Reference Based Image Encoding

S.D. Yamini Devi, Raja Santhanakumar and K.R. Ramakrishnan

Abstract This paper describes a scheme to encode an image using another *reference image* in such a way that an end user can retrieve the encoded image *only* with the reference image. Such encoding schemes could have a potential application in secure image sharing. The proposed scheme is simple and similar to fractal encoding; and a key feature is it simultaneously performs compression and encryption. The encoding process is further speeded up using PatchMatch. The performance in terms of encoding time and PSNR is examined for the different encoding methods.

Keywords Fractal encoding · Reference-based encoding · PatchMatch

1 Introduction

Images can be securely shared using well known techniques in steganography and cryptography [2, 7, 8]. Since raw image data are huge in size, in general, they have to be compressed before sharing. A natural way to share images securely would be to compress them first, followed by scrambling or encrypting. On the other end, a user would need to decrypt and decode in order to retrieve the encoded image. Essentially, in the above scheme, the image encoding and decoding is a 2-step process. In this paper, we propose an image encoding scheme, similar to fractal encoding [3], which simultaneously achieves compression and encryption in a single step.

Few earlier works have attempted to do secure image sharing using fractal codes in two separate steps: compression and encryption. Shiguo lian [5] encrypts some of the fractal parameters during fractal encoding to produce the encrypted and encoded

S.D.Y. Devi (✉) · K.R. Ramakrishnan (✉)
Department of Electrical Engineering, Indian Institute of Science, Bangalore, India
e-mail: yamini@ee.iisc.ernet.in

K.R. Ramakrishnan
e-mail: krr@ee.iisc.ernet.in

R. Santhanakumar
Airbus Group Innovations, Airbus Group India, Bangalore, India

B. Raman et al. (eds.), *Proceedings of International Conference on Computer Vision and Image Processing*, Advances in Intelligent Systems and Computing 460,
DOI 10.1007/978-981-10-2107-7_13

data. In [6], Jian Lock et al. by modifying Li et al. [4], *actually* perform compression and symmetric key encryption in a 2-step process wherein fractal codes of an image is multiplied with a equal sized Mandelbrot image generated through an equation. Before the multiplication, the fractal code and Mandelbrot image matrices are permuted. The product matrix is transmitted along with some few other parameters. Decryption is done through inverse permutation and some matrix manipulations.

In contrast to the above approaches, we propose a single step reference-based image encoding wherein an image is encoded using another "reference" image in such a way that decoding is possible only by the receiver or a user having the same reference image. In other words, the reference image serves like a key for secure image sharing. To begin with in Sect. 2, we give a brief overview of fractal encoding and decoding, followed by a description of our proposed reference-based approach and highlight important differences between the two encoding methods. In Sect. 3 we describe how the PatchMatch algorithm [1] is used in order to reduce encoding time. Experiments and results are provided in Sect. 4 along with some insights about the functioning of PatchMatch in encoding. Finally, Sect. 5 concludes the paper.

2 Reference-Image Based Image Encoding

Since the proposed reference-based encoding is very similar to fractal encoding, for the sake of completeness, in what follows we provide a brief description of fractal image encoding and decoding. For more elaborate details on the theoretical and implementation aspects, please see [3]. Let f be the image to be encoded. Fractal encoding is essentially an inverse problem where an Iterated Function System (**IFS**) W is sought such that the fixed point of W is the image to be encoded f. An approximate solution to this problem is to partition f into M disjoint sub-blocks ("range blocks") f_i such that $f = \cup_{i=1}^{M} f_i$; for each f_i, we find a block of twice the size ("domain block") elsewhere in the same image which best matches to f_i after resizing, spatial transformation (isometry) and intensity modifications (brightness and contrast). The mapping between f_i and its corresponding matching block is a contractive mapping w_i. This process of searching self-similar block can be seen as having two copies of the original image; one is called the range image and the other is called the domain image. The range image contains non-overlapping range blocks f_i, and the domain image contains overlapping domain blocks of twice the size D_j. The relationship between f_i and the best matching block in D_j is represented through the contractive mapping w_i. It can be shown that $W = \cup_{i=1}^{M} w_i$ is also a contractive mapping, and in practice, the fixed point of W is very close to the original f which allows to encode the image f as the parameters of $w_i, i = 1, \ldots, M$. Unlike the encoding, fractal decoding is a much simpler process. Since the encoder is based on the concept of contractive mappings, *starting from any initial image*, application of the contractive mappings $w_i, i = 1, \ldots, M$ repeatedly will converge to the same fixed point which will be an approximation to the encoded image. Image compression is achieved because the image is entirely encoded in terms of the parameters of the contractive mappings w_i, and the number of the parameters is the same $\forall w_i, i = 1, \ldots, M$. Note that an interesting consequence of fractal encoding is that the size of the encoded data depends only on the number of sub-blocks M; therefore all images with the same number of partitioned sub-blocks will have the encoded data of equal size.

In contrast to the fractal encoding wherein the original image can be recovered from the encoded data *starting from a arbitrary initial image*, our proposed scheme must encode the given image in such a way that the original image (or an approximation) can be recovered only from a *specific reference image*.

2.1 Proposed Encoding Scheme

Unlike fractal encoding described above where the same image is used as domain and range images, in our proposed scheme, we use the reference image as the domain image, and the original image as the range image. For encoding the non-overlapping range blocks, the best matching block among the overlapping domain blocks of equal size in the domain image (reference) is found. By the above two simple modifications, the proposed encoding method is sufficient to meet our objective. In other words, for decoding, the same domain image used for encoding is *necessary as a key* to recover the original (range) image. To keep the reference-based encoder simple, while searching for a matching domain block, no spatial transformations are applied to the domain blocks. Given a range block f_i and a candidate domain block we find only the optimal “contrast” and “brightness” which will bring the domain block and range block “closer”. Equations for computing optimal values of contrast and brightness are given in [3], p. 21. Although the encoded data has compression properties similar to a fractal encoder, the encoding quality depends on finding suitable domain blocks in the reference image for the range blocks. The block diagrams showing a comparison between reference-based and fractal encoding are shown in Fig. 1, and the pseudo-code for the reference-based encoding is given in Algorithm 1.

Algorithm 1: Reference-based image encoding

```
Data: Range image $I$, Reference image $I_1$
Result: ImageCodes
for $i = 1 : M$ (No: of Range blocks) do
    for $k = 1 : N$ (No: of Domain blocks) do
        From $I_1(D_k)$ pick a domain block
        Find $s,o$ (intensity modifying parameters contrast and brightness)
        $[s, o] \leftarrow \min_{s,o} ||f_i - T(D_k; s, o)||$
    end
    Best matching domain block $D_k = \arg\min_{D_k} \left\{ \min_{s,o} ||f_i - (D_k; s, o)|| \right\}$
end
```

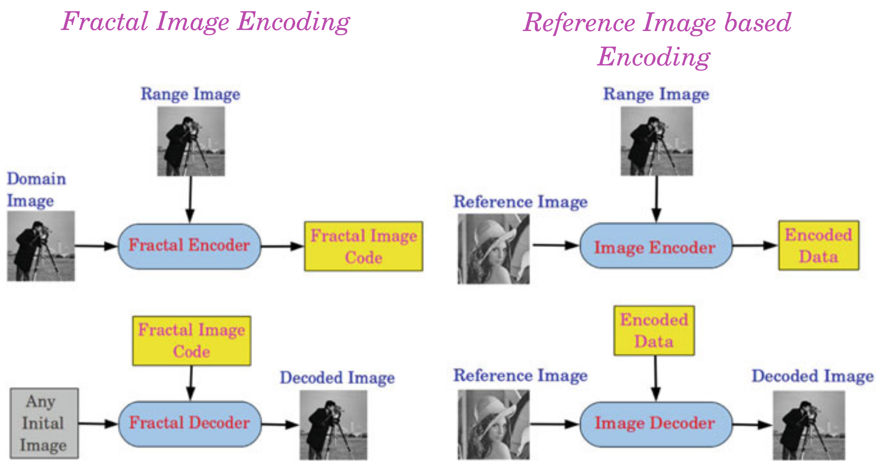

Fig. 1 Fractal versus reference-based image encoding

Given the reference-based image codes, *decoding involves a single step* wherein the mappings w_i are applied on the reference image used during encoding. Note that unlike fractal decoding, *No* iterative process is involved in the our proposed method. In a fractal encoder, the domain block is twice the size of the range block which is important to make the mapping contractive because contractivity is a necessary requirement for the nature of fractal decoding. The pseudo-code for the reference-based decoding is given in Algorithm 2.

Algorithm 2: Reference-based image decoding

```
Data: ImageCodes: $\{(p(D_i), s_i, o_i)\}_{i=1}^{M}$
Result: Decoded image
Initialization: Domain image $I_1$;
for $i = 1 : M$ (No: of Range blocks) do
    From $p(D_i)$ crop domain block: $D_i = I_1(p(D_i))$
    intensity modification with $(s_i, o_i)$
    DecodedImage($p(f_i)$) $\leftarrow T(D_i; s_i, o_i)$
end
```

Note $\cup_{i=1}^{M} f_i$ covers the entire image

The search process in the above encoding algorithm compares a range block with every candidate domain block to find the best match. Since this makes the encoding time very long, in the next section, we propose a way to reduce the search time by finding an *approximate* matching block using PatchMatch.

3 PatchMatch Reference Encoder

PatchMatch [1] is a iterative randomized algorithm for finding the best match between image patches across two different images A and B. Initially, for a patch centred at (x, y) in image A, a random matching patch (nearest neighbor) is assigned at a offset $f(x, y)$ or $\mathbf{v}$ in image B. Let $D(\mathbf{v})$ denote the error distance between the patches at (x, y) in A and at $((x, y) + \mathbf{v})$ in B. In every iteration, the algorithm refines the nearest neighbor for every patch of image A in two phases: (1) propagation and (2) random search.

Propagation Phase: Assuming the neighboring offsets of (x, y) are likely to be the same, the nearest neighbor for a patch at (x, y) is improved by "propagating" the known offsets of neighbors of (x, y) only if the patch distance error $D(\mathbf{v})$ improves. Propagation of these offsets use different neighbors during odd and even iterations: the updated offset $\mathbf{v}$ in odd iterations is, $\mathbf{v} = \arg\min\{D(f(x, y)), D(f(x-1, y)), D(f(x, y-1))\}$, and in even iterations, $\mathbf{v} = \arg\min\{D(f(x, y)), D(f(x+1, y)), D(f(x, y+1))\}$.

Random Search: If $\mathbf{v}_0$ is the updated offset after the propagation phase, further improvement is done by constructing a window around $\mathbf{v}_0$ and searching through a sequence of offsets $\{\mathbf{u}_i\}$ at an exponentially decreasing distance from $\mathbf{v}_0$. Let w be the maximum search distance around $\mathbf{v}_0$, and $\mathbf{R}_i$, a sequence of uniformly distributed random points in $[-1, 1] \times [-1, 1]$. The sequence of random offsets is given by $\mathbf{u}_i = \mathbf{v}_0 + w\alpha^i \mathbf{R}_i$, where $\alpha = 1/2$, and $i = 1, 2, \ldots$. The number of random offsets searched is determined by w with the condition that the last search radius $w\alpha^i$ is less than 1 pixel. In this phase, the updated offset is $\mathbf{v} = \arg\min\{D(\mathbf{v}_0), D(\mathbf{u}_1), D(\mathbf{u}_2), \ldots\}$; in other words, the offset is only updated if the patch distance error reduces.

In order to reduce the search time for finding a matching domain block for a range block, PatchMatch is modified in the following ways and incorporated in the reference encoder.

- As an initial step, every range block f_i in the original image (A) is randomly assigned a domain block in domain image (reference or B). An image code is generated for this range block; consisting of the domain block position, range block position, scale and offset. In contrast to PatchMatch where for all patches (overlapping) in A, matching patches in image B are found, in our modification, matching domain blocks need to be found only for range blocks which are non-overlapping.
- Block matching in PatchMatch involves computing Euclidean distance between blocks and finding the block in B with the least Euclidean distance. We, however, change the distance error by finding two parameters for intensity modification, contrast (scale) and brightness (offset), which minimizes the matching error. The expressions for optimal brightness and contrast are the same as in the reference coder (and fractal encoder [3]).

Apart from the above two differences, block matching is similar to PatchMatch, including two phases of propagation and random search to improve the matching domain block for a range block iteratively. The pseudo-code for the PatchMatch reference encoder is given in Algorithm 3.

Algorithm 3: Reference-based encoding with PatchMatch

```
Data: Given Image A and B(Reference Image)
Result: Decoded image
Initialization: Randomly assign $D_i$ to $f_i$ and compute $[s_i, o_i]$
for $k = 1 : MaxIterations$ do
    for $k = 1 : M$ (No: of Range blocks) do
        Propagation: update if neighbors are better $(D_i, s_i, o_i)$ Random Search: search
        around $v_o = D_i$ for random matching domain blocks
        Update if candidates are better based on RMS error
    end
end
```

The decoding process is the same as described in Sect. 2.1 wherein the original image is recovered in a single step by applying the image codes on the same domain image that was used for encoding.

4 Experiments and Results

Reference Encoding: To illustrate a typical result of the reference-based encoding, we have taken the "cameraman" as the range image (Fig. 2a), and "lena" as the domain (reference) image (Fig. 2b). The range block size is chosen to be 8×8, and for finding a matching domain block an exhaustive process is used. The decoded result shown in Fig. 2c indicates a fair reconstruction of the original image.

PatchMatch Encoding: The "lena" image (Fig. 3a) is PatchMatch encoded with range blocks of size 8×8, and using "cameraman" as the reference. The decoded result in Fig. 3b is noticeably different in the boundaries when compared with the original. The same information is reflected in Fig. 3c which shows the matching distance of all the range blocks.

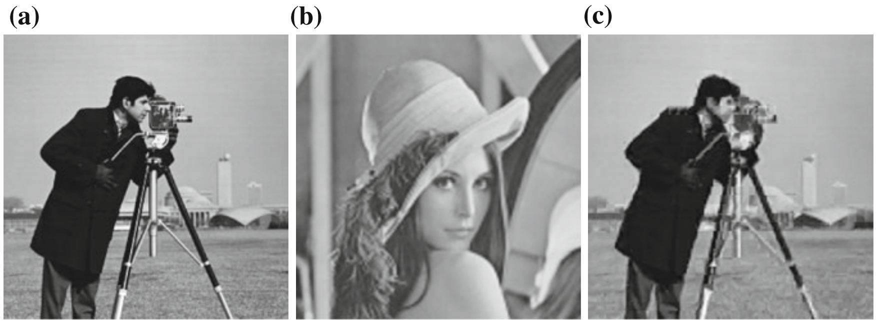

Fig. 2 Reference-based image encoding: **a** range image, **b** reference (domain) image, **c** decoded image

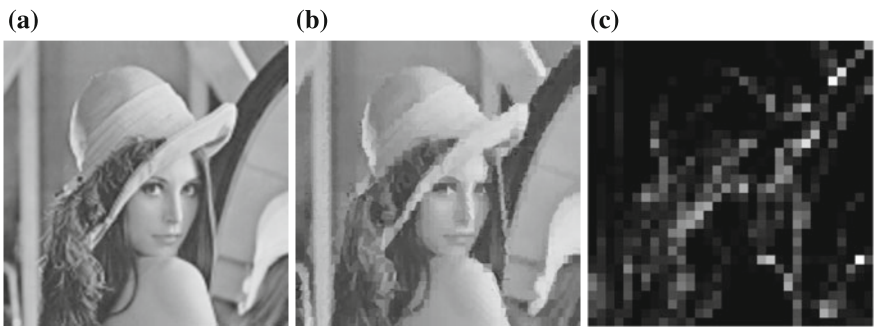

Fig. 3 PatchMatch: **a** original image, **b** decoded image, **c** matching distance of range blocks

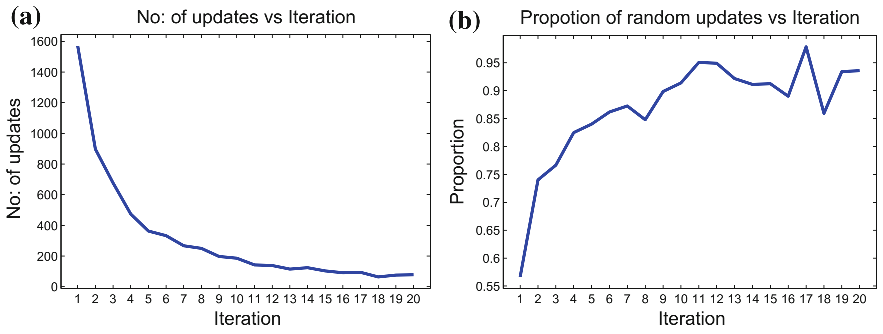

Fig. 4 **a** No: of updates versus iteration, **b** random update versus iteration

To better understand the functioning of PatchMatch in the encoding, we plot two graphs: (1) No: of updates vs iteration in Fig. 4a, and (2) No: of random updates versus iteration in Fig. 4b. As mentioned earlier, in every iteration of PatchMatch, the nearest neighbor (matching domain block) for a range block gets refined either by a spatial neighbor or from a random search. An interesting observation in Fig. 4a, b is that beyond 15 iterations there are significantly less updates from the neighborhood information, and relatively more updates take place in the random search.

The PatchMatch encoder is tested on a synthetic image having many uniform regions (Fig. 5a) with range blocks of size 4×4. The decoded result (Fig. 5b) and the block matching distance (Fig. 5c) show that the range blocks with large matching distance are on the region boundaries. Matching domain blocks for the uniform regions are more likely to be found in first few iterations of the propagation phase (Fig. 6a), but after certain number of iterations most of the updates come from the random search phase (Fig. 6b) for finding matches for range blocks on image boundaries. More study is required to determine the maximum number of iterations for efficiently finding matching blocks.

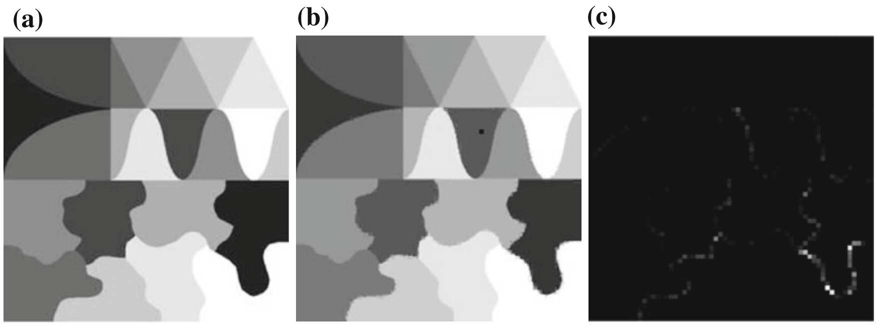

Fig. 5 PatchMatch encoding of an image with many uniform regions: **a** original image, **b** decoded image, **c** matching distance

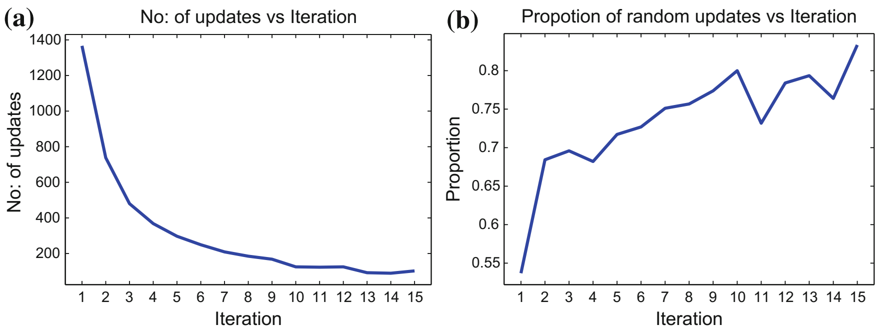

Fig. 6 **a** No of updates versus iteration for image with uniform region, **b** random updates versus iteration

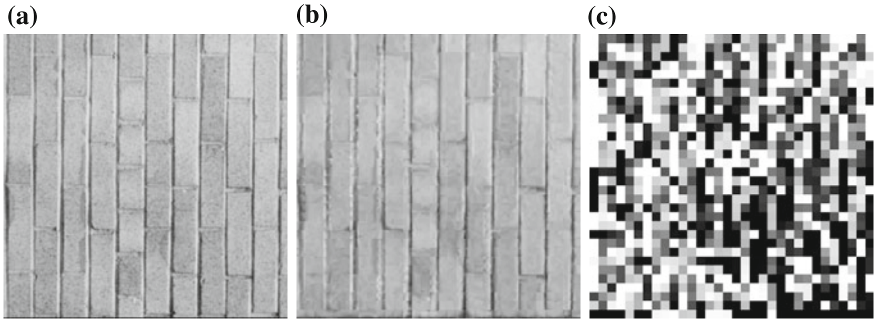

Fig. 7 **a** Original image, **b** decoding using reference image, **c** decoding without the same reference image

For the texture image in Fig. 7a the decoded result using "lena" as reference is shown in Fig. 7b. When an image with all pixels as 255 is used as reference while decoding, the resulting output shown in Fig. 7c indicates that the decoder fails if the appropriate reference is not used.

The tables below summarize the key differences between three types of image encoders: (1) fractal, (2) reference image-based and (3) PatchMatch-based. Table 1 gives the major differences between the original PatchMatch algorithm and the reference-based encoder, while Table 2 highlights some of the key differences between fractal encoding and PatchMatch encoder. Table 3 compares the performance of the above three techniques in terms of PSNR and the encoding time. Fractal encoding results in the best PSNR, but with the highest encoding time compared to reference-based and PatchMatch encoders. Since the reference-based image encoder does not use spatial transforms while searching for matching blocks, the encoding time is less in comparison to fractal encoder. For the "lena" image (Fig. 8a), the decoded result from the three different encoders are shown in Fig. 8b–d.

Table 1 Comparison of PatchMatch algorithm and reference-based encoding

PatchMatch	Reference-based encoder
Best match needed at every (x, y)	Best match needed only at $f_i, i = 1, 2, \dots M$
Random search and neighbor propagation	Exhaustive search for each f_i
Has no intensity modification	Block matching finds optimal contrast and brightness
Convergence of nearest neighbor is fast	Search is very time consuming

Table 2 Comparison of fractal image encoding and PatchMatch encoder

Fractal encoder	Patchmatch encoder
Range and domain images are the same	Range and domain images are different
Exhaustive search for block matching	Randomized search for block matching
Search time depends on no: of domain blocks	Search time depends on image size
No specific image needed for image decoding	Requires same reference image (domain) for decoding

Table 3 Performance comparison of encoding techniques on "lena" image (256×256) with 4×4 range blocks

	Fractal	Reference	PatchMatch
Iteration	NA	NA	15
PSNR	41.75	38.59	34.90
Time (sec)	2560	1350	963

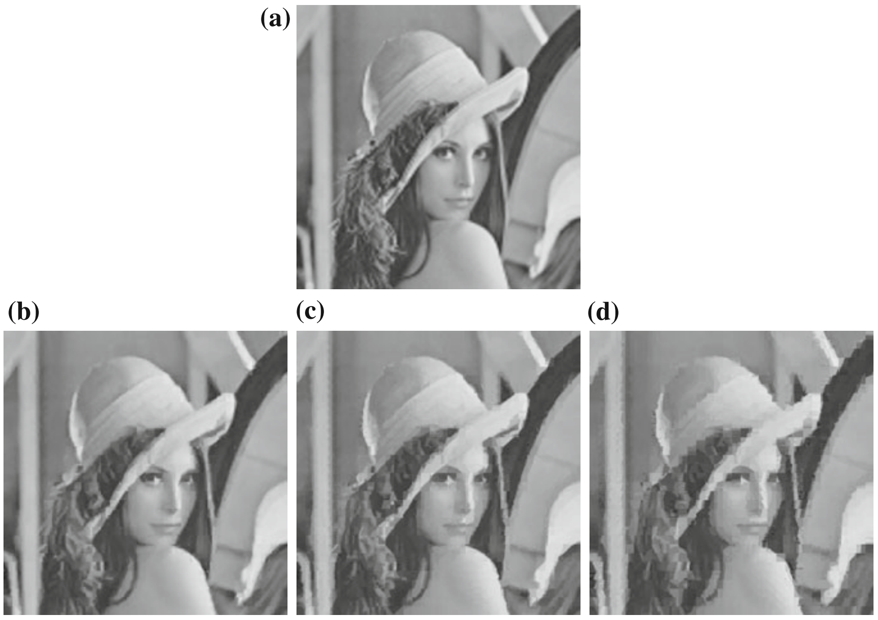

Fig. 8 Comparison of different encoding techniques: **a** original image, **b** fractal, **c** reference-based, **d** PatchMatch-based

5 Conclusions

In this paper we have proposed methods to share images in a secure way using a single step encoding process which combines compression and encryption. Results show that the user will be able to decode a meaningful image only when the "reference image" is given. The proposed work, though not similar to fractal encoding, adopts some of the key features of fractal encoding, a lossy compression technique. PatchMatch has been leveraged in order to speed up the encoding process. However, the results obtained are inferior in comparison to the reference-based encoder. One future direction to improve the PSNR could be selecting an "appropriate" reference image from an image set.

References

1. Connelly Barnes, Ei Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structured Image Editing. pages 24:1–24:11. ACM, 2009.
2. Chin-Chen Chang, Min-Shian Hwang, and Tung-Shou Chen. A New Encryption Algorithm for Image Cryptosystems. *Journal of Systems and Software*, 58(2):83–91, 2001.
3. Yuval Fisher. *Fractal Image Compression: Theory and Application*. Springer Verlag, 1995.

4. Xiaobo Li, Jason Knipe, and Howard Cheng. Image Compression and Encryption Using Tree Structures. *Pattern Recognition Letters*, 18(11):1253–1259, 1997.
5. Shiguo Lian. Secure fractal image coding. *CoRR*, abs/0711.3500, 2007.
6. A. J. J. Lock, Chong Hooi Loh, S.H. Juhari, and A Samsudin. Compression-Encryption Based on Fractal Geometric. In *Computer Research and Development, 2010 Second International Conference on*, pages 213–217, May 2010.
7. Debasis Mazumdar, Apurba Das, and Sankar K Pal. MRF Based LSB Steganalysis: A New Measure of Steganography Capacity. In *Pattern Recognition and Machine Intelligence*, volume 5909, pages 420–425. Springer Berlin Heidelberg, 2009.
8. Ren Rosenbaum and Heidrun Schumann. A Steganographic Framework for Reference Colour Based Encoding and Cover Image Selection. In *Information Security*, volume 1975, pages 30–43. Springer Berlin Heidelberg, 2000.