# Fast Recovery of Unknown Coefficients in DCT-Transformed Images

Simying Ong[a], Shujun Li[b], KokSheik Wong[c], KuanYew Tan[d]

[a] *Faculty of Computing and Information Technology, Tunku Abdul Rahman University College, Malaysia.*
[b] *Department of Computer Science, University of Surrey, Guildford, UK.*
[c] *School of Information Technology, Monash University Malaysia, Malaysia.*
[d] *Faculty of Computer Science and Information Technology, University of Malaya, Malaysia.*

**Abstract**

The advancement of cryptography and cryptanalysis has driven numerous innovations over years. Among them is the treatment of cryptanalysis on selectively encrypted content as a recovery problem. Recent research has shown that linear programming is a powerful tool to recover unknown coefficients in DCT-transformed images. While the time complexity is polynomial, it is still too high for large images so faster methods are still desired. In this paper, we propose a fast hierarchical DCT coefficients recovery method by combining image segmentation and linear programming. In theory the proposed method can reduce the overall time complexity by a linear factor which is the number of image segments used. Our experimental results showed that, for 100 test images of different sizes and using a naive image segmentation method based on Otsu's thresholding algorithm, the proposed method is faster for more than 92% cases and the maximum improvement observed is more than 19 times faster. While being mostly faster, results also showed that the proposed method can roughly maintain the visual quality of recovered images in both objective and subjective terms.

*Keywords:* DCT transform, image recovery, selective encryption, visual quality, image segmentation, hierarchical

## 1. Introduction

Multimedia data have been widely used in today's digital world. For the purpose of privacy and security, multimedia data is often encrypted prior to transmission and storage nowadays. In this regard, selective encryption methods are often deployed to balance computational overhead, format compliance, compression efficiency and visual security [1, 2]. Among all selective encryption methods, many are proposed for DCT-transformed images or video because DCT is the most widely used transformation in multimedia coding, which covers image and video coding standards such as JPEG, MPEG-1/2, MPEG-4 AVC/H.264 and the more recent HEVC [3].

DCT (Discrete Cosine Transform) is an orthogonal transform that can compact most energy of a highly-correlated discrete signal into a few coefficients, hence allowing more efficient compression methods to be developed in the DCT domain [4]. Among these coefficients, the first (i.e., the one with the lowest frequency) coefficient is called the *DC coefficient* while the rest are named as *AC coefficients*. Each coefficient carries distinct information of the transformed signal, although the DC coefficient is considered the most important one because it carries the average intensity of transformed signal. Most multimedia coding standards working in DCT domain apply the DCT transform to smaller blocks sequentially to reduce the overall time complexity of the transformation. As a result, the assemble of DC coefficients from all transformed blocks resembles the original signal, but at a lower resolution. Therefore, many selective multimedia encryption methods

*Email addresses:* ongsy@acd.tarc.edu.my (Simying Ong), Shujun.Li@surrey.ac.uk (Shujun Li), wong.koksheik@monash.edu (KokSheik Wong), daryltan0202@siswa.um.edu.my (KuanYew Tan)

manipulate DC coefficients [5, 6, 7, 8]. Some selective encryption methods also manipulate AC coefficients to provide a greater protection of the multimedia data. Those selective encryption methods are largely based on key-dependent substitution [9] and permutation [5, 6] of encrypted DCT coefficients.

In the past, selective encryption was argued to be offering a reasonable level of security because it conceals important visual information and the original image cannot be easily recovered in ciphertext-only setting. However, recently researchers have shown that some visual information such as edges of the original image can be retrieved by manipulating the corresponding ciphertext image [10, 11]. In addition, results reported in [12, 13, 14] also suggest that the security of selective image encryption in DCT domain had been over-estimated. In particular, Uehara et al. showed the vulnerabilities of selective image encryption by revealing the concealed DC coefficients. Specifically, the DC coefficients can be estimated by using the remaining AC coefficients and the correlations among neighboring blocks [12]. Nevertheless, the method reported in [12] suffers from pixel value overflows and underflows, which were then addressed by Li et al. in [13] using an optimization-based approach. Later on, Li et al. further proposed a generic framework to recover unknown DC and AC coefficients from the remaining coefficients that are available [14]. Specifically, Li et al. [14] utilize linear programming (LP) to search for an optimal solution in solving the DCT coefficients recovery problem.

This work aims to achieve faster recovery of unknown DCT coefficients by reducing the complexity of Li et al.'s LP model in [14] without compromising image quality. Section 2 briefly reviews the LP model in [14]. Section 3 presents the proposed segmentation based method for reducing complexity. Section 4 discusses the performance of the proposed method when compared to [14], and Section 5 concludes this paper.

## 2. Review of Li et al.'s Model [14]

In [14], Li et al. model the missing coefficient problem as a linear optimization problem which tries to minimize the linear sum of absolute differences between all pairs of neighboring pixels in horizontal and vertical directions.

$$\begin{aligned}
\text{minimize} \quad & \sum h_{i,j,i',j'} && (1)\\
\text{subject to} \quad & x(i,j) - x(i',j') \le h_{i,j,i',j'}, && (2)\\
& x(i',j') - x(i,j) \le h_{i,j,i',j'}, && (3)\\
& x = A \cdot y, && (4)\\
& x_{\min} \le x(i,j) \le x_{\max}, && (5)\\
& y(k,l) = y^*(k,l). && (6)
\end{aligned}$$

Here, $x(i,j)$ denotes the pixel value at $(i,j)$, and $y(k,l)$ denotes the DCT coefficient at $(k,l)$, while $y^*(k,l)$ denotes the known coefficient at $(k,l)$. On the other hand, $A$ is the DCT transform matrix and $h$ is the sum ranging over all pairs of neighboring pixels $(i,j)$ and $(i,j)$. The optimization problem's objective Eq. (2) matches the common assumption that the difference between two neighboring pixels in a natural image obeys the Laplacian distribution with zero mean and small variance. The optimization problem has a number of constraints which describe the relationship between DCT coefficients and pixel values (Eq. (5)), and the dynamic range of all pixel values (Eq. (6)). A number of auxiliary variables $\{h_{i,j,i',j'}\}_{i,j,i',j'}$ and two more linear constraints (Eqs. (3) and (4)) are added to linearize the sum of absolute values of all pixel value differences. The model is in principle able to recover any set of unknown DCT coefficients, although experiments in [14] considered $U$ number of lowest (frequency) DCT coefficients in each block for $U \ge 1$. This model covers the DC recovery problem as a special case where $U = 1$.[1]

[1]The DC recovery problem can also be solved using a much more computationally effective method based on the min-cost flow algorithm as reported in [15].

## 3. Proposed Method

The main idea behind the proposed method is "divide and conquer", i.e., to split the bigger problem of recovering DCT coefficients of the whole image into a number of smaller problems of recovering DCT coefficients of smaller image segments. Since the sum of adding the solutions of all image segments together is linear (of lower order compared with the polynomial-time linear programming), we expect we can save the time complexity to some degree. It is also our hope that by segmenting the image properly and merging the results of all image segments properly, visual quality of the recovered image will not be compromised. Figure 1 illustrates the process flow of the proposed segmentation based method, where two passes (phases) are involved to form a hierarchical process.

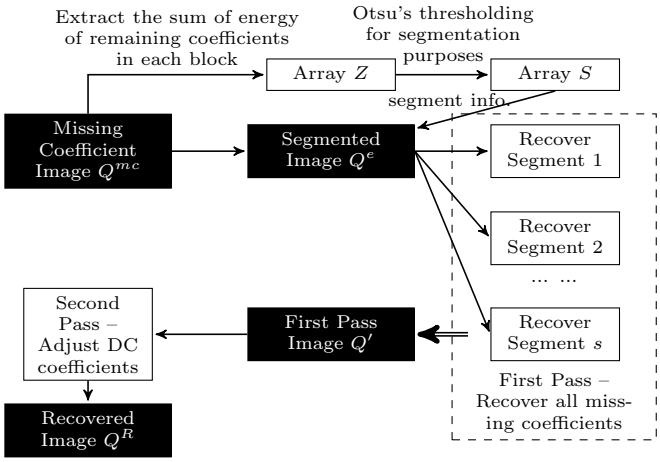



Figure 1: The proposed hierarchical segmentation based method.

### *3.1. Two-Pass Processes*

In the first pass, the input image of size $m \times n$ is divided into segments based on known DCT coefficients only and each segment is processed individually using Li et al.'s model. The image segmentation can be done this way because it is shown in [16, 17] that an array consisting of the sum of energy of AC coefficients in all blocks resembles a sketch of the original image. This observation is exploited here to identify different segments in the image with missing coefficients $Q^{mc}$. To obtain the sketch image, the sum of energy of the known DCT coefficients in each $N \times N$ block of $Q^{mc}$ is collected into an array $Z$ of size $(m/N) \times (n/N)$. The array $Z$ can be treated as an ordinary image and be segmented using any image segmentation algorithm. Let *bixel* refer to the elements in $Z$, where each *bixel* corresponds to a block of pixels in the image. All the segment information are then stored in a container named $S$ which is then utilized to divide $Q^{mc}$ into different segments and produce the segmented image $Q^e$. Next, each segment in the image is then treated as an independent DCT coefficients recovery problem to be solved. Then, Li et al.'s method [14] is adapted to recursively model and solve the smaller problems.

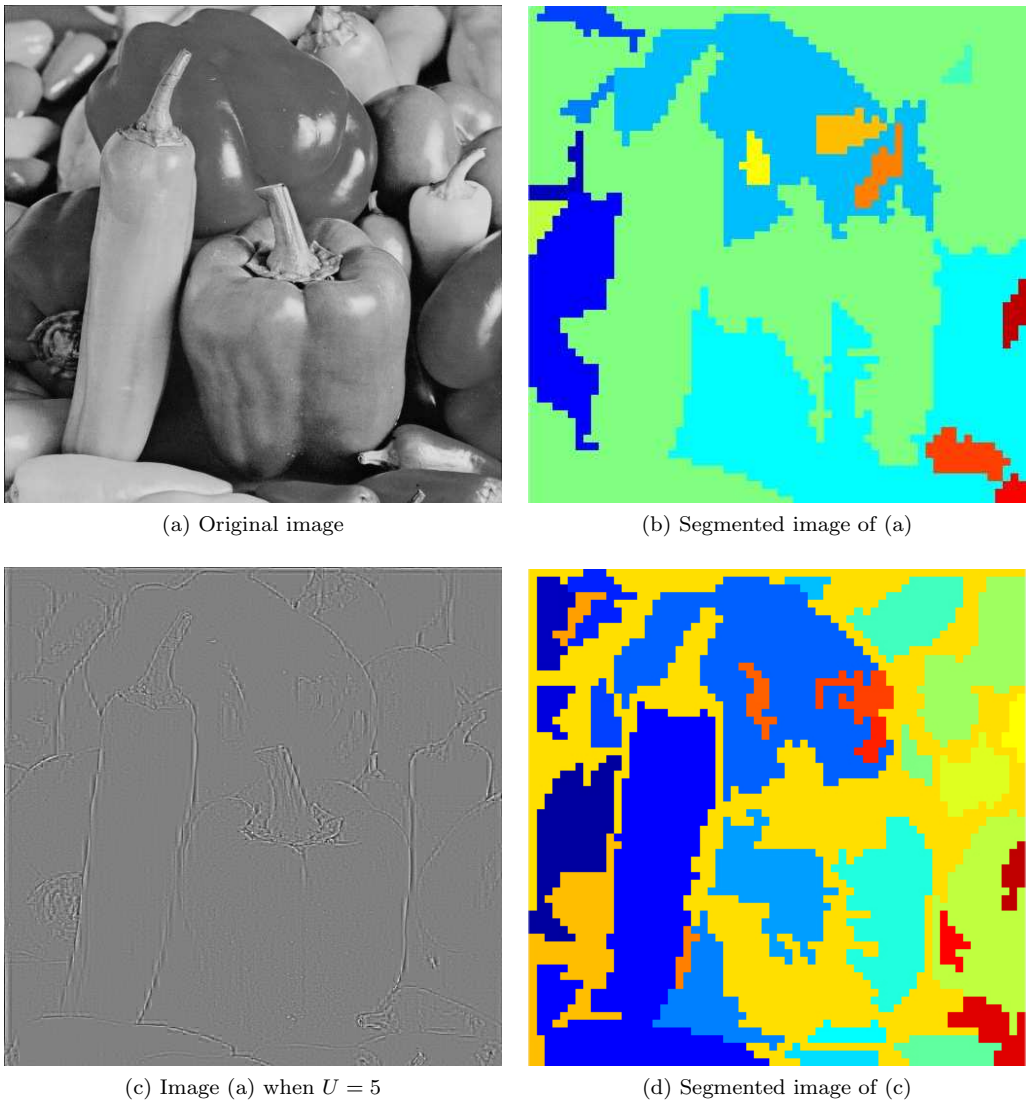
Figure 2 shows the segmentation results of the standard test image "Peppers" of size $512 \times 512$ when $U$ is 0 (i.e., original image) and 5. Since the range of $Z$ is larger than that of pixel values (i.e., 0 - 255) and not all parts of the range is useful for the segmentation task (e.g., values below the first peak $Z$ value correspond to mostly background pixels), we propose to transform $Z$ values into a smaller range defined by a starting point $z_{sp}$ and an ending point $z_{ep}$. Specifically, $z_{sp}$ is set as the first peak value in $\tau$, where $\tau$ is the bin count of $Z$, while $z_{ep}$ is computed as follows:

$$z_{ep} = \arg\min_{z_{ep}} \left\{ \sum\nolimits_{z=z_{sp}}^{z_{ep}} \tau(z) \geq \omega \sum\nolimits_{z=z_{sp}}^{z_{E_{\max}}} \tau(z) \right\}, \tag{7}$$

where $z_{E_{\max}} = \max\{Z(x,y)\}$ and $\omega \in (0,1]$. Next, $Z$ is transformed into a new array $Z'$ as follows:

$$Z'(x,y) = \begin{cases} z_{sp} & \text{if } Z(x,y) < z_{sp}; \\ z_{ep} & \text{if } Z(x,y) > z_{ep}; \\ Z(x,y) & \text{otherwise.} \end{cases} \tag{8}$$

In addition, $Z'$ is binarized using Otsu's algorithm [18]. The binary image is then segmented into different connected regions based on 4-connectivity. Note that the segmentation is performed along the boundary of DCT-transformed blocks. Finally, smaller regions with $R$ number of *bixels* or less are merged into the neighboring region with the smallest difference along the region boundary. Here, $R$ is a threshold value for region size, which determines whether a region should be merged with the neighboring region or left alone. From the segmentation results in Fig. 2, we can see that even with unknown DCT coefficients the segmentation can still be done quite accurately.



(a) Original image

(b) Segmented image of (a)

(c) Image (a) when $U = 5$

(d) Segmented image of (c)

Figure 2: Segmentation results of the image "Peppers" when $U = 0$ (original image) and $U = 5$, where $R = 10$ and $\omega = 0.75$.

At the end of the first pass, we get a number of image segments recovered separately and their levels of brightness are normally not aligned so there will be discontinuing artifacts along boundaries of image

segments. Therefore, we run a second pass to adjust the average level of brightness of each image segment to minimize the discontinuing artifacts. This can be modeled as a DC recovery problem for the whole image by resetting all recovered DC coefficients to 0 and running a DC recovery process for the whole image. Since the DC recovery problem has a smaller complexity and can also be sped up using the much faster min-cost flow based algorithm in [15], we consider the computational overhead of the second pass marginal. At the end of this hierarchical process, the final recovered image $Q^R$ is obtained.

### *3.2. Complexity Analysis*

As reported in [14], the time complexity of Li et al.'s method is $O((nm)^2U)$ and the space complexity is $O(nmU)$, where $U$ is the number of missing coefficients from each DCT block.

On the other hand, in the proposed method, the image is first divided into $s$ segments. Assuming that each segment has the same number of pixels, then there are $nm/s$ pixels in each segment. Therefore, the time complexity of the first pass will be $O((nm/s)^2U \times s) = O((nm)^2U/s)$. Since the second pass is just a DC recovery process, its time complexity will be $O((nm)^{1.5})$ if the min-cost flow based method in [15] is used[2]. As a whole the overall time complexity will be $O((nm)^{1.5} + (nm)^2U/s) = O((nm)^2U/s)$, which when $s \leq \sqrt{nm}U$ the overall time complexity will be $O((nm)^2U/s)$ which means a linear reduction by $s$ times; 2) when $s > \sqrt{nm}U$ the overall time complexity will be $O((nm)^{1.5})$ which means the reduction is exponential. In either case we can see a significant reduction of the overall complexity. In real-world cases, $s$ is normally not very large and much smaller than $\sqrt{nm}U$, so the first case will be more common. Please note that the assumption of all segments having equal size is not normally true, so the above complexity analysis just gives an optimistic prediction, and we still need to run actual experiments to find out the actual complexity reduction (if any). In terms of the space complexity, there is no much space for reduction since the original complexity is linear.

## 4. Experimental Results

The proposed method and Li et al.'s methods [14] were implemented and tested on an Intel Xeon Processor E5-2687W with 128GB of main memory. To test the performance, 120 grayscale images of different sizes were collected from various sources [19, 20, 21, 22, 23, 24, 25]. These images are divided into 6 subsets (i.e., 20 images each) based on their sizes, namely 128×128, 256×256, 256×384, 384×512, 512×512, and 768×512. These images and the source code are made available online at `http://mspih.fsktm.um.edu.my/resources-spic-shortcom/`. For the number of missing coefficients in each DCT block, we chose $U \in \{3, 5, 7, 10\}$. For all experiments, $R = 10$ was used because it was observed to achieve the best quality across resolutions.[3] The performance of the proposed method was compared to that of Li et al.'s [14] in terms of processing time and quality of the recovered image. To simplify our implementation, for the second pass, we used Li et al.'s linear programming based model to solve the DC recovery problem rather than the min-cost flow algorithm.

### *4.1. Processing Time*

The total processing time (i.e., the first pass plus the second pass) of the proposed method and that of Li et al.'s method were both recorded. Figure 3 shows the proposed method's improvement over Li et al.'s method in terms of CPU time speed-up ratio defined as follows for 20 $512 \times 768$ test images:

$$\frac{\text{CPU time of Li et al.'s method}}{\text{CPU time of the proposed method}}. \tag{9}$$

As can be seen, for all images and values of $U$, the CPU time speed-up ratio is larger than 1 and in the best case a speed-up of nearly 13 times were observed. Similar results were observed for other image sizes.

[2]It will be $O((nm)^2)$ if the linear programming based method is used but the faster method is preferred.

[3]Experiments were conducted using various values of $R$. Results suggest that the value of $R$ influences the performance in terms of speed-up and image quality. Smaller $R$ leads to better speed-up but lower image quality, and vice versa. $R = 10$ is considered in this work to achieve speed-up while suppressing image quality degradation.

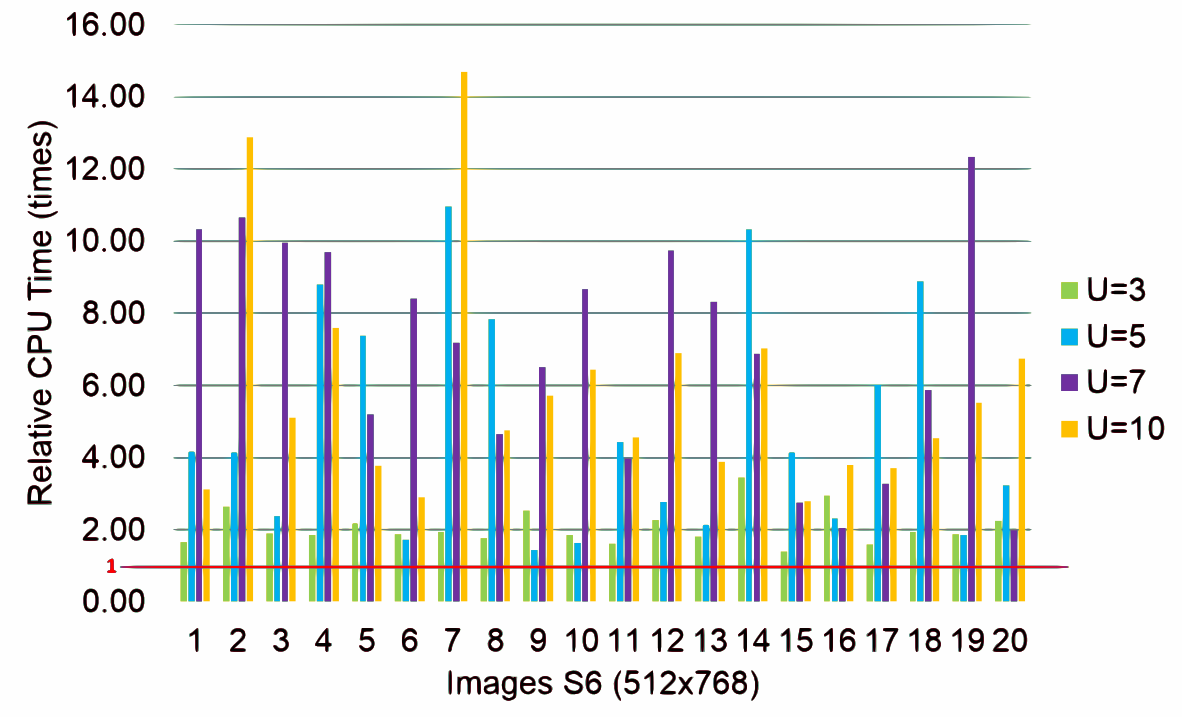



Figure 3: CPU time speed-up ratios achieved by the proposed method over Li et al.'s method for 20 512×768 images.

Table 1: Average CPU time improvement the proposed method achieved over Li et al.'s method.

| Image Size | $128 \times 128$ | $256 \times 256$ | $256 \times 384$ | $384 \times 512$ | $512 \times 512$ | $768 \times 512$ |
|---|---|---|---|---|---|---|
| $U = 3$ | 1.44 | 1.69 | 1.74 | 1.59 | 1.70 | 2.06 |
| $U = 5$ | 1.07 | 2.12 | 1.94 | 3.39 | 4.72 | 4.82 |
| $U = 7$ | 0.90 | 2.90 | 2.83 | 6.82 | 7.12 | 6.92 |
| $U = 10$ | 1.54 | 3.80 | 3.41 | 4.44 | 5.95 | 5.82 |

Table 1 presents the average performance of the proposed method in terms of CPU time improvement for different values of $U$ and different image sizes. The table shows that the average performance has a minimum value of 0.9 (i.e., slightly slower than [14]) but improvement is observed for all other cases, and improvement has a tendency to further increase as the image size and $U$ increase. In total for 37 cases our proposed method failed to achieve any speed-up, representing ∼7.8% of all 480 cases. Notably, the best improvement is observed when $U = 7$ and for $512 \times 512$ test images, in which the proposed method was on average ∼7.12 times faster than Li et al.'s method.

Most inferior cases are observed in images with lower resolutions, ranging from $128 \times 128$ to $384 \times 512$, where 78% are of resolution $128 \times 128$. While the time for solving the LP models is faster in smaller size image, the time taken to create the models for each smaller problems becomes dominant. Nevertheless, while the improvement is smaller for some test images, for each image size there are always images with very significant improvements, e.g., there is one $512 \times 512$ image giving an improvement of more than 19 times. More results are shown in the Appendix.

### *4.2. Recovered Image Quality*

Table 2 shows the average PSNR difference (dB) of recovered images between the proposed method and Li et al.'s method for different values of $U$ and different image sizes. On average, the absolute difference for all 120 test images is −0.435dB, which suggests that our image quality closely matches that of [14]. The raw

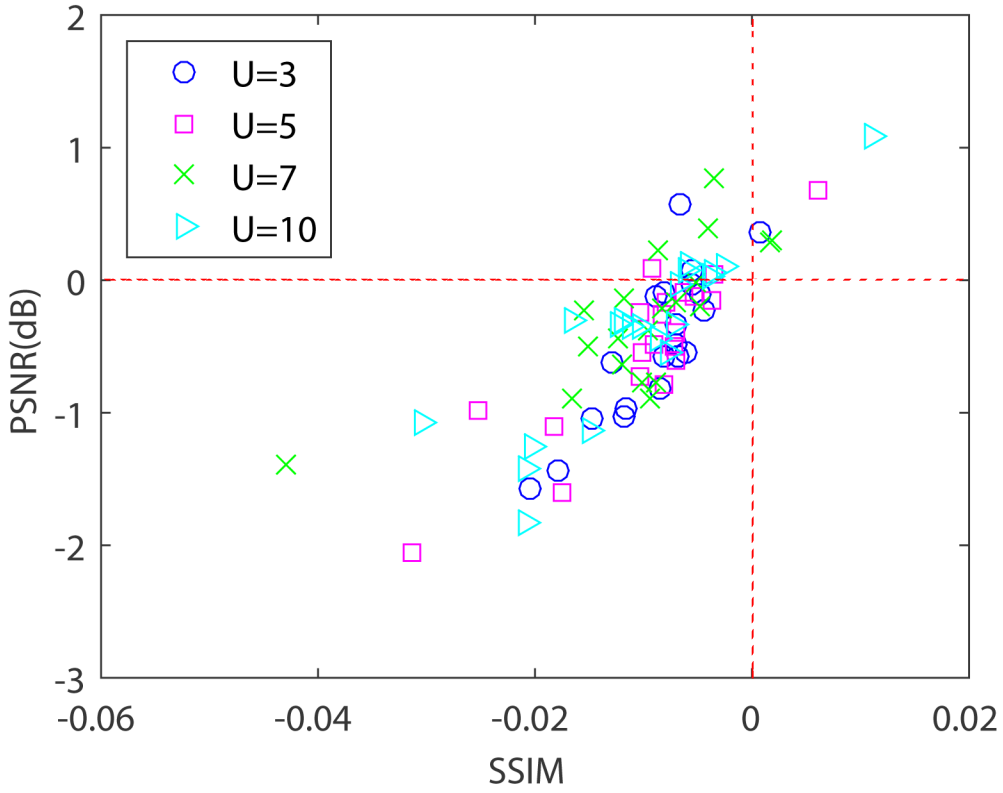



Figure 4: Visual quality differences of images recovered by the proposed method and Li et al.'s for 20 512 × 768 test images.

Table 2: Image quality performance improvement (PSNR, dB) achieved by the proposed method over Li et al.'s.

| Image Size | 128 × 128 | 256 × 256 | 256 × 384 | 384 × 512 | 512 × 512 | 768 × 512 |
|---|---|---|---|---|---|---|
| $U = 3$ | -0.210 | -0.260 | -0.647 | -0.283 | -0.154 | -0.476 |
| $U = 5$ | -0.190 | -0.305 | -0.308 | -0.097 | -0.161 | -0.500 |
| $U = 7$ | -0.281 | -0.284 | -0.193 | -0.097 | -0.176 | -0.281 |
| $U = 10$ | -0.289 | -0.058 | -0.109 | -0.143 | -0.068 | -0.410 |

Table 3: Image quality performance improvement (SSIM) achieved by the proposed method over Li et al.'s.

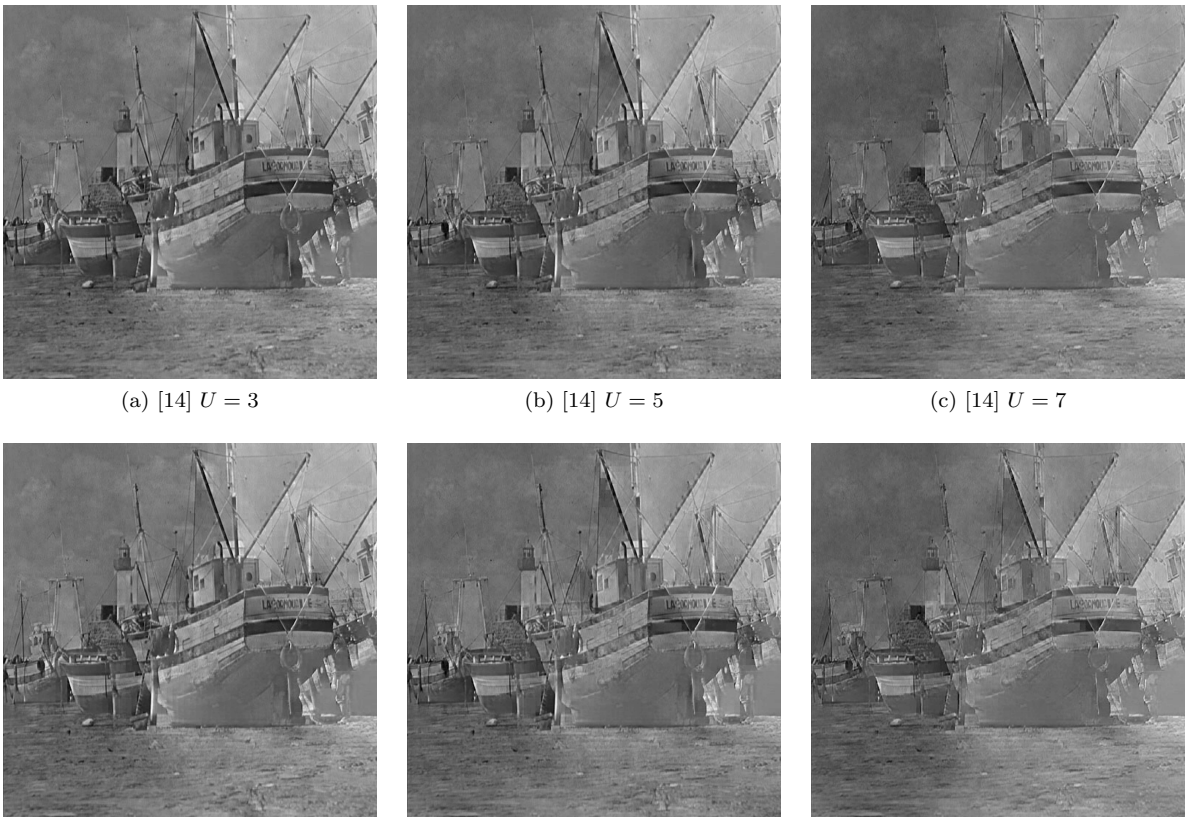
| Image Size | 128 × 128 | 256 × 256 | 256 × 384 | 384 × 512 | 512 × 512 | 768 × 512 |
|---|---|---|---|---|---|---|
| $U = 3$ | -0.009 | -0.011 | -0.013 | -0.008 | -0.008 | -0.008 |
| $U = 5$ | -0.007 | -0.011 | -0.010 | -0.005 | -0.008 | -0.009 |
| $U = 7$ | -0.010 | -0.010 | -0.010 | -0.007 | -0.009 | -0.010 |
| $U = 10$ | -0.014 | -0.008 | -0.009 | -0.007 | -0.007 | -0.010 |

differences range from $-2.266$ to 0.187dB, indicating that the quality of some images are slightly improved while some other images have a lower PSNR value. Similar results can be observed in Table 3 for the case of SSIM [26], with an average absolute difference of $-0.010$. Although the average results for both metrics are negative, the subjective quality of the images recovered by the proposed method and of those by Li et al.'s method are almost identical.

For further comparison, the graph of PSNR vs. SSIM differences is plotted in Fig. 4 for 20 512 × 768

images. This figure shows the differences are largely clustered around ($-0.5$dB,$-0.01$) with some bias toward the negative sides for both PSNR and SSIM axes, but the negative values are small in magnitude. As a representative example, the recovered images using the proposed method and Li et al.'s method are shown in Fig. 5, from which one can hardly see any noticeable quality differences (which largely holds for all test images).

From the above results, we are confident to say that the proposed method exhibits its competency in achieving faster recovery (i.e., lower complexity) while maintaining the recovery quality of the model proposed by Li et al. in [14]. More results are shown in the Appendix.



(a) [14] $U = 3$ (b) [14] $U = 5$ (c) [14] $U = 7$

(d) Proposed $U = 3$ (e) Proposed $U = 5$ (f) Proposed $U = 7$

Figure 5: Recovered images by Li et al.'s method (a)-(c) and by the proposed method (d)-(f) for different values of $U$.

## 5. Conclusion

A hierarchical segmentation method is proposed in this paper to achieve faster recovery of unknown DCT coefficients in DCT-transformed images, which improves the performance of the state-of-the-art method proposed by Li et al. in [14]. In particular, the input image with unknown coefficients are segmented and processed in two passes to reduce the time complexity of Li et al.'s method. Our experimental results confirmed that, without compromising on image quality, the proposed method can improve the time complexity and the improvement tends to increase when the number of unknown coefficients increase. The average time was reduced by a factor of 3.45 times as we observed in experiments.

The effect of more advanced image segmentation techniques and automated identification of images which can benefit from the proposed method will be our future work.

## Acknowledgement

This work was supported by the Monash University Malaysia, School of Information Technology Internal Funding Scheme (Project title: Prediction of Coefficients in Compressed Content and Its Applications).

## References

[1] A. Uhl, A. Pommer, Image and Video Encryption: From Digital Rights Management to Secured Personal Communication, Springer Science + Business Media, Inc., Boston, MA, USA, 2004.

[2] S. Li, Perceptual encryption of digital images and videos, in: R. Lukac (Ed.), Perceptual Digital Imaging: Methods and Applications, CRC Press, LLC, 2012, Ch. 14, pp. 431–468. doi:10.1201/b13016-15.

[3] J. Zeng, O. C. Au, W. Dai, Y. Kong, L. Jia, W. Zhu, A tutorial on image/video coding standards, in: Proceedings of 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA 2013), 2013. doi:10.1109/APSIPA.2013.6694346.

[4] W. B. Pennebaker, J. L. Mitchell, JPEG Still Image Data Compression Standard, Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[5] L. Tang, Methods for encrypting and decrypting MPEG video data efficiently, in: Proceedings of 4th ACM International Conference on Multimedia, ACM, 1996, pp. 219–229.

[6] B. Bhargava, C. Shi, S.-Y. Wang, MPEG video encryption algorithms, Multimedia Tools and Applications 24 (1) (2004) 57–79.

[7] S. Li, G. Chen, A. Cheung, B. Bhargava, K.-T. Lo, On the design of perceptual MPEG-video encryption algorithms, IEEE Transactions on Circuits and Systems for Video Technology 17 (2) (2007) 214–223. doi:10.1109/TCSVT.2006.888840.

[8] X. Niu, C. Zhou, J. Ding, B. Yang, JPEG encryption with file size preservation, in: Proceedings of 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2008, pp. 308–311.

[9] M. Takayama, K. Tanaka, Y. Nakajima, T. Kouchi, A scalable video scrambling method in MPEG compressed domain, in: Proceedings of 2008 International Symposium on Communications, Control and Signal Processing, 2008, pp. 1035–1040.

[10] W. Li, Y. Yuan, A leak and its remedy in JPEG image encryption, International Journal of Computer Mathematics 84 (9) (2007) 1367–1378.

[11] K. Minemura, K. Wong, R. Phan, K. Tanaka, A novel sketch attack for H.264/AVC format-compliant encrypted video, IEEE Transactions on Circuits and Systems for Video Technology, in press. doi:10.1109/TCSVT.2016.2589742.

[12] T. Uehara, R. Safavi-Naini, P. Ogunbona, Recovering DC coefficients in block-based DCT, IEEE Transactions on Image Processing 15 (11) (2006) 3592–3596.

[13] S. Li, J. J. Ahmad, D. Saupe, C.-C. J. Kuo, An improved DC recovery method from AC coefficients of DCT-transformed images, in: Proceedings of 17th IEEE International Conference on Image Processing (ICIP 2010), IEEE, 2010, pp. 2085–2088. doi:10.1109/ICIP.2010.5653467.

[14] S. Li, A. Karrenbauer, D. Saupe, C.-C. J. Kuo, Recovering missing coefficients in DCT-transformed images, in: Proceedings of 18th IEEE International Conference on Image Processing (ICIP 2011), IEEE, 2011, pp. 1537–1540. doi:10.1109/ICIP.2011.6115738.

[15] S. Cornelsen, A. Karrenbauer, S. Li, Leveling the grid, in: Proceedings of the Meeting on Algorithm Engineering & Experiments (ALENEX 2012), SIAM, 2012, pp. 45–54.
URL `http://siam.omnibooksonline.com/2012ALENEX/data/papers/016.pdf`

[16] K. Minemura, K. Wong, X. Qi, K. Tanaka, A scrambling framework for block transform compressed image, Multimedia Tools and Applications, in press. doi:10.1007/s11042-016-3338-x.

[17] S. Ong, K. Minemura, K. Wong, Progressive quality degradation in JPEG compressed image using DC block orientation with rewritable data embedding functionality, in: Proceedings of 2013 IEEE International Conference on Image Processing (ICIP 2013), 2013, pp. 4574–4578.

[18] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man and Cybernetics 9 (1) (1979) 62–66.

[19] Signal and Image Processing Institute (SIPI), University of Southern California, The USC-SIPI image database, `http://sipi.usc.edu/database/` (since 1977).

[20] Center for Image Processing Research (CIPR), Rensselaer Polytechnic Institute, CIPR still images, `http://www.cipr.rpi.edu/resource/stills/` (2002).

[21] Computer Vision Group, University of Granada, CVG - UGR - image database, `http://decsai.ugr.es/cvg/dbimagenes/` (2002-2014).

[22] S. Li, 98 test images collected from the different standard test image databases plus two images taken by Shujun Li, `http://hooklee.com/Papers/Data/AC2DC/Images.zip` (2010).

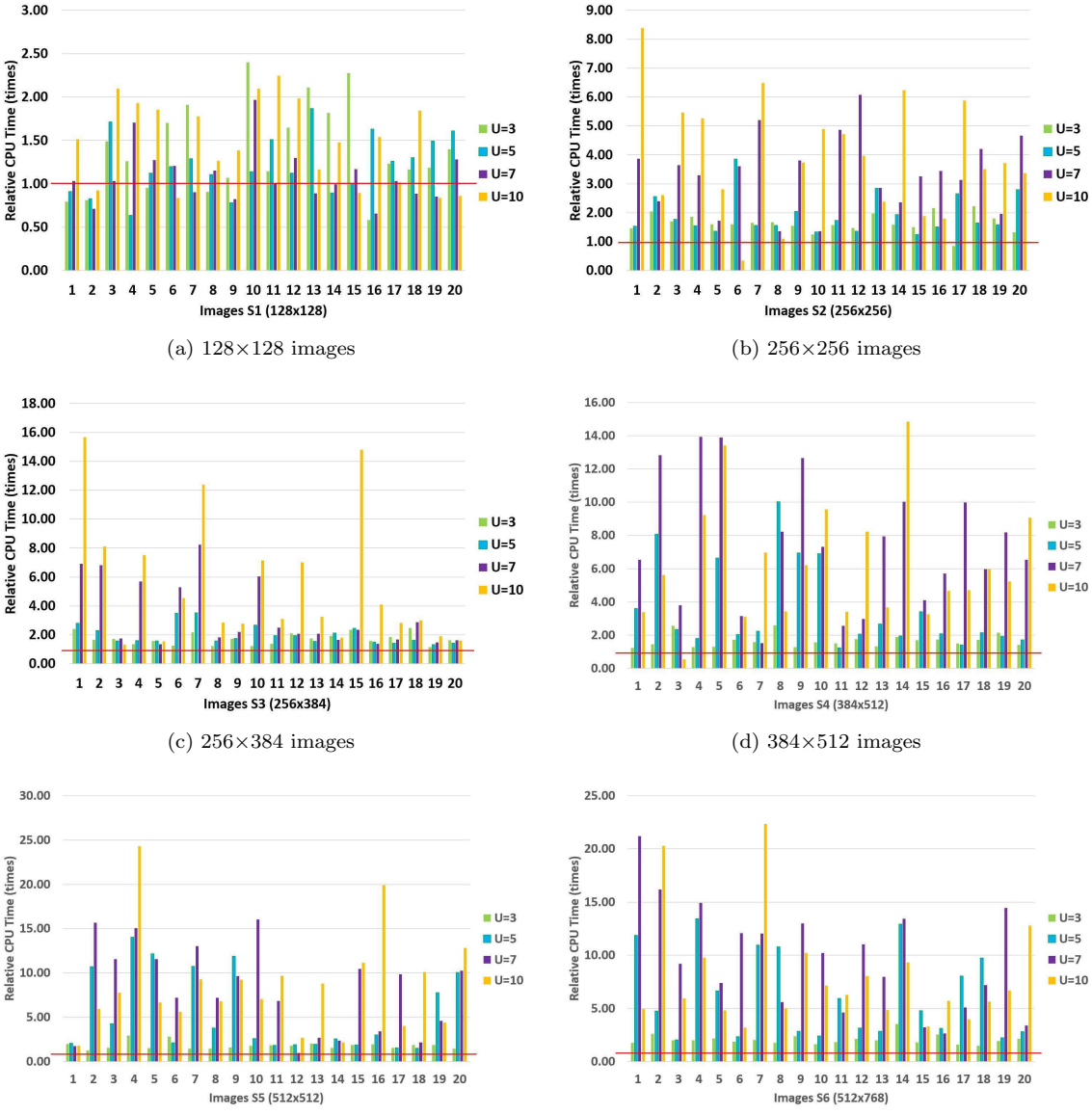
[23] G. Schaefer, M. Stich, UCID - an uncompressed colour image database, in: Storage and Retrieval Methods and Applications for Multimedia 2004, Vol. 5307 of Proceedings of SPIE, 2004, pp. 472–480. doi:10.1117/12.525375.

[24] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: Proceedings of 2006 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2006), 2006, pp. 2169–2178.

[25] L. Zhang, X. Wu, A. Buades, X. Li, Color demosaicking by local directional interpolation and nonlocal adaptive thresholding, Journal of Electronic Imaging 2 (20) (2011) 023016.

[26] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612.

## Appendix




(a) 128×128 images

(b) 256×256 images

(c) 256×384 images

(d) 384×512 images

(e) 512×512 images

(f) 512×768 images

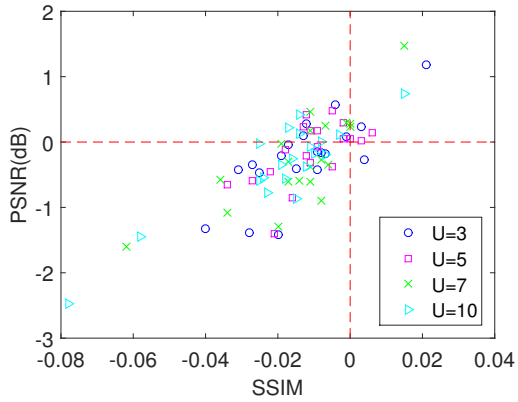
Figure 6: CPU time speed-up ratios achieved by the proposed method over Li et al.'s method for $R = 0$ images.

Table 4: Average result the proposed method achieved over Li et al.'s method in $R = 0$.

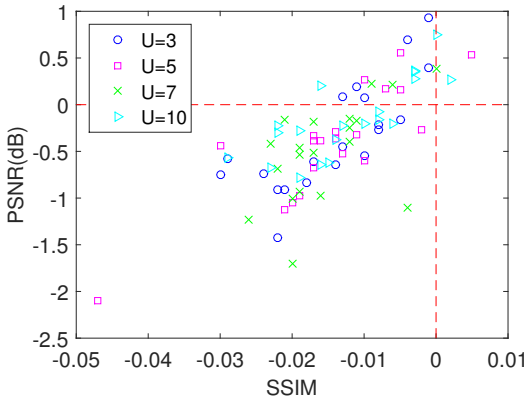
| Image Size | $U = 3$ | | | $U = 5$ | | |
|---|---|---|---|---|---|---|
| | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM |
| 128×128 | 1.39 | -0.272 | -0.014 | 1.22 | -0.151 | -0.011 |
| 256×256 | 1.64 | -0.335 | -0.014 | 1.92 | -0.407 | -0.014 |
| 256×384 | 1.72 | -0.670 | -0.018 | 2.00 | -0.318 | -0.015 |
| 384×512 | 1.66 | -0.306 | -0.011 | 3.56 | -0.173 | -0.009 |
| 512×512 | 1.80 | -0.272 | -0.014 | 5.41 | -0.160 | -0.013 |
| 512×768 | 2.05 | -0.599 | -0.012 | 6.18 | -0.478 | -0.012 |
| Image Size | $U = 7$ | | | $U = 10$ | | |
| | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM |
| 128×128 | 1.09 | -0.239 | -0.012 | 1.47 | -0.353 | -0.019 |
| 256×256 | 3.35 | -0.501 | -0.015 | 3.92 | -0.156 | -0.012 |
| 256×384 | 3.28 | -0.224 | -0.017 | 5.36 | -0.149 | -0.016 |
| 384×512 | 7.39 | -0.068 | -0.011 | 6.22 | -0.207 | -0.013 |
| 512×512 | 8.11 | -0.246 | -0.015 | 8.50 | -0.174 | -0.014 |
| 512×768 | 9.74 | -0.324 | -0.013 | 8.01 | -0.488 | -0.014 |

Table 5: Average result the proposed method achieved over Li et al.'s method in $R = 1$.

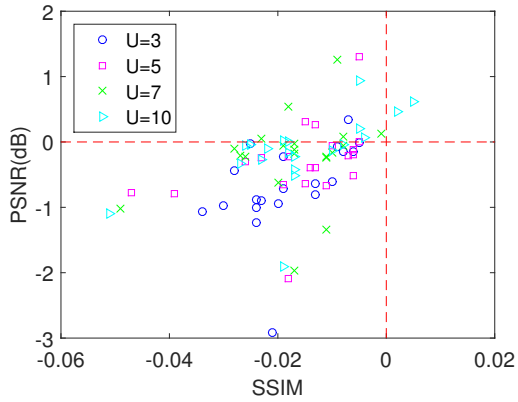
| Image Size | $U = 3$ | | | $U = 5$ | | |
|---|---|---|---|---|---|---|
| | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM |
| 128×128 | 1.42 | -0.202 | -0.012 | 1.18 | -0.119 | -0.009 |
| 256×256 | 1.59 | -0.396 | -0.013 | 2.04 | -0.347 | -0.012 |
| 256×384 | 1.67 | -0.623 | -0.015 | 1.97 | -0.278 | -0.013 |
| 384×512 | 1.66 | -0.314 | -0.011 | 3.84 | -0.148 | -0.008 |
| 512×512 | 1.69 | -0.196 | -0.011 | 5.07 | -0.188 | -0.011 |
| 512×768 | 2.02 | -0.570 | -0.011 | 5.35 | -0.429 | -0.010 |
| Image Size | $U = 7$ | | | $U = 10$ | | |
| | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM |
| 128×128 | 0.98 | -0.259 | -0.012 | 1.47 | -0.289 | -0.014 |
| 256×256 | 2.95 | -0.493 | -0.014 | 3.87 | -0.111 | -0.010 |
| 256×384 | 3.04 | -0.198 | -0.014 | 5.10 | -0.119 | -0.013 |
| 384×512 | 7.03 | -0.056 | -0.009 | 6.78 | -0.157 | -0.011 |
| 512×512 | 8.63 | -0.243 | -0.013 | 8.22 | -0.145 | -0.012 |
| 512×768 | 7.96 | -0.346 | -0.012 | 7.10 | -0.480 | -0.013 |



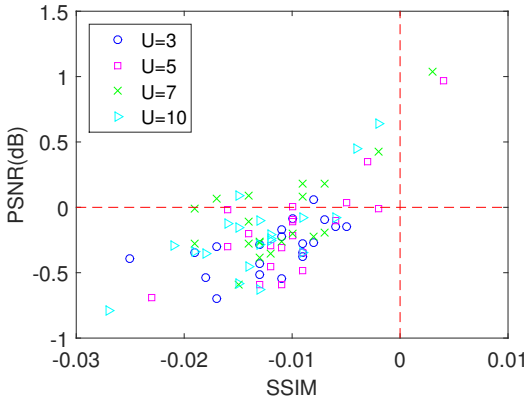

(a) 128×128 images



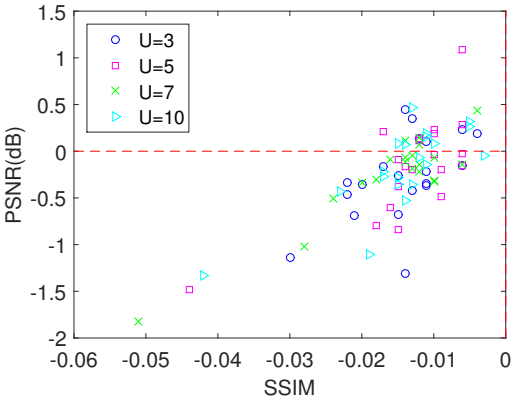

(b) 256×256 images



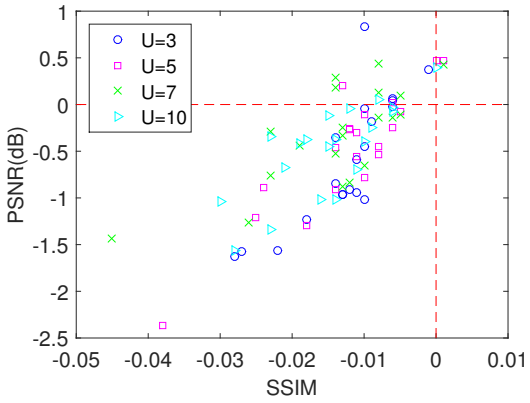

(c) 256×384 images




(d) 384×512 images

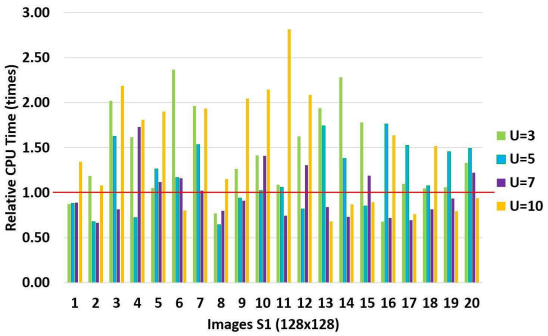



(e) 512×512 images

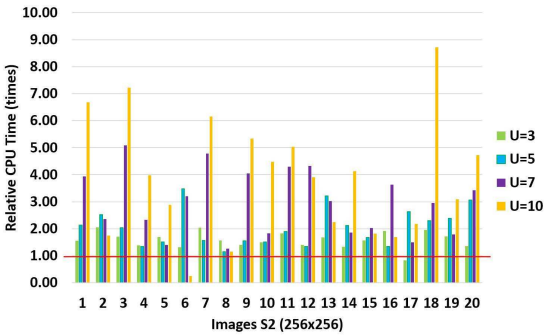



(f) 512×768 images

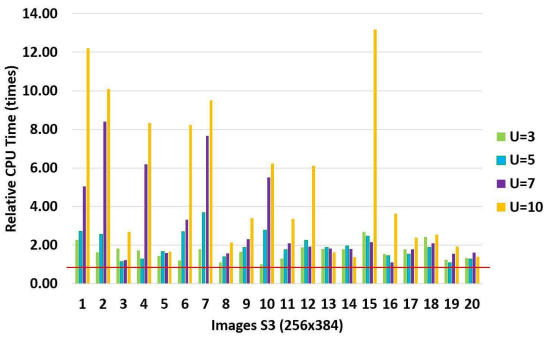
Figure 7: Visual quality differences of images recovered by the proposed method and Li et al.'s for $R = 0$ images.



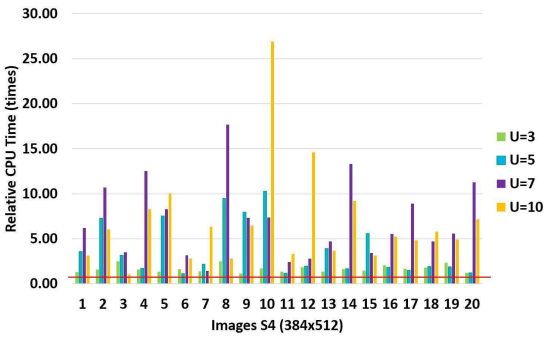

(a) 128×128 images



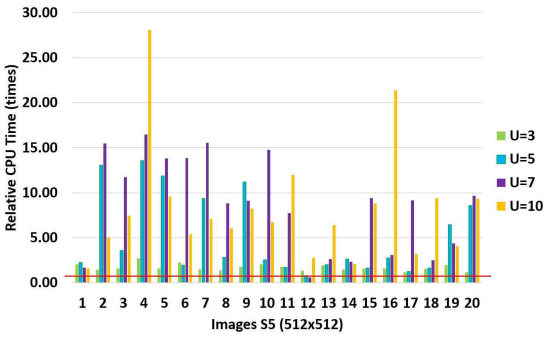

(b) 256×256 images



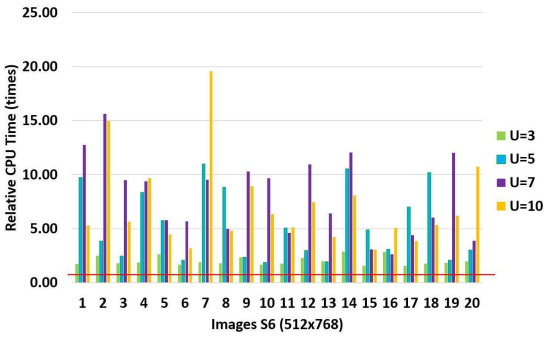

(c) 256×384 images




(d) 384×512 images

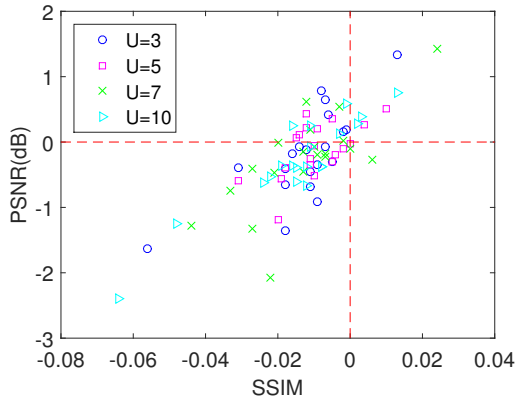



(e) 512×512 images

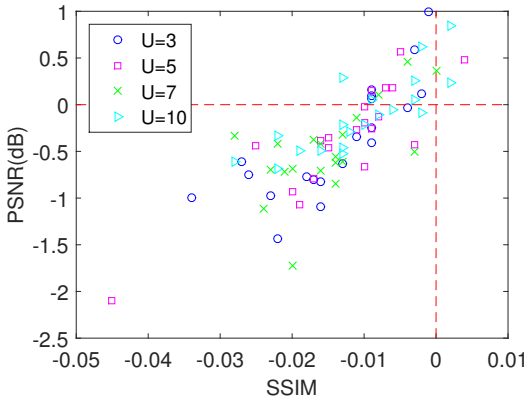



(f) 512×768 images

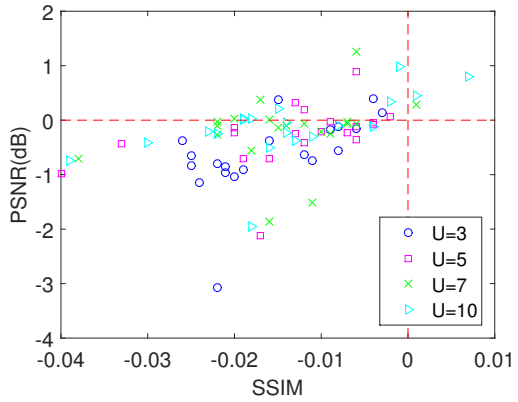
Figure 8: CPU time speed-up ratios achieved by the proposed method over Li et al.'s method for $R = 1$ images.



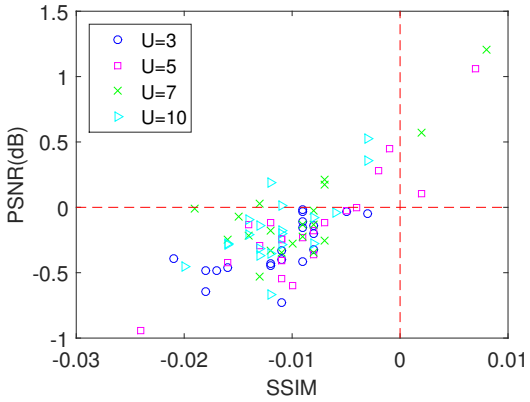

(a) 128×128 images



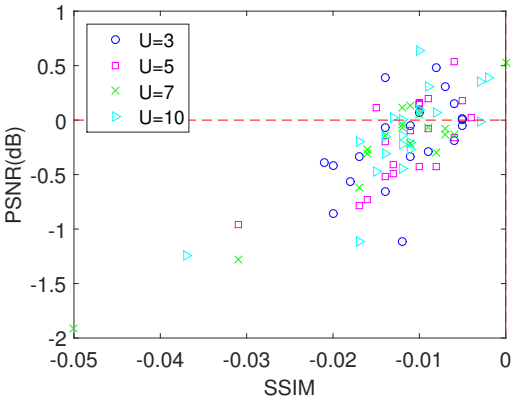

(b) 256×256 images



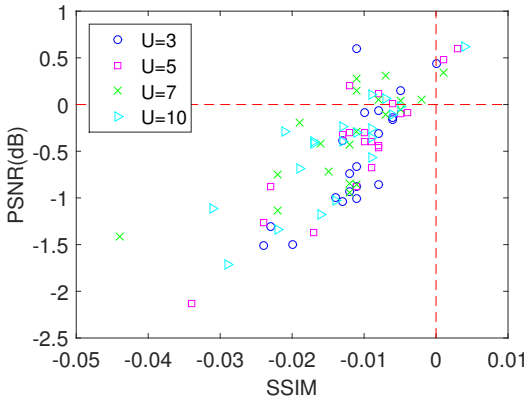

(c) 256×384 images




(d) 384×512 images

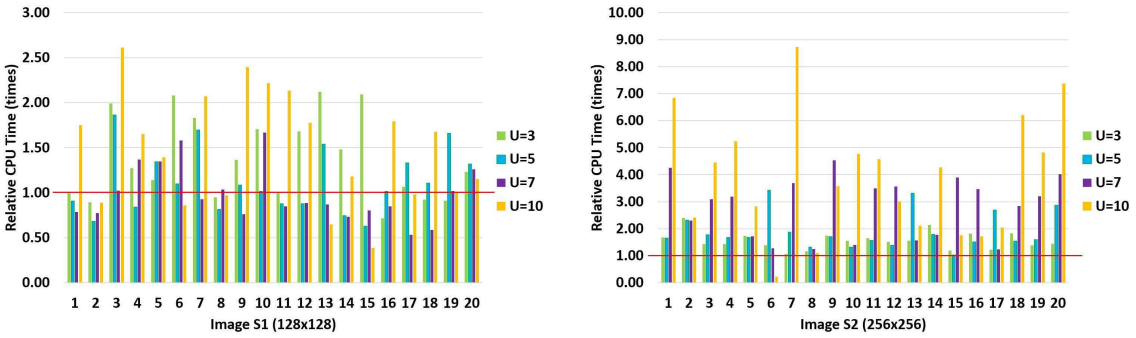



(e) 512×512 images




(f) 512×768 images

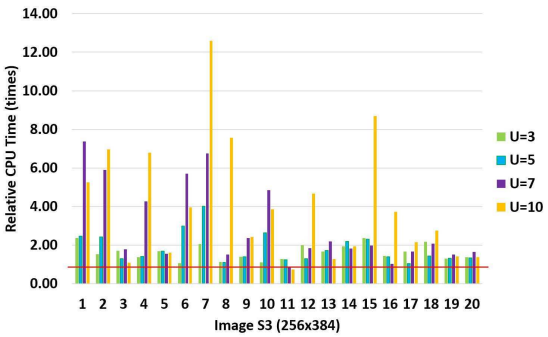
Figure 9: Visual quality differences of images recovered by the proposed method and Li et al.'s for $R = 1$ images.



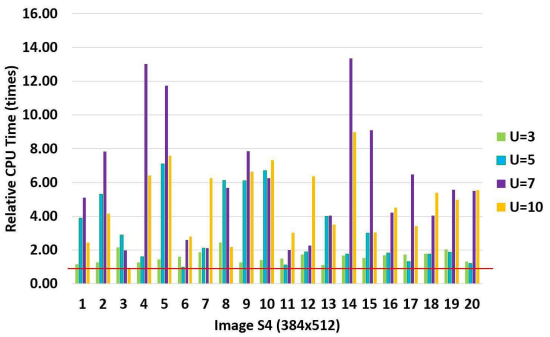

(a) 128×128 images



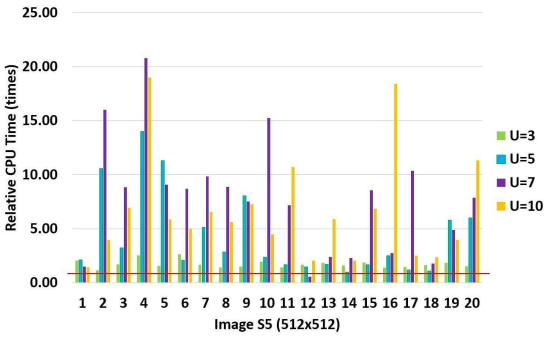

(b) 256×256 images



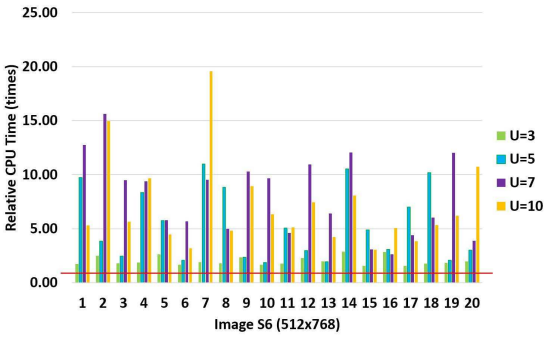

(c) 256×384 images




(d) 384×512 images

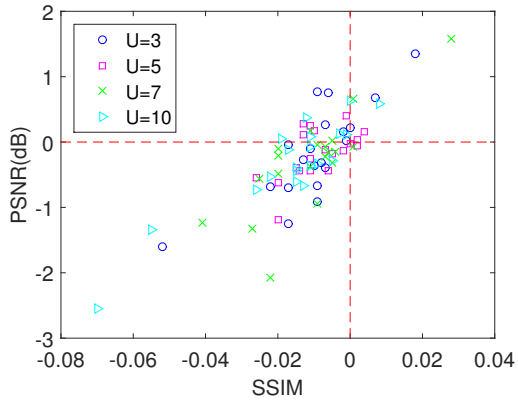



(e) 512×512 images

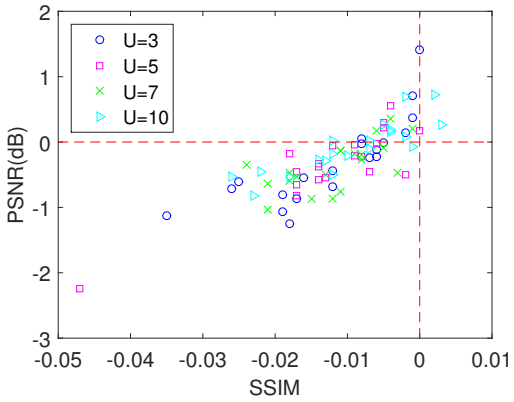



(f) 512×768 images

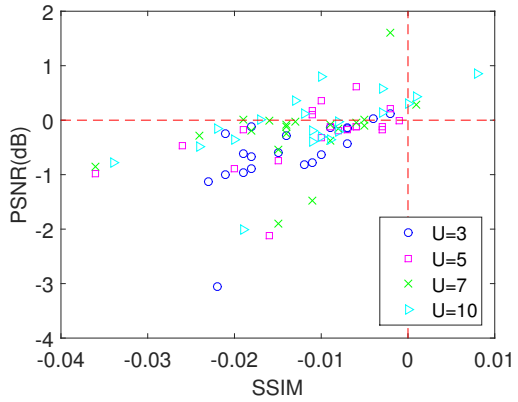
Figure 10: CPU time speed-up ratios achieved by the proposed method over Li et al.'s method for $R = 5$ images.



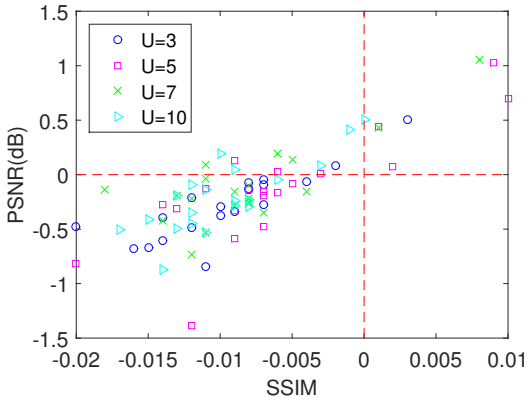

(a) 128×128 images



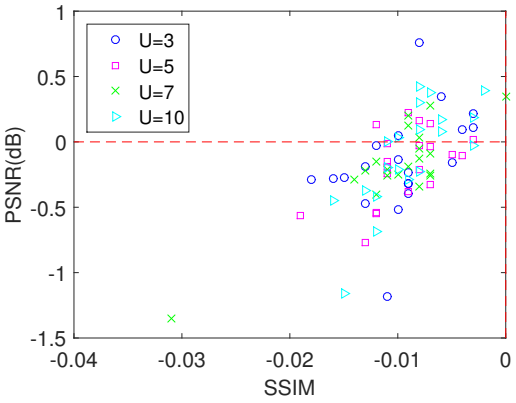

(b) 256×256 images



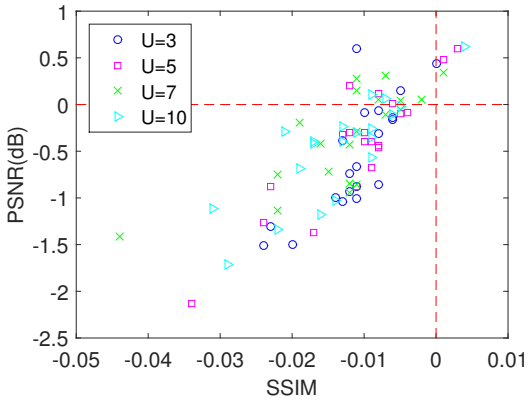

(c) 256×384 images




(d) 384×512 images

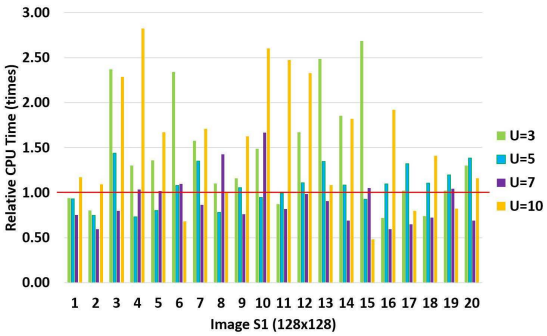



(e) 512×512 images

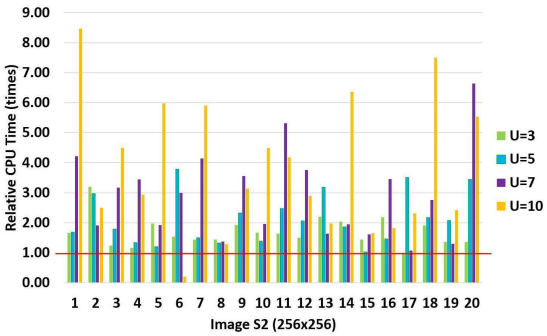



(f) 512×768 images

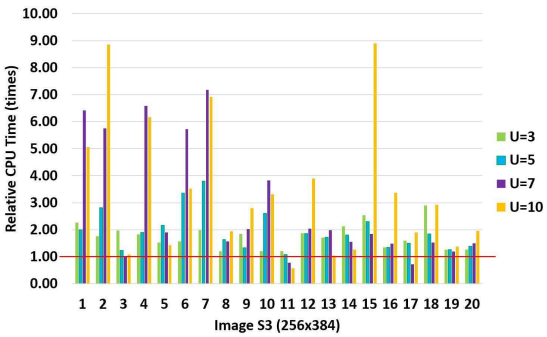
Figure 11: Visual quality differences of images recovered by the proposed method and Li et al.'s for $R = 5$ images.



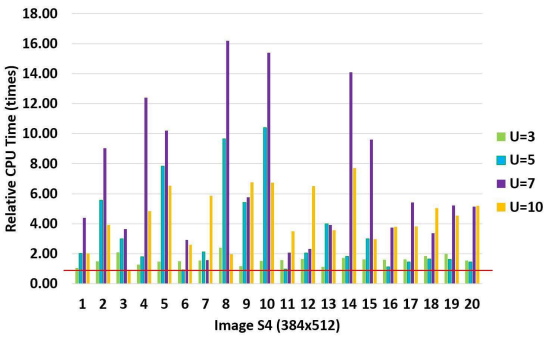

(a) 128×128 images



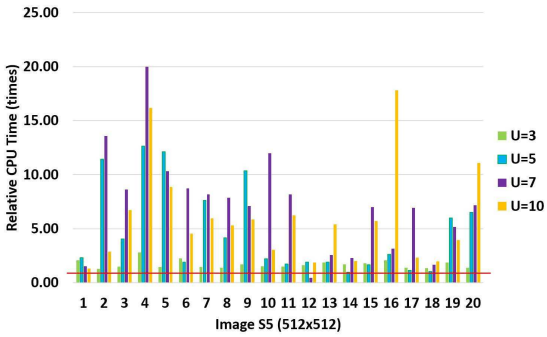

(b) 256×256 images




(c) 256×384 images

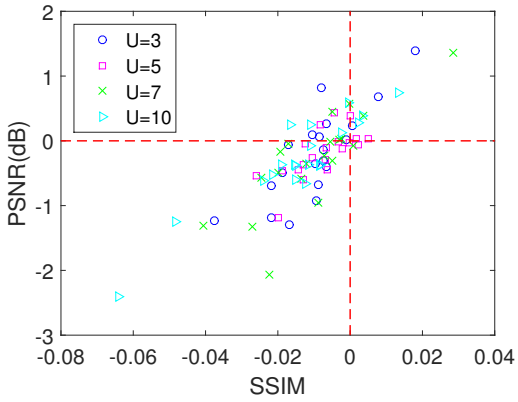



(d) 384×512 images

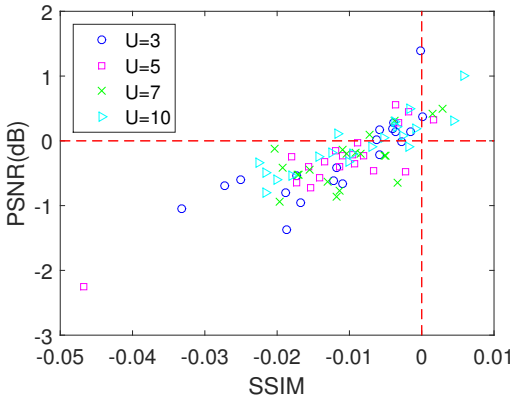



(e) 512×512 images

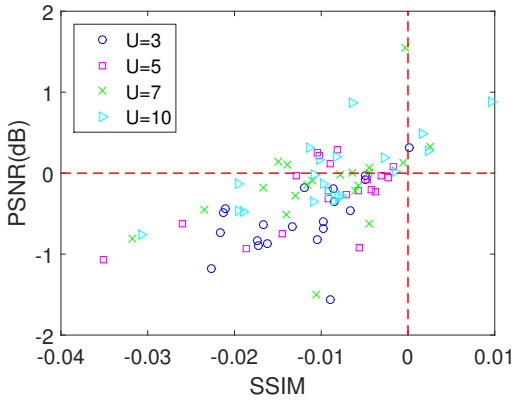
Figure 12: CPU time speed-up ratios achieved by the proposed method over Li et al.'s method for $R = 10$ images.



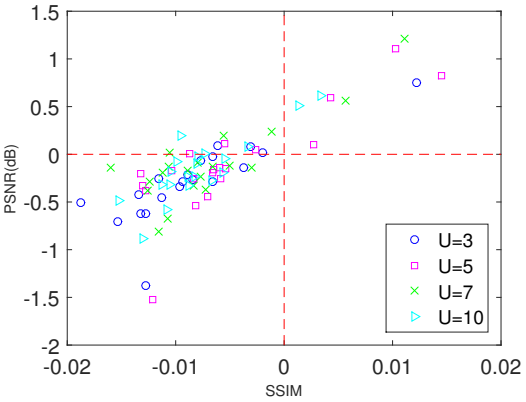

(a) 128×128 images



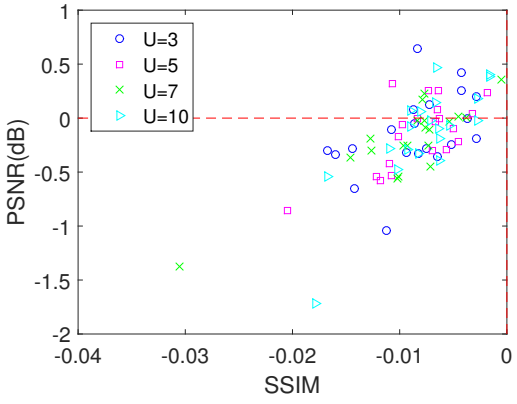

(b) 256×256 images




(c) 256×384 images




(d) 384×512 images




(e) 512×512 images

Figure 13: Visual quality differences of images recovered by the proposed method and Li et al.'s for $R = 10$ images.

Table 6: Average result the proposed method achieved over Li et al.'s method in $R = 5$.

| Image Size | $U = 3$ | | | $U = 5$ | | |
|---|---|---|---|---|---|---|
| | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM |
| 128×128 | 1.37 | -0.155 | -0.009 | 1.12 | -0.172 | -0.008 |
| 256×256 | 1.56 | -0.302 | -0.012 | 1.90 | -0.321 | -0.011 |
| 256×384 | 1.63 | -0.627 | -0.013 | 1.84 | -0.263 | -0.011 |
| 384×512 | 1.59 | -0.280 | -0.009 | 3.12 | -0.131 | -0.006 |
| 512×512 | 1.72 | -0.160 | -0.009 | 4.28 | -0.176 | -0.009 |
| 512×768 | 1.99 | -0.513 | -0.009 | 5.43 | -0.442 | -0.009 |
| Image Size | $U = 7$ | | | $U = 10$ | | |
| | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM | Mean CPU Time (times) | Mean PSNR (dB) | Mean SSIM |
| 128×128 | 0.98 | -0.296 | -0.010 | 1.47 | -0.323 | -0.016 |
| 256×256 | 2.78 | -0.367 | -0.011 | 3.90 | -0.076 | -0.009 |
| 256×384 | 2.93 | -0.228 | -0.012 | 4.04 | -0.085 | -0.011 |
| 384×512 | 6.03 | -0.107 | -0.008 | 4.77 | -0.195 | -0.009 |
| 512×512 | 7.74 | -0.171 | -0.010 | 6.61 | -0.098 | -0.008 |
| 512×768 | 6.92 | -0.299 | -0.010 | 6.13 | -0.441 | -0.015 |