BIG DATA-DRIVEN FAST REDUCING THE VISUAL BLOCK ARTIFACTS OF DCT COMPRESSED IMAGES FOR URBAN SURVEILLANCE SYSTEMS

Ling Hu and Qiang Ni

School of Computing and Communications, Lancaster University, LA1 4WA, UK

ABSTRACT

The Urban Surveillance Systems generate huge amount of video and image data and impose high pressure onto the recording disks. It is obvious that the research of video is a key point of big data research areas. Since videos are composed of images, the degree and efficiency of image compression are of great importance. Although the DCT based JPEG standard are widely used, it encounters insurmountable problems. For instance, image encoding deficiencies such as block artifacts have to be removed frequently. In this paper, we propose a new, simple but effective method to fast reduce the visual block artifacts of DCT compressed images for urban surveillance systems. The simulation results demonstrate that our proposed method achieves better quality than widely used filters while consuming much less computer CPU resources.

KEYWORDS

Big Data, JPEG, DCT, Image, Urban Surveillance Systems

1. JPEG INTRODUCTION

The widespread use of video cameras in Urban Surveillance Systems has resulted in an enormous amount of image and video data, hence the compression of those image and video data become more and more important in big data research areas. In the mid 80's, the joint collaboration of International Telecommunication Union (ITU) and International organization for Standard (ISO) introduced the standard to compress images which is called the Joint Photographic Experts Group (JPEG) [1]. The core codec method of JPEG is Discrete Cosine Transform (DCT). It may get high compression ratio and it is easy to be implemented, hence the DCT based JPEG is widely used [2]. DCT is a Fourier-related transform expresses, it sum up a sequence of cosine functions at different frequencies.

The two-dimension DCT transform is:

$$F(k,l) = a_k a_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cos\frac{(2m+1)k\pi}{2M} \cos\frac{(2n+1)l\pi}{2N} \quad (1)$$

$$k = 0,1,2 \ldots M-1; \qquad l = 0,1,2 \ldots N-1$$

DOI : 10.5121/ijdkp.2016.6402

$$a_k = \begin{cases} \frac{1}{\sqrt{M}} , & k = 0 \\ \sqrt{\frac{2}{M}} , & 1 \le k \le M-1 \end{cases}; \quad a_l = \begin{cases} \frac{1}{\sqrt{N}} , & l = 0 \\ \sqrt{\frac{2}{N}} , & 1 \le l \le N-1 \end{cases}$$

The two-dimension DCT transform is used in JPEG image compression. And the block diagram of DCT based JPEG transform can be shown below:

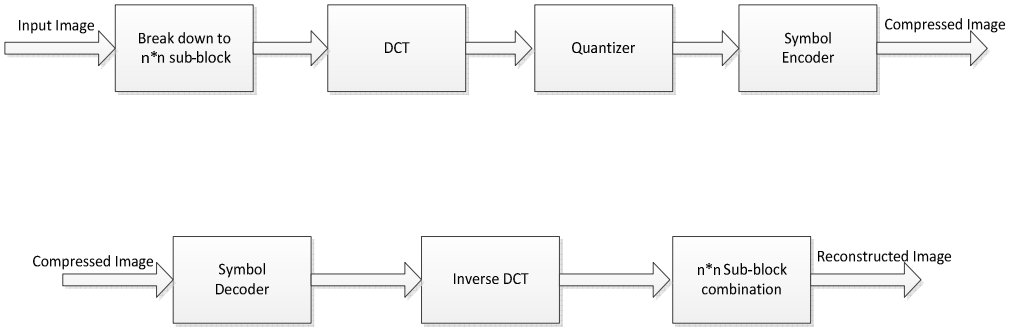

Figure 1: the Block Diagram of DCT based JPEG Transform

In the coding process, we first break down the image to n*n sub-blocks, normally 8*8 sub-blocks are chosen. The two-dimension DCT transform is used to get 64 coefficients for every sub-block. From the two-dimension DCT transform, we can get the 64 coefficients as follows:

From the two-dimension DCT transform, we can get the 64 coefficients as follows:

$$F(u,v) = \sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) a_u \cos\frac{(2x+1)u\pi}{16}\ a_v\ \cos\frac{(2y+1)v\pi}{16} \qquad (2)$$

$$u = 0,1,\dots 7; \qquad v = 0,1,\dots 7$$

$$a_u = \begin{cases} \frac{1}{\sqrt{8}} , & u = 0 \\ \sqrt{\frac{2}{8}} , & u > 0 \end{cases}; \quad a_v = \begin{cases} \frac{1}{\sqrt{8}} , & v = 0 \\ \sqrt{\frac{2}{8}} , & v > 0 \end{cases}$$

Although the application of the DCT formulas would require O(N2) operations, it is possible to compute the same thing with only O(N log N) complexity. This method is known as fast cosine transform (FCT) algorithms. In this way, by paying the cost of more additions, we get speed faster since the CPUs are more excellent to do additions. It makes the DCT become applicable in real image compression processes. Due to good compression ratio, DCT get efficient memory utilization and it is widely used in practice.

In fact, the DCT transform does not compress the images, but the quantization process brings errors into the system. The quantization process is to divide the 64 coefficients by the quantization matrix and round the results into integers. We know this process will incur errors into the systems.

Let us assume the quantization matrix is Q[u,v], If the quantization error are expressed as e[u,v], then the quantitated coefficient matrix are:

$$F_Q[u,v] = \frac{F[u,v]}{Q[u,v]} + e[u,v] \qquad (3)$$

In order to compress the image, some high frequency information is discarded since human eye are not sensitive to high frequency information, but this process also incur errors into the systems.

It is obvious that the errors will distribute inside the whole rebuilt image. Every sub-block bring different errors since they are calculated independently, hence the correlation between different sub-blocks are destroyed. This leads to the block artifacts effect. For the images, higher compression ratio leads to more severe block artifacts. Since the block artifacts effect is coming from the DCT transform, it became the major disadvantage of the JPEG.

In order to show the results of DCT transform, we first simulate the DCT compression effects. We use an 'autumn' image to be our experiment object. Inside our code, we use different bit rates to compress the image.

Figure 2 and Figure 3 are both run with the scheme of 8*8 sub-blocks. The compressed images are more and more blur, and the block artifacts effect shows up more and more severely.

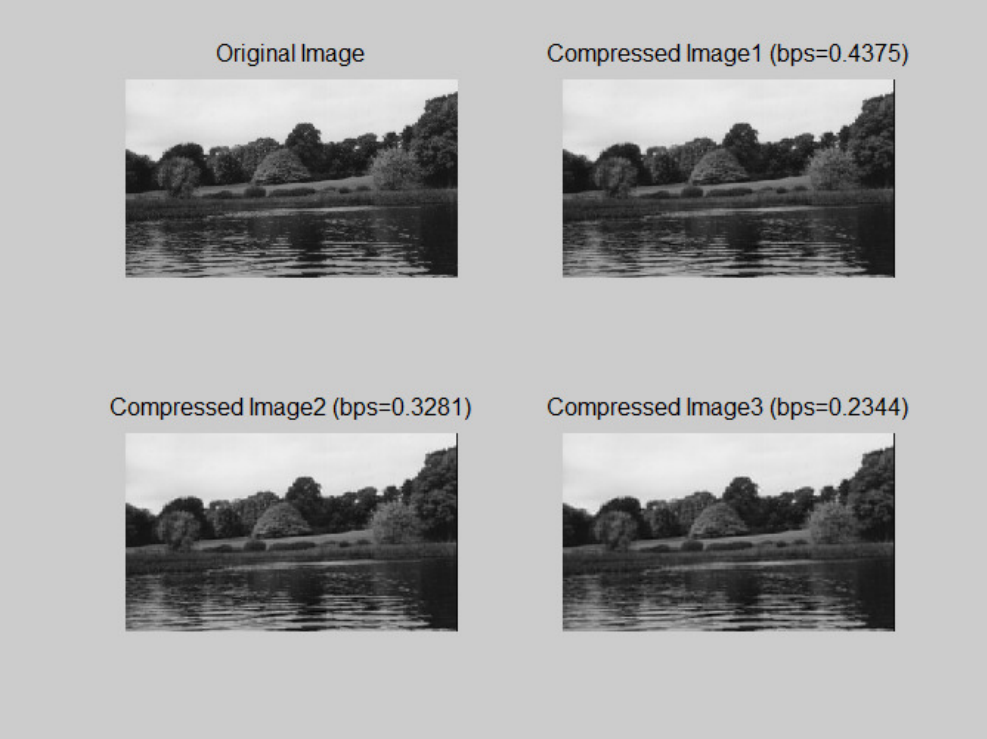

Figure 2. DCT Transform –Autumn 1

Figure 2 shows the original image and some DCT transformed images with light compression. Since the qualities of the compressed images are falling down little by little, the details are shown clearly and the qualities are satisfying for our eyes. The DCT transform appears to be very strong with light compression.

Figure 3. DCT Transform – Autumn 2

Figure 3 shows the DCT transformed images with lower bit rate than that of Figure 2. Inside this Figure, the block artifacts become more and more obvious, and it is not easy to view the details inside the image.

2. BACKGROUND

In order to reduce the block artifacts with minimal change of the original coding system, many researchers have tried different methods. One common method to deal with images is to use filters to filter the received images. Among the filters, median filtered is a kind of non-linear filter and is considered suitable to eliminate the random noise [3]. Median filters are widely used in digital image processing. The idea of the median filter is to run through the signal pixel by pixel, replacing each pixel with the median value of neighbouring pixels. The pattern of neighbours is called the "window". For image signals, the windows are normally chosen as 3*3 pixels boxes, other shapes may be chosen. Another type of widely used filter is adaptive filter [4]. A common used adaptive filter is wiener filter. Wiener filter is a filter used to produce an estimate of a desired or target random process by linear filtering of an observed noisy process. It minimizes the mean square error between the estimated random process and the desired process.

Other methods are used to reduce the block artifacts effect. In [5], the authors proposed a hybrid method which acts both in the frequency and the spatial domains, to enhance the visual result of the reconstructed image and reduce the blocking artifacts. In [6], a semi-local approximation scheme to large-scale Gaussian processed was proposed. This allows learning of task-specific image enhancements from example images without reducing quality. Similar knowledge-based algorithms also include simultaneously removed JPEG blocking artifacts and recovers skin features [7]. On the other hand, some researchers use the property that the original pixel levels in the same block provide continuity and the correlation between the neighbouring blocks to reduce

the discontinuity of the pixels across the boundaries [8]. [9] Proposes to remove artificial edges with a one-time nonlinear smoothing. But the shortcoming for all these methods is that they are very complex, which will consume a lot of computer CPU and memory resources. Note that the Urban Surveillance Systems accumulate huge number of images every second, it is impossible to deal with every image with long time process. It is important to propose very simple but effective methods to solve the block artifacts effect. The best method should be kept at a single step after receiving the already affected image. In this way, all the invested hardware and software may be kept and the cost will be kept low. This motives us to propose the following new simple but effective method.

3. OUR PROPOSED METHOD AND RESULTS

We notice that the block artifacts bring some sharp changes between the vertical and horizontal adjacent blocks. The situation is shown in Figure 4. Hence our idea is to only deal with the sudden changes. In order to keep as much as the original image, our proposed method is to smooth only the connection edges of the vertical and horizontal adjacent of the blocks while keeping other parts of the image unchanged.

Figure 4. Vertical and horizontal adjacent blocks

Using an example of horizontal direction, our proposed method is illustrated as follows:

First we calculate the difference between the two adjacent blocks, which can be written as Δ=abs(x(i)-x(i-1)) , where x(i) and x(i-1) are the grey values at the right and left of the line of the sudden change. Then we divide the Δ by 3. The last step is to adjust the grey values of the right and left of the sudden change line by adding or subtracting the value by $\Delta/3$. In this way, the sudden change is smoothed while keeping most pixels unchanged, hence to alleviate the block artifacts. The flow chart of our proposed method is shown in Figure 5 (shown for horizontal direction only).

We simulate the effect of our proposed method and compare with the two kinds of widely used filters: the median filter and wiener filter. As we already know, at those lightly compressed images, the block artifacts are not obvious, hence we only need to deal with the poor quality received images, which are the images with low bit rate. The results are shown in Figures 6, 7 and 8.

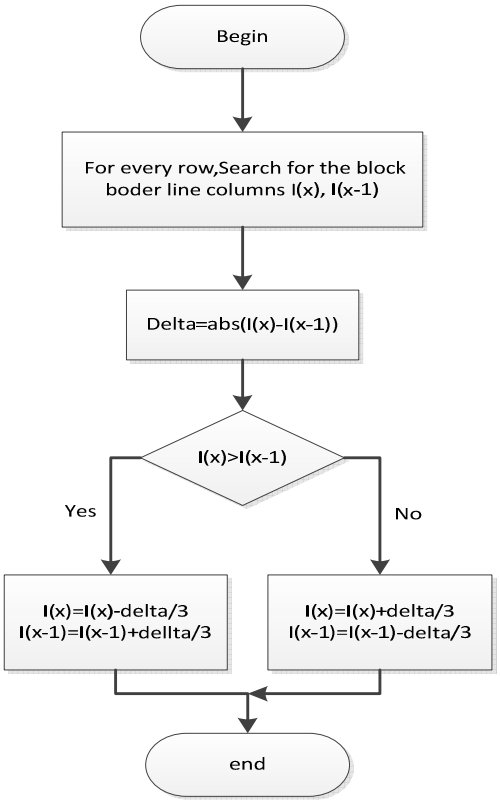

Figure 5. Flow chart of our proposed method (for horizontal line)

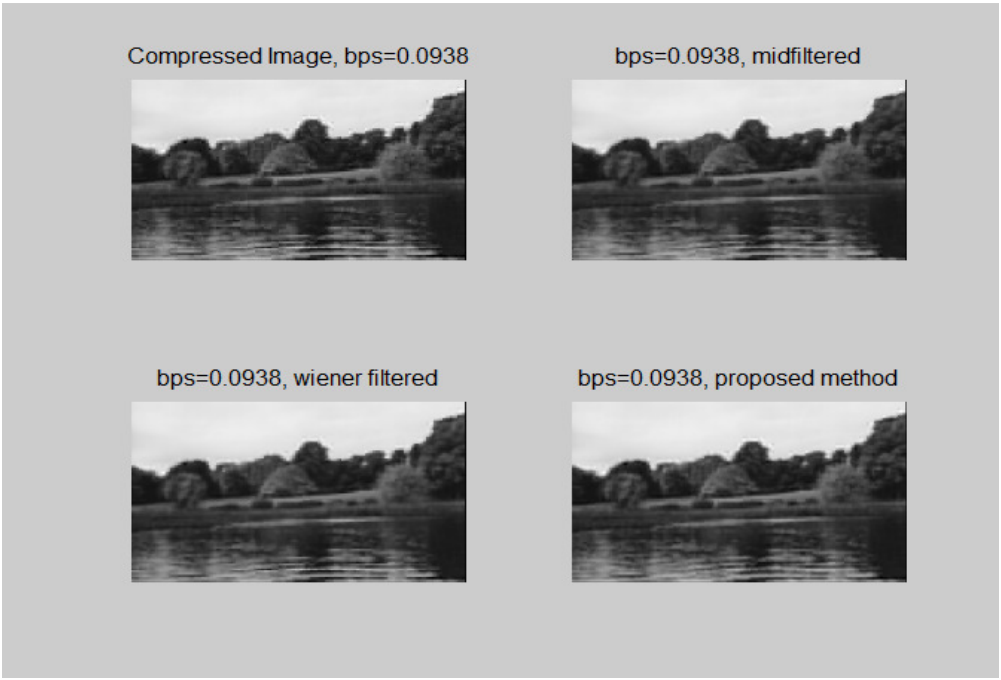

Figure 6. The effects of different methods (bps=0.0938)

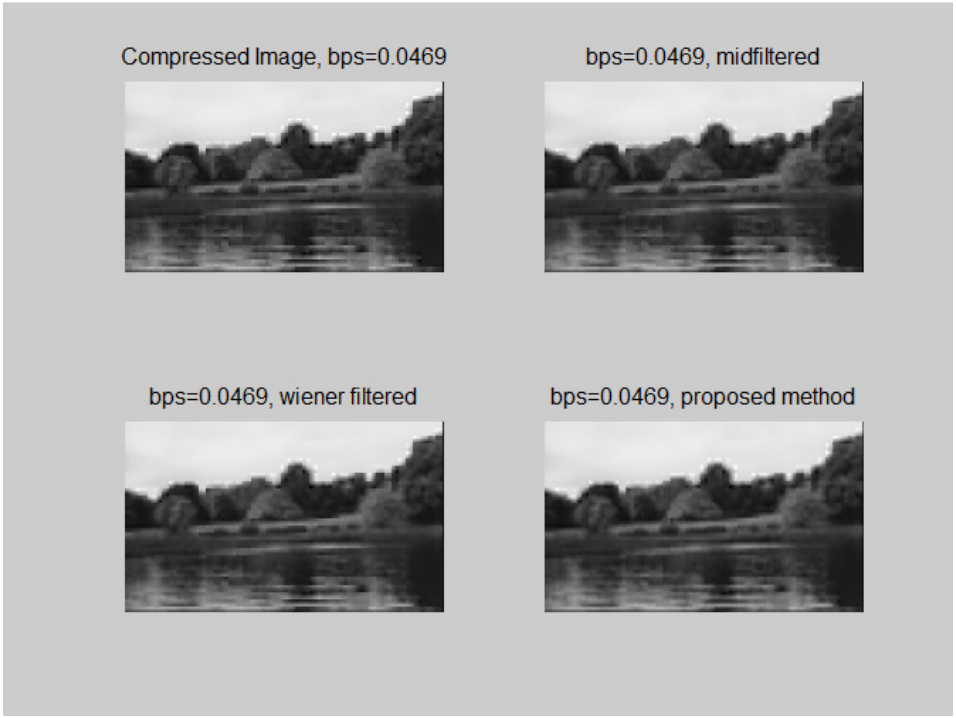

Figure 7. The effects of different methods (bps=0.0469)

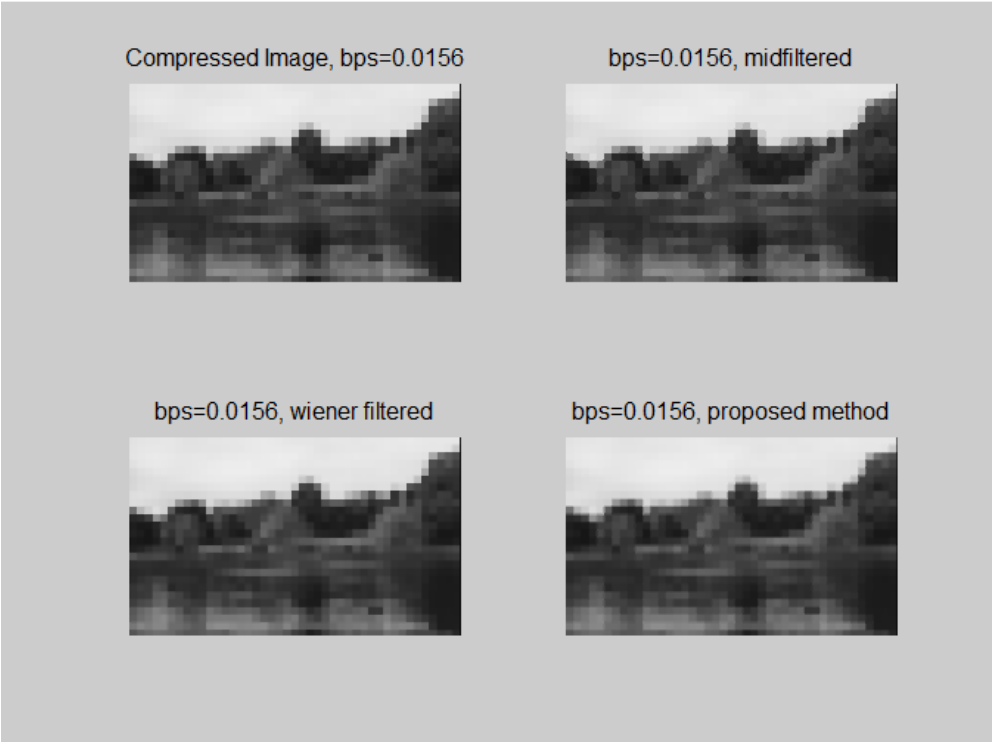

Figure 8. The effects of different methods (bps=0.0156)

Since it is not easy to compare the effects by our eyes, we need some actual numerical results. Hence, we calculated the Peak Signal to Noise Ratio (PSNR) of all the different methods. PSNR represents a measure of the peak error; it is a widely used metric to compare image compression quality. With higher PSNR value, the quality of an image is considered better. Here we show the PSNR results in Table 1.

Table 1. The PSNR of different situations (Autumn)

Bit rate	Unfiltered	Median filtered	Wiener filtered	Our method
0.0938	24.9777	25.0860	25.0572	25.2193
0.0469	23.1011	23.2140	23.2074	23.3213
0.0156	20.9932	21.0514	21.1097	21.4136

This table shows that, when the images are heavily compressed, the filters and our proposed method do effect to smooth the sudden changes caused by block artifacts, hence improve the quality of the images. But our proposed method got better results than these widely used filters. Our method results in higher PSNRs, which means better qualities.

In order to test our proposed method, we simulate the effects of our proposed method and the two kinds of widely used filters on another famous image 'cameraman'. The results are shown in Figures 9, 10 and 11. We also calculated the PSNRs of all the different methods. Here we show the PSNR results in Table 2.

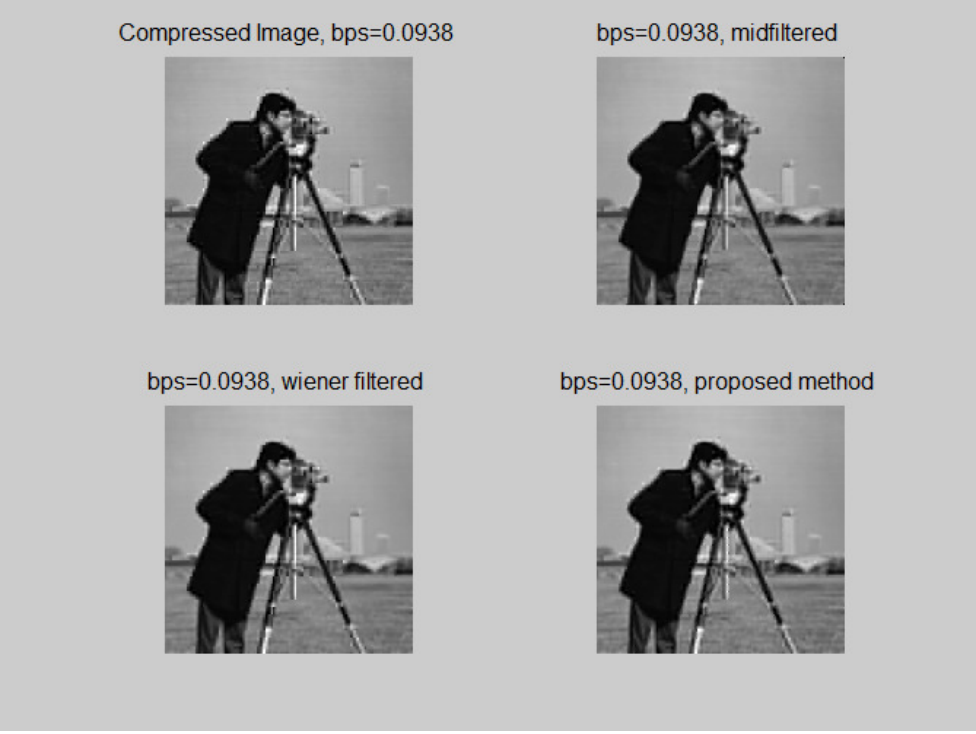

Figure 9. The effects of different methods (bps=0.0938)

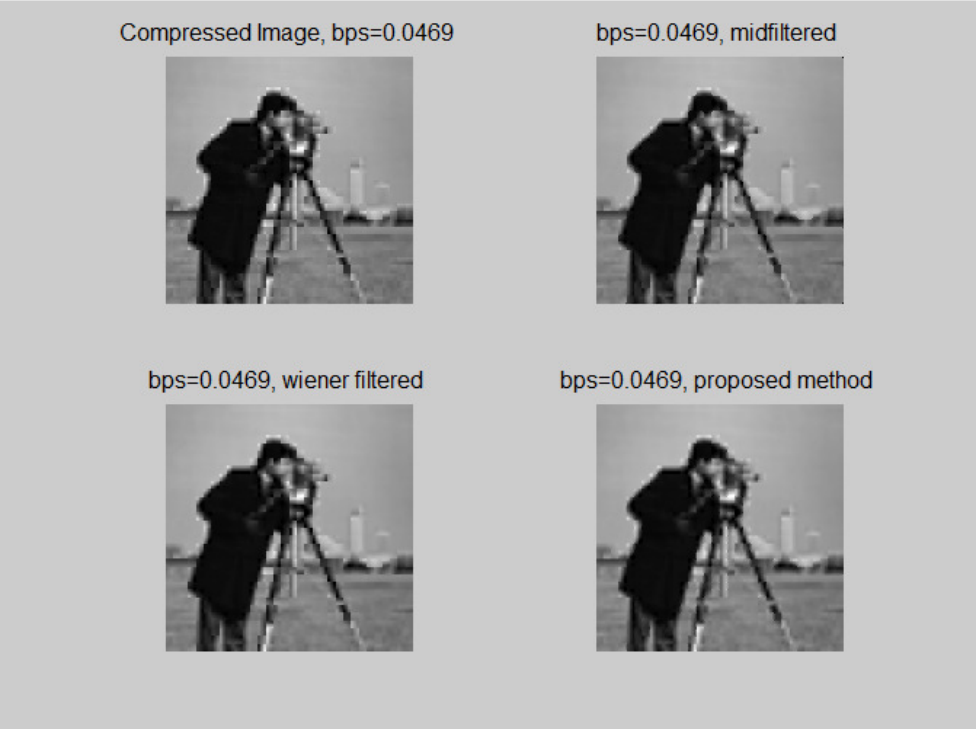

Figure 10. The effects of different methods (bps=0.0469)

Figure 11. The effects of different methods (bps=0.0156)

Table 2. The PSNR of different situations (Cameraman)

Bit rate	Unfiltered	Median filtered	Wiener filtered	Our method
0.0938	23.7897	23.9410	23.8822	23.9571
0.0469	21.9168	22.1191	22.0286	22.1938
0.0156	19.4785	19.5574	19.5837	19.8362

Table 2 again shows that our proposed method results in better quality. The main benefit of our proposed method is that the comparably better PSNR results are gotten with very simple calculations. Considering that normal filters need to calculate every pixel for an M*N image while our proposed method only need to calculate the pixels at the block borders, our method only need a small fraction of the CPU calculation source comparing with a normal image filter.

4. CONCLUSIONS

The Urban Surveillance Systems are facing the problem of huge amount of video data, and the compression technique is the most fundamental and important to deal with the data. Although the DCT based JPEG standard are widely used, it encounters problems such as block artifacts. In this paper, in order to reduce the block artifacts, we proposed a very simple but effective method. The simulation results show that our proposed method results in better quality than widely used filters while consuming much less computer CPU resources. In this way, we could reduce the visual block artifacts of DCT compressed images fast for Urban Surveillance Systems.

ACKNOWLEDGEMENTS

This work was supported by the Lancaster China Smart City Urban Surveillance Systems project under Grant number CSA7035.

REFERENCES

[1] R. Singh and V.K. Srivastava, “JPEG2000: A review and its performance comparison with JPEG”. Power, Control and Embedded Systems (ICPCES), 2012 2nd International Conference on, 2012, Pages: 1-7.

[2] M. Jha, A.S.Yumnam, D. Raju, “Comparison between image codecs: JPEG and JPEG2000”, Computing for Sustainable Global Development (INDIACom), 2014 International Conference on, 2014, Pages: 532 - 535.

[3] S. Vishaga and S. L. Das, “A survey on switching median filters for impulse noise removal”, Power and Computing Technologies (ICCPCT), 2015 International Conference on Circuit, 2015, Pages: 1 - 6.

[4] A. N. Venetsanopoulos and K. N. Plataniotis, “Adaptive filters for color image processing: A survey”, Signal Processing Conference, 2000 10th European, 2000, Pages: 1 - 4.

[5] G. A. Triantafyllidis, D. Tzovaras, D. Sampson, M. G. Strintzis, “A hybrid algorithm for the removal of blocking artifacts”, ICME '02. Proceedings. 2002 IEEE International Conference on Multimedia and Expo, 2002. Volume: 1, Pages: 161 - 164.

[6] Y. Kwon, K. I. Kim, J. Tompkin, J. H. Kim, C. Theobalt, “Efficient Learning of Image Super-Resolution and Compression Artifact Removal with Semi-Local Gaussian Processes”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, Volume: 37, Issue: 9, Pages: 1792 - 1805.

[7] C. Tang, A. W. K. Kong, N. Craft, "A knowledge-based Algorithm to remove blocking artifacts in skin images for forensic analysis", 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011, Pages: 1928 - 1931.

[8] Y. Luo, R. K. Ward, "Removing the blocking artifacts of block based DCT compressed images", IEEE Transactions on Image Processing, 2003, Volume: 12, Issue: 7, Pages: 838 - 842.

[9] J. Chou, M. Crouse, K. Ramchandran, "A simple algorithm for removing blocking artifacts in block-transform coded images", IEEE Signal Processing Letters, 1998, Volume: 5, Issue: 2, Pages: 33 - 35.