LOW BIT-RATE COMPRESSION OF VIDEO AND LIGHT-FIELD DATA USING CODED SNAPSHOTS AND LEARNED DICTIONARIES

Chandrajit Choudhury[††], *Yellamraju Tarun*[††*], *Ajit Rajwade*[†], *and Subhasis Chaudhuri*[††]

[††]Dept. of Electrical Engineering and [†]Dept. of Computer Science & Engineering, IIT Bombay, Mumbai, India
[††]{chandrajit,ytarun,sc}@ee.iitb.ac.in,[†]ajitvr@cse.iitb.ac.in

ABSTRACT

The method of coded snapshots has been proposed recently for compressive acquisition of video data to overcome the space-time trade-of inherent in video acquisition. The method involves modulation of the light entering the video camera at different time instants during the exposure period by means of a different and randomly generated code pattern at each of those time instants, followed by integration across time, leading to a single coded snapshot image. Given this image and knowledge of the random codes, it is possible to reconstruct the underlying video frames - by means of sparse coding on a suitably learned dictionary. In this paper, we apply a modified version of this idea, proposed formerly in the compressive sensing literature, to the task of compression of videos and light-field data. At low bit rates, we demonstrate markedly better reconstruction fidelity for the same storage costs, in comparison to JPEG2000 and MPEG-4 (H.264) on light-field and video data respectively. Our technique can cope with overlapping blocks of image data, thereby leading to suppression of block artifacts.

1. INTRODUCTION

Image/video compression of videos is a mature field, with standards such as JPEG, JPEG2000 and MPEG-4 which exploit the fact that small blocks from images/video have a sparse/compressible representation in an orthonormal basis, *eg*, the discrete cosine transform (DCT) or wavelet transform. These bases are not data-specific. Recent advances in dictionary learning and sparse representation allow inference of data-specific dictionaries. PCA outputs a data-specific orthonormal basis for the most compact representation of a set of data-points. Techniques such as vector quantization along with geometric feature extraction have shown promising results for compression of some image classes (*eg* faces) at low-bit rates [1]. However, an overcomplete dictionary (where the number of vectors exceeds the dimensionality of each vector) [2] is typically known to allow for much sparser representation. This property has seen many applications in image processing [3], but applications to data compression have been much fewer [2], [4].

Compressive sensing is a paradigm for *acquisition* of data directly in a compressed format, as against the conventional concept of complete acquisition and subsequent compression. Conversion of the compressive measurements to the conventional formats requires solving an under-determined system of equations. Compressive sensing theory states that this system is in fact well-posed under the conditions that (1) the underlying data have a sparse/compressible representation in some orthonormal basis, and (2) the forward model of the measuring device is incoherent (in a precise mathematical sense) with the chosen orthonormal basis [5]. The basic theory has also been extended to overcomplete dictionaries [6]. Examples of actual compressive cameras include the Rice Single Pixel Camera [7], and video cameras using coded snapshot images [8].

Here, we apply ideas inspired from [8] to compression of video and light-field data. During encoding, we modulate each image from a group of images (some T consecutive time-frames for video, or images with T consecutive shifts of the camera for light-field data) with a known but randomly generated code, followed by an addition operation to generate a single *coded snapshot image* as shown in Figure 1. The decoder performs the task of converting these coded snapshots back to regular video frames or light-field images, given the codes used for encoding and an overcomplete dictionary which can sparsely represent small blocks of data. This dictionary is learned *offline* from a representative training set. A major factor that decides the compression rate is T. Since most videos contain more than T images, the encoded data essentially consists of a sequence of coded snapshot images which can be MPEG-4 encoded. In the case of light-field data, each coded snapshot can be encoded using JPEG2000. Thus, our method is *specifically designed* to combine coded snapshots with existing compression standards, but in a way that avoids blocking artifacts during reconstruction (as will be discussed in Section 3).

In this paper, Section 2 describes the video compressive sensing architecture from [8]. Section 3 describes our encoder and decoder for video/light-field data. Section 4 presents experimental results, followed by a discussion in Section 5.

*Chandrajit Choudhury and Yellamraju Tarun are authors with equal contribution.

2. VIDEO COMPRESSIVE SENSING USING CODED SNAPSHOTS

Consider the video $I(x, y, t)$ as a space-time volume of size $N_1 \times N_2 \times N_T$ with N_T frames each of size $N_1 \times N_2$. A conventional video camera separately acquires each frame from this space-time volume. However, the camera from [8] seeks a T-fold gain in temporal resolution, in the following manner. Let $C(x, y, t)$ be a function that modulates each pixel (x, y) at time t. Consider a captured snapshot image of the form

$$I_s(x, y) = \sum_{t=1}^{T} I(x, y, t) \cdot C(x, y, t). \quad (1)$$

For a conventional camera, we have $\forall(x, y, t), C(x, y, t) = 1$, however in [8], we have $\forall(x, y, t), C(x, y, t) \in \{0, 1\}$. We now need to estimate the T frames of I given C and I_s. As the system is under-determined, one imposes the constraint that small blocks from I are sparse in a known dictionary. We use an overcomplete dictionary for sparser representation. Consider a $p \times p \times T$ block from I reshaped to produce a $p^2T \times 1$ vector $\mathbf{q}$. We express $\mathbf{q}$ as $\mathbf{q} = \mathbf{D}\theta = \sum_{k=1}^{K} \mathbf{D}_k\theta_k$, where $\mathbf{D}$ is a dictionary matrix of size $p^2T \times K$ with $\mathbf{D}_k$ being the k^{th} column, and θ is a vector of K dictionary coefficients, which we assume to be sparse, i.e. $\|\theta\|_0 \ll K$. Now, any $p \times p$ patch from the snapshot image I_s (reshaped to form a $p^2 \times 1$ vector $\mathbf{q_s}$) can be expressed as $\mathbf{q_s} = \mathbf{C_s}\mathbf{q} = \mathbf{C_s}\mathbf{D}\theta$ where $\mathbf{C_s}$ is a matrix of size $p^2 \times p^2T$ representing modulation by $C(x, y, t)$ for suitable (x, y, t). The inversion problem basically requires the estimation of θ from $\mathbf{q_s}$, $\mathbf{D}$ and $\mathbf{C_s}$, by solving:

$$\min_\theta \|\theta\|_0 \text{ such that } \|\mathbf{q_s} - \mathbf{C_s}\mathbf{D}\theta\|_2^2 \leq \epsilon \quad (2)$$

for a small-valued ϵ. This optimization problem is known to be NP-hard and the solution uses approximation algorithms such as orthogonal matching pursuit (OMP) or basis pursuit (BP). The dictionary $\mathbf{D}$ is learned offline on a representative training set of similarly sized blocks, using the well-known dictionary learning technique K-SVD [2]. The reader is referred to [8] for details of the hardware implementation. However, we emphasize here that code C is time-dependent, which is *critically important* for successful reconstruction, given the incoherence conditions required for success of the reconstruction algorithms for a sparsifying dictionary [5].

In terms of actual hardware in [8], the modulation of $I(x, y, t)$ with $C(x, y, t)$ is performed by means of a liquid crystal on silicon (LCoS) device which basically consists of a 2D array of tiny mirrors. These mirrors are randomly turned on (code 1) or off (code 0) with some additional constraints such as the bump length (total 'on' time). The code changes T times during one single exposure period of the video camera, as the LCoS device operates at a rate that is T times the frame rate of the video camera. The modulated image thus produced strikes a CCD array. This occurs T times during a single exposure period, and the CCD array performs an addition of these images to produce the final coded snapshot image I_s. In effect, such a video camera acquires a stream of coded snapshot images, each corresponding to the addition of T consecutive video frames. The randomly generated codes are time-dependent, which is *critically important* for successful recovery of the underlying videos from the snapshot sequence, given the incoherence conditions required for success of the reconstruction algorithms given a sparsifying dictionary [5].

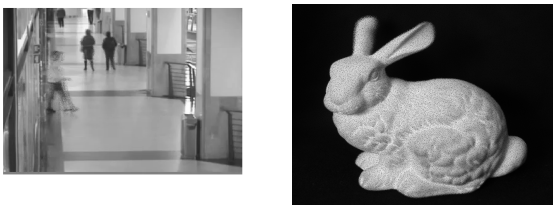

Fig. 1. Sample coded snapshot images for Video (left) and Light-field data (right). *Zoom in the pdf for clearer view of the coded blur artifacts over edges and moving objects.*

3. PROPOSED SCHEME: ENCODER/DECODER

Video: Given T consecutive frames from a video, a coded snapshot image I_s is computed as follows, with a subtle difference from Eqn 1:

$$I_s(x, y) = \sum_{t=1}^{T} I(x, y, t) \cdot C(x, y, t) / \sum_{t=1}^{T} C(x, y, t). \quad (3)$$

The original architecture requires binary codes, but we have used $C(x, y, t) \in [0, 1]$ as ours is a video *compression* scheme. It is clear from Figure 1, that the coded snapshot images carry information about the structural details of the scene and the object motion. Hence, a sequence of coded snapshot images (each corresponding to the addition of T modulated consecutive frames from the original video) can be encoded using MPEG-4 at some chosen quality factor (which affects the bit rate). We shall refer to such a video as a 'MPEG-4-encoded coded snapshot sequence' (MECSS). The MECSS file contains a sequence of intensity-quantized coded snapshot images. From each snapshot image, the decoder needs to recover the original sequence of T video frames. For this, each snapshot image is divided into overlapping blocks, and the corresponding block from the original video is reconstructed by finding θ from Eqn. 2 given $\mathbf{D}$. Since the blocks are overlapping, there will be multiple values produced at most pixel locations. These values are averaged to yield a smoother reconstruction, avoiding block-seam artifacts.

The slightly different method of computing snapshots in Eqn 3 (as compared to Eqn 1) yields significant advantages for videos with moving foregrounds superimposed on static

backgrounds, a common scenario in surveillance videos. It can be seen that for pixel (x, y) located in the static background, we now have $\forall t, I_s(x, y) = I(x, y, t)$, which obviates the need to 'reconstruct' patches lying entirely in the static background. Therefore, the reconstruction algorithm based on inference of θ needs to focus only on patches lying entirely in the foreground or straddling the foreground-background boundary. This leads to *significant speedup* of the reconstruction and also slightly improves reconstruction accuracy. In our experiments, we marked a pixel (x, y) in snapshot I_{s1} as foreground or background based on the following rule:

$$\begin{aligned} |I_{s1}(x, y) - I_{s2}(x, y)| \geq \tau &\rightarrow \text{foreground pixel} \\ |I_{s1}(x, y) - I_{s2}(x, y)| < \tau &\rightarrow \text{background pixel} \end{aligned} \quad (4)$$

where I_{s1} and I_{s2} are two consecutive snapshots in the snapshot sequence and τ is a threshold. While our method has a special advantage for surveillance videos, it must be emphasized though, that it is in no way limited to processing videos with this feature.

Light-field Data: A set of light-field images consists of images acquired by a camera from consecutive, closely spaced viewpoints. While the individual images can be compressed independently, say using JPEG2000, it is desirable to make use of the tremendous inter-image redundancy in such sets of images. This broad idea has been used earlier in [9] (disparity compensation in a 4D wavelet coding framework),[10] (ideas from video compression), [11] (geometrical information about camera viewpoints) and [12] (constructing view-dependent models of a representative image from the entire set). The idea of dictionary learning has not been applied for this task, to the best of our knowledge. Here, we generate a coded snapshot from some T images at consecutive viewpoints, all related to each other by spatial shifts. The coded snapshot image is JPEG2000 encoded at some chosen quality factor. Note that this snapshot is intensity-quantized. From each such snapshot image, the decoder needs to recover the original set of light-field images. For this, each snapshot image is divided into overlapping blocks, and each block is reconstructed by finding θ from Eqn. 2 given $\mathbf{D}$. Since the blocks are overlapping, block-seam artifacts are avoided during reconstruction.

In case of both video as well as light-field compression, we are able to avoid block seam artifacts. This was possible only because we are storing coded snapshot images to form our encoded files, and not dictionary coefficients unlike the methods in [2], [4]. Note that MPEG-4 as well as JPEG2000 divide the image into non-overlapping blocks, however we are *still* free to choose overlapping patches from the coded snapshot images for the purpose of reconstruction. Also, it should be noted that our techniques are *specifically designed* to work in conjunction with MPEG-4 and JPEG2000.

4. EXPERIMENTAL RESULTS

For the **video data**, we performed training on all overlapping spatio-temporal patches of size $6 \times 6 \times T$ from spatially and temporally cropped versions of 10 videos (size 268×384, frame rate 25 fps) from the well-known CAVIAR Dataset[1]. This dataset contains videos of people moving around in departmental stores, i.e. these videos consist of a static background and a relatively sparse moving foreground portion. For the purpose of training alone, the videos were spatially cropped to size 150×200, to focus on the foreground portions. In each video, 75 consecutive frames (total duration = 3 sec per video) containing significant motion were taken as part of the training set. The dictionary $\mathbf{D}$ was learned using KSVD [2], and had a size of $36T \times K$ where $K = 5400$ which is $50 \times 36T$ for $T = 3$. Separate dictionaries were learned for four different values of $T \in \{3, 5, 7, 9\}$. The number of patches on which $\mathbf{D}$ was learned, was between 250,000 for $T = 9$ to 800,000 for $T = 3$. The video compression algorithm was tested on a 3 second portion of 13 different videos without spatial cropping, including 10 videos from the training set and 8 other videos from the same dataset which were not part of the training set. The threshold τ for foreground-background segmentation in Eqn 5 was set to 5 in our experiments (other nearby values of τ produced very similar results).

We compared our technique to MPEG-4 (H.264 coding) on the original (uncoded) videos, by plotting ROC curves of PSNR versus bit rate measured in kbps (the videos available from the database were treated as the reference for measuring PSNR). For MPEG-4 encoding, we used the MATLAB (version 2013a) VideoWriter class with default parameters on a Windows 7 machine. The bit rate of the video was calculated as #storage bits per pixel (bpp) $\times$ #pixels per frame $\times$ video frame rate, where bpp = total file size in bits / (#pixels per frame $\times$ #frames). For any given value of T, we generated sample-points for different bit-rates of the MECSS (MPEG-4-encoded coded snapshot sequence) by varying the quality factor of MPEG-4 from $Q = 10$ to $Q = 100$ in steps of 10. Different Q values produce varied levels of quantization, however we set the value of ϵ in Equation 2 to 0.1 always (this did not adversely affect reconstruction quality). The quality factor of the uncoded MPEG-4 videos was varied from $Q \in \{0, 1, 2, 3, 4, 5 : 5 : 100\}$. The ROC curves are plotted in Figure 2. In many cases, more than one Q value gave similar values of PSNR and bit-rate. As can be observed from the plots, our technique sometimes outperforms MPEG-4 by as much as 2 dB in the low bit-rate regime, i.e. around 100 kbps. We observed this especially in cases where the object motion was significant. The reconstructed videos have a PSNR around 30-34 dB and acceptable visual quality. Moreover, at low bit rates ($Q \leq 5$), MPEG-4 encoded videos exhibit distinct blocking artifacts especially

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

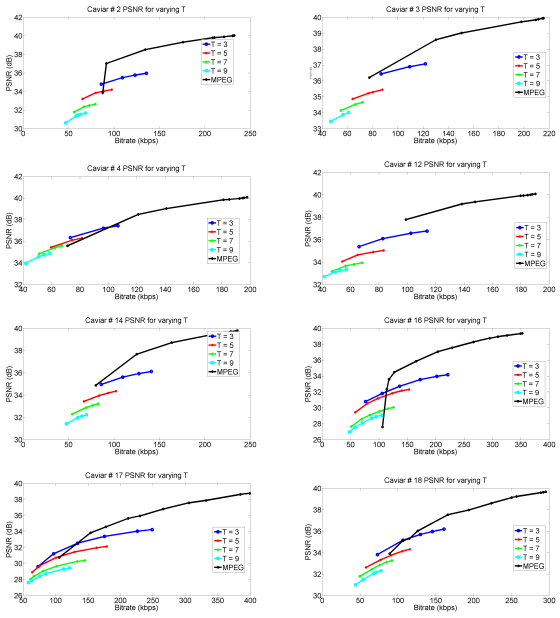

Fig. 2. ROC curves for our compression algorithm for 8 different videos, each for $T \in \{3, 5, 7, 9\}$ in comparison with MPEG-4 using H.264 encoder on a Windows 7 machine. Color codes: MPEG-4, H.264 (black); $T = 3, 5, 7, 9$ is blue, red, green and cyan respectively.

Fig. 3. In each row, leftmost: result of our technique for $T = 7$, middle: original frame, right: video frame when the video was coded with $Q = 1$, *i.e.* same bit rate as MECSS for $T = 7$. *Zoom in pdf file for a better view - notice block-seam aritfacts for MPEG-4 encoding at low bit rates.*

around moving objects. These artifacts are absent in videos reconstructed by our method. In Figure 3, we show a sample reconstruction of four frames from a test video (which was not part of the training set for dictionary learning) using $T = 7$, in comparison with (a) the original frames and (b) MPEG-4 encoded frames at $Q = 1$ (file size 29.3 KB). The file size of the MECSS was 20.3 KB. The reconstructions reveal that the motion of the objects is faithfully reconstructed. The complete set of video reconstructions is available at `http://www.cse.iitb.ac.in/~ajitvr/MMSP2015_Supplemental/Video` (you may also see the file Video.rar inside the folder). In this supplemental material, we present results for each video from the CAVIAR dataset, for each different value of T, as follows: the original video, the reconstruction result for our method and the MPEG encoded result at approximately equal bit rates, as well as the MECSS are all displayed side by side for ready comparison in the form of one large mp4 file. We also tested the same dictionary on two unrelated videos - the Miss America sequence and the Foreman sequence. ROC curves for both sequences and reconstruction results for Foreman are shown in Figures 4. While the reconstructions are satisfactory, we believe that related training and test data would have produced better reconstructions.

For **light-field data**, we performed experiments with four image sets from the Stanford Light-field Archive[2]: lego-truck, bunny, stone, chess. Each image set consisted of 256 images in different poses (in the form of a 16×16 array, based on camera shifts in the X and Y directions). Each image had a size of 996×1296, except for stone whose images had size 1020×756. We created a singe coded snapshot for each value of $T \in \{16, 36, 64\}$ with a random binary code, and learned a separate dictionary for each value of T on non-overlapping patches of size $12 \times 12 \times T$. The training was performed on the lego-truck image set. The number of dictionary columns was set to around 3000, 5000 and 12,500 for $T = 16, T = 36, T = 64$ respectively. For $T = 16$, the training set consisted of images corresponding to the the first 16 horizontal shifts with zero vertical shift. For $T = 36$ and $T = 64$, the training set consisted of images corresponding to a 6×6 and 8×8 subarray of the lowest horizontal and vertical shifts respectively. The compression capabilities of the learned dictionary were tested by reconstructing each of the four sequences from its coded snapshot image, at every different value of T. For testing, the exact same set of horizontal/vertical shifts were selected as those chosen for the training set for that particular T. We compared our compression scheme to JPEG2000 and JPEG, by plotting ROC curves of PSNR versus bpp (average number of bits per pixel). For

[2] `http://lightfield.stanford.edu/`

any given value of T, we generated sample-points for different bpp values by varying the compression ratio of JPEG2000 or JPEG (to store the coded snapshot images).

The ROC curves are plotted in Figure 5 (top row). For the stone dataset, at bpp of 0.005, our technique outperforms JPEG2000 by 2.58 dB for $T = 16$, by 1.8 dB for $T = 36$ and 0.8 dB for $T = 64$. The PSNR of the reconstructed image set is around 27 dB, an acceptable quality level. At bpp > 0.025, we observe that JPEG2000 begins to outperform our method. However, our method outperforms JPEG up to bpp around 0.13. We show a sample reconstruction of 4 frames each from the bunny and stone datasets compared to the original frames, in Figure 5 for $T = 36$ and quality factor of 100 for JPEG-encoding of the coded snapshot. The complete set of reconstructions is available at `http://www.cse.iitb.ac.in/~ajitvr/MMSP2015_Supplemental/Lightfield` (you may see the file Lightfield.rar inside the folder).

More Observations: The reader may have noticed that we have experimented with much larger values of T for the light-field data in comparison to the video data. This is because our video datasets contained fast objects with different motion, whereas the motion between consecutive images of the light-field dataset was much smaller and more 'global'. We have observed that it is difficult to accurately reconstruct sequences with large motion, if the value of T was large. From Figures 2, 4 and 5, one can observe that our technique allows us to reach far lower bit-rates than MPEG-4 or JPEG2000 allows for upto quality factor of 0. This is accomplished by increasing the value of T, albeit at a possible loss of PSNR.

Note that for video as well as light-field data, we are presenting comparative results at *low* bit-rates. This is due to the fact that representations using learned dictionaries are inherently lossy, even more so under sensor noise. Hence at higher bit-rates, our method is outperformed by MPEG/JPEG2000 even though it performs better at lower bit-rates.

5. CONCLUSION

We have presented a compression scheme that combines coded snapshots created by addition of randomly modulated video frames or light-field images, with existing standards such as MPEG-4 or JPEG 2000, and uses the concept of learned dictionaries. Our method outperforms the original schemes at low bit rates, with visually acceptable reconstructions. Our method is especially suitable for compression of large databases of similar or related images/videos, or for surveillance videos where the degree of inter-frame redundancy is high. The dictionary can be learned from a subset of images of the same database, and has negligible storage or transmission costs. Directions for future work include (a) improving the quality of reconstruction at moving object boundaries by taking into account temporal redundancy between successive coded snapshots, (b) increasing the speed of the reconstruction algorithm, and (c) exploring efficient methods to judge optimal T values for creating the coded snapshots based on the content of the underlying data.

6. REFERENCES

[1] M. Elad, R. Goldenberg, and R. Kimmel, "Low bit-rate compression of facial images," *IEEE Trans. on Image Processing*, vol. 16, no. 9, pp. 2379–2383, 2007.

[2] M. Aharon, M. Elad, and A. Bruckstein, "KSVD: An algorithm for designing over-complete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[4] O. Bryt and M. Elad, "Compression of facial images using the k-svd algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–282, 2008.

[5] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, 2008.

[6] E. Candes, Y. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[7] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, 2008.

[8] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. Nayar, "Video from a single coded exposure photograph using a learned over-complete dictionary," in *ICCV*, 2011.

[9] B. Girod, C.-L. Chang, P. Ramanathan, and X. Zhu, "Light-field compression using disparity-compensated lifting," in *ICME*, 2003, pp. 373–376.

[10] M. Magnor and B. Girod, "Data compression for light-field rendering," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, pp. 338–343, 2000.

[11] N. Gehrig and P. L. Dragotti, "Distributed compression of the plenoptic function," in *ICIP*, 2004, pp. 529–532.

[12] Y. Taguchi and T. Naemura, "View-dependent coding of light-fields based on free-viewpoint image synthesis," in *ICIP*, 2006, pp. 509–512.

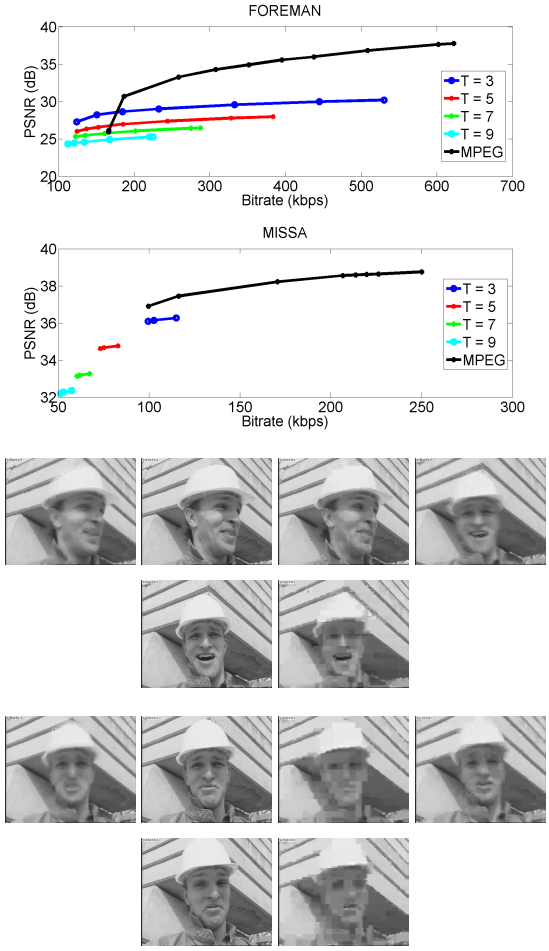

Fig. 4. Top Row: ROC curves for our compression algorithm for $T \in \{3, 5, 7, 9\}$ compared to MPEG-4 using H.264 encoder on a Windows machine for Foreman (left) and Miss America (right) sequences. Color codes: MPEG-4 in black, $T = 3, 5, 7, 9$ in blue, red, green and cyan respectively. Bottom Two Rows: Reconstruction of 4 frames of Foreman sequence at $T = 7$ with $Q = 20$ for MECSS (1st and 4th image), alongside the corresponding frame of the original video (2nd and 5th image) and lower quality MPEG-4 video at the same bit rate as MECSS for $T = 7$ (3rd and 6th image). *Zoom in pdf file for a better view - notice block-seam aritfacts for MPEG-4 encoding at low bit rates.*

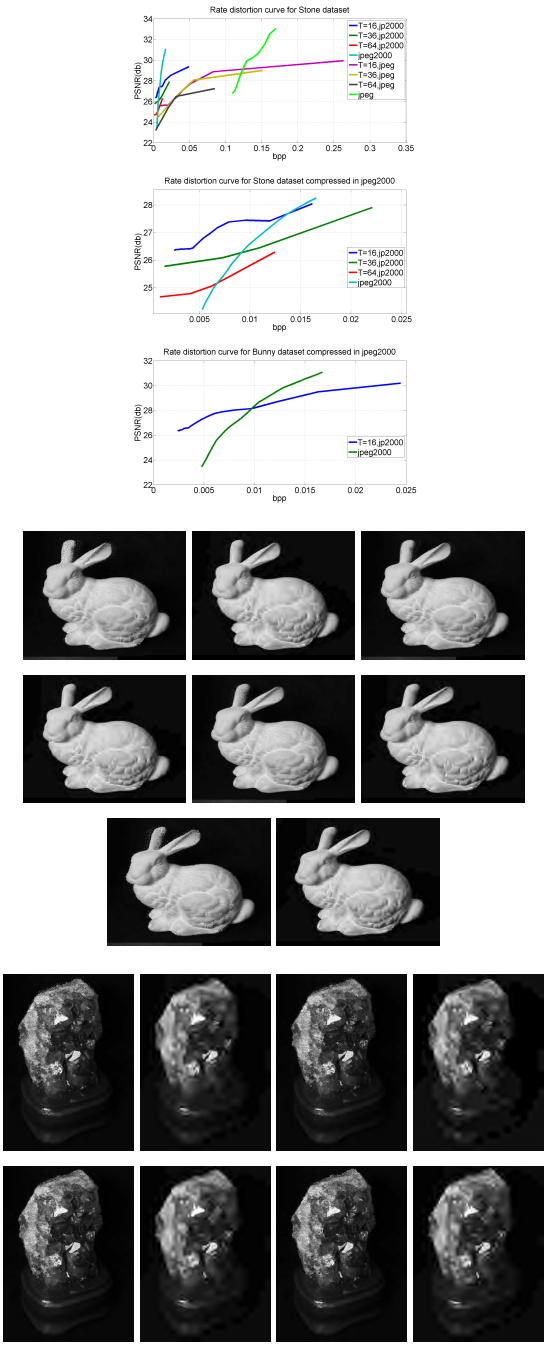

Fig. 5. **Row 1** Left: ROC curves for our compression algorithm, JPEG and JPEG-2000 for the stone data for $T \in \{16, 36, 64\}$. Color codes: JPEG2000 (cyan), JPEG (bright green); $T = 16, 36, 64$ with snapshots coded in JPEG2000: blue, dark green, red respectively; $T = 16, 36, 64$ with snapshots coded in JPEG: purple, yellow, black respectively. Middle: An expanded version of the left image for a smaller bpp range. Right: The ROC curve for the bunny image set comparing our method for $T = 16$ with JPEG2000. **Row 2:** Reconstruction of 4 images from the bunny image set at $T = 16$ alongside the JPEG2000 image at bpp 0.0068 - presented in pairs (images reconstructed with our method on the left followed by JPEG2000 to its right). **Row 3:** for the stone image set with $T = 36$, bpp = 0.0068. *Zoom in pdf for more detail.*