

Using Structure and Texture Filling-in of Missing H.264 Image Blocks in Fading Channel Transmission

Tran Duc Hai Du

First Consulting Group (FCG)
111 West Ocean Blvd, Long Beach
CA, 90802 USA
(904) 635-4957

Abstract - A complexity reduction algorithm for an H.264 encoder is proposed. This comes at the cost of the number of increased macroblock modes and the complex mode decision procedure using the rate-distortion optimization [1]. H.264 demands a high degree of computational complexity to be able to make use of the advantages of its new techniques. Experiment results in [1] show that the proposed algorithm can reduce the encoding time by 29.67% on average and the rate-distortion computation by 89.14% (depending on the source sequence) with no significant loss of rate-distortion performance. In this paper, before transmitting images through any channels, we must compressed image using H.264. However, missing image blocks still occur due to the problem of transmission. An approach for filling-in blocks of missing data in H.264 image transmission is presented. In this paper, H.264 compression is used for lossy JPEG as part of the transmission process, images are first tiled into blocks of 8x8 pixels. When such images are transmitted over fading channels, the effects of noise can kill entire blocks of the image. Instead of using common retransmission query protocols, we aim to reconstruct the lost data using correlation between the lost block and its neighbors. If the lost block contained structure, it is reconstructed using an image in-painting algorithm, while texture synthesis is used for the textured blocks. The switch between the two schemes is done in a fully automatic fashion based on the surrounding available blocks. The performance of this method is tested for various images and combinations of lost blocks. The viability of this method for image compression, in association with lossy JPEG is also discussed.

Keywords: H.264, filling-in, texture synthesis, and in-painting.

1 Introduction

H.264 supports seven macroblock modes. Also, H.264 allows five inter macroblock mode: *SKIP*, *16x16*, *16x8*, *8x16*, and *P8x8*. *P8x8* also allows four sub-inter mode: *8x8*, *8x4* and *4x4* for each 8x8 block (see Figure 1). In this paper, general purpose images are most commonly compressed by lossy JPEG. JPEG divides the image into blocks of 8x8 pixels and calculates a 2-D DCT, followed

by quantization and Huffman encoding, see [2]. In common, the image is transmitted over the channel block-by-block. Due to several fading, we may lose an entire block, even several consecutive block of an image. In [3] the authors report that average packet loss rate in transmission environment is 3.6% and occurs in a bursty fashion. In the worst case, a whole line of image blocks might be lost. Note that JPEG uses differential encoding for storing the average (*dc*) value of successive pixels. Hence, even if a single block is lost, the remaining blocks in that line (or reset interval) might be received without their correct average (*dc*) value. Two common techniques to make the transmission robust are Forward Error Correction (*FEC*) and Automatic Retransmission Query Protocol (*ARQ*). Of these, *FEC* needs extra error correction packets to be transmitted. As noted in [4], *ARQ* lowers data transmission rates and can further increase the network congestion which initially induced the packet loss. Instead, we show that it is possible to satisfactorily reconstruct the lost blocks by using the available information surrounding them. This will result in an increase in bandwidth efficiency of the transmission. The basic idea is to first automatically classify the block as textured or structured, and then fill-in the missing block with information propagated from the surrounding pixels. In the case of structured blocks, the in-painting algorithm in [5] is used, while for textured regions we follow [6].

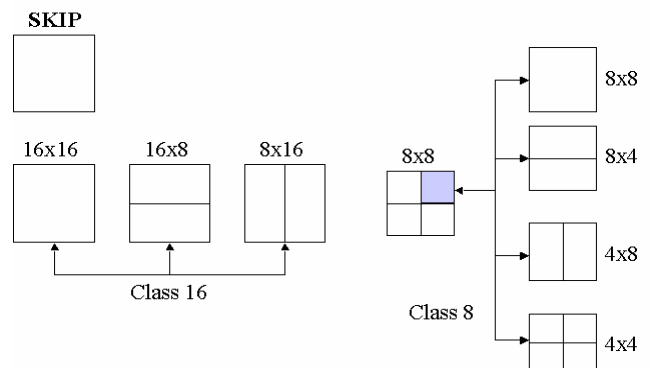


Figure 1. Classes and Inter-modes

The author has test the proposed scheme with a variety of images and simulated block losses and also combine this approach with JPEG compression itself, where the encoder voluntarily skips blocks, and these are reconstructed at the decoder in the same fashion as in the transmission scenario. This process improves the compression ratio, at little or no quality degradation.

2 Related Works

Most schemes reported in the literature deal with image transmission in error-prone environments using a combination of source and channel coding. The author in [1] states that it is very difficult to measure performance with only the encoding time. In the inter macroblock modes, author differentiates *SKIP* an *P8x8* from other macroblock modes with the defined based mode, and then consider the intra macroblock mode conditionally to refine the best inter macroblock mode in term of the rate-distortion characteristics. The authors in [3] describe a packetization scheme in which the DCT coefficients array generated by JPEG is grouped such that bursty (consecutive) packet loss during transmission is scattered into a pseudo-random loss in the image domain. The ensuing reconstruction scheme benefits because, mot frequency components can be recovered from adjacent blocks. However, large bursts may cause the errors to cluster in the image, and reconstruction suffers. It should be noted that the packetization scheme proposed in [3], when used with the reconstruction scheme descript in this paper, is expected to future improve on the results reported here, and provide satisfactory reconstruction results even for very large bursts.

Also, the author in [4] notes that interleaving the image data before packetization avoids loss of contiguous areas in an image, facilitating reconstruction. This paper demonstrates reconstruction in the transform domain by expressing the lost data as a linear combination of blocks in the 4 neighborhood of the lost block. Four optimal weights (coefficients) need to be calculated per block based on combinations of available adjacent blocks. These weights, which result in a 10% space overhead, are used later in reconstruction. Strong diagonal edges are not well reconstructed by this method. Additional work on the reconstruction of missing data in block-based compression schemes is reported in [3], where the DCT coefficients of a missing block are interpolated from those with the same position in the neighboring blocks.

For this paper, the proposed scheme is in the separation of the lost blocks into different classes, followed by the used of state-of-the-art image filling-in algorithms for textured and structured regions. This is done in a complete automatic fashion and without any side information.

3 The Proposed Algorithm

3.1 The Problem: Block-based coding

H.264 blocked-based video coders suffers from annoying blocking artifacts when they are applied in low bit-rate coding because interblock correlation is lost by block-based prediction, transformation, and quantization (see Figure.2). In order to overcome the blocking artifact problem, various nonblock-based coding schemes have been proposed.

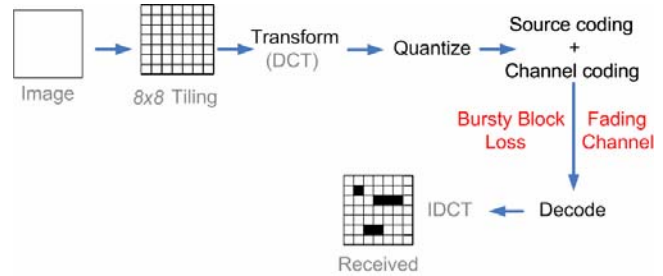


Figure 2. Block-based coding and its problem

Figure 2 shows that images are transmitted over fading channel with the effect of noise can kill entire blocks of image. In order to find an efficient deblocking, some Authors investigate smoothing features in a video sequence in term of the Human Visual System (HVS) [11].

3.2 The Proposed Algorithm

In this paper, the reconstruction of lost blocks follows these three steps:

1. Classify lost blocks into texture and structure
2. Synthesize blocks which were classified as texture (use Texture Synthesis)
3. Fill in blocks which were classified as structure (use Image In-painting)

We now proceed to describe each one of these components.

3.3 Classifying Lost Blocks

The very first step in reconstruction is to classify the lost blocks into texture or structure. This decision is taken at the receiver by querying the region surrounding the lost block. Lost blocks are, of course, excluded from the querying process (see Figure. 3). At the core of this classification is the method proposed in [11]. To determine whether or not, this paper uses a simple coarseness measure given by the number of local extrema in the neighborhood of the lost blocks. The number of local extrema is simply the pixels which are local row extrema as well as local column extrema.

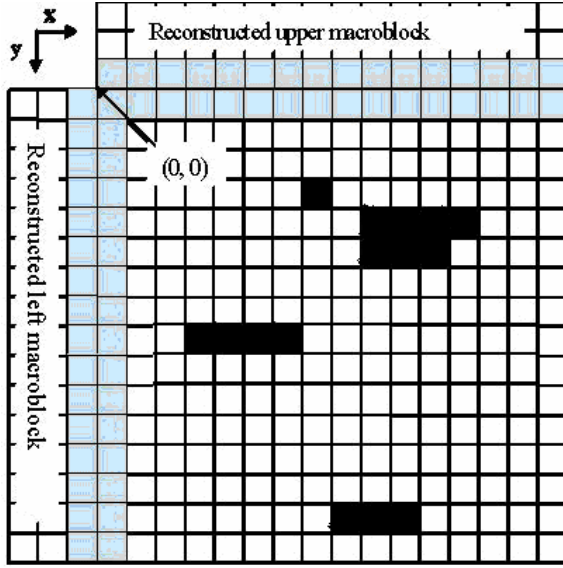


Figure 3. Blocks Classification

Using the method of [13], the number of local extrema in a window of size $s * s$ is given by:

$$n = s^2 \frac{(UB + LB)}{2} \quad (1)$$

Where UB and LB are the “Upper Bounds” and “Lower Bounds” respectively for texture coarseness and are selected by the user. These coarseness values vary from 0 (no extrema) to 1 (all pixels in the selected window are extrema). As suggested from [13], UB and LB are used as follow:

$$UB = 0.16 \text{ and } LB = 0.04$$

In this implementation, $s = 8$, giving $n = 6.4$. Thus, if a 8×8 block has fewer than n extrema, it will be classified to have structure, else it will be consider to contain texture.

The above technique is applied for each available block of 8×8 pixels in the immediate neighborhood of the lost block. Even if a single block from this neighborhood contains structure, we have first considered a decision in favor of structure. However, being reconstruction is our primary goal, these criteria alone might be insufficient, as we illustrate now. Consider for example that we have lost a block containing an edge between two textured regions. The edge between two regions is certainly an expression of structure, and needs to be given precedence over texture even if the block in question has more than the necessary coarseness. The logic behind this will be understood in the next section, wherein, we require the textured region surrounding the block to fill it up. If we were to classify a block containing an edge as texture, we would not be able to reconstruct the edge later, as will become clear after examining the texture synthesis algorithm. To overcome this limitation, we impose an additional constraint as

follows. We consider the 8 neighborhood of a 8×8 block and calculate difference between the average values of the blocks on opposite sides of the center block (considering only available blocks). If the 4 resulting differences are below a threshold, we decide that an edge does indeed pass through the textured block. We then designate the block as structure, notwithstanding its high coarseness. This simple additional constraint has provided a correct classification in all tested images.

3.4 Texture Synthesis

Following [5], we conclude that when a block is classified as having texture, the entire 8 neighborhood of that block has texture. The missing block is then filled-in with the texture from its surrounding. Let the region to be filled be denoted by Ω . The lost block will now be filled, pixel by pixel, in a raster fashion.

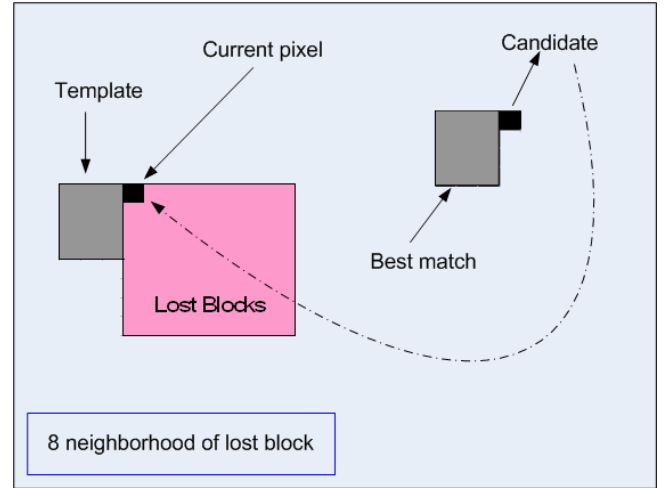


Figure 4. Texture Synthesis

Let I_t be a representative template touching the left of a pixel $p(x, j) \in \Omega$. We proceed to find a I_t^\wedge from the available neighborhood, such that a given distance $d(I_t, I_t^\wedge)$ is minimized. As per [5], d is a normalized Sum of Squared Difference (SSD) metric. Once such a I_t^\wedge is found, we choose the pixel to the immediate right of I_t^\wedge , as our candidate $p(x, j) \in \Omega$. For stochastic textures, the algorithm selects at random one of the pixels neighboring I_t^\wedge .

The template I_t can be a simple seed-block of 3×3 pixels as shown in Figure. 4. Then, of all possible 3×3 blocks in the 8 neighborhood, the one with the minimum normalized SSD is found and a pixel to its right is copied into the current pixel in the lost block, as shown. This algorithm is considerably fast when using the improvements.

3.5 Image Inpainting

The image in-painting technique is a method of repainting a defected area on an image or a procedure of fitting proper image pixels to an area where a particular object has been removed. In the ancients, people retained professional artists to repair the defected artifacts in order to preserve the valuable masterpieces. Due to the widespread use of digital technology currently, the need for the preservation of digital data has increased as well. In digital image in-painting techniques, the author in [11] utilized the Partial Differential Equations (*PDEs*) for image in-painting, but it is only fitted for low-resolution images with light scratches or little areas. The drawbacks of *PDEs* method for replacing larger regions or high-resolution images are lack of consideration for the priorities of the in-painting block sequences and the extension of image textures. Thus, the quality of the image in-painting results will decrease visibly.

Once again let Ω be the region to be filled in (in-painted) and $\delta\Omega$ be its boundary. The basic idea in in-painting is to smoothly propagate the information surrounding Ω in the direction of the isophotes entering $\delta\Omega$. Both gray values and isophotes directions are propagated inside the region. Denoting by I the image, this propagation is achieved numerically solving the partial differential equation (t is an artificial time marching parameter).

$$\frac{\partial I}{\partial t} = \nabla(\Delta I) * \nabla^\perp I \quad (2)$$

Where ∇ , Δ , and ∇^\perp stand for the gradient, Laplacian, and orthogonal-gradient (isophotes direction) respectively. This equation is solved only inside Ω , with proper boundary conditions in $\delta\Omega$ for the gray values and isophotes directions. In the scenario, if

$$\frac{\partial I}{\partial t} = 0 \text{ and } \nabla(\Delta I) * \nabla^\perp I = 0$$

This means that ∇I is constant in the direction $\nabla^\perp I$ of the isophotes, thereby achieving a smooth continuation of the Laplacian inside the region to be in-painted. For details on the numerical implementation of this in-painting technique, which follows the techniques introduced in [15, 16], as well as numerous examples and applications [4].

4 Experiment Results

In this paper, we have assumed, though it is not completely necessary, that the average (*dc*) value of a 8×8 block is known at the receiver. This is not needed by the block reconstruction algorithm, but it is needed for the available surrounding blocks due to the differential

encoding of *dc* values in JPEG. In case the *dc* value is not known, the method proposed in [11] gives a technique to estimate and correct the *dc* value of the lost block and the following blocks in the same line. Retransmission of the *dc* value of a lost block would not be a significant overhead either since we just need 1 byte to be retransmitted per block.

One more problem, we have no control over the fading channel, there is no prior information about the relative locations and number of blocks that can be lost in the process. We present various examples ranging from low to drastic losses of image information, and demonstrate our proposed technique to restore the lost information.

Figure 5, 6, and 7 show the results of reconstruction (from left to right, original, compressed, transmitted, and reconstructed image). Table 1 shows the amount of missing data along with the *PSNR* values after reconstruction.

Table 1. % Data Lost and PSNR

Compressed Image	% Data Lost	PSNR [db]
Batt	6.70%	40.78
Mango	1.90%	43.26
Mixed Fruit	7.20%	37.06

We can see that, if the background color of an image is black (Figure. 6), the percentage of missing data is less than the others. Throughout three experimental tests, we can meet the following result:

1. When single blocks are missing from the image, they are satisfactorily reconstructed from the surrounding context. Note how the reconstructed image is almost identical to the original one, as expected from the algorithms used for filling-in.
2. When a few contiguous blocks are missing, the algorithm still reconstructs the blocks so as to be visually unrecognizable from the original.
3. As a drastic condition, if an entire line is missing, then a good reconstruction is not always possible. This is mainly due to the fact that most probably such a significant loss will cover entire objects. In general, when feature sizes are smaller than 8×8 pixels or are totally covered by the missing line, it will be impossible to reconstruct the image correctly (Figure. 7). In such cases, we will force to request retransmission of the lost block (or an error block). Such instances will become increasingly rare when the image resolution is increased. Further, if the packetization is used, then it is extremely rare to lose an entire line in the image domain. In that case, only independent lost blocks would need to be reconstructed from their neighborhood, and as indicated by Figure. 6 and Figure. 7, the above algorithm restores isolated blocks reliably.

5 Conclusions

5.1 Conclusions

In this paper, we have proposed a new technique for the filling-in of missing blocks in H.264 compressed image transmission of JPEG (or block based). We have shown that as long as the features in the image are not completely lost, they can be satisfactorily reconstructed using a combination of computationally efficient image in-painting and texture synthesis algorithms. This eliminates the need for re-transmission of lost blocks. When the image resolution is increased, the quality of reconstruction improves and a retransmission request is rarely required, resulting in a better effective data transmission rate.

Further, by intentionally (and automatically) dropping image blocks, and using this filling-in approach, we can improve the compression ratio provided by lossy JPEG, without altering the existing JPEG algorithm. As shown in Table. 1, the improvement in compression ration becomes more significant as the image resolution is increased.

5.2 Future Works

This paper is expanding our previous paper [1]. A number of research directions should be taken following the results reported here. We have tried to use image-independent information. For instance, texture and structure are used to enhance the performance of JPEG. The compression ratio can be further increased by finding better masks by providing more image information. In a more general setting, the extension of the approach presented here, to color data needs to be investigated. Since the missing blocks in the different channels need not be in the same image position, information from different channels can be used in the block classification and reconstruction. Adding this to the current neighboring information used is expected to improve even further the quality of the results.

For future works, we found out plenty of ways to reduce the missing data and improve the compression in transmission, which include:

1. For areas of texture, remove as many blocks as possible, since they can be satisfactorily reconstructed from a single seed block [7].
2. Remove a few, but not all blocks along an edge, so that the direction of the edge is properly preserved. Ideally, alternate blocks along the edge should be removed. Presently, a block containing an edge is masked only if the regions on either side of the edge are flat. For instance, the gradient is steep on both sides of the block [9].
3. For blocks with smooth variations, remove alternate blocks. Inpainting always restores these smooth variations.

6 Acknowledgments

This work started when the author was working in US as a Business Analyst for MiTek industries, Ltd. He would like to send his thanks to his family, Phuong To, and Israelsen Martin who always support, encourage and help him to finish this paper.

7 References

- [1] Tran Duc Hai Du, "Macroblock Mode Decision for H.264," *Proceedings of the 7th ACM SIGMM International workshop on Multimedia Information Retrieval* (2005). ISBN: 1-59593-244-5, p. 167 - 172.
- [2] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, p. 30-44, 1991.
- [3] E. Chang, "An image coding and reconstruction scheme for mobile computing", *Proceeding of the 5th IDMS (Springer-Verlag LNCS 1483)*, p.137 – 148.
- [4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," *Computer Graphics (SIGGRAPH 2000)*, p. 417 – 424.
- [5] S. S. Hemami, "Digital image coding for robust multimedia transmission," in *Symposium on Multimedia Communications and Video Coding*, 1995.
- [6] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. Image Processing*, to appear.
- [7] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," *IEEE International Conference on Computer Vision, Corfu, Greece*, p. 1033 – 1038.
- [8] T. Chan and J. Shen, "Mathematical models for local deterministic inpaintings," *UCLA CAM Report*.
- [9] D. Heeger and J. Bergen, "Pyramid based texture analysis/synthesis," *Computer Graphics (SIGGRAPH 1995)*, p. 229 – 238.
- [10] E. Simoncelli and J. Portilla, "Texture characterization via joint statistics of wavelet coefficient magnitudes," *Proc. 5th IEEE Int'l Conf. on Image Processing*.
- [11] A. Marquina and S. Osher, "Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal," *UCLA CAM Report*

- [12] K. Karu, A. K. Jain, and R. M. Bolle, "Is there any texture in the image?," *Pattern Recognition*, vol. 29, no. 9, p. 1437 – 1446.
- [13] L. Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," *Computer Graphics (SIGGRAPH 2000)*.
- [14] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, no. 60, p. 259 – 268.
- [15] S. Shirani, F. Kossentini, and R. Ward, "Reconstruction of baseline JPEG coded images in error prone environments," *IEEE Transactions on Image Processing*, vol. 9, p. 1292-1299.
- [16] Yafan Zhao and Iain E G Richardson, "Computational Complexity Management of Motion Estimation in Video Encoders", *Proc. IEEE DCC2002*.
- [17] R. Braspenning, G.d. Haan, and C. Hentschel, Complexity scalable motion estimation. *Proc. VCIP 2002*, Jan.2002.
- [18] Yafan Zhao and Iain E G Richardson, "Macroblock skip-mode prediction for Complexity control of Video Encoders", *Proceedings of IEE Visual Information Engineering*, July 2003.R.L. de Queiroz, Variable complexity DCT approximations driven by an HVQ-based analyzer. *IEEE Trans. On Circuits and System for video technology*, Nov. 2002. 12(11): p.1021-1024.
- [19] G. Gorla, V. Interrant, and G. Sapiro, "Growing fitted textures," *IMA Technical Report* (www.ima.umn.edu).
- [20] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, G. N. Feng, D. J. Wu and S. Wu, "Fast Mode Decision Algorithm for JVT Intra Prediction", *JVT Doc. G013*, March 2003.
- [21] G. Sullivan and G. Bjontegaard, "Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-scan Source Material", *ITU-T Q.6/16, Doc. VCEG-N81*.
- [22] ITU-T, "Recommendation H.264: Advanced video coding for generic audiovisual services," May 2003.
- [23] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression." *IEEE Signal Processing Magazine*, pp. 23-50,1998.
- [24] T. Chang and J. Shen, "Mathematical models for local deterministic inpainting", *UCLA CAM Report*, March. 2000.
- [25] J. Tumblin and G. Turk, "LCIS: A boundary hierarchy for detail-preserving contrast reduction," *Computer Graphics*, p. 83-90, *SIGGRAPH 99*, 1999.
- [26] E. Simoncelli and J. Portilla, "Texture characterization via joint statistics of wavelet coefficient magnitudes", *Proceeding 5th IEEE Int'l Conf. on Image Processing*, 1998.
- [27] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-C167, "Committee draft number 1, version 0 (CD-1)," *ITU-T Recommendation H.262L*, May 2002.
- [28] S. Masnou and J. Morel, "Level-lines based dissolution," *IEEE Int. Conf. Image Processing*, 2003.



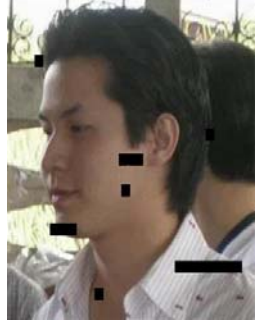

			
Original Image 368400 bytes, 24 BPP (307x400)	Compressed Image by H.264 10403 bytes, 0.68 BPP PSNR 40.78 db	Received Image	Reconstructed Image

Figure 5. Reconstruction results for 1, 2, or 3 of lost blocks





			
Original Image 526350 bytes, 24 BPP	Compressed Image by H.264 15865 bytes, 0.72 BPP PSNR 43.26 db	Received Image	Reconstructed Image

Figure 6. Reconstruction results for image with Black background

			
Original Image 603900 bytes, 24 BPP	Compressed Image By H.264 15589 bytes, 0.62 BPP PSNR 37.06 db	Received Image	Reconstructed Image

Figure 7. Reconstruction results for more complexity image