Data Embedding Method Using Photo Effects with Resistance to Compression

William K.W. Yeong,* Simying Ong,* and KokSheik Wong†
* University of Malaya, Malaysia
E-mail: {woa180021@siswa., simying.ong@}um.edu.my Tel: +60-3-79576300
† Monash University Malaysia, Malaysia
E-mail: wong.koksheik@monash.edu Tel: +60-3-55146090

***Abstract*— This work aims to integrate data embedding capability into common photo effects, which simplifies the 2-step process of achieving photo effect and data embedding. Specifically, three novel data embedding methods using photo effect are put forward. A preliminary study is first conducted on popular editing software to identify the commonly available photo effects. The photo effects are found to be sketch, halftone, and vintage, and they are modified to incorporate the data embedding capability. The data embedding algorithms are designed in a way so that the embedded data can survive compression. Three prototypes are built as proof of concepts to verify the feasibility of achieving photo effect generation and data embedding simultaneously. Experiments are carried to verify the basic performance of the proposed data embedding method, including embedding capacity, image quality and robustness against compression.**

***Index Terms*—data embedding, photo effect, sketch, halftone, vintage**

I. Introduction

Thanks to the affordable price tag for ubiquitous data network services and smart devices, contents such as image are increasingly generated and shared online. To put statistics into context, in every minute, an average of $55,140$ photos are posted online on the Instagram platform in the year of 2019 [1] while $14,700$ photos are uploaded into Facebook in the year 2020 [2]. To achieve self-presentation [3, 4], users share photo via various social networking service (SNS) platforms, including Facebook, Snapchat, and Instagram. In fact, this high rate of photo posting is generating a vast pool of free photos, which can be exploited by certain parties. For example, these photos often contain users' private information, and they are uncovered for purposes such as social phishing and advertisement [5, 6].

As the number of users of SNS platforms and photo sharing sites increases, the risk of copyright and privacy infringement also increases, particularly when the platforms / sites allow a user to save, download or share other users' photos by a single click of a button. Infringing on an author's copyright by re-posting, sharing or downloading the photo without owner's consent is no different than other forms of copyright violation [7]. Furthermore, the increasing number of online photos also causes issues related to storage and image retrieval. Issue such as lack of coherent metadata for images have also caused some challenges for the users in perceiving their favorite image content [8, 9].

Therefore, online images need to be better managed. Data embedding, which inserts some data into an image, is one of the solutions for addressing the aforementioned problems. Here, the data can be external to the image, derived from the image, or a combination of both. For instance, an embedded watermark is utilized to claim ownership of the image [10], a fingerprint is inserted as a trace to identify user who distributed the photo [11], a metadata is added for multimedia enrichment such as image retrieval and hyper-linking related content [12], etc.

Various data embedding techniques are proposed over the years. The techniques include Least Significant Bit (LSB) insertion [13, 14, 15, 16], Histogram Shifting (HS) [17, 18, 19] and Prediction Error Expansion (PEE) [20, 21, 22]. As the name implies, LSB insertion hides data into the image by replacing the LSB bitplane with the data to be embedded. HS utilizes the peak bin (i.e., the pixel value with the highest frequency) and the adjacent emptied bin to represent data. PEE uses the errors between the original and predicted pixel values to achieve data embedding. Regardless of their purposes, most conventional methods [13] - [22] hide information by modifying the pixels or coefficients in the transformed domain. However, these direct modifications on the image pixels or coefficients are distorting or damaging the image, which is a common drawback in the conventional data embedding methods. In addition, they also change the statistical properties of the image, which may attract attacker's attention.

Therefore, in this work, we exploit the commonly utilized photo effects to embed data. Our proposal is motivated by the fact that photo editing is one of the important steps performed prior to the sharing of a photo on SNS. By adopting our proposal, user does not need to face the trouble of switching between apps or programs to edit the photo and embed data. This integrated process will eliminate extra step(s) needed for data embedding, and also encourage user to protect his/her photo prior to sharing it online. Furthermore, from the perspective of photo editor, metadata can be inserted into the image for managerial purposes in both the online and local environments.

This work was part of the research project entitled *Information Hiding using Pattern Image Synthesis Approach* (Project ID:BK025-2018) supported by the BKP Faculty Grant awarded by University of Malaya, Malaysia.

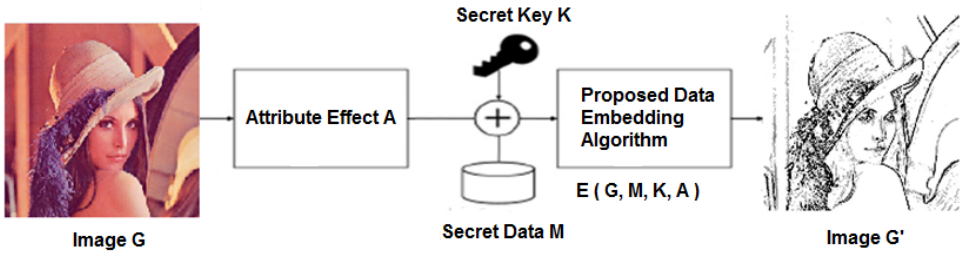

Fig. 1: Proposed photo effect-based data embedding framework.

TABLE I: Results of preliminary study on photo editing applications and software.

Photo Effect	PicsArt	GIMP	Photoshop	Paint.net	Pixlr
Halftone	✓	✓	✓	✓	✓
Vintage	✓	✓	✓	✓	✓
Sketch	✓	✓	✓	✓	✓

II. Photo Effects

Figure 1 shows the proposed photo effect-based data embedding framework. Let G denotes the image of interest, which will be transformed by adding the attribute effect A for embedding the payload bit M with the key K. Specifically, G, M, and K are the input of the embedding function E, i.e.,

$$G' \leftarrow E(G, M, K, A), \tag{1}$$

where A is the effect attribute information and G' is the output image with embedded data. In particular, M is encoded by using the effect attribute in the redesigned data embedding algorithm using photo effects.

To identify the popular photo effects, we surveyed various photo editors, including Adobe Photoshop, GIMP, Paint.net, PicsArt and Pixlr. The findings are summarized in Table I. Three common effects, namely, Sketch, Halftone and Vintage, are considered in this work. Specifically, these effects are short-listed because they contain attributes which can be redesigned for data embedding purposes. Another deciding factor is that they are available in various popular photo editors. It is important to ensure that the proposed method is applicable in common photo effects so that the proposed photo effect-based data embedding method can be utilized in place of its photo-effect-only filter.

III. Proposed Method

The proposed photo effect-based data embedding methods, each using a different photo effect, are elaborated in the following subsections. For the rest of the presentation, assume that the input image G is of size $M \times N$ pixels, and each pixel is referred to as $G(x, y)$ where $x \in \{1, \ldots, M\}$ and $y \in \{1, \ldots, N\}$.

A. Prototype 1 (P1): Sketch Effect

Sketch effect turns the image into a rough pencil drawing which concentrates on the essential features of the image. Essentially, the sketch function in P1 converts the color input image into 24-bit grayscale image G^* (i.e., three 8-bit RGB color channels, denoted as $Ł_R, Ł_G, Ł_B$). To generate the Sketch Effect, the Sobel kernel is applied for edge detection, and the adopted algorithm [23] selects the specific segments from the image then converts them white or black segments. Sobel kernel is utilized in this prototype to search for the smooth regions in a simple and time-efficient manner [24].

Next, the resulting pixel values are exploited to encode the payload M. In particular, the carrier pixels are pixels with their RGB values each less than a predefined threshold value ϕ. Here, τ is chosen based on empirical experiments and it is chosen to achieve two objectives: (i) identify the black pixels which are utilized to embed the payload, and; (ii) to classify the payload bit from the carrier pixel, even after the image G' is compressed. The threshold should be chosen to for the best performance in terms of payload recovery, even after compression.

Subsequently, all selected carrier pixels in each row are modified by replacing them by a certain value to embed one payload bit. Here, we show an example to step through the embedding process:

- S1 Convert the message into binary representation. Without loss of generality, consider the payload $M = 9_{10} = 1001_2$.
- S2 Scan the pixels of the sketched image G^*, row-by-row, to identify the embeddable row. Here, an embeddable row is a row of pixels consisting of at least one pixel where $Ł_{R,G,B} < \phi$, while Figure 2(a) shows an example of the searching process for embeddable rows when $\phi = 20$.
- S3 Each embeddable row is manipulated to carry one payload bit. For instance, if the payload bit is '1', the algorithm modifies the value of $Ł_G$ of all carrier pixels in that selected row while maintaining the values of $Ł_R$ and $Ł_B$, as shown in the first row of Figure 2(a). Otherwise, if the payload bit is '0', the $Ł_G$ value of all carrier pixels are replaced by 0 while the values in other channels remain unchanged. Furthermore, the values of

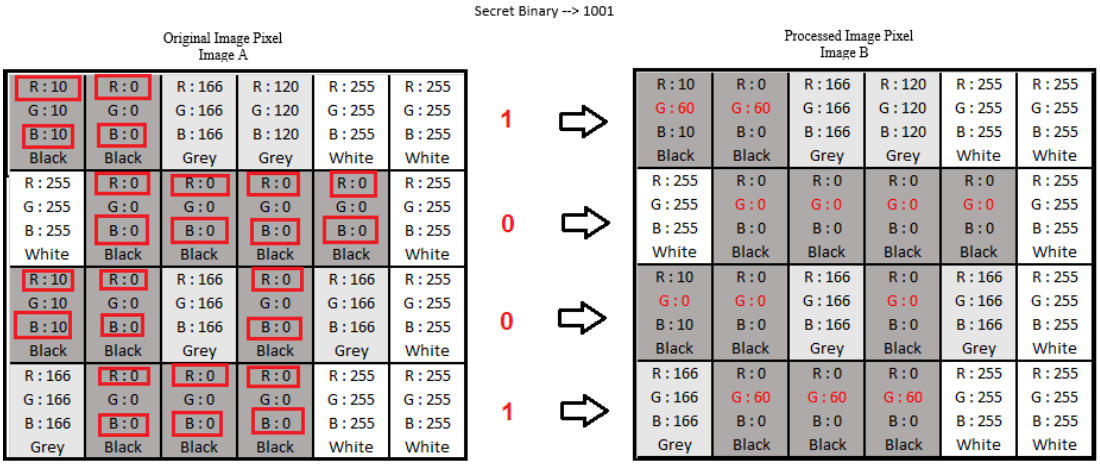

Fig. 2: Illustration of the embedding processing for handling the payload $M = 1001_2$ using $\phi = 60$.

those non-carrier pixels in the embeddable row are also maintained, i.e., not modified.

S4 S2 and S3 are repeated until all the rows were processed. Figure 2 shows the final outcome after embedding $M = 1001$ into the embeddable rows. Finally, the output image with embedded data, G', is generated.

The algorithm summarized above (i.e., P1) creates a pattern in three channels, namely, $Ł_R$, $Ł_G$ and $Ł_B$ to embed data. In particular, a pixel's RGB-channels are modified to assume to pattern *low-high-low* to encode '1', while the pattern *low-low-low* encodes '0'. The purpose of exploiting color channel pattern for data embedding is to ensure that the processed image can maintain the relative pattern even after compression is applied.

At the receiver's end, the algorithm has to first identify the embeddable rows with carrier pixels. To ensure the correct payload bit is extracted, a majority vote strategy is also utilized to determine the payload bit encoded by each embeddable row.

Based on our empirical experiments, compression often increases the values in the range of $[20, 40]$. Therefore, the threshold value ϕ is set at 100 for decoding purpose. For instance, if any of the RGB-channels for a particular pixel is less than 100, it will be identified as a carrier pixel. Similarly, if the row consists of at least one carrier pixel, it is categorized as an embeddable row. For each embeddable row, each carrier pixel is examined. In our example, $Ł_G$ of each carrier pixel is modified to encode either '1' or '0'. If the $Ł_G$ is of *high* pattern, then it encodes '1', otherwise it encodes '0'. Therefore, any value between 0 and 60 (viz., our previously modified values in the embedding algorithm) can be selected to distinguish the *low-high* pattern. The decoded bits from all carrier pixels in an embeddable row are collected accordingly. If the majority bits are of value '1', then payload bit '1' is output. On the other hand, if the majority yielded '0', then the payload bit '0' is output. The decoding process is repeated until all rows are analyzed to recover the entire payload.

B. Prototype 2 (P2): Halftone Effect

Halftone is a reprographic technique that simulates continuous-tone imagery through an assemble of dots with varying sizes and spacing, thus generating a gradient-like effect [25]. For our second prototype, Halftone effect is generated by implementing the standard error diffusion algorithm using Floyd and Steinberg's weights. We observed that Praveen et al.'s algorithm [26] generates 8-bit grayscale image with halftone effect G^*, which is made up of a large number of black dots. Hence, in our proposed algorithm, the black dots are not only utilized to generate gradient-like feature in producing halftone effect, but they are also redesigned to carry data.

S1 Similar to P1, the message is first converted into binary representation. Again, we consider the payload $M = 1001$.

S2 We calculate and record the total number of black dots for each row τ_i, i.e., $T = \{\tau_1, \tau_2, \ldots, \tau_M\}$. A pixel is called *black dot* if it is less than a threshold value ψ. For instance, by referring to the example shown in Figure 3, the recorded black dot information for each row is $T = [3, 4, 3, 3]$ when $\psi = 80$.

S3 Our halftone effect-based data embedding method utilizes odd-oven τ value to encoded the payload bit. In particular, odd τ encodes '1', while τ encodes '0'. For instance, consider the example given in Figure 4. If τ_1 is odd and the first payload bit is '1', then no changes are needed for the first row. However, if the total number of black dots in a row cannot represent the secret bit (e.g., the third row), some modifications are performed to enforce relevant odd-even value. Here, the embedding algorithm transforms one non-black-dot value to become a black dot (viz., modify to the value 0). However, the selected non-black-dot to be converted must fulfill the following two conditions: (i) it has a value more than 200, and (ii) located adjacent to another black

255	255	255	0	0	0
0	255	0	255	0	0
0	255	255	255	0	0
0	0	255	0	255	255

Fig. 3: Count the number of black dots.

255	255	255	0	0	0
0	255	0	255	0	0
0	255	255	0	0	0
0	0	255	0	255	255

Fig. 4: Data embedding using P2 algorithm.

pixel to ensure the correct data extraction and enhance imperceptibility of output image. In the case where there is no black dot in the row of interest, our method chooses an arbitrarily pixel in the row to modify.

S4 Repeat S3 and S4 until all rows are processed to encode the payload bit. Figure 5 shows the final output of image rows after data embedding. Finally, the output image with embedded data, denoted by G', is generated.

In the decoding stage, the reserve steps are performed on G' to identify T. Based on empirical experiment, $\psi = 80$ is chosen as the threshold to identify the black dot. The total number of identified black dots are collected and stored in the string T in sequential order. Finally, the payload bits are extracted by examining the odd-oven property of all τ values in the string T.

C. Prototype 3 (P3): Vintage Effect

Vintage effect transforms an image to mimics dated and retro-liked photo. Our method adds jagged border (viz., uneven pattern) on the edges of the vintage image for data embedding purposes while enhancing the natural appearance of the vintage effect. First, the input image G is pre-processed as follows:

$$\begin{aligned} S(Ł_R) &= 0.383 \cdot G(Ł_R) + 0.769 \cdot G(Ł_G) + 0.189 \cdot G(Ł_B) \\ S(Ł_G) &= 0.349 \cdot G(Ł_R) + 0.686 \cdot G(Ł_G) + 0.168 \cdot G(Ł_B) \\ S(Ł_B) &= 0.274 \cdot G(Ł_R) + 0.534 \cdot G(Ł_G) + 0.131 \cdot G(Ł_B) \end{aligned} \quad (2)$$

where $G(Ł_R), G(Ł_G), G(Ł_R)$ are the RGB channels, operator $(\cdot)$ and operator $(+)$ denote matrix multiplication and addition operations, respectively. The output $S(Ł_R), S(Ł_G), S(Ł_R)$ are the RGB channels of the transformed image after applying

255	255	255	0	0	0
0	255	0	255	0	0
0	255	255	0	0	0
0	0	255	0	255	255

Fig. 5: Output with payload bit 1001 embedded.

sepia toning. Next, noise is added[1] to S, and the resulting image is overlapped with a vintage background image downloaded online [27] to create the vintage image G^*. Subsequently, the following steps are performed to embed data:

S1 Convert the payload into binary conversion. Again, we consider the payload $M = 1001$.

S2 A white frame is created on the vintage image G^* by manipulating the pixels near the image border. We designed our algorithm so that the four borders of the white frame are of the same width. Specifically, let ω denote the width of the white frame in terms of number of pixels. The pixel values at the image borders (i.e., top ω rows, bottom ω rows, left ω columns and right ω columns) are set to $\{255, 255, 255\}$ to produced an artificial white frame.

S3 Starting from the $\omega + 2$th row to the $M - \omega - 2$th row, each row is processed to embed one payload bit. Here, no jagged pattern is added to the row if the payload bit is '0'. On the other hand, the jagged pattern is added next to the white frame pixels of a row to represent the payload bit '1'. The jagged pattern is added by transforming two pixels located next the white frame into white pixels. To create a balance jagged pattern, if it is an odd row, the RGB values of $G^*(x, \omega + 1)$ and $G^*(x, \omega + 2)$ are modified to the value of 255. On the contrary, if it is an even row, the values in $G^*(x, N - \omega - 1)$ and $G^*(x, N - \omega - 2)$ are set to 255. The rows are processed until all the payload bit are embedded. Figure 6 shows the results after modifying G^* to embed '1001'.

S4 An extra processing step is performed to enhance the overall appearance of the jagged patterns around the vintage image. In particular, Figure 7 shows an example where arbitrarily pixels are converted into white pixels (viz., set to 255) (see to those in red boxes) around the white frame.

S5 Finally, the output image with embedded data, denoted by G', is generated.

In the decoding stage, the algorithm detects the jagged pattern near the white frame pixels to determine the payload bit. If there is no jagged pattern on both sides, the payload bit '0' is extracted. On the other hand, if there is a jagged border

[1]In this work, the `imnoise` function in Matlab v2019 is utilized to add noise.

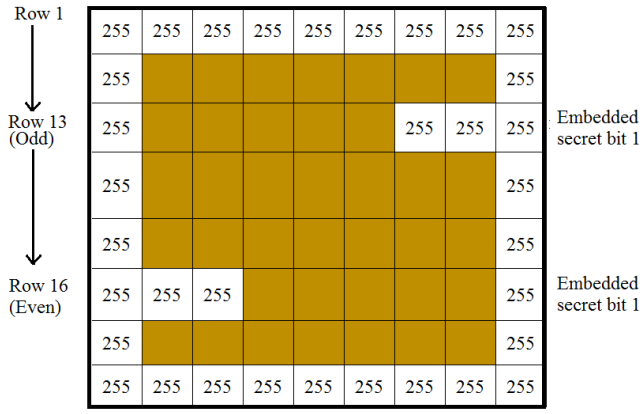

Fig. 6: Data embedding by modifying two pixels near the white frame pixels.

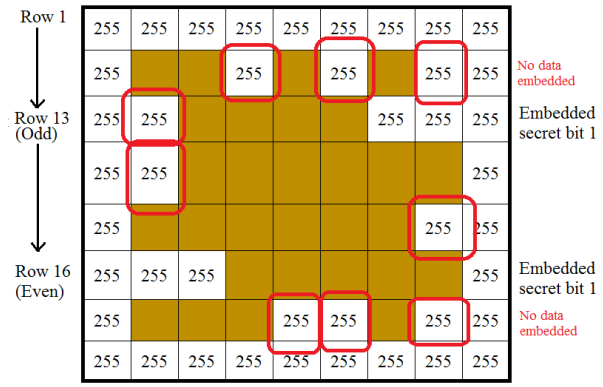

Fig. 7: Adding random jagged pattern to enhance the natural appearance of the output image G'.

(viz., two white pixels) on either side of the row, the payload bit '1' is extracted.

IV. Experiment Results

The aforementioned data embedding methods are implemented in MATLAB v2019. To evaluate the performance of the proposed methods, the Berkeley Segmentation Dataset (BSD300) [28] is utilized as the test images. The current public distribution of the BSD300 dataset contains 300 color images each of dimension 481×321 pixels (or 321×481 pixels). BSD is consider for conducting experiments because they are not biased towards any specific images properties, thus they can realize an objective evaluation which covers most types of image. Performance in terms of embedding capacity, extraction rate after applying JPEG compression with different quality factors, and image quality are evaluated. In all experiments, the prototypes will pre-process the input image by resizing them into the dimension of 512×512 pixels.

A. Embedding Capacity

Each prototype is implemented by utilizing only one photo effect. In other words, either (i) sketch effect which enforces pattern in all three color channels, (ii) halftone effect which utilizes the odd-even count of black dots, or (iii) vintage effect which utilizes the jagged pattern, is utilized. Table II shows the average embedding capacity for each prototype for 300 images in the BSD300 dataset. The results indicate that P2 has the highest embedding capacity (i.e., 512 bits) among the proposed methods. This is also the maximum capacity that can be achieved by using the proposed algorithm because each row is designed to carry exact one bit of the payload data.

TABLE II: Average embedding capacity per image for each prototype.

Prototype (Effect)	Average Embedding Capacity [bits]
P1 (Sketch)	468.1
P2 (Halftone)	512.0
P3 (Vintage)	490.0

TABLE III: Average image quality for each prototype when comparing between the output images with and without embedded data.

Prototypes (Effect)	SSIM	PSNR [dB]
P1 (Sketch)	0.994	30.573
P2 (Halftone)	0.993	30.111
P3 (Vintage)	0.978	31.296

On another hand, P1 and P3 each has a lower embedding capacity in comparison to P2 because they are not able to utilize all rows for data embedding purposes. Recall that P1 relies on the carrier pixels in an image to embed data. Some images may have less or even no black pixels at all in some rows, thus not all the rows are embeddable. In the case of P3, it can only embed, on average, $\sim$490 bits of payload data per image (for the settings of $\omega = 10$) because some of the rows in the images are reserved for the artificial white frames and jagged patterns.

B. Image Quality

To quantify the distortion caused by data embedding, image quality is measured between the original image (i.e., image with photo effect only), and the 'processed' image (i.e., image with photo effect and embedded data). Specifically, the quality will be investigated in terms of SSIM (Structural Similarity Index Measure [29] and PSNR (Pixel Signal-to-Noise Ratio) [30]. Results are recorded in Table III. In general, SSIM are high for all 3 proposed methods (i.e., near to unity), which indicates that these data embedding methods are able to preserve the perceived quality of the image. Specifically, the luminance, contrast and structure properties are well maintained even after data embedding.

In terms of PSNR, the proposed methods are able to achieve around 30dB - 32dB, which is considerably low. It is an expected outcome because PSNR compares pixel-to-pixel and the proposed methods make drastic changes to the pixel values during data embedding. For instance, the carrier pixel values of $Ł_G$ in P1 are modified to *high* pixel values to carry payload bit. Likewise, for P3, the white frames are introduced near the

Fig. 8: Output images (with embedded data) generated by P1 (top row), P2 (middle row) and P3 (last row).

Fig. 9: Input images to be used by P1 (top row), P2 (middle row) and P3 (last row) for experiments

image borders, and the jagged patterns are also modified by changing all the relevant pixels to $\{255, 255, 255\}$.

Figure 8 shows the output images (with embedded data) produced by our prototypes by using original input images as shown in Figure 9. Although the PSNR values reveal some degradation in terms of pixel changes, the modification caused by data embedding is blending well with the image

TABLE IV: Average data extraction rate for each prototype after applying JPEG compression with various quality factors.

Prototype (Effect)	η [%] of various quality factors			
	60	70	80	90
P1 (Sketch)	82.06	82.46	82.77	85.12
P2 (Halftone)	99.98	99.98	99.99	99.98
P3 (Vintage)	91.11	96.50	99.44	99.92

from the subjective evaluation point of view. This is because our methods are designed to incorporate the data embedding processes into the photo effect algorithms.

C. Robustness against compression

Recall that the motivation of this work is to encourage users to protect their images before they upload them to any SNS platforms. However, most SNS platforms (including Facebook and Snapchat) compress all uploaded images to reduce the file size [31]. Following that, the embedded data can potentially be destroyed by the compression operation. Therefore, robustness against compression, i.e., the ability to extract the embedded data after G' is compressed, is greatly desired. Since most SNS platforms support the JPEG standard, which is one of the most widely adopted image compression standards [32], we evaluate our proposed methods against JPEG compression. Specifically, the robustness is measured by calculating the data extraction rate (referred to as η) by using the following equation:

$$\eta = \frac{\text{correctly extracted bits}}{\text{embedding capacity}} \times 100. \tag{3}$$

Here, JPEG compression is performed on the output image G' by using the quality factors of $60, 70, 80$, and 90.

Table IV records η the results for the proposed methods.Results suggest that, among the proposed methods, P2 has the highest robustness against JPEG compression because it is able to extract the payload bits correctly, with $\eta \sim 100\%$. This is because P2 operates on 8-bit grayscale images, which do not consists of redundant color information (in comparison to 24-bit images generated by P1 and P3). Hence, the output image generated in P2 is less affected by compression.

V. Discussions

A. Comparison with conventional methods

The proposed method exploits photo effect to hide data, which simplifies the photo-uploading process for actual applications. When the proposed data embedding methods are incorporated into general photo editing applications (e.g., Adobe Photoshop, GIMP, PixsArt), the benefits are threefolds, namely: (i) no additional steps / applications are needed to embed data because data embedding becomes part of the photo editing process; (ii) metadata can be added into a photo either by the photo editing application or by the owner herself for managerial purposes, and (iii) by using the proposed method, the pixels are not modified solely for data embedding purpose. Therefore, no extra and obvious distortion is introduced. To the best of our knowledge, there is no

similar work which integrates data embedding capability into the application of photo effects. Although we acknowledge that there are some work in data hiding as part of the image enhancement processes [33, 34], but they are not robust against JPEG compression.

B. Possible future improvements

In this work, the experiment results infer the general feasibility of our proposed photo effect-based data embedding methods. That is, the proposed algorithms are able to embed data in the host image by using effect attributes, and they are able to achieve high image quality and robustness against JPEG compression.

However, the embedding capacity is limited. In the proposed method, embedding capacity is limited to ensure high data extraction accuracy after image compression, i.e., a trade-off between capacity and robustness. This is based on the fact that all images uploaded to any SNS platforms are compressed prior to sharing. Hence, in comparison to high embedding capacity, achieving high robustness is of a higher priority in this preliminary study. In the current implementation, the highest achievable embedding capacity is 512 bits per image, but the embedding capacity of each algorithm can be further enhanced. For instance, each row can be divided into p partitions to carry p payload bits for both P1 and P2, instead of using all available spaces to carry only a single payload bit in each row. On the other hand, for P3, the jagged pattern can be further extended (e.g., include some number of pixels to create jagged patterns) to represent more payload bits. The aforementioned possible improvements will be explored as our future work.

VI. Conclusions

In this research, three novel data embedding methods based on photo effects are proposed. The proposed methods embed data during the generation of photo effect. To the best of our knowledge, they are the first of the kind, and they are robust against the legacy JPEG compression standard. Experiment results suggest the feasibility of the proposed methods for the purposes of data embedding.

As future work, we focus on improving the embedding capacity and identify innovative ways to integrate the join utilization of the proposed photo effect based data embedding methods.

References

[1] Domo Inc. Data never sleeps 7.0 infographic — domo. https://www.domo.com/learn/data-never-sleeps-7, 2019. Accessed on 15 August 2020.

[2] Domo Inc. Data never sleeps 8.0 infographic — domo. https://www.domo.com/learn/data-never-sleeps-8, 2020. Accessed on 15 August 2020.

[3] C. M. Cunningham and N. Brody. Social networking and impression management: Self-presentation in the digital age. 2012.

[4] Zizi Papacharissi. The presentation of self in virtual life: Characteristics of personal home pages. *Journalism Mass Communication Quarterly*, 79:643 – 660, 2002.

[5] Markus Jakobsson Filippo Menczer Tom N Jagatic, Nathaniel A Johnson. Social phishing. *Communications of the ACM*, 50:94–100, 2007.

[6] Zhiyong Zhang and Brij B. Gupta. Social media security and trustworthiness: Overview and new direction. *Future Generation Computer Systems*, 86:914–925, September 2018.

[7] Amanda G. Ciccatelli. Photo sharing on social media & copyright infringement: What you need to know - ipwatchdog.com — patents & patent law. https://www.ipwatchdog.com/2017/12/15/photo-sharingsocial-media-copyright-infringement/id=91022/, December 2017. Accessed on 08/15/2020.

[8] Hyuk-Jin Lee and Diane Neal. A new model for semantic photograph description combining basic levels and user-assigned descriptors. *Journal of Information Science*, 36(5):547–565, July 2010.

[9] Rachana C. Patil and Prof. S. R. Durugkar. Content based image re-ranking using indexing methods. *International Journal of Emerging Technology and Advanced Engineering*, 5:447–453, August 2015.

[10] Sahar Haddad, Gouenou Coatrieux, and Michel Cozic. A new joint watermarking-encryption-JPEG-LS compression method for a priori & a posteriori image protection. page 1688–1692, October 2018.

[11] Jutta Hammerle-Uhl, Christian Koidl, and Andreas Uhl. Multiple blind re-watermarking with quantisation-based embedding. September 2011.

[12] B. Ferreira, J. Rodrigues, J. Leitão, and H. Domingos. Privacy-preserving content-based image retrieval in the cloud. pages 11–20, 2015.

[13] R. van Schyndel, A. Tirkel, and C. Osborne. A digital watermark. 2:86,87,88,89,90, November 1994.

[14] K Thangadurai and G Sudha Devi. An analysis of lsb based image steganography techniques. pages 1–4, 2014.

[15] M.S. Sutaone and M.V. Khandare. Image based steganography using LSB insertion. 2008.

[16] Andysah Putera Utama Siahaan. Technique of hiding information in image using least significant bit. November 2018.

[17] Zhicheng Ni, Yun-Qing Shi, N. Ansari, and Wei Su. Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):354–362, March 2006.

[18] Zhibin Pan, Sen Hu, Xiaoxiao Ma, and Lingfei Wang. Reversible data hiding based on local histogram shifting with multilayer embedding. *Journal of Visual Communication and Image Representation*, 31:64–74, August 2015.

[19] C. Tseng, Y. Chiu, and Y. Chou. A histogram shifting-based reversible data hiding scheme using multi-pattern strategy. pages 125–128, 2015.

[20] Jun Tian. Reversible data embedding using a difference expansion. *IEEE transactions on circuits and systems for video technology*, 13(8):890–896, 2003.

[21] Zhicheng Ni, Yun-Qing Shi, Nirwan Ansari, and Wei Su. Reversible data hiding. *IEEE Transactions on circuits and systems for video technology*, 16(3):354–362, 2006.

[22] X. Li, B. Yang, and T. Zeng. Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. *IEEE Transactions on Image Processing*, 20(12):3524–3533, 2011.

[23] Lin Bai. converting natural image to sketch style using vision hdl - file exchange - matlab central. https://la.mathworks.com/matlabcentral/fileexchange/72215-converting-natural-image-to-sketch-style-using-vision-hdl?stid=profcontriblnk, July 2019. Accessed on 08/15/2020.

[24] GeeksforGeeks. Matlab - image edge detection using sobel operator from scratch - geeksforgeeks. https://www.geeksforgeeks.org/matlab-image-edge-detection-using-sobel-operator-from-scratch/, May 2020. Accessed on 10/19/2020.

[25] Fan Zhang, Zhenzhen Li, Xingxing Qu, and Xinhong Zhang. Inverse halftoning algorithm based on SLIC superpixels and DBSCAN clustering. pages 466–471, 2018.

[26] Praveen Settipalli. Error diffusion algorithm - file exchange - matlab central. February 2005. Accessed on 08/15/2020.

[27] Canvas - all white background. http://www.allwhitebackground.com/canvas-background.html/download/19993. Accessed on 08/15/2020.

[28] David Martin and Charless Fowlkes. The berkeley segmentation dataset and benchmark. https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/, June 2007. Accessed on 08/15/2020.

[29] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[30] Ratnakirti Roy and Suvamoy Changder. Quality evaluation of image steganography techniques: A heuristics based approach. *International Journal of Security and Its Applications*, 10(4):179–196, April 2016.

[31] Vijay Koushik. Data compression: How social media changes quality of your photos. https://svijaykoushik.github.io/blog/2017/05/10/how-social-media-changes-quality-of-your-photos/, May 2017. Accessed on 10/19/2020.

[32] T. Chuman, K. Iida, and H. Kiya. Image manipulation on social media for encryption-then-compression systems. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 858–863, 2017.

[33] H. Wu, W. Mai, S. Meng, Y. Cheung, and S. Tang. Reversible data hiding with image contrast enhancement based on two-dimensional histogram modification. *IEEE Access*, 7:83332–83342, 2019.

[34] Simying Ong and KokSheik Wong. Information hiding in image enhancement. In *2020 IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, UAE, Oct. 25-28, 2020*, pages To–appear. IEEE, 2020.