Image compression methodology based on fuzzy transform using block similarity

Petr Hurtik[1] **Irina Perfilieva**[1]

[1] petr.hurtik@osu.cz, irina.perfilieva@osu.cz

Abstract

The aim of the work is to continue in improvement of the image compression algorithm based on the F-transform. The image is decomposed into blocks and characterized by the F-transform components. The latter constitutes a simple lossy compression. For better quality of reconstructed images, we compress certain areas (neighborhoods of edges) non-lossy. The proposed approach is based on establishing similarity between various blocks and making compression of only one representative. Last but not least, the proposed compression algorithm is supplied with smart technique of joining adjacent blocks.

Keywords: Image compression, F-transform, Image similarity

1. Introduction

By image compression we mean a reduction in size of the image with the purpose to save space and by this, a transmission time of data. Digital images are usually identified with their two-dimensional intensity functions which, being measured in the interval $[0, 1]$, can be represented by fuzzy relations. Therefore, in the literature on fuzzy sets and their applications, a continuously growing interest to the problems of image compression was expected. Below, we will give a short overview of main ideas that influenced a progress in image compression on the basis of fuzzy sets.

A pioneering publication of Lotfi A. Zadeh discussed the issue of data summarization and information granularity. It has been noticed that a $\max - \min$ - composition with a fuzzy relation works as a summarization/compression tool. Then in a series of papers, the idea to associate image compression with the theory of fuzzy relation equations was intensively investigated. The correspondence between a quality of reconstruction and a t-norm in a generalized $\max - t$ - composition with a fuzzy relation was analyzed. A new idea which influences a further progress in fuzzy based image compression came with the notion of F-transform [4]. In [8], [5], it has been shown that the F-transform based image compression is better than the best possible fuzzy relation based one. However, the former was still worse than the JPEG technique. A certain improvement of the F-transform based image compression was announced in [6].

2. F-transform

The F-transform [4] method was published in 2001. By the time, F-transform succeed in many various field such as image compression, image resize, edge detection, time series, signal filtering and many others.

2.1. Used F-transform type

The direct and inverse F-transform of a function of two (and more) variables is a direct generalization of the case of one variable. We introduce the discrete version only, because it is used in our applications below. Let us refer to [4] for more details.

Suppose that the universe is a rectangle $[a, b] \times [c, d] \subseteq \mathbb{R} \times \mathbb{R}$ and that $x_1 < \ldots < x_n$ are fixed nodes of $[a, b]$ and $y_1 < \ldots < y_m$ are fixed nodes of $[c, d]$ such that $x_1 = a$, $x_n = b$, $y_1 = c$, $y_m = d$ and $n, m \geq 2$. Assume that $A_1, \ldots, A_n$ are basic functions that form a generalized fuzzy partition of $[a, b]$ and $B_1, \ldots, B_m$ are basic functions that form a generalized fuzzy partition of $[c, d]$. Then, the rectangle $[a, b] \times [c, d]$ is partitioned into fuzzy sets $A_k \times B_l$ with the membership functions $(A_k \times B_l)(x, y) = A_k(x)B_l(y)$, $k = 1, \ldots, n$, $l = 1, \ldots, m$.

In the discrete case, an original function f is assumed to be known only at points $(p_i, q_j) \in [a, b] \times [c, d]$, where $i = 1, \ldots, N$ and $j = 1, \ldots, M$. In this case, the (discrete) F-transform of f can be introduced in a manner analogous to the case of a function of one variable. This case is important for applications of the F-transform to image processing.

The discrete F-transform $F[f]$ of f is given by the following matrix of components:

$$F[f] = \begin{pmatrix} F_{11} & \cdots & F_{1m} \\ \vdots & \vdots & \vdots \\ F_{n1} & \cdots & F_{nm} \end{pmatrix} \tag{1}$$

where for all $k = 1, \ldots, n$, $l = 1, \ldots, m$,

$$F_{kl} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{M} f(p_i, q_j) A_k(p_i) B_l(q_j)}{\sum_{j=1}^{M} \sum_{i=1}^{N} A_k(p_i) B_l(q_j)}.$$

The inverse F-transform of a discrete function f of two variables is defined as follows.

$$\hat{f}(p_i, q_j) = \frac{\sum_{k=1}^{n} \sum_{l=1}^{m} F_{kl} A_k(p_i) B_l(q_j)}{\sum_{k=1}^{n} \sum_{l=1}^{m} A_k(p_i) B_l(q_j)} \tag{2}$$

3. Image similarity measures

Let image be given by (identified with) an image function $f : D_{N,M} \to \{0, 1, \ldots, 255\}$ where $D_{N,M}$ is a finite (rectangular) domain given formally by $D_{N,M} = \{1, 2, \ldots, N\} \times \{1, 2, \ldots, M\}$. We say that N (resp. M) is the *width* (resp. *height*) of the image. The set of images on $D_{N,M}$ will be denoted $\mathcal{I}_{N,M}$, and the set of all images on finite rectangular domains will be denoted by $\mathcal{I}$.

Image similarity is an important notion that is used in the below proposed compression algorithm. Informally speaking, an image similarity measures how the image functions are close to each other. In the image processing, the following measures are often used for the estimation of closeness: MSE, PEN, SSIM[2]. Let us remark that the first two are based on the Euclidean distance.

We assume that an image similarity can be characterized with respect to the following unary operations over images f in $\mathcal{I}_{N,M}$:

- *Rotation* r of f over the origin by α.
- *Resizing* t of f by a ratio ρ: if $\rho < 1$ then t is called *reduction*, and if $1 < \rho < \infty$ then t is called *enlargement* (magnification) .
- *Negation* $\neg$ of f where $(\forall x, y) : \neg f(x, y) = 255 - f(x, y)$.

Let us formally introduce the image $(*)$-similarity $s : \mathcal{I} \times \mathcal{I} \to [0, 1]$ (where $*$ is a t-norm) as a mapping which characterizes closeness of two images (not necessarily on the same domain) in such a way that the following properties are fulfilled for all $f_1, f_2, f_3 \in \mathcal{I}$:

S1. $s(f_1, f_1) = 1$,

S2. $s(f_1, f_2) = s(f_2, f_1)$,

S3. $s(f_1, f_2) * s(f_2, f_3) \leq s(f_1, f_3)$,

S4. $s(f_1, t(f_1)) \neq 0$.

S5. $s(f_1, r(f_1)) \neq 0$.

The first three properties are standard. The property **S4** expresses that the similarity can be measured even if images f_1 and $t(f_1)$ have different sizes. The following proposition [1] shows a relationship between an arbitrary pseudo-metrics and a $(*)$-similarity in the case when the t-norm $*$ has a continuous additive generator.

Proposition 1 *Let X be a universe of discourse and $*$ a continuous Archemedean t-norm with the continuous additive generator t. Let moreover, d be a pseudometric on X. Then the mapping $E_d : X \times X \to [0, 1]$, given by $E_d = t^{(-1)} \circ d$ is a (*)-similarity on X.*

It is clear that if a pseudometric and similarity are connected as described in Proposition 1, they can be interchanged in estimation of closeness. Moreover, it turned out that a black car is more similar to the same car in white color than to a table.

3.1. Proposed image similarity measure

We propose to measure similarity with the help of F-transform components computed hierarchically on various levels of discretization of an original image. The lowest (first) level is comprised by the F-transform components of an original image f and corresponds to the discretization given by the respective fuzzy partition of the domain. This first level $F^{(1)}[f]$ is given by the F-transform of f so that

$$F^{(1)}[f] = F[f] = (F_{11}, ..., F_{nm}), \tag{3}$$

where the vector of the F-transform components $(F_{11}, ..., F_{nm})$ is a linear representation of the matrix (1). This first level of components serves as a new image for the F-transform components of the second level and so on. For a higher level ℓ we propose the following recursive formula:

$$F^{(\ell)}[f] = F[F^{(\ell-1)}] = (F_{11}^{(\ell-1)}, ..., F_{n_{(\ell-1)} m_{(\ell-1)}}^{(\ell-1)}). \tag{4}$$

The top (last) level $F^{(t)}[f]$ consists of only one final component F^{fin}.

The F-transform based similarity S of two image functions $f, g \in \mathcal{I}$ is proposed to be as follows:

$$S(f, g) = 1 - \exp\left(-\frac{|F^{fin} - G^{fin}| + \sum_{k=1}^{n} |F_{kl} - G_{kl}|}{nm + 1}\right) \tag{5}$$

where F^{fin}, G^{fin} are the top F-transform components of f and g, and F_{kl}, G_{kl}, $k = 1, \ldots, n$, $l = 1, \ldots, m$ are the first level F-transform components of f and g, respectively.

Let us justify that the measure S fulfills the above given properties **S1 - S5** where the property **S3** is taken with respect to the product t-norm. We remind that this t-norm has the function $\exp(-x)$ as an additive generator.

At first, we will notice that the following part of the expression in the right-hand side of (5)

$$(|F^{fin} - G^{fin}| + \sum_{k=1}^{n} |F_{kl} - G_{kl}|)/(nm + 1)$$

represents a distance between two $(nm + 1)$-dimensional vectors $\tilde{F}[f] = (F^{fin}, F_{11}, \ldots, F_{nm})$ and $\tilde{G}[g] = (G^{fin}, G_{11}, \ldots, G_{nm})$. At the same time, it represents a pseudo-distance between respective functions f and g. Therefore, by Proposition 1, the whole expression in the right-hand side of (5) represents a $(\cdot)$-similarity ($\cdot$ is the notation of product) of functions f and g.

At second, the property **S4** follows from the fact that the measure S requires the same number of basic functions in partitions of corresponding domains of functions f and g. The domains itself may be different.

4. Similarity based compression using F-transform with dynamic area decomposition

The proposed algorithm is based on the previous work [6] improved by back-joining partition of images and by computing similarity of blocks. The proposed algorithm will be called DSFTR.

Image compression means a reduction in size of the image. Below, we will refer to f as to an intensity function or to f as an image. By compression we mean a certain transformation of f which results in a new image function f' defined on $[1, N'] \times [1, M']$ where $N' < N, M' < M$. A compression is characterized by its ratio CR which is equal to $N'M'/NM$.

We will be focused on the following two problems:

- reduce size of compressed image,
- obtain decompressed image most similar to original one.

We propose a compression algorithm which is based on the discrete F-transform in combination with memorizing essential details (e.g., edges) and similarity relationships between blocks. This algorithm consists of the following steps:

C1. Search for essential details and store them nonlossy.

C2. Evaluate range of intensity over an image block and make a decision regarding further partition of this block.

C3. Choose similarity and find similar blocks; memorize a representative of every group of similar blocks.

C4. Combining similar and adjoining blocks into one group.

C5. Apply the F-transform to representatives of groups of similar blocks and memorize components.

Steps C1 - C4 are described detailed in sections 4.1 - 4.4, see them for explanation.

Let us make a short overview of some contemporary technique that are used for compression. The idea of partition of an image area into blocks according to respective ranges of the intensity function is taken from png graphics format. The decomposition techniques is called quad-tree [7] - an image block is recursively divided into four smaller subblocks.

There are methods based on lossy compression. The lossy compression can be represented by some type of transform, such as discrete cosine transform, or Burrows-Wheeler.

In our approach, we combine both lossy and nonlossy compression - essential areas are stored by non-lossy format and representative blocks by lossy F-transform. We propose decompression of an image after compression. A decompression of a compressed by the algorithm DSFTR image is proposed to be performed by the respective inverse F-transform.

4.1. Essential areas for image compression

Inputs: $N \times M$ image f, threshold T delimiting number of essentials pixels
Output: Set of essentials pixels to save nonlossy.

Let g be descriptor of pixel essential. The g is a two dimensional function such that $g : [1, N] \times [1, M] \to R$ where the essential characterizes changes in values of intensity of neighborhood pixels. The neighborhood is determined as a mask of 3×3 pixels centered around a pixel of essential. The intensity changes can be determined by existing algorithms, such as Sobel, Prewitt etc. We propose an algorithm which measures changes as

$$g(E) = max_{(x,y \in E)} f(x, y) - min_{(x,y \in E)} f(x, y), \quad (6)$$

where E is a block (area) of an image. It is called max-min operator.

The block with high values of g is not a subject of compression. Due to this fact, a sharpness of a reconstructed image is as good as in the original one. The proposed approach is sensitive to noise, more than if partial derivatives are computed by e.g., Sobel operators. In order to reduce that kind of sensitivity, we propose to use a dynamic threshold T for selecting high values of the function g. Due to a space limitation, we will skip a detailed description of choosing T.

4.2. Image decomposition into blocks

Inputs: $N \times M$ image f, minimal size of block Z, maximal intensity change inside blocks D
Output: Quad-tree structure consisting of ℓ level of blocks

Image compression algorithm is usually applied to smaller image blocks. The main problem is how to determine a size of this block. For instance, if we have a large block of one color, with a small detail of different intensity, we have two options: we can compress it as one block, but the detail will be lost. Or we can divide the block into smaller sub-blocks in order to keep information about that small detail. In the second case, we have to store many small subblocks of the same intensity. We propose to solve this problem by using the F-transform with a nonuniform partition.

Each block E is characterized by its:

- reference to another block $r(E)$,
- width of the block $w(E)$,
- height of the block $h(E)$,
- x-position of top left corner $x(E)$,
- y-position of top left corner $y(E)$.

At the beginning of the algorithm we set $w(E) = N; h(E) = M$.

One of the decomposition criteria is a range of changes D. For measure intensity change we can use

max-min operator (6). If $g(E) \leq D$, $D \in [0, 255]$ we choose the respective block E as an element of the partition of the F-transform. Otherwise, we divide block E into four symmetrical sub-blocks and continue recursively. Threshold D is determined by a user and affect compressed image quality. For general purpose, the result with the best ratio of quality and compression ratio is obtained for $D = 20$.

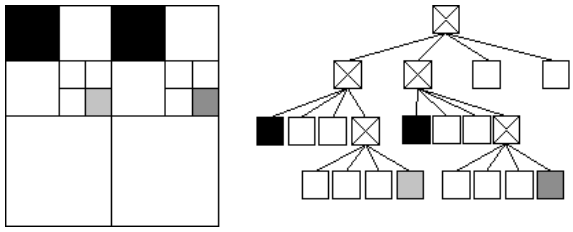

Figure 1: Left: Example of an original image to compress. Right: Image divided into quad-tree structure. Blocks with cross represent decomposed block.

The second decomposition criterion is choosing a size of a minimal block size Z. The decomposition terminates , if $w(E) \leq Z \vee h(E) \leq Z$. The two values Z and D are defined by a user, and both of them influence quality of the compressed image. The dividing is provided as stacked quad-tree. Finally, the algorithm performs decomposition on the actual level of quad-tree and after processing them continues to decompose the next level (see Fig. 1).

The maximal number of blocks can be estimated as $\frac{NM}{Z^2}$.

4.3. Search for similar blocks

Inputs: Step C2 (section 4.2.), threshold delimiting minimal difference between blocks B. *Output:* Quad-tree structure with the minimal number of non-leaf blocks

On every level of decomposition, the corresponding quad-tree (section 4.2.) is available to get all blocks E_o. For those block we can compute similarity to each other by algoritmh proposed in section 3.1. If the block solve the similarity threshold B, one of them need to remember reference to the second. The second block is not decomposing. In one moment, every block can hold maximally one reference to an other block. When is decomposed block holding reference, blocks on the next level inherit reference from them. Finally, when decomposition pass for all blocks, the data are sorted and saved only blocks without reference.

4.4. Image block composition

Inputs: Quad-tree structure from step C3 (section 4.3.)
Output: Quad-tree structure with the minimal number of non-leaf blocks

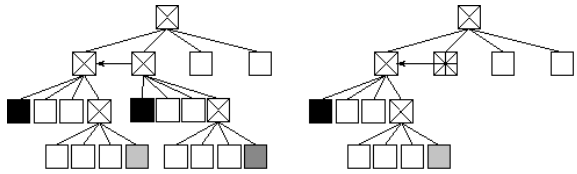

Figure 2: Left: Quad-tree structure with marked reference to similar block. Right: Updated quad-tree without blocks redundant in similarity. The block with star represent block with reference to another one.

The block is decomposed, if satisfy conditions described in section 4.2. It does not matter, if the intensity difference points are in the left top corner, or in the center of the original block. From these reason, in many cases the decomposition create two, or three block with the same component value and one with different for the next decomposition. We can boost compress ratio by joining blocks

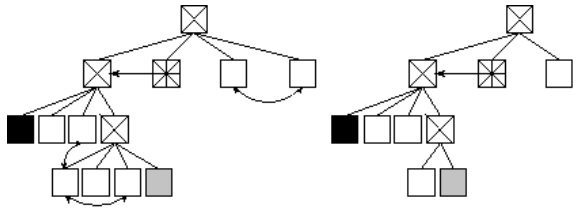

Figure 3: Left: Quad-tree structure with marked blocks to join. Right: updated Quad-tree structure.

two blocks E_1 and E_2, if $val(E_1) = val(E_2)$ and $R(E_1) = R(E_2)$ and pass all of (I)–(III) or (IV)–(VI) of following condition:

(I) $x(E_1) = x(E_2)$

(II) $w(E_1) = w(E_2)$

(III) $|y(E_1) - y(E_2)| \leq min(h(E_1), h(E_2))$

(IV) $y(E_1) = y(E_2)$

(V) $h(E_1) = h(E_2)$

(VI) $|x(E_1) - x(E_2)| \leq min(w(E_1), w(E_2))$

When the condition (I)–(III) pass, the new block E_N is created as: $x(E_N) = x(E_1)$; $y(E_N) = min(y(E_1), y(E_2))$; $w(E_N) = w(E_1)$; $h(E_N) = h(E_1) + h(E_2)$. In case of solving conditions (IV)–(VI) the new block E_N is created as: $x(E_N) = min(x(E_1), x(E_2))$; $y(E_N) = y(E_1)$; $w(E_N) = w(E_1) + w(E_2)$; $h(E_N) = h(E_1)$. In both cases value of F-transform component for the block is: $val(E_N) = val(E_1)$. The back composition is recursively computed from the lowest to the top level of quad-tree, until some blocks are joined.

5. Decompression

The decompression(reconstruction) is a transform from $M'N'$ space back to MN space. We propose

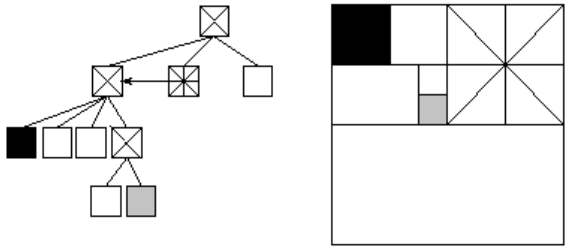

Figure 4: Left: Final quad-tree structure. Right: the quad-tree structure visualized as an image.

to make the decompression on the basis of the inverse F-transform. Because an application of the direct and inverse F-transform leads to the lossy decompression, our goal is to minimize data loss. We propose to minimize the loss by decompression of the stored essential pixels.

5.1. Decompression of essentials pixels

The block with high values of the function g is added to the image reconstructed by the inverse F-transform. We have to put pixels from this block into their own layer above the currently decompressed layer. After that we can merge layers hierarchically.

6. Benchmarks

The results are measured for gray-scale images with resolution 512×512px. The resulting table contains comparison with previous results in [5] and [6].

6.1. Estimation of a quality of reconstruction

The following criterion is used for estimation of a quality of a reconstructed image. $PSNR$ (Peak Signal to Noise Ratio) measures a similarity between an original image and its reconstruction after. Higher value of $PSNR$ means better quality of result.

$$PSNR = 20log\left(\frac{max(f)}{\sqrt{MSE}}\right)[dB] \tag{7}$$

$$MSE = \frac{1}{M \cdot N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x,y) - q(x,y))^2$$

By $max(f)$ we mean the maximum value of the intensity of the original image f. By q we mean the intensity value of the decompressed image.

6.2. Resulting tables

For the demonstration of results we chose two well-known images: Lena (Table 1) and Cameraman (Table 2). Meanwhile the Lena image is photo-like image with many details and textures, the Cameraman image contain big area of sky which is almost homogeneous. As results show, in both cases a significantly improvement compare to the last version

Figure 5: PSNR estimation of the reconstructed images compressed by the proposed algorithm. Top left: original image. Top right: $CR = 0.23$, PSNR $= 41dB$. Bottom left: $CR = 0.14$, PSNR $= 34dB$. Bottom right: $CR = 0.02$,PSNR $= 27dB$.

CR	JPEG	FTR	DFTR	DSFTR
0.03	29	23	24	28
0.06	32	24	27	29
0.14	35	26	30	34
0.25	37	28	33	37
0.44	38	30	39	42

Table 1: PSNR of Lena image

of algorithm without similarity computation and block rejoining [6]. In comparison with the JPEG format compression, our algorithm provides better results, and especially for the image Cameraman. This shows that our compression is more suitable to to cartoon-like images.

7. Conclusion

In this paper, we proposed a new image compression algorithm on the basis of the F-transform. The main idea is taken from [6] - image is dynamically partitioned with the quad-tree algorithm and compressed by applying the F-transform. The additional improvement of previous F-transform-based algorithms consists in establishing groups of similar

CR	JPEG	FTR	DFTR	DSFTR
0.03	25	20	25	27
0.06	28	21	28	30
0.14	33	23	30	34
0.25	38	25	37	41
0.44	45	27	43	48

Table 2: PSNR of cameraman image

blocks and applying compression to single representatives of groups. The effectiveness of the proposed algorithm is additionally improved by optimization of hierarchic topology, when similar parts are joined and described by only one F-transform component.

Resulting tables contain PSNR estimations of the reconstructed images Lena and Cameraman compressed by the proposed algorithm. The results show that the proposed algorithm is comparable with one of most used algorithm for image compression - JPEG.

The proposed algorithm can be applied for both type of images (photo like and cartoon like), and it is more effective for the images of the second type.

The future research will be focused on the reduction of computation complexity of the proposed algorithm.

8. Acknowledgment

This work was partially supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and the project SGS14/PrF/2013.

References

[1] Klement, E. P. and Mesiar, R. and Pap, E. *Triangular Norms*, Kluwer, Dordrecht, 2000.

[2] Z. 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*, vol. 13, no. 4, 600–612, 2004.

[3] I. Perfilieva, B. De Baets, Fuzzy transforms of monotone functions with application to image compression, *Information Sciences* vol. 180, 3304–3315, 2010.

[4] I. Perfilieva, Fuzzy transforms: Theory and applications, *Fuzzy Sets and Systems*, vol. 157, 993–1023, 2006.

[5] F. Di Martino, V. Loia, I. Perfilieva, and S. Sessa, An image coding/decoding method based on direct and inverse fuzzy transforms, *Int. Journ. of Appr. Reasoning*, vol. 48, 110–131, 2008.

[6] P. Hurtik and I. Perfilieva, Image compression methodology based on fuzzy transform, *Advances in Intelligent and Soft Computing. Proc. Intern. Conf. on Soft Computing Models in Industrial and Environmental Applications (SoCo2012)*, 525–532, 2012.

[7] R. Finkel and J.L. Bentley, Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica* vol. 4, 1–9, (1974).

[8] F. Di Martino, S. Sessa: Compression and decompression of images with discrete fuzzy transforms. *Inf. Sci.*, vol. 177(11), 2349–2362, 2007.