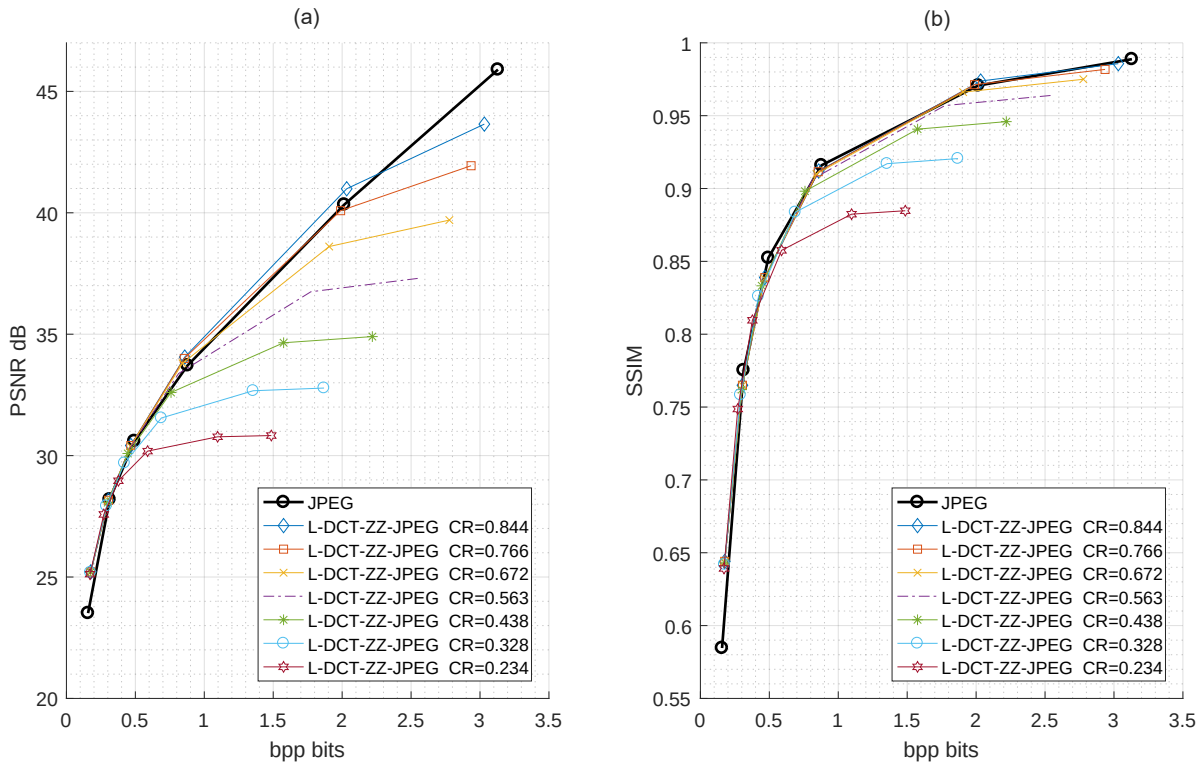


Figure 6.6: Comparison of (a) PSNR and (b) SSIM for L-DCT-ZZ-JPEG quantized and coded using the JPEG style algorithm at different $C_R = M/N$ ratios, with standard JPEG, on Set14. The plots show that L-DCT-ZZ-JPEG can achieve the performance of JPEG but measuring a fraction - C_R - of all the transform coefficients.

standard JPEG which measures all transform coefficients. To transmit images at 0.4 bpp, say, 15 out of 64 ($C_R = 15/64 = 0.234$) 8×8 transform coefficients in zigzag scan order are sensed and then quantized and coded using the JPEG algorithm. The PSNR is marginally better than standard JPEG due to the use of the Q_{MPEG} matrix. If more energy to transmit bits is available, a different quality factor Q_F can be used to transmit the image at 1.5 bpp, increasing PSNR to 31 dB, using the same L-DCT-ZZ measurements. However, had all transform coefficients been collected, standard JPEG would have reached 37 dB at 1.5 bpp. This PSNR can be exceeded by increasing C_R to 0.766 and selecting a appropriate Q_F. The SSIM plot in figure 6.6b shows a similar trend for SSIM.

The results for the AL-DCT-BBV-JPEG and AL-DCT-DD-JPEG algorithms are nearly identical to those for L-DCT-ZZ-JPEG in figure 6.6 and are not reported further. This behaviour results from the fact that quantization of the L-DCT-ZZ, AL-DCT-DD and AL-DCT-BBV transform coefficients results in a very similar set of non-zero TCs. Thus, L-DCT-ZZ becomes adaptive in the sense that each 8×8 sub-image block now has a different number of TCs.

IDA reconstruction

When L-DCT-ZZ-JPEG encoded images are reconstructed using IDA, the PSNR and SSIM performance improves considerably, as shown in figure 6.7. All the PSNR and SSIM performance values exceed JPEG's at all compression ratios and Q_F points except for the last Q_F value in the

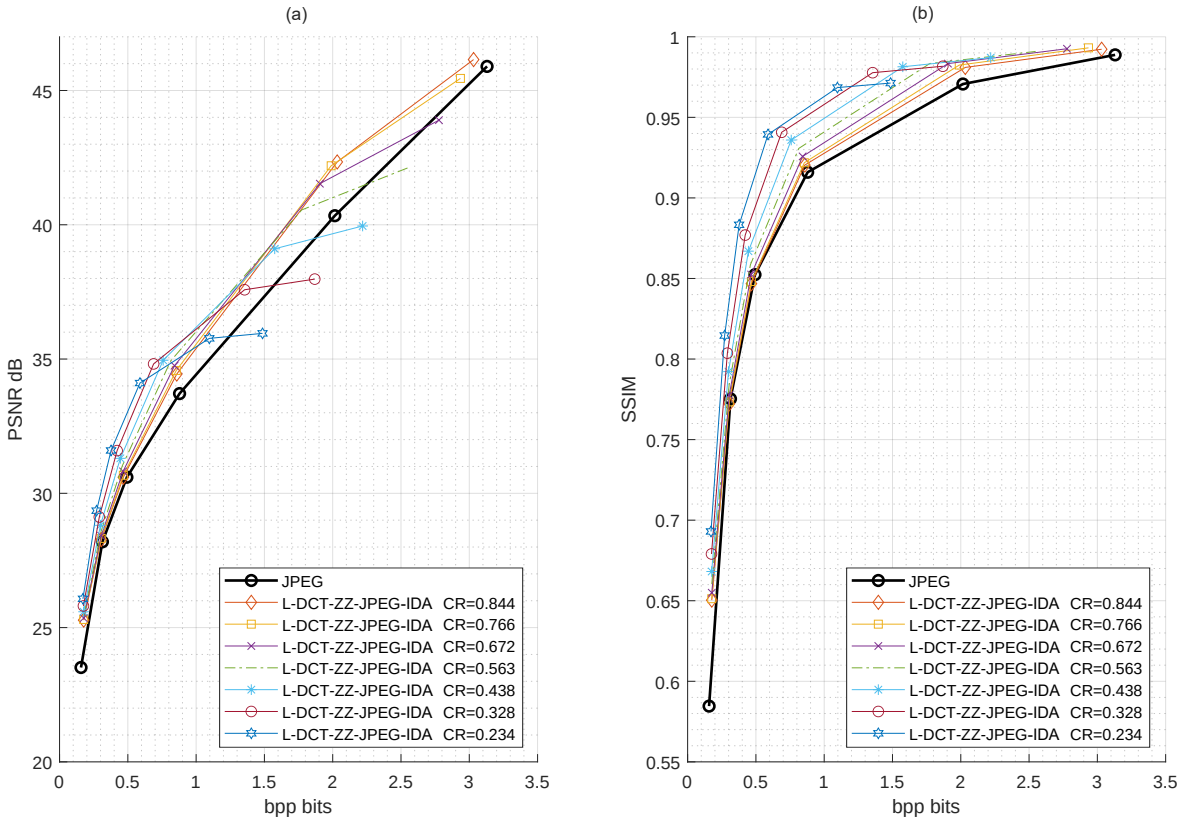


Figure 6.7: Comparison of (a) PSNR and (b) SSIM for L-DCT-ZZ-JPEG-IDA quantized and coded using the JPEG style algorithm at different $C_R = M/N$ ratios, against standard JPEG, on Set14. The plots show that L-DCT-ZZ-JPEG-IDA can substantially exceed the performance of JPEG even when measuring a fraction - C_R - of all the transform coefficients.

PSNR plot. The maximum improvement in PSNR is 3 dB at 0.7 bpp and SSIM increases from 0.88 to 0.95 at the same rate. This represents a remarkable improvement in the JPEG quality and can be applied retrospectively to JPEG encoded images.

The L-DCT-ZZ-JPEG-[IDA]-RD algorithms

The L-DCT-ZZ-JPEG-[IDA]-RD algorithms are now defined that select the C_R and Q_F points in the L-DCT-ZZ-JPEG-[IDA] algorithms, such that C_R is the lowest possible and the PSNR is better than JPEG's. The values selected from figures 6.6 and 6.7 by the RD algorithm are tabulated in table 6.2. The PSNR and SSIM results of the L-DCT-ZZ-JPEG-[IDA]-RD algorithms are plotted in figure 6.8.

The L-DCT-ZZ-JPEG-IDA-RD performs better than JPEG at all rates. Between 0.4 bpp and 0.5 bpp L-DCT-ZZ-JPEG-IDA-RD approaches the PSNR performance of JPEG 2000. Above 0.3 bpp L-DCT-ZZ-JPEG-IDA-RD exceeds the SSIM performance of JPEG 2000 on Set14. The PSNR performance of L-DCT-ZZ-JPEG-RD is the same as JPEG, however, the latter has better SSIM performance above 0.3 bpp.

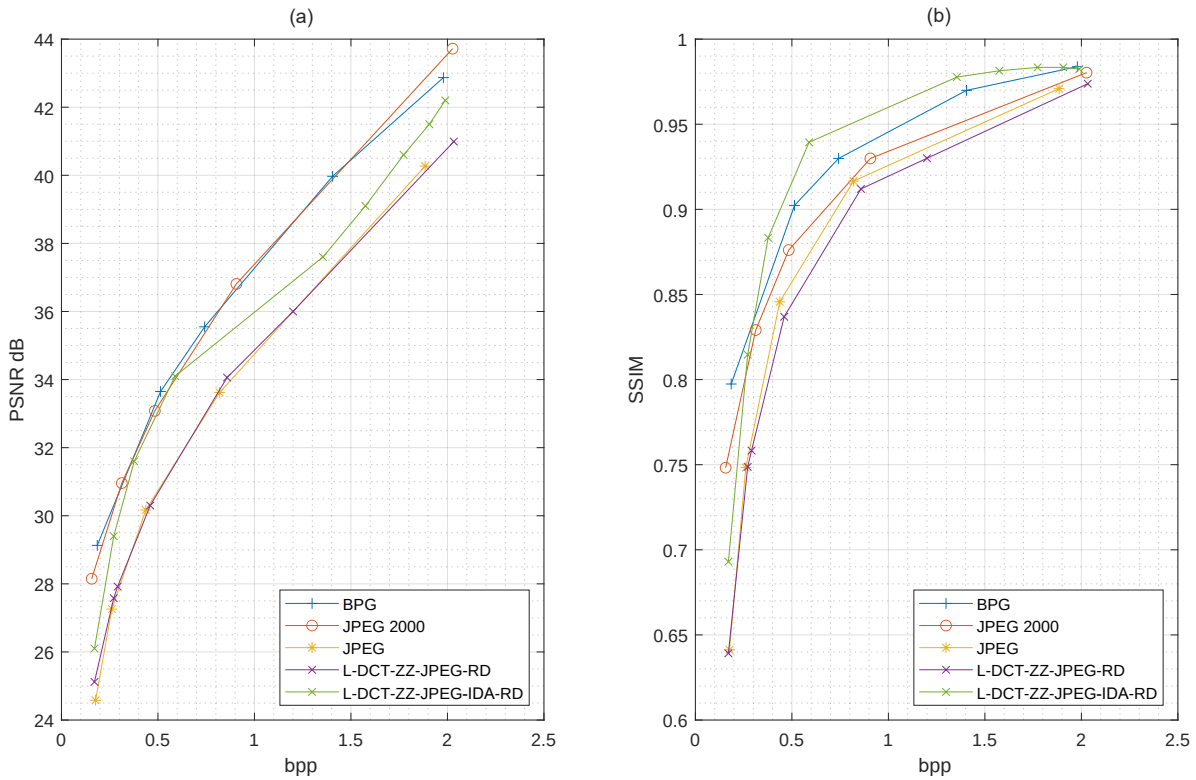


Figure 6.8: (a) PSNR and (b) SSIM vs bpp for Better Portable Graphics (BPG) [123], JPEG 2000 [55], JPEG [54], L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD with $B = 8$ on Set14. The plots show that L-DCT-ZZ-JPEG-RD matches the performance of JPEG, whereas L-DCT-ZZ-JPEG-IDA-RD exceeds it significantly. L-DCT-ZZ-JPEG-IDA-RD matches the PSNR performance of BPG and JPEG between $C_R = 0.6$ and $C_R = 0.7$ whereas its SSIM exceeds that of all other algorithms above $C_R = 0.3$.

Comparison with state-of-the-art image coding algorithms

It has been proposed that the intramode coding algorithm of the latest High Efficiency Video Coding (HEVC) standard [52] is currently the best compression algorithm for still images or video frames [126]. The BPG format was specified by Fabrice Bellard [123] and is based on a subset of HEVC. Figure 6.8 compares PSNR versus rate in bpp for Set14 using the JPEG 2000 [55], JPEG [54], BPG [123], L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD compression algorithms.

JPEG 2000 and JPEG were studied using the respective option of the Matlab *imwrite*() function, that allows setting the target rate in JPEG 2000 and the target quality in JPEG. BPG was downloaded from [127]. The command line versions of the BPG encoder and decoder (bpgenc.exe and bpgdec.exe are invoked [127]) as external calls from within Matlab to generate the results.

Figure 6.8 shows that BPG and JPEG 2000 achieve similar PSNR performance on Set14, and perform 1.5 dB better than JPEG. The L-DCT-ZZ-JPEG-RD algorithm with an 8×8 block-size, matches the JPEG performance. IDA reconstruction improves the performance and L-DCT-ZZ-JPEG-IDA-RD matches the PSNR performance BPG and JPEG 2000 when the rate is between 0.4 and 0.6 bits per pixel. The PSNR is consistently higher than JPEG and L-DCT-ZZ-JPEG-RD. The SSIM performance of L-DCT-ZZ-JPEG-IDA is even more more remarkable,

L-DCT-ZZ-JPEG-RD				
C_R	Q_F	Bpp	PSNR	SSIM
0.234	9.524	0.171	25.12	0.6392
0.234	4.878	0.271	27.58	0.7487
0.328	2.985	0.291	27.92	0.7583
0.563	2.985	0.460	30.30	0.8369
0.844	1.389	0.858	34.06	0.9120
0.844	1.389	1.200	36.00	0.9300
0.844	0.392	2.033	40.99	0.9738

L-DCT-ZZ-JPEG-IDA-RD				
C_R	Q_F	Bpp	PSNR	SSIM
0.234	9.524	0.171	26.1	0.6930
0.234	4.878	0.271	29.4	0.8146
0.234	2.985	0.377	31.6	0.8833
0.234	1.389	0.589	34.1	0.9394
0.328	0.392	1.355	37.6	0.9777
0.438	0.392	1.575	39.1	0.9814
0.563	0.392	1.773	40.6	0.9834
0.672	0.392	1.906	41.5	0.9834
0.766	0.392	1.989	42.2	0.9824
0.844	1.185	3.031	46.2	0.9923

Table 6.2: L-DCT-ZZ-JPEG-[IDA]-RD Rate/Distortion values, on Set14. C_R is the compression ratio, Q_F is the quantization matrix multiplication factor, and Bpp the rate in bits per pixel. This table shows the quality factors Q_F of the JPEG algorithm that when combined with the compression ratio C_R of the L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD algorithms, generate the rate distortion points plotted in figure 6.8.

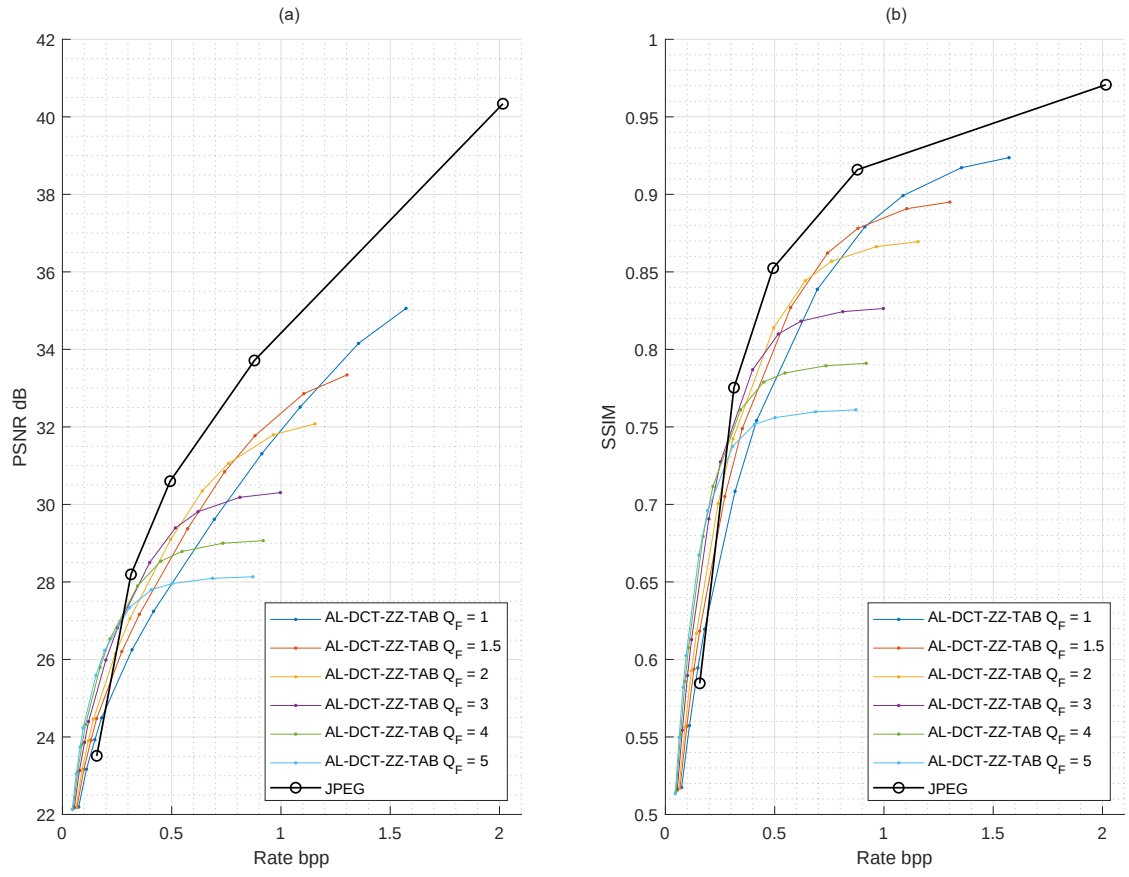


Figure 6.9: (a) PSNR and (b) SSIM versus Rate in bpp for L-DCT-ZZ-TAB with blocksize B=32 and various compression ratios, on Set14. The compression ratio points on the TAB curves were generated using $C_F = \{1.5, 2, 3, 4, 6, 12, 17, 35, 45, 65, 106\}$. The plots shows that the PSNR performance of AL-DCT-ZZ-TAB is better than JPEG below $C_R = 0.3$.

significantly outperforming all other algorithms between a rate of 0.3 and 1.9 bits per pixel.

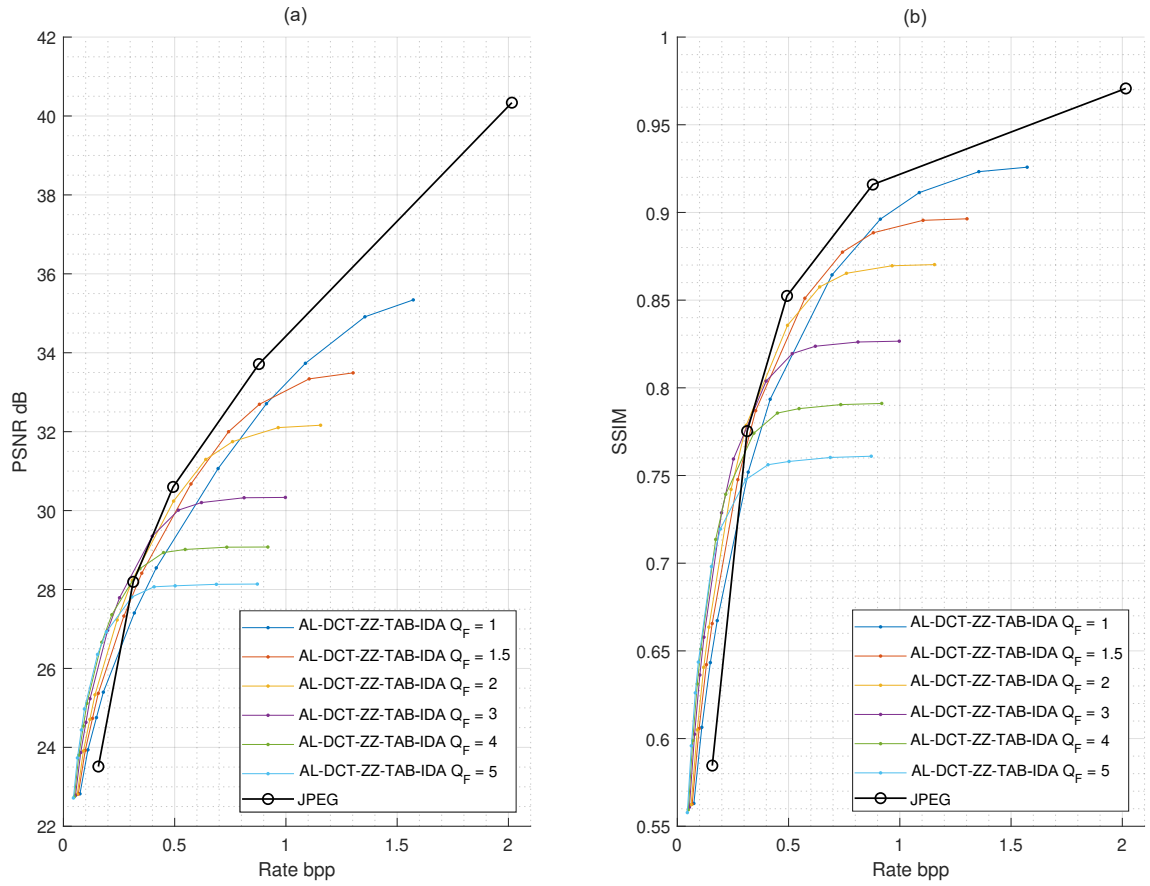


Figure 6.10: (a) PSNR and (b) SSIM versus Rate in bpp for L-DCT-ZZ-TAB-IDA with blocksize B=32 and various compression ratios, on Set14. The compression ratio points on the TAB curves were generated using $C_F = \{1.5, 2, 3, 4, 6, 12, 17, 35, 45, 65, 106\}$. The plots shows that the PSNR performance of AL-DCT-ZZ-TAB-IDA is better than JPEG below C_R = 0.4.

6.5.2 End-to-end transmission of compressively sensed images using the TAB algorithm

The PSNR and SSIM performance of L-DCT-ZZ-TAB with 32×32 blocks and 2D IDCT reconstruction, is better than JPEG below 0.25 bpp, as shown in figures 6.9. These results were generated using a blocksize B=32, and a quantization matrix derived from the 8×8 quantization matrix in [128], by using the Matlab imresize function with bicubic interpolation, to augment its size to 32×32.

The quantization matrix multiplication factors Q_F used were { 1, 1.5, 2, 3, 4 and 5 }. The compression factors $C_F = \{1.5, 2, 3, 4, 6, 12, 17, 35, 45, 65, 106\}$ were used to generate the rate distortion points on the Q_F curves. The PSNR and SSIM performance improves when reconstruction uses the IDA algorithm as show in figure 6.10. PSNR increases by around 1 dB at a rate of 0.3 bpp, and SSIM increases from 0.74 to 0.78.

Six TAB variants [A]L-DCT-{ZZ|BBV|DD}-TAB-[IDA] are compared in figure 6.11. L-DCT-ZZ-TAB-IDA performs better than AL-DCT-DD-TAB-IDA and significantly better than AL-DCT-BBV-TAB-IDA. This occurs because the TAB algorithm is better at compressing the more uniform number of TCs per block in L-DCT-ZZ than the adaptive number of TCs in AL-

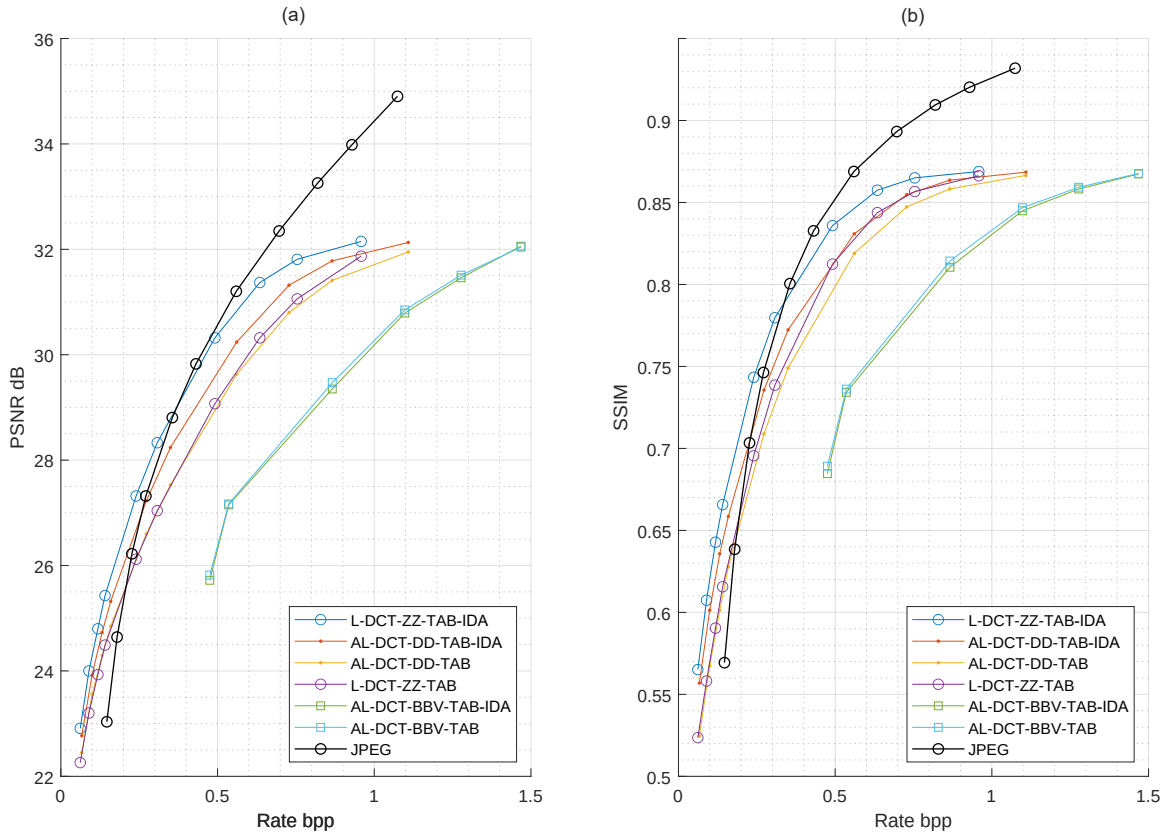


Figure 6.11: (a) PSNR versus bpp and (b) SSIM versus bpp for the TAB algorithm variants with B=32 on Set14. The plots show that the ZZ variants perform better than the DD variants and significantly better than the BBV variants.

DCT-DD and AL-DCT-BBV. The latter has the most variability and is the least compressed by TAB.

L-DCT-ZZ-TAB-[IDA]-RD

As in the case of the L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-TAB-RD and L-DCT-ZZ-TAB-IDA-RD are defined as the algorithms that pick out the best rate distortion points in figures 6.9 and 6.10 respectively. The results of these four algorithms are compared with JPEG 2000 and JPEG in figure 6.12. Below 0.17 bpp L-DCT-ZZ-TAB-IDA-RD and L-DCT-ZZ-TAB-RD perform better than L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-JPEG-RD and JPEG because of the larger block size.

6.5.3 End-to-end ISR: Shrink-LLC-SR

Figure 6.13 compares the PSNR for Set14 compressed with JP2, JPEG, L-DCT-ZZ-JPEG-RD and the schemes that use Shrink (the SPC and MPC version of imresize) followed by Lossless Coding (LLC) schemes which include the lossless mode of JPEG 2000, PNG and LS-JPEG, and DPCM-based, lower complexity schemes namely: DPCM-AC, DPCM-Huffman and DPCM-GOLOMB-EE1. These schemes are super resolved to the original size either using IMDN [109], VDSR [106] or the Matlab imresize function with bicubic interpolator. These schemes

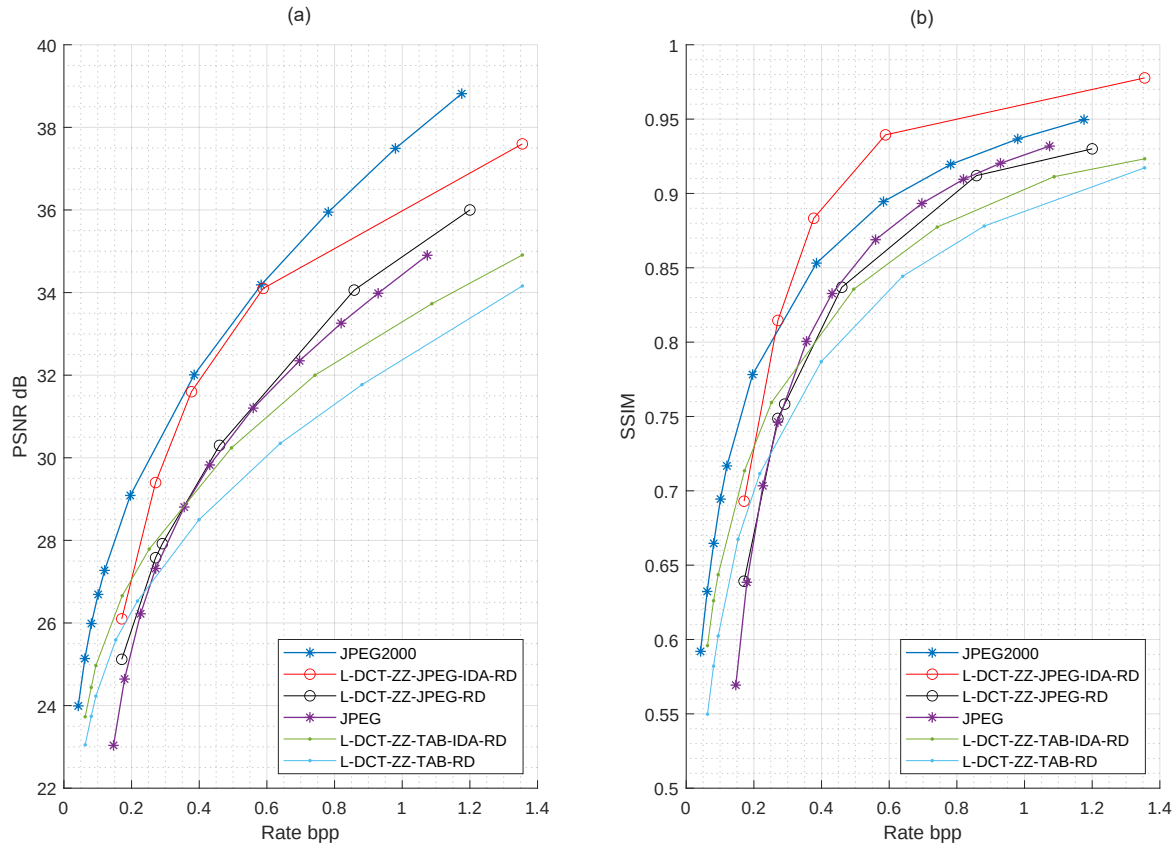


Figure 6.12: (a) PSNR versus Bpp for (b) SSIM versus Bpp for the L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-IDA-RD, L-DCT-ZZ-TAB-RD, JPEG 2000 and JPEG on Set14. The plots show that there is a rate above which it is better to use L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-JPEG-IDA-RD algorithms, but below which it is better to use L-DCT-ZZ-TAB-RD and L-DCT-ZZ-TAB-IDA-RD.

can be generically referred to as Shrink-LLC-SR.

The IMDN code was downloaded from [129]. The code was run using Pytorch 1.4. The VDSR code was downloaded from [130]. We trained the VDSR CNN to super resolve images at subsampling factors $S_F = \{2,3,4,5,6\}$ on a Linux platform. The same 291-image training set used in the original VDSR paper [106], was augmented as instructed in [130] and used for training. The tests were executed on a Windows platform with the aid of MatConvNet [114].

In figures 6.13 and 6.14, Shrink-DPCM-ENTROPY plots the entropy in the DPCM residual as a benchmark. Shrink-LLC-IMDN meets the performance of L-DCT-ZZ-JPEG-RD at 0.325 bpp. Below 0.2 bpp, Shrink-LLC-VDSR schemes perform better than L-DCT-ZZ-JPEG-RD and JPEG from a PSNR perspective. Above 0.2 bpp, L-DCT-ZZ-JPEG-RD performs better. The Shrink-LLC-VDSR schemes perform 2.5 dB lower than JPEG 2000 at low rate. However, the performance gap increases to between 5 dB to 6 dB at 1 bpp.

Figure 6.14 compares SSIM for Set14 for the same compression schemes. Again, Shrink-LLC-IMDN performs better than Shrink-LLC-VDSR, and the latter schemes perform better than L-DCT-ZZ-JPEG-RD and JPEG. The SSIM performance gap between JPEG 2000 and the Shrink-LLC-VDSR schemes remains constant at around 0.075 between 0.15 bpp and 1.15 bpp.

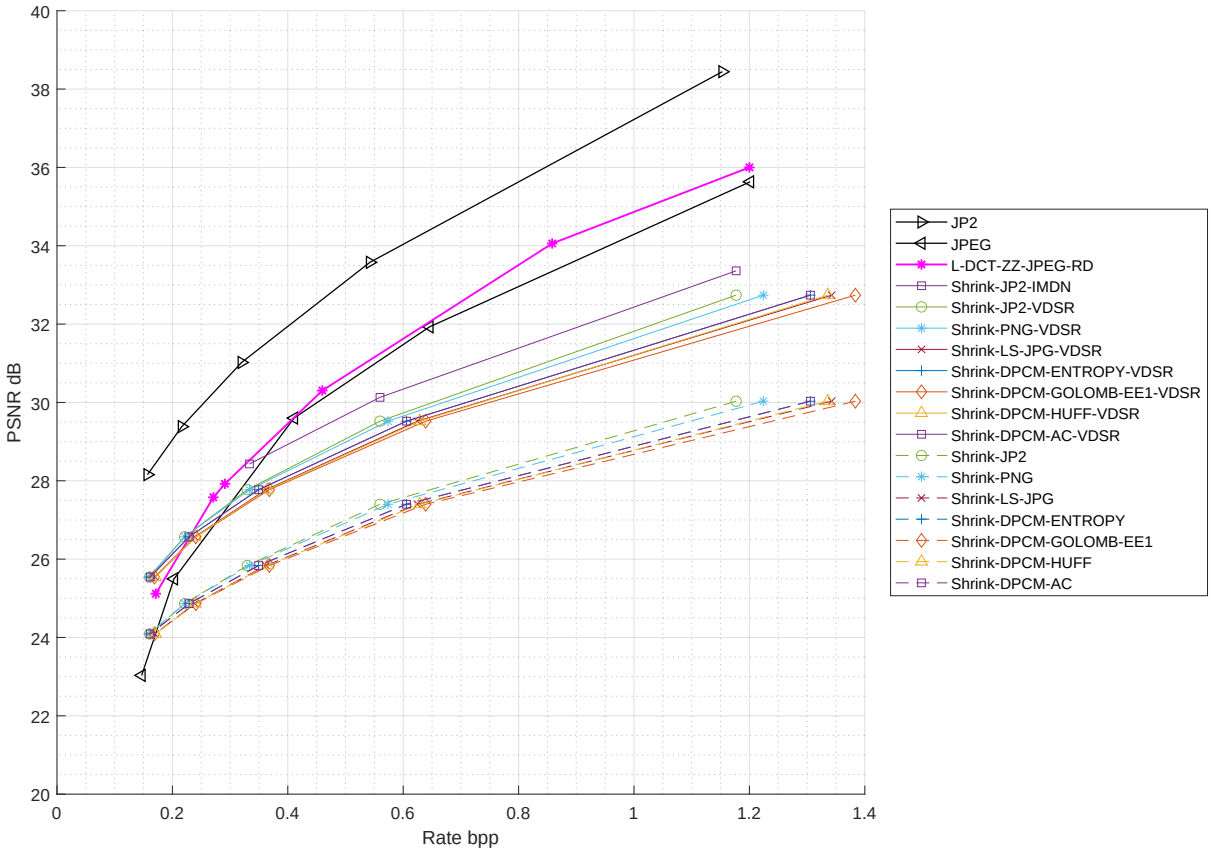


Figure 6.13: PSNR for Set14 - JPEG 2000 (JP2), JPEG, L-DCT-ZZ-JPEG, Shrink-LLC-IMDN, Shrink-LLC-VDSR. The plots show that VDSR upsampling results in a significant improvement in PSNR over bicubic upsampling, for the losslessly coded (LLC) variants following downsampling using Shrink. The Lossless compression schemes investigated are JPEG 2000 (JP2), PNG, Lossless JPEG (LS-JPG) and our DPCM algorithms that compress the difference with Huffman coding (HUFF), Arithmetic Coding (AC) and Golomb coding with m parameter estimator EE1. Below $C_R = 0.2$ the performance of the Shrink-LLC-VDSR algorithm is better than L-DCT-ZZ-JPEG-RD and JPEG. IMDN has the potential to improve the VDSR results.

6.5.4 Quantization and entropy coding of state-of-the-art CS CNNs

One of the stated advantages of random CS is that each measurement is independent and contributes equally to the reconstructed image quality [74]. This means, however, that quantization and entropy-coding will not contribute significantly to the Rate Distortion performance.

A uniform quantizer was implemented to quantize CS-Net [19], DR2-Net [46] and ISTA-Net+ [20], modifying the code downloadable from the respective authors' websites. The PSNR results for the quantized schemes are shown in figure 6.15 (a) and show that CS-Net provides the better performance down to 0.6 bpp with DR2-Net performing better below 0.6 bpp, mainly because the CS measurements tolerate a higher level of quantization than the other two schemes.

6.5.5 Comparison with CNN-based, image CS algorithms

Our L-DCT-ZZ-{JPEG|TAB}-[IDA]-RD and Shrink-PNG-{Bicubic|IMDN|VDSR} algorithms are compared with three CNN-based CS methods in figures 6.16(a) and 6.16(b), alongside JPEG 2000 and JPEG, on Set14. Colour images are converted to grayscale using the Matlab *rgb2ycbcr*

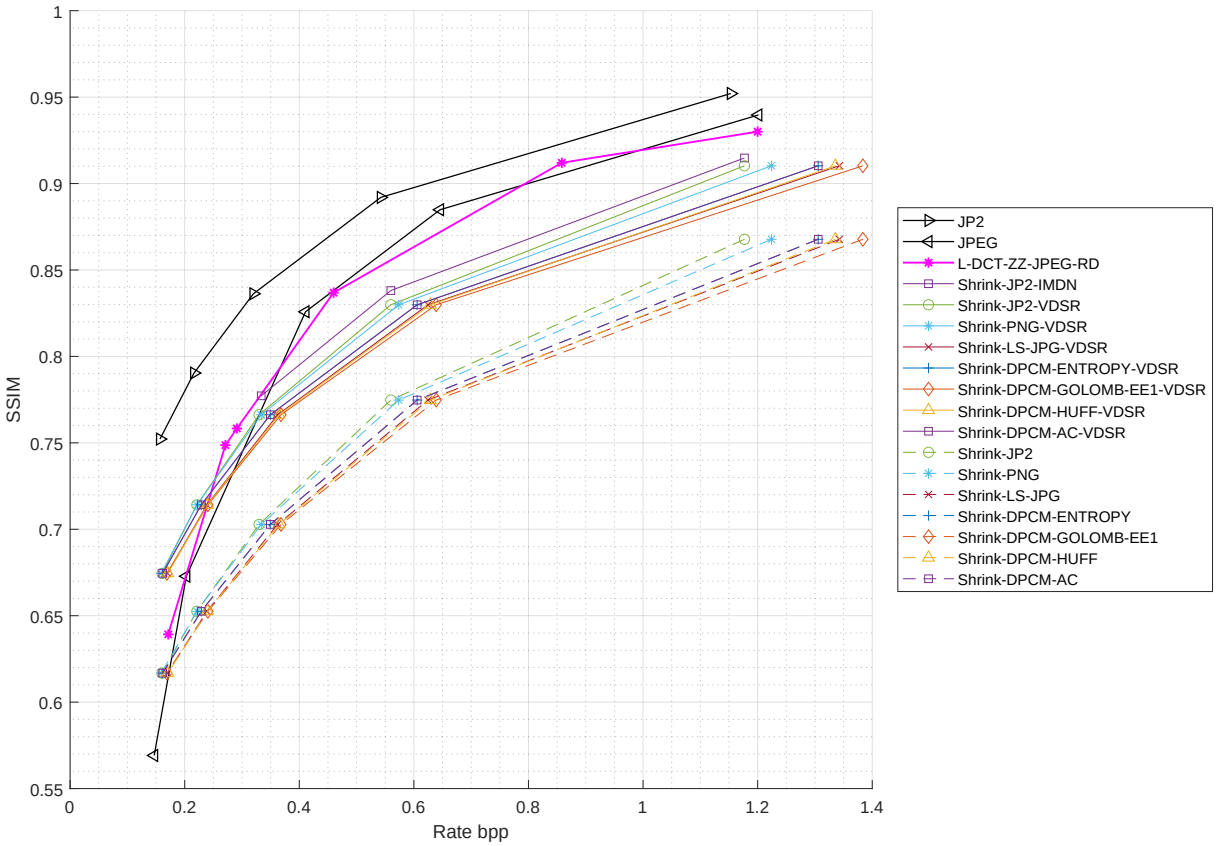


Figure 6.14: SSIM for Set14 - JPEG 2000 (JP2), JPEG, L-DCT-ZZ-JPEG, Shrink-LLC-IMDN, Shrink-LLC-VDSR. The plots show that VDSR upsampling results in a significant improvement in SSIM over bicubic upsampling, for the losslessly coded (LLC) variants following downsampling using Shrink. The Lossless compression schemes investigated are JPEG 2000 (JP2), PNG, Lossless JPEG (LS-JPG) and our DPCM algorithms that compress the difference with Huffman coding (HUFF), Arithmetic Coding (AC) and Golomb coding with m parameter estimator EE1. Below C_R = 0.2 the performance of Shrink-LLC-VDSR algorithm is better than L-DCT-ZZ-JPEG-RD and JPEG. IMDN has the potential to improve the VDSR results.

function.

All our algorithms exceed the three CNN-based CS methods on both PSNR and SSIM. Shrink-PNG-VDSR performs 2 dB better than CS-Net at a rate of 0.8 bpp. Its performance is similar to L-DCT-ZZ-TAB-RD with PSNR and SSIM marginally better below 0.55 bpp. Shrink-PNG-IMDN performs marginally better than Shrink-PNG-VDSR. However IDA reconstruction allows L-DCT-ZZ-TAB-IDA-RD to exceed the Shrink-PNG-VDSR and Shrink-PNG-IMDN performance.

L-DCT-ZZ-JPEG-RD performs better than L-DCT-ZZ-TAB-IDA-RD above 0.4 bpp, matching the JPEG PSNR performance and exceeding it on SSIM. It can also be improved by up to 2 dB using IDA reconstruction. Below 0.2 bpp, Shrink-PNG-VDSR and L-DCT-ZZ-TAB-RD perform very close to each other and are only outperformed by L-DCT-ZZ-TAB-IDA-RD and JPEG 2000. IMDN was not trained below $S_F = 4$ and it is possible that Shrink-PNG-IMDN would exceed Shrink-PNG-VDSR below 0.2 bpp as well.

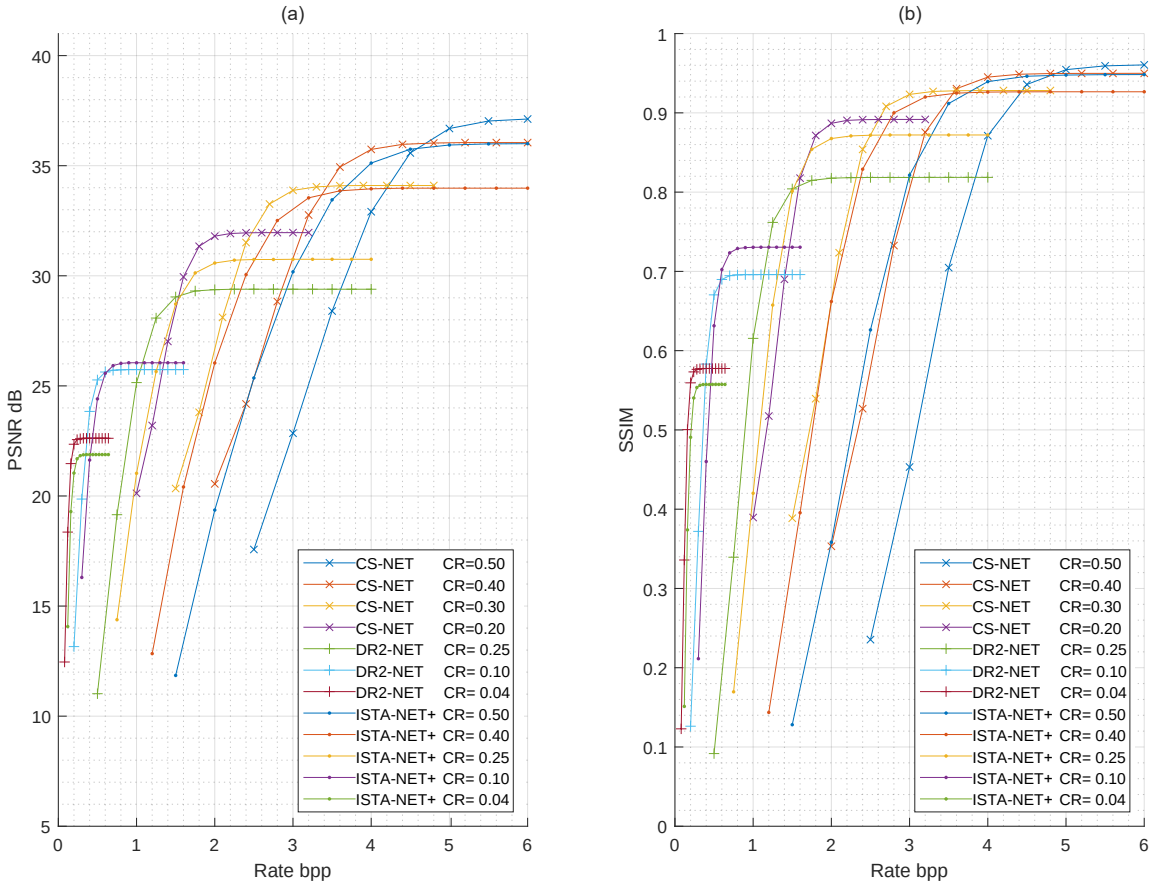


Figure 6.15: (a) PSNR vs bpp and (b) SSIM vs bpp for uniformly quantized CS schemes. The plots show that CS-NET performs best above a rate of 1.6 bpp. Below 1.6 bpp, DR2-Net performs better than ISTA-NET+ and CS-NET.

6.6 Comparison using the low-energy sensor framework

The framework developed in section 3.2 proposes nine parameters to compare low-energy image sensors. In this section these parameters are used to compare the CS and SR end-to-end systems proposed in this chapter. To compare the techniques fairly, a grayscale image set was created, composed of the Set11 [18] images cropped to 480×480 resolution. The set was generated starting from colour 512×512 pixel images from Set11, cropping them to 480×480 pixels, and then converting them to grayscale using the Matlab *rgb2ycbcr* function. This resolution was chosen so that the images would be exactly divisible by the subsampling factors $S_F = \{2,3,4,5,6\}$ and by the block sizes $\{8,16,30,32\}$. This was not efficiently executable with Set14.

6.6.1 Rate/Distortion performance

The rate distortion performance of L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-IDA-RD, L-DCT-ZZ-TAB-RD, Shrink-Golomb-IMDN, Shrink-Golomb-VDSR and Shrink-Golomb-Bicubic are compared with each other, and with JPEG and JPEG 2000 in figures 6.17(a) and 6.17(b). L-DCT-ZZ-JPEG-IDA-RD and L-DCT-ZZ-JPEG-RD algorithms use an 8×8 block size, whereas L-DCT-ZZ-TAB-IDA-RD and L-DCT-ZZ-TAB-RD use 32×32

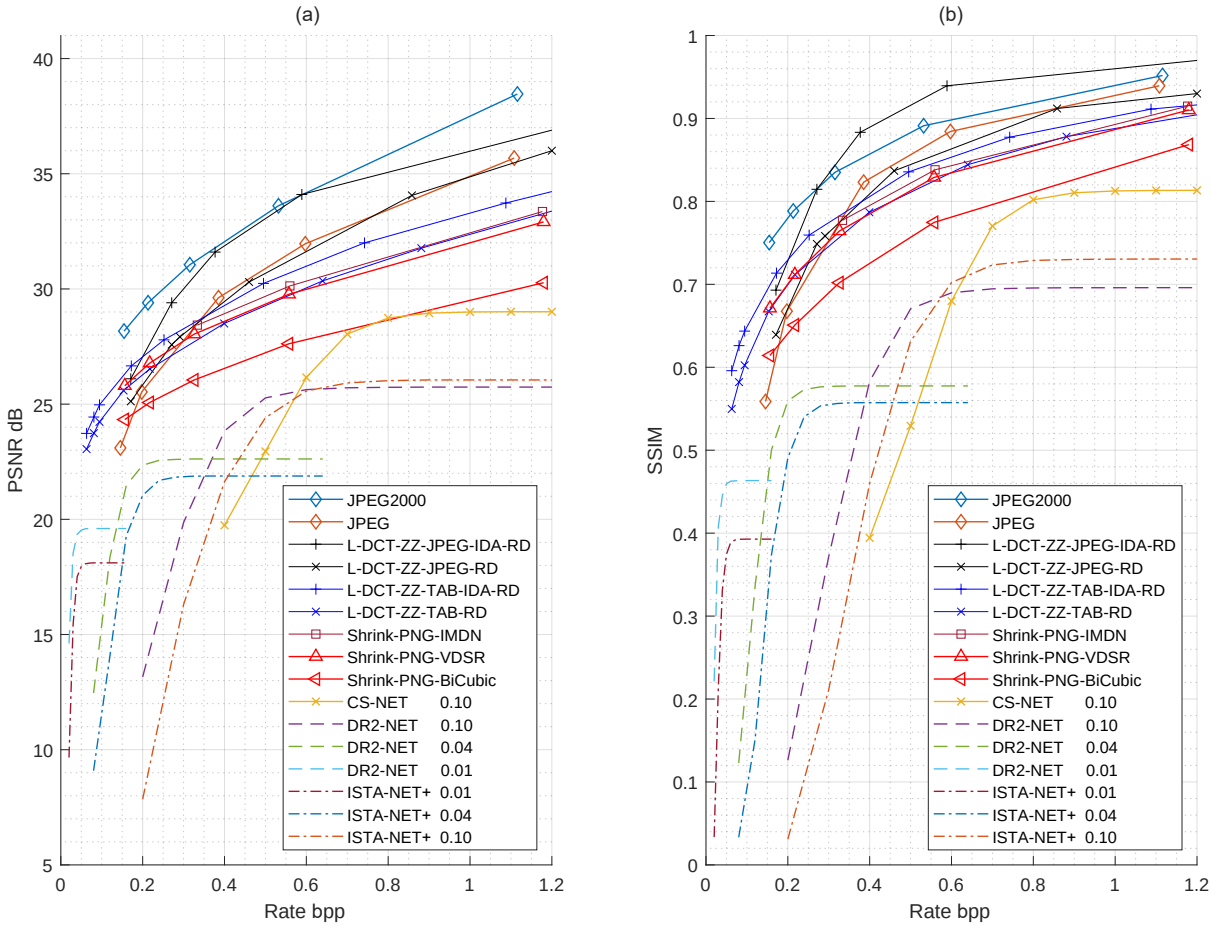


Figure 6.16: Comparison of (a) PSNR and (b) SSIM for L-DCT-ZZ-{JPEG|TAB}-[IDA]-RD, Shrink-PNG-{Bicubic|IMDN|VDSR}, JPEG 2000, JPEG, and CS CNN-based schemes on Set14 (a) PSNR (b) SSIM. The plots show that our L-DCT-ZZ-based and Shrink-based schemes perform better than the uniformly quantized CS CNN-based schemes, even the real-time schemes that do not employ IDA reconstruction.

blocks. The Shrink algorithms use the low complexity Golomb coder with the EE1 m estimation algorithm and block sizes of 30×30 and 60×60 for $S_F = 4$ only.

L-DCT-ZZ-JPEG-RD matches the PSNR performance of JPEG, performing slightly better below a rate of 0.8 bpp and slight worse above. This is due to the quantization matrix employed which is better than the default JPEG quantization matrix. When using L-DCT-ZZ-JPEG-IDA-RD with IDA reconstruction, the PSNR increases by between 2.5 dB to 4.5 dB and is better than JPEG 2000 between 0.15 bpp and 0.55 bpp.

If real-time operation is important, than above 0.2 bpp, it is possible to use L-DCT-ZZ-JPEG-RD at JPEG performance and improve the quality to JPEG 2000 quality by post-processing using IDA, if required. For real-time operation below 0.2 bpp, Shrink-Golomb-VDSR performs better than L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-TAB-RD. L-DCT-ZZ-TAB-RD performs better than L-DCT-ZZ-JPEG-RD below 0.15 bpp. IDA reconstruction allows L-DCT-ZZ-TAB-IDA-RD to match the Shrink-Golomb-VDSR PSNR performance below 0.45 bpp and exceed it above this value. However, L-DCT-ZZ-JPEG-RD PSNR improves best with IDA, as explained above.

The SSIM performance of the algorithms follows the PSNR performance closely, expect

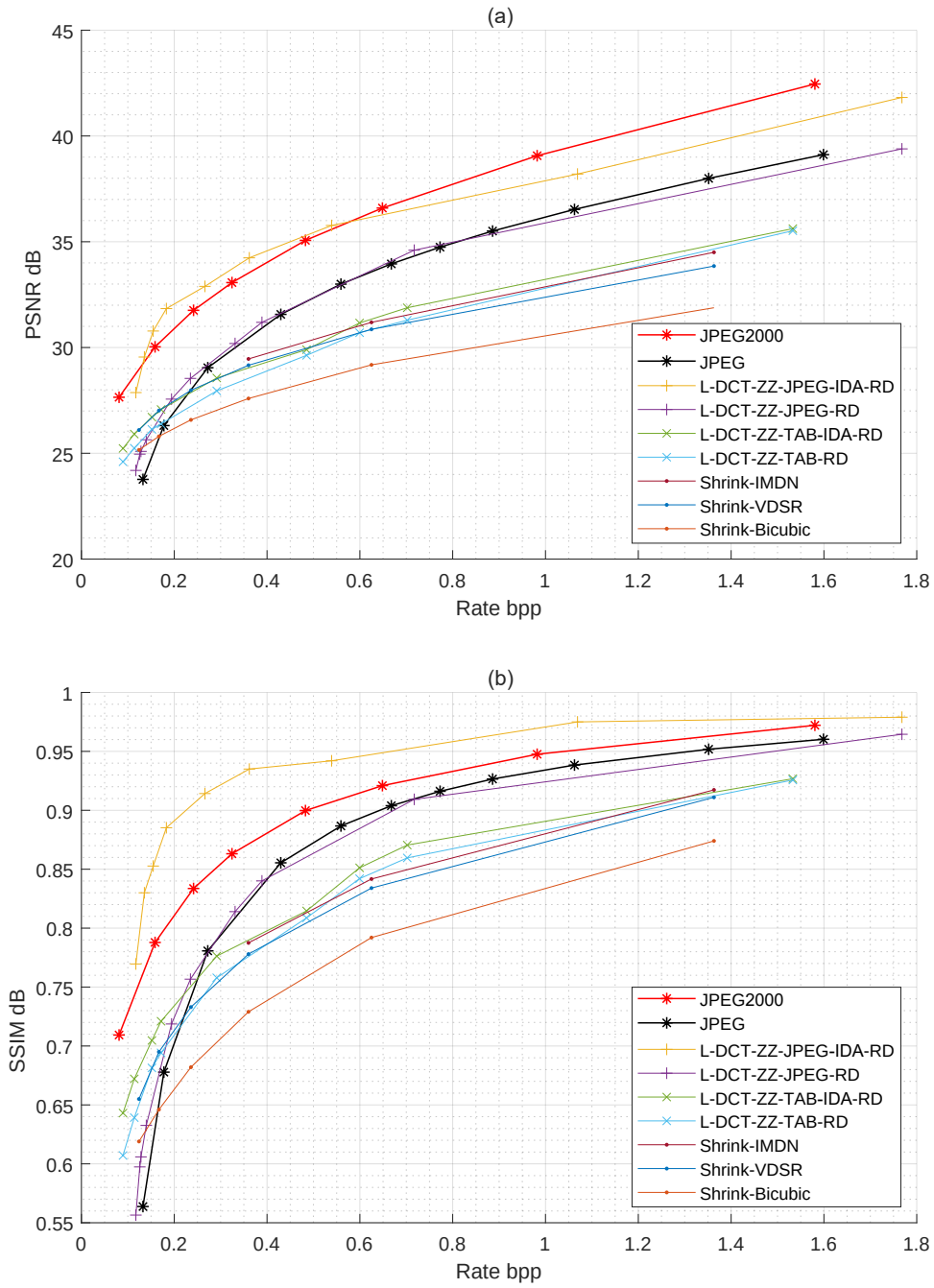


Figure 6.17: Comparison of JPEG 2000, JPEG, L-DCT-ZZ-JPEG-IDA-RD, L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-IDA-RD, L-DCT-ZZ-TAB-RD, Shrink-Golomb-IMDN, Shrink-Golomb-VDSR and Shrink-Golomb-Bicubic on Set480. The plots investigate the performance at low bit rates of interest for both real-time and IDA reconstruction. Below 0.2 bpp, the best real-time performance is provided by Shrink-VDSR. However, above 0.2 bpp, L-DCT-ZZ-JPEG-RD provides the best performance. The non-real time L-DCT-ZZ-JPEG-IDA-RD returns the best performance by far, exceeding the JPEG 2000 PSNR between 0.15 bpp and 0.55 bpp, and its SSIM above 0.1 bpp.

that the SSIM performance of L-DCT-ZZ-JPEG-IDA-RD is remarkable, reaching 0.94 and exceeding JPEG 2000's 0.87 at 0.6 bpp. IDA reconstruction can be used to improve JPEG images considerably.

The rate distortion points for the RD algorithms in figure 6.17 were obtained from the same set of subsampling factors $S_F = \{2,3,4,5,6,7\}$ as summarized in table 6.3. However, the low-complexity coding techniques applied to the measurements, result in different rates and hence

Algorithm		L-DCT-ZZ-JPEG-RD				L-DCT-ZZ-TAB-RD				Shrink-Golomb-EE1			
Reconstruction		IDCT		IDA		IDCT		IDA		Bicubic		VDSR	
S_F	C_R	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
2	0.2500	0.3311	30.19	0.5392	35.77	0.7021	31.29	0.7021	31.88	1.363	31.88	1.363	33.85
3	0.1111	0.1942	27.56	0.3618	34.24	0.4851	29.63	0.4851	29.90	0.625	29.18	0.625	30.86
4	0.0625	0.1283	25.09	0.2661	32.89	0.1716	26.39	0.1716	27.07	0.360	27.59	0.360	29.16
5	0.0400	0.1258	24.96	0.1357	29.55	0.1139	25.24	0.1139	25.91	0.236	26.58	0.236	27.98
6	0.0278	0.1173	24.19	0.1173	27.87	0.0896	24.60	0.0896	25.23	0.167	25.80	0.167	27.02

Table 6.3: Comparison of PSNR (dB) and SSIM of L-DCT-ZZ-JPEG-RD, L-DCT-ZZ-TAB-RD, and Shrink-GOLOMB-EE1 on Set480 at different rates in bpp generated by different compression ratios $CR = 1/S_F^2$. These points are plotted in figure 6.17.

give different rate distortion values. Some of the points from figure 6.17 are reproduced in table 6.3.

6.6.2 Energy efficiency

The rate distortion performance of the algorithms studied in this section is accompanied by a substantial reduction in compressive measurements and Analogue to Digital Converter (ADC) conversions in comparison with classical JPEG and JPEG 2000 compression, where all the pixels are required. The analysis framework now requires us to compare the energy efficiency of the six algorithms.

The energy required to capture and transmit one frame in a compressive system is given by equation 3.2. The energy to compressively sense one frame is given by the first two terms, that is $E_{CAP}^{'} = E_S^{'} + E_{ADC}^{'}$. It is also given by equation 3.15, that is $E_{CAP} = N_M.\varepsilon_M + N_{ADC}.\varepsilon_{ADC}$. The values for N_M and N_{ADC} can be calculated using the equations in table 6.4. ε_M and ε_{ADC} would depend on the hardware used to implement a system.

The L-DCT-ZZ-JPEG algorithm requires 8×8 blocks whereas L-DCT-ZZ-TAB uses 32×32 blocks. In the case of the Shrink algorithm, the size of the MPC or Virtual MPC (vMPC) block is chosen to be 30 so that it is possible to compare $E_{CAP}^{'}$ on 480×480 images, at compression factors $1/S_F^2$ where $S_F = \{2,3,5,6\}$, these values being factors of 30. When using $S_F = 4$, the block size changes to 60×60 since $B = 60$ is exactly divisible by $S_F = 4$. A size of 480×480 exactly accommodates blocks of size 30×30, 32×32 and 60×60.

Table 6.5 reports the measurement matrix storage requirements (N_{MEM}), energy-related parameters (N_M and N_{ADC}) and capture time (N_C) parameter, when the six algorithms in section 6.6.1 compress Set480 with subsampling factors $S_F = \{2,3,4,5,6\}$ and block sizes 8, 16 and 32 for L-DCT-ZZ-related algorithms and 30 and 60 for the Shrink algorithms. These factors do not depend on subsequent low-complexity compression schemes adopted such as JPEG-style, TAB or DPCM-Golomb-EE1.

To capture a 480×480 grayscale image at $S_F = 2$, that is $C_F = S_F^2 = 4$, on vMPC or MPC, L-DCT-ZZ algorithms require $N_M = 3.386$ million mirror modulations when the block size is 8×8 and the number increases in proportion to the number of pixels in a block B^2. N_M is also

	L-DCT-ZZ		Shrink
SPC	$N_{MEM} = H_c.W_c.M$ $N_M = H_c.W_c.M$ $N_{ADC} = M$ $N_C = M$		$N_{MEM} = K^2$ $N_M = M.K^2$ $N_{ADC} = 2.M$ $N_C = 2.M$
	L-DCT-ZZ-JPEG	L-DCT-ZZ-TAB	Shrink
vMPC on SPC	$N_{MEM} = M_B.B^2$ $N_M = M_B.n_B.B^2$ $N_{ADC} = M_B.n_B$ $N_C = M_B.n_B$	$N_{MEM} = M_B.B^2$ $N_M = M_B.n_B.B^2$ $N_{ADC} = M_B.n_B$ $N_C = M_B.n_B$	$N_{MEM} = K^2$ $N_M = M_B.n_B.K^2$ $N_{ADC} = 2.M_B.n_B$ $N_C = 2.M_B.n_B$
	L-DCT-ZZ-JPEG	L-DCT-ZZ-TAB	Shrink
MPC	$N_{MEM} = M_B.B^2$ $N_M = M_B.n_B.B^2$ $N_{ADC} = M_B.n_B$ $N_C = M_B$	$N_{MEM} = M_B.B^2$ $N_M = M_B.n_B.B^2$ $N_{ADC} = M_B.n_B$ $N_C = M_B$	$N_{MEM} = K^2$ $N_M = M_B.n_B.K^2$ $N_{ADC} = 2.(n_S^2 + 8.n_S + 16).n_B$ $N_C = 2.(n_S^2 + 8.n_S + 16)$

Table 6.4: Comparison of parameters for L-DCT-ZZ-JPEG, L-DCT-ZZ-TAB and Shrink algorithms.

proportional to the number of measurements M over the whole image. When Shrink-related algorithms are implemented on vMPC and MPC, N_{ADC} and N_C are both equal to twice the number of measurements over the whole image. The situation changes when these are implemented on an MPC, with both parameters increasing significantly because of the bicubic interpolation and anti-aliasing kernels extending into adjacent blocks close to the block boundaries.

Table 6.5 shows that on a vMPC, L-DCT-ZZ requires the least energy to modulate the Digital Micromirror Devices (DMDs) with a block size of 8×8, and the required energy increases with the block size. The energy required by Shrink remains rather constant at around 4 million mirror modulations, because as the number of measurements M decreases the kernel size $K \times K$ increases. The energy required by L-DCT-ZZ to perform ADC conversions does not change with block size and is equal to the number of measurements M. Shrink requires approximately twice this energy, as it performs two ADC conversion operations per measurement, one for the positive kernel coefficients and the other for the negative ones.

On an MPC, 8×8 L-DCT-ZZ requires the least energy to modulate mirrors followed by 16×16 L-DCT-ZZ, and then by 32×32 L-DCT-ZZ. The Shrink algorithm matches the energy requirements of L-DCT-ZZ with 8×8 blocks at $S_F = 2$, but then its requirements increase, exceeding those of L-DCT-ZZ with 16×16 blocks above $S_F = 4$. The energy required by L-DCT-ZZ for ADC conversion is still M, irrespective of the block size. However, the energy required by Shrink is substantially worse because of the mismatch between the Bicubic Convolutional Filter (BCF) kernel size and the MPC block size, besides the requirement of two ADC conversions per measurement to collect the positive and negative components separately. Shrink requires more than three times the energy for ADC conversions than L-DCT-ZZ at the same compression factor.

The energy to transmit an image depends on the bits per pixel, and L-DCT-ZZ-JPEG algorithms generate the lowest bits per pixel as shown in table 6.3, followed by L-DCT-ZZ-TAB algorithms and then Shrink algorithms.

It is finally required to compare E'_{LL} in equation 3.2. Computing E'_{LL} is non-trivial since this depends on the complexity of the algorithms, the optimization of the implementation, and the energy efficiency of the computing platform on which the low-complexity algorithms run. In this thesis, we report the execution time as a measure of complexity for the algorithms we implement and test directly. This is the common practice in the literature.

6.6.3 Frame capture time

The time T_F required to capture, encode and transmit an image or frame is given by equation 3.3, that is $T'_F = T'_M + T'_{ADC} + T'_{LL} + T'_{TX}$. $T_M{}'$ is the time to modulate the mirrors in DMD based SPCs and MPCs , or to perform analogue computations in Focal Plane Processing Sensors (FPPS). Therefore, the number of mirror modulations reported here, also applies in the context of FPPS and other sensors introduced in section 2.4.

In our framework, in SPCs and vMPCs, the time to sense an image is calculated by the total number of mirror modulations N_M multiplied by a notional time to move one mirror Δ_M, that is $T'_M = N_M.\Delta_M$. This is a worst case scenario because actual DMD devices can move multiple mirrors at a time, (for example Texas Instruments' DLP9500 DMD [131] can move one row of 1920 mirrors at a time).

In the case of L-DCT-ZZ algorithms, capturing images on MPC instead of an SPC or vMPC, allows the N_M mirror modulations and N_{ADC} ADC conversions to happen in parallel in n_B blocks. Thus the capture time T_{CAP} is reduced by a factor n_B, so that the time taken to sense an image is given by $T'_M = \frac{N_M.\Delta_M}{n_B}$ where n_B is the number of blocks.

The capture times in SPC, MPC and vMPC implementations for both CS and SR algorithms are given by equations 3.16, 3.17 and 3.18 respectively, and include the analogue to digital conversion time in addition to the mirror modulation time.

In Shrink, the number of ADC conversion cycles N_C is given by equation 3.12. During each cycle, $K \times K$ mirror modulations are contained in 1, 2 or 4 blocks. An upper limit to the number of mirror modulations that have to happen in parallel, in each block, is therefore given by $N_C \times K^2$. This is an overestimate because not all blocks have to modulate $K \times K$ mirrors in each cycle.

6.7 Conclusions

All the real-time L-DCT-ZZ-based and Shrink-based, end-to-end image compressed sensing algorithms perform substantially better than the CNN-based CS-NET, DR2-NET and ISTA-NET+ image CS algorithms in the literature.

Above $C_R = 0.2$, the best PSNR results are produced by the non-real time L-DCT-ZZ-JPEG-IDA-RD algorithm, with a block size of 8×8 pixels, that approaches the performance of JPEG

Algorithm	SPC & MPC $H_c = 480$, $W_c = 480$, $N = 230,400$								vMPC on SPC				MPC			
	S_F	B	C_F	n_S	M	K	M_B	n_B	N_{MEM}	N_M	N_{ADC}	N_C	N_{MEM}	N_M	N_{ADC}	N_C
L-DCT-ZZ	2	8	4.0	-	57,600	-	16	3600	1,024	3,686,400	57,600	57,600	1,024	3,686,400	57,600	16
L-DCT-ZZ	2	16	4.0	-	57,600	-	64	900	16,384	14,745,600	57,600	57,600	16,384	14,745,600	57,600	64
L-DCT-ZZ	2	32	4.0	-	57,600	-	256	225	262,144	58,982,400	57,600	57,600	262,144	58,982,400	57,600	256
Shrink	2	30	4.0	15	57,600	8	225	256	64	3,686,400	115,200	115,200	64	3,686,400	184,832	722
L-DCT-ZZ	3	8	9.1	-	25,200	-	7	3600	448	1,612,800	25,200	25,200	448	1,612,800	25,200	7
L-DCT-ZZ	3	16	9.1	-	25,200	-	28	900	7,168	6,451,200	25,200	25,200	7,168	6,451,200	25,200	28
L-DCT-ZZ	3	32	9.0	-	25,650	-	114	225	116,736	26,265,600	25,650	25,650	116,736	26,265,600	25,650	114
Shrink	3	30	9.0	10	25,600	13	100	256	169	4,326,400	51,200	51,200	169	4,326,400	100,352	392
L-DCT-ZZ	4	8	16.0	-	14,400	-	4	3600	256	921,600	14,400	14,400	256	921,600	14,400	4
L-DCT-ZZ	4	16	16.0	-	14,400	-	16	900	4,096	3,686,400	14,400	14,400	4,096	3,686,400	14,400	16
L-DCT-ZZ	4	32	16.0	-	14,400	-	64	225	65,536	14,745,600	14,400	14,400	65,536	14,745,600	14,400	64
Shrink	4	60	16.0	15	14,400	16	225	64	256	3,686,400	28,800	28,800	256	3,686,400	46,208	722
L-DCT-ZZ	5	8	21.3	-	10,800	-	3	3600	192	691,200	10,800	10,800	192	691,200	10,800	3
L-DCT-ZZ	5	16	25.6	-	9,000	-	10	900	2,560	2,304,000	9,000	9,000	2,560	2,304,000	9,000	10
L-DCT-ZZ	5	32	25.0	-	9,225	-	41	225	41,984	9,446,400	9,225	9,225	41,984	9,446,400	9,225	41
Shrink	5	30	25.0	6	9,216	21	36	256	441	4,064,256	18,432	18,432	441	4,064,256	51,200	200
L-DCT-ZZ	6	8	32.0	-	7,200	-	2	3600	128	460,800	7,200	7,200	128	460,800	7,200	2
L-DCT-ZZ	6	16	36.6	-	6,300	-	7	900	1,792	1,612,800	6,300	6,300	1,792	1,612,800	6,300	7
L-DCT-ZZ	6	32	36.6	-	6,300	-	28	225	28,672	6,451,200	6,300	6,300	28,672	6,451,200	6,300	28
Shrink	6	30	36.0	5	6,400	24	25	256	576	3,686,400	12,800	12,800	576	3,686,400	41,472	162

Table 6.5: Comparison of CS (e.g. L-DCT-ZZ) and SR (e.g. Shrink) on a 480×480 image with varying block size B and sub-sampling factor S_F. C_F is the measured compression factor, $n_S = B/S_F$ is the number of Shrink sub-samples in one-dimension of a block of size B at shrink factor S_F, M is the number of compressed measurements, K is the one-dimensional filter size in Shrink, M_B is the number of measurements in a block, n_B is the number of blocks, N_{MEM} is the memory size to store the measurement matrix, N_M is the number of DMD mirror modulations, N_{ADC} is the number of ADC conversions and N_C is the number of ADC conversion cycles to capture one image.

2000. The SSIM performance of L-DCT-ZZ-JPEG-IDA-RD exceeds that of JPEG 2000.

Below $C_R = 0.2$, the best results are produced by the non-real time L-DCT-ZZ-TAB-IDA-RD algorithm that uses a block size of 32×32 pixels. However, the real-time L-DCT-ZZ-TAB-RD and Shrink-PNG-VDSR algorithms also perform very well in this region.

When quantized, the performance of L-DCT-ZZ becomes equivalent to an adaptive scheme because low valued transform coefficients are quantized to zero. Therefore, the performance of L-DCT-{ZZ|BBV|DD} with 8×8 pixel blocks and 2D IDCT reconstruction, is the same under JPEG-style encoding. The performance also matches that of normal JPEG compression, but with a significant reduction in the number of transform coefficients that need to be collected from the compressive image sensor, thereby reducing energy requirements. IDA reconstruction improves PSNR and SSIM substantially above JPEG's and with the same advantage with respect to the lower transform coefficients measured.

The block-based L-DCT-ZZ-TAB algorithm, with 32×32 pixel blocks and 2D IDCT reconstruction, performs better than JPEG and L-DCT-ZZ-JPEG below a rate of 0.3 bits per pixel. IDA reconstruction improves PSNR by about 1 dB and SSIM from 0.740 to 0.775 at 0.3bpp, without increasing the rate.

From a PSNR and SSIM quality perspective, a possible strategy would be to use L-DCT-ZZ-TAB at low compression ratios below 0.5 and L-DCT-ZZ-JPEG at higher compression ratios.

However, the compressive image sensor would have to be capable of collecting measurements with two different block sizes.

The IDCT reconstructed algorithms can be used for real-time image and video sensing and transmission, whereas IDA reconstruction can improve quality substantially but not in real-time. Therefore this is useful as a post-processing technique.

Subsampling followed by super-resolution, for example using IMDN and VDSR, provides better quality than L-DCT-ZZ-JPEG-RD and L-DCT-ZZ-TAB-IDA-RD at rates below 0.2 bpp, and in real-time. The technique should also be investigated for real-time video compressive systems.

The substantial improvement in performance provided by AL-DCT-BBV and AL-DCT-DD over L-DCT-ZZ in the dimensionality reduction application studied in chapter 4, especially with IDA reconstruction, did not materialise in this chapter. This is due to the quantization process rendering all three techniques equal. However, this may not be the case when the techniques are embedded in video coding algorithms, where temporal correlation may be leveraged, and hence adaptive algorithms need to be investigated further.

Chapter 7

Distributed video compressive sensing and video super resolution algorithms

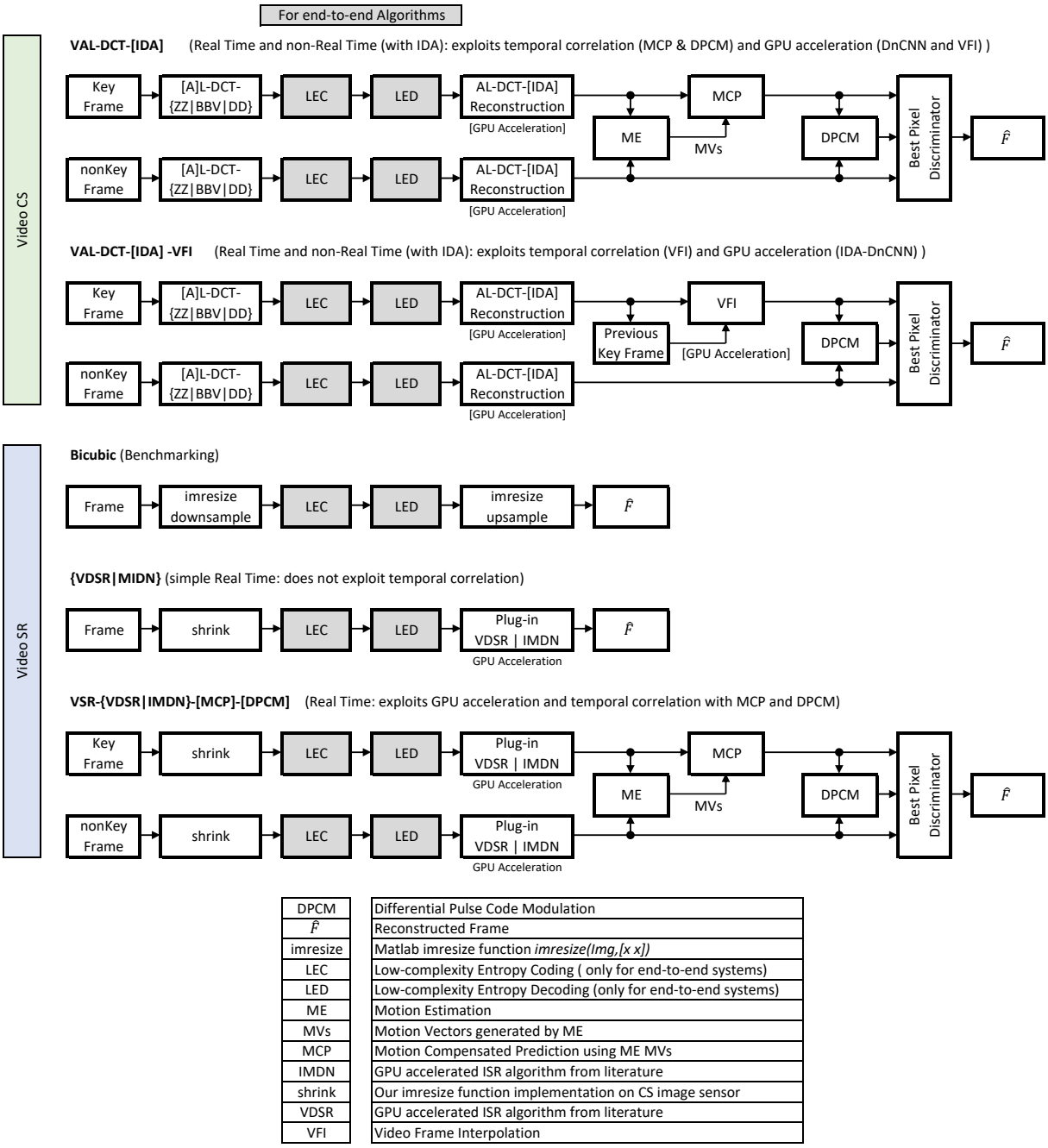
7.1 Introduction

In this chapter, the image compressive sensing algorithms developed in chapter 4 and the image super resolution algorithms re-purposed for compression in chapter 5 are integrated into distributed video coding algorithms. In distributed video encoding, the encoder is designed to be as lightweight as possible, with computationally intensive operations such as Motion Estimation (ME) and Video Frame Interpolation (VFI), performed at the decoder side. The Video Compressive Sensing (VCS) and Video Super Resolution (VSR) algorithms depicted in figure 7.1 are studied in this chapter from a redundancy reduction perspective. End-to-End systems are studied in chapter 8.

The video Compressive Sensing (CS) algorithms use the adaptive L-DCT algorithms for video (VAL) whereas the video Super Resolution (SR) algorithms use image super resolution techniques re-purposed for video compression. VAL algorithms can leverage Iterative Denoising Algorithm (IDA) to improve quality. VSR-{VDSR|IMDN} use either the Very Deep Super Resolution (VDSR) or Information Multi-Distillation Network (IMDN) image super resolution convolutional neural networks to 'decompress' the sub-sampled images from the encoder.

The two algorithms adopt a Group of Pictures (GOP) structure with one initial key frame that is transmitted with higher quality than the non-key frames in the GOP. At the decoder, motion estimation followed by Motion Compensated Prediction (MCP), or VFI, is used to estimate high-quality predictions of the non-key frames, from the higher quality key frames. The non-key frames are combined with their higher resolution estimates using Differential Pulse Code Modulation (DPCM) implemented in the Discrete Cosine Transform (DCT) domain. The MCP frames, the DPCM frames and the non-key frames are processed by a Best Pixel Discriminator (BPD) block that assess the quality of the pixels from the three sources and outputs the pixel that is deemed to possess the best quality.

All the algorithms developed operate in real-time, except for those that use IDA to improve the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM)



DPCM	Differential Pulse Code Modulation
$\hat{F}$	Reconstructed Frame
imresize	Matlab imresize function *imresize(Img,[x x])*
LEC	Low-complexity Entropy Coding (only for end-to-end systems)
LED	Low-complexity Entropy Decoding (only for end-to-end systems)
ME	Motion Estimation
MVs	Motion Vectors generated by ME
MCP	Motion Compensated Prediction using ME MVs
IMDN	GPU accelerated ISR algorithm from literature
shrink	Our imresize function implementation on CS image sensor
VDSR	GPU accelerated ISR algorithm from literature
VFI	Video Frame Interpolation

Figure 7.1: Block diagram of the distributed video CS and video SR schemes proposed and studied in this chapter 7. End-to-End systems are studied in chapter 8.

or Multiscale SSIM (MS-SSIM) quality. Real-time operation is achieved by using Graphics Processing Unit (GPU) accelerated plug-in algorithms such as VDSR [106], IMDN [109] and Depth-aware Video Frame Interpolation (DAIN) [132]. Even IDA uses the GPU-accelerated Denoising CNN (DnCNN) denoiser but the iterative nature of IDA still means that the algorithm is non-real-time, although it does achieve State-of-the-Art (SOTA) results amongst video CS and video SR algorithms published in the literature.

The MS-SSIM [69] measure is used to report perceived quality in sections 7.4 and 7.5.1, instead of the SSIM measure used in the rest of the thesis. This is necessary to compare the results of the VAL-MCP and VAL-VFI algorithms proposed in this chapter with the state-of-the-art results published in [24] since the Structural Group Sparsity and Optical Flow estimation

(SGS-OF) code was not available.

Algorithms

Specifically, the following distributed video compressive sensing algorithms are developed in this chapter.

- VAL-MCP is the main distributed video compressive sensing algorithm using a Group of Pictures with key and non-key frames, AL-DCT-DD or L-DCT-ZZ on all frames at the encoder, and optionally exploiting temporal correlation at the decoder using block-based ME, MCP and DPCM. The algorithm operates in real-time.
- VAL-IDA-MCP is the VAL-MCP algorithm using IDA to reconstruct images prior to MCP and DPCM, but not in real-time.
- VAL-MCP-ZZ is the VAL-MCP algorithm using L-DCT-ZZ instead of AL-DCT-DD on the frames at the encoder.
- VAL-VFI is similar to the VAL-MCP algorithm but replaces the block-based ME and MCP operations with a GPU-based VFI Convolutional Neural Network (CNN).
- VAL-IDA-VFI is the VAL-VFI algorithm using IDA to reconstruct images prior to VFI and DPCM.
- The VAL algorithms use algorithms THB2 and THI2 to estimate the number of transform coefficients to collect from each block in key frames. THB2 collects the same number of transform coefficients from each block, whereas THI2 collects a different number of transform coefficients from each block.
- The VAL algorithms exploit temporal correlation to better estimate the number of transform coefficients to collect from each block in non-key frames using three algorithms; RDD, PDD and MDD. RDD uses the key frame transform coefficients to estimate the number of transform coefficients to collect from each block. PDD uses the previous frame transform coefficients for the same purpose. MDD uses both the key frame and current frame transform coefficients to estimate the number of transform coefficients per block.
- Shrink-Bicubic, Shrink-SRCNN, Shrink-VDSR and Shrink-IMDN use the Shrink algorithm to compress frames independently at the encoder and reconstruct them at the decoder, without exploiting temporal correlation at the decoder using Bicubic interpolation or the GPU-based SRCNN, VDSR or IMDN CNNs.
- VSR-Bicubic, VSR-VDSR and VSR-IMDN are distributed video compressed sensing algorithms that use the shrink algorithm to compress frames at the encoder and then use

Bicubic interpolation or the GPU-based VDSR or IMDN CNNs SR algorithms. Block-based ME, MCP and DPCM are used to exploit temporal correlation at the decoder. Key frames and non-key frames use the same compression ratio.

- VSR-MCP is a family of distributed video compressive sensing algorithms that uses the Shrink algorithm to down-sample key and non-key frames at the encoder, up-samples them at the decoder, and exploits temporal correlation at the decoder using block-based ME, MCP, and DPCM. Key-frames use a larger compression ratio than non-key frames in this algorithm. VSR-Bicubic-MCP, VSR-VDSR-MCP and VSR-IMDN-MCP use Bicubic interpolation or the GPU-based VDSR or IMDN CNNs to up-sample the frames.
- VSR-VFI is a family of distributed video compressive sensing algorithms that includes VSR-Bicubic-VFI, VSR-VDSR-VFI and VSR-IMDN-VFI, similar to VSR-MCP but replacing the block-based ME and MCP with a GPU-based VFI CNN.

7.2 Related work

Fowler and Mun [133] proposed the MC-BCS-SPL algorithm which incorporates block-based ME and Motion Compensation (MC) at the decoder. In a GOP with P pictures, they recover the first $P/2$ frames with forward reconstruction starting from the intra-frame, and the remaining frames, by reconstructing them from the reference frame in the next GOP. They then refine the frames by bi-directional prediction from the previous and following frames. Chen et al. [50] took inspiration from the this framework and proposed multi-hypothesis predictions (BCS-SPL-MH) as an improvement for both still images and video. An initial reconstruction is performed using standard BCS-SPL [133], and weighted Tikhonov regularization is then used to form predictions of the residual to iteratively improve the reconstruction.

Recently, Zhao et al. [17] proposed a two-phase hybrid intra-frame, inter-frame algorithm. The first exploits spatial correlation to produce high-quality reference frames. In the second, the reweighted sparsity of the residual difference between frames is recovered using an algorithm based on split Bregman iteration. Experimental results are presented for the first 16 frames of six popular commonCommon Intermediate Format (CIF) video sequences. The GOP consists of an intra-frame followed by seven inter-frames. Block-based sensing is assumed using Gaussian random projections on 32×32 pixel blocks. Key frames are sensed at a compression ratio of 0.7 and non-key frames at 0.2. Multi-hypothesis prediction is used to reconstruct the inter-frames. The proposed Reweighted Residual Sparsity (RRS) scheme is compared against four representative video CS reconstruction methods, and was shown to achieve state-of-the-art performance. However, the reconstruction time per frame was reported to exceed 5 minutes.

SGS-OF [24] is closest to our work, both in terms of concept and results. The authors adopt a GOP with size 8 and compressively sense key and non-key frames with compression ratios 0.7 and 0.1. They propose a Structural Group Sparsity (SGS) model and employ the Alternating Direction Method of Multipliers (ADMM) variant of the Augmented Lagrangian Method (ALM),

to reconstruct the independent frames, and then use the Coarse-to-Fine optical flow method to create a fused image using the forward and backward OF motion estimates. Finally they use the BCS-SPL [133] algorithm to refine the original reconstruction using residual estimation and compensation. The authors investigate both Gaussian and partial DCT measurement matrices and report large gains with the deterministic partial DCT matrices. The authors quote a reconstruction time of around 15 s per frame using the partial DCT sensing matrix.

7.3 Distributed video compressive sensing

Inspired by the work in distributed video coding and distributed video compressive sensing techniques, reviewed in sections 7.2 and 8.2, we have developed the family of VAL-MCP algorithms described in section 7.3.2 and evaluated empirically in section 7.4.

7.3.1 Overview

In consonance with most current video coding algorithms, the VAL-MCP algorithm processes frames as a sequence of group-of-pictures, with one leading key frame and a number of non-key frames. As in other distributed compressive video coding systems in the literature [24] [133], the key frames in VAL-MCP are usually coded with a higher compression ratio and therefore have a higher quality than non-key frames.

Distributed video coding, transfers the computationally intensive operations from the encoder to the decoder. Similarly, in the VAL-MCP algorithm, key and non-key frames are captured at the encoder using the computationally simple adaptive compressive sensing techniques developed in chapter 4, to exploit spatial correlation, but the exploitation of temporal correlation is performed at the decoder.

The VAL-MCP algorithm, uses three decoder-side techniques to increase the quality of the non-key frames exploiting the higher quality of the key frames: MCP, DPCM and BPD.

In classical hybrid DPCM/DCT video encoders, temporal correlation between frames is exploited by subtracting a prediction of the current frame using previous (or future) frame information. This DPCM operation results in a difference frame that is more compressible using the spatial 2D-DCT transform. In section 7.3.6, we show that temporal DPCM between a key frame and a non-key frame can be performed at the decoder and results in an increase in quality of the predicted non-key frame.

Classical video encoding, performs motion compensated prediction at the encoder so that the resulting residual error can be better compressed prior to transmission. In the VAL-MCP algorithm, MCP is used at the decoder to increase the quality of the reconstructed non-key frames. The low quality non-key frames are split into macroblocks and motion estimation is used to estimate the position of the best corresponding block in the current and future key-frames. Then, the resulting motion vectors are used to predict the current non-key frame macroblocks from the higher-quality key-frame, improving the quality of the current key-frame.

The VAL-MCP algorithm improves the reconstruction quality by first using motion compensated prediction, followed by further quality improvement using DPCM. At this stage there are three versions of the non-key frame at the decoder: the low-quality non-key frame received from the encoder, the higher-quality motion compensation prediction $\mathbf{R}_{n,i}$ and the further improved DPCM prediction $\mathbf{D}_{n,i}$. In static areas of the image sequence, the values of the MCP and MCP-DPCM pixels should be close to each other. However, in dynamic regions the difference between the MCP and DPCM pixels will be higher, especially where block motion estimation results in image artefacts at predicted block edges. Therefore, a BPD algorithm is designed to choose the best pixel to output by comparing the pixels in $\mathbf{R}_{n,i}$ and $\mathbf{D}_{n,i}$, and if the difference is too high, it prefers the non-key frame pixel received from the encoder in $\bar{\mathbf{K}}_{n,i}$ over the pixel in $\mathbf{D}_{n,i}$. This simple algorithm is found to further improve the visual quality of the reconstructed frames.

7.3.2 VAL-MCP system block diagram

The block diagram of the proposed VAL-MCP algorithm is shown in figure 7.2. The GOP structure in figure 7.3 is used wherein key frames are sensed with a high average key sub-rate $\delta_K = M/N$ so that the reconstructed quality is high, where N is the number of pixels and M the number of measurements. Non-key frames between key frames are sensed with a substantially lower average non-key sub-rate $\delta_{\bar{K}}$. The key frames are placed at the start of the GOP and then predict the non-key frames bidirectionally from the key frames in the current and followingGOP, using MCP.

The encoder captures frames $\mathbf{F}_{n+1,i}$ using the adaptive AL-DCT-DD algorithm. $\mathbf{F}_{n+1}$ is the Key frame in GOP $n+1$. $\mathbf{F}_{n+1,i}$ is the i-th non-key frame in GOP $n+1$, $i = 2..g$ where g is the GOP size. The encoder then transmits the compressively sensed 2D-DCT transform coefficients $\mathbf{T}_{n+1,i}$ and the number of transform coefficients per block $m_{n+1,i}$ sensed in the i-th frame of the GOP.

The decoder decodes the key and non-key frames, in real-time, using the AL-DCT-DD algorithm, and buffers one GOP worth of transform coefficients and AL-DCT-DD reconstructed frames. This is necessary because the MCP algorithm requires the key frame in the current GOP $\mathbf{K}_n$ and that in the following GOP $\mathbf{K}_{n+1}$, besides the i-th non-key frame $\bar{\mathbf{K}}_{n,i}$ in the current GOP. The motion compensated predictions of the current non-key frame referenced to key frames $\mathbf{K}_n$ and $\mathbf{K}_{n+1}$ are $\mathbf{M}_{n,i}$ and $\mathbf{M}_{n+1,i}$ respectively. These are then averaged to produce the weighted average MCP reference $\mathbf{R}_{n,i}$.

The decoder then computes the temporal DPCM in the transform domain to predict the received frame from the reference frame $\mathbf{R}_{n,i}$ and non-key transform coefficients $\mathbf{T}_{n,i}$. The DPCM output $\mathbf{D}_{n,i}$ is then compared with $\mathbf{R}_{n,i}$ in the best pixel discriminator block, as described in sections 7.3.1 and 7.3.8, that outputs $\mathbf{D}_{n,i}$ or $\bar{\mathbf{K}}_{n,i}$ to produce the reconstructed non-key frame $\bar{\mathbf{F}}_{n,i}$. The reconstructed key frame $\hat{\mathbf{F}}_n$ is equal to $\mathbf{K}_n$.

Better quality can be achieved if the AL-DCT-DD reconstruction is carried out using our IDA algorithm with the DnCNN denoiser. The operations in the block diagram are described in

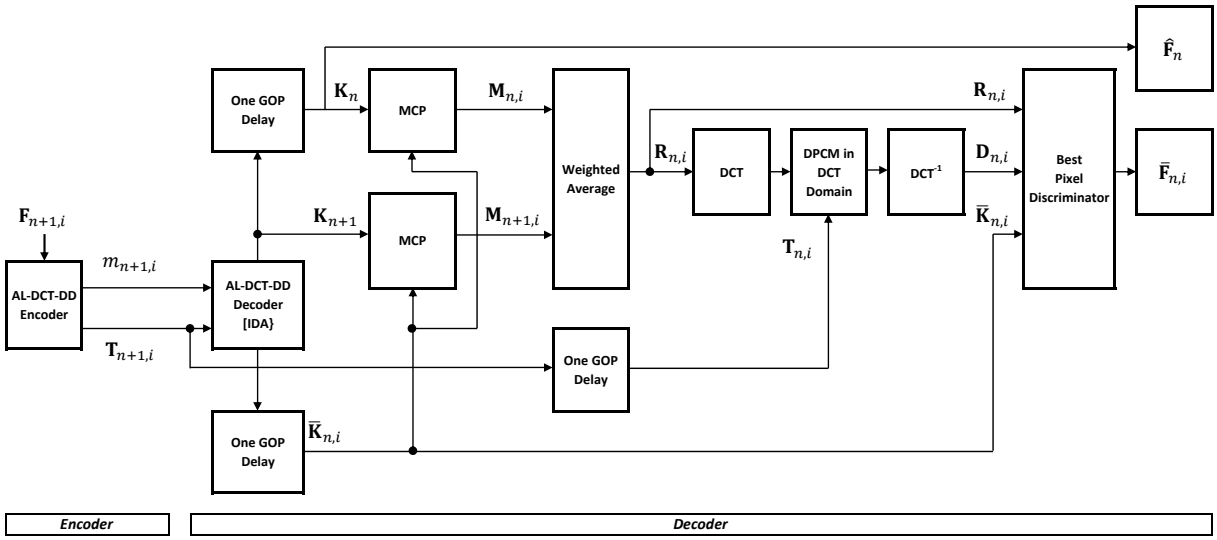


Figure 7.2: Block diagram of the distributed VAL-MCP algorithm. The AL-DCT-DD decoder has optional IDA reconstruction.

more detail below.

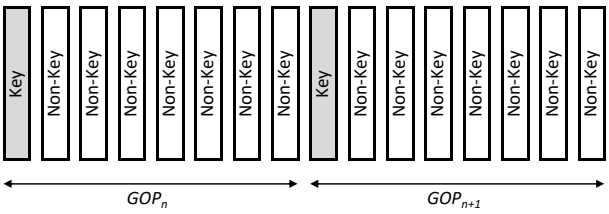


Figure 7.3: GOP Structure. n is the GOP number.

7.3.3 Adaptive block compressive sensing of video frames

Chapter 4 describes the adaptive Block Boundary Variation (BBV) and DCT Domain (DD) algorithms. BBV was found to work best at high values of $\delta = M/N$ and DD to work better at low values, and initially AL-DCT-BBV and L-DCT-ZZ were used on key frames and AL-DCT-DD on non-key frames. However, AL-DCT-DD was found to give better results than AL-DCT-BBV for the key frames in our sequences. Hence AL-DCT-DD was used throughout.

The adaptive AL-DCT-DD algorithm described in algorithm 4.5 adapts the number of phase-2 coefficients based on the number of coefficients that exceed a threshold T_B in the same block. The coefficient estimation part of this algorithm is referred to as the Threshold per Block (THB2) algorithm that is described in algorithm 7.1.

This algorithm is modified such that the number of phase-2 coefficients are selected based on the proportion of the largest phase-1 coefficients that occur in all blocks, not just the current one. To do this, a threshold T_I is set which is equal to the absolute value of the $M_1/2$-th transform coefficient from the sorted set of M_1 phase-1 transform coefficients from all the blocks in the image. This algorithm is called Threshold over all Image Blocks (THI2). Again, this is similar to the measurement estimation part of the AL-DCT-THI algorithm, but it does not have access to

Algorithm 7.1 THB2

Input: Image $\mathscr{I}$, compression factor C_F, block size B, image width W, image height H.
Output: Number of measurements m_i per block i, m_i measurements per block

1. Crop and partition, $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ blocks;
2. Calculate the number of non-adaptive measurements per block $m_i^1 = \lfloor B^2/(2 \cdot C_F) \rfloor$;
3. Collect m_i^1 L-DCT-ZZ coefficients from each block;
4. Estimate the number of L-DCT-ZZ transform coefficients to collect per block in phase 2, m_i^2, from equation (4.8);
5. Collect m_i^2 additional coefficients as required;
6. Output $m_i = m_i^1 + m_i^2$ number of measurements per block;
7. Output $M = \sum_{i=1}^{n_B} (m_i^1 + m_i^2)$ measurements.

Algorithm 7.2 THI2

Input: Image $\mathscr{I}$, compression factor C_F, block size B, image width W, image height H.
Output: Number of measurements m_i per block i, m_i measurements per block

1. Crop and partition, $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ blocks;
2. Calculate the number of non-adaptive measurements per block $m_i^1 = \lfloor B^2/(2 \cdot C_F) \rfloor$;
3. Collect m_i^1 L-DCT-ZZ coefficients from each block;
4. Sort all phase 1 coefficients in descending order of absolute magnitude;
5. Set a threshold T_I which is equal to the absolute value of the $M/4$-th coefficient in the sorted list of coefficients, where $M = (H \times W)/C_F$;
6. Calculate m_I, the number of coefficients in each block whose absolute value exceeds T_I;
7. Set the number of coefficients to collect from each block in phase 2, m_i^2, to $2 \times m_I$;
8. Collect m_i^2 additional coefficients as required;
9. Output $m_i = m_i^1 + m_i^2$, the number of measurements per block;
10. Output the m_i measurements per block.

all Transform Coefficients (TCs) when estimating the phase-2 TCs, just the TCs from phase-1.

When compressively sensing successive non-key frames in a video sequence, one would have the benefit of having collected many more transform coefficients per block from the key frame, which is sensed at a substantially higher rate. We thus propose another adaptivity algorithm that selects the number of phase-2 measurements per block equal to the number of the M largest phase-1 coefficients in the key frame that fall in each block. This adaptivity algorithm is referred to as the Reference-frame DCT Domain (RDD) algorithm.

Algorithm 7.3 MDD

Input: Image $\mathscr{I}$, transform coefficients of reference image $\mathscr{T}_{Ref}$, compression factor C_F, block size B, image width W, image height H.
Output: Number of measurements m_i per block i, m_i measurements per block

1. Crop and partition $\mathscr{I}$ into $n_B = \lfloor H/B \rfloor \cdot \lfloor W/B \rfloor$, $B \times B$ blocks;
2. Calculate the number of non-adaptive measurements per block $m_i^1 = \lfloor B^2/(2 \cdot C_F) \rfloor$;
3. Collect m_i^1 L-DCT-ZZ coefficients from each block;
4. Replace the m_i^1 low pass coefficients in $\mathscr{T}_{Ref}$ by the TCs collected in step 3;
5. Set m_i^2 equal to the number of largest M transform coefficients in $\mathscr{T}_{Ref}$ by absolute value, that fall in each block i;
6. Collect m_i^2 additional coefficients as required;
7. Output $m_i = m_i^1 + m_i^2$, the number of measurements per block;
8. Output the m_i measurements per block.

It is also feasible to use the phase-1 TCs from the previous frame, whether key or non-key, to estimate the number of phase-2 measurements in the current frame. This algorithm is referred to as the Previous-frame DCT Domain (PDD) algorithm.

Finally, a fifth adaptivity algorithm was investigated that adapts the phase-two measurements based on the reference key frame phase-1 TCs but replacing the low-pass TCs with the phase-1 TCs of the current frame. This is referred to as the Mixed-mode DCT Domain (MDD) algorithm.

Thus, when adaptively sensing key frames, it is possible to either use THB2 or THI2 to adapt the number of phase-2 measurements. When sensing non-key frames, one of the five algorithms - THB2, THI2, RDD, PDD or MDD - can be used. In the results section, the best PSNR and MS-SSIM performance was achieved using the THI2 algorithm on the key frames (refer to algorithm 7.2) and MDD on the non-key frames (refer to algorithm 7.3).

7.3.4 IDA reconstruction

The algorithm proposed in this section, VAL-[IDA]-MCP can use the non-real-time IDA algorithm to increase the PSNR and MS-SSIM of the reconstructed video. The improvement in PSNR and MS-SSIM is not large as shown in table 7.1, but significant since improving VAL-MCP is very difficult because of its inherent high quality. For example, VAL-MCP reconstruction quality exceeds that of Motion Estimation/Motion Compensation (ME/MC) and RRSs by significant amounts as shown in the same table.

7.3.5 Motion compensated prediction of non-key frames

Key frames are encoded with a higher quality (i.e., with lower compression) than non-key frames. Most of the time, there are few changes between frames, and the higher quality key

frame block is a better prediction than the current low-quality non-key frame block. This idea is extended with MCP. ME is used to estimate the block within the high-quality key frame that best matches the current low-quality non-key frame block, which becomes the prediction for the current block.

In the encoder-side hybrid DPCM/DCT algorithm, the prediction is usually referenced from intracoded frames and predicted frames. In the decoder-side version of the algorithm, a high-resolution prediction of the current non-key frame is created from the MCP key frame of the current GOP $\mathbf{M}_{n,i}$ and that of the following GOP $\mathbf{M}_{n+1,i}$. Four algorithms are investigated to predict the i-th non-key frame $\mathbf{R}_{n,i}$ in GOP n, namely:

1. the average of $\mathbf{M}_{n,i}$ and $\mathbf{M}_{n+1,i}$:

$$\mathbf{R}_{n,i} = [\mathbf{M}_{n,i} + \mathbf{M}_{n+1,i}]/2, \tag{7.1}$$

2. the weighted average of $\mathbf{M}_{n,i}$ and $\mathbf{M}_{n+1,i}$:

$$\mathbf{R}_{n,i} = [(g-i)\mathbf{M}_{n,i} + i\mathbf{M}_{n+1,i}]/g, \tag{7.2}$$

   where g is the number of frames in the GOP, and i = 2..g. This weighted average is chosen because the closer key frame is expected to be more correlated with the current non-key frame. This is probable as long as there are no scene changes in the current GOP.

3. a collage of blocks from $\mathbf{M}_{n,i}$ and $\mathbf{M}_{n+1,i}$ selected based on the lowest Mean Absolute Difference (MAD) between the reference and current frame blocks, collected during motion estimation. This is referred to as the Min-MAD algorithm.

4. setting $\mathbf{R}_{n,i} = \mathbf{M}_{n,i}$ if $i < g/2$, $\mathbf{R}_{n,i} = \mathbf{M}_{n+1,i}$ if $i > g/2$ and according to equation 7.1 if $i = g/2$. This is referred to as the Closest Reference Frame algorithm.

7.3.6 Differential pulse code modulation

Temporal DPCM exploits correlation between frames. In our VAL-[IDA]-MCP algorithms, non-key frames are predicted from a weighted average of two motion compensated key frames as computed by equation 7.2. In the following, the weighted-average key frame is referred to as key frame $\mathbf{R}_{n,i}$.

Consider the block hybrid DPCM/DCT codec in figure 7.4. A $B \times B$ pixel block from key frame $\mathbf{K}$ is transformed using a 2D DCT operation generating $B \times B$ transform coefficients $\mathbf{T}_K$. In the L-DCT-ZZ algorithm, $(L+M)$ lowpass transform coefficients are retained from the weighted key frame. L is the number of transform coefficients retained in a non-key frame and M are the number of extra mid-band DCT coefficients retained in a key frame, over and above those in the non-key frame. The resulting $\mathbf{C}_K$ transform coefficients are transmitted to the decoder, where an inverse 2D DCT reconstructs the key frame estimate $\hat{\mathbf{K}}$.

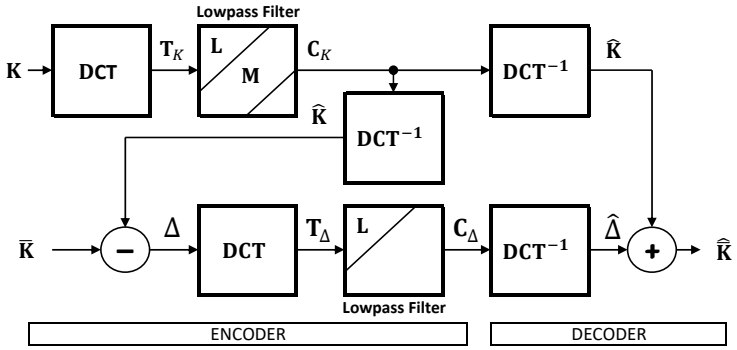


Figure 7.4: Non-distributed, non-compressive DPCM. Key frame blocks $\mathbf{K}$ are 2D DCT transformed to generate transform coefficients $\mathbf{T}_K$, that are lowpass filtered by Hadamard multiplication with lowpass matrix $(\mathbf{L}+\mathbf{M})$, to output lowpass transform coefficients $\mathbf{C}_K$. Non-key frame blocks $\bar{\mathbf{K}}$ and the locally reconstructed key reference blocks $\hat{\mathbf{K}}$ are subtracted to form the difference blocks Δ that when 2D-DCT transformed produce the transform coefficients $\mathbf{T}_\Delta$ of the difference blocks. These are lowpass filtered by Hadamard multiplication with the lowpass filter matrix $\mathbf{L}$ to produce lowpass difference coefficients $\mathbf{C}_\Delta$. At the decoder, $\mathbf{C}_K$ and $\mathbf{C}_\Delta$ are inverse 2D DCT transformed to produce $\hat{\mathbf{K}}$ and $\hat{\Delta}$ that added together produce the reconstructed non-key blocks $\hat{\bar{\mathbf{K}}}$.

To exploit the correlation between a key frame block and a non-key frame block $\bar{\mathbf{K}}$, the key frame block estimate $\hat{\mathbf{K}}$ is subtracted from the non-key frame block. The resulting difference block Δ is transformed using the 2D DCT, generating transform coefficients $\mathbf{T}_\Delta$ which are lowpass filtered, retaining only $\mathbf{L}$ upper triangular transform coefficients $\mathbf{C}_\Delta$. At the decoder, $\mathbf{C}_\Delta$ is inverse transformed to reconstruct an estimate of the difference block $\hat{\Delta}$ that is added to the frame block estimate $\hat{\mathbf{K}}$ to estimate the non-key frame block $\hat{\bar{\mathbf{K}}}$.

Let the key frame pixel block $\mathbf{K}$ be arranged into a column vector $\mathbf{k}$. The linear unitary 2D DCT transform $\mathbf{D}$ then transforms $\mathbf{k}$ into the column vector of transform coefficients $\mathbf{t}_K$

$$\mathbf{t}_K = \mathbf{D}\mathbf{k} \tag{7.3}$$

where $\mathbf{D}$ is the $N \times N$ transform matrix and $N = B \times B$. The lowpass filtering operation is accomplished by element-wise multiplication (denoted by $\odot$) of $\mathbf{t}_K$ with a matrix consisting of $L+M$ 1's in the upper left hand triangle and 0's elsewhere, arranged as a vector $\mathbf{f}_{L+M}$, such that the L-DCT-ZZ vectorized coefficients of the key frame $\mathbf{c}_K$ are given by

$$\begin{aligned} \mathbf{c}_K &= \mathbf{t}_K \odot \mathbf{f}_{L+M} \\ &= (\mathbf{D}\mathbf{k}) \odot \mathbf{f}_{L+M} \\ &= (\mathbf{D}\mathbf{k}) \odot \mathbf{f}_L + (\mathbf{D}\mathbf{k}) \odot \mathbf{f}_M \end{aligned} \quad . \tag{7.4}$$

The L-DCT-ZZ vectorized coefficients of the difference block are given by

$$\begin{aligned} \mathbf{c}_\Delta &= [\mathbf{D}(\bar{\mathbf{k}} - \mathbf{D}^{-1}\mathbf{c}_K)] \odot \mathbf{f}_\mathbf{L} \\ &= [\mathbf{D}\bar{\mathbf{k}} - \mathbf{c}_K] \odot \mathbf{f}_L \\ &= (\mathbf{D}\bar{\mathbf{k}}) \odot \mathbf{f}_L - (\mathbf{D}\mathbf{k}) \odot \mathbf{f}_{L+M} \odot \mathbf{f}_L \\ &= (\mathbf{D}\bar{\mathbf{k}}) \odot \mathbf{f}_L - (\mathbf{D}\mathbf{k}) \odot \mathbf{f}_L \end{aligned} \quad , \tag{7.5}$$

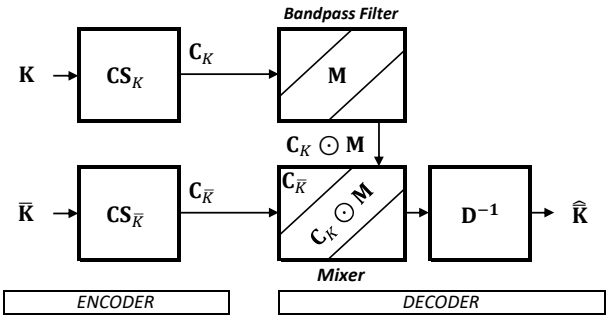


Figure 7.5: Distributed compressive DPCM. $\mathbf{CS}_K$ and $\mathbf{CS}_{\bar{K}}$ represent the CS operations on the key and non-key blocks. $\mathbf{K}$ and $\bar{\mathbf{K}}$ are the reference and non-reference blocks, $\mathbf{C}_K$ are the filtered transform coefficients of the key frame, $\mathbf{C}_{\bar{K}}$ are the filtered transform coefficients of the non-key block, $\mathbf{M}$ is a mid-band filter, $\mathbf{C_k} \odot \mathbf{M}$ are the mid-band transform coefficients of the key block, $\mathbf{D^{-1}}$ is the inverse DCT and $\hat{\bar{\mathbf{K}}}$ is the reconstructed non-key block.

since $\mathbf{f}_{L+M} = \mathbf{f}_L + \mathbf{f}_M$ and $(\mathbf{f}_L + \mathbf{f}_M) \odot \mathbf{f}_L = \mathbf{f}_L$, given that $\mathbf{f}_L$ and $\mathbf{f}_M$ are disjoint ($\mathbf{f}_L \cap \mathbf{f}_M = \{\overrightarrow{0}\}$).

Equation (7.4) shows that the key frame entails transmitting the L lowpass and M mid-band L-DCT-ZZ coefficients. Equation (7.5) shows that the DPCM difference transform coefficients $\mathbf{c}_\Delta$ is composed of the lowpass coefficients of the non-key frame from which are subtracted the lowpass coefficients of the key frame. Therefore, the encoder can be simplified to just two L-DCT-ZZ encoders one for the key frames retaining $(L+M)$ transform coefficients and one for the non-key frames retaining just L coefficients. The difference block coefficients $\mathbf{c}_\Delta$ can then be calculated at the decoder by subtracting the key frame lowpass coefficients $(\mathbf{Dk}) \odot \mathbf{f}_L$ from the non-key frame lowpass coefficients $(\mathbf{D\bar{k}}) \odot \mathbf{f}_L$. This holds for the non-adaptive L-DCT-ZZ algorithm as well as the AL-DCT-BBV and AL-DCT-DD adaptive algorithms.

7.3.7 Real-time reconstruction

The AL-DCT-DD encoder transmits the transform coefficients and the number m_i of transform coefficients per block to the decoder. For the key frames, the decoder then reconstructs each image block using the inverse linear 2D-DCT. The reconstructed key block arranged as a column vector is given by

$$\hat{\mathbf{k}} = \mathbf{D^{-1}}\mathbf{c}_K. \tag{7.6}$$

This can be achieved in real-time using matrix multiplication or fast implementations of the inverse 2D-DCT.

The non-key frame can be computed by first calculating $\mathbf{c}_\Delta$ at the encoder as described in 7.3.6, multiplying it by the inverse 2D DCT matrix $\mathbf{D^{-1}}$ to generate the difference block and then adding it to key frame. In vector form

$$\hat{\bar{\mathbf{k}}} = \mathbf{D^{-1}}\mathbf{c}_\Delta + \hat{\mathbf{k}}. \tag{7.7}$$

However, it is also possible to write

$$\begin{aligned} \hat{\bar{\mathbf{k}}} &= \mathbf{D^{-1}}(\mathbf{c}_\Delta + \mathbf{c}_K) \\ &= \mathbf{D^{-1}}[(\mathbf{D\bar{k}}) \odot \mathbf{f}_L + (\mathbf{Dk}) \odot \mathbf{f}_M]. \end{aligned} \tag{7.8}$$

Thus the non-key frame can also be computed by first mixing (adding) the mid-band transform coefficients of the key frame to the lowpass transform coefficients of the non-key frame and calculating the inverse 2D DCT transform as shown in the distributed version of the DPCM operation in figure 7.5. Note that in figure 7.2, $\mathbf{F}_{n+1,0}$ corresponds to $\mathbf{K}$ in figure 7.5, and $\mathbf{F}_{n+1,i}$ to $\bar{\mathbf{K}}$ for $i > 0$.

7.3.8 Best pixel discriminator

In unchanging parts of a frame, selecting key frame blocks results in a higher-quality rendition. Block-based MCP results in estimating high-quality key frame blocks and the subsequent DPCM process combines them with the low-pass current blocks. If the MCP is not accurate, image artefacts appear in the output blocks and this occurs where the image is changing most dynamically.

Following reconstruction of the current non-key frame and MCP from the key frame, three versions of the reconstructed frame are available; $\bar{\mathbf{K}}_{n,i}$ is low resolution non-key frame with no artefacts, $\mathbf{R}_{n,i}$ is the higher resolution motion compensation predicted frame but with the possibility of image artefacts, and $\mathbf{D}_{n,i}$ which is the DPCM output that may also have visible artefacts. To reduce the visibility of MCP artefacts, one option would be to average the two reconstructions together and this will indeed result in a better PSNR and MS-SSIM.

A second option is to compare the pixels at location (i, j) in both $\mathbf{R}_{n,i}$ and $\mathbf{D}_{n,i}$ and if the modulus of the difference $|\mathbf{R}_{n,i} - \mathbf{D}_{n,i}|$ is less than a threshold T_D then the output pixel is set to $\mathbf{D}_{n,i}$; otherwise it is set to $\bar{\mathbf{K}}_{n,i}$.

This selection operation was found to improve the output quality considerably. The value of T_D was investigated empirically. A value of $T_D = 25$ was found to give good results with this BPD block in the real-time versions of our algorithms, whereas in the non-real-time algorithms $T_D = 10$ gave the best results.

The BPD results in lowpass pixels at the edges of moving objects in the current output frame. However the visual quality is better as the human vision system is less sensitive to errors at moving boundaries.

7.3.9 VAL-MCP parameters

The VAL-MCP algorithm is characterised by a number of parameters that have to be taken into consideration, ideally optimized, namely:

1. DCT block size: 8, 16 or 32;
2. Adaptive algorithm: THB2, THI2, RDD, PDD or MDD, for both key and non-key frames;
3. GOP size;
4. Compression ratio for key and non-key frames;
5. Prediction algorithm: Average, Weighted Average, Min-MAD or Closet Reference Frame;
6. IDA damping factor and iterations, if used;

7. Motion estimation algorithm: Full Search or the adaptive rood pattern search [134];
8. Motion Estimation block size and search area width;
9. BPD threshold T_D;
10. Post filtering using one iteration of the DnCNN algorithm or using the BM3D denoiser as a filter.

The first four parameters impact the encoder whereas the latter pertain solely to the decoder. The implementation of the VAL-MCP algorithms also allows ablation studies by supporting turning off MCP, DPCM and BPD independently.

7.4 VAL-MCP experimental results

The simulations in this chapter were executed on a server equipped with an Intel Xeon CPU E5-160 v3 clocked at 3.50 GHz, with 32 GB of RAM, running MATLAB version 2019a on Windows 10. IDA requires the D-AMP-Toolbox from [135]. The DnCNN code and models were downloaded from [136] and require the MatConvNet [137] package from [138].

Our real-time VAL-MCP and non-real-time VAL-IDA-MCP algorithms were compared with three algorithms in the literature ME/MC [50], RRSs [23] and SGS-OF [24]. Six popularCIF video sequences were used for testing as in [24]. This set is referred to as VidSet6. The first seventeen frames of VidSet6 were used with the seventeenth frame used to compute the weighted key frame in the secondGOP, but PSNR and MS-SSIM results are only reported for the luminance (Y) plane of the first sixteen frames.

The implementation of the real-time VAL-MCP algorithm is depicted in figure 7.2 with IDCT reconstruction. It uses blocks of size B = 16, a key subrate of 0.7 and non-key subrate of 0.1. The motion estimation algorithm uses the Adaptive Rood Pattern Search (ARPS) [134] and uses blocks of size 32 with a search window delta of size 16. The GOP size is 8 with the key frame occurring at position 1 in the GOP. The BPD threshold T_D is set to 25 empirically. The VAL-MCP algorithm is post-filtered using one iteration of the DnCNN denoiser.

The non-real-time VAL-IDA-MCP algorithm, also depicted in figure 7.2, reconstructs the key and non-key frames using the IDA algorithm with the DnCNN denoiser [27]. Two key and non-key subrate combinations were investigated. The key frame compression ratio is set to 0.7 with GOP = 8 or 0.4 with GOP = 4 and the corresponding non-key compression ratio to 0.1 in both cases. This results in the same average compression ratio of 0.175 over the whole GOP. The other parameters are the same as for the VAL-MCP algorithm. The VAL-IDA-MCP algorithm is also post-filtered using one iteration of the DnCNN denoiser.

The original ME/MC code from [139] was modified to use the same MATLAB-based implementation of the ARPS motion estimation and motion compensation algorithms used in our algorithms and the measurement matrix was set to the same partial DCT matrix. The RRSs code from [17] was modified by setting the measurement matrix to the same one used by our

Video Sequence	ME/MC		RRS		SGS-OF		VAL-MCP		VAL-IDA-MCP		VAL-IDA-MCP*	
	GOP = 8	$\delta_k = 0.7$	GOP = 8	$\delta_k = 0.7$	GOP = 8	$\delta_k = 0.7$	GOP = 8	$\delta_k = 0.7$	GOP = 8	$\delta_k = 0.7$	GOP = 4	$\delta_k = 0.4$
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
Paris	29.63	0.9872	27.07	0.9853	<u>32.00</u>	**0.9946**	31.71	0.9890	**32.77**	0.9928	31.47	<u>0.9930</u>
Foreman	32.85	0.9829	36.51	0.9943	<u>37.20</u>	**0.9952**	36.23	0.9874	36.36	0.9921	**37.61**	<u>0.9929</u>
Coastguard	30.14	0.9470	31.20	0.9535	32.40	0.9692	32.70	0.9673	32.63	0.9696	**33.71**	**0.9823**
Hall	41.12	0.9960	35.68	0.9947	40.30	<u>0.9953</u>	**41.84**	0.9932	40.88	0.9948	40.53	0.9937
Mobile	21.39	0.9059	23.96	0.9610	**27.13**	**0.9897**	25.98	0.9780	26.42	0.9792	26.24	0.9825
News	38.39	0.9967	34.11	0.9965	39.88	**0.9978**	40.75	0.9948	**41.34**	0.9973	40.91	0.9971
Average	32.26	0.9693	31.42	0.9809	34.82	**0.9903**	34.87	0.9850	<u>35.07</u>	<u>0.9876</u>	**35.08**	**0.9903**

Table 7.1: Comparison of ME/MC, RRS, SGS-OF, VAL-MCP and VAL-IDA-MCP. VAL-MCP executes in real-time. All algorithms use the L-DCT-ZZ measurement matrix, GOP = 8, $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ except VAL-IDA-MCP* for which GOP = 4, and $\delta_K = 0.4$. Best results in bold, second best underlined.

algorithms. The use of the low-pass DCT measurement matrix improved the results of both ME/MC and RRSs over the respective published results. SGS-OF code was not available and therefore, the SGS-OF results in table 7.1 are from the original paper in [24].

Average PSNR and MS-SSIM results

The PSNR and MS-SSIM results are compared in table 7.1. The PSNR is computed using the equation $10\log_{10}(255^2/MSE)$ where MSE is the mean square error between the reconstructed and original frame. The MS-SSIM is measured using the MS-SSIM function [70] implemented in MATLAB from [140].

The PSNR performance of the real-time VAL-MCP algorithm exceeds that of SGS-OF by 0.05 dB and its the average MS-SSIM performance reaches 0.9850, at the same GOP size and compression ratios. When the frames are reconstructed using the IDA algorithm, the PSNR increases to 35.07 dB and the MS-SSIM to 0.9876. However, the MS-SSIM value is still some way off the SGS-OF MS-SSIM result of 0.9903.

We note that our algorithms achieve the best performance reported when; B=16, the key frame uses the THI2 adaptive algorithm, the non-key frames use the MDD adaptive algorithm, the motion estimation blocks are of size 32×32 pixels with the search area extending ± 16 pixels horizontally and vertically, the weighted average prediction algorithm is used, T_D is set to 10 for VAL-IDA-MCP and to 25 for VAL-MCP, and one iteration of post-reconstruction filtering is employed using the DnCNN denoiser. It is interesting to note that the full-step search motion estimation algorithm is eight times slower than the ARPS algorithm and its performance is marginally worse.

The effect on VAL-MCP performance of decreasing δ_K in small steps from 0.7 to 0.175 was studied and the results are tabulated in table 7.2. PSNR decreases from a peak at $\delta_K = 0.7$ to a minimum at $\delta_K = 0.175$. MS-SSIM increases as δ_K decreases from 0.7 to 0.4, when it reaches a peak of 0.9875, and then decreases again as δ_K continues decreasing to 0.175. Comparing the results when $\delta_K = 0.175$ to when it is equal to 0.7, PSNR increases by 6.24 dB and MS-SSIM from 0.9797 to 0.9847. This shows clearly that VAL-MCP exploits temporal correlation and relies heavily on the unequal measurement of key and non-key frames.

The performance of the non-real-time VAL-IDA-MCP algorithm as δ_K decreases in small

Sequence	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$
	0.7	0.1	0.6	0.1143	0.5	0.1286	0.4	0.1429	0.3	0.1571	0.2	0.1714	0.175	0.175
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
Paris	31.69	0.9890	31.40	0.9900	30.29	0.9902	28.86	0.9898	27.00	0.9873	24.82	0.9803	24.28	0.9774
Foreman	36.21	0.9873	35.97	0.9892	35.70	0.9904	35.32	0.9913	34.49	0.9919	33.31	0.9916	32.84	0.9910
Coastguard	32.69	0.9673	32.57	0.9716	32.44	0.9743	32.16	0.9758	31.56	0.9759	30.18	0.9708	29.69	0.9687
Hall	41.82	0.9931	41.54	0.9933	40.84	0.9938	39.37	0.9940	36.90	0.9941	32.44	0.9924	31.38	0.9913
Mobile	25.67	0.9766	25.45	0.9784	25.09	0.9787	24.67	0.9776	23.88	0.9729	22.61	0.9613	22.27	0.9577
News	40.76	0.9948	40.21	0.9953	40.29	0.9964	38.53	0.9964	35.75	0.9961	32.14	0.9933	30.96	0.9918
Average	**34.81**	0.9847	34.52	0.9863	34.11	0.9873	33.15	**0.9875**	31.60	0.9864	29.25	0.9816	28.57	0.9797

Table 7.2: Comparison of VAL-MCP for $\delta_K = 0.7 \ldots 0.175$ and $\delta_{\bar{K}} = 0.1 \ldots 0.175$ with GOP = 8. No post-reconstruction filtering is employed. Best results in bold.

Sequence	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$	δ_K	$\delta_{\bar{K}}$
	0.7	0.1	0.6	0.1143	0.5	0.1286	0.4	0.1429	0.3	0.1571	0.2	0.1714	0.175	0.175
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
Paris	32.81	0.9908	32.91	0.9917	32.41	0.9924	31.56	0.9929	29.91	0.9924	27.16	0.9884	26.38	0.9862
Foreman	37.34	0.9892	37.31	0.9908	37.32	0.9919	37.32	0.9929	36.74	0.9936	35.64	0.9937	35.29	0.9936
Coastguard	32.79	0.9663	32.70	0.9708	32.55	0.9735	32.36	0.9755	31.87	0.9761	30.74	0.9724	30.36	0.9709
Hall	40.97	0.9923	41.67	0.9934	41.55	0.9939	41.01	0.9944	39.47	0.9950	36.68	0.9954	35.72	0.9953
Mobile	26.46	0.9788	26.38	0.9811	26.30	0.9828	26.07	0.9833	25.60	0.9820	24.39	0.9754	23.95	0.9724
News	41.56	0.9957	41.74	0.9963	41.46	0.9968	40.91	0.9974	39.55	0.9975	36.97	0.9969	35.79	0.9963
Average	35.32	0.9855	**35.45**	0.9874	35.26	0.9886	34.87	**0.9894**	33.86	**0.9894**	31.93	0.9870	31.25	0.9858

Table 7.3: Comparison of VAL-IDA-MCP for $\delta_K = 0.7..0.175$ and $\delta_{\bar{K}} = 0.1..0.175$ with GOP = 8. No post-reconstruction filtering is employed. Best results in bold.

steps from 0.7 to 0.175 is tabulated in 7.3. PSNR peaks at $\delta_K = 0.6$ whereas MS-SSIM peaks at $\delta_K = 0.3$ to 0.4. Although PSNR is above that of SGS-OF for $\delta_K \geq 0.4$, MS-SSIM does not quite reach its MS-SSIM performance of 0.9903, and peaks at 0.9894 instead.

Inspired by the work of Artigas et al. on the DISCOVER DVC algorithm [26], the GOP size is reduced from 8 to 4 and set $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ such that the average compression ratio per frame is the same, 0.175. The non-real-time VAL-IDA-MCP algorithm now achieves state-of-the-art PSNR performance exceeding the PSNR performance of the SGS-OF algorithm by 0.26 dB and matching its MS-SSIM at 0.9903 as shown in table 7.1.

Per-frame PSNR and MS-SSIM results

The PSNR and MS-SSIM performance for the first 16 frames of the six sequences in table 7.1 are shown in figures 7.6 and 7.7 respectively.

The PSNR performance of our VAL-MCP and VAL-IDA-MCP is superior to that of SGS-OF for most frames in the Coastguard, Hall and News sequences. SGS-OF PSNR is superior in the Foreman and Mobile sequences whereas in the Paris sequence, SGS-OF edges PSNR performance in the first GOP with our algorithms performing better in the second.

VAL-IDA-MCP* achieves better average PSNR performance for non-key frames, in the Foreman, Coastguard and News sequences even though the key frame performance is low since the compression ratio is 0.4 instead of 0.7. The non-key frame compression ratio is the same, 0.1, in both cases. The PSNR in frames 5 and 13 is always better than that of all the schemes that have GOP = 8. The dynamic range of the PSNR variation is significantly lower with VAL-

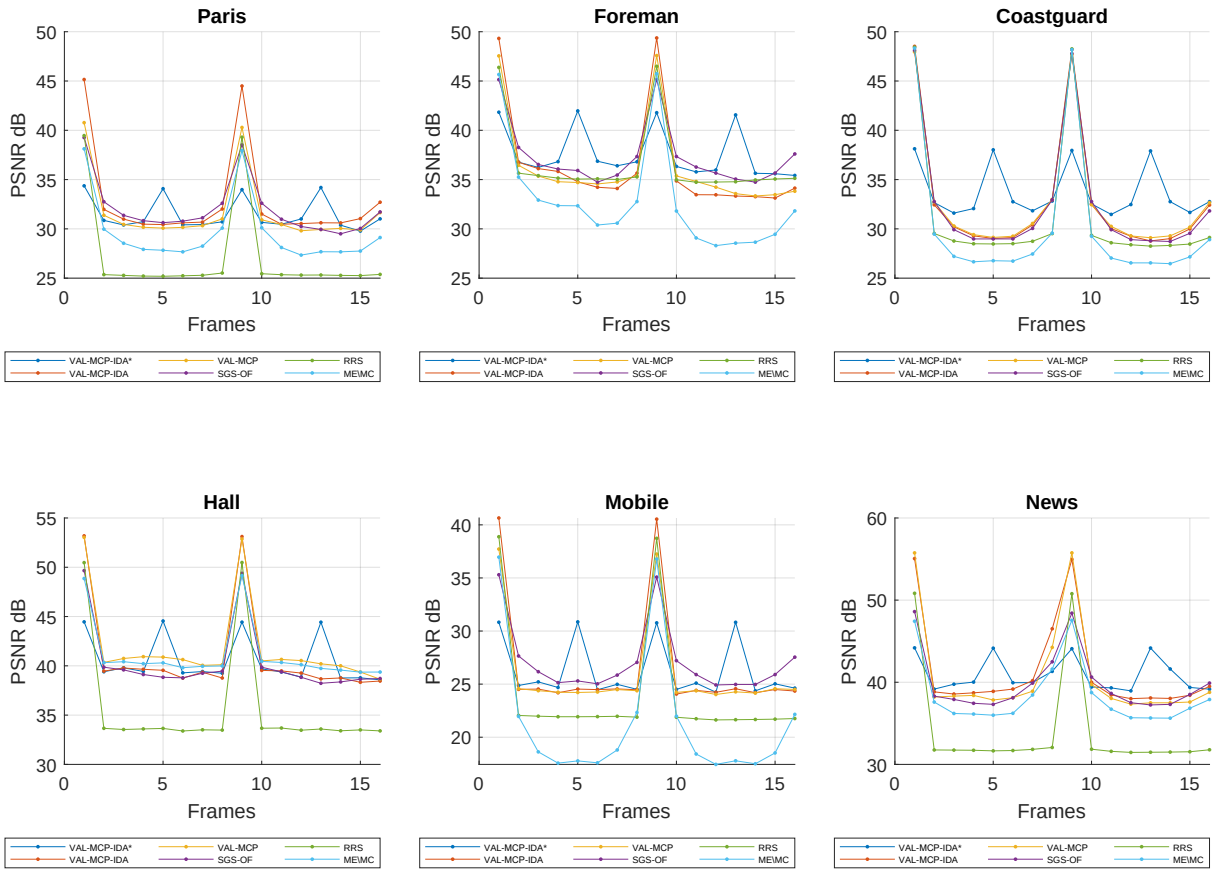


Figure 7.6: Comparison of PSNR per frame for the six algorithms. All sequences were measured using $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ with a GOP size of 8, except for VAL-IDA-MCP* for which the respective values are 0.4 and 0.1 and GOP size is 4.

IDA-MCP* and may lead to a better subjective quality.

The SGS-OF MS-SSIM performance is superior to our schemes for all the sequences except the coastguard sequence. This is due to the better optical flow estimation of the SGS-OF algorithm. However, VAL-MCP* with GOP = 4 has better MS-SSIM performance in frames 5 and 13 in all sequences.

These figures show that with our algorithms, at the same average compression factor of 0.175, it is better to use $\delta_K = 0.4$ and $\delta_{\bar{K}} = 0.1$ with GOP = 4 than $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ with GOP = 8 even though the key frame PSNR and MS-SSIM suffers. The performance improvement of non-key frames more than compensates for the lower key frame performance, even though the non-key frame compression ratio is the same.

Visual quality

The visual quality of the algorithms can be assessed from figure 7.8. ME/MC has visible reconstruction artefacts around the helmet and on the lips of the foreman image. The Siemens text is rendered very well. The RRSs algorithm appears very smooth without visible artefacts but the Siemens text is illegible. The VAL-MCP and VAL-IDA-MCP algorithms produce more details without visible artefacts. The Siemens text is legible although not as sharp as ME/MC. The best quality is shown by VAL-IDA-MCP* that is figure 7.8 (e).

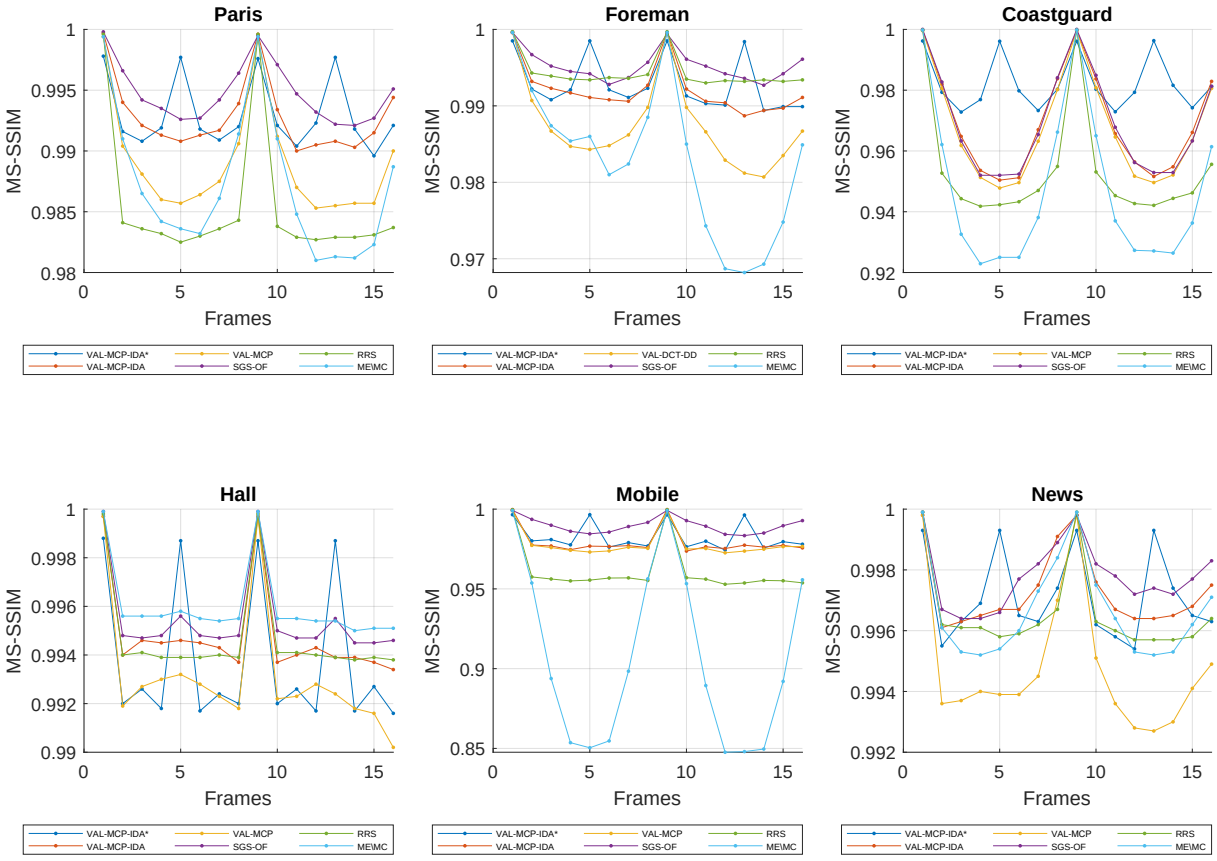


Figure 7.7: Comparison of MS-SSIM per frame for the six algorithms. All sequences were measured using $\delta_K = 0.7$ and $\delta_{\bar{K}} = 0.1$ with a GOP size of 8, except for VAL-IDA-MCP* for which the respective values are 0.4 and 0.1 and GOP size is 4.

Reconstruction times

Average reconstruction times per frame for the six sequences in table 7.1 are given in table 7.4. VAL-MCP yields the fastest reconstruction time at an average of 84 ms per frame. The non real-time VAL-IDA-MCP requires around 3 s to reconstruct a frame even though it uses the GPU-accelerated DnCNN denoiser. The reconstruction time of VAL-MCP can be reduced significantly by using optimised code, for example for the motion estimation function.

The code for SGS-OF was not available. The fastest reconstruction times reported in the original paper [24] are 294.47 s and 227.36 s for the first 17 frames of the Paris and Foreman video sequences respectively. VAL-IDA-MCP requires 56.06 s and 58.89 s for the same frames on our platform, and VAL-MCP 1.717 s and 1.768 s respectively.

Ablation study

Table 7.5 tabulates the results of an ablation study conducted on the VAL-MCP algorithm when compressing the VidSet6 video sequences. In the study; the key and non-key compression ratios are 0.7 and 0.1 respectively, the block-size is 16, the GOP size is 8, the motion estimation block size is 32 and the search parameter is 16, the key frame adaptivity algorithm is THI2, the non-key frame adaptivity algorithm is MDD, and the bidirectional MCP predictor uses the weighted average algorithm.

Figure 7.8: Frame 6 of the Foreman sequence (a) Original (b) ME/MC (c) RRSs (d) VAL-MCP (e) VAL-IDA-MCP* (f) VAL-IDA-MCP.

Algorithm	Reconstruction Time per frame s
VAL-MCP	0.084
VAL-IDA-MCP	2.93
ME/MC	15.2
RRS	528

Table 7.4: Comparison of reconstruction time for the various algorithms executed in our test environment.

Turning off the DPCM function incurs a decrease in PSNR of 2.80 dB and average MS-SSIM decreases from 0.9847 to 0.9652. If the MCP function is switched off, the PSNR decreases by 1.79 dB and MS-SSIM decreases to 0.9659. Turning off both DPCM and MCP functions, which also turns off the best-pixel discriminator, the PSNR falls by 5.35 dB and MS-SSIM falls to 0.9563. This shows the effectiveness of the DPCM, MCP and BPD functions in exploiting temporal correlation.

7.5 Improving VAL-MCP using video frame interpolation

Our VAL-MCP and VAL-IDA-MCP algorithms perform badly on the Mobile sequence when compared with SGS-OF. This is due to the superiority of the optical flow based motion estimation used in SGS-OF over the fast block-based motion estimation and motion-compensation algorithms employed in our VAL-MCP and VAL-IDA-MCP implementations. This section investigates whether using VFI achieves better results.

MCP	DPCM	T_D	PSNR	MS-SSIM
Y	Y	25	34.81	0.9847
Y	N	25	32.01	0.9652
N	Y	25	33.01	0.9659
N	N	0	29.46	0.9563

Table 7.5: Ablation study on the VAL-MCP algorithm.

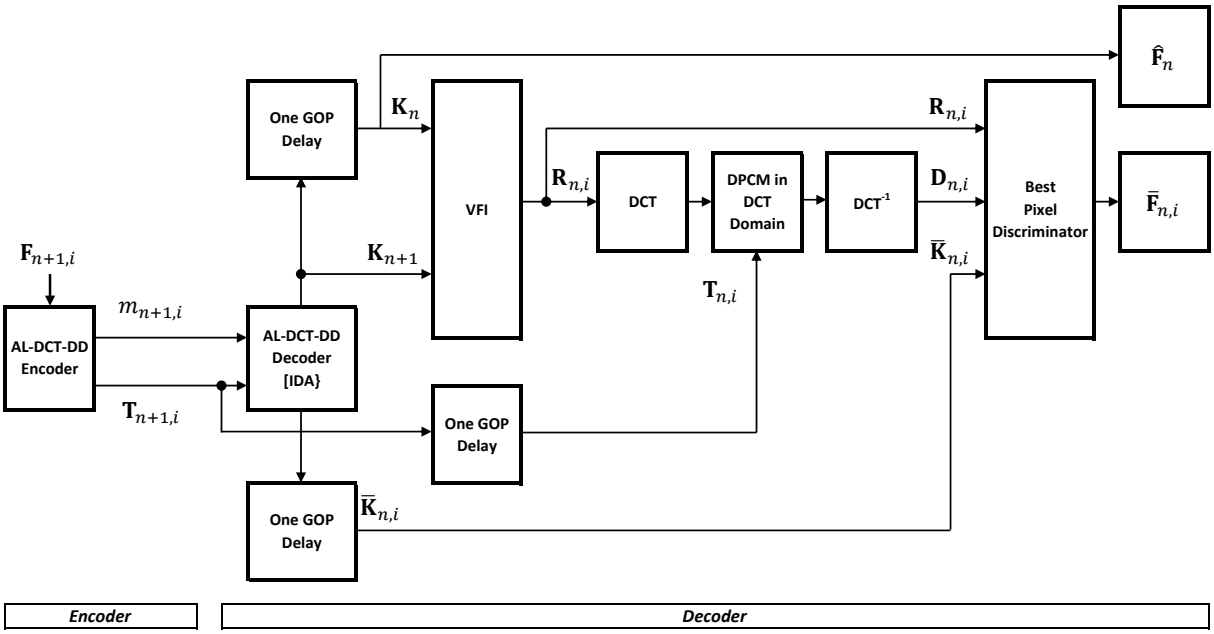


Figure 7.9: Block diagram of the distributed VAL-VFI algorithm. The VFI block replaces the bi-directional MCP in figure 7.2.

7.5.1 VAL-VFI system block diagram

VFI has been extensively studied over the past decades to up-sample the frame rate of video content to match the ever increasing video monitor frame rates and produce smoother playback and slow-motion effects. Recently, CNNs have been leveraged to generate interpolated frames in real-time [111]. One of the better performing CNN-based systems is DAIN [132] which can interpolate frames in real-time. The bi-directional MCP blocks are therefore replaced with the DAIN VFI sub-system and use it to interpolate the frames between two-key frames. The block diagram of the new VAL-VFI scheme is shown in figure 7.9.

7.5.2 VAL-VFI results

GOP = 8, $\delta_k = 0.7$

The average PSNR and MS-SSIM results of VAL-VFI (real-time) and VAL-IDA-VFI are compared with SGS-OF, VAL-MCP and VAL-IDA-MCP in table 7.6. When the GOP size is 8, and the key and non-key frame compression ratios are 0.7 and 0.1 respectively, all our four algorithms achieve better PSNR performance with VAL-VFI (real-time) performing better than VAL-IDA-MCP (non-real-time) and VAL-IDA-VFI (non-real-time) performing best. However,

SGS-OF achieves better MS-SSIM than even VAL-IDA-VFI, 0.9903 against 0.9885. The average compression ratio $\bar{\delta} = 0.175$.

Video Sequence	SGS-OF		VAL-MCP		VAL-IDA-MCP		VAL-VFI		VAL-IDA-VFI	
	GOP = 8	$\delta_K = 0.7$	GOP = 8	$\delta_K = 0.7$	GOP = 8	$\delta_K = 0.7$	GOP = 8	$\delta_K = 0.7$	GOP = 8	$\delta_K = 0.7$
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
Paris	32.00	0.9946	31.71	0.9890	32.77	0.9928	32.28	0.9933	33.82	0.9932
Foreman	37.20	0.9952	36.23	0.9874	36.36	0.9921	35.45	0.9903	37.19	0.9891
Coastguard	32.40	0.9692	32.70	0.9673	32.63	0.9696	32.35	0.9680	32.70	0.9670
Hall	40.30	0.9953	41.84	0.9932	40.88	0.9948	42.02	0.9959	42.21	0.9946
Mobile	27.13	0.9897	25.98	0.9780	26.42	0.9792	28.33	0.9900	28.75	0.9896
News	39.88	0.9978	40.75	0.9948	41.34	0.9973	40.91	0.9974	42.18	0.9979
Average	34.82	**0.9903**	34.87	0.9850	35.07	0.9876	<u>35.22</u>	<u>0.9892</u>	**36.14**	0.9885

Table 7.6: VAL-VFI and VAL-IDA-VFI algorithms compared with SGS-OF, VAL-MCP and VAL-IDA-MCP with GOP = 8, $\delta_k = 0.7$ and $\delta_{\bar{K}} = 0.1$. Best results in bold, second best underlined.

GOP = 8, $\bar{\delta} = 0.175$

If the compression ratio of the key frame is reduced to 0.4, reconstructing the sequences using IDA, allows the VAL-IDA-VFI algorithm to exceed the SGS-OF PSNR by 0.28 dB and marginally exceed its MS-SSIM, 0.9904 to 0.9903, as shown in table 7.7.

GOP = 4, $\bar{\delta} = 0.175$

This section investigates reducing the size of the GOP whilst retaining the average compression ratio per frame at 0.175 with the results shown in table 7.7. The real-time VAL-VFI exceeds SGS-OF performance, achieving a PSNR of 35.12 dB and an MS-SSIM of 0.9921, when GOP = 4 and $\delta_K = 0.5$. VAL-IDA-VFI at GOP = 4 and $\delta_k = 0.5$ easily exceeds the PSNR and MS-SSIM of all the other algorithms in table 7.7.

Therefore replacing MCP with VFI allows our algorithms to achieve state-of-the-art both in real-time without IDA enhancement, and using non-real-time IDA enhancement. VAL-VFI with GOP = 4 and $\delta_K = 0.5$ exceeds SGS-OF PSNR by 0.30 dB and and its MS-SSIM reaches

Video Sequence	SGS-OF		VAL-IDA-MCP		VAL-VFI		VAL-IDA-VFI		VAL-IDA-VFI	
	GOP = 8	$\delta_K = 0.7$	GOP = 4	$\delta_K = 0.4$	GOP = 4	$\delta_K = 0.5$	GOP = 8	$\delta_K = 0.4$	GOP = 4	$\delta_K = 0.5$
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
Paris	32.00	0.9946	31.47	0.9930	31.26	0.9941	31.79	0.9938	33.84	0.9953
Foreman	37.20	0.9952	37.61	0.9929	35.51	0.9916	37.10	0.9924	37.42	0.9913
Coastguard	32.40	0.9692	33.71	0.9823	34.31	0.9832	32.10	0.9745	34.58	0.9832
Hall	40.30	0.9953	40.53	0.9937	41.05	0.9959	41.19	0.9951	41.80	0.9950
Mobile	27.13	0.9897	26.24	0.9825	27.82	0.9903	27.33	0.9890	29.15	0.9921
News	39.88	0.9978	40.91	0.9971	40.77	0.9977	41.11	0.9978	41.97	0.9978
Average	34.82	0.9903	35.08	0.9903	<u>35.12</u>	<u>0.9921</u>	35.10	0.9904	**36.46**	**0.9925**

Table 7.7: VAL-VFI and VAL-IDA-VFI algorithms compared with SGS-OF and VAL-IDA-MCP with $\bar{\delta} = 0.175$. Best results in bold, second best underlined.

0.9921 exceeding SGS-OF's 0.9903. VAL-IDA-VFI exceeds the SGS-OF PSNR by 1.64 dB and its MS-SSIM reaches 0.9925.

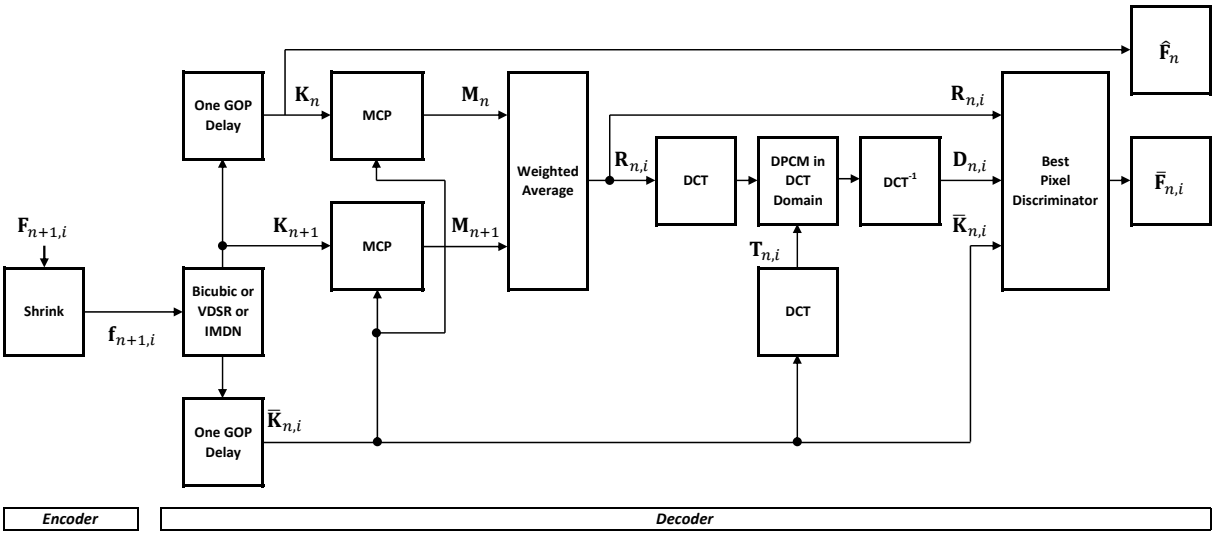


Figure 7.10: Block diagram of the distributed VSR-MCP algorithm.

7.6 Distributed video super resolution

In chapters 5 and 6 it was shown that image super resolution using sub-sampling followed by CNN-based super resolution achieves competitive results when compared to image compressive sensing. This section therefore studies whether the distributed video CS algorithm developed above can be improved by replacing the DCT-based CS measurements with a bicubic filter at the encoder and the non-real-time IDA with a real-time CNN-based image SR block.

7.6.1 VSR-MCP system block diagram

The VSR-MCP system block diagram is shown in figure 7.10. At the encoder, the Shrink algorithm introduced in section 5.3.1 is used instead of the AL-DCT-DD algorithm. The AL-DCT-DD block in the encoder and the IDA block in the decoder are replaced but the MCP and discriminator blocks are retained. DPCM is still carried out in the DCT domain and so it is not necessary to convert both the interpolated frames and non-key frames as part of the DPCM operation.

7.6.2 VSR-MCP results

Initially, the algorithms are tested without exploiting Key Frames in the GOP structure, that is the compression ratio for key frames δ_K is the same as that for the non-key frames $\delta_{\bar{K}}$. In this case, the output key frames are $\hat{\mathbf{F}}_n$ and the non-key frames $\bar{\mathbf{F}}_{n,i}$. Subsequently, the same algorithms are tested with unequal $\delta_K > \delta_{\bar{K}}$ such that key frames are transmitted with greater quality than non-key frames. The same hardware platform is used as in section 7.4.

All of our simulations are run on a Matlab platform. Some of the plug-in functions require MatConvNet to operate, others run in native Matlab mode, but some require Python and a CNN platform such as Pytorch. Python/Pytorch code was either used to generate frames that were subsequently ingested into Matlab scripts or the Matlab scripts used the system function to launch the Python/Pytorch code and exchange data files in real-time.

Video SR without key frames

The results in table 7.8 were generated from the Y component of the four sequences in Vid4 comprising the calendar, city, foliage and walk colour sequences [141]. The *RGB* frames were converted to YC_BC_R and only the Y component was processed by our algorithms. The bicubic algorithm uses the Matlab imresize function to downscale the frames by $\times 2$, $\times 3$ and $\times 4$ in each dimension with a resulting compression ratio of $1/4 = 0.25$, $1/9 \approx 0.1111$ and $1/16 = 0.0625$. At the decoder, the same Matlab function is used to upscale the image to its original size.

Chapter 5 section 5.3.1 showed how the imresize function can be implemented as a convolutional filter on CS sensors using the Shrink algorithm. Then in the Shrink-{SRCNN|VDSR|IMDN} algorithms the Super Resolution CNN (SRCNN) [104], VDSR [106] and IMDN [109] super resolution engines used at the decoder, were downloaded from their authors' websites. The original SRCNN code is a Matlab CPU-based script and we modified it to use GPU-acceleration. VDSR uses the MatConvNet CNN package for Matlab. IMDN is implemented in Pytorch and so the frames were first upsampled the downsampled frames in the Python/Pytorch framework, and then imported into our Matlab scripts. The Shrink-{VDSR|IMDN} algorithms, including the plugins, allow real-time operation with the support of a GPU.

The main VSR-{Bicubic|VDSR|IMDN} algorithm was implemented in Matlab with the VDSR engine directly integrated into Matlab but IMDN was used offline. The three versions of the algorithm can run in real-time but would need an accelerated motion-estimation and motion-compensation function.

The Shrink and the Video SR algorithms now are compared with our L-DCT-ZZ and AL-DCT-DD algorithms operating on independent frames (i.e., not exploiting temporal correlation) and with VAL-IDA-MCP that uses DPCM, ME and MCP to exploit temporal correlation. Table 7.8 shows that even when the key frames and non-key frames use the same δ, the CS reconstruction can improve PSNR and SSIM, for example compare AL-DCT-DD (does not use Motion Estimation/Motion Compensation Prediction (ME/MCP) and DPCM) with VAL-MCP (uses ME/MCP and DPCM).

The algorithms are also compared with the state of the art 3DSRnet [142], VSRResNet [143] and VSRResNetGAN [143] algorithms that do not explicitly use motion estimation and compensation but benefit from it implicitly. VSRNet [144], VESPCN [145] and SPMC-VSR [146], on the other hand, do use ME/MCP explicitly in their algorithms.

Algorithm	Real	$\times 2$, $\delta = 0.25$		$\times 3$, $\delta = CR = 0.1111$		$\times 4$, $\delta = 0.0625$	
	Time	PSNR dB	SSIM	PSNR dB	SSIM	PSNR dB	SSIM
Independent frame compression - no motion estimation							
Shrink-Bicubic	Y	28.45	0.8978	25.28	0.7311	23.77	0.6303
Shrink-SRCNN	Y	30.72	0.9169	26.52	0.7935	24.71	0.6917
Shrink-VDSR	Y	31.36	0.9259	26.78	0.8093	24.86	0.7089
Shrink-IMDN	Y	32.04	0.9321	27.09	0.8216	25.25	0.7241
Exploiting temporal correlation without explicit motion estimation							
3DSRnet smallSet	Y	31.98	0.9386	27.64	0.8476	25.46	0.7498
3DSRnet largeSet	Y	32.25	0.9410	27.70	0.8498	**25.71**	0.7588
VSRResNet	Y	31.87	**0.9426**	26.53	0.8148	24.50	0.7023
VSRResNetFeatGAN	Y	30.90	0.9241	27.80	**0.8571**	25.51	0.7530
Exploiting temporal correlation with explicit motion estimation							
VSRnet	Y	31.30	0.9278	26.79	0.8098	24.84	0.7049
VESPCN	Y	x	x	27.25	0.8245	25.35	0.7301
SPMC-VSR	Y	x	x	27.49	0.8400	25.52	**0.7600**
Exploiting temporal correlation with explicit motion estimation							
L-DCT-ZZ	Y	31.47	0.9211	26.36	0.7814	24.45	0.6718
AL-DCT-DD	Y	32.21	0.9235	26.85	0.7884	24.81	0.6803
VAL-MCP	Y*	32.45	0.9274	27.06	0.7987	25.00	0.6931
VAL-IDA-MCP	N	**33.47**	0.9416	**27.91**	0.8311	25.68	0.7322
Exploiting temporal correlation with explicit motion estimation							
VSR-Bicubic	Y*	28.44	0.8669	25.39	0.7351	23.63	0.6169
VSR-VDSR	Y*	31.07	0.9192	26.68	0.8025	24.80	0.7019
VSR-IMDN	Y*	31.60	0.9247	26.93	0.8144	25.18	0.7180

Table 7.8: Comparing Shrink-{Bicubic|SRCNN|VDSR|IMDN}, L-DCT-ZZ, AL-DCT-DD, VAL-[IDA]-MCP and VSR-{Bicubic|VDSR|IMDN} with algorithms in the literature that exploit motion implicitly - 3DSRNet, VSRResNet and VSRResNetFeatGAN - and explicitly - VSRnet, VESPCN and SPMC-VSR on VidSet4. VAL and VSR algorithms have $GOP = 4$, and $\delta_K = \delta_{\bar{K}}$. Best results in bold. $\times 2$, $\times 3$ and $\times 4$ refer to 2, 3 and 4 times super resolution in both vertical and horizontal dimensions. The corresponding δ values are 0.25, 0.1111, 0.0625.

Discussion

The performance of the Shrink-{Bicubic|VDSR|IMDN} algorithms without ME/MCP is better than that of the VSR-{Bicubic|VDSR|IMDN} algorithms even though the latter use ME/MCP. As seen in section 7.6.2 the video SR algorithms do significantly better when the key frames are coded with a higher δ_K than non-key frames. The recent IMDN image super resolution algorithm returns the best results when employed in our Shrink and Video SR algorithms.

The simple Shrink-IMDN algorithm performs better than our real-time L-DCT-ZZ, AL-DCT-DD and VAL-MCP algorithms, except that AL-DCT-DD and VAL-MCP have better PSNR at $\times 2$. However, our non-real-time VAL-IDA-MCP algorithm achieves better PSNR than all other algorithms in table 7.8.

It is to be noted that 3DSRnet, VSRResNet, VSRResNetFeatGAN, VSRnet, VESPCN and SPMC-VSR generally perform better than our real-time algorithms from a PSNR and SSIM perspective across all three compression ratios. VAL-IDA-MCP performs very competitively from a quality perspective, but the other algorithms execute in real-time. It should again be pointed out that the Video SR algorithms in the literature including those listed in table 7.8 can all be deployed in systems that use a CS sensor, using our Shrink algorithm to perform bicubic convolution filtering prior to sub-sampling, directly on the CS sensor.

VSR-MCP with key frames

The algorithms in the previous section do not exploit unequal key frame and non-key frame compression ratios. This section explores whether this technique would improve our video SR algorithms. It is noted that by adopting a GOP of 4 frames, with the key frame sub-sampled at 1/4 and the three non-key frames sub-sampled at 1/16, the overall average compression ratio is $7/64 = 0.1094 \approx 0.1111 = 1/9$. In the following, the Video SR and Video CS algorithms are therefore set up with GOP = 4, $\delta_K = 0.25$ and $\delta_{\bar{K}} = 0.0625$. The performance is then compared against other published Video SR algorithms and Shrink-{Bicubic|SRCNN|VDSR|IMDN} at $\delta = 1/9 \approx 0.1111$.

Table 7.9 shows that the performance of VAL-MCP-ZZ, VAL-MCP and VSR-IMDN-MCP exceeds all other algorithms by significant amounts. VAL-MCP-ZZ and VAL-MCP perform better than VSR-IMDN-MCP. This is because the DPCM DCT-Domain mechanism is tuned to our AL-DCT-{BBV|DD} transform coefficients matrices, that exist inherently in our AL-DCT-{BBV|DD} and VAL-MCP algorithms but not in the Video SR algorithms. VAL-MCPhas a 1.96 dB advantage over VSRResNetFeatGAN and its SSIM is 0.8808 to the latter's 0.8571. Both can operate in real-time although VAL-MCP would benefit from an optimized Motion Estimation engine.

The best performance is achieved by the non-real-time VAL-IDA-MCP algorithm, which has a 0.71 dB advantage on VAL-MCP and its SSIM is 0.8976 to VAL-MCP's 0.8808.

Algorithm	Real	$\times 3$, $\delta = 0.1111$	
	Time	PSNR dB	SSIM
Independent frames without motion estimation			
Bicubic	Y	25.28	0.7311
Shrink-SRCNN	Y	26.78	0.8093
Shrink-VDSR	Y	27.09	0.8216
Shrink-IMDN	Y	26.52	0.7935
Exploiting temporal correlation without explicit motion estimation			
3DSRnet smallSet	Y	27.64	0.8476
3DSRnet largeSet	Y	27.70	0.8498
VSRResNet	Y	26.53	0.8148
VSRResNetFeatGAN	Y	27.80	0.8571
Exploiting temporal correlation with explicit motion estimation			
VSRnet	Y	26.79	0.8098
VESPCN	Y	27.25	0.8245
SPMC-SR	Y	27.49	0.8400
Our algorithms exploiting temporal correlation with explicit motion estimation			
VAL-MCP-ZZ	Y*	29.22	<u>0.8838</u>
VAL-MCP	Y*	<u>29.76</u>	0.8808
VAL-VFI	Y	30.28	0.8988
VAL-IDA-MCP	N	<u>30.47</u>	<u>0.8976</u>
VAL-IDA-VFI	N	**30.92**	**0.9124**
VSR-VDSR-MCP	Y*	27.57	0.8402
VSR-IMDN-MCP	Y*	28.28	0.8653
VSR-IMDN-VFI	Y	28.86	0.8733

Table 7.9: Comparing our Shrink, VAL and VSR algorithms with those in the literature that exploit motion implicitly - 3DSRNet, VSRResNet and VSRResNetFeatGAN - and explicitly - VSRnet, VESPCN and SPMC-VSR on VidSet4. VAL and VSR algorithms have $GOP = 4$, $\delta_K = 0.25$ and $\delta_{\bar{K}} = 0.0625$. Best results in bold, second best underlined. $\times 3$ refers to 3 times super resolution in both vertical and horizontal dimensions. The corresponding δ value is 0.1111.

7.7 Improving VSR-MCP with VFI

7.7.1 VSR-VFI system block diagram

As already done with VAL-MCP, VSR-MCP is modified by replacing the MCP function with the DAIN VFI algorithm to produce the VSR-VFI algorithm. The block diagram is shown in figure 7.11.

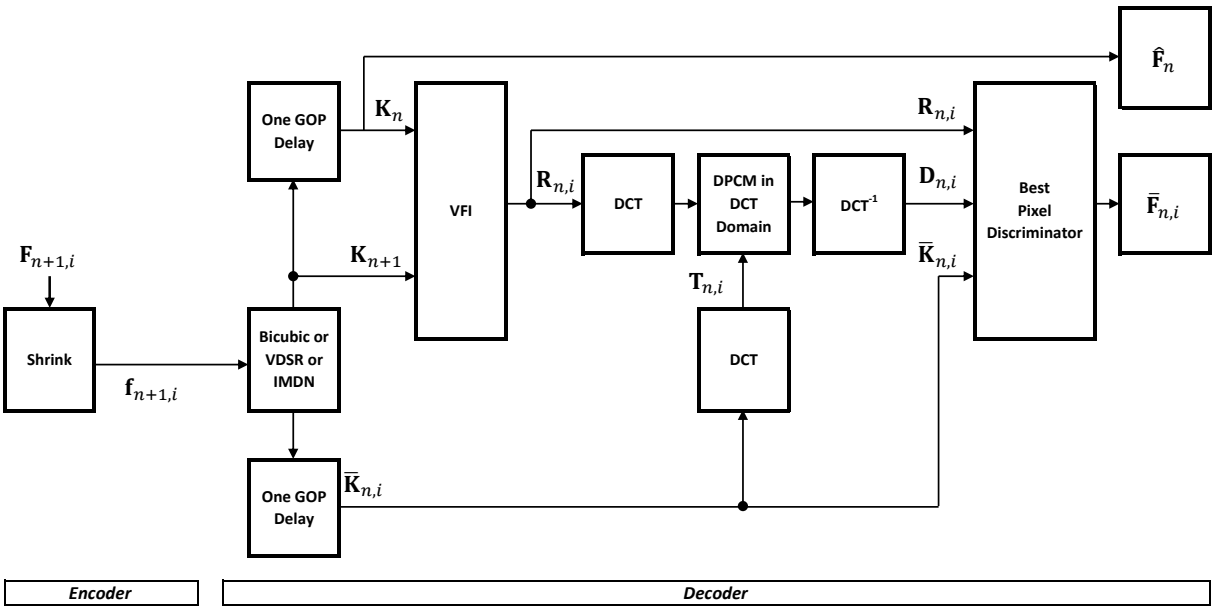


Figure 7.11: Block diagram of the distributed VSR-VFI algorithm.

7.7.2 VSR-VFI results

PSNR and SSIM

Table 7.9 also compares VSR-IMDN-VFI with the algorithms in table 7.8 and with VAL-IDA-VFI. The latter achieves the best performance, improving on VAL-IDA-MCP by 0.45 dB PSNR and SSIM reaching 0.9124. VSR-IMDN-VFI also improves on VSR-IMDN-MCP by 0.58 dB PSNR and its SSIM reaches 0.8773, but still lags behind our VAL algorithms. This is because DPCM in the DCT domain is significantly better with VAL than with VSR.

7.8 Execution time

In VAL-MCP, the computationally intensive operations are the forward and inverse DCTs performed on the sub-image blocks, and the two motion compensated predictions on the key frames. VAL-MCP will run in real-time on our platform with compiled MCP code. Currently, the non-mex compiled code executes at around 10 frames per second on our experimental Matlab platform. Execution speed should increase if an optimised version of the code is compiled.

The DAIN VFI algorithm runs on the GTX 1080 Ti GPU and requires 0.227 s to interpolate one 672×480 pixel frame whereas a Quarter CIF (QCIF) 176×144 frame is interpolated in 0.039 s. It is therefore possible to optimise VAL-VFI to execute in real-time on a workstation equipped with a top-of-the-range GPU, when processingQCIF,CIF and 4CIF video frames.

In VSR the VDSR and IMDN super resolution algorithms run on the GPU in less time than that required by DAIN. The execution time of MCP and VFI are the same as described above for VAL-MCP and VAL-VFI. The remaining operations in figure 7.11 run in real-time on the CPU. Therefore VSR-VDSR-VFI and VSR-IMDN-VFI can both execute in real-time on a workstation with a top-of-the-range GPU.

The IDA algorithm increases the PSNR, SSIM or MS-SSIM of both VAL and VSR algorithms. It uses the GPU-accelerated DnCNN denoiser that requires 30 ms to denoise 256×256 frames. However the IDA algorithm requires 20 to 40 iterations of the denoiser to generate the results reported in this section, and hence renders the VAL-IDA-{MCP|VFI} schemes non-real-time.

7.9 Chapter summary

This chapter establishes VAL-[IDA]-MCP, a novel block-based, adaptive, compressive sensing algorithm that can compressively sense video using single-pixel cameras, multi-pixel cameras or focal plane processing image sensors, and reconstruct them in real-time using CPU-based decoders. Whilst the 2D-DCT exploits spatial correlation of pixels, DPCM, ME/MCP and VFI are used to leverage correlation between frames.

Key and non-key frames are sensed in the encoder using an adaptive deterministic partial DCT measurement matrix. The number of sensed transform coefficients per block and the coefficients themselves are transmitted to the decoder.

The adaptive compressive measurements can be reconstructed in real-time using an inverse 2D-DCT. DPCM and MCP are performed at the decoder to simplify the encoder considerably. MCP and VFI are used to improve the quality of non-key frames by predicting blocks from the higher resolution key frame. DPCM is then performed in the transform domain to combine the motion compensated prediction frame with the non-key frame data. A novel selection algorithm selects the best pixels from the DPCM/MCP predicted reconstruction and the directly reconstructed non-key frame.

The PSNR and MS-SSIM quality of the reconstructed video can be improved using an iterative denoising algorithm - VAL-IDA-MCP, inspired by the D-AMP algorithm and using the DnCNN GPU-accelerated denoiser. However, VAL-IDA-MCP cannot reconstruct the video in real-time.

The PSNR performance of our real-time VAL-MCP algorithm exceeds that of the non-real-time, state-of-the-art SGS-OF algorithm by 0.05 dB and its MS-SSIM performance reaches 0.9850 to SGS-OF's 0.9903 with a GOP of 8 frames and $\delta_K = 0.7$. Enhancing VAL-MCP with IDA, allows VAL-IDA-MCP to exceed the SGS-OF PSNR, i.e. 35.07 dB compared to 34.82 dB, although the MS-SSIM performance is slightly worse at 0.9876.

Varying the GOP and the compression ratio allocation between the key and non-key frames, allows us to trade off PSNR and MS-SSIM in our VAL-MCP and VAL-IDA-MCP algorithms. The VAL-IDA-MCP PSNR exceeds SGS-OF by 0.26 dB and matches its MS-SSIM, even though SGS-OF uses a CNN-implemented optical flow algorithm. VAL-IDA-MCP execution is faster than the reported SGS-OF time, but not on the same platform, since the SGS-OF code has not been published.

Inspired by recent developments in VFI, we replaced the bi-directional MCP block with a CNN-based VFI sub-system. This allows our algorithms to achieve state-of-the-art performance

both in real-time without IDA enhancement, and using the non-real-time IDA enhancement. VAL-VFI with GOP = 4 and $\delta_K = 0.5$ exceeds SGS-OF PSNR by 0.30 dB and MS-SSIM reaches 0.9921, whereas VAL-IDA-VFI exceeds the same SGS-OF PSNR by 1.64 dB and MS-SSIM reaches 0.9925 as shown in table 7.7.

VAL-MCP and VAL-IDA-MCP have applications in low-power wireless sensor networks and video surveillance. The reconstruction time of the real-time algorithm is around 100 ms in MATLAB on the setup described above. This allows 10 frames per second to be reached even in this unoptimized environment. The reconstruction time of the non-real-time VAL-IDA-MCP algorithm is around 3s per frame. The algorithm will benefit from unfolding as GPU implementations, but will then require a GPU-based decoder.

Video SR based on our Shrink algorithm has the benefit of a very simple encoder-side implementation. Shrink-IMDN with independent frame compression, that is without motion estimation or DPCM, comes close to the performance of the best algorithms in table 7.8. However, without using unequal compression ratios for the key and non-key frames, motion estimation and compensation lead to lower performance. When the key frames and non-key frames are compressed with a compression ratio of 1/4 and 1/16 respectively, the average compression ratio is approximately 1/9. Then, VSR-IMDN-MCP and VSR-IMDN-VFI both generate better results than in the literature as shown in table 7.9. However, they do not match the superior results of our VAL algorithms. This is due to the decoder-side DPCM working better with the VAL algorithms, for which it was designed.

Chapter 8

End-to-end distributed video compressive sensing

This chapter studies the end-to-end implementation of the distributed Video Compressive Sensing (VCS) algorithms proposed and studied from a dimensionality reduction perspective, in chapter 7. In that chapter, it was established that VCS is better suited for video compression than the Video Super Resolution (VSR) algorithms and hence the work in this chapter concentrates on the former.

Distributed video coding schemes, such as the DIStributed COding for Video sERvices (DISCOVER) algorithm [26], adopt the classical Group of Pictures (GOP) approach to break video sequences into groups of frames in which there is at least one key frame that is coded in intra mode, meaning it is independently decodable. In DISCOVER and derivatives, the key frame is fully sensed (all the pixels are captured prior to compression) and then a computationally intensive coding scheme is used to compress it with high coding efficiency. The non-key frames are coded using a significantly less complex algorithm, for example the 2D Discrete Cosine Transform (DCT), in which the number of transform coefficients per block and the quantization bits per transform coefficient vary adaptively, under the control of a Wyner-Ziv decoder, that requests additional bits from the encoder over a feedback channel.

Our low-complexity, real-time video compression algorithms capture the key and non-key frames compressively, from Compressive Sensing (CS) sensors, using the AL-DCT-{ZZ|DD} algorithms developed in chapter 4. The compressive samples are then quantized and further compressed using low-complexity coding algorithms. The distributed video CS algorithms proposed in this chapter do not make use of a feedback channel, unlike classical Distributed Video Coding (DVC) schemes such as DISCOVER, reducing the video sensor complexity and power consumption.

The tabular transmission encoding algorithm developed in chapter 6 for image CS is improved and deployed to transmit the compressively sensed video. Two sets of video algorithms, one using Motion Compensated Prediction (MCP) and the other VFI, are evaluated from a rate-distortion perspective in section 8.4. The end-to-end VCS algorithms are compared with the DISCOVER algorithm to establish the performance gap. Finally, the distributed VCS end-to-

end algorithms are analysed using the framework developed in chapter 3.

Algorithms

The following end-to-end distributed video compressive sensing algorithms are developed in this chapter.

- TAB-PNG takes the transform coefficients produced by L-DCT-ZZ, AL-DCT-BBV and AL-DCT-DD, stores them in a table, quantizes them, shifts them to occupy the range [0..255] and then uses the PNG algorithm to compress the image-like table prior to transmission.
- VAL-TP uses the TAB-PNG algorithm to encode key frames in a video sequence for digital transmission, compressed using L-DCT-ZZ, AL-DCT-BBV or AL-DCT-DD. These frames are then reconstructed using an inverse 2D-DCT transform.
- VAL-TP-IDA improves the quality of key frames using the Iterative Denoising Algorithm (IDA) algorithm.
- VAL-TP-MCP adopts distributed video coding principles to exploit temporal correlation by using block-based Motion Compensated Predication and Differential Pulse Code Modulation at the decoder, to improve reconstruction quality.
- VAL-TP-MCP-IDA uses the IDA to improve the quality of VAL-TP-MCP.
- VAL-TP-VFI improves the VAL-TP algorithm by using a GPU-based Video Frame Interpolation algorithm instead of block-based Motion Compensation Prediction.
- VAL-TP-VFI-IDA improves the quality of the VAL-TP algorithm using IDA on both the key and non-key frames.

8.1 Introduction

Two of our main objectives are to capitalize on the availability of compressive image sensors that (i) perform linear transforms in the analogue domain power-efficiently and (ii) reduce the number of linear transforms in proportion to the underlying sparsity of the image or video frame. This latter property will also lead to a reduction in the analogue to digital conversions required to only one per transform coefficient rather than one per pixel.

In classical distributed video coding, such as the DISCOVER codec [26], the key frame is coded by capturing and converting all the pixels to digital and then employing a powerful intracoding technique to compress it as much as possible whilst retaining the highest possible quality. Wyner-Ziv distributed video coding theory [3], is then leveraged to exploit the correlation between the key frame and one or more subsequent frames to code the non-key frames

with the same quality, but with a lower rate such that the overall rate-distortion performance improves. The DVC codec is block-based and adaptive, with the decoder requesting more information bits from each block recursively until a non-key frame can be successfully decoded based on side-information computed from the decoded key frame and the received cumulative information with respect to the non-key frame.

The fact that the key frame is coded using an advanced coding algorithm that assumes that all the pixels are available to it (i.e. it is fully sensed), makes it difficult for our AL-DCT-{ZZ|DD} and AL-DCT-{ZZ|DD}-IDA based key frame encoding and reconstruction schemes to match the rate-distortion performance of DVC key frames that use, say, the intra mode of the the H.264/AVC [147] standard.

A straightforward way of deploying DVC techniques at an encoder provisioned with an Single Pixel Camera (SPC) or Multi Pixel Camera (MPC) is to sense all the pixels one pixel at a time and then digitize the key frame using the H.264/AVC intra mode. The non-key frame DCT transform coefficients would be directly computed on the compressive sensor with more transform coefficients measured, in adaptive mode, as directed by the DVC feedback channel. However, this requires that key frames are fully sensed and the assumption is that this is prohibited a priori. Secondly, the DVC architecture requires a feedback channel from the decoder to the encoder to continuously solicit more parity bits from a Low-density Parity-Check (LDPC) encoder.

Therefore, this chapter investigates how our VAL-[IDA]-{MCP|VFI} algorithms compare with the classical DVC algorithm. The VAL algorithms are combined with the entropy coding techniques studied in chapter 6 to implement end-to-end video codecs and compare them against the DISCOVER DVC algorithm.

8.2 Related work

In 1999 Pradhan and Ramchadran proposed Distributed Source Coding Using Syndromes (DISCUS) [148] a technique based on the Slepian-Wolf [2] and Wyner-Ziv [3] theorems in distributed source coding.

This work inspired the work of Aaron et al. at Stanford who developed a technique for distributed video coding based on Wyner-Ziv coding [67], that became known as the Stanford distributed video coding architecture. Aaron proposed a simplified distributed video codec with intraframe coding at the encoder, a Slepian-Wolf Turbo Encoder, and an interframe decoder incorporating a Turbo decoder that requests further parity bits from the Turbo Encoder until the Wyner-Ziv frame is decoded successfully. A conventional intraframe encoder is used to transmit key frames. When decoded the key frames are used to predict the next frame to produce side information for the Turbo decoder. The Stanford architecture has spawned an extensive amount of derivative work reported in the literature [149].

In 2002, Puri and Ramchandran [47] independently proposed Power-efficient, Robust, hIgh compression, Syndrome based Multimedia coding (PRISM), with the stated goal of shifting the

computational burden from the encoder to the decoder. They simplify the task of the encoder to intra-frame coding, and perform full-motion inter-frame decoding at the decoder. The concept of moving the computationally burdensome Motion Estimation (ME) from the encoder to the decoder is attributed to Rabiner and Chandrakasan [150]. The authors in [47] reported that their scheme came within 1.5 dB of the H.263+ algorithm on Football and Euronews sequences.

In 2004, Aaron et al. extended the pixel-domain DVC architecture to operate in the transform-domain by processing the Wyner-Ziv video frames with the DCT [151]. This architecture is usually referred to as the Transform Domain Wyner-Ziv (TDWZ) architecture to distinguish it from the Pixel Domain Wyner-Ziv (PDWZ) configuration.

In 2007, Artigas et al. introduced the DISCOVER video codec, based on the Stanford TDWZ architecture [26]. The key frames are coded using H.264/AVC intra mode. The non-key Wyner-Ziv frames are DCT transformed and adaptively quantized. The quantized coefficients are then split into bit planes that are encoded using a Slepian-Wolf channel encoder, based on the rate-compatible LDPC Accumulate (LDPCA) codes as proposed in [152]. At the decoder, the H.264/AVC key frames are used to generate the side information that drives the Wyner-Ziv decoder. The authors report that their codec gets within 2 dB of H.264/AVC (with no MC employed). In 2008, Pereira et al. [153] described eleven applications that are suitable for DVC including wireless video cameras, wireless low-power surveillance and wireless capsule endoscopy within the EU funded DISCOVER project [154].

In 2015 Imran et al. [149] reviewed the state-of-the-art in DVC architectures and identified improvements to both PDWZ and TDWZ distributed video coding. However, CS was not included in this review.

Do et al. [51] were amongst the first to combine DVC with CS, proposing their Distributed Compressed Video Sensing (DISCOS) solution in 2009. The authors of the paper show that DISCOS performed 8 to 10 dB better than intraframe CS encoding and decoding. In their scheme, key frames were encoded using conventional MPEG/H.26x intra coding standards, with Block-based CS (BCS) used to encode inter-frames based on an inter-frame sparsity model.

Mun and Fowler exploited motion estimation and motion compensation at the decoder for BCS of video [133]. The authors reported that their Motion Estimation/Motion Compensation (ME/MC) scheme achieved an improvement of 0.5 dB for the Foreman sequence and between 0.75 dB and 2 dB for the Coastguard sequence, over the work of Do et al. [51].

Baig et al. studied a distributed video codec in which both the key frames and non-key frames were processed using CS encoding at the encoder [155]. The CS measurements are captured using random projections that are quantized using Gaussian quantizers prior to transmission. Both key and non-key frames are reconstructed using GPSR [156] reconstruction. The reconstruction of the non-key Wyner-Ziv frames is aided with side-information in the form of a dictionary based on correlation between neighboring frames.

Our block-based, adaptive video CS approach is inspired by the work of Aaron et al. [67], L. Gan [157], Mun et al. [133] and Do et al. [51]. However, the random measurement matrices are replaced by 2D DCT matrices, adaptive block measurements are introduced, the frames are

reconstructed in real-time using an inverse DCT, and optionally enhanced using the IDA algorithm developed in section 4.5. Temporal correlation is exploited mainly at the decoder with Differential Pulse Code Modulation (DPCM) and motion estimation or video frame interpolation performed there. However, the encoder also exploits temporal correlation by adapting the block measurements based on the transform coefficients measured in previous frames, and in the low-complexity coders that encode transform coefficients using DPCM prior to transmission.

8.3 End-to-end distributed VCS

8.3.1 System diagram

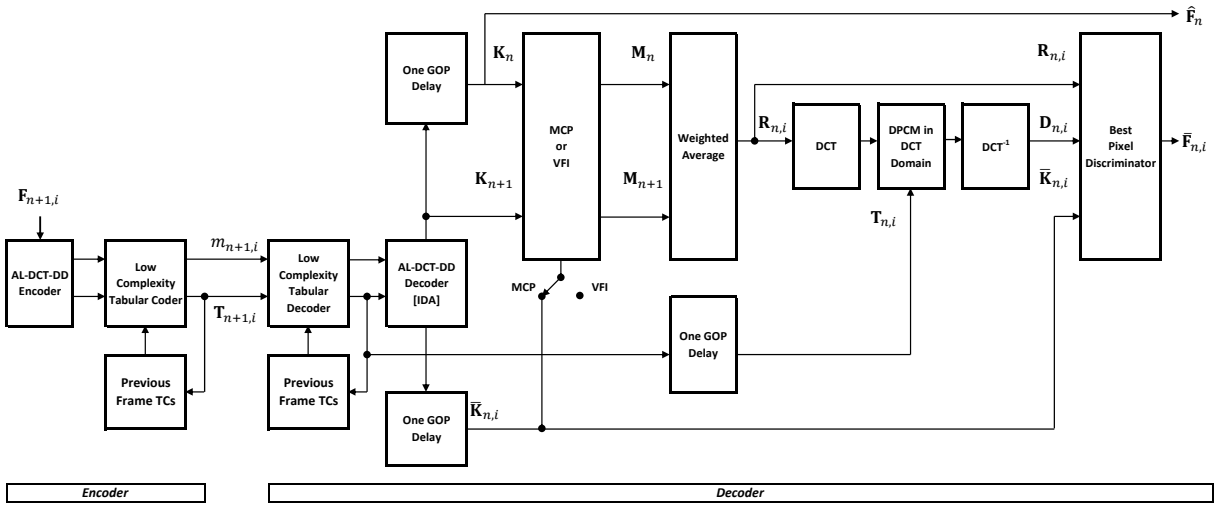


Figure 8.1: Block diagram of the distributed VAL-[IDA]-{MCP|VFI}-TAB algorithm

The VAL-[IDA]-{MCP|VFI} codec was modified to quantize and low-complexity encode the adaptive transform coefficients at the encoder. The system block diagram is shown in figure 8.1. At the decoder, the low-complexity code is decoded and inverse quantization recovers the transform coefficients that are then processed by the unmodified VAL-[IDA]-{MCP|VFI} algorithms.

The key frame and non-key frame quantizers are separate and do not capitalize on temporal correlation between the frames. However, the entropy coder stores the transmitted transform coefficients of the previous frames and entropy encodes the difference of transform coefficients. Therefore, the entropy encoder capitalizes on correlation between the quantized coefficients of the current and previous frames.

8.3.2 Quantization

A modified JPEG quantization matrix, QJpeg-HVS, matched to the Human Vision System (HVS) [128] is used to quantize the DCT transform coefficients. The QJpeg-HVS quantization matrix is of size 8. In order to generate a 16×16 quantization matrix the Matlab imresize function is used to expand the matrix. In order to vary quantization, all the quantizers in this

Algorithm 8.1 TAB-PNG Encoder

Input: Image $\boldsymbol{I}$, compression factor C_F, block size B.

Output: Compressed difference matrix $\boldsymbol{D}$, column position of last non-zero Transform Coefficient (TC) in each row $\mathbf{m_i}$, position vector of removed all-zero columns $\mathbf{p_z}$, negative TC with the largest absolute value $\mathscr{T}min$, column, row and value vectors of clipped TCs $(\mathbf{x}, \mathbf{y}, \mathbf{v_C})$.

1. Sense an image using the block-based [A]L-DCT-{ZZ|BBV|DD} algorithm at compression factor C_F and block size B.
2. Quantize the resulting $B \times B$ TC blocks.
3. Store the qauntized TCs in a tabular array, one block of zigzag-scanned, quantized TCs per row.
4. Transmit the column position of the rightmost non-zero TCs per row, to the decoder, equal to $\mathbf{m_i}$.
5. Subtract the previous frame's TCs from the current TCs to create a TC difference matrix.
6. Remove all-zero columns and send the position vector $\mathbf{p_z}$ of these to the decoder.
7. Subtract the the negative TC with the largest absolute value $\mathscr{T}_{min}$, from the difference matrix such that it becomes positive definite. Send $\mathscr{T}_{min}$ to the decoder.
8. Clip any shifted TC value that exceeds 255, transmitting the clipped values to the decoder using a fixed length $(\mathbf{x}, \mathbf{y}, \mathbf{V_C})$ code where $\mathbf{x}$ and $\mathbf{y}$ are the row and column addresses and $\mathbf{V_C}$ are the clipped values.
9. Code the positive definite difference matrix, whose elements have values between 0 and 255 and are stored as bytes, using the PNG image coding algorithm and transmit the compressed difference matrix $\boldsymbol{D}$ to the decoder.

matrix are multiplied by a factor Q_M. In table 8.1 Q_M values of 2, 1 and 0.75 are employed, for example.

8.3.3 Low-complexity entropy coding

The TAB algorithm introduced in section 6.3.2 uses DPCM to code the difference of the quantized TCs efficiently. The GOP structure adopted means that the key frames are compressed with higher quality than non-key frames and hence have more quantized transform coefficients. Key frames will therefore have a big impact on the overall transmission bit rate achieved.

A further level of structure is therefore introduced in the entropy encoded layer, that of intra and inter entropy coded quantized transform coefficients. The quantized transform coefficients of the intra entropy coded key frames are entropy coded independently. In inter-coded key and non-key frames, the transmitted quantized transform coefficients of the previous frame, stored in the previous frame TCs store (as shown in figure 8.1) are subtracted from those of the current quantized TC coefficients prior to entropy coding. This capitalizes on the fact that quantized transform coefficients are correlated with those at the same position, in previous frames. This

Algorithm 8.2 TAB-PNG Decoder

Input: Compressed difference matrix $\boldsymbol{D}$, column position of last non-zero TC in each row $\mathbf{m_i}$, position vector of removed all-zero columns $\mathbf{p_z}$, negative TC with the largest absolute value $\mathscr{T}_{min}$, column, row and value vectors of clipped TCs $(\mathbf{x}, \mathbf{y}, \mathbf{v_C})$, block size B, image resolution $H \times W$.
Output: Output Image $\boldsymbol{\mathscr{O}}$

1. Decompress the difference matrix $\boldsymbol{D}$ using the PNG decoding algorithm.
2. Add the clipped values $(\mathbf{x}, \mathbf{y}, \mathbf{v_C})$ to the TCs.
3. Add $\mathscr{T}_{min}$ to the TC difference matrix so that its elements become integers.
4. Insert any zero columns back into the difference matrix at $\mathbf{p_z}$.
5. Add the difference matrix $\boldsymbol{D}$ to the TCs of the previous frame stored at the decoder, adding only the TCs from the DC value up to the rightmost non-zero AC TCs as indicated by $\mathbf{m_i}$.
6. Reconstitute the TCs into $B \times B$ blocks and reconstruct them using the inverse 2D DCT operation.
7. Form the blocks into the output image $\boldsymbol{\mathscr{O}}$.

also allows the larger key frames to benefit because their low-pass quantized coefficients can also be differentially encoded with the previous non-key frame TCs as reference. This scheme is similar to that used by Mun and Fowler in [116], where they used DPCM to code CS measurements at the same position in block-based compressed sensing of images.

The TAB algorithm described in section 6.3.2 was modified to improve its performance. The ensuing TAB-PNG encoder algorithm is described in algorithm 8.1, and the decoder in algorithm 8.2.

8.4 Results

All the proposed algorithms in this section use the TAB-PNG (TP) low complexity entropy coding and decoding schemes described in algorithms 8.1 and 8.2 respectively. Unless otherwise stated, the AL-DCT-DD algorithm is employed at the encoder as this exhibited the best performance in chapter 7.

8.4.1 Rate-distortion performance of key frames

In DVC, key frames are crucial to the rate-distortion performance of the video coding algorithm. Following the publication of the DISCOVER algorithm [26], many DVC schemes in the literature code key frames using the H.264/AVC Main Profile intra mode algorithm [147]. This algorithm assumes that the whole image is available at the encoder. Our algorithms are limited to measuring the transform coefficients, albeit with a high compression ratio δ_K.

The Peak Signal-to-Noise Ratio (PSNR) versus bit rate performance of the intracoding

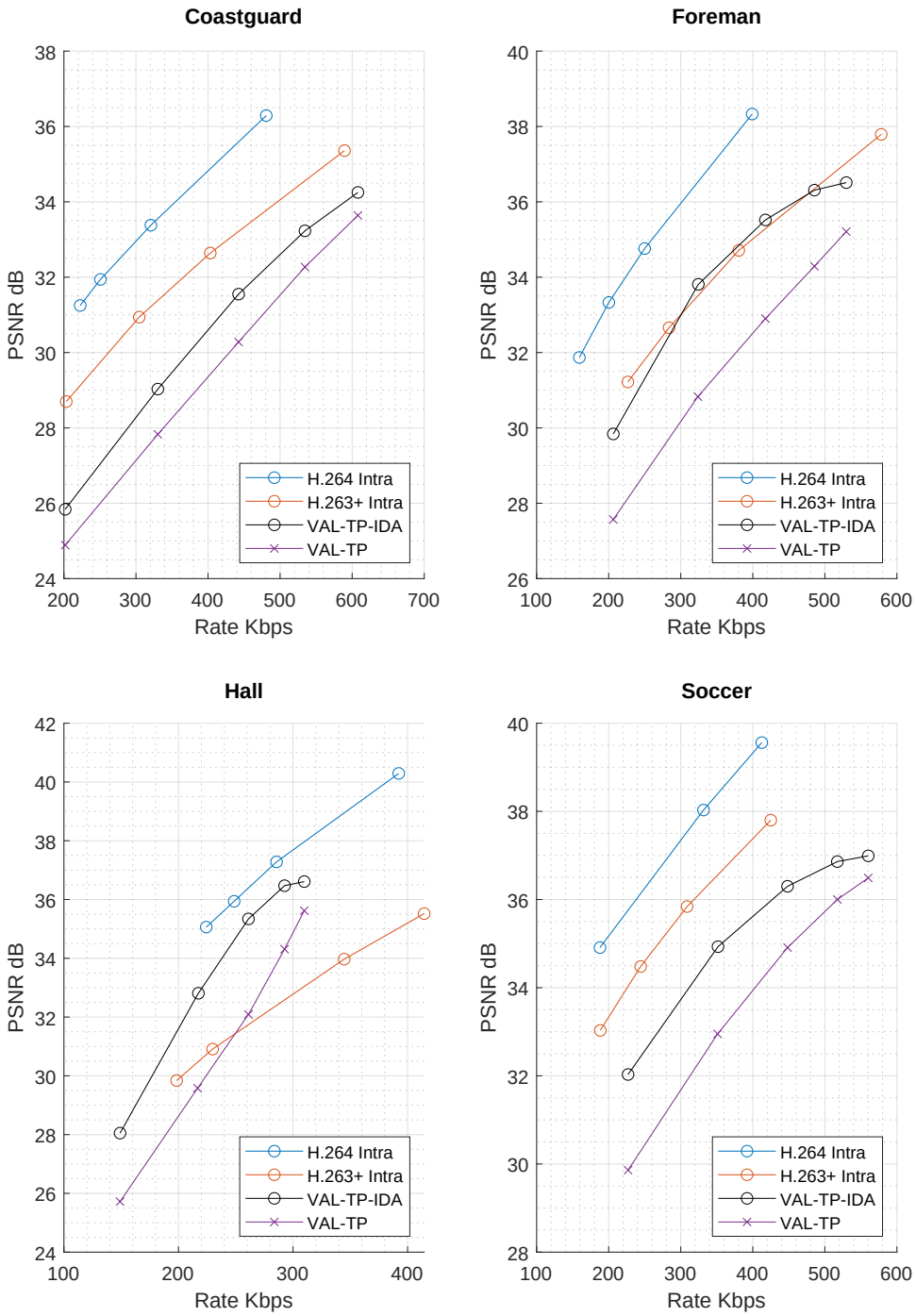


Figure 8.2: VAL-TP and VAL-TP-IDA compared with the intra mode of H263+ and H264/AVC when coding 144 frames of the four Quarter CIF (QCIF) video sequences Coastguard, Foreman, Hall and Soccer, sub-sampled to 15 frames per second. The plotted VAL-TP and VAL-TP-IDA points were measured with a compression ratio $\delta_k = \{0.10..0.50\}$.

modes of the H.264/AVC and H.263+ video coding standards are compared with the VAL-TP and VAL-TP-IDA algorithms when coding the key frames without the use of DPCM, MCP or Video Frame Interpolation (VFI), in figure 8.2. The VAL-TP-[IDA] algorithms use a block size of 16×16 pixels, a GOP size of 4, and the IDA algorithm uses a damping factor of 1.4 and 40 iterations of the DnCNN denoiser. In this chapter H.264/AVC intra, H.263+ intra and DISCOVER results were retrieved from the website accompanying the paper [26], as were the four QCIF (176×144 pixels) sequences used in the tests.

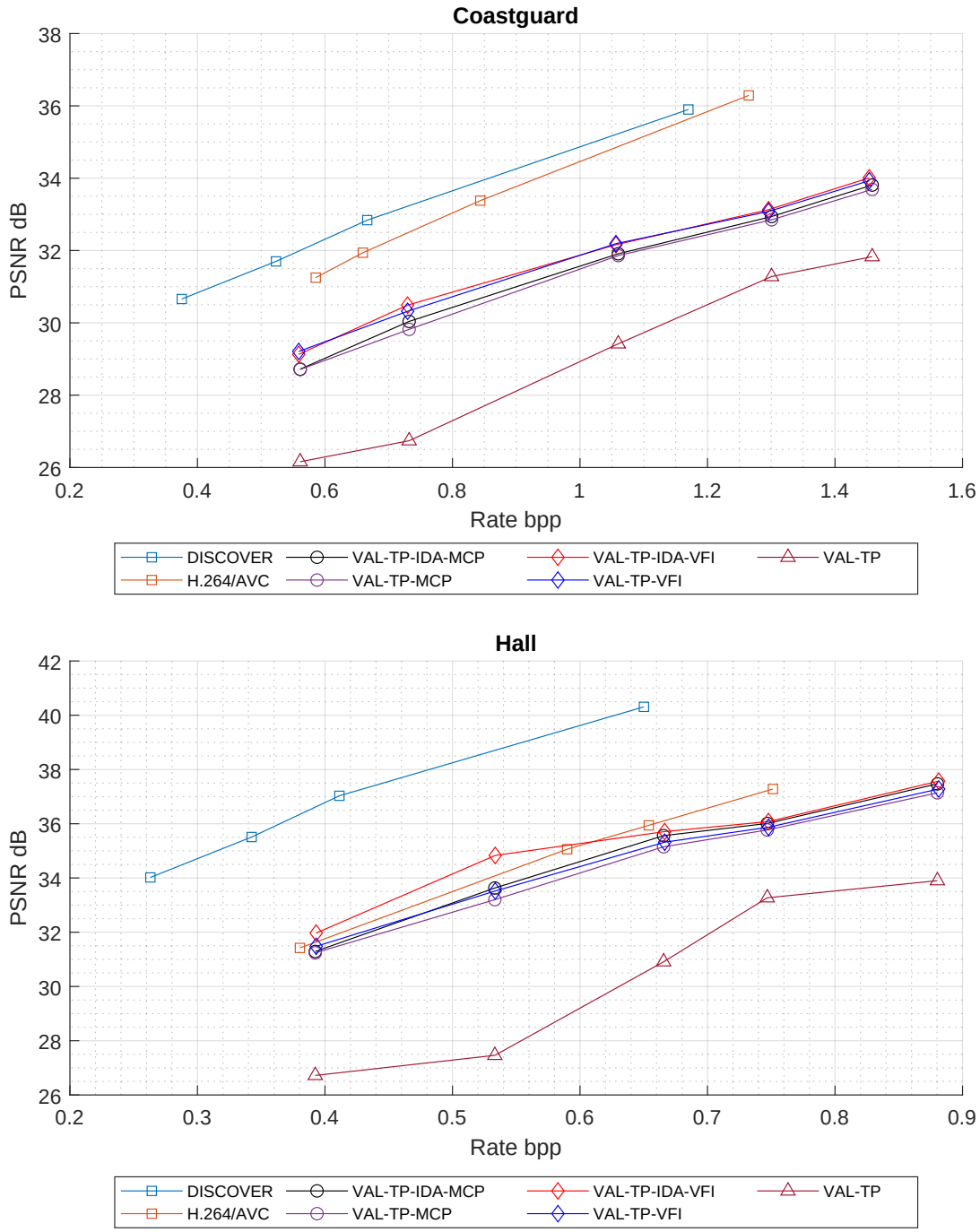


Figure 8.3: VAL-TP-[IDA]-{MCP|VFI} compared to DISCOVER and the intra mode of H264/AVC on the Coastguard and Hall QCIF sequences.

From figure 8.2, VAL-TP-IDA outperforms VAL-TP by around 2 dB consistently, but this comes at the expense of complexity with VAL-TP-IDA being two orders of magnitude slower than VAL-TP. H.264/AVC intra mode is clearly superior to the H.263+ intra mode and the VAL-TP and VAL-TP-IDA key frame algorithms. However, the performance gap is dependent on the video sequence with VAL-TP-IDA exceeding the H.263+ intra-mode and approaching H.264/AVC intra-mode in the Foreman and Hall sequences.

Given that DISCOVER is designed to improve the rate-distortion performance by using the superior H.264/AVC intra-mode and our VAL-TP-[IDA]-{MCP|VFI} algorithms improve the non-key frames based on the VAL-TP and VAL-TP-IDA key frames in figure 8.2, it is clear that

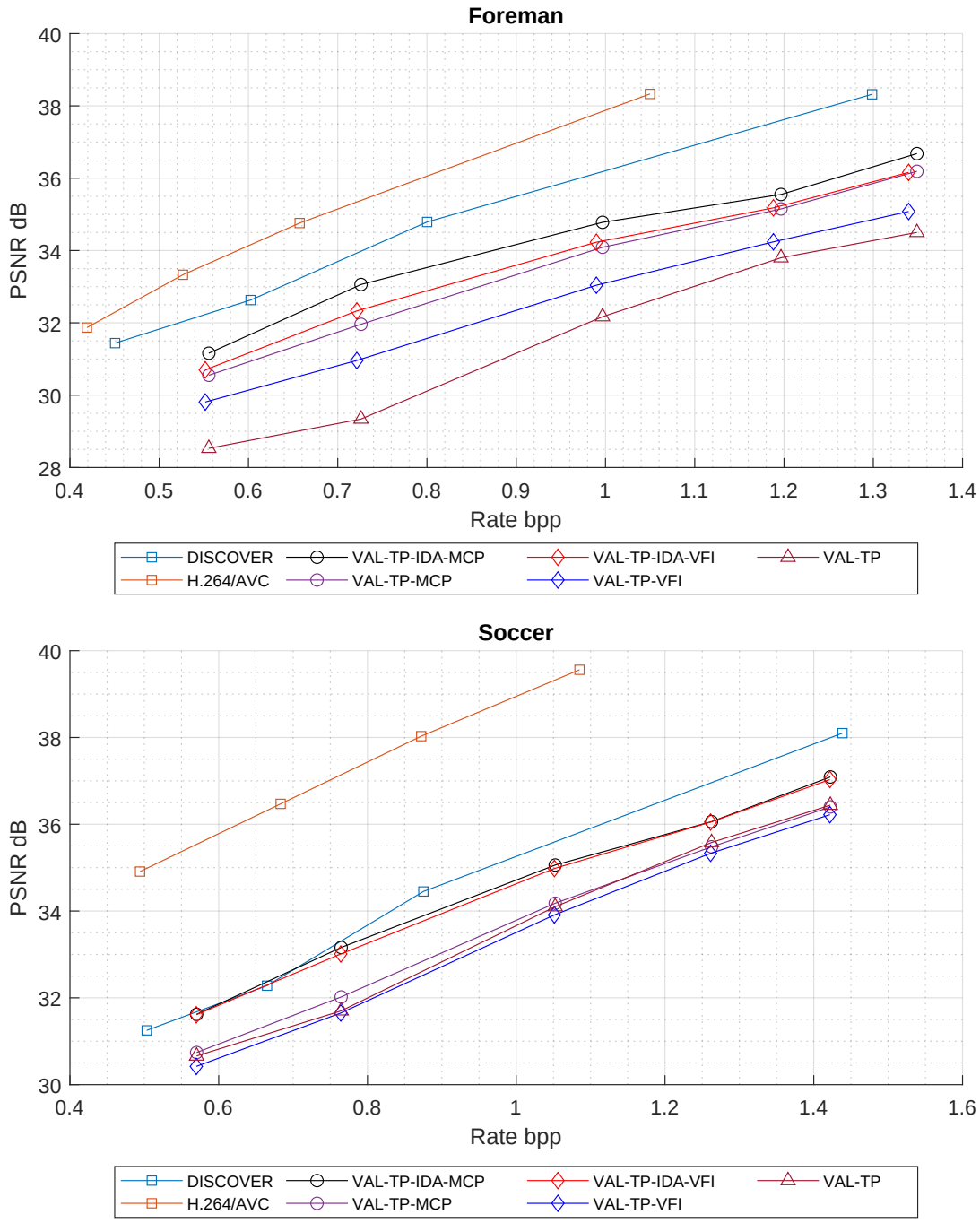


Figure 8.4: VAL-TP-[IDA]-{MCP|VFI} compared to DISCOVER and the intra mode of H264/AVC on the Foreman and Soccer QCIF sequences.

DISCOVER has a significant head start.

8.4.2 Rate-distortion performance when exploiting temporal correlation

The performance of the VAL-TP-IDA-{MCP|VFI} algorithms are shown in figures 8.3 and 8.4. The DISCOVER and H.264/AVC results were downloaded from the DISCOVER website [154]. The DISCOVER and VAL-TP-[IDA]-{MCP|VFI} results are for the GOP with four frames. In the VAL-TP-[IDA]-MCP algorithms, the block size used is 16×16, motion estimation uses the ARPS algorithm with motion vector block of size 16×16 and the motion search area ± 16. The

G	δ_K	$\delta_{\bar{K}}$	$\tilde{\delta}$	Q_M
4	0.4	0.1	0.175	2
4	0.4	0.1	0.175	1
4	0.6	0.2	0.3	1
4	0.7	0.3	0.4	1
4	0.7	0.3	0.4	0.75

Table 8.1: The parameters used to derive the rate distortion points in the figures in this chapter with the top row pertaining to the worst performance and the bottom row equivalent to the best performance.

VAL-TP-[IDA]-VFI algorithms use the Depth-aware Video Frame Interpolation (DAIN) VFI Convolutional Neural Network (CNN) introduced in section 7.5. The IDA algorithm is used optionally to enhance the frames, with a damping factor of 1.4 and executes 40 iterations of the DnCNN denoiser. The best pixel discriminator threshold is set to 10 when IDA is in use and to 25 otherwise.

The GOP size G, compression ratios δ_K and $\delta_{\bar{K}}$ and the quantization matrix multiplier Q_M for the VAL-TP-IDA-MCP algorithm, used to generate the PSNR points in figures 8.3 and 8.4 are shown in table 8.1. In order to reduce energy consumption, compressive sensing is employed with an average compression ratio $\tilde{\delta} = 0.175$, sensing 1 measurement every 5.71 pixels, for example . The average compression ratio is given by $\tilde{\delta} = \frac{(\delta_K + (G-1)\delta_{\bar{K}})}{G}$.

VAL-TP-IDA-MCP comes within 1 dB of the performance of the DISCOVER codec in the Foreman sequence and matches its performance in the Soccer sequences below 0.7 bpp. However, the minimum PSNR performance gap in the Coastguard and Hall sequences shown in figure 8.3 is 3.5 dB and 4.2 dB respectively. VAL-TP-IDA-MCP PSNR performance matches that of the H.264/AVC intra mode but remains far from the DISCOVER PSNR.

The performance of VAL-TP-IDA-VFI is better than VAL-TP-IDA-MCP on the Coastguard and Hall sequences, whereas VAL-TP-IDA-MCP is better on the Foreman and the Soccer sequences. The coastguard and Hall sequences have smoother motion, whereas the Foreman and Soccer sequences contain more complex motions. This implies that VFI is better at estimating smoother motion than MCP.

8.4.3 Ablation study

Exploiting temporal correlation

Figures 8.5 and 8.6 show the improvement in PSNR and Structural Similarity Index Measure (SSIM) performance from spatial compression without exploiting temporal correlation, to MCP-DPCM (or VFI-DPCM), and then improving the spatial quality again using IDA .

The results were generated using the same parameters in table 8.1. The VAL-TP results were generated without DPCM and MCP, and represent the PSNR and SSIM performance of the spatial AL-DCT-DD algorithm. The exploitation of temporal correlation, using DPCM and MCP, increases the PSNR by up to 3 dB, 2 dB, 6 dB and 1 dB for the Coastguard, Foreman,

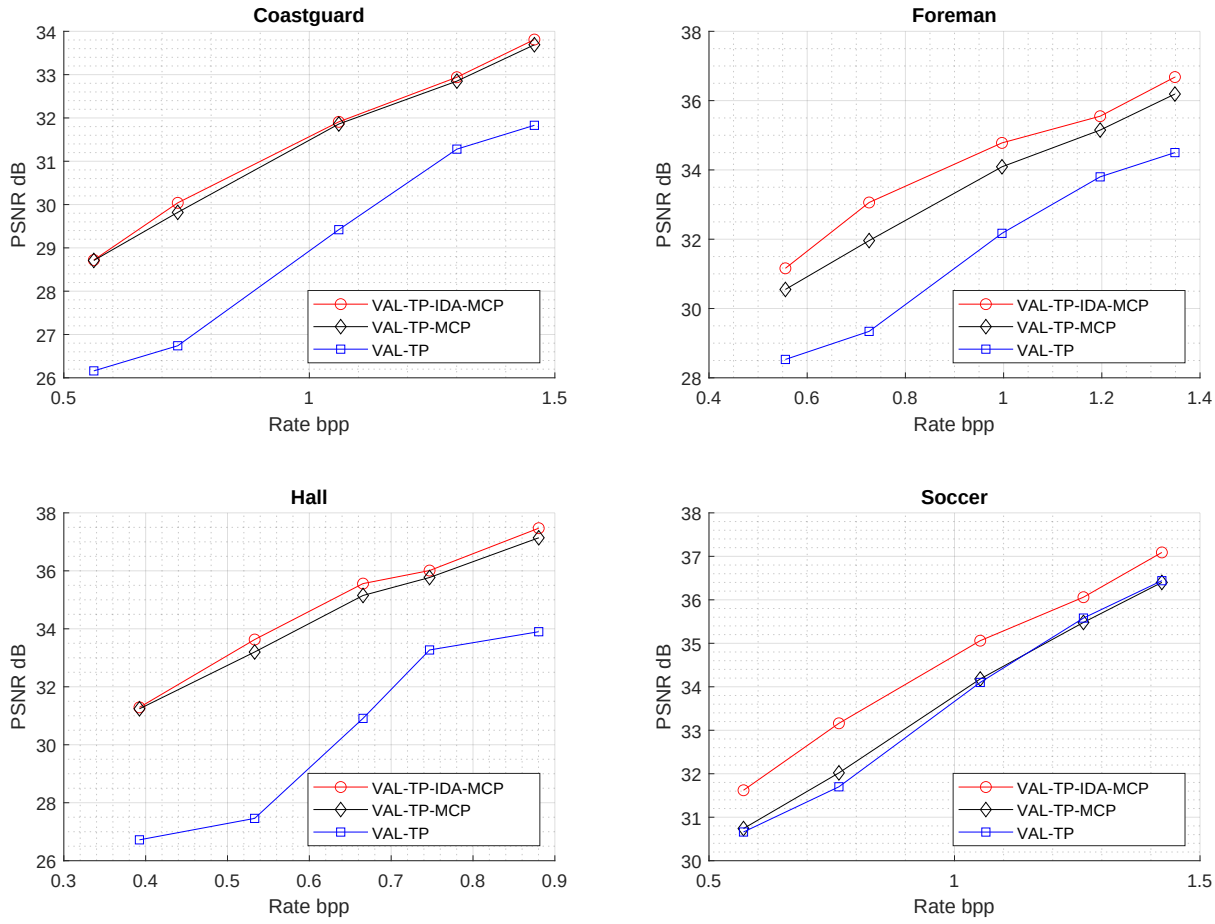


Figure 8.5: Improvement in VAL PSNR performance with MCP-DPCM and IDA .

Hall and Soccer QCIF sequences, respectively. Employing IDA , further increases performance by up to 1 dB. The same trend in performance increase is observed in SSIM as shown in figure 8.6.

Comparison between VAL and VLZ algorithms

The VAL algorithms use the AL-DCT-DD algorithm at the encoder. It is of interest to compare VAL algorithms with VLZ algorithms that use L-DCT-ZZ at the encoder instead.

Figure 8.7 shows the comparison between VAL-TP-IDA-VFI and VLZ-TP-IDA-VFI on the four QCIF sequences. The PSNR performance is very close on all sequences. This is unlike the behaviour of L-DCT-ZZ-TAB and AL-DCT-DD-TAB on the image tests in chapter 6 where the TAB algorithm was better suited at compressing the quantized L-DCT-ZZ coefficients. An inspection of figure 8.7 shows that the VAL-TP-IDA-VFI points have a higher PSNR than the respective VLZ-TP-IDA-VFI points, at the same compression ratio. Therefore, if TAB-PNG can be improved further, the rate-distortion performance of VAL-TP-IDA-VFI will exceed that of VLZ-TP-IDA-VFI. This can be pursued by designing a scheme that better preserves the high frequency content recovered by AL-DCT-DD.

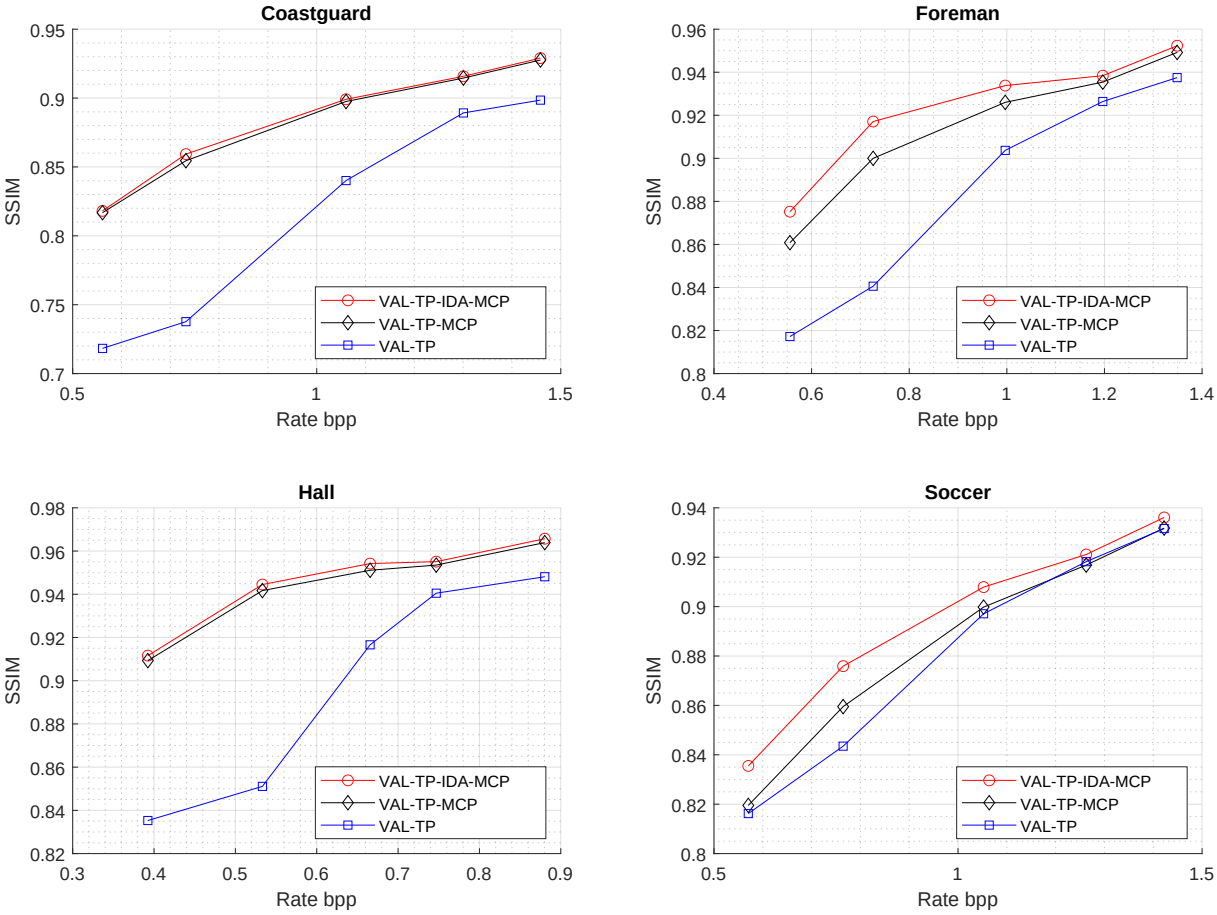


Figure 8.6: Improvement in VAL SSIM performance with MCP-DPCM and IDA .

8.5 Sensor analysis

This section analyzes the block adaptive VAL algorithm using the techniques developed in chapter 3 and section 3.2. Seven of the framework parameters are compared.

8.5.1 PSNR quality performance

In this chapter the quality of the VAL-TP-[IDA]-{MCP|VFI} algorithms have been compared with the DISCOVER codec as shown in figures 8.3 and 8.4. VAL-TP-VFI comes within 2.75 dB of DISCOVER on the Coastguard sequence at 0.6 bpp. VAL-TP-IDA-VFI comes within 4 dB of the Hall sequence at 0.5 bpp. VAL-TP-IDA-MCP comes within 1 dB at 0.7 bpp and matches its performance on the Soccer sequence below 0.7 bpp.

8.5.2 Energy to modulate mirrors

The VAL-TP-IDA-{MCP|VFI} algorithms use a GOP with key frames and non-key frames having compression ratio δ_K and $\delta_{\bar{K}}$ respectively. If the size of the frame is $N = H \times W$ pixels and the blocks are $B \times B$, then the energy $E_{M,K}$ to sense one key frame is given by

$$E_{M,K} = \delta_K N B^2 e_M, \tag{8.1}$$

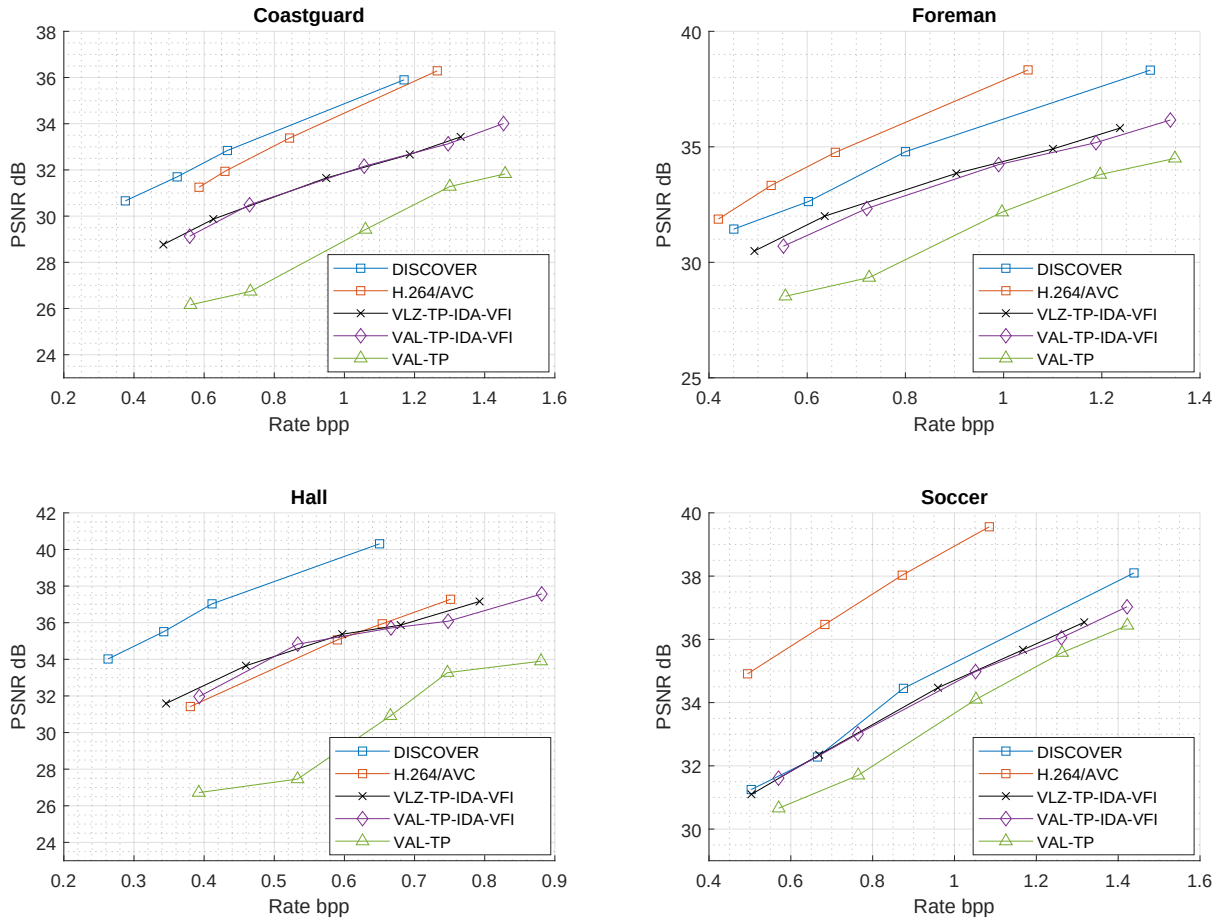


Figure 8.7: Comparison between VAL-TP-IDA-VFI and VLZ-TP-IDA-VFI.

and for a non-key frame

$$E_{M,\bar{K}} = \delta_{\bar{K}} N B^2 e_M, \tag{8.2}$$

where e_M is the energy required to move one mirror. The energy to capture one GOP with G frames is then given by

$$E_{M,G} = E_{M,K} + (G-1)E_{M,\bar{K}}. \tag{8.3}$$

8.5.3 Energy to perform analogue to digital conversion

Each of the $\delta_K \times N$ and $\delta_{\bar{K}} \times N$ key frame and non-key frame transform coefficients requires one analogue to digital conversion. Hence the energy to convert one GOP is given by

$$E_{ADC,G} = N(\delta_K + (G-1)\delta_{\bar{K}})e_{ADC}, \tag{8.4}$$

where e_{ADC} is the energy required for one Analogue to Digital Converter (ADC) conversion. Sensors that capture full frames require $E_{ADC} = N \times e_{ADC}$ and are thus less energy efficient by a factor of $\frac{1}{(\delta_K + (G-1)\times\delta_{\bar{K}})}$ when performing ADC conversions.

8.5.4 Encoder complexity

The VAL encoder has low complexity compared to classical DVC encoders. The key and non-key transform coefficients are collected in the analogue domain and require $\delta_K \times N$ and $\delta_{\bar{K}} \times N$ analogue to digital conversions respectively. Quantization requires one division per transform coefficient. The adaptive algorithms are dominated by the sorting of $\delta_K \times N$ transform coefficients to determine phase-2 measurements.

The TAB-PNG algorithm first stores the phase-1 and phase-2 quantized coefficients in an n_B by $B \times B$ array and removes columns with all zero entries. It then finds the minimum value in one scan of the array and shifts all entries to be positive. The resulting positive definite array is then compressed using the Matlab imwrite version of the PNG [118] compression algorithm.

The complexity of the TAB-PNG algorithm is comparable with the DVC Wyner-Ziv frame coding algorithm in [26], that also computes 2D DCT blocks, except that our algorithms do not collect all the pixels in the first place but compute the M transform coefficients in the analogue domain. The complexity of the TAB-PNG algorithm is proportional to the compression ratio of frames, for example $\delta_K : \delta_{\bar{K}}$, so that key frames have a complexity that is less than or equal to four times that of non-key frames, from table 8.1. Artigas et al. report that the encoding time complexity of a Wyner-Ziv frame is ten times smaller than a key frame [158]. Therefore the DISCOVER algorithm is more than twice as complex as the VAL algorithm over one GOP.

The VAL algorithm does not require the Slepian-Wolf codec with the feedback channel, and if this is also factored in, then the VAL algorithms would be even less complex than DVC.

8.5.5 Energy for entropy encoding

The operations required to encode pixels or transform coefficients prior to transmission are referred to here as Entropy encoding operations. It is difficult to estimate the energy required for this operation as this would be dependent on the Central Processing Unit (CPU) that is used to compute these algorithms. The proper solution to this problem, would be to run the algorithms on the target hardware and measure their energy consumption. Another option is to compare their run-time on a simulation platform using the same simulation environment, e.g. Matlab. But even in this case, a fair comparison would require that the compared algorithms benefit the same from code acceleration. The comparison proposed here requires the estimation of the algorithms' complexity measured in operations per frame that is then multiplied by the hypothetical energy consumed per operation e_{EE}. Therefore, given that the VAL complexity is less than half that of DISCOVER, it is possible to conclude that it is potentially twice as energy efficient.

8.5.6 Energy to transmit a video frame

The power to transmit video from the sensor is proportional to the bit rate. Assuming that the algorithms are tuned to produce the same bit rate and frame rate, then the energy required to transmit a frame would be the same. In such a case, the PSNR achieved will be different. For

example DISCOVER has a PSNR advantage of 1.25 dB at 0.8 bpp over VAL-TP-IDA-MCP on the Foreman sequence. However, if two encoders are compared at the same PSNR, then the energy required to transmit a frame would be proportional to the bit rate. For example, considering the Foreman sequence in figure 8.4, when the PSNR is 34 dB, DISCOVER has a 0.14 bpp advantage over VAL-TP-IDA-MCP, that translates to $(0.14 \times 176 \times 144 \times 15)/1000 =$ 53.2 kbps.

8.5.7 Time to capture one frame

The time to capture one frame T_F on a SPC is proportional to the time it takes to modulate N mirrors for each of the M measurements. If the time to modulate one mirror is t_m, then:

$$T_{M,SPC} = t_m NM = t_m \delta N^2, \tag{8.5}$$

where δ is the compression ratio. It is reasonable to assume that the ADC and entropy coding blocks are tuned such that they can process frames as fast as they are captured from the SPC. If an MPC is used instead (or an FPPS), and it is assumed that measurements are collected from each block simultaneously, then the capture time becomes

$$T_{M,MPC} = t_m(\delta N)B^2 = t_m MB^2. \tag{8.6}$$

Thus an MPC allows a video sensor to sense frames $(\delta N^2)/(N_{MB}B^2)$ faster. This also requires one ADC conversion per block that can process TCs at this same rate. Alternatively, a faster ADC can be multiplexed to read all block measurements.

Assuming MPC block sizes of $B^2 = 8^2, 16^2$ and 32^2, an average δ of 0.175, and a frame rate F_R of 30 fps, a Digital Micromirror Device (DMD) is required to support $\delta \times B^2 \times F_R$ measurements per second, i.e. 336, 1,344 and 5,376 measurements per second respectively. The DLP9500 1920×1080 pixel DMD from Texas Instruments [63] supports the measurement of 8 bit binary patterns at 2,893 Hz. Therefore, an MPC with 8×8, 16×16 and 32×32 blocks can be sensed at 258, 64 and 16 fps respectively.

8.6 Chapter summary

This chapter has introduced an end-to-end video compressive transmission architecture that uses an adaptive block-based DCT to measure transform coefficients and encode these with low complexity, using the TAB-PNG algorithm described in section 8.3.3. The decoder uses MCP or VFI and DPCM to capitalise on temporal correlation between key and non-key frames. The VAL-TP-IDA-VFI algorithm produces results that match those of the DISCOVER codec's PSNR on the Soccer sequence below 0.7 bpp but lags behind on the other sequences. This is mainly due to the use of the significantly more complex H264/AVC intra mode coder used for key frames. However VAL-TP-IDA-VFI and VAL-TP-IDA-MCP sense both key frames and

non-key frames compressively, have lower encoder complexity and do not require a feedback channel.

Future work can investigate whether the performance of the VAL-MCP or VAL-VFI algorithms can be improved by exploiting feedback, for example by using Slepian-Wolf [2] techniques to transmit non-key frame data.

The comparison between VAL-TP-IDA-VFI and VLZ-TP-IDA-VFI shows that there is further room for improving the low complexity entropy coding algorithm. This is because TAB-PNG still performs marginally better on the latter algorithm, whereas AL-DCT-DD-IDA has a clear advantage on L-DCT-ZZ-IDA as shown in chapter 4.

Chapter 9

Conclusions and further work

This dissertation studies the design, implementation and performance of Video Compressive Sensing (VCS) algorithms using deterministic sensing matrices, that allow low computational complexity reconstruction in real time. A target application is low-power video sensors, for example as used in wireless video sensor networks, with a frame rate of at least ten frames per second. Another application is real-time video super resolution sensed on Multi Pixel Cameras (MPCs). The main conclusions drawn from this dissertation are presented in this chapter, and possible areas for future work are summarised.

9.1 Conclusions

The main conclusion in this dissertation is that given a Single Pixel Camera (SPC), Multi Pixel Camera (MPC) or Focal Plane Processing Sensors (FPPS), Distributed Video Compressive Sensing (DVCS) algorithms can be implemented, using the lowpass coefficients of a 2D Discrete Cosine Transform (DCT) in zigzag order as the deterministic sensing matrix, that achieve state-of-the-art Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) performance.

The main algorithms developed, adopt a Group of Pictures (GOP) structure that contains both key frames and non-key frames that are equivalent to the intra-coded and inter-coded frames in the classical motion-compensated hybrid DPCM/DCT video coders that currently dominate digital video systems. Both key and non-key frames are acquired, without prediction, from the compressive sensors; Motion Compensated Prediction (MCP) or Video Frame Interpolation (VFI), and Differential Pulse Code Modulation (DPCM) are computed at the decoder, to lower sensor-side complexity and power consumption.

Adaptive block compressive sensing of still frames

Increasing the quality of the key frames is essential in this configuration as it leads to an improvement of the non-key frames, and the overall video quality. Adaptive, block-based, compressed sensing techniques that exploit spatial correlation of individual key and non-key frames

are studied in chapter 4. The 2D DCT is chosen as the measurement matrix because it allows initial reconstruction in tens of milliseconds.

Block-based image Compressive Sensing (CS) is adopted because it allows real-time capture on MPC and FPPS. However Block-based CS (BCS) suffers from problems related to the mismatch between the variable sparsity in the sub-image blocks and a fixed compression ratio. The AL-DCT-BBV and AL-DCT-DD algorithms are proposed that adapt the compression ratio to the sparsity in each block at the encoder. The two algorithms estimate sparsity from an initial set of measurements and then continue to collect further Transform Coefficients (TCs) based on this estimate. Both sets of measurements are transmitted to the decoder, the only overhead being the number of TCs per block. AL-DCT-DD was found to perform better at low compression ratios, whereas AL-DCT-BBV performed better at $C_R > 0.1$.

Blocking artefacts can be removed in real-time by applying either of two low-complexity filters developed for the purpose, Zig-Zag Scan Filter (ZZF) and Threshold Filter (THF). Alternatively, deblocking and an improvement in quality is achieved using an iterative denoising algorithm developed from one of the most successful image CS reconstruction techniques in the literature, Denoising AMP (D-AMP) [14]. This is achieved by interpreting the D-AMP algorithm as a feedback system and simplifying the feedback loop. This stabilizes the loop whilst accelerating the algorithm.

The algorithms reconstructed using Iterative Denoising Algorithm (IDA), AL-DCT-DD-IDA and AL-DCT-BBV-IDA, have been shown to achieve state-of-the-art performance on still images and individual video frames. However, IDA cannot be reconstructed in real-time even though it uses the Graphics Processing Unit (GPU) accelerated Denoising CNN (DnCNN) [27] as the loop denoiser.

Downsampling and super resolution as a low-complexity compression scheme

The MPC with an $H \times W$ Digital Micromirror Device (DMD), and an $h \times w$ 2D array of sensors, $h < H$ and $w < W$ has been described as an image super resolution technique in the literature. In chapter 5, inspired by recent advances in GPU-implemented image and video super resolution Convolutional Neural Networks (CNNs), image subsampling followed by CNN-based super-resolution is proposed as a viable, real-time compression technique. The best image super resolution results in the literature are obtained by training the super resolution CNNs with low resolution images obtained by downsampling using the Matlab imresize function. The work in this chapter shows how this interpolation and anti-aliasing function can be implemented as a convolution operation on an SPC, MPC or FPPS, resulting in the Shrink algorithm. Surprisingly, the performance of image Super Resolution (SR) algorithms is shown to be superior to that of image CS algorithms from a dimensionality reduction perspective below $C_R = 0.3$, although AL-DCT-DD-IDA is better above $C_R = 0.3$. This advantage does not persist in end-to-end systems.

End-to-end transmission of still images and frames

End-to-end image or video frame transmission requires the compressive measurements in CS, or the downsampled samples in SR, to be quantized and encoded for digital transmission. Chapter 6, investigates how to achieve this with low complexity on image or video sensors.

In the case of image compressive sensing, two algorithms are investigated to digitally encode L-DCT-ZZ, AL-DCT-DD and AL-DCT-BBV CS measurements prior to transmission. The first algorithm adapts the quantization and run-length coding scheme in JPEG for this purpose. The simpler TAB algorithm, arranges block TCs in a table and then encodes the columns using fixed length codes adapted to the TCs in the columns.

When using the JPEG-style compression algorithm, it is found that L-DCT-ZZ-JPEG has the same performance of AL-DCT-DD-JPEG, because the JPEG quantization matrix renders L-DCT-ZZ adaptive as well, since a variable number of the L-DCT-ZZ block TCs are quantized to zero. A rate distortion algorithm, RD, is defined for both JPEG and TAB encoding, that selects the best PSNR performance whilst minimizing the number of compression measurements and bit rate. L-DCT-ZZ-JPEG-RD achieves the same performance as JPEG, but with the transform coefficients sensed energy-efficiently on the CS camera. When IDA is used for reconstruction, the PSNR and SSIM performance exceeds that of JPEG substantially, but not in real time. L-DCT-ZZ-TAB-RD exceeds the performance of L-DCT-ZZ-JPEG-RD below 0.3 bpp, mainly because it uses 32×32 blocks instead of the 8×8 blocks in the JPEG variant.

In the case of the Shrink image super resolution algorithm, a low-complexity technique based on DPCM followed by Golomb coding, is investigated in the same chapter. A very fast and low-complexity technique, EE1, is developed to estimate the Golomb m parameter energy-efficiently. Alternatively, normal image compression schemes, such as JPEG2000, JPEG or PNG can also compress the low-resolution images produced by Shrink. The PNG variant of the Shrink algorithm, super resolved using the plug-and-play CNN-based IMDN algorithm - Shrink-PNG-IMDN - also exceeds JPEG performance below 0.3 bpp.

Distributed video compressive sensing: dimensionality reduction

In chapter 7, effective and efficient video CS and video SR algorithms are investigated that leverage the image CS and SR techniques developed in chapters 4 and 5. Furthermore, these algorithms exploit correlation between frames at the CS encoder and at the decoder, inspired by distributed video coding techniques.

In the case of VCS, temporal correlation is first exploited at the encoder, by using the correlation between frames to produce improved temporal DCT domain adaptive algorithms. These add little computational complexity at the encoder, leveraging the transform coefficients that are collected in previous frames to improve the estimation of sparsity in sub-image blocks.

Temporal correlation is exploited at the decoder by employing either block motion compensated prediction or CNN-based video frame interpolation, to predict non-key frames from the higher quality key frames. The predicted non-key frames are combined with the received non-key frames in a temporal DPCM operation that further improves the quality of the non-

key frames. The DPCM is performed in the DCT domain using a technique developed for the L-DCT-ZZ based algorithms. At this stage, there are three versions of the non-key frame, the low-resolution non-key frame reconstructed from the received non-key TCs, the high-resolution version predicted by MCP or VFI, and the DPCM output. A Best Pixel Discriminator (BPD) is defined that selects the best pixel to output based on these three versions and leads to an improvement in PSNR and SSIM performance.

An ablation study shows that VAL-MCP developed in this chapter exhibits a large PSNR improvement of around 5.4 dB over the VAL algorithm that does not exploit temporal correlation. Multiscale SSIM (MS-SSIM) increases from 0.9563 to 0.9847 This shows the effectiveness of the DPCM and MCP algorithms developed in this chapter. Temporal DPCM contributes 3.5 dB to the PSNR improvement and 0.01 to the MS-SSIM improvement, whereas MCP and VFI contribute a further 2 dB and 0.02.

The VAL-MCP algorithm, using block-based MCP can transmit and reconstruct video on Matlab at around 10 frames per second without using GPU acceleration. The PSNR performance is 0.05 dB better than that of the non-real-time, state-of-the-art SGS-OF algorithm, whereas the MS-SSIM is 0.9850 slightly lower than SGS-OF at 0.9903. Enhancing VAL-MCP with non-real time IDA reconstruction, allows VAL-IDA-MCP to exceed the SGS-OF PSNR, i.e. 35.07 dB compared to 34.82 dB. Replacing the MCP block with a GPU-accelerated CNN-based video frame interpolation algorithm, allows the VAL-VFI algorithm to achieve state-of-the-art, real-time performance with a PSNR of 35.12 dB and an SSIM of 0.9921 exceeding the previous, non-real time SGS-OF results of 34.82 dB and 0.9903 respectively. Reconstructing VAL-IDA-VFI with the IDA algorithm increases PSNR and SSIM to 36.46 dB and 0.9925.

The two Video Super Resolution (VSR) algorithms developed in this chapter, VSR-IMDN-MCP and VSR-IMDN-VFI, exceed the PSNR results of the state of the art VSRResNetFeatGAN by 0.48 dB and 1.06 dB respectively. The SSIM of VSR-IMDN-MCP and VSR-IMDN-VFI are also better, 0.8653 and 0.8773, exceeding the VSRResNetFeatGAN SSIM of 0.8571. However, they do not match the state of the art performance performance of VAL-[IDA]-MCP and VAL-[IDA]-VFI.

The VAL algorithms developed in this chapter are very suitable for implementing MPC-based systems that super resolve low resolution video sensors to replace expensive high-resolution sensors, for example thermal video cameras.

End-to end distributed video compressive sensing

Chapter 8 takes the best performing VCS algorithms developed in chapter 7 and quantizes and entropy encodes them using an improved version of the low complexity TAB encoder developed in chapter 6. TAB-PNG replaces the fixed length coders in the original TAB algorithm with a novel codec incorporating the PNG [118] image coding algorithm. The two main algorithms developed in this chapter, VAL-TP-IDA-MCP and VAL-TP-IDA-VFI are compared with the classical DISCOVER DVC codec using a comparison framework developed in chapter 3. Although the energy efficient VAL algorithms use compressive sensing and the low complexity

TAB-PNG codec, the PSNR performance on two of the test sequences come very close to the DISCOVER performance with a GOP of four frames.

Thus, the proposed VAL algorithms hold promise in low-energy end-to-end systems, although there is still a substantial PSNR performance gap to classical DVC systems that use high complexity key frame encoders, such as the H264/AVC intra mode.

9.2 Future work

In general, the state-of-the-art image and video compressive sensing algorithms developed in this dissertation have a considerable number of hyper-parameters that can be tuned to optimise performance. It is possible that performance can be improved by further theoretical analysis or longer simulation runs.

The main algorithms developed in this dissertation contain plug-and-play components, for example image denoisers, motion estimation blocks and video frame interpolation. As better performing algorithms are developed, the algorithms presented in this dissertation can be upgraded to improve their performance.

Image CS

The success of the IDA algorithm when reconstructing partial DCT measurements, merits further study. The IDA algorithm was designed after interpreting the D-AMP algorithm as a feedback system and simplifying the feedback loop. However, the theory underpinning this behaviour needs to be developed further.

The IDA algorithm can improve the quality of JPEG images by extracting the block measurement matrix associated with each image, and then running the IDA algorithm to deblock and improve the PSNR and SSIM. This technique can be developed into a post-processing enhancement technique that can enhance JPEG images to meet the performance of JPEG2000, at least at certain bpp.

The Block Boundary Variation (BBV) adaptivity algorithm is used in conjunction with [A]L-DCT algorithms in this dissertation. However, BBV is also suited for adapting the number of block compressive samples using any measurement matrix since the exploratory measurements are few and can be used in the reconstruction process. Therefore, it is of interest to study BBV in other adaptive CS schemes.

End-to-end image CS

The BBV sparsity estimation algorithm was found to exceed the performance of the DCT Domain (DD) algorithm when the compression ratio was above 0.5. This made it suitable for coding key frames in video CS applications. However, the TAB and TAB-PNG algorithms developed in this dissertation could not efficiently compress the highly variable number of TCs that are collected using the BBV algorithm. It is therefore of interest to develop a low-complexity

coder that can better compress BBV estimated blocks. This would also benefit DD estimated blocks that suffer from the same behaviour, but to a much lower extent.

Accelerating the Video CS algorithms

VAL-MCP achieves a frame rate of around 10 frames per second when executing on Matlab without GPU acceleration, the block motion estimation code included. The VAL-MCP decoder code can be re-written in C/C++ to accelerate it considerably, maybe also exploiting GPU acceleration.

The VAL-MCP and VSR-MCP algorithms can be unrolled and implemented entirely as GPU-accelerated CNNs. This would guarantee real-time operation and also allow the CNNs to be trained to optimise the many hyper-parameters that were manually tuned in this dissertation.

Video SR

The better end-to-end performance of image SR algorithms over image CS algorithms indicated that video SR could potentially challenge video CS algorithms in terms of image quality. However, when the non-key frame [A]L-DCT algorithm was replaced with the Shrink algorithm with GPU-accelerated SR, the resulting VSR algorithms could not match the PSNR and SSIM performance of the VCS algorithms. The major loss in performance was traced to the decoder-side DCT Domain DPCM module, that was designed for the AL-DCT-DD. Further work is required to investigate whether this loss of performance can be offset by designing an SR-matched decoder-side DPCM module.

Sensor energy-efficiency comparison framework

A framework for comparing energy-constrained compressive sensors, employing an SPC, MPC or FPPS, is proposed in this dissertation. It has been applied with some success to compare real-time video transmission algorithms from energy-constrained sensors. However, comparing the computational complexity of the many algorithms that can be implemented on the sensor platforms is a very challenging task. Implementing the algorithms proposed in this dissertation on a commercially available sensor platform, would allow realistic comparisons to be made.

End-to-End distributed video compressive sensing

When comparing VAL with the DISCOVER codec it was noted that the DCT part of the Wyner-Ziv frame encoder can possibly be implemented using compressive DCT measurements on an MPC or FPPS. It would be interesting to study this further, possibly also exploiting the compressive measurements to simplify part of the complex key frame coding algorithm.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer networks*, vol. 51, no. 4, pp. 921–960, 2007.

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

[3] A. 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, 1976.

[4] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, pp. 489–509, Feb 2006.

[5] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, April 2006.

[6] V. K. Goyal, A. K. Fletcher, and S. Rangan, "Compressive sampling and lossy compression," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 48–56, 2008.

[7] Lu Gan, "Block compressed sensing of natural images," in *2007 15th International Conference on Digital Signal Processing*, pp. 403–406, July 2007.

[8] J. E. Fowler, S. Mun, E. W. Tramel, *et al.*, "Block-based compressed sensing of images and video," *Foundations and Trends in Signal Processing*, vol. 4, no. 4, pp. 297–416, 2012.

[9] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[10] J. Zammit and I. J. Wassell, "Adaptive block compressive sensing: Toward a real-time and low-complexity implementation," *IEEE Access*, vol. 8, pp. 120999–121013, 2020.

[11] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[12] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 269–280, 2009.

[13] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[14] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5117–5144, 2016.

[15] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1336–1343, IEEE, 2015.

[16] C. Metzler, A. Mousavi, and R. Baraniuk, "Learned d-amp: Principled neural network based compressive image recovery," in *Advances in Neural Information Processing Systems*, pp. 1772–1783, 2017.

[17] C. Zhao, J. Zhang, R. Wang, and W. Gao, "Cream: Cnn-regularized admm framework for compressive-sensed image reconstruction," *IEEE Access*, vol. 6, pp. 76838–76853, 2018.

[18] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 449–458, 2016.

[19] W. Shi, F. Jiang, S. Zhang, and D. Zhao, "Deep networks for compressed image sensing," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 877–882, IEEE, 2017.

[20] J. Zhang and B. Ghanem, "Ista-net: Interpretable optimization-inspired deep network for image compressive sensing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1828–1837, 2018.

[21] W. Shi, F. Jiang, S. Liu, and D. Zhao, "Scalable convolutional neural network for image compressed sensing," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12282–12291, 2019.

[22] S. Mun and J. E. Fowler, "Block compressed sensing of images using directional transforms," in *2010 Data Compression Conference*, pp. 547–547, March 2010.

[23] C. Zhao, S. Ma, J. Zhang, R. Xiong, and W. Gao, "Video compressive sensing reconstruction via reweighted residual sparsity," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 6, pp. 1182–1195, 2016.

[24] J. Chen, Z. Chen, K. Su, Z. Peng, and N. Ling, "Video compressed sensing reconstruction based on structural group sparsity and successive approximation estimation model," *Journal of Visual Communication and Image Representation*, vol. 66, p. 102734, 2020.

[25] W. Shi, S. Liu, F. Jiang, and D. Zhao, "Video compressed sensing using a convolutional neural network," *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[26] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The discover codec: architecture, techniques and evaluation," in *picture coding symposium (PCS'07)*, no. CONF, 2007.

[27] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[28] S. Foucart and H. Rauhut, "A mathematical introduction to compressive sensing," *Bull. Am. Math*, vol. 54, pp. 151–165, 2017.

[29] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 83–91, March 2008.

[30] A. L. Pilastri and J. M. R. Tavares, "Reconstruction algorithms in compressive sensing: an overview," in *11th edition of the Doctoral Symposium in Informatics Engineering (DSIE-16)*, 2016.

[31] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[32] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[33] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[34] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, pp. 504–507, July 2006.

[35] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 399–406, 2010.

[36] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3d?," pp. 2392–2399, IEEE, June 2012.

[37] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, (USA), pp. 341–349, Curran Associates Inc., 2012.

[38] A. Majumdar, "Real-time dynamic mri reconstruction using stacked denoising autoencoder," *arXiv preprint arXiv:1503.06383*, 2015.

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, Dec. 2015.

[40] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep Fully-Connected Networks for Video Compressive Sensing," *arXiv:1603.04930 [cs]*, Mar. 2016. arXiv: 1603.04930.

[41] Dengyu Liu, Jinwei Gu, Y. Hitomi, M. Gupta, T. Mitsunaga, and S. K. Nayar, "Efficient Space-Time Sampling with Pixel-Wise Coded Exposure for High-Speed Imaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 248–260, Feb. 2014.

[42] A. Adler, D. Boublil, M. Elad, and M. Zibulevsky, "A Deep Learning Approach to Block-based Compressed Sensing of Images," *arXiv:1606.01519 [cs]*, June 2016. arXiv: 1606.01519.

[43] M. Borgerding and P. Schniter, "Onsager-Corrected Deep Networks for Sparse Linear Inverse Problems," *arXiv preprint arXiv:1612.01183*, 2016.

[44] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 416–423, IEEE, 2001.

[45] A. Mousavi and R. G. Baraniuk, "Learning to Invert: Signal Recovery via Deep Convolutional Networks," *arXiv:1701.03891 [cs, math, stat]*, Jan. 2017. arXiv: 1701.03891.

[46] H. Yao, F. Dai, S. Zhang, Y. Zhang, Q. Tian, and C. Xu, "Dr2-net: Deep residual reconstruction network for image compressive sensing," *Neurocomputing*, vol. 359, pp. 483–493, 2019.

[47] R. Puri and K. Ramchandran, "Prism: A new robust video coding architecture based on distributed compression principles," in *Proceedings of the annual allerton conference on communication control and computing*, vol. 40, pp. 586–595, Citeseer, 2002.

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

[49] R. Puri, A. Majumdar, P. Ishwar, and K. Ramchandran, "Distributed video coding in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 94–106, 2006.

[50] C. Chen, E. W. Tramel, and J. E. Fowler, "Compressed-sensing recovery of images and video using multihypothesis predictions," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pp. 1193–1198, Nov 2011.

[51] T. T. Do, Y. Chen, D. T. Nguyen, N. Nguyen, L. Gan, and T. D. Tran, "Distributed compressed video sensing," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 1393–1396, IEEE, 2009.

[52] I. Rec, "H.265 and ISO/IEC 23008-2: High efficiency video coding (HEVC)," 2013.

[53] R. Zhang, S. Wu, Y. Wang, and J. Jiao, "High-performance distributed compressive video sensing: Jointly exploiting the hevc motion estimation and the l1–l1 reconstruction," *IEEE Access*, vol. 8, pp. 31306–31316, 2020.

[54] I. IEC, "Information technology-digital compression and coding of continuous-tone still images: Requirements and guidelines," *Standard, ISO IEC*, pp. 10918–1, 1994.

[55] I. S. ISO, I. JTC, *et al.*, "Information technology-jpeg 2000 image coding system-part 1: Core coding system," *ISO/IEC 15444-1*, 2001.

[56] T. Richter, J. Keinert, A. Descampe, and G. Rouvroy, "Entropy coding and entropy coding improvements of jpeg xs," in *2018 Data Compression Conference*, pp. 87–96, IEEE, 2018.

[57] D. Takhar, J. N. Laska, M. B. Wakin, M. F. Duarte, D. Baron, S. Sarvotham, K. F. Kelly, and R. G. Baraniuk, "A new compressive imaging camera architecture using optical-domain compression," in *Computational Imaging IV*, vol. 6065, p. 606509, International Society for Optics and Photonics, 2006.

[58] A. Zarandy, R. Domínguez-Castro, and S. Espejo, "Ultra-high frame rate focal plane image sensor and processor," *IEEE Sensors Journal*, vol. 2, no. 6, pp. 559–565, 2002.

[59] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "Matia: a programmable 80/spl mu/w/frame cmos block matrix transform imager architecture," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, 2006.

[60] R. Robucci, J. D. Gray, L. K. Chiu, J. Romberg, and P. Hasler, "Compressive sensing on a CMOS separable-transform image sensor," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1089–1101, 2010.

[61] R. G. Baraniuk, T. Goldstein, A. C. Sankaranarayanan, C. Studer, A. Veeraraghavan, and M. B. Wakin, "Compressive video sensing: algorithms, architectures, and applications," *IEEE Signal Processing Magazine*, vol. 34, no. 1, pp. 52–66, 2017.

[62] J. Wang, M. Gupta, and A. C. Sankaranarayanan, "Lisens-a scalable architecture for video compressive sensing," in *2015 IEEE International Conference on Computational Photography (ICCP)*, pp. 1–9, IEEE, 2015.

[63] T. Instruments, "Dlp9500 dlp(r) 0.95 1080p 2x lvds type a dmd." http://www.ti.com, 2012 revised 2018, accessed June 2017.

[64] X. Zhang, A. Wang, B. Zeng, L. Liu, and Z. Liu, "Adaptive block-wise compressive image sensing based on visual perception," *IEICE TRANSACTIONS on Information and Systems*, vol. 96, no. 2, pp. 383–386, 2013.

[65] S. Zhu, B. Zeng, and M. Gabbouj, "Adaptive sampling for compressed sensing based image compression," *Journal of Visual Communication and Image Representation*, vol. 30, pp. 94–105, 2015.

[66] A. Akbari, D. Mandache, M. Trocan, and B. Granado, "Adaptive saliency-based compressive sensing image reconstruction," in *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–6, July 2016.

[67] 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, 2002.*, vol. 1, pp. 240–244, IEEE, 2002.

[68] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 114–125, 2017.

[69] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, April 2004.

[70] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402, Ieee, 2003.

[71] J. Romberg, "Imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.

[72] J. E. Fowler, S. Mun, and E. W. Tramel, "Block-based compressed sensing of images and video," *Foundations and Trends in Signal Processing*, vol. 4, no. 4, pp. 297–416, 2012.

[73] E. W. Tramel and J. E. Fowler, "Video compressed sensing with multihypothesis," in *2011 Data Compression Conference*, pp. 193–202, IEEE, 2011.

[74] X. Yuan and R. Haimi-Cohen, "Image compression based on compressive sensing: End-to-end comparison with jpeg," *arXiv preprint arXiv:1706.01000*, 2017.

[75] A. Wang, L. Liu, B. Zeng, and H. Bai, "Progressive image coding based on an adaptive block compressed sensing," *IEICE Electronics Express*, vol. 8, no. 8, pp. 575–581, 2011.

[76] W. Guicquero, A. Dupret, and P. Vandergheynst, "An adaptive compressive sensing with side information," in *2013 Asilomar Conference on Signals, Systems and Computers*, pp. 138–142, Nov 2013.

[77] Z. Gao, C. Xiong, L. Ding, and C. Zhou, "Image representation using block compressive sensing for compression applications," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 885–894, 2013.

[78] H. Zheng and X. Zhu, "Sampling adaptive block compressed sensing reconstruction algorithm for images based on edge detection," *The Journal of China Universities of Posts and Telecommunications*, vol. 20, no. 3, pp. 97–103, 2013.

[79] J. Luo, Q. Huang, S. Chang, and H. Wang, "Fast reconstruction with adaptive sampling in block compressed imaging," *IEICE Electronics Express*, vol. 11, no. 6, pp. 20140056–20140056, 2014.

[80] T. Nguyen Canh, "Adaptive block compressive sensing: Toward a complete approach," *10.13140/RG.2.1.1807.4728*, 2015.

[81] W. Wang, W. Yang, and J. Li, "An adaptive sampling method of compressed sensing based on texture feature," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 2, pp. 648–654, 2016.

[82] J. Wu, Y. Wang, K. Zhu, and Y. Zhu, "Perceptual sparse representation for compressed sensing of image," in *2016 Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2016.

[83] A. Averbuch, S. Dekel, and S. Deutsch, "Adaptive compressed image sensing using dictionaries," *SIAM Journal on Imaging Sciences*, vol. 5, no. 1, pp. 57–89, 2012.

[84] Nguyen et al, "Rate allocation method for block-based compression sensing," *Journal of Broadcasting Engineering*, vol. 20, pp. 398–407, 05 2015.

[85] J. Liu and C. Ling, "Adaptive compressed sensing using intra-scale variable density sampling," *IEEE Sensors Journal*, vol. 18, pp. 547–558, Jan 2018.

[86] B. Shi, Q. Lian, and S. Chen, "Compressed sensing magnetic resonance imaging based on dictionary updating and block-matching and three-dimensional filtering regularisation," *IET Image Processing*, vol. 10, no. 1, pp. 68–79, 2016.

[87] B. Shi, Q. Lian, X. Huang, and N. An, "Constrained phase retrieval: when alternating projection meets regularization," *JOSA B*, vol. 35, no. 6, pp. 1271–1281, 2018.

[88] B. Shi, Q. Lian, and H. Chang, "Deep prior-based sparse representation model for diffraction imaging: A plug-and-play method," *Signal Processing*, vol. 168, p. 107350, 2020.

[89] X. Yuan and R. Haimi-Cohen, "Image compression based on compressive sensing: End-to-end comparison with jpeg," *arXiv preprint arXiv:1706.01000*, 2017.

[90] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[91] "Data: all reconstructed images and results," available at: http://dx.doi.org/10.21227/rf85-v397, 2020.

[92] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[93] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[94] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[95] S. Zhou, Y. He, Y. Liu, and C. Li, "Multi-channel deep networks for block-based image compressive sensing," *arXiv preprint arXiv:1908.11221*, 2019.

[96] "D-amp toolbox: This package contains the code to run the bm3d, bm3dsapca, bls-gsm, and nlm variants of the d-amp, d-vamp, and d-prgamp algorithms," Apr. 2017. original-date: 2017-02-09, available at https://github.com/ricedsp/D-AMP_Toolbox.

[97] Z. Krusevac and Z. Bojkovic, "Bit allocation and coding gain in mpeg video compression standard for optimum picture ordering," in *Fifth Asia-Pacific Conference on ... and Fourth Optoelectronics and Communications Conference on Communications,*, vol. 2, pp. 927–930 vol.2, 1999.

[98] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[99] A. S. Glassner, *Graphics gems*. Elsevier, 2013.

[100] B. Zeng and A. N. Venetsanopoulos, "A jpeg-based interpolative image coding scheme," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 393–396, IEEE, 1993.

[101] A. M. Bruckstein, M. Elad, and R. Kimmel, "Down-scaling for better transform compression," *IEEE Transactions on Image Processing*, vol. 12, no. 9, pp. 1132–1144, 2003.

[102] V.-A. Nguyen, Y.-P. Tan, and W. Lin, "Adaptive downsampling/upsampling for better video compression at low bit rate," in *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1624–1627, IEEE, 2008.

[103] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: A benchmark," in *European Conference on Computer Vision*, pp. 372–386, Springer, 2014.

[104] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*, pp. 184–199, Springer, 2014.

[105] S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey," *arXiv preprint arXiv:1904.07523*, 2019.

[106] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1646–1654, 2016.

[107] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*, pp. 711–730, Springer, 2010.

[108] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 723–731, 2018.

[109] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight image super-resolution with information multi-distillation network," in *Proceedings of the 27th ACM International Conference on Multimedia (ACM MM)*, pp. 2024–2032, 2019.

[110] R. Timofte, S. Gu, J. Wu, and L. Van Gool, "Ntire 2018 challenge on single image super-resolution: Methods and results," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 852–863, 2018.

[111] S. Nah, R. Timofte, S. Gu, S. Baik, S. Hong, G. Moon, S. Son, and K. Mu Lee, "Ntire 2019 challenge on video super-resolution: Methods and results," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[112] A. Lugmayr, M. Danelljan, and R. Timofte, "Ntire 2020 challenge on real-world image super-resolution: Methods and results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 494–495, 2020.

[113] "Source code: Pytorch," downloaded from: https://pytorch.org/.

[114] "Source code: Matconvnet – convolutional neural networks for matlab," downloaded from: https://www.vlfeat.org/matconvnet/.

[115] "Source code: Pytorch," downloaded from: https://www.tensorflow.org/.

[116] S. Mun and J. E. Fowler, "Dpcm for quantized block-based compressed sensing of images," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 1424–1428, IEEE, 2012.

[117] X. Yuan and R. Haimi-Cohen, "Image compression based on compressive sensing: End-to-end comparison with jpeg," *IEEE Transactions on Multimedia*, 2020.

[118] T. Boutell, "Rfc2083: Png (portable network graphics) specification version 1.0," 1997.

[119] D. Odom and P. Milanfar, "Modeling multiscale differential pixel statistics," in *Computational Imaging IV*, vol. 6065, p. 606504, International Society for Optics and Photonics, 2006.

[120] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.

[121] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.

[122] R. Franzen, "Kodak lossless true color image suite," *source: http://r0k. us/graphics/kodak*, 1999.

[123] F. Bellard, "Bpg specification." https://bellard.org/bpg/bpg_spec.txt, 2017, Accessed April, 21, 2018.

[124] I. Rec, "H. 262 and iso/iec 13818-2 (mpeg-2 video), generic coding of moving pictures and associated audio information part 2: Video," *ed: ITU, ISO Std., Rev*, 1994.

[125] F. Sjoberg, "Ltu-mediacoding.tgz." http://www.sm.luth.se/csee/courses/sms/047/2004/, 2004, Accessed July, 07, 2018.

[126] Y. Lui, W. Hamidouche, and O. Dforges, "Intra coding performance comparison of hevc, h. 264/avc, motion-jpeg2000 and jpegxr encoders," 2019.

[127] F. Bellard, "Bpg." https://bellard.org/bpg/libbpg-0.9.8.tar.gz, 2014, Accessed April, 21, 2018.

[128] C.-Y. Wang, S.-M. Lee, and L.-W. Chang, "Designing jpeg quantization tables based on human visual system," *Signal Processing: Image Communication*, vol. 16, no. 5, pp. 501–506, 2001.

[129] H. Zheng, "Imdn." https://github.com/Zheng222/IMDN, 17-Aug-2017, Accessed May. 05, 2019.

[130] J. Kim, "Vdsr." https://cv.snu.ac.kr/research/VDSR/, July, 28, 2016, accessed June, 9, 2018.

[131] B. Lee, "Introduction to ± 12 degree orthogonal digital micromirror devices (dmds)," *Texas Instrument Incorporated*, vol. 3, 2018.

[132] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3703–3712, 2019.

[133] S. Mun and J. E. Fowler, "Residual reconstruction for block-based compressed sensing of video," in *2011 Data Compression Conference*, pp. 183–192, IEEE, 2011.

[134] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Transactions on image processing*, vol. 11, no. 12, pp. 1442–1449, 2002.

[135] C. A. Metzler, "D-amp toolbox." https://github.com/ricedsp/D-AMP_Toolbox, 08-Jul-2017, Accessed May. 05, 2019.

[136] C. A. Metzler, "Dncnn." https://github.com/ricedsp/D-AMP_Toolbox/tree/master/ Packages/DnCNN, 08-Jul-2017, Accessed May. 05, 2019.

[137] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," in *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[138] A. Vedaldi and K. Lenc, "Matconvnet." https://github.com/ricedsp/D-AMP_Toolbox, 17-Aug-2017, Accessed May. 05, 2019.

[139] J. Fowler, "Mc-bcs-spl version 1.1-1." http://www.ece.msstate.edu/ fowler/BCSSPL, 31-Jan-2017, Accessed May. 05, 2019.

[140] O. Batra, "Msssim." https://github.com/obartra/ssim/tree/master/assets/ms-ssim, 5-Oct-2016, Accessed May. 05, 2019.

[141] H. Liu, Z. Ruan, P. Zhao, F. Shang, L. Yang, and Y. Liu, "Video super resolution based on deep learning: A comprehensive survey," *arXiv preprint arXiv:2007.12928*, 2020.

[142] S. Y. Kim, J. Lim, T. Na, and M. Kim, "3dsrnet: Video super-resolution using 3d convolutional neural networks," *arXiv preprint arXiv:1812.09079*, 2018.

[143] A. Lucas, S. Lopez-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3312–3327, 2019.

[144] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[145] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4778–4787, 2017.

[146] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, "Detail-revealing deep video super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4472–4480, 2017.

[147] H. Series, "Audiovisual and multimedia systems; infrastructure of audiovisual services—coding of moving video. H.264. advanced video coding for generic audiovisual services," *International Telecommunication Union. Version*, vol. 12, 2007.

[148] S.S.Pradhan and K. Ramchandran, "Distributed sourcecoding using syndromes (discus): Design and construction," in *2011 Data Compression Conference, DCC '99*, pp. 158–167, 1999.

[149] N. Imran, B.-C. Seet, and A. Fong, "Distributed video coding for wireless video sensor networks: a review of the state-of-the-art architectures," *SpringerPlus*, vol. 4, no. 1, p. 513, 2015.

[150] W. B. Rabiner and A. P. 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.

[151] A. Aaron, S. D. Rane, E. Setton, and B. Girod, "Transform-domain wyner-ziv codec for video," in *Visual Communications and Image Processing 2004*, vol. 5308, pp. 520–528, International Society for Optics and Photonics, 2004.

[152] D. Varodayan, A. Aaron, and B. Girod, "Rate-adaptive codes for distributed source coding," *Signal processing*, vol. 86, no. 11, pp. 3123–3130, 2006.

[153] F. Pereira, L. Torres, C. Guillemot, T. Ebrahimi, R. Leonardi, and S. Klomp, "Distributed video coding: selecting the most promising application scenarios," *Signal Processing: Image Communication*, vol. 23, no. 5, pp. 339–352, 2008.

[154] X. Artigas, "The discover codec evaluation," in *http://www.img.lx.it.pt/∼discover/home.html, last accessed 2020-09-05*, 2007.

[155] Y. Baig, E. M. Lai, and A. Punchihewa, "Distributed video coding based on compressed sensing," in *2012 IEEE International Conference on Multimedia and Expo Workshops*, pp. 325–330, IEEE, 2012.

[156] M. A. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of selected topics in signal processing*, vol. 1, no. 4, pp. 586–597, 2007.

[157] L. Gan, T. T. Do, and T. D. Tran, "Fast compressive imaging using scrambled block hadamard ensemble," in *2008 16th European Signal Processing Conference*, pp. 1–5, IEEE, 2008.

[158] X. Artigas, "The discover codec evaluation," in *www.img.lx.it.pt/∼discover/enc_time_complexity_gop248.html, last accessed 2020-09-05*, 2007.