UNIVERSITY OF OSTRAVA
FACULTY OF SCIENCE
DEPARTMENT OF MATHEMATICS

IMAGE PROCESSING USING SOFT-COMPUTING METHODS

Ph.D. THESIS

AUTHOR: Petr Hurtik
SUPERVISOR: Irina Perfilieva

2016

OSTRAVSKÁ UNIVERZITA V OSTRAVĚ
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATIKY

ZPRACOVÁNÍ OBRAZU METODAMI SOFT COMPUTINGU

DOKTORSKÁ DISERTAČNÍ PRÁCE

AUTOR: Petr Hurtik
VEDOUCÍ PRÁCE: Irina Perfilieva

2016

Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.

Ostrava ....

(podpis)

Beru na vědomí, že tato doktorská disertační práce je majetkem Ostravské univerzity (autorský zákon Č. 121/2000 Sb., §60 odst. 1), bez jejího souhlasu nesmí být nic z obsahu práce publikováno.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Ostravské univerzity.

Ostrava ....

(podpis)

Acknowledgements

I would like to thank my supervisor, prof. Irina Perfilieva, not only for her knowledge but also for her patience, energy and approachability. During my studies I also found support in my colleagues, namely Nicolas Madrid, Petra Števuliáková, Martin Štěpnička, Marek Vajgl and Vilém Novák to name just few. These people provided me with their advise, co-authored papers and showed me correct formulations and provided new ideas. They also helped me, not only from the professional point of view, but by becoming my friends.

The last but not the least, I have to express my gratitude to my dear wife Gabriela for providing the perfect family atmosphere which made all the difficult periods that little bit easier.

Anotace

Práce se zabývá zpracováním obrazu v kontextu soft-computingu. V práci je každá úloha zpracování obrazu identifikována jako trojice skládající se ze vstupních dat, příslušnou metodou a požadovaným výstupem. Cílem práce je zachování požadovaného výstupu při nahrazení klasického matematického přístupu ve vstupních datech a metodě soft-computingovými reprezentacemi a metodám jim upraveným. V práci jsou analyzovány dvě soft-computingové reprezentace obrazu. Oba přístupy jsou speciálním tvarem rastrové reprezentace obrazu a nahrazují původní obrazovou matici intenzit. V prvním případě maticí komponent F-transformace, kde každá komponenta je projekcí obrazu na lineární podprostor ortogonálních polynomů v závislosti na stupni F-transformace. V druhém případě dochází k nahrazení maticí speciálních fuzzy množin, konkrétně fuzzy čísel s trojúhelníkovou funkcí příslušnosti vytvořenou z okolí původních obrazových bodů.

V textu jsou rozlišeny metody a aplikace. Část "Metody zpracování obrazu" popisuje navržené metody redukce obrazu, zvětšení obrazu a detekce geometrických primitiv. V části "Aplikace metod zpracování obrazu" jsou popsány řešení komprese obrazu, hledání vzorů, registraci obrazů a řidičově asistentu.

Ve všech navržených řešeních jsou použity soft-computingové reprezentace obrazu jako vstup a dle něj upravena příslušná metoda. Dosažené výsledky jsou srovnány s výsledky existujících řešení za použití rozdílných metrik pro hodnocení kvality a komplexity. Z práce plyne, že soft-computingové přístupy ve zpracování obrazu poskytují vyšší výpočetní rychlost, případně vyšší kvalitu výstupu, než standardní řešení.

Práce zároveň slouží jako shrnující komentář a úvod k deseti publikovaným článkům, které jsou do práce přiloženy. Tyto články byly vybrány jako reprezentativní vzorek hlavních autorových publikací.

Klíčová slova: Zpracování obrazu; Fuzzy; F-transformace; Komprese obrazu; Redukce obrazu; Hledání vzoru; Registrace obrazu.

Summary

The thesis focuses on image processing in the context of soft-computing. The goal is to apply different from ordinary mathematical tools to input data and in particular, methods inspired by the theories of soft-computing. Two soft-computing image representations are analyzed. Both are based on a specific transformation of a matrix of intensities to the matrix of the so called F-transform components or to the matrix of fuzzy numbers. In the first case, F-transform components coincide with image projections on corresponding linear subspaces of orthogonal polynomials whose degrees are determined by the degree of the F-transform. In the second case, fuzzy numbers are of triangle shapes and constructed from pixels' neighborhoods.

In the text, we distinguish between methods and applications. The part "Image processing method" describes the proposed methodologies for image reduction, upscaling and detection of geometric primitives. The next part "Image processing applications" covers particular solutions to general problems: compression, registration, pattern matching, as well as to the one specific problem known as driver assistant.

In all proposed solutions, we use the soft-computing representations as inputs and modify ordinary methods accordingly. The achieved results are compared with the existing approaches using various quality and complexity criteria. The thesis demonstrates that the proposed soft-computing methods provide higher computation speed and better quality outputs than the majority of standard solutions.

The thesis is comprised of ten enclosed publications and the summarizing text. The included publications were selected as the best representatives of the author's works.

Keywords: Image processing; Fuzzy; F-transform; Image compression; Image reduction; Pattern matching; Image registration.

Table of Contents

Motivation

Image processing is a discipline involving many tasks such as image compression [18], image scaling [19], image content analysis [34] and for each of these tasks there is a number of various methods to solve it. Generally speaking, every image processing task can be identified as having three components: input data, a particular method and a characterization of the acceptable result. This algorithmic approach is in the case of image processing extended further by the aspect of human perception reflected in all components of the triplet. The first component, input data, is algorithmically treated as a matrix of given values. However, the human brain perceives the image as a whole forming interaction within itself [23]. Consequently, two pixels of the same colour will be perceived differently depending on their surroundings. A human being also projects an image into objects he knows and recognizes in this way. The second component, a method, has the basics in statistics and integral calculus. Equivalently, there are methods inspired by the behaviour of real animals, esp. in the task of image segmentation. The last component, the desired output of the method, can be compared with the actual output with the use of metrics (e.g., image compression output). Some outputs can only be compared subjectively by the vote of people (e.g., image sharpening output).

This thesis is focused on two sub-parts. In the first part, the standard input data was used and a unifying soft-computing method, which would allow us to solve a plurality of different image processing tasks, was searched for. In the second part, an own way of image representation which better reflects human perception of relationships within the image was introduced. The existing methods were adapted to work with this representations.

As a unifying method for the first part was chosen a fuzzy (F)-transform [40] which performs a transformation from the input data space into the space of the F-transform components and back again to the original data space. An identification tasks where this property can be used has been made; F-transform has been modified for the use in that tasks and compared with the original solution. Specifically, the tasks of image compression [18], image reduction [45], edge detection [44], image registration [21] and pattern searching [20] were tackled.

The second part is focused on a newly proposed representation of an image, for which the existing algorithms were modified so that it formed a new version of the input data. The main idea of the representation is to describe the relationship of a visual element with its surroundings using fuzzy numbers. By means of this, the representation reflects human perception of specific parts of the image depending on their surroundings. A supposition that the modification of existing methods for working with a matrix of fuzzy numbers can reach a subjective and objective improvement has been verified on the tasks of image interpolation [16] and on detection of geometric objects in the image [34]. A property that this representation also leads to a reduction in computational complexity was demonstrated on the task of removing noise from a video [22].

The purpose of this thesis is to adapt existing algorithms for image processing using the above-mentioned alternative image representations, to analyze the resulting output and compare it by an appropriate metrics with the existing solutions. The aim of this preface is to introduce a simplified overall view of the issues solved and specific definitions. Detailed descriptions of the solutions are shown in the enclosed publications.

The entire text of the thesis is motivated by two questions: *Why?* and *How?*. *Why?* emphasizes rationale and applicability of the method – the mere existence of the means to get the result is not sufficient unless it not leads to a solution to the real problem. At the same time, *How?* addresses precise formulation of a problem, description and justification of the proposed solution.

Image representation

Let us consider a function of two variables defined as

$$f : D \to L,$$

where $D \subseteq \mathbb{R}^2$ a $L \subseteq \mathbb{R}$. In this thesis our attention is turned to the two-dimensional finite discrete domain of natural numbers of size $M \times N$, where M is the width and N is the height and one-dimensional finite range of values of natural numbers including zero. Each such a function will be identified with a two-dimensional gray-scale image. In the concept of informatics the elements of the domain are called pixels (derived from picture elements) and elements of field values intensity.

In the field of computer graphics, two main representations of the image function are used – raster and vector representation. The raster one represents f as a matrix I with the definition of all functional values:

$$I = \begin{bmatrix} f(1,1) & f(2,1) & \dots & f(M,1) \\ f(1,2) & f(2,2) & \dots & f(M,2) \\ \vdots & \vdots & \ddots & \vdots \\ f(1,N) & f(2,N) & \dots & f(M,N) \end{bmatrix}.$$

On the contrary, the vector representation does not define individual functional values. Instead, it describes a representation of geometric primitives by means of control points and primitive equation. In the case of image visualization for an arbitrary large domain size, the relative coordinates of the control points of the primitive are recalculated, the remaining points are calculated using the primitive equation and are displayed as a raster image. This thesis will further examine only the raster representation.

As mentioned in motivation, the considered raster image representation provides a complete list of image values but it does not contain any information about the purpose, objects, etc. The subject of this thesis is a soft-computing representations research and subsequently image processing methods, namely representation using the F-transform components and using the fuzzy function. The advantage of these representations is a comprehensive view of the area of each pixel and thus closer to human understanding.

Image representation using F-transform components

The F-transform technique[42, 40] is used to reduce the number of elements of the domain used for the image representation. The image F-transform is represented by a matrix of components $F[I] : D' \to L'$, where $D' \subseteq D$ and $L' \subseteq L$ and describes an image as a function of fuzzy approximation space. Depending on the degree of F-transform each component coincides with the image projected on the linear subspace of orthogonal polynomials. The essential is that orthogonality of each projection has the weight of so-called basic function which defines the fuzzy partition of the image domain. The thesis primarily focuses on F^0-transform (zero degree) where $F[I]$ is a matrix of weighted averages of functional values when weights are determined by specific basic functions forming a fuzzy partition of the domain. In this thesis we assume $|D'| < |D|$, which allows us to solve computationally complex tasks (especially pattern searching [14]) trivially in a short computation time. Using standard trivial methods, it would not be possible to solve these tasks with regards to the length of calculation.

Image representation using a fuzzy function

The basic image processing methods (image blurring and sharpening, gradient detection) convolve the image function by the core relevant to the task. This process is slow, because for the core matrix K is realized $|D| \cdot |K|$ operations. An alternative soft-computing image representation has been proposed in our work [34] – instead of the range of values $L \subseteq \mathbb{N}$, the use of range of values $L \subseteq \mathbb{F}$ is suggested, where $\mathbb{F}$ is the relevant class of fuzzy sets. We also assume specific types of fuzzy sets – a fuzzy number that is represented by a triangular membership function. Each fuzzy number is constructed by a supremum and infimum of subset in the surrounding of the corresponding (i.e., central) pixel. The image representation designed in this way corresponds to a fuzzy function. To work with the image represented by a fuzzy function, a known apparatus for work with fuzzy sets and numbers is used [39]. The proposed representation allows us to describe pixel ties with its surroundings using fuzzy numbers, thanks to which there is no need to use convolutions in basic image processing tasks. This leads to a reduction in computational complexity [16, 22], and at the same time to the increase in the output quality of existing algorithms [34], which have been adapted to work with this representation.

Image processing methods

In this part, the image processing methods will be described: image reduction [45], image enlargement [16] and lines detection [34]. These methods were examined from a soft-computing point of view and applied in specific applications.

Image reduction

Images are reduced to save the storage space, in order to reduce the transmission time, or due to image adjustment when the number of pixels is greater than the number of pixels displayable on a particular device. The advantage of a smaller number of pixels can be used for a pre-processing in a task, where the image is input for computationally intensive algorithm. In contrast with the image compression, the reduced image can be directly visualized without applying a decompression algorithm. Principles of the image reduction are as follows: a new domain is defined using a reduction ratio and intensities are calculated from the original image. The difference between the methods is in modalities for calculating the new domain intensities.

The easiest (and the fastest) transformation used to reduce an image is sub-sampling [6]. It selects the intensity of each n^{th} pixel of the original image, where n represents the reduction ratio. The drawback is the possible distortion of objects smaller than $2n$ due to the violation of Shannon sampling theorem.

Two main groups of image reduction methods can be categorized as follows: the first is based on aggregation functions [3], see application [2]; the second group uses a polynomial interpolation [8] – bilinear [37], bicubic [27] or Lanczos [9].

This thesis is concerned with reducing the image using the F-transform. Since the image representation by F-transform components is a reduced model of an image, it is a natural task. To improve the quality of the reduced image, its sharpness, the F-transform definition has been modified – fuzzy partition has been generalized to fuzzy coverage. Image reduction using F- transform was published in [19, 45]. The proposed algorithm was compared with existing aggregation and interpolation algorithms. In comparison, a higher degree of similarity between the reduced image and the original has been proven and a higher processing speed has been demonstrated.

Image upscaling

This technique is also known as image enlargement, magnification or zooming. A typical application is to increase the resolution (number of pixels) of the video [26], which consists in the enlargement of each image, because videos are usually stored in a small size domain. Another application is zooming, where the subset of the domain is transformed into a larger one in order to obtain an image more vivid to humans. Image enlargement is linked to image reconstruction [46]: this task deals with undefined intensities, which are calculated according to the existing ones in their surroundings. We note that image enlargement cannot add any new information to the new image function (detail in the image) which did not exist in the original image function.

Magnification process can be understood as the inverse procedure to image reduction. We define a new domain from the inversion of reduction ratio (here the magnifying ratio). The original set of intensities is than extended into a new domain and the intensities of added pixels are interpolated. Polynomial interpolations used are again bilinear [37], bicubic [27] and Lancszos [9]. Other methods (mostly in the form of a theoretical use) are fractal interpolation [50], hq$n$x interpolation [29] or methods based on fuzzy interference system [7].

In our work [16], the bilinear interpolation was used to enlarge the image and its main drawback – blurred output image – was identified. For the input was used a proposed image represented by a fuzzy function and the bilinear interpolation was modified to work with this new input. This representation also allows us to approximate the image gradient, which was used to emphasize high frequencies in the final image. The output of the modified bilinear representation was compared [16] with the original one combined with sharpening via the Laplace operator. A significantly lower computational time was reached while the same results were achieved.

Detection of geometric primitives

Detection (position identification) of geometric primitives means finding a subset of the domain whose points satisfy the equation that defines the geometric primitive or another defined condition. As a geometric primitive we assume a line, circle, ellipse, etc. This task

is used in applications of image segmentation, image vectorization or pattern searching. Techniques used to solve problems are mathematical morphology [49, 30], combinatorial optimization [35], the Hough transform [10] and testing hypothesis and paradigms [31].

In our work [34], we focused on the Hough transform in the context of lines detection. For this purpose, we will use Hough space $R \times \Theta$ in which every line is identified by its unique location point $O = (r, \theta)$, where r denotes the length of the perpendicular drawn from to point $(0, 0)$ and θ denotes the angle between the perpendicular and the x axis. The goal is to find such a points O for which hold that value in Hough space is higher, than some given threshold.

The Hough transform technique was used in our work [34], where the original technique was modified for the input represented by a fuzzy function. This modifications led to an increase in detection accuracy of broken or damaged lines and also to reduction in the computational complexity.

Image processing applications

This section describes selected applications to which were applied the techniques of image processing using soft-computing and which were examined during the survey. Namely - image compression [18], image registration [21], pattern matching [14] and line detection in terms of detection roadsides and centerline on the road [33].

Image compression

Image compression is as well as image reduction performed to reduce the domain size. Unlike the reduction, field values can also be transformed. In addition, the result of compression – the compressed image – must be transformed back to the original domain and range of values using the so called decompression algorithm to be visualized. In image compression lossy and lossless methods are considered. Regarding lossy methods, the output is an approximation of the original image function. Regarding lossless methods, the image function is identical to the original.

Let us mention three compression formats in raster image representation: *jpeg* which is lossy, used mainly for photos; *png* - lossless format suitable for images containing text; and *bmp* - historical lossless format which only describes an image originally without compression. However, there are modifications of these formats. Jpeg can be lossless [5], png can be lossy and bmp can use compression methods such as RLE (Run Length Encoding).

Compression algorithms used in compression formats can be divided according to how they access the image information. A discrete cosine transform (used in jpeg) suppresses small changes of intensity that a person may not recognize. A Quad-tree technique [11] is used to recursively divide the image and to describe a subset of the image by only one value. Fractals [12] use a self-similarity of image subsections. Fuzzy relational equations [38] may be used to describe relationships within the image. When considering the image as a generic data, compression algorithms known from signal theory as LZW or Huffman coding can be applied. Highly significant is algorithm Deflate (a combination of LZW and Huffman coding) [36] which use compression formats to further compress an already compressed image to a binary file.

In our work [18], two main ideas for image compression are used – the first one involves extracting a subset of the image which represents so-called important pixels that are significant for human visual perception. This significance is determined by a gradient operator and constitutes high frequencies in the image, the details. This subset is then removed from the picture and stored separately in a lossless format. The second idea is the use of Quad-tree decomposition which divides an original domain into four disjoint subsets of the same size and those then recursively divides further. As stopping conditions we used the minimum size of the subset and the lack of high frequencies in the subset.

The proposed algorithm [17], [18] initially extracts a subset of important pixels, then applies the Quad-tree decomposition and for each area calculates F-transform components. This algorithm provides a high degree of backward decompressed image similarity to the original image. The reason is the presence of visually important points in the decompressed image and decompressed linearity function (without important pixels) rather than piecewise linear function in the case of using the average value as the standard algorithm. By means of controlling the size of the set of important points and stopping conditions the strength of compression algorithm can be altered. In our work [18] we applied both, a standard and proposed algorithm for the same size of the compressed images, made comparisons and demonstrated that the approach based on the technique of F-transform provides considerably more similarity to the original than the standard Quad-tree compression algorithm.

Image registration

The goal of the image registration is to find the same subsections of two or more images in order to link these images into one. Images taken from multiple sensors, from multiple perspectives, of a different size or taken at different times are the example of the input. A classic use is to create panoramas, i.e., link more images that partially overlap into one larger image.

Methods used to solve the problem are SIFT [32], SURF [1] or ORB [47]. These methods have the same basis and are decomposable as a solution to the four subtasks [51]: extraction of important points and describing features; matching the features across

images; finding an appropriate transformation function for each image describing shift, scaling, and the rotation between the images; an image interpolation to the same domain and an image fusion. In our work [21] we designed an algorithm which processes the following steps using the F-transform.

The extraction of important points consists in identifying the positions of the corner points (points in whose neighborhood is changing the gradient angle). These corner points can be found in parts of the image remaining the same even if rotated, scaled, or constantly changing intensity. In our work, we propose to use the F-transform of the first degree [43] to calculate the gradient and detect these points. The feature is the surrounding of the point which is described by F-transform components.

To match the features found among the images, i.e., a consensus in the content of the important points, an algorithm calculating the distance between the F-transform components, which are hierarchically (recursively) calculated from themselves, has been designed. This approach allows us to find similarities in the surrounding of the point which has been slightly rotated. Originally this approach was used in the image compression [18].

To achieve a perspective distortion, changes in scale, rotation and shift, a standard homography approach [24] was used, when from the position of the four important points in the two different images is calculated a transformation matrix, by which is one of the images multiplied.

The last step, image fusion, is applied to those parts of images that overlap. The image fusion goal is to gain one point or create a new one from multiple entry points, which represents entry points the best. The output is a final image that contains suitably aggregated information from all inputs. A detailed description of the fusion algorithm based on the F-transform can be found in [48].

In our work [21], we have demonstrated that a single technique (F-transform) can be used for various subtasks in the image registration. The benefit is a potential algorithm optimization and savings in the time required for development of the application.

Pattern matching

Pattern matching in the object is a process, which identifies the position of the pattern (in general a vector, matrix or r-dimensional space) in an object that contains this pattern and simultaneously proclaims the absence of such a pattern in an object that does not contain it. The premise is that the pattern has either fewer or the same number of elements of the domain as the object in the database where it is searched for.

An algorithm for pattern matching in one-dimensional space is called *string-matching*. As a reference are considered methods Rabin-Karp [25], Knuth-Morris-Pratt [28] and Boyre-Moore [4]. These methods are matching the pattern as an exact match, i.e., match between the pattern values and its accurate and any difference is not allowed in the database. If such a difference is allowed, we called it an approximate match. The above mentioned algorithms may also be used after the modification and with the help of other algorithms to work with two-dimensional space, but no difference is allowed.

In our work [15] we developed the algorithm designed for one-dimensional fuzzy matching. The main idea was to transform the original string (the vector) into the F-transform [41] components. This led to the reduction in the original string. Matching was implemented by a naive algorithm comparing all combinations of the pattern in the object database. Without the domain reduction matching would lead to too much computing time.

In other parts of our work we extended our algorithm to work with two-dimensional data [14] and then with the general r-dimensional space [20]. At the same time, we justified the basic principles of F-transform selection, the corresponding decomposition and precisely defined individual steps. The proposed algorithm is suitable for both, the exact and approximate pattern matching. A general use extends to informatics, where the algorithm can be used for reduction and a subsequent matching in the database whose original size exceeds the capacity of RAM and would have to be processed in parts from a slower hard drive. At the same time, the matching speed is in comparison with existing algorithms much higher.

Driver assistant

Driver assistant is software that monitors the road ahead of the car in real time and watches the lanes crossing. This software was developed and implemented on a mobile phone that can be placed under a car windshield and warns the driver when leaving the lane.

The software core forms the algorithm for detecting lines. Lines correspond to the segments of lines in the image and represent the side and center line painted on the road. As an algorithm for detecting lines was used Hough transform which we modified to work with the image input represented by a fuzzy function [34]. The original algorithm transformed an image to the so-called dual space (Hough space) consisting of perpendicular distance from the origin and the slope of a line. Line detection took place in this space, which reduced the computational time as compared to the naive approach. In experiments, this algorithm has demonstrated little robustness in detection of a damaged line or a line that has been deformed (curved) and hence not completely straight.

In order to increase the robustness we replaced the original raster input with the image represented by fuzzy function, where each picture element is described by a fuzzy number with a triangular membership function. Subsequently the original Hough transform algorithm was modified. The difference lies in two aspects. The first one is in calculating the gradient image, when the gradient approximation is expressed by the support size of a specific fuzzy number. This operation requires significantly less processing time than required by the standard (Sobel) gradient operator. The second aspect is the search for the desired intensity of the line when the intensity is the membership function parameter and the result is a degree of membership. Since each fuzzy number is defined on the basis of the surrounding pixels, minor lines interruptions are suppressed. The algorithm then works only with the picture elements whose gradient is greater than a defined threshold, while the degree of membership is higher than the second threshold.

The proposed algorithm was implemented and experiments have shown that processing speed and robustness against deformation and lines interruption were greatly improved by our soft-computing approach.

Conclusion

In the text above, we presented a summary of image processing using soft-computing methods. In the beginning, we focused on raster image representation and described the two soft-computing representations, namely image represented by the F-transform components and the image represented by fuzzy function. These representations have been used in the methods of the image reduction, magnification and the detection of geometrical primitives. They were at the same time applied in applications of image compression, image registration, pattern matching and practical application for tracking road lanes. The aim of the thesis, namely the introduction of these representations and their applications to selected technologies and tasks has been achieved by publishing the results, which are attached to the work.

In this thesis, three important milestones were achieved. First, we showed that even formal evidence-based approaches can capture the human factor of the image perception, i.e., a simplified view of the image, perception of the object surrounding. Secondly, comparing with traditional approaches we proved that the alternative approaches are distinguished by low processing complexity and when compared with the relevant metrics they provide a better quality output. Thirdly, we made presentations at conferences and published these results in proceedings and journals where our procedures were also accepted by the scientific community. Also, our work has been applied in commercial projects, which have demonstrated a practical applicability and reasonability of our proposals.

When estimating future developments in the area described, it can refer to the technique called ”deep learning”, which at the time of writing this thesis, takes on a special role in solving problems of image processing. This approach has the same motivation as this thesis (inspired by human behavior), its disadvantage is its internal spatial complexity and sophistication resulting in low interpretability of the whole process and the results obtained. As a consequence, it is difficult to justify the results achieved, which may be limiting for some applications where it is necessary not only to receive the result but also to explain the procedure of getting it. Therefore, the author expects that the approaches introduced by him, including the defendability and explainability of results obtained may be a more appropriate means of addressing the examined types of tasks in the future.

Bibliography

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] G. Beliakov, H. Bustince, and D. Paternain. Image reductions using means of discrete product lattices. *IEEE Transactions on Image Processing*, 21(3):1070–1083, 2012.

[3] G. Beliakov, A. Pradera, and T. Calvo. *Aggregation functions: A guide for practitioners*, volume 221. Springer, 2007.

[4] R. S. Boyer and J. S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

[5] R. Brennecke, U. Bürgel, G. Rippin, F. Post, H.-J. Rupprecht, and J. Meyer. Comparison of image compression viability for lossy and lossless jpeg and wavelet data reduction in coronary angiography. *The international journal of cardiovascular imaging*, 17(1):1–12, 2001.

[6] P. Camana. Image processing techniques for compression. In *NAECON 1979*, volume 1, pages 1298–1302, 1979.

[7] J.-L. Chen, J.-Y. Chang, and K.-L. Shieh. 2-d discrete signal interpolation and its image resampling application using fuzzy rule-based inference. *Fuzzy sets and systems*, 114(2):225–238, 2000.

[8] P. J. Davis. *Interpolation and approximation*. Courier Corporation, 1975.

[9] C. E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, 1979.

[10] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Graphics and Image Processing, Communications of the ACM*, 15(1), 1972.

[11] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[12] Y. Fisher. Fractal image compression. *Fractals*, 2(03):347–361, 1994.

[13] P. Hodáková. *Fuzzy (F-)transform of functions of two variables and its application in image processing*. University of Ostrava, Ostrava, 2014.

[14] P. Hurtik and P. Hodakova. Ftip: Tool for image plagiarism detection. In *Soft Computing and Pattern Recognition*. IEEE, November 2015. In press.

[15] P. Hurtik, P. Hodakova, and I. Perfilieva. Fast string searching mechanism. In *16th World Congress of the International Fuzzy Systems Association (IFSA) 9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, 2015.

[16] P. Hurtik and N. Madrid. Bilinear interpolation over fuzzified images: enlargement. In *The 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015)*, 2015.

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

[18] P. Hurtik and I. Perfilieva. Image compression methodology based on fuzzy transform using block similarity. In *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*. Atlantis Press, 2013.

[19] P. Hurtik and I. Perfilieva. Image reduction/enlargement methods based on the f-transform. *European Centre for Soft Computing, Asturias*, pages 3–10, 2013.

[20] P. Hurtik and I. Perfilieva. Submitted: Approximate pattern matching algorithm. 2016.

[21] P. Hurtík, I. Perfilieva, and P. Hodáková. Fuzzy transform theory in the view of image registration application. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 143–152. Springer, 2014.

[22] P. Hurtik, M. Vajgl, and N. Madrid. Accepted: Enhancement of night movies using fuzzy representation of images. In *IEEE World Congress on Computational Intelligence (IEEE WCCI)*, 2016.

[23] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10):1489–1506, 2000.

[24] K. Kanatani and Y. KANAZAWA. Optimal homography computation with a reliability measure. *IEICE Transactions on Information and Systems*, 83(7):1369–1374, 2000.

[25] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

[26] S. H. Keller. *Video upscaling using variational methods*. PhD thesis, Ph. D. thesis, Faculty of Science, University of Copenhagen, 2007.

[27] R. Keys. Cubic convolution interpolation for digital image processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 29(6):1153–1160, 1981.

[28] D. E. Knuth, J. H. Morris, Jr, and V. R. Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

[29] J. Kopf and D. Lischinski. Depixelizing pixel art. In *ACM Transactions on graphics (TOG)*, volume 30, page 99. ACM, 2011.

[30] P. Kupidura. Application of mathematical morphology operations for the improvement of identification of linear objects preliminarily extracted from classification of vhr satellite images. In Z. Bochenek, editor, *New Developments and Challenges in Remote Sensing*, pages 225–232, 2007.

[31] W. Liu and D. Dori. A generic integrated line detection algorithm and its object-process specification. *Computer Vision and Image Understanding*, 70(3):420–437, 1998.

[32] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[33] N. Madrid and P. Hurtik. Lane departure warning for mobile devices based on a fuzzy representation of images. *Fuzzy Sets and Systems*. submitted.

[34] N. Madrid and P. Hurtik. Lane departure warning for mobile devices based on a fuzzy representation of image. *Fuzzy Sets and Systems*, 2015.

[35] M. Mattavelli, V. Noel, and E. Amaldi. Fast line detection algorithms based on combinatorial optimization. In *Visual Form 2001*, pages 410–419. Springer, 2001.

[36] J. Miano. *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.

[37] P. Miklos. Image interpolation techniques. In *2nd Siberian-Hungarian Joint Symposium On Intelligent Systems*, 2004.

[38] H. Nobuhara, K. Hirota, F. D. Martino, W. Pedrycz, and S. Sessa. Fuzzy relation equations for compression/decompression processes of colour images in the rgb and yuv colour spaces. *Fuzzy Optimization and Decision Making*, 4(3):235–246, 2005.

[39] V. Novák, I. Perfilieva, and J. Mockor. *Mathematical principles of fuzzy logic*, volume 517. Springer Science & Business Media, 2012.

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

[41] I. Perfilieva. Fuzzy transforms: Theory and applications. *Fuzzy sets and systems*, 157(8):993–1023, 2006.

[42] I. Perfilieva and E. Chaldeeva. Fuzzy transformation. In *Proceedings of IFSA 2001 World Congress*, 2001.

[43] I. Perfilieva, P. Hodáková, and P. Hurtík. F 1-transform edge detector inspired by canny's algorithm. In *Advances on Computational Intelligence*, pages 230–239. Springer, 2012.

[44] I. Perfilieva, P. Hodáková, and P. Hurtík. Differentiation by the f-transform and application to edge detection. *Fuzzy Sets and Systems*, 2014.

[45] I. Perfilieva, P. Hurtik, F. Di Martino, and S. Sessa. Image reduction method based on the f-transform. *Soft Computing*, 2015. In press.

[46] I. Perfilieva and P. Vlasanek. Image reconstruction by means of f-transform. *Knowledge-Based Systems*, 70:55–63, 2014.

[47] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[48] M. Vajgl, I. Perfilieva, and P. Hod'áková. Advanced f-transform-based image fusion. *Advances in Fuzzy Systems*, 2012:4, 2012.

[49] S. Valero and et al. Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images. *Pattern Recognition*, 31(10):1120–1127, 2010.

[50] X. Xu, L. Ma, S. H. Soon, and C. Tony. Image interpolation based on the wavelet and fractal. *International Journal of Information Technology*, 7(2), 2001.

[51] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

List of author's contributions

Contributions marked as * are enclosed.

Journal contributions

*1. I. Perfilieva, P. Hurtik, S.Sessa and F. Di Martino. Image Reduction Method Based on the F-Transform. *Soft Computing*, 2015. In press.

*2. N. Madrid and P. Hurtik. Lane departure warning for mobile devices based on a fuzzy representation of images. *Fuzzy Sets and Systems*, 291, 144-159, 2016.

3. V. Novak, P. Hurtik, H. Habiballa, and M. Stepnicka. Recognition of damaged letters based on mathematical fuzzy logic analysis. *Journal of Applied Logic*, 13(2), 94–104, 2015.

4. I. Perfilieva, P. Hodakova, and P. Hurtik. Differentiation by the f-transform and application to edge detection. *Fuzzy Sets and Systems*, 288, 96–114, 2016.

5. F. Di Martino, P. Hurtik, I. Perfilieva, and S. Sessa. A color image reduction based on fuzzy transforms. *Information Sciences* 266, 101-111, 2014.

6. J. Krhut, M. Gartner, R. Sykora, P. Hurtik, M. Burda, L. Lunacek, K. Zvarova, and P. Zvara. Comparison between uroflowmetry and sonouroflowmetry in recording of urinary flow in healthy men. *International Journal of Urology*, 2015.

7. J. Krhut, M. Gartner, R. Sykora, P. Hurtik, M. Burda, K. Zvarova, and P. Zvara. Validation of a new sound-based method for recording voiding parameters using simultaneous uroflowmetry. *The Journal of Urology* 193, no. 4, 2015

8. M. Gartner, J. Krhut, P. Hurtik, M. Burda, K. Zvarova, and P. Zvara. Evaluation of voiding parameters in healthy women using sound analysis. *Lower Urinary Tract Symptoms*, 2016. In press.

Conference proceedings

9. P. Hurtik, M. Burda, and I. Perfilieva. An image recognition approach to classification of jewelry stone defects. In *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, 727–732. IEEE, 2013.

*10. P. Hurtik and N. Madrid. Bilinear interpolation over fuzzified images: enlargement. In *The 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015)*, 1–8, IEEE, 2015.

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

*12. P. Hurtik and I. Perfilieva. Image compression methodology based on fuzzy transform using block similarity. In *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*. Atlantis Press, 2013.

13. I. Perfilieva and P. Hurtik. F-transform for Image Reduction. *Proceedings of the 16th Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty.* Jindrichuv Hradec, 205-214, 2013.

14. P. Hurtik and I. Perfilieva. Image Reduction/Enlargement Methods Based on the F-transform. *MIBISOC 2013*, European Centre for Soft Computing, Asturias, 3–10, 2013.

*15. P. Hurtik, I. Perfilieva, and P. Hodakova. Fuzzy transform theory in the view of image registration application. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 143–152. Springer, 2014.

16. V. Novak, P. Hurtik, and H. Habiballa. Recognition of heavily distorted characters on metal. In *Proceedings of the 2013 Joint IFSA World Congress NAFIPS Annual Meeting (IFSA/NAFIPS)*, 733–738. IEEE, 2013.

17. I. Perfilieva, P. Hodakova, and P. Hurtik. F^1-transform Edge DetectorIinspired by Canny's Algorithm. In *Advances on Computational Intelligence*, 230–239, Springer, 2012.

*18. P. Hodakova, I. Perfilieva and P. Hurtik. F-transform and its Extension as Tool for Big Data Processing. *Proc. of the 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2014)*, 374–383, 2014.

19. M. Vajgl, P. Hurtik, I. Perfilieva and P. Hodakova. Image Composition Using F-Transform. *Fuzzy Systems (FUZZ-IEEE)*. Beijing, China: IEEE, 1112-1117, 2014.

*20. P. Hurtik, P. Hodakova, I. Perfilieva, M. Liberts, and J. Asmuss. Network Attack Detection and Classification by the F-transform. In *The 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015),* 2015.

21. P. Hurtik, P. Hodakova, and I. Perfilieva. Fast String Searching Mechanism. *16th IFSA World Congress and 9th EUSFLAT Conference*, 412–418, Atlantis Press, 2015.

22. P. Hurtik, M. Burda, J. Krhut, P. Zvara, and L. Lunacek. Automatic diagnosis of voiding dysfunction from sound signal. In 2015 IEEE Symposium Series on Computational Intelligence: IEEE Symposium on Computational Intelligence in Healthcare and e-health (2015 IEEE CICARE), 1331–1336. IEEE, 2015.

*23. P. Hurtik and P. Hodakova. FTIP: Tool for image plagiarism detection. *Proceedings of 2015 Seventh International Conference of Soft Computing and Pattern Recognition (SoCPaR 2015)*, 42–47, IEEE, 2015.

*24. P. Hurtik, M. Vajgl, and M. Burda. Jewelry Stones Classification: Case Study. *Proceedings of 2015 Seventh International Conference of Soft Computing and Pattern Recognition (SoCPaR 2015)*, 205–210, IEEE, 2015.

25. V. Ferdianova, P. Hurtik and A. Kolcun. Reconstruction of the borehole wall using video records. *Aplimat 2012: 11th International Conferenc*, 265–270, 2012.

26. P. Hurtik, P. Hodakova and I. Perfilieva. Approximate Pattern Matching Algorithm. *Proc. of the 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2016)*, 2016, accepted.

*27. P. Hurtik, M. Vajgl and N. Madrid. Enhancement of Night Movies Using Fuzzy Representation of Images. *IEEE World Congress on Computational Intelligence (IEEE WCCI)*, 2016, in press.

Enclosed publications

I. Perfilieva, P. Hurtik, S.Sessa and F. Di Martino. Image Reduction Method Based on the F-Transform. *Soft Computing*, 2015. In press.

Soft Comput
DOI 10.1007/s00500-015-1885-0

METHODOLOGIES AND APPLICATION

Image reduction method based on the F-transform

Irina Perfilieva[1] · **Petr Hurtik**[1] · **Ferdinando Di Martino**[2] · **Salvatore Sessa**[2]

© Springer-Verlag Berlin Heidelberg 2015

Abstract We present a new method of (color) image reduction based on the F-transform technique with a generalized fuzzy partition. This technique successfully combines approximation (when reduction is performed) and interpolation (when reconstruction is produced). The efficiency of the proposed method is theoretically justified by its linear complexity and by comparison with interpolation, and aggregation-based reductions. We also analyze the measures (MSE, PEN, and SSIM) that are commonly used to estimate the quality of reduced images and show that these measures have better values using the newly proposed method.

Keywords F-Transform · Generalized partition · Image reduction · Image resize · Interpolation

Communicated by V. Loia.

✉ Petr Hurtik
petr.hurtik@osu.cz

Irina Perfilieva
irina.perfilieva@osu.cz

Ferdinando Di Martino
fdimarti@unina.it

Salvatore Sessa
sessa@unina.it

[1] Institute for Research and Applications of Fuzzy Modelling, University of Ostrava, NSC IT4Innovations, Ostrava, Czech Republic

[2] Dipt. di Costruzioni e Metodi Matematici in Architettura, Universita degli Studi di Napoli "Federico II", Naples, Italy

1 Introduction

Various signal processing methods have been developed in recent decades to address numerous problems of multimedia and communication applications and the proliferation of signal processing software. This work is focused on the issue of *image reduction*, which is related to the compact visual representation of an image; the latter is required by mobile phones, photo cameras, tablets, etc. Moreover, problems such as image segmentation can be efficiently solved in reduced images and subsequently returned to the initial space (Di Martino et al. 2010).

There are at least two different meanings of the term image reduction. In Karabassis and Spetsakis (1995), image reduction is a (shrinking) operator that reduces the resolution of an image to speed up the computation. A low-pass filter is usually used for this purpose. In Beliakov et al. (2012), image reduction is a technique that is analogous to image compression and aims at the following:

(i) minimizing the number of bits that is required to represent an image,
(ii) maintaining an acceptable quality of a reduced image.

In our paper, we use image reduction in the second meaning.

The problems of image reduction and compression are not identical. Reduction is a twofold problem that includes (i) and (ii) as subproblems, whereas compression coincides with subproblem (i). Thus, reduction is focused on the quality of the reduced image, whereas compression is focused on the quality of the reconstructed image.

Here, we give a short overview of the frequently cited methods of image reduction (a detailed characterization is in Sect. 3, see also the overview in Thévenaz et al. (2000) on medical image interpolation and resizing). The easiest

Published online: 30 December 2015

method (it is called subsampling) consists of partitioning an image into disjoint blocks and replacing each block by one of its pixels (the central pixel is commonly used). Because the computation time of this method is notably small (it is the fastest known algorithm), it is widespread in graphic software. However, for most images, the quality of reduction by subsampling is notably low (for justification see, e.g., Beliakov et al. 2012).

Unlike subsampling, the sophisticated image reduction methods are based on interpolation with various kernels (Karabassis and Spetsakis 1995; Duchon 1979). Interpolation methods are used for image reduction in both meanings: they are low-pass filters if the number of blocks and the number of pixels coincide, and they are compression methods if the number of blocks is less than the number of pixels. In image reduction, we distinguish two types of interpolation methods: standard (non-adaptive) and advanced (adaptive). The first group includes subsampling, bilinear, bicubic, and Lanczos interpolation; these methods are used in Adobe Photoshop, Corel Paint Shop Pro, Gimp, and InfranView. The second group includes interpolation methods (bilinear, bicubic) in combination with an edge detection technique (see Hwang and Lee 2004). For example, Liu et al. (2005) demonstrates how adaptive interpolation can be used for NURBS curves. However, the usage of adaptive interpolation is notably limited because of their high computational cost and narrow focus.

In Beliakov et al. (2012), an aggregation-based image reduction method was proposed. Unlike subsampling and interpolation, this type of reduction takes a block in a partition, aggregates its pixels, and uses this aggregated value in the reduced image. Three algorithms for image reduction were proposed and tested on a set of color images that were taken from the available image dataset at the web.[1] Each algorithm chooses the best aggregation operator among $k \geq 2$ aggregation operators [with respect to the minimization of a certain penalty function (Calvo and Beliakov 2010)] and applies it for reduction. In Beliakov et al. (2012) the following aggregation operators are used: minimum, maximum, median, arithmetic, and geometric average. Moreover, reduction is independently performed for each color band, and the reduced images are aggregated.

From the given overview, it is obvious that simple, effective, and easily implemented reduction methods remain useful. Moreover, reduction methods should be estimated from various angles and on a representative set of benchmarks.

In this paper, we present a new method for color-image reduction based on the F-transform technique. We justify its suitability for image reduction by proving that the sequential application of the direct and inverse F-transform to an image works as an approximator. In parallel, we discuss various measures that are traditionally used to estimate the image quality.

The novelties of the F-transform-based reduction are the following:

- successful combination of approximation (when reduction is performed) and interpolation (when reconstruction is produced),
- the F-transform-based reduction and reconstruction are (in some sense) mutually inverse, which is reflected in the good quality of this method (the details are in Sect. 6),
- blocks in a corresponding partition (according to the scheme "block-to-pixel") overlap (the details are in Sect. 4.1), which allows us to achieve a better MSE value than the best aggregation-based reduction,
- the algorithm can be upgraded into sharpening version by extending overlapping and using negative values.

Moreover, the F-transform approach for image reduction successfully combines the advantages of aggregation when removing the noise and the precision of interpolation during reconstruction. The F-transform technique has various and easily adjustable kernels. Finally, the method itself is clear, computationally effective, and can be easily implemented.

We compare the results of the new F-transform-based reduction algorithm with the interpolation and aggregation results. The comparison is performed based on (1) three quality measures: MSE, PEN, and SSIM, (2) the computation time, and (3) the noise-removing ability. We characterize each quality measure and explain its selection in Sect. 2. In addition, we discuss what is actually measured by the quality measures MSE, PEN, and SSIM.

In the final section, we present the results of our experiments and the comparison with the interpolation- and aggregation-based reduction. For this purpose, we chose 53 color images from sipi database.[2] The images have different resolutions: 512 × 512 px (20 pieces), 256 × 256 px (8 pieces), 1024 × 1024 px (24 pieces), and 2250 × 2250 px (one piece).

The structure of our paper is as follows. In Sect. 3, we discuss the problem of image reduction, its quality measures and the conventional reduction methods. In Sect. 4, we introduce a new F-transform method and show its application to the problem of image reduction. We estimate the complexity of the F-transform-based reduction method and prove that it is linear with respect to the input length. In Sect. 6, we present the results of our experimental tests. Finally, the conclusions are provided.

[1] http://decsai.ugr.es/cvg/dbimagenes/index.php.

[2] http://sipi.usc.edu/database/database.php/volume=textures.

2 Preliminaries: image reduction and quality measures

This work is focused on the issue of *image reduction*, which is a technique that aims at a compact representation while maintaining acceptable quality.

A (gray-scale) image is identified with representing it (intensity) function $u : [1, N] \times [1, M] \to [0, 255]$, where the domain $[1, N] \times [1, M] = \{(i, j) \mid i = 1, \ldots, N; j = 1, \ldots, M\}$ and the range $[0, 255]$ contain only natural numbers. A color RGB-image is represented by three intensity functions u^R, u^G, and u^B, each of which is in the respective color band. If not explicitly mentioned, we assume that the image is gray-scale. A *reduced (compact) representation* $\bar{u} : n \times m \to [0, 255]$ of u is determined by the *reduction ratio* $\rho = \frac{NM}{nm}$, where $n < N$, $m < M$, and N and M (n, m) are the sizes of u and $\bar{u}$, respectively. The reduction ratio is commonly written in the form $\rho : 1$. There is no explicit requirement for the image $\bar{u}$. Because $\bar{u}$ and u have different domains, these images cannot be compared. The relationship between $\bar{u}$ and u can be established after the reconstruction (or enlargement) procedure, which transforms the reduced image $\bar{u}$ into the enlarged image $\hat{u}$, which is defined on the initial domain $[1, N] \times [1, M]$. If we denote

$$\hat{u} = E(\bar{u}) \text{ where } E : [0, 255]^{[1,n]\times[1,m]} \to [0, 255]^{[1,N]\times[1,M]},$$

then we can formulate the image reduction problem as follows:

Given image $u : [1, N] \times [1, M] \to [0, 255]$, *ratio* $\rho = \frac{NM}{nm}$ *where* $n < N$, $m < M$, *and the reconstruction* $E : [0, 255]^{[1,n]\times[1,m]} \to [0, 255]^{[1,N]\times[1,M]}$, *find image* $\bar{u} : n \times m \to [0, 255]$ *such that a chosen quality of reduction* Q *is close to its minimal/maximal value, i.e., the value* $Q(u, E(\bar{u}))$ *is as small (large) as possible.*

In this paper, we assume that reduction is performed based on the "block-to-pixel" scheme. This assumption implies that the domain $[1, N] \times [1, M]$ of the image function u is partitioned into $N_b \times M_b$-sized blocks $B_{1,1}, \ldots, B_{n,m}$, each block $B_{i,j}$ is replaced by one pixel (i, j), and this pixel is assigned a new intensity value $\bar{u}(i, j)$. The organization of the partition into blocks and the computation of the value $\bar{u}(i, j)$ specify the reduction method: subsampling, interpolation (Karabassis and Spetsakis 1995; Duchon 1979), aggregation (Beliakov et al. 2012), or F-transform (below).

The principal difference (and novelty) of the proposed F-transform-based reduction is that the blocks in the corresponding partition *overlap* (the details are in Sect. 4.1). Algorithm without overlapping (subsampling, Beliakov et al. 2012) computes the value of a block independently of surrounding blocks. Overlapping computes value of one block even with connection to other surrounding blocks. This feature allows us to achieve better value of MSE than the best aggregation-based reduction.

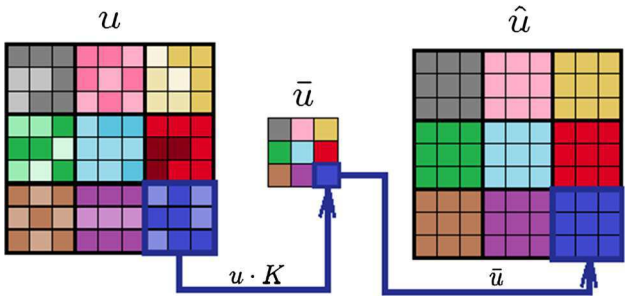

Fig. 1 Scheme of image reduction and reconstruction

A *quality of reduction* Q is measured by a criterion that expresses how different a reconstructed image $\hat{u}$ (which is enlarged from the reduced image to the original size) is from the original one. As the evaluation criteria MSE, PEN, and SSIM were chosen. These metrics need the same size domain of both images; therefore, a reconstruction to a reduced image is applied. In this paper, we use two types of reconstruction: the simplest one and the "inverse" one with respect to a corresponding reduction. The simplest reconstruction is performed based on the "pixel-to-block" scheme. In other words, each pixel (i, j) of the reduced image $\bar{u}$ is enlarged into an $N_b \times M_b$-sized block $B_{i,j}$. The value of the reconstructed image function over the block $B_{i,j}$ is identical to $\bar{u}(i, j)$. Figure 1 shows scheme of described general image reduction via interpolation (see (4)) with pixel-to-block reconstruction.

The three criteria, MSE, PEN, and SSIM, are chosen to estimate the quality of the reconstructed image and compare among various reduction methods including our proposed method. We characterize each criterion and explain its selection below.

Mean square error (MSE)

$$\mathrm{MSE}(u, \hat{u}) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} (u(i, j) - \hat{u}(i, j))^2}{NM}, \tag{1}$$

where u denotes the original image, and $\hat{u}$ denotes its reconstruction. MSE is the squared Euclidean distance between u and $\hat{u}$ in the corresponding vector space. For example, if u is a color image in the R, G, B scheme, i.e., $u = (u^R, u^G, u^G)$, then the arithmetic mean of three MSEs in the three color bands is used as a criterion (see Beliakov et al. 2012). It is not difficult to show that if the MSE is used as a criterion, reduction (reconstruction) is performed based on the "block-to-pixel" ("pixel-to-block") scheme, and the blocks are disjoint, then the optimal reduction is the arithmetic mean aggregation (see Proposition 1 in the Appendix for the proof).

Thus, under the given conditions, the best reduction (from the MSE viewpoint) is produced using the arithmetic mean of a corresponding block. Therefore, only methods, which break the condition "disjoint blocks, can have better MSE than the arithmetic mean aggregation.

Penalty-based error (PEN)

$$\mathrm{PEN}(u,\hat{u}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\sum_{c\in\{R,G,B\}}|u^c(i,j)-\hat{u}^c(i,j)|\right)^2}{3NM}, \quad (2)$$

where $u = (u^R, u^G, u^G)$ denotes the original color image, and $\hat{u} = (\hat{u}^R, \hat{u}^G, \hat{u}^G)$ denotes its reconstruction. PEN was introduced in Beliakov et al. (2012) as a penalty function to choose the best aggregation operation and use it for reduction. PEN was specially designed for color images and was aimed at measuring the reduction quality of the aggregation. It cannot be decomposed into a sum of penalties for each color band. Unlike the MSE case, there is no analytical expression for the minimizer of PEN. Therefore, the optimal aggregation with respect to PEN can only be found using a numeric method. Finally, PEN was chosen for the objectivity of the comparison with the reduction methods in Beliakov et al. (2012).

Structure similarity index (SSIM)

$$\mathrm{SSIM} = [\ell(u,\hat{u}]^{\alpha}\cdot[c(u,\hat{u})]^{\beta}\cdot[s(u,\hat{u})]^{\gamma} \quad (3)$$

This equation is a compound function that includes the measures of luminance ℓ, contrast c, and structure s. The detailed expression of SSIM and particular choosing of $\alpha, \beta, \gamma > 0$ (see Wang et al. 2004) require extended explanations and is thus omitted. SSIM was introduced to measure the quality of a single (gray-scale) image by comparing it with the ideal representation of the same image. Similarly to MSE, the SSIM value of a color image can be obtained as an arithmetic mean of three SSIM values in the three color bands. Unlike MSE and PEN, the SSIM value measures the similarity of two images according to the principle of the higher, the better. We chose SSIM because it is consistent with the human-eye perception and is used to estimate the compression quality of JPEG and JPEG2000 (Wang et al. 2004).

There is no single objective criterion to estimate the quality of a reduced image because there is no ideally reduced sample. All criteria are applied to the reconstructed images and compare them with the original ones. For better objectivity, we chose three different criteria that are focused on various aspects of the quality: the distance from the original image (MSE), the color preservation (PEN), and the consistency with the human-eye perception (SSIM).

3 Image reduction methods

There are two principal techniques for image reduction: interpolation and approximation. Although interpolation methods are more popular, we argue that approximation methods are more computationally efficient because interpolation is more restricted than approximation. Moreover, the computed pixels (that replace blocks) in the reduced image approximate those (in the corresponding blocks) in the original.

On the other side, interpolation is preferable in the task of reconstruction (magnification) because it preserves the pixels in the reduced image and completes them using additional ones.

In this study, we focus on the image reduction problem and consider the reconstruction task as complementary because it is required to estimate the quality. We propose to apply the F-transform technique to image reduction (and reconstruction) because it successfully combines approximation (when reduction is performed, see Sect. 4.2) and interpolation (when reconstruction is produced, see Remark 4). Moreover, F-transform-based reduction and reconstruction are mutually inverse in some sense, which contributes to the good quality of this method (see our following tests).

In this section, we provide a short overview of reduction methods that are commonly used to make reductions.

3.1 Image reduction via interpolation

Interpolation methods are widely used in image rescaling and particularly image reduction. In our tests, we used three (most popular) interpolation methods: bilinear, bicubic, and Lanczos. All of them have a kernel representation. Therefore, in this section, we will briefly characterize interpolation-based reduction methods with a kernel representation.

We accept the scheme "block-to-pixel" and assume that the reduction has the ratio ρ. Then a reduced image $\bar{u}$ via interpolation can be calculated as follows:

Algorithm *INT* of image reduction via interpolation
Inputs: $N \times M$ image u, reduction ratio ρ.
Output: Reduced image $\bar{u}$.
Step 1. Divide domain $[1, N] \times [1, M]$ of the image function u into $N_b \times M_b$ disjoint blocks $B_{1,1}, \ldots, B_{n,m}$ such that $N_b \cdot M_b = \rho$.
Step 2. Specify the interpolation method (e.g., bilinear, bicubic, Lanczos, etc.) and formally represent it in the form of a discrete convolution of image samples with the interpolation kernel K:

$$S(x, y) = \sum_{l,m} u(l, m) K(x - l) K(y - m). \tag{4}$$

Step 3. Inside each block $B_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, choose interpolating nodes (l, m) (their positions and numbers are specified by the method) and compute the corresponding interpolation function (4) at a certain point $(x_i, y_j) \in B_{i,j}$ according to (4).
Step 4. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, set

$$\bar{u}(i, j) = S(x_i, y_j). \tag{5}$$

Because of the smoothness of the kernel K, the computed value $\bar{u}(i, j)$ is close to the values of $u(l, m)$ at the interpolating nodes $(l, m) \in B_{i,j}$. Moreover, it is easily observed from (4) and (5) that the value $\bar{u}(i, j)$, which is assigned to block $B_{i,j}$, aggregates values $u(l, m)$ where $(l, m) \in B_{i,j}$. In this respect, reduction via interpolation is a particular case of reduction via aggregation, which we explain below.

3.2 Image reduction via aggregation

An aggregation function of l variables in $[0, 1]$ is a function that is non-decreasing in each argument and idempotent at the boundaries $(0, \ldots, 0)$ and $(1, \ldots, 1)$ (see Grabisch et al. 2009). Well-known examples are the minimum, arithmetic mean, and maximum. Three algorithms for image reduction via aggregation were proposed in Beliakov et al. (2012). Each algorithm chooses the best aggregation among $k \geq 2$ possibilities (in the tests, $k = 5$ aggregations are used: minimum, geometric mean, arithmetic mean, median, and maximum) and applies it for reduction. The best aggregation is the one that minimizes a certain penalty function (Calvo and Beliakov 2010). The following two aspects are essential for algorithms in Beliakov et al. (2012): (1) the tested images are colored according to the RGB scheme; and (2) the penalty function PEN (2) aggregates first over colors and subsequently over pixels. Therefore, the proposed approach cannot be split into two steps: the reduction of an image in each color constituent and the aggregation of single colored images. Below, we will refer to the reduction algorithms in Beliakov et al. (2012) to the *aggregation-based image reduction algorithms*.

The following algorithm [the first reduction algorithm from Beliakov et al. (2012)] is selected as the referential.

Algorithm *AGG* of image reduction via aggregation
Inputs: $N \times M$ image u, reduction ratio ρ.
Output: Reduced image $\bar{u}$.
Step 1. Divide the domain $[1, N] \times [1, M]$ of the image function u into $N_b \times M_b$ disjoint blocks $B_{1,1}, \ldots, B_{n,m}$ such that $N_b \cdot M_b = \rho$.
Step 2. Specify the penalty function P and choose the aggregation functions $Ag_1, \ldots, Ag_k$.
Step 3. For each block $B_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, find the aggregation function $Ag_{i,j}$ (among $Ag_1, \ldots, Ag_k$) that minimizes P and compute the value $Ag_{i,j}$ over $B_{i,j}$.
Step 4. For each $i = 1, \ldots, n$, $j = 1, \ldots, m$, set

$$\bar{u}(i, j) = Ag_{i,j}. \tag{6}$$

Besides Algorithm *AGG*, there are two other image reduction algorithms that have been proposed in Beliakov et al. (2012). Although all three algorithms in Beliakov et al. (2012) are different, their quality measures including MSE (1) and PEN (2) are almost equal. On all test images, the differences in quality measures are mostly in decimals, whereas the ranges of intensity values are within the interval $[0, 255]$. Therefore, if a quality of a reduced image is identified with any measure from Beliakov et al. (2012), then all three algorithms in Beliakov et al. (2012) are equally preferable. Thus, we only use one Algorithm *AGG* in the sequel as the referential.

Remark 1 If the quality of a reduced image is identified with MSE and the blocks in the reduction/reconstruction scheme are disjoint, by Proposition 1 (see Appendix), the best reduction is produced by the arithmetic mean over a corresponding block. Because the arithmetic mean is included into the set of aggregation functions $Ag_1, \ldots, Ag_k$ (cf. *Step 2*), Algorithm *AGG* should choose it on the *Step 3* and use it for reduction. This observation shows that for MSE, Algorithm *AGG* is not required because the best reduction method is known in advance from the theoretical considerations.

To illustrate Remark 1, we compare the MSE values of two reduction algorithms: *AGG* and reduction via arithmetic mean aggregation *Ar mean*. In *AGG*, the arithmetic mean is one of the possible aggregations used for reduction, whereas in *Ar Mean*, the arithmetic mean is the only aggregation. Both algorithms were applied to color images from the data-

Table 1 MSE of *AGG* and *Ar mean*

Img no.	Alg *AGG*	*Ar mean*
1	243	**242**
2	**46**	**46**
3	**47**	**47**
4	172	**170**
5	126	**124**
6	192	**191**
7	79	**78**
8	122	**121**
9	302	**301**
10	420	**416**
11	344	**342**
Mean	190.27	**188.90**

base[3] that was used in Beliakov et al. (2012). The comparison results are shown in Table 1.

It is evident that for all tested images, the reduction algorithm *Ar mean* has better (lower) or equal MSE values than the algorithm *AGG*.

4 F-transform for functions of one variable

In this section, we introduce a new modification of the F-transform method and explain the difference from the original version in Perfilieva (2006). According to Perfilieva (2006), the F-transform of a function F is determined by a fuzzy partition of the domain of F. The proposed modification consists of using fewer axioms in the definition of the fuzzy partition. In particular, we drop three of them: normality, convexity, and orthogonality (the latter is also known as the Ruspini condition). As a result, a newly defined generalized fuzzy partition has additional degrees of freedom that can be tuned.

4.1 Generalized fuzzy partitions

A *generalized fuzzy partition* appeared in Perfilieva et al. (2009) in connection with the notion of a higher-degree F-transform. Its even weaker version was implicitly introduced in Hurtik and Perfilieva (2013) to satisfy the requirements of image compression. We summarize both of these notions and propose the following definition:

Definition 1 Let $[a, b]$ be an interval on the real line $\mathbb{R}$, $n \geq 2$, and let $x_1, \ldots, x_n$ be nodes such that $a \leq x_1 < \ldots < x_n \leq b$. Let $[a, b]$ be covered by the intervals $[x_k - h'_k, x_k + h''_k] \subseteq [a, b]$, $k = 1, \ldots, n$, such that their left and right margins $h'_k, h''_k \geq 0$ fulfill $h'_k + h''_k > 0$.

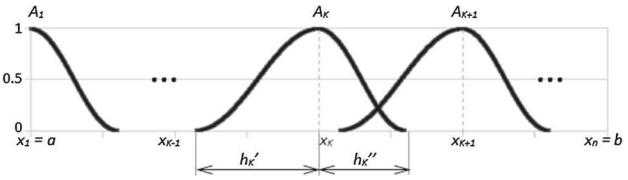

Fig. 2 Generalized fuzzy partition $A_1, \ldots, A_n$ of $[a, b]$ with nodes $x_1, \ldots, x_n$ and margins h'_k, h''_k, $k = 1, \ldots, n$

The basic functions $A_1, \ldots, A_n : [a, b] \to [0, 1]$[4] constitute a *generalized fuzzy partition* of $[a, b]$ (with nodes $x_1, \ldots, x_n$ and margins h'_k, h''_k, $k = 1, \ldots, n$) if for every $k = 1, \ldots, n$, the following three conditions are fulfilled:

1. (*locality*)—$A_k(x) > 0$ if $x \in (x_k - h'_k, x_k + h''_k)$, and $A_k(x) = 0$ if $x \in [a, b] \setminus (x_k - h'_k, x_k + h''_k)$;
2. (*continuity*)—A_k is continuous on $[x_k - h'_k, x_k + h''_k]$;
3. (*covering*)—for $x \in [a, b]$, $\sum_{k=1}^{n} A_k(x) > 0$.

It is important to remark that by the conditions of *locality* and *continuity*,

$$\int_a^b A_k(x)\mathrm{d}x > 0.$$

A generalized fuzzy partition is illustrated in Fig. 2.

In this work, we omit the word "generalized" whenever we refer to a fuzzy partition. Moreover, we assume that in every partition that is considered below, the basic functions $A_1, \ldots, A_n$ are *normalized* in the sense that $A_k(x_k) = 1$, $k = 1, \ldots, n$. A fuzzy partition $A_1, \ldots, A_n$ is called *Ruspini* if the following condition holds for all $x \in [a, b]$:

$$\sum_{k=1}^{n} A_k(x) = 1.$$

We claim that a fuzzy partition $A_1, \ldots, A_n$ is (h, h')-*uniform* if the nodes are h-equidistant and the margins (except for h'_1, h''_n) are equal, i.e., for all $k = 1, \ldots, n$, $x_k = a + h(k - 1)$, where $h = (b - a)/(n - 1)$, and $h''_1 = h'_2 = \cdots = h''_{n-1} = h'_n = h'$ where $h' > h/2$. Moreover, two additional properties are satisfied: 4. $A_k(x) = A_{k-1}(x - h)$ for all $k = 2, \ldots, n - 1$ and $x \in [x_k, x_{k+1}]$, and $A_{k+1}(x) = A_k(x - h)$ for all $k = 2, \ldots, n - 1$ and $x \in [x_k, x_{k+1}]$. 5. $h'_1 = h''_n = 0$ and for all $k = 2, \ldots, n - 1$ and all $x \in [0, h']$, $A_k(x_k - x) = A_k(x_k + x)$.

Another possibility to define an (h, h')-uniform fuzzy partition is to use the notion of *generating function*. Recall

[3] http://decsai.ugr.es/cvg/dbimagenes/index.php.

[4] A basic function can be considered a membership function of a corresponding fuzzy set. Thus, the partition is called "fuzzy".

Perfilieva et al. (2009) that a function $A_0 : [-1, 1] \to [0, 1]$ is called generating if it is even,[5] continuous, and positive everywhere except on the boundaries, where it vanishes. If the basic functions $A_1, \ldots, A_n$ with nodes $x_1, \ldots, x_n$ establish a (h, h')-uniform fuzzy partition, then they are shifted copies of a corresponding generating function A_0 in the sense that the nodes are h-equidistant and

$$A_1(x) = \begin{cases} A_0\left(\frac{x-x_1}{h'}\right), & x \in [x_1, x_1 + h'], \\ 0, & \text{otherwise}, \end{cases}$$

and for $k = 2, \ldots, n-1$,

$$A_k(x) = \begin{cases} A_0\left(\frac{x-x_k}{h'}\right), & x \in [x_k - h', x_k + h'], \\ 0, & \text{otherwise}. \end{cases}, \tag{7}$$

$$A_n(x) = \begin{cases} A_0\left(\frac{x-x_n}{h'}\right), & x \in [x_n - h', x_n], \\ 0, & \text{otherwise}. \end{cases}$$

For example, the function

$$A_0(x) = 1 - |x|, \tag{8}$$

is a generating function of a uniform triangular-shaped partition, and the function

$$A_0(x) = \cos\left(\frac{\pi x}{2}\right), \tag{9}$$

is a generating function of a uniform sinusoidal-shaped partition.

The illustration of (h, h')-uniform fuzzy partitions where $h = 3$ and $h' = 2$ is in Fig. 3.

4.2 Direct F-transform

In this section, we define the integral and discrete (direct) F-transform according to Perfilieva (2006) and recall its useful properties for image reduction. We assume that the universe is an interval $[a, b]$ and $x_1 < \cdots < x_n$ are fixed nodes from $[a, b]$ such that $x_1 = a$, $x_n = b$ and $n \geq 2$. Let $A_1, \ldots, A_n$ be basic functions that form a fuzzy partition of $[a, b]$ according to Definition 1. The latter will be fixed throughout this Section. Let $C([a, b])$ be the set of continuous functions on the interval $[a, b]$. The following definition introduces the integral F-transform of a function $f \in C([a, b])$.

[5] The function $A_0 : [-1, 1] \to \mathbb{R}$ is even if for all $x \in [0, 1]$, $A_0(-x) = A_0(x)$.

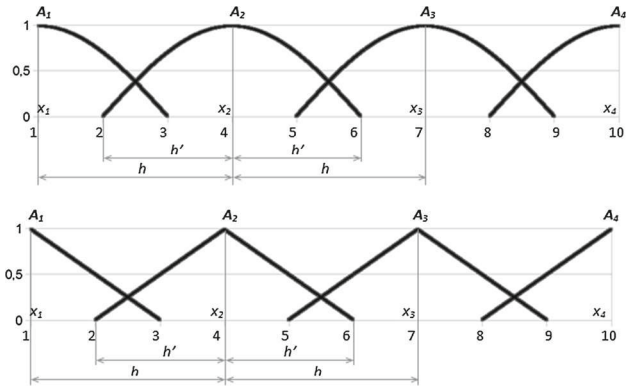

Fig. 3 Generalized (3, 2)-uniform fuzzy partitions: sinusoidal-shaped (*top*) and triangular-shaped (*bottom*)

Definition 2 Let $A_1, \ldots, A_n$ be basic functions that form a fuzzy partition of $[a, b]$ and f be any function from $C([a, b])$. We say that the n-tuple of real numbers $\mathbf{F}[f] = (F_1, \ldots, F_n)$ given by

$$F_k = \frac{\int_a^b f(x)A_k(x)\mathrm{d}x}{\int_a^b A_k(x)\mathrm{d}x}, \quad k = 1, \ldots, n, \tag{10}$$

is the integral F-transform of f with respect to $A_1, \ldots, A_n$.

The discrete form of the F-transform is applied to functions f that are defined on a finite set $P = \{p_1, \ldots, p_l\} \subseteq [a, b]$. We assume that the set P is *sufficiently dense with respect to the fixed partition*, i.e.,

$$(\forall k)(\exists j)A_k(p_j) > 0.$$

Then, the discrete F-transform $\mathbf{F}[f] = (F_1, \ldots, F_n)$ of f is defined as follows:

$$F_k = \frac{\sum_{j=1}^{l} f(p_j)A_k(p_j)}{\sum_{j=1}^{l} A_k(p_j)}, \quad k = 1, \ldots, n. \tag{11}$$

The elements $F_1, \ldots, F_n$ are called *components of the F-transform*. If $A_1, \ldots, A_n$ is an (h, h)-uniform Ruspini partition, then the expression (10) may be simplified as follows:

$$\begin{aligned} F_1 &= \frac{2}{h}\int_{x_1}^{x_2} f(x)A_1(x)\mathrm{d}x, \\ F_n &= \frac{2}{h}\int_{x_{n-1}}^{x_n} f(x)A_n(x)\mathrm{d}x, \\ F_k &= \frac{1}{h}\int_{x_{k-1}}^{x_{k+1}} f(x)A_k(x)\mathrm{d}x, \quad k = 2, \ldots, n-1. \end{aligned} \tag{12}$$

The following list of properties of the F-transform of f (with respect to $A_1, \ldots, A_n$) is used.

(a) If for all $x \in [a,b]$, $f(x) = C$, then $F_k = C$, $k = 1,\dots,n$;
(b) If $f = \alpha g + \beta h$, then $\mathbf{F}[f] = \alpha\mathbf{F}[g] + \beta\mathbf{F}[h]$;
(c) If $[c,d] = \{f(x) \mid x \in [a,b]\}$, then $F_k = \arg\min_{[c,d]} \int_a^b (f(x)-y)^2 A_k(x)\mathrm{d}x$, $k = 1,\dots,n$;
(d) Let the fuzzy partition $A_1,\dots,A_n$ be (h,h')-uniform, where $h/2 < h' \le h$. Then for each $k = 1,\dots,n-1$,

$$|f(t) - F_k| \le 2\omega(h,f), \quad |f(t) - F_{k+1}| \le 2\omega(h,f),$$

where $t \in [x_k, x_{k+1}]$, and

$$\omega(h,f) = \max_{|\delta|\le h} \max_{x\in[a,b-\delta]} |f(x+\delta) - f(x)|, \tag{13}$$

is the modulus of continuity of f.

Thus, the F-transform is a result of a linear map $\mathbf{F}$ between a set of continuous/discrete functions and the set of n-dimensional (real) vectors [property (b)]. Each component F_k of the F-transform of f is a weighted local mean of the f values over an area covered by the basic function A_k [property (c)]. Moreover, the difference between f and F_k in the area covered by A_k is restricted by the modulus of continuity of f, i.e., a smoother f corresponds to a smaller difference [property (d)].

Remark 2 The F-transform is fully characterized by the corresponding fuzzy partition. If the latter is uniform, then it has a generating function (it is also called a kernel). As it is shown in Theorem 2 below, the influence of a kernel is tangible if the corresponding F-transform is applied to a non-smooth function. In the image reduction, a kernel is an additional parameter that can be involved in optimization.

4.3 Inverse F-transform

The inverse F-transform establishes a backward correspondence from the set of n-dimensional vectors to the set of continuous/discrete functions. This correspondence is not the inverse of the direct F-transform, but if both are sequentially applied, the result approximates the original function.

Definition 3 Let $A_1,\dots,A_n$ be basic functions that form a generalized fuzzy partition of $[a,b]$ and f be a function from $C([a,b])$. Let $\mathbf{F}[f] = (F_1,\dots,F_n)$ be the F-transform of f with respect to $A_1,\dots,A_n$. Then, the function $\hat{f} : [a,b] \to \mathbb{R}$, which is represented by

$$\hat{f}(x) = \frac{\sum_{k=1}^n F_k A_k(x)}{\sum_{k=1}^n A_k(x)}, \quad x \in [a,b], \tag{14}$$

is called *the inverse F-transform*.

In the discrete case, the inverse F-transform $\hat{f}$ is defined using the same expression (14) that is applied to set P, where the original discrete function was defined.

Remark 3 If a fuzzy partition of $[a,b]$ fulfills the Ruspini condition, then the inversion formula (14) can be simplified to

$$\hat{f}(x) = \sum_{k=1}^n F_k A_k(x)).$$

Theorem 1 demonstrates that the inverse F-transform $\hat{f}$ approximates a continuous function f with arbitrary precision. Thus, the F-transform can be successfully applied to image reduction. If reduction is associated with the direct F-transform and reconstruction is associated with the inverse one, then Theorem 1 assures that the reconstruction is close to the original image.

Theorem 1 *Let f be a continuous function on $[a,b]$. Then, for any $\varepsilon > 0$, there exists h_ε such that for any $h_\varepsilon/2 < h' \le h_\varepsilon$ and any (h_ε, h')-uniform generalized fuzzy partition of $[a,b]$, the corresponding inverse F-transform $\hat{f}_\varepsilon$ of f fulfills*

$$|f(x) - \hat{f}_\varepsilon(x)| \le \varepsilon, \quad x \in [a,b]. \tag{15}$$

The proof of this theorem is in the Appendix.

Theorem 2 shows that it is sufficient to compute the F-transform with respect to the simplest fuzzy partitions: triangular (8) or sinusoidal (9). In the following tests, we attempted both and subsequently chose sinusoidal functions because they showed slightly better results (see Sect. 6).

Theorem 2 *Let f be any continuous function on $[a,b]$, and let $A'_1,\dots,A'_n$ and $A''_1,\dots,A''_n$, for $n \ge 3$ be basic functions that form different (h,h')-uniform fuzzy partitions of $[a,b]$ where $h = \frac{b-a}{n-1}$ and $h/2 < h' \le h$. Let $\hat{f}'$ and $\hat{f}''$ be the two inverse F-transforms of f with respect to different sets of basic functions $A'_1,\dots,A'_n$ or $A''_1,\dots,A''_n$. Then, for arbitrary $x \in [a,b]$,*

$$|\hat{f}'(x) - \hat{f}''(x)| \le 4\omega(h,f),$$

where $\omega(h,f)$ is the modulus of continuity (13) of f on the interval $[a,b]$.

The proof of Theorem 2 can be easily obtained from the proof of the analogous theorem in Perfilieva (2006).

Remark 4 It follows from (14) that if a fuzzy partition of $[a,b]$ is (h,h')-uniform and $h/2 < h' \le h$, then the inverse F-transform works as an interpolation on a set of data points $\{(x_k, F_k) \mid k = 1,\dots,n\}$ where $x_1,\dots,x_n$ are nodes of the fuzzy partition and $F_1,\dots,F_n$ are corresponding components of the direct F-transform. This fact puts the inverse

F-transform in line with the interpolation methods when the enlargement problem is considered.

5 New F-transform based image reduction

Image compression was the first application of the F-transform to image processing. In Perfilieva (2006), we proposed to represent a compressed image by a matrix of F-transform components, which is computed over a uniform Ruspini partition of the image domain. The reconstruction to a full-size image was performed using the inverse F-transform. This method (which we call the "simple F-transform-based compression") does not take advantage of any property of the original image; therefore, its quality is not notably high. In Di Martino et al. (2008), Perfilieva and De Baets (2010), and Hurtik and Perfilieva (2013), we proposed another compression method and proved that a proper choice of a fuzzy partition improves the quality of the reconstructed image.

In this section, we introduce a new F-transform-based image reduction algorithm based on a generalized fuzzy partition. Similar to the case of compression, tuning the fuzzy partition leads to better reduction results. However, what is good for compression cannot be blindly applied to reduction. Furthermore, the reduced image should maintain the proportions of the original one. Therefore, all blocks in the reduction scheme "block-to-pixel" should have identical sizes. This condition is not the case of image compression where the sizes of the reduced areas may vary, which explains why we cannot apply the technique of Di Martino et al. (2008), Perfilieva and De Baets (2010) and Hurtik and Perfilieva (2013) to image reduction. To achieve high-quality results, we propose to use the flexibility of a generalized fuzzy partition and find parameters that guarantee the optimal solution. A generalized fuzzy partition is based on *overlapping blocks*, and this feature has been remarked in the Introduction as one of the novelties of the F-transform-based reduction. Because of this property, we achieved better results than interpolation- and aggregation-based reductions from the MSE viewpoint (see Table 5 in Sect. 6.1 and Table 8 in Sect. 6.2).

We will show that the F-transform method based on a specially designed (generalized) fuzzy partition is the most suitable reduction method from both quality (measured by MSE, PEN, SSIM) and complexity viewpoints. To support this claim, we compare the results of the new F-transform-based reduction with those obtained using the interpolation method and aggregation algorithms in Beliakov et al. (2012) (see Sect. 6).

5.1 Proposed algorithm and its complexity

We introduce a new F-transform-based reduction algorithm to apply to an image function $u : [1, N] \times [1, M] \to [0, 255]$, where the domain and the range contain only natural numbers. The following expression for the F-transform components $U_{kl}, k = 1, \ldots, n, l = 1, \ldots, m$, of u is a direct generalization of (11):

$$U_{kl} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} u(i,j)A_k(i)B_l(j)}{\sum_{i=1}^{N}\sum_{j=1}^{M} A_k(i)B_l(j)}. \tag{16}$$

In (16), it is assumed that the basic functions $A_1, \ldots, A_n$ ($B_1, \ldots, B_m$) establish a fuzzy partition of $[1, N]$ ($[1, M]$) and the set $[1, N]$ ($[1, M]$) is sufficiently dense with respect to $A_1, \ldots, A_n$ ($B_1, \ldots, B_m$).

The algorithm is described in terms of the procedures, i.e., without unnecessary technical details.

Algorithm *FT* of image reduction based on the F-transform with a generalized fuzzy partition
Inputs: $N \times M$ image u, reduction ratio ρ.
Output: Reduced image $\bar{u}$.
Step 1. Find values $n, m \geq 2$ such that $\frac{NM}{nm} = \rho$. Let $h_x = \frac{N-1}{n-1}$, $h_y = \frac{M-1}{m-1}$.
Step 2. Choose n h_x-equidistant nodes $x_1, \ldots, x_n \in [1, N]$ and m h_y-equidistant nodes $y_1, \ldots, y_m \in [1, M]$.
Step 3. Choose the margins h'_x and h'_y and the generating functions A_{0x} and A_{0y} and establish (h_x, h'_x)- and (h_y, h'_y)-uniform fuzzy partitions of $[1, N]$ and $[1, M]$, respectively.
Step 4. Compute the F-transform components U_{kl}, $k = 1, \ldots, n$, $l = 1, \ldots, m$, of u based on (16) and arrange them into the matrix $\mathbf{F}[u]$. Take $\mathbf{F}[u]$ as the output reduced image $\bar{u}$.

We estimate the complexity of Algorithm *FT* and use it as an additional argument in favor of the F-transform-based reduction. We claim that the complexity is *linear* with respect to the length of the input. To justify the claim, we estimate the complexity of the main *Step 4*. According to (16), the F-transform component U_{kl} can be computed over the pixels (i, j) that are "covered by" the product $A_k B_l$, i.e., those that fulfill $A_k(i)B_l(j) > 0$. Because of the uniformity of the partition, the number of such pixels depends on the initial choice of the margins h'_x and h'_y and is a constant characteristic of the partition. Therefore, there is a constant number C of operations that are involved in the computation of each component U_{kl}. Consequently, the total number of operations required by *Step 4* is equal to Cnm or CNM/ρ. Thus, the complexity of *Step 4* is indeed *linear* with respect to the product NM or the length of the input.

5.2 Optimal choice of parameters

In this section, we analyze the parameters of Algorithm *FT* to choose their optimal values that minimize MSE and PEN

and maximize SSIM. For any input image, the output of the algorithm (reduced image comprised of the F-transform components) is fully determined by the choice of the fuzzy partition. Thus, the optimal values of the parameters of Algorithm *FT* are actually the optimal values of the parameters of a generalized fuzzy partition. We use the notation of Algorithm *FT* below.

We assume that the input image u is defined on a square domain so that $N = M$, and the number of basic functions in a fuzzy partition of $[1, N]$, which is determined by the input ratio ρ, i.e., $n = \frac{N}{\sqrt{\rho}}$, is an integer. The distances between nodes on both axes are assumed to be identical, i.e., $h_x = h_y = h$, and h is determined by the values of N and n or equivalently, by N and ρ (see *Step 1*). The generating functions A_{0x} and A_{0y} are *sinusoidal*.

Only two free parameters remain: the margins h'_x and h'_y. We assume that they are equal, i.e., $h'_x = h'_y = h'$, so that only one optimal value of h' (if exists) that minimizes MSE and PEN and maximizes SSIM should be chosen. To find the optimal value of h', we tested eleven images from the set of available color images at the web.[6] These images have been proposed in Beliakov et al. (2012). In these test images, the optimal value of h' is identical for all three quality measures.

Let us be more specific: the test include images with resolution $N = M = 1024$ and the requirement is $\rho = 9$. Number of components is determined as $n = 1024/\sqrt{9} = 341$ (the number 341 is rounded). For that $h_x = h_y = h = 1023/340 = 3$. The same computation is computed for the rest of $N = M$ used in the test: 256, 512, and 2250, for all cases $h = 3$.

In Tables 2, 3 and 4, we show the quality measures MSE, PEN, and SSIM for Algorithm *FT*, where the reduction ratio is $\rho = 9{:}1$ and the reconstruction is "pixel-to-block". The other parameters are as follows: the distance between nodes is $h = 3$, and the margins are $h' = 2, 3, 4$. Except for $h' = 3$ (Ruspini partition), the fuzzy partitions with $h' = 2, 4$ are generalized partitions. The optimal values of the quality measures are highlighted.

It is immediate from Tables 2, 3 and 4 that the uniform $(3, 2)$-fuzzy partition (where $h = 3$ and $h' = 2$) is the optimal partition with respect to the chosen input images and the quality measures MSE, PEN and SSIM.

In Figs. 4 and 5, two color images (number 5 and 8) illustrate the reductions produced by Algorithm *FT*, where $h' = 2, 3, 4$.

From both Figs. 4 and 5, it is visible that the sharpest reduction corresponds to the margin value $h' = 2$. With the previously discussed parameters, this value is used in Algorithm *FT* to compare with other algorithms.

Table 2 MSE for Algorithm *FT*

Img no.	$h' = 2$	$h' = 3$	$h' = 4$
1	**241**	262	307
2	**45**	48	56
3	**46**	49	57
4	**164**	178	210
5	**123**	131	154
6	**191**	205	237
7	**77**	83	97
8	**121**	131	158
9	**290**	307	348
10	**414**	445	523
11	**343**	367	414
Mean	**186.82**	200.55	232.82

Table 3 PEN for Algorithm *FT*

Img no.	$h' = 2$	$h' = 3$	$h' = 4$
1	**704**	764	896
2	**129**	138	161
3	**133**	141	162
4	**408**	447	530
5	**299**	321	381
6	**514**	554	642
7	**192**	208	244
8	**279**	303	366
9	**778**	821	929
10	**1051**	1136	1339
11	**974**	1042	1170
Mean	**496.45**	534.09	620.00

Table 4 SSIM for Algorithm *FT*

Img no.	$h' = 2$	$h' = 3$	$h' = 4$
1	**0.95**	0.94	0.93
2	**0.98**	**0.98**	0.97
3	**0.98**	**0.98**	0.97
4	**0.93**	**0.93**	0.91
5	**0.98**	0.97	0.97
6	**0.95**	**0.95**	0.94
7	**0.98**	0.97	0.97
8	**0.98**	**0.98**	**0.98**
9	**0.94**	0.93	0.92
10	**0.94**	0.93	0.92
11	**0.88**	0.87	0.85
Mean	**0.95**	0.94	0.93

[6] http://decsai.ugr.es/cvg/dbimagenes/index.php.

Fig. 4 Image 5 (*left*) and its 9:1-reductions, which correspond to $h' = 2$ (*top-right*), $h' = 3$ (*middle-right*), and $h' = 4$ (*bottom-right*)

Fig. 5 Image 8 (*left*) and its 9:1-reductions, which correspond to $h' = 2$ (*top-right*), $h' = 3$ (*middle-right*), and $h' = 4$ (*bottom-right*)

6 Comparison with interpolation and aggregation-based image reduction algorithms

In this section, various reduction algorithms are compared on the data set that contains 53 color images taken from Sipi Database[7] (further on "the data set" $D53$). The images have different resolutions: 512×512 px (20 pieces), 256×256 px (8 pieces), 1024×1024 px (24 pieces), and 2250×2250 px (one piece). The reduction ratio is $\rho = 9:1$, and the reconstruction is performed based on two schemes: "pixel-to-block" (Sect. 6.1) and "corresponding-to-reduction" (Sect. 6.2). The following algorithms were tested: interpolation *INT* (bilinear *bl*, bicubic *bc* and Lanczos *Lns*), 2*AGG* and F-transform *FT* with the optimal setting values: $h = 3$ and $h' = 2$. The algorithms are compared based on (1) three quality measures: MSE, PEN and SSIM (all are applied to the reconstructed images), (2) the computation time (Sect. 6.3), and (3) the noise-removing ability (Sect. 6.4).

[7] http://sipi.usc.edu/database/database.php/volume=textures.

Table 5 MSE for *FT*, INT_{bl}, INT_{bc}, INT_{Lns}, *AGG*

Stat	FT	INT_{bl}	INT_{bc}	INT_{Lns}	AGG
Min	**32.6**	43.6	44.7	41.9	**32.6**
Q_1	**81.2**	113.7	113.8	121.4	82.0
Median	**102.1**	145.6	150.0	170.8	103.7
Mean	**146.5**	197.3	199.6	212.0	152.4
Q_3	**163.0**	246.3	242.0	262.6	167.3
Max	**517.0**	606.0	607.0	626.7	527.3

Table 6 PEN for *FT*, INT_{bl},INT_{bc}, INT_{Lns}, *AGG*

Stat	FT	INT_{bl}	INT_{bc}	INT_{Lns}	AGG
Min	75.7	101.8	104.2	97.9	**75.4**
Q_1	**207.8**	273.3	273.4	283.9	210.3
Median	**260.6**	374.9	386.6	415.6	262.5
Mean	**375.0**	506.7	512.2	546.8	387.3
Q_3	**424.9**	640.2	651.4	693.6	434.6
Max	**1328.8**	1617.1	1673.9	1728.3	1367.7

Table 7 SSIM for *FT*, INT_{bl}, INT_{bc}, INT_{Lns}, *AGG*

Stat	FT	INT_{bl}	INT_{bc}	INT_{Lns}	AGG
Min	**0.84**	0.79	0.79	0.76	0.83
Q_1	**0.89**	0.85	0.85	0.85	**0.89**
Median	**0.92**	0.90	0.90	0.88	**0.92**
Mean	**0.92**	0.89	0.89	0.89	**0.92**
Q_3	**0.95**	0.93	0.93	0.93	**0.95**
Max	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**

For brevity, we display the standard statistics of MSE, PEN and SSIM for all five algorithms. In other words, for each algorithm, we compute three sets of corresponding quality measures; each set contains 53 elements, each of which corresponds to an image. For every set of quality measures, we compute its standard statistics: the minimal element, the 1st quantile (Q_1), the 2nd quantile (median), the mean value, the 3rd quantile (Q_3), and the maximal value.

6.1 "Pixel-to-block" reconstruction

From Tables 5, 6 and 7, it follows that the F-transform-based reduction is better than Algorithm *AGG* and significantly better than the interpolation methods (in each table, the best quality measures are printed in bold). The F-transform superiority (even from the MSE viewpoint) is based on a different approach to the partition of a domain of an original image, i.e., the blocks are not disjoint, and their overlapping is adjusted to the chosen reduction.

Table 8 MSE for *FT*, INT_{bl},INT_{bc}, INT_{Lns}, *AGG*

Stat	FT	INT_{bl}	INT_{bc}	INT_{Lns}	AGG
Min	30.6	33.2	**29.7**	32.0	32.6
Q_1	**67.4**	84.5	71.0	70.9	82.0
Median	97.8	110.5	**96.3**	99.7	103.7
Mean	**131.6**	158.8	131.9	134.8	152.4
Q_3	141.8	165.6	**141.6**	142.7	167.3
Max	512.4	582.8	**509.0**	514.3	527.3

Table 9 PEN for *FT*, INT_{bl}, INT_{bc}, INT_{Lns}, *AGG*

Stat	FT	INT_{bl}	INT_{bc}	INT_{Lns}	AGG
Min	71.0	77.3	**69.0**	74.8	75.4
Q_1	176.1	214.5	**170.1**	181.8	210.3
Median	**237.1**	**273.7**	242.5	243.2	262.5
Mean	336.4	405.4	**334.5**	345.5	387.3
Q_3	367.3	425.6	**328.1**	363.1	434.6
Max	**1285.9**	1501.9	1332.8	1348.7	1367.7

Table 10 SSIM for *FT*, INT_{bl},INT_{bc}, INT_{Lns}, and *AGG*

Stat	FT	INT_{bl}	INT_{bc}	INT_{Lns}	AGG
Min	0.83	0.80	**0.84**	**0.84**	0.83
Q_1	**0.90**	0.88	**0.90**	**0.90**	0.89
Median	0.92	0.92	**0.93**	**0.93**	0.92
Mean	0.92	0.91	**0.93**	**0.93**	0.92
Q_3	**0.96**	0.95	**0.96**	**0.96**	0.95
Max	**0.99**	0.98	**0.99**	**0.99**	0.98

Table 11 CPU time

Resolution (px)	*AGG* (ms)	*FT* (ms)	INT_{bl} (ms)
256×256	**3**	**3**	**3**
512×512	12	**9**	**9**
1024×1024	46	37	**34**
2250×2250	256	**121**	149

6.2 Reconstruction that corresponds to reduction

In this section, we compare the same algorithms on the same data set $D53$ of color images as above. We only change the reconstruction algorithms: the inverse F-transform was selected for the F-transform-based reduction, and the corresponding interpolation methods were selected for the interpolation-based reductions. Thus, if reduction is performed using a bilinear interpolation, the same method is used for reconstruction. The "pixel-to-block" reconstruction was left for the aggregation-based reduction.

In Tables 8, 9 and 10, we show the standard statistics of MSE, PEN, and SSIM for all five algorithms. We observe that the F-transform-based reduction has better MSE and PEN quality measures if its corresponding reconstruction is made by the inverse F-transform and not by the "pixel-to-block" method. Moreover, the F-transform-based reduction has better MSE and PEN quality measures than Algorithm *AGG* and the bilinear and Lanczos interpolation methods. It has similar quality with the bicubic interpolation, which on the other side, has higher complexity (see Sect. 6.3). The quality measure SSIM remains similar for all methods. As above, the best quality measures are printed in bold.

6.3 Computation time

In Sects. 6.1 and 6.2, we saw that if the quality is measured by MSE or PEN, then the F-transform-based reduction is visibly better than any interpolation-based reduction and slightly better than Algorithm *AGG*. However, if the quality is measured by SSIM, then all five reduction methods are similar. In this section, we compare the computation time of three reduction methods: the F-transform-based method, the aggregation-based method, and the bilinear interpolation, which have similar MSE and PEN. We chose the bilinear interpolation, because it is the fastest interpolation method among the three considered. We observe that the average computation time of the F-transform-based reduction (setting values are $h = 3$ and $h' = 2$) is twice less than that of Algorithm *AGG*. The effect begins to show in images of higher resolution. The theoretical justification is provided in Sect. 5.1, where we estimate the complexity of the F-transform-based reduction and show that it is linear—each pixel in a reduced image is computed from $(2h' + 1)^2$ pixels. By computation we mean multiplication (pixel intensity value with membership degree), which is one of the fastest operation on a processor. At the same time, the F-transform method is as fast as the bilinear interpolation. In Table 11, we show the averaged CPU time (in seconds) of all three algorithms that are applied to the data set $D53$. The algorithms were processed on Dell XPS 13 with $\rho = 9 : 1$.

6.4 Noise-removing ability

Because all image reduction methods work as low-pass filters, they remove the noise, which is another criterion by which we can compare their effectiveness.

All five algorithms were tested on noised images and compared based on the quality measures MSE, PEN, and SSIM. The noised inputs were created for all images from $D53$ as follows: we took an original image and added 30 % of random color noise using Corel Paint Shop Pro. The obtained noised images were then processed by the tested algorithms and reconstructed by the "pixel-to-block" scheme.

Table 12 MSE, PEN and SSIM for noised images

Stat	MSE	PEN	SSIM
Min	984.2	2202	0.38
Q_1	1297.5	3220	0.61
Median	1329.9	3307	0.74
Mean	1333.4	3312	0.74
Q_3	1407.3	3524	0.87
Max	1532.8	3832	0.94

Table 13 MSE for *FT, INT*$_{bl}$, *INT*$_{bc}$, *INT*$_{Lns}$, and *AGG*

Stat	FT	INT$_{bl}$	INT$_{bc}$	INT$_{Lns}$	AGG
Min	194	**126**	148	212	195
Q_1	247	**199**	225	289	245
Median	**269**	285	310	330	273
Mean	**310**	331	354	379	316
Q_3	334	395	418	426	**331**
Max	**676**	782	799	773	692

Table 14 PEN for *FT, INT*$_{bl}$, *INT*$_{bc}$, *INT*$_{Lns}$, *and AGG*

Stat	FT	INT$_{bl}$	INT$_{bc}$	INT$_{Lns}$	AGG
Min	448	**289**	339	485	433
Q_1	578	**470**	525	667	562
Median	639	682	729	800	**623**
Mean	739	813	858	910	**733**
Q_3	800	999	1025	1059	**775**
Max	**1668**	2034	2055	2020	1698

Reconstructed images were compared with the original ones using standard statistics of the quality measures. The results are shown in Tables 13, 14, and 15 (the best quality measures are printed in bold). To stress the noise removing ability of the reduction methods, we compute identical standard statistics (Table 12) of identical quality measures for noised (versus original) images and compare Table 12 with Tables 13, 14, and 15. For example, the mean value of MSE decreased from 1333.4 to 310 after the F-transform-based reduction was applied.

Tables 13, 14, and 15 evidently show that the two algorithms *FT* and *AGG* have similar mean values of all three quality measures. Moreover, these mean values are visibly better than their counterparts of the interpolation algorithms.

In Fig. 6, we demonstrate an example of original, noised, and reduced/reconstructed images after being processed by all tested algorithms.

Remark 5 Let us remark that the noise-removing ability is not a primary focus of reduction methods. Therefore, we cannot expect that a method, which is suitable for reduction, overcomes specially designed noise removing filters.

Table 15 SSIM for *FT, INT*$_{bl}$, *INT*$_{bc}$, *INT*$_{Lns}$, *and AGG*

Stat	FT	INT$_{bl}$	INT$_{bc}$	INT$_{Lns}$	AGG
Min	**0.76**	0.74	0.73	0.68	0.75
Q_1	**0.83**	0.80	0.80	0.79	**0.83**
Median	**0.87**	0.86	0.85	0.83	**0.87**
Mean	**0.87**	0.86	0.85	0.84	**0.87**
Q_3	**0.93**	0.91	0.91	0.91	**0.93**
Max	**0.97**	0.96	0.96	**0.97**	**0.97**

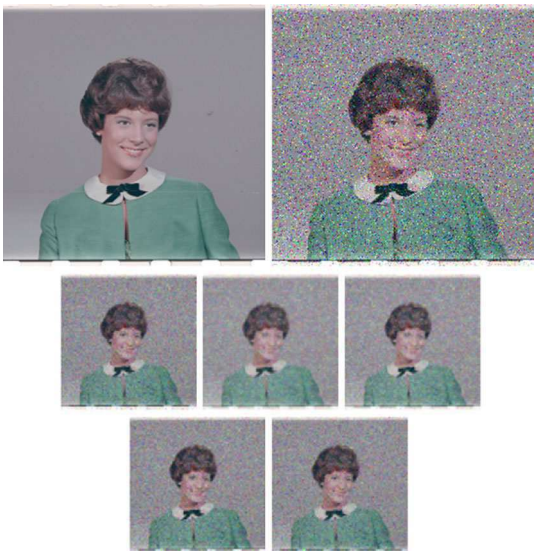

Fig. 6 From *left* to *right*, from *top* to *bottom*: original, 30 % noised, and reduced images after being processed by *FT, INT*$_{bl}$, *INT*$_{bc}$,*INT*$_{Lns}$, *and AGG.*

6.5 Reduction with sharpening

There are some cases where emphasizing of details is needed, for example on X-ray images, space images, car plate photos, etc. One way to emphasize details is a local contrast enhancement. The contrast can be enhanced using nonlinear filters (Polesel et al. 2000), usually based on gradient operators. The F-transform-based reduction can be extended into sharpening version too. Figure 7 shows several reductions of an image taken from the Hubble telescope.[8] The original image resolution 7800 × 3900 px was reduced into 260 × 130 px (30× smaller). The figure demonstrates that after the sharpening modification there is a better visibility of stars, nebula contours, and the image looks like that consisting of more details.

[8] http://hubblesite.org/gallery/.

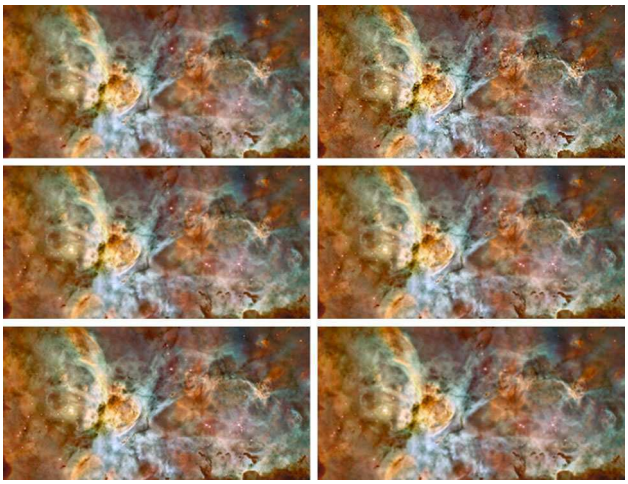

Fig. 7 Reduced image from Hubble telescope. From *left* to *right*, from *top* to *bottom*: FT, FT with sharpening, INT_{bl}, INT_{bc},INT_{Lns}, and AGG

7 Conclusion

A new method for (color) image reduction was introduced based on the F-transform that uses a generalized fuzzy partition. We showed that the F-transform-based reduction with a generalized fuzzy partition is very much suitable for the image reduction.

To support this claim, we compared the results of the new F-transform-based reduction algorithm with those of interpolation (the most efficient methods were selected) and aggregation. The comparison is performed based on (1) three quality measures MSE, PEN, and SSIM, (2) the computation time, and (3) the noise removing ability. The comparison showed that

- if a quality is measured by MSE or PEN, then the F-transform-based reduction is better or the second best than all interpolation- or aggregation-based reductions; if it is the second best then it faster than the best;
- if a quality is measured by SSIM, then all considered reduction methods are similar;
- *FT* and *AGG* have better noise-removing effectiveness than the interpolation algorithms;
- the computation time of the F-transform-based reduction is twice less than that of Algorithm *AGG* and as fast as the bilinear interpolation.

We estimated the complexity of the F-transform-based reduction and proved that it is linear with respect to the length of the input.

The run time of the F-transform based reduction is smaller than that of the bilinear or bicubic interpolation, whereas the quality results are comparable, or even better.

Acknowledgments Support was provided by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

Compliance with ethical standards

Conflict of interest The authors declare that they have no potential conflict of interest.

Appendix

The proposition below shows that if MSE measures the quality of reduction, and the reduction (reconstruction) is performed on the basis of the "block-to-pixel" ("pixel-to-block") scheme, and the blocks are disjoint, then the optimal reduction is the arithmetic mean aggregation over a corresponding block.

Proposition 1 *Let $u : N \times M \to [0, 255]$ be an image function, $\bar{u} : n \times m \to [0, 255]$ a reduced representation of u, and $\rho = \frac{NM}{nm}$ a reduction ratio. Assume that $N = M$, $n = m$ and $\frac{N}{n} = \sqrt{\rho}$ is a natural number which we denote by d. Let $B_{1,1}, \ldots, B_{n,n}$ be disjoint $d \times d$ blocks that partition the domain of u. Let $\hat{u} : N \times N \to [0, 255]$ be a reconstructed image where the reconstruction follows the scheme "block-to-pixel", i.e., if $(x, y) \in B_{i,j}$, then $\hat{u}(x, y) = \bar{u}(i, j)$. If $\hat{u}$ minimizes* MSE *in* (1)*, then*

$$\bar{u}(i, j) = \frac{1}{d^2} \sum_{(x,y)\in B_{i,j}} u(x, y), \quad i, j = 1, \ldots, n, \tag{17}$$

i.e., the value of the reduction $\bar{u}$ at each pixel (i, j) of the reduced domain is the arithmetic mean of values of u over the corresponding block $B_{i,j}$ in the domain of u.

Proof Let an image function $u : N \times M \to [0, 255]$ be fixed and all assumptions of the proposition be fulfilled. Because the domain of u is partitioned into n^2 disjoint blocks $B_{1,1}, \ldots, B_{n,n}$, we rewrite the expression for MSE as follows:

$$\mathrm{MSE}(u, \hat{u}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{(x,y)\in B_{i,j}} (u(x, y) - \hat{u}(x, y))^2}{N^2}.$$

Furthermore,

$$\mathrm{MSE}(u, \hat{u}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{(x,y)\in B_{i,j}} (u(x, y) - \bar{u}(i, j))^2}{N^2},$$

so that $\mathrm{MSE}(u, \hat{u})$ actually depends on n^2 unknown values $\bar{u}(i, j)$, $i, j = 1, \ldots, n$, i.e.,

$$\mathrm{MSE}(u, \hat{u}) = \mathrm{MSE}(\bar{u}(1, 1), \ldots, \bar{u}(n, n)).$$

By the necessary condition for the existence of a (relative) minimum, all partial derivatives of MSE by $\bar{u}(i,j)$, $i,j = 1,\ldots,n$, should be equal to 0. This leads to the following system of n^2 linear equations:

$$\sum_{(x,y)\in B_{i,j}} (u(x,y) - \bar{u}(i,j)) = 0, \quad i,j = 1,\ldots,n.$$

Because there are d^2 pixels in each block $B_{i,j}$, we easily come to Eq. (17). □

We repeat the formulation of Theorem 1 and give its proof.

Theorem 3 *Let f be a continuous function on $[a,b]$. Then, for any $\varepsilon > 0$, there exist h_ε such that for any $h_\varepsilon/2 < h' \le h_\varepsilon$ and any (h_ε, h')-uniform fuzzy partition of $[a,b]$, the corresponding inverse F-transform $\hat{f}_\varepsilon$ of f fulfills*

$$|f(x) - \hat{f}_\varepsilon(x)| \le \varepsilon,\ x \in [a,b].$$

Proof Let us choose some $\varepsilon > 0$. By assumption, the function f is continuous and thus, uniformly continuous on $[a,b]$. Therefore, for the chosen ε we can find $h_\varepsilon > 0$ such that for all $x', x'' \in [a,b]$, $|x' - x''| < h_\varepsilon$ implies $|f(x') - f(x'')| < \varepsilon/2$. Let us assume that the value $(b-a)/h_\varepsilon$ is an integer (otherwise we choose the least natural number n_ε such that $n_\varepsilon > 2$ and $(b-a)/(n_\varepsilon - 1) \le h_\varepsilon$) and find the h_ε-equidistant nodes $x_1,\ldots,x_{n_\varepsilon} \in [a,b]$, where $n_\varepsilon = (b-a)/h_\varepsilon + 1$, such that $a = x_1 < \cdots < x_{n_\varepsilon} = b$. Then we choose $h_\varepsilon/2 < h' \le h_\varepsilon$ and establish a (h_ε, h')-uniform fuzzy partition of $[a,b]$ determined by the chosen nodes and constituted by basic functions $A_1,\ldots,A_{n_\varepsilon}$.

Let us proof that the inverse F-transform $\hat{f}_\varepsilon$ of f fulfills the requested inequality. For this purpose, we choose some $x \in [a,b]$ and find k such that $x \in [x_k, x_{k+1})$.

Let $F_1,\ldots,F_{n_\varepsilon}$ be the components of the F-transform of f w.r.t. basic functions $A_1,\ldots,A_{n_\varepsilon}$. By the property (d) from the list on the page 7, for the chosen $x \in [x_k, x_{k+1})$, we have

$$|f(x) - F_k| \le 2\omega(h_\varepsilon, f) < \varepsilon,$$

and analogously,

$$|f(x) - F_{k+1}| < \varepsilon,$$

where we used the fact that $h' \le h_\varepsilon$. Therefore, for the chosen x we can write the chain of inequalities:

$$|f(x) - \hat{f}_\varepsilon(x)| = \left| f(x) - \frac{\sum_{i=1}^{n_\varepsilon} F_i A_i(x)}{\sum_{i=1}^{n_\varepsilon} A_i(x)} \right|$$
$$= \frac{|f(x)\sum_{i=1}^{n_\varepsilon} A_i(x) - \sum_{i=1}^{n_\varepsilon} F_i A_i(x)|}{\sum_{i=1}^{n_\varepsilon} A_i(x)}$$
$$\le \frac{\sum_{i=1}^{n_\varepsilon} A_i(x)|f(x) - F_i|}{\sum_{i=1}^{n_\varepsilon} A_i(x)}.$$

Because $h_\varepsilon/2 < h' \le h_\varepsilon$, there are at most two values $A_k(x)$ and $A_{k+1}(x)$ that may be different from zero. Therefore,

$$|f(x) - \hat{f}_\varepsilon(x)| \le \frac{\sum_{i=1}^{n_\varepsilon} A_i(x)|f(x) - F_i|}{\sum_{i=1}^{n_\varepsilon} A_i(x)}$$
$$= \frac{\sum_{i=k}^{k+1} A_i(x)|f(x) - F_i|}{\sum_{i=1}^{n_\varepsilon} A_i(x)}$$
$$< \varepsilon \frac{\sum_{i=k}^{k+1} A_i(x)}{\sum_{i=1}^{n_\varepsilon} A_i(x)} = \varepsilon \frac{\sum_{i=1}^{n_\varepsilon} A_i(x)}{\sum_{i=1}^{n_\varepsilon} A_i(x)} = \varepsilon.$$

Because x was chosen arbitrary inside $[a,b]$, the chain of inequalities proves the claim. □

References

Beliakov G, Bustince H, Paternain D (2012) Image reduction using means on discrete product lattices. Image Process IEEE Trans 21(3):1070–1083

Calvo T, Beliakov G (2010) Aggregation functions based on penalties. Fuzzy Sets Syst 161(10):1420–1436

Di Martino F, Loia V, Perfilieva I, Sessa S (2008) An image coding/decoding method based on direct and inverse fuzzy transforms. Int J Approx Reason 48(1):110–131

Di Martino F, Loia V, Sessa S (2010) A segmentation method for images compressed by fuzzy transforms. Fuzzy Sets Syst 161(1):56–74

Duchon CE (1979) Lanczos filtering in one and two dimensions. J Appl Meteorol 18(8):1016–1022

Grabisch M, Marichal J, Mesiar R, Pap E (2009) Aggregation functions. Cambridge University Press, Cambridge

Hurtik P, Perfilieva I (2013) Image compression methodology based on fuzzy transform. In: International joint conference CISIS12-ICEUTE′12 special sessions. Springer, pp 525–532

Hwang JW, Lee HS (2004) Adaptive image interpolation based on local gradient features. IEEE Signal Process Lett 11:359–362

Karabassis E, Spetsakis ME (1995) An analysis of image interpolation, differentiation, and reduction using local polynomial fits. Graph Models Image Process 57(3):183–196

Liu X, Ahmad F, Yamazaki K, Mori M (2005) Adaptive interpolation scheme for nurbs curves with the integration of machining dynamics. Int J Mach Tools Manuf 45(4):433–444

Perfilieva I (2006) Fuzzy transforms: theory and applications. Fuzzy Sets Syst 157(8):993–1023

Perfilieva I, De Baets B (2010) Fuzzy transforms of monotone functions with application to image compression. Inf Sci 180(17):3304–3315

Perfilieva I, Danková M, Bede B (2009) Towards f-transform of a higher degree. In: IFSA/EUSFLAT conference, Citeseer, pp 585–588

Polesel A, Ramponi G, Mathews VJ et al (2000) Image enhancement via adaptive unsharp masking. IEEE Trans Image Process 9(3):505–510

Thévenaz P, Blu T, Unser M (2000) Image interpolation and resampling. In: Handbook of medical imaging, processing and analysis, pp 393–420

Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. Image Process IEEE Trans 13(4):600–612

N. Madrid and P. Hurtik. Lane departure warning for mobile devices based on a fuzzy representation of images. *Fuzzy Sets and Systems*, 291, 144-159, 2016.

Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems 291 (2016) 144–159

FUZZY sets and systems

www.elsevier.com/locate/fss

Lane departure warning for mobile devices based on a fuzzy representation of images

Nicolás Madrid *, Petr Hurtik

Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic

Received 1 December 2014; received in revised form 8 September 2015; accepted 15 September 2015

Available online 9 October 2015

Abstract

The use of driver assistance systems is a current trend in the car industry. However, most driver assistance systems require the use of multiple sensors, increasing the cost of their implementation in cars and making them inaccessible to many users. In this paper, we present an "*efficient*" Lane Departure Warning system for mobile devices, which is real-time, accurate and accessible to most users. On the one hand, the accuracy of the system relies on Line Detection based on Hough Transforms, a combination that has proven to be reliable for this goal in many approaches. On the other hand, the reduction of the computational time, to achieve real-time processing, is due mainly to a proposed fuzzy representation of images. Finally, we also present various tests that show that the system proposed runs at up to 20 FPS on a mobile device (with a dual-core CPU at 1.0 GHz) with a resolution of 320×240 px.
© 2015 Elsevier B.V. All rights reserved.

Keywords: Fuzzy image; Hough Transform; Image processing; Lane detection; Android; Intelligent transportation system; Driver assistant

1. Introduction

The proper detection of lanes is crucial for the development of autonomous cars and driver assistance systems. For instance, the development of *Lane Departure Warning Systems* [13], *Drive Attention Monitoring Systems* [23], and *Lane Tracking Systems* [7,27] depends strongly on the correct determination of the boundaries of a road. Despite most of activating driver tasks requiring the use of multiple sensors, such as laser scanners (LIDARs), differential GPS, stereo-cameras, sonars, omni-directional cameras, gyroscopes etc., in most *Lane Detection Systems*, the only sensor required is a vision camera; one exception is [20], which also requires radar for the detection of trials.

Lane detector systems can be roughly described by focusing mainly on two features: road assumptions and feature extraction. In road assumptions, the system models the structure and geometry of a road by assuming various features in roadways. For instance, it can be assumed that roadways are on a flat plane [7,15,32] or on a manifold [24], that lane marks are white on a dark background [23], that right and left boundaries are parallel [4], etc. In some approaches,

* Corresponding author.
E-mail addresses: nicolas.madrid@osu.cz (N. Madrid), petr.hurtik@osu.cz (P. Hurtik).

http://dx.doi.org/10.1016/j.fss.2015.09.009
0165-0114/© 2015 Elsevier B.V. All rights reserved.

these assumptions are described explicitly, while they are implicit in the underlying methods of other approaches. On the one hand, these assumptions simplify, from a computational point of view, the positioning of the lane boundaries, but on the other hand, they restrict the scope of the systems; for instance, the assumptions used to model highways with several lanes are not valid when modeling an urban road. According to road assumptions, lane detectors can be split in two groups: those focused on *structured roads* that allow and impose strong road assumptions (such as streets or highways where lane marks are usually clearly visible) and those focused on *off roads* that require weak road assumptions [16,20] (such as trails or mountain paths). This paper is framed concerning the former group. The way a lane detector system extracts the features of a lane depends strongly on the road assumptions. To the best of our knowledge, lane detector systems based on structured roads have a common strategy aimed at detecting lane marks. The procedure usually applies in some step a binarization procedure, mainly an edge detector [33,26,15] or an intensity thresholding [30,7]. Subsequently, most systems search for the boundaries of a lane by means of assumed shapes of lane marks (such as straight lines [7,30], parabolas [17], clothoids [24], etc.) by Hough Transforms [32, 30,26], Bayesian models [7,17,15,20], Mathematical morphology [31,18] (extracting roads from satellite images) or other methods [13,33,15].

A common drawback among Lane Detector Systems is the hard computational cost and/or hardware designed specifically for such a task. Only a few approaches have focused on a procedure in real-time.[1] The first approach dealing with the real-time challenge was [5] in 1995. Their system was implemented in *PAPRICA*, a massively parallel architecture, and they reached a speed of 17 FPS with a resolution of 128×128 (pixels per frame). Subsequently, in 1998, the system was modified to cover the detection of obstacles as well [4], reaching a speed of 10 FPS with the same resolution. Later, in 2011, [27] developed a system that also reached 10 FPS (with a resolution of 200×400) via a PC implementation (Intel dual-core E5300, 2.5 GHz). Recently, various approaches have implemented lane detectors on FPGA hardware [21,1,11] with great efficiency. Specifically, in terms of computational time, the system in [11] runs at 25 FPS with a resolution of 256×256, the one in [21] runs at 30 FPS with a resolution of 752×480, and in the case of [1], the system runs at 40 FPS with a resolution of 752×320. Last but not least, [25] shows an implementation on image processing specific hardware (Texas Instruments C6x DSP running at 600 MHz) reaching up to 160 FPS with a resolution of 720×480. It is worth mentioning that the specific hardware used in the approaches above is not installed in mobile devices, which should process all the data via a CPU.

Among all possible systems related to lane detection, we focus in this paper on *Lane Departure Warning Systems*. Specifically, we present a reliable system implemented on a CPU that sends, in real-time, a warning to the driver when the car is outside of a lane of the highway. The system has been tested on a Notebook (Intel i7M, 3.1 GHz) and on a mobile device (dual-core 1.0 GHz). In both cases, real-time performance is convincingly reached: 26 FPS on the Notebook and up to 20 FPS on the mobile device, with a resolution of 640×480 and 320×240, respectively. To the best of our knowledge, the only Lane Detection System implemented on mobile phones in the literature is that of [26],[2] which runs at only 1 FPS. The improvement in the computational cost is achieved mainly because of a novel fuzzy representation of images. The idea is to link a fuzzy set to each pixel to represent the uncertainty associated with the intensity assigned to it by the image. This representation is quite similar to the one given in [14,6] in terms of intervals but with a remarkable difference namely, the storage of the intensity assigned originally by the image. The extraction of such a representation (which can, and must, be called fuzzification) is an extra procedure that is not performed in other *Lane Departure Warning Systems*. Thus, it appears at first sight that our approach would require more computational time than others. However, the reality is different, as computing some operations from the fuzzification is considerably faster. For instance, we show that our gradient operator (included the fuzzification) is obtained more than 2 times faster than Sobel and Prewitt gradients for 3×3 windows and 5 times faster for 5×5 windows. In addition, the fuzzy representation also allows the statement "*the intensity of a pixel is C*" to be coded in fuzzy terms. This is useful when we search for white pixels in lane marks that are distorted for different reasons.

The structure of the rest of the paper is as follows. To write as self-contained an article as possible, Section 2 presents some basic notions regarding image processing and fuzzy sets. Subsequently, in Section 3, we show how to

[1] For us, real-time is at least an analysis of 10 FPS (i.e., the borders of the roadway are determined in less than 100 ms including preprocessing steps). Thus, papers that do not achieve this level of performance (or do not specify the computational time) have been omitted in the comparison, even if they state that their systems run in real-time.

[2] Perhaps this is the only approach fully comparable with ours, as the target of both approaches is the development of Lane Detector Systems on mobile phones.

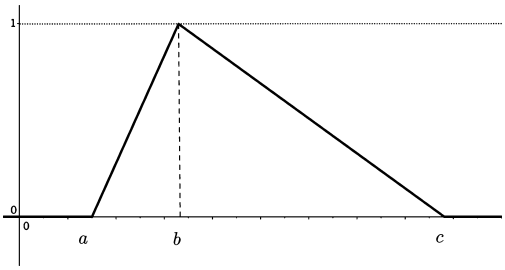

Fig. 1. Graph of a triangular membership function.

represent an image in terms of fuzzy sets and a procedure to obtain such a representation in a satisfactory computational time. Then, Section 4 recalls the mathematical basics of Hough Transforms and presents the line detection algorithm based on Hough Transforms for both standard gradient operators and our fuzzy gradient operator. A comparison between classical gradients and our fuzzy version is conducted in terms of results and computational time. In Section 5, we present a detailed explanation of our *Lane Departure Warning System*. First, we implement and compare two basic systems based on Hough Transforms by using the Sobel gradient and our fuzzy gradient. Second, we modify the basic system above to improve the accuracy and speed of the procedure by means of road assumptions. Finally, in Section 6 we present Conclusions and Future works.

2. Preliminaries

With the aim of writing a self-contained article, we briefly recall some basic notions of Fuzzy Set Theory and Image Processing in this section. Let us begin by recalling that, in Image Processing, an *image* is defined as a function $f\colon D \subseteq \mathbb{R}^2 \to L$. The domain D is commonly considered discrete (i.e. a set of indivisible points called pixels). Thus, D is usually discomposed as $D = M \times N$, where M and N denote the width and height of the image, respectively. The structure of L depends on the framework in which we are working. For instance, to represent black and white images, we can consider the binary set $L = \{0, 1\}$; in the case of 8-bit *grayscale images*, the set L is $\{0, 1, \ldots, 255\}$; and, in the case of *color images*, we can consider L as the set of triples (x, y, z) with $x, y, z \in \{0, 1, \ldots, 255\}$ to represent 24-bit *RGB* or *YUV* images. From a theoretical point of view, we will consider only grayscale images in this paper. Note that from an applied point of view we can work with color images as well by processing intensity components separately.

The notion of a fuzzy set is given as follows.

Definition 1. A fuzzy set is a pair $A = (\mathcal{U}, \mu_A)$ where $\mathcal{U}$ is a set (called the *universe*) and μ_A a mapping from $\mathcal{U}$ to the unit interval $[0, 1]$ (called the membership function).

A special type of fuzzy set is given by triangular membership functions on the universe $\mathcal{U} = \mathbb{R}$. Given $a, b, c \in \mathbb{R}$ such that $a \leq b \leq c$, we can define a triangular membership function as:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a} & \text{if } a \leq x \leq b, \\ \frac{x-c}{b-c} & \text{if } b < x \leq c, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The name ‘triangular membership’ came from the fact that the function above has the graph (see Fig. 1) which recalls the shape of a triangle. In this paper, we will use only fuzzy sets represented by triangular membership functions, hereafter called triangular fuzzy sets and denoted directly as triples, e.g., (a, b, c).

3. Fuzzification of an image

For the sake of simplicity, the fuzzification procedure is described only for grayscale images. The idea is to assign to each pixel $(x, y) \in D$ a triangular fuzzy set $f^F_{(x,y)}$ on the universe $[0, 255]$ determined as follows:

Fig. 2. Optical illusion.

- First, we consider the symmetric window $W_{x,y}$ of length $\delta > 0$ defined by:

$$W_{x,y} = \{f(x_i, y_i) \in D \mid \delta \geq |x - x_i|,\ \delta \geq |y - y_i|\}.$$

- Then, we define $f^F_{(x,y)}$ as the triangular fuzzy set given by the triple:

$$(\min(W_{x,y}),\ f(x, y),\ \max(W_{x,y})).$$

The fuzzification is based on the idea that the gray intensity assigned to one pixel has an inherent uncertainty. This uncertainty can be due to different reasons, e.g., the pixel represents an area that is not necessarily uniform and, the intensity assigned is then an average of the intensities in that area or the focus of the image is not well adapted and the intensities of the surroundings thus interfere with the real intensity of the pixel, etc. At any rate, we assume that such uncertainty can be represented by a triangular fuzzy set. Actually, this representation of images can be related to a well known human eye behavior, namely, that images are perceived as complex structures where there is an iteration between the intensities of surrounding pixels. Specifically, the gray level perceived in one pixel, by a human eye, depends on the pixels in its neighborhood. The following example shows this behavior with an image of an optical illusion and how this image is represented by our fuzzification.

Example 1. Let us consider Fig. 2. The gray level of pixels in the centered rectangle is the invariant intensity 126. However, this fact is not evident to a human eye due to the variability in the intensities of the surroundings; thus, it looks like the pixels on the left side are brighter than those on the right side. If we fuzzify the image by using our proposal, the fuzzy set assigned to pixels on the left side, in the center and on the right side of the center rectangle are (65, 126, 126), (125, 126, 127) and (126, 126, 216), respectively. Note that, similar to the human eye, the estimation of the gray level is more precise in the center than on the sides of the image because the support of the fuzzy sets increases the closer a pixel is to the sides of the image.

This fuzzification procedure is similar to those used in [14,6] but with a remarkable difference. Roughly speaking, those approaches also assume that the gray intensity assigned to each pixel is uncertain and that the uncertainty can be determined by the gray intensity of the neighborhood pixels. Keeping the notation used in the fuzzification described above, the approaches of [14,6] assign to each pixel $(x, y) \in D$ the interval value $[\min(W_{x,y}), \max(W_{x,y})]$. From our point of view, these approaches are lacking because they do not consider the proper gray intensity of each pixel to represent its uncertainty. In that way, they lose an important reference (see Example 2).

Example 2. Let us consider again Fig. 2. In the approaches of [14,6], the pixels on the left side in the centered rectangle and the adjacent ones outside the centered rectangle have the same representation, namely, [65, 126]. However, the gray intensities perceived by the human eye are not the same. Note that, in our fuzzification, the values are different, namely, (65, 126, 126) and (65, 65, 126), respectively.

Let us talk now about the computational cost of the fuzzification procedure. Note that by considering the smallest window (i.e., $\delta = 1$ or, equivalently a 3×3 window), we need to perform, a priori, 16 comparisons for each pixel to obtain the respective fuzzy set (8 to obtain $\min(W_{x,y})$ and 8 to obtain $\max(W_{x,y})$). Note also that the number of comparisons increases considerably by increasing the size of the window. In summary and roughly speaking, this means that the naive approach above has a high computational cost. To reduce the computational cost, note first that we can calculate both bounds at the same time in a parallel manner. Specifically, let us consider a window W with $2n + 1$ pixels ($n \in \mathbb{N}$); note that in our case study, every window has an odd number of pixels. Let us start by dividing W into n disjoint pairs of pixels plus one single pixel. Then, the maximum and minimum related to each pair of pixels is obtained in just one comparison. Finally, the maximum (resp. the mini-

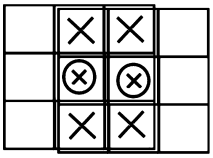

Fig. 3. Overlapping of two adjacent windows 3×3.

6	3	3	7
2	1_{WN}	1_{EN}	4
2	1_{WS}	1_{ES}	4
9	5	5	8

Fig. 4. Overlapping of four adjacent windows 3×3.

mum) of intensities in W can be reached by comparing those n maximums (n minimums) and the single pixel. This procedure reduces the total number of comparisons from $4n$ (the number of comparisons performed by the naive approach) to $3n$. Thus, the ratio of reduction of this method, applied to an odd number of pixels, is constant and exactly 25%.

The reduction above can be improved significantly yet. Note that another shortcoming of the naive approach is that it compares the same set of pixels several times. For instance, Fig. 3 shows that to compute the respective triangular fuzzy sets of two adjacent pixels (marked by o) with 3×3 windows, the naive approach compares the six pixels in the intersection of both windows (marked by $\times$) twice.

Delving deeply into that idea, let us consider 3×3 windows and four adjacent pixels forming a 2×2 square. The goal is to calculate the maximum and minimum associated with these four pixels simultaneously. By following the naive approach, the pairwise intersection of the four respective windows is compared twice, and the four original pixels are compared four times. A better strategy would be to perform local comparisons in each pairwise intersection and, finally, unify them to achieve the final result. Let us be more specific. Let us consider Fig. 4, where the four adjacent pixels are marked by the number (1) and individually by *WN* (west–north), *EN* (east–north), *ES* (east–south) and *WS* (west–south). Let us denote by (o) the set of pixels marked by the number $o \in \{1, \ldots, 9\}$ in Fig. 4. Then, the 3×3 windows centered on *WN*, *EN*, *ES* and *WS* can be respectively decomposed by:

- $W_{WN} = (1) \cup (2) \cup (3) \cup (6)$,
- $W_{EN} = (1) \cup (3) \cup (4) \cup (7)$,
- $W_{ES} = (1) \cup (4) \cup (5) \cup (8)$,
- $W_{WS} = (1) \cup (2) \cup (5) \cup (9)$.

Thus, we determine the values $\min(W_o)$ and $\max(W_o)$ with $o \in \{WN, EN, ES, WS\}$ as follows:

- First, we calculate the minimum and maximum of (1). This requires 4 comparisons, one to order the two pixels above, another to order the two pixels below and two to compare both minimums and maximums.
- Second, we calculate the minimum and maximum of the sets (2), (3), (4) and (5). Note that we only need 1 comparison to compute each pair of maximums and minimums, as each set has only two elements.
- Finally, each pair of values $\min(W_o)$ and $\max(W_o)$, with $o \in \{WN, EN, ES, WS\}$, can be computed by means of 6 comparisons by following the decomposition given above for W_{WN}, W_{EN}, W_{ES} and W_{WS}.

In summarize, with the procedure above, we compute the fuzzy sets associated with the four pixels simultaneously by using a total of 32 comparisons instead of 64, as is required by the naive approach. That is exactly a 50% reduction in comparisons.

13	12	7	7	7	16	17
10	11	3	3	3	15	14
6	2	1_{NW}	1_{N}	1_{NE}	4	8
6	2	1_{W}	1_{C}	1_{E}	4	8
6	2	1_{SW}	1_{S}	1_{SE}	4	8
22	23	5	5	5	19	18
25	24	9	9	9	20	21

Fig. 5. Overlapping of nine adjacent windows 5×5.

We can extend the procedure above for arbitrary window sizes, but this requires different decompositions. For instance, for the 5×5 case ($\delta = 2$), we can simultaneously compute the fuzzy sets associated with nine pixels by considering the decomposition associated with Fig. 5, where the nine pixels (the centers of the windows) are marked by WN, N, EN, E, ES, S, WS, W and C. Note that, for the specific cases of the windows centered on WN, S and C, we have the following respective decompositions:

- $W_{WN} = (1) \cup (2) \cup (3) \cup (6) \cup (7) \cup (10) \cup (11) \cup (12) \cup (13)$,
- $W_S = (1) \cup (2) \cup (4) \cup (5) \cup (9) \cup (19) \cup (20) \cup (23) \cup (24)$, and
- $W_C = (1) \cup (2) \cup (3) \cup (4) \cup (5) \cup (11) \cup (15) \cup (19) \cup (23)$,

where, as before, (o) denotes the set of pixels marked by the number $o \in \{1, \ldots, 25\}$ in Fig. 5. By this decomposition, we can reduce the number of comparisons to 90 instead of 216 (which means a reduction of 58%). Actually, the larger the window, the higher the reduction reached by this method.

4. Hough Transform

One of the most important tasks in lane detection is the procedure used for detecting geometrical lines. This task can be performed by using different techniques, such as Mathematical Morphology [31,18], combinatorial optimization [22], the Hough Transform [9] or the hypothesis and test paradigm [19]. Because the computational complexity plays an important role in this paper (let us recall that our final target is to implement a Lane Detector on a mobile device), we have considered the Hough Transform technique, which is a fast and reliable procedure for detecting geometrical structures via image processing. The seed of the Hough Transform is in the US-patent "*Method and Means for Recognizing Complex Patterns*" of Paul V.C. Hough. The idea germinated in the approaches of Duda and Hart in [9] and Ballard in [2]. The former presented the mathematical background that is commonly used nowadays, whereas the latter extended the procedure to arbitrary geometric primitives such as circles, ellipses, etc. Subsequently, other approaches were developed to reduce the computational complexity of the Hough Transform procedure, e.g., the *Fast Hough Transform* [10] and the *Randomized Hough Transform* [34]; it is worth noting that in the latter case, the computational complexity is reduced to the detriment of the accuracy [12].

In this paper we consider the "*standard*" approach of the Hough Transform [9] for three main reasons. First, we show in Section 5 that the standard procedure is sufficient to reach an acceptable computational speed and accuracy. Second, the use of the standard approach simplifies the presentation. Third, one of our goals is to show the improvement of the computational speed by using the fuzzification preprocessing presented in Section 3. Thus, focusing on a more complex implementation of the Hough Transform could hide such improvement. This section is divided as follows. We begin by recalling the mathematical background of the Hough Transform for line detection, then we describe the standard algorithm for line detection based on the Hough Transform, and finally, we present our proposal based on the fuzzification given in Section 3.

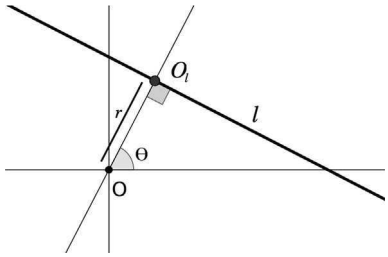

Fig. 6. The Hough Transform's representation of lines.

4.1. Mathematical background of the Hough Transform

The basis of the Hough Transform resides in identifying each line l in $\mathbb{R}^2$ with an element of the set $\{(r,\theta) \mid r \in [0,\infty)$ and $\theta \in [0,2\pi)\}$ by means of normal lines. Specifically, given a line l, the normal line of l (i.e., the line perpendicular to l that contains the origin) can be identified via the closest point of l to the origin, denoted by O_l (see Fig. 6). This identification is because, on the one hand, given a line l, the point O_l is unique, and because, on the other hand, for every point $O_l \in \mathbb{R}^2 - \{(0,0)\}$, we can associate the unique line l containing O_l and that is normal to the line determined by the origin and O_l. Thus, given a point $O_l = (r_l, \theta_l)$ with $r \in [0,\infty)$ and $\theta \in [0, 2\pi)$, we can define a unique line l by the formula

$$x\cos(\theta_l) + y\sin(\theta_l) - r_l = 0, \tag{2}$$

and every line can be described in such a way. Note that r_l represents the distance between the line l and the origin and that θ_l represents the angle between the x-axis and the normal line. Note as well that, although lines crossing the origin are not representable by means of normal lines, they are also represented by the formula (2) (cases where $r_l = 0$).

Let us consider now a set of points in the plane $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \in \mathbb{R}^2$ and let us check if they are collinear. For each (x_i, y_i), let us consider the following curve:

$$x_i\cos(\theta) + y_i\sin(\theta) - r = 0, \qquad \text{with } r \in [0,\infty) \quad \text{and} \quad \theta \in [0,2\pi). \tag{3}$$

Note that every pair of values (r,θ) in the curves above represents a line containing the point (x_i, y_i). Therefore, the set of points is collinear if and only if the entire set of curves intersects at one point. Thus, the problem of detecting collinear points can be transformed into a problem of finding concurrent curves. In Example 3, we show a simple example with three points. However, before reading such example, it is convenient to take into account the following remark.

Remark 1. Note that we can generate lines by means of the formula (3) by considering negatives values of r as well. In such a case (i.e., if $r \in \mathbb{R}$), the representation of lines is not unique, but each line is associated with two pairs of values, (r,θ) and $(-r, \pi+\theta)$ in $\mathbb{R} \times [0, 2\pi)$. Note that this duplicity can be eliminated simply by considering angles in $[0,\pi)$. The advantage of using the domain $\mathbb{R} \times [0,\pi)$ is that, for each pair of values $(x, y) \in \mathbb{R}^2$ and $\theta \in [0,\pi)$, there exists $r \in \mathbb{R}$ such that (r,θ) is a line crossing (x, y).[3] As a consequence, we can generate lines crossing a fixed point in $\mathbb{R}^2$ simply by considering θ in equation (3) as a variable and computing the respective value r. In summary, for the sake of the computation, the variables r and θ belong to an interval of the form $-R \le r \le R$ and $0 \le \theta < \pi$, respectively.

Example 3. Let us consider the three points $(1,1), (0,2)$ and $(-2,4)$ in $\mathbb{R}^2$. The curves associated with them by formula (3) are drawn in Fig. 7 on the left. Note that the three curves have a common intersection, which means that the three points are collinear. Such intersection at the point $(\pi/4, \sqrt{2})$ represents the line crossing the points $(1,1), (0,2)$ and $(-2,4)$ in the Hough Transform representation. Alternately, if we consider the three points $(1,1), (0,1)$ and $(1,3)$

[3] Note that such property of existence cannot be guaranteed in the domain $[0,\infty) \times [0,2\pi)$; take, for instance, the point $(-1,0)$ and the angle $\theta = 0$.

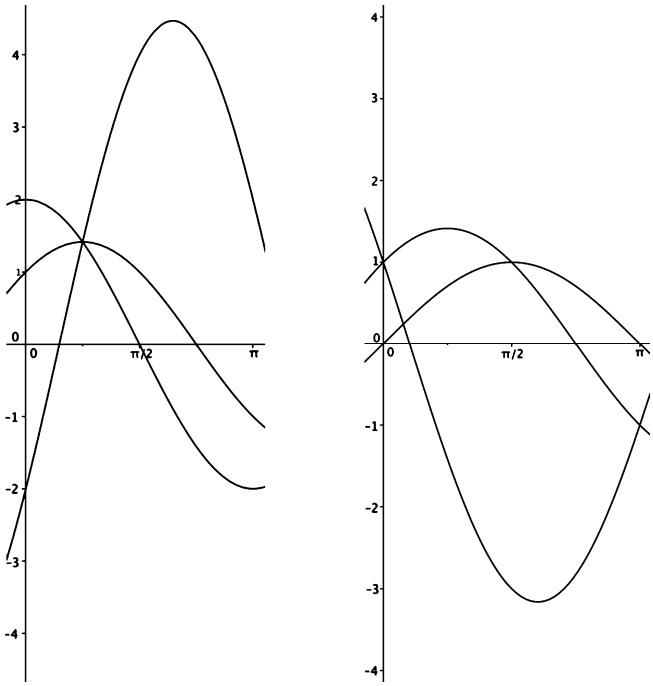

Fig. 7. Curves associated to three collinear points (left) and three non-collinear points (right).

in $\mathbb{R}^2$ and study the respective curves (Fig. 7 on the right), they do not intersect at a same point. Therefore, the points $(1, 1)$, $(0, 1)$ and $(1, 3)$ are not collinear.

To finish this section, note that if the number of points is large, computing all the lines and checking the intersections among them has a considerable computational cost. To reduce the complexity, instead of considering every line generated by the Formula (3), we can assume that the variables r and θ belong to a discrete and prefixed set of pairs in $[-R, R) \times [0, \pi)$; this set of values is called *the set of accumulators*. Note that by assuming this discretization, we assume as well that the set of accumulators represents the set of lines we can detect. For instance, in [9], the authors consider 9 angles and 170 values for r in a 120×120 image, which means they study the existence, or not, of 1530 lines in the images. Obviously, this assumption reduces the complexity to the detriment of the accuracy.

4.2. The standard Hough Transform for line detection on grayscale images

The mathematical theory behind the Hough Transform is given in set theory. As a consequence, if we want to apply such an idea to grayscale images, we need a preprocessing procedure to transform our original image into a binary one. For such a task, many different procedures can be applied, e.g., edge detection or intensity thresholding. In this approach we use gradient thresholding for the sake of the computational cost. Roughly speaking, the changes of intensities between pixels are determined by a gradient operator (for instance, Roberts, Prewitt, or Sobel), and, if the change of intensity is greater than a threshold T_G, then the pixel belongs to the edge of the image, i.e., to the binary image. In the case where we want to detect lines of a specific intensity, we need to place an extra condition for a pixel belonging to the edge of an image. Specifically, given an image $f\colon D \to L$ and an intensity $C \in L$, we say that the pixel $(x, y) \in D$ has a similar intensity to C if and only if $\|C - f(x, y)\| \leq T_C$ for a fixed threshold T_C.

The algorithm for line detection based on the Hough Transform is described in Fig. 8. The procedure detects lines of a given intensity C in a given image $f\colon D \to L$. In the initial step, the algorithm determines the accumulators and defines the function A, which will count the number of points detected in each accumulator (obviously, the value is 0 in the initial step). Then, for each pixel at the edge of the image (step 2) and with an intensity similar to C (step 3), the algorithm computes all the lines in the accumulator set crossing the pixel (steps 4 and 5). Note that in the computation of the line (r, θ), we write $\equiv$ instead of $=$. This means that the value of r is the best approximation to the value

Procedure: *HoughTrans*(f, G, T_G, T_C, C, k)
output: Straight lines detected in f
inputs: An image $f: D \to L$
G: Gradient operator to determine the edge of f
T_G: Threshold to determine the edge of f
T_C: Threshold to determine the similarity with an intensity
C: Intensity of the lines we are searching
k: Threshold to determine lines
init:
$\Theta = \{0, \ldots, \pi\}$
$R = \{r_1, \ldots, r_n\}$
$A(r, \theta) = 0$ for any accumulator $(r, \theta) \in R \times \Theta$;
1. **for each** $(x, y) \in D$
2. **if** $G(f)(x, y) \geq T_G$ **then**
3. **if** $\|f(x, y) - C\| \leq T_c$ **then**
4. **for each** $\theta \in \Theta$
5. $r \equiv x\cos(\theta) + y\sin(\theta)$;
6. $A(r, \theta) = A(r, \theta) + 1$;
7. **for each** $(r, \theta) \in R \times \Theta$
8. **if** $A(r, \theta) \geq k$ **then** include the line (r, θ) in the output
9. **end**;

Fig. 8. Hough Transform algorithm.

$x\cos(\theta) + y\sin(\theta)$ satisfying that (r, θ) is an accumulator. In step 6, we count all the accumulators obtained thus far. Finally, in step 8, if one accumulator has been calculated at least k times, which means that there are at least k pixels in such a line at the edge of the image with a similar intensity to C, we include the accumulator in the output.

4.3. Hough Transform based on our fuzzification

As we showed in the previous section, the standard algorithm for lane detection needs to create a binary image. As above, the binarization is done by means of an edge detection based on a gradient operator. The difference is that the gradient operator considered here is not a standard one, such as *Sobel*, *Prewitt* or *Laplace*, but one defined by means of the fuzzification procedure. The idea behind gradient operators is to measure the variation of a function (in this case, an image) in the surroundings of a point (in this case, a pixel). This variation can be measured from the fuzzification of an image by considering the size of the supports. Specifically, we can define the fuzzy gradient of a pixel $(x, y) \in D$ in the image $f: D \to [0, 255]$ as the value given by:

$$\nabla^F(f)(x, y) = \max(W_{x,y}) - \min(W_{x,y}). \tag{4}$$

Note that $\nabla^F(f)$ is in fact a grayscale image. Fig. 9 shows the result of applying this gradient operator to one image. Note that the result of this approach is really comparable with those obtained by the *Sobel*, *Prewitt* and *Laplace* operators. As a side note, the image obtained by the proposed fuzzy gradient is slightly more blurred. In some applications, a blurring process is sometimes applied before the gradient operator. The reason for this is that standard gradient operators are very sensitive to noise. In our experiments, our gradient has better behavior in the presence of low levels of noise. The main advantage of using this fuzzy gradient is the speed of the computation with respect to standard methods. In Table 1, we show the time required by those algorithms in *ms* according to the size of the windows and resolution of the images. In the time associated with our approach, we have included the fuzzification preprocessing as well. As the reader can see, for small images, Laplace is the only gradient operator comparable in time with our approach; in the rest of the cases, our approach is clearly faster.

This gradient operator is not novel in the literature, actually, it can be considered as a particular case of two different approaches. On the one hand, ∇^F belongs to the family of gradients defined in [14,6] aimed at detecting (binary) edges of an image. The difference with respect to our approach resides in the underlying preprocessing of the image. While [14,6] removes the original intensity assigned to each pixel, we keep it as a dominant piece of information in our fuzzification. Note that such a piece of information, codified as the central element of a triangular fuzzy set, is used in Step 7 of our Hough Transform algorithm (Fig. 10) to compute the value $f^F_{(x,y)}(C)$. Moreover, our approach provides an improvement in terms of time complexity thanks to the way the fuzzification is carried out.

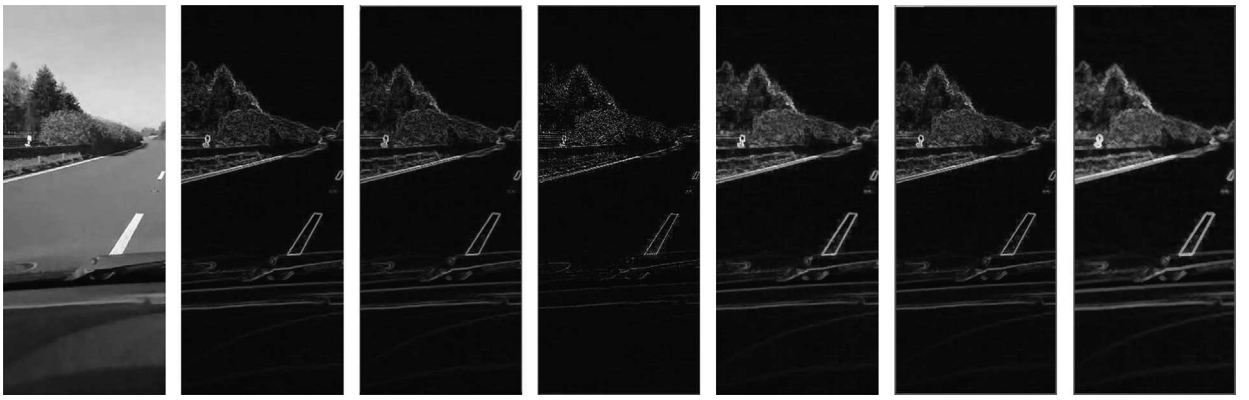

Fig. 9. From left to right: original image, Sobel, Prewitt, Laplace, fuzzy gradient and two Beucher gradients by a diamond structuring element. In each gradient, we use 3×3 windows except in the last one, where we use a 5×5 diamond.

Table 1
Gradient detection time [ms].

Algorithm	640×480 px		1920×1080 px		6000×3844 px	
	3×3	5×5	3×3	5×5	3×3	5×5
Sobel	40	118	281	781	3886	9831
Prewitt	40	109	301	812	3682	9438
Laplace	21	–	208	–	2280	–
Fuzzy gradient	22	**24**	**125**	**156**	**1383**	**1696**
Beucher Gradient Square	28	82	204	532	2254	6385
Beucher Gradient Diamond	**18**	40	126	281	1396	3171

Procedure: *FHoughTrans*(f, G, T_G, T_C, C, k)
output: Straight lines detected in f
inputs: An image $f: D \to L$
G: Gradient operator to determine the edge of f
T_G: Threshold on the gradient of f
T_C: Threshold to determine the similarity with an intensity
C: Intensity of the lines we are searching
k: Threshold to determine lines
init:
$\Theta = \{0, \ldots, \pi\}$
$R = \{r_1, \ldots, r_n\}$
$A(r, \theta) = 0$ for any accumulator $(r, \theta) \in R \times \Theta$;
1. Compute the fuzzification of f
2. **for each** $(x, y) \in D$
3. **if** $\bigtriangledown^F(f)(x, y) \geq T_G$ **then**
4. **if** $f^F_{(x,y)}(C) \geq T_C$ **then**
5. **for each** $\theta \in \Theta$
6. $r \equiv x\cos(\theta) + y\sin(\theta)$;
7. $A(r, \theta) = A(r, \theta) + f^F_{(x,y)}(C)$;
8. **for each** $(r, \theta) \in R \times \Theta$
9. **if** $A(r, \theta) \geq k$ **then** include the line (r, θ) in the output
10. **end**;

Fig. 10. Hough Transform algorithm based on the fuzzification.

On the other hand, $\bigtriangledown^F$ can be considered as well as a special case of a Beucher gradient [3]. Beucher gradients are the most common gradient operator considered in Grayscale [28,29] (also called Umbra) and Fuzzy Mathematical Morphology [8]. Specifically, and without going into detail, the identity is reached by taking the flat structuring element with support of the window used to define the fuzzy image. However, although the final result is the same, there

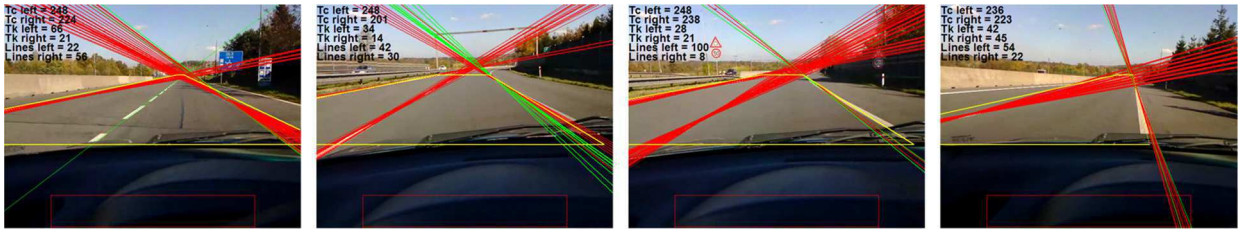

Fig. 11. Images used for test.

is an important difference with respect to Mathematical Morphology approaches, namely, the computation induced by our fuzzification. Table 1 shows that such a preprocessing (namely, the computation of f^F) saves a considerable amount of computational time. Note first that the morphological gradients (and then also Beucher gradients) are implemented, allowing arbitrary shapes of flat structuring elements. Second, note that saving time is crucial for the particular target of the paper, namely, to implement a real-time lane detector. Actually, the only Beucher gradient able to compete with our gradient with regards to time is the one obtained by using a flat structuring element with the shape of a diamond/cross. However, as we show in Section 5.1, the result is not satisfactory.

Another distinguishable feature from the standard approach is the fuzzy interpretation of the intensity of a pixel. We recall that the algorithm is defined to search lines of a specific intensity C. Thus, given a pixel (x, y) and the fuzzy set $f^F_{(x,y)}$ assigned to (x, y), the value $f^F_{(x,y)}(C)$ represents the similarity between C and the intensity assigned to the pixel (x, y) by the image. Such a value is used twice in our algorithm (Fig. 10), once to determine if "*a pixel* (x, y) *has an intensity* C" (step 4) and again to "*weight the counting*" of accumulators computed (step 7). It is worth mentioning that now in step 7, we are not exactly counting (as the standard algorithm does), as we are adding the value $f^F_{(x,y)}(C)$ instead of the value 1.

The line detection algorithm based on this fuzzy gradient and the Hough Transform is described in Fig. 10. Note the strong similarity with the standard algorithm given in Fig. 8. Actually, the three differences have already been mentioned above, namely, the fuzzification procedure carried out in the first step, the gradient operator used (step 3) and the counting of accumulators (step 7).

5. An application for lane departure warning systems

In this section, we present an application of the algorithm given in Section 4.3 for *lane departure warning systems*. As mentioned in the Introduction, the target of a lane departure warning system is to warn a driver if the vehicle goes outside of the marks demarcating the borders of a road. The application has been developed in two steps. First, we implement the standard and our Hough Transform algorithm on a notebook and compare the results, concentrating on different parameters. Subsequently, we propose an improvement of our algorithm for adaptation to the specific task of lane detection. Moreover, we show that the algorithm is reliable and sufficiently fast to be implemented on mobile devices. We use the same set of images for testing in all cases, specifically a movie of a road journey, recorded from a mobile device situated just behind the windscreen of a car and with a length of 286 frames (with 130 of them out of the borders of the line).

The video recorded by the mobile device is in the YUV color space. This means that each frame has three components, one to represent the luminance (Y) and the other two to represent the chrominance (UV). Each of those components has a component of the structure of the grayscale image.[4] Therefore, each component can be processed by our line detection algorithm independently. For the sake of computational speed, we process only the Y component. The reason is because the luminance (the information in Y) is the most important piece of information in the YUV color space, and the color (the information in U and V) does not play an important role in our lane detection algorithm. Fig. 11 shows several frames from the movie used for testing after processing.[5]

[4] It is important to point out that the bits of the Y component can be different than the bits of the U and V components.

[5] The full movie can be downloaded from http://graphicwg.irafm.osu.cz/movies/m4.avi.

Table 2
Comparison between the standard with Sobel and Beucher gradient (3×3 diamond dilatation–erosion) and fuzzy-based algorithm.

Measure	Standard (Sobel)	Standard (Beucher)	Fuzzy
FPS	6	8	10
Missing lines detected	25%	23%	11%
False lines detected	8%	22%	10%
Missed warnings	74%	54%	47%
False warnings	0%	2%	0%

5.1. First implementation on a standard desktop

This preliminary lane departure warning system algorithm has been created by slightly modifying the line detection algorithms presented in Section 4. Specifically, four different types of elements are drawn in the image: red lines, green lines, blue lines and a red box. The red lines represent lines matched by the algorithm, while the green ones are lines that are close to being detected (the respective accumulators are close to the threshold). Two blue lines are set manually to represent the front border of the car and the horizon. These two lines will be useful in the next section to improve the computation time of the algorithm. The red box is fixed and is also determined manually depending on the position of the phone with respect to the windscreen and the road. The red box represents the width of the car on the screen, as a warning is returned by the system if a red line crosses the box. It is worth mentioning that the thresholds used in the "*standard*" and "*fuzzy*" line detection algorithms are also set manually, i.e., no any adaptive control is implemented. For that reason, the success rates of both algorithms are moderate.

Table 2 shows a comparison between the standard Hough Transform (marked in the table as *standard*) and our proposed approach (marked in the table as *fuzzy*), described in Sections 4.2 and 4.3, respectively. In both cases, the results are related to the best setting possible. The implementation was written in QT C++ with the same style for both approaches and tested on a Dell XPS 13 notebook with an Intel i7-2637M @1.8 GHz processor. We measure 5 different aspects:

- *Frames per second* (FPS): this measures the average number of frames computed per second.
- *Missing lines detection*: this measures the percent of lane borders of the road that are not detected properly. The counter is incremented by one or two if the algorithm fails to detect one or both borders.
- *False lines detection*: this measures the percentage of frames where a line is detected due to different elements from lane marks (such as cars, traffic signals, etc.).
- *Missed warnings*: this measures the percentage of frames where the car is out of the lane but the algorithm does not return the warning.
- *False warnings*: this measures the percentage of frames where the system returns a warning despite the car being between the borders of the lane.

Table 2 shows that the Hough Transform algorithm based on our fuzzification yields better results than the standard one by using either a Sobel or a Beucher gradient with a flat diamond structuring element. For that reason, we focus only on the Hough Transform algorithm based on our fuzzification, in the subsequent section, as it is the best choice concerning both accuracy and speed.

5.2. A lane departure warning system on a mobile device

In this section, we present an improved version of the previous lane detection algorithm and adapt it to the following target: the development of a functional lane departure warning system on mobile devices. To reach such a goal, it is necessary to improve the reliability (by considerably reducing the missed warnings) and the speed of processing (by addressing real-time processing). Specifically, for the sake of reliability, we modify of lane detection algorithm as follows:

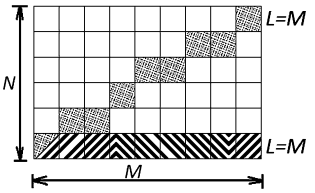

Fig. 12. Length of a diagonal and horizontal line in the discrete image.

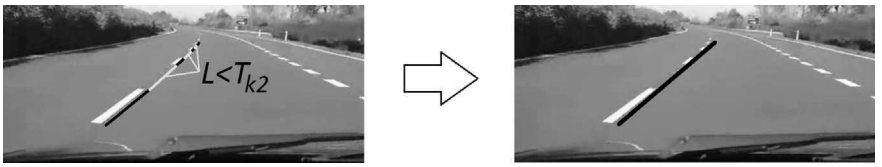

Fig. 13. Illustration of a line with gaps shorter than the threshold. Then, the line is reconstructed and detected.

- *Self-winding gradient thresholding*. The number of lines expected to be detected in a road is somewhat stable. However, due to different reasons, such as the luminosity or the appearance of other elements (e.g., cars or shadows), the number of lines detected is variable. Therefore, if the number of lines detected in a frame is very low or very high,[6] the parameter of the threshold T_G is modified in the next iteration of the algorithm.
- *Weighted increment of accumulators*. The area where we search for lines is a rectangle with a long width in comparison to the height. This means that the number of points in horizontal and diagonal lines is much larger than in vertical ones. Because of the manner by which accumulators are counted in the algorithm *FHoughTrans* (Fig. 10), it is much easier to find horizontal and diagonal lines than vertical ones. To make all of them equally accessible by the algorithm, we proceed with a pseudo-normalization that is implemented directly in the counting of accumulators by means of weights. Note that in image processing, the metric in pixels[7] is equivalent to the metric given by the maximum distance. Let us consider that the screen measurements are $M \times N$ pixels with $M \geq N$. Let us consider the diagonal of the screen l_d crossing the origin. We know that the length of l_d, in pixels, is exactly the length of the screen, i.e., M (see Fig. 12). Let $(\alpha, 0)$ be the accumulator associated with l_d. Then, for all accumulators $(\theta, 0)$, the length (in pixels) of the corresponding line is:

$$\begin{cases} N & \text{if } 0 \leq \theta \leq \frac{\pi}{4} \quad \text{or} \quad \frac{3\pi}{4} \leq \theta \leq \pi, \\ N \cdot \left|\frac{\cos(\theta-\pi)}{\sin(\theta-\pi)}\right| & \text{if } \frac{\pi}{4} \leq \theta \leq \alpha \quad \text{or} \quad \pi - \alpha \leq \theta \leq \frac{3\pi}{4}, \\ M & \text{if } \alpha \leq \theta \leq \pi - \alpha. \end{cases}$$

  The weight associated with an accumulator (θ, r) is exactly the inverse of the length of the line associated with $(\theta, 0)$. Note that only lines crossing two opposite borders of the screen are accurately normalized. However, this feature has a convenient consequence: lines close to corners are penalized. The closer to a corner and the more horizontal a line is, the more difficult it is to detect that line. Note that lane marks close to the corners of the image are irrelevant for a lane departure warning system.
- *Discontinuity detection*. Broken lines in roads are not easy to be detected for the same reason as vertical lines, i.e., they have less points than solid ones. To address this drawback, we check if the lines that are close to being detected (displayed in green in the previous version of the algorithm) are in fact discontinuous. Then, only if the check returns positive is the line considered as detected. The discontinuity detection is illustrated in Fig. 13. Two remarks: first, this step requires the inclusion of another line coding in the algorithm presented in Fig. 10, similar to line 9, but associated with another threshold $k_2 \leq k$. Second, the checking of the broken lines is not excessively complex because the number of lines to check is small and because we already know the parameterization of such lines.

[6] In the algorithm low is considered as less than 5 and high as more than 20. These two values can also be set manually.
[7] By metric in pixels, we mean the measurement of geometrical structures by the number of pixels.

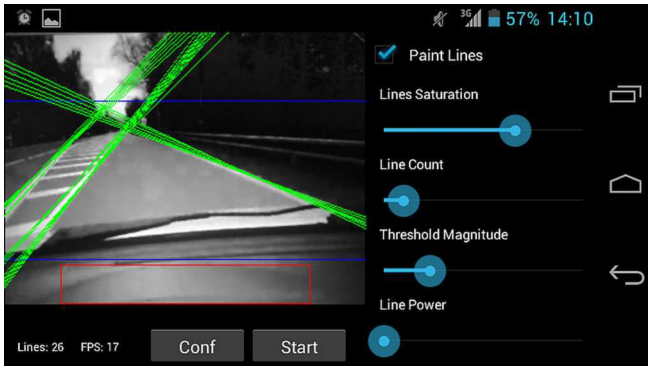

Fig. 14. Screen shot of a mobile phone while it is running the application.

Table 3
Results of the final app implemented on Android OS.

Measure	Notebook	S-Phone
FPS	26	14–20
Missing lines detected	4%	8%
False lines detected	10%	13%
Missed warnings	0%	7%
False warnings	0%	0%

- *Triple confirmation*. The warning is activated by the system only in the case that three successive frames detect the car out of the lane. Similarly, to switch off the warning, it is necessary for three successive frames to confirm that the car is in a lane again. This triple confirmation avoids the occurrence of missed and false warnings by reducing the impact of false and missed lines in line detection.

For the sake of the computational time, we include the following changes:

- *Pre-computation of trigonometric values*. The values of the sin and cos functions associated with the angles of the accumulators are pre-computed and stored.
- *Smaller area of consideration*. We reduce the area in which lines are searched. Specifically, the system only searches lines between the horizon line and the border of the car (blue-colored lines in Figs. 11 and 14).
- *Downscaled images*: The scale of the image retrieved by the mobile device is downscaled. For the test performed on the Notebook, the size of the image is reduced to 640×480 px, whereas, for the test performed on the mobile device, the reduction is elevated to 320×240 px. The downscaling is carried out by the subsampling algorithm.

This second version of the algorithm is implemented for Android devices. We have used two different languages: Java and JNI C++. Specifically, the line-detection algorithm and the image downscaling has been implemented in JNI C++, whereas the rest of the tasks (i.e., decoding the image from YUV21 to a grayscale image, drawing lines on the display, the triple confirmation warning system and adaptive thresholding) are implemented in Java with the Android SDK. The algorithm has been tested on two different devices, a notebook (Dell XPS 13 with an Intel i7-2637M @1.8 GHz processor) and a low-class smart phone (ZTE Blade G with a 1.0 GHz dual-core processor and 512 MB of RAM). These two tests show us, on the one hand, the potential of the app (the notebook test) and, on the other hand, the current status for use by the highest number of smartphone users. The results are shown in Table 3 according to the parameters described in Section 5.1. The only difference with respect to the parameters used in the previous section is that now a *Missed warning* (resp. *False warning*) only occurs when the system does not return the warning (resp. does return the warning) after three frames showing the car out of (resp. in) the lane.

Some final remarks about the results shown in Table 3:

- The algorithm is clearly in the range of being considered real-time.[8] The variability in the computation time concerning the last column is due to the OS-architecture of smartphones. However, it is worth noting that rarely is the performance below 16 FPS.
- The difference in the accuracy depends mainly on the size of the images considered for each test (640×480 px for the notebook and 320×240 px for smartphones).
- The modifications given in this section increase the average number of lines detected by frame. By that reason, the percentage of missing lines detected is reduced.

Fig. 14 shows a screen shot of the application running on a mobile device. Videos showing the smartphone version of the lane detector system running under different circumstances can be download from:
http://graphicwg.irafm.osu.cz/movies/out1.mp4
http://graphicwg.irafm.osu.cz/movies/out2.mp4
http://graphicwg.irafm.osu.cz/movies/out3.mp4
http://graphicwg.irafm.osu.cz/movies/out4.mp4
http://graphicwg.irafm.osu.cz/movies/out5.mp4
Videos showing a smartphone running the app in real-time can be download from:
http://graphicwg.irafm.osu.cz/movies/v1.avi
http://graphicwg.irafm.osu.cz/movies/v2.avi

6. Conclusions and future work

This paper has presented two main contributions. The first is theoretical and has addressed the representation of an image by means of triangular fuzzy sets that measure the uncertainty of the intensities assigned by the image to pixels. Concerning this fuzzification, we have proposed a way to compute it efficiently as well. We have presented the definition of a fuzzy gradient, and we have shown that its computation is considerably faster than that of traditional gradients. In the application plane, the second contribution has been a lane departure warning system to be implemented on smartphones. The basis of the system resides on a Hough Transform algorithm customized for use under our fuzzification. The advantages of our Hough Transform version are, first, the use of the fuzzy gradient and, second, the use of weights based on our fuzzification to count the accumulators computed. Finally, we have shown by experimentation that this system is reliable and can run in real-time on smartphones.

The research regarding our fuzzification procedure has just begun. It will be interesting to study the importance of the shapes of the windows and other different possible variations of our approach. Note also that this paper modifies just the Hough Transform procedure to allow the use of fuzzified images. Thus, such a slight modification allows us to generalize our approach to other extensions of the Hough Transforms such as the *Fast Hough Transform* [10] and the *Randomized Hough Transform* [34]. Moreover, some preliminary tests have shown us that it is potentially applicable to the segmentation and reconstruction of images. Thus, applied research in these two fields will be performed. Concerning lane detection, it is still necessary to improve the accuracy in specific situations, e.g., during the night, with the presence of white (or bright) cars, etc. Finally, the development of an obstacle detector to be implemented together with our lane departure warning system in-real time is also under consideration for the future.

Acknowledgements

This work was supported by the European Regional Development Funds TIN12-39353-C04-04 (by the Spanish Ministry of Science), CZ.1.07/2.3.00/30.0010, CZ.1.05/1.1.00/02.0070 VP6 (IT4Innovations Centre of Excellence projects), SGS18/PRF/2014, SGS13/PRF/2015 and IT4I XS project number LQ1602.

References

[1] X. An, E. Shang, J. Song, J. Li, H. He, Real-time lane departure warning system based on a single FPGA, EURASIP J. Image Video Process. 2013 (1) (2013) 38.

[8] Let us recall that films in cinemas are played at 24 FPS and that, starting at 10 FPS, the human eye perceives a chain of frames as continuous.

[2] D. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (2) (1981) 111–122.
[3] S. Beucher, Segmentation d'images et morphologie mathématique, PhD thesis, Ecole des Mines de Paris, 1990.
[4] M. Bertozzi, A. Broggi, GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection, IEEE Trans. Image Process. 7 (1) (Jan. 1998) 62–81.
[5] A. Broggi, Robust real-time lane and road detection in critical shadow conditions, in: International Symposium on Computer Vision, Nov. 1995, pp. 353–358.
[6] H. Bustince, E. Barrenechea, M. Pagola, J. Fernandez, Interval-valued fuzzy sets constructed from matrices: application to edge detection, Fuzzy Sets Syst. 160 (13) (July 2009) 1819–1840.
[7] J. Crisman, C. Thorpe, SCARF: a color vision system that tracks roads and intersections, IEEE Trans. Robot. Autom. 9 (1) (Feb. 1993) 49–58.
[8] B. De Baets, E. Kerre, M. Gupta, The fundamentals of fuzzy mathematical morphology part 1: basic concepts, Int. J. Gen. Syst. 23 (2) (1995) 155–171.
[9] R.O. Duda, P.E. Hart, Use of the hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (Jan. 1972) 11–15.
[10] N. Guil, J. Villalba, E.L. Zapata, A fast hough transform for segment detection, IEEE Trans. Image Process. 4 (11) (1995) 1541–1548.
[11] P.-Y. Hsiao, C.-W. Yeh, S.-S. Huang, L.-C. Fu, A portable vision-based real-time lane departure warning system: day and night, IEEE Trans. Veh. Technol. 58 (4) (May 2009) 2089–2094.
[12] R. Josth, M. Dubska, A. Herout, J. Havel, Real-time line detection using accelerated high-resolution hough transform, in: Lecture Notes in Computer Science, vol. 6688, Springer, Berlin Heidelberg, 2011, pp. 784–793.
[13] C. Jung, C. Kelber, A lane departure warning system based on a linear-parabolic lane model, in: IEEE Symposium on Intelligent Vehicles, June 2004, pp. 891–895.
[14] A. Jurio, D. Paternain, C. Lopez-Molina, H. Bustince, R. Mesiar, G. Beliakov, A construction method of interval-valued fuzzy sets for image processing, in: IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems, April 2011, pp. 16–22.
[15] J. Kim, M. Lee, Robust lane detection based on convolutional neural network and random sample consensus, in: Lecture Notes in Computer Science, vol. 8834, Springer, 2014, pp. 454–461.
[16] H. Kong, J.-Y. Audibert, J. Ponce, General road detection from a single image, IEEE Trans. Image Process. 19 (8) (Aug. 2010) 2211–2220.
[17] C. Kreucher, S. Lakshmanan, LANA: a lane extraction algorithm that uses frequency domain features, IEEE Trans. Robot. Autom. 15 (2) (Apr. 1999) 343–350.
[18] P. Kupidura, Application of mathematical morphology operations for the improvement of identification of linear objects preliminarily extracted from classification of VHR satellite images, in: New Developments and Challenges in Remote Sensing, 2007, pp. 225–232.
[19] W. Liu, D. Dori, A generic integrated line detection algorithm and its object-process specification, Comput. Vis. Image Underst. 70 (3) (1998) 420–437.
[20] B. Ma, S. Lakshmanan, A. Hero, Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion, IEEE Trans. Intell. Transp. Syst. 1 (3) (Sep. 2000) 135–147.
[21] R. Marzotto, P. Zoratti, D. Bagni, A. Colombari, V. Murino, A real-time versatile roadway path extraction and tracking on an FPGA platform, Comput. Vis. Image Underst. 114 (11) (2010) 1164–1179.
[22] M. Mattavelli, V. Noel, E. Amaldi, Fast line detection algorithms based on combinatorial optimization, in: Lecture Notes in Computer Science, vol. 2059, Springer, 2001, pp. 410–419.
[23] J. McCall, M. Trivedi, Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation, IEEE Trans. Intell. Transp. Syst. 7 (1) (March 2006) 20–37.
[24] S. Nedevschi, R. Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, 3D lane detection system based on stereovision, in: The 7th International IEEE Conference on Intelligent Transportation Systems, Oct. 2004, pp. 161–166.
[25] B.P. Prasad, S.K. Yogamani, A 160-fps embedded lane departure warning system, in: International Conference on Connected Vehicles and Expo, ICCVE, Dec. 2012, pp. 214–215.
[26] F. Ren, J. Huang, M. Terauchi, R. Jiang, R. Klette, Lane detection on the iphone, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 30, Springer, Berlin Heidelberg, 2010, pp. 198–205.
[27] J. Ruyi, K. Reinhard, V. Tobi, W. Shigang, Lane detection and tracking using a new lane model and distance transform, Mach. Vis. Appl. 22 (4) (2011) 721–737.
[28] J. Serra, Image Analysis and Mathematical Morphology, in: Theoretical Advances, vol. 2, Academic Press, Inc., Orlando, FL, USA, 1988.
[29] P. Soille, Morphological Image Analysis: Principles and Applications, Springer-Verlag, 1999.
[30] T.-T. Tran, J.-H. Son, B.-J. Uk, J.-H. Lee, H.-M. Cho, An adaptive method for detecting lane boundary in night scene, in: Lecture Notes in Computer Science, vol. 6216, Springer, 2010, pp. 301–308.
[31] S. Valero, J. Chanussot, J.A. Benediktsson, H. Talbot, B. Waske, Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images, Pattern Recognit. 31 (10) (2010) 1120–1127.
[32] V. Voisin, M. Avila, B. Emile, S. Begot, J.-C. Bardet, Road markings detection and tracking using hough transform and Kalman filter, in: Lecture Notes in Computer Science, vol. 3708, Springer, Berlin Heidelberg, 2005, pp. 76–83.
[33] Y. Wang, E. Teoh, D. Shen, Lane detection using b-snake, in: International Conference on Information Intelligence and Systems, 1999, pp. 438–443.
[34] L. Xu, E. Oja, P. Kultanen, A new curve detection method: randomized hough transform, Pattern Recognit. Lett. 11 (5) (1990) 331–338.

P. Hurtik and N. Madrid. Bilinear interpolation over fuzzified images: enlargement. In *The 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015)*, 1–8, IEEE, 2015.

Bilinear Interpolation over fuzzified images: enlargement

Petr Hurtik
Institute for Research and Applications of Fuzzy Modeling
University of Ostrava, Czech Republic
petr.hurtik@osu.cz

Nicolas Madrid
Institute for Research and Applications of Fuzzy Modeling
University of Ostrava, Czech Republic
nicolas.madrid@osu.cz

***Abstract*—The paper explores Bilinear Interpolation applied to image enlargement after a fuzzification pre-processing. On the one hand, and from a theoretical point of view, we show some interesting relationships between Bilinear Interpolation and the Fuzzification. On the other hand, from an applied point of view we apply the interpolation obtained to enlargement and show that the obtained results are firstly, faster and secondly, comparable with the standard interpolation procedures.**

I. Introduction

The necessity of rescaling images arises in many different situations. For instance in the adjustment of an image to a screen, in the configuration of an image to be printed or in a simple zoom done on a display are examples of common rescaling situations. Basically there are two different kinds of rescaling depending on whether we are interested on *enlarge* or *shrink* the original image. In both cases, the most common procedures use interpolation methods.

There are many different interpolation methods for image processing. The most widely used are the Nearest Neighbor Interpolation, Bilinear Interpolation [1], Bicubic Interpolation [2] and Lanczos Interpolation [3]. Although there are other relatively new interpolation methods in the Literature that have shown to be effective as well; as the fractal interpolation hq$n$x [4] or an interpolation method based on fuzzy rules inference [5]. Among all of them we focus on Bilinear Interpolation which is still widely used in the Literature - only for year 2013 Google scholar founds $\approx$ 4500 articles with the phrase "Bilinear Interpolation". Perhaps the reason of its wide use is because it brings great balance between computation speed and precision of the interpolation.

In this paper we research the combination of Bilinear Interpolation with a novel fuzzification procedure for gray-scale images [6]. The fuzzification is based on the idea that the gray intensity assigned to one pixel has an inherent uncertain. This uncertainty can be due to different reasons, for instance the pixel represents an area which is not necessarily uniform and then, the intensity assigned is an average of the intensities in such area; or because the focus of the image is not well adapted and then, the intensities of the surroundings interfere the real intensity of the pixel. Actually, this representation of images is related to a well known human eye behavior: the *visual saliency* [7], [8]. Namely, images are perceived as complex structures where there is an iteration between the intensities of different pixels. Specifically, the gray level perceived in one pixel depends on the pixel in its neighborhood. Thus, our goal is to substitute the intensity value of each pixel (which it is represented by a crisp number) by a fuzzy number. Motivated by visual saliency, these fuzzy numbers represent the relationship between the intensity of pixels and the intensities in their surroundings.

For the lack of space, in this paper we cannot present a full study of the relationship between Bilinear Interpolation and our fuzzification procedure. For such a reason we restrict our study only to the case of image enlargement. Thereby, in the theoretical part of the paper we show that the Bilinear Interpolation has a good relationship with our Fuzzification procedure. Specifically we prove that the fuzzification commutes with Bilinear Interpolation in the original pixels of the image. In the applied part, we present a comparison of the enlargement proposed in this paper with respect to Bilinear Interpolation with pre and post sharpening processing.

The structure of the paper is given as follows. Section II recalls the basics of Bilinear Interpolation (in the crisp case) and its application to enlargement with sharpening modification. Subsequently, in Section III the fuzzification procedure is introduced together a method for its computation. Then, in Section IV we present the notion of Fuzzy Bilinear Interpolation and some theoretical results. Section V shows some examples and experiments of Fuzzy Bilinear Interpolation. Finally, conclusion and future works are given in Section VI.

II. Basics about Enlargement via Bilinear Interpolation

A. Image Enlargement

A gray-scale image (hereafter called just image) is a function $f\colon W\times H\to L=\{0,1,\ldots,255\}$, where $W=\{x_1,\ldots,x_w\}$ and $H=\{y_1,\ldots,y_h\}$ are the coordinates width and height of pixels, respectively. An image enlargement is formally defined as the process of creating, from an image $f\colon W\times H\to L$, an image $\overline{f}\colon \overline{W}\times\overline{H}\to L$ such that $W\subseteq\overline{W}$, $H\subseteq\overline{H}$. Usually the quality of the enlargement is given via similarity. That is, the more similar is the new image $\overline{f}$ to f, the better the enlargement. The problem with this kind of evaluation is that *"similarity"* is determined by mean of the subjectivity of the reader, instead of by means of a formal notion. Anyway, following such idea it is usual also to require that $\overline{f}(x,y)=f(x,y)$ for all $(x,y)\in W\times H$.

The W-ratio and H-ratio of enlargement are defined as the values $R_W=|\overline{W}|/|W|$ and $R_H=|\overline{H}|/|H|$, respectively.

Note that in enlargement[1] $R_W > 1$ and $R_H > 1$. If both ratios coincide, we will denote them by R. For the sake of simplicity we consider in this paper only integer ratios of enlargement. Usually, the set $\overline{W}$ is defined from $W = \{x_1, x_2, \ldots, x_w\}$ by means of a ratio of enlargement R as the set $\overline{W} = \{x_1, x_{12}, \ldots, x_{1R}, x_2, x_{22}, \ldots, x_{2R}, \ldots, x_w, x_{w2}, \ldots, x_{wR}\}$ (analogously for $\overline{H}$). In the description of interpolation methods we assume such structure and notation for elements in $\overline{W}$ and $\overline{H}$. Note that by identifying each element in $\overline{W}$ and $\overline{H}$ with the position in the chain above, we can define straightforwardly the operators addition, subtraction and distance. Moreover, note that the distance of δ between two elements in W correspond with a distance $R \cdot \delta$ in $\overline{W}$.

B. Enlargement via Bilinear Interpolation

Bilinear interpolation is an extension of the one dimensional Linear Interpolation in two dimensions. Basically, the procedure computes values assigned to one new pixel as a lineal combination of the four closest pixels in the original image. Formally, for each $x \in \overline{W}$ we define its *lower nearest neighbor* $x^{\downarrow}$ and *higher nearest neighbor* $x^{\uparrow}$ as:

$$x^{\downarrow} = \begin{cases} x & \text{if } x \in W \\ x_i & \text{if } x = x_{ij} \end{cases}$$

and

$$x^{\uparrow} = \begin{cases} x & \text{if } x \in W \\ x_{i+1} & \text{if } x = x_{ij} \end{cases},$$

respectively. Analogously, we can define the *lower nearest neighbor* $y^{\downarrow}$ and *higher nearest neighbor* $y^{\uparrow}$ for all $y \in \overline{H}$. The Bilinear Interpolation of an image $f\colon W \times H \to L$ is defined as the image $\overline{f}\colon \overline{W} \times \overline{H} \to L$ that assigns to each pixel in $\overline{W} \times \overline{H}$ the value.

$$\overline{f}(x,y) = \begin{pmatrix} \frac{x^{\uparrow}-x}{R} & \frac{x-x^{\downarrow}}{R} \end{pmatrix} \begin{pmatrix} f(x^{\downarrow},y^{\downarrow}) & f(x^{\downarrow},y^{\uparrow}) \\ f(x^{\uparrow},y^{\downarrow}) & f(x^{\uparrow},y^{\uparrow}) \end{pmatrix} \begin{pmatrix} \frac{y^{\uparrow}-y}{R} \\ \frac{y-y^{\downarrow}}{R} \end{pmatrix}. \quad (1)$$

Note firstly that $x^{\downarrow}, x^{\uparrow} \in W$ and $y^{\downarrow}, y^{\uparrow} \in H$, thus the values $f(x^{\downarrow},y^{\downarrow}), f(x^{\downarrow},y^{\uparrow}), f(x^{\uparrow},y^{\downarrow})$ and $f(x^{\uparrow},y^{\uparrow})$ are always well defined and then, they can be used to interpolate the value of $\overline{f}(x,y)$. Secondly, the value $\overline{f}(x,y)$ is a distance-based-weighted combination of the four nearest neighbors. Moreover, the four nearest neighbors bound the value of $\overline{f}(x,y)$ as follows:

$$\overline{f}(x,y) \geq \min\{f(x^{\downarrow},y^{\downarrow}), f(x^{\downarrow},y^{\uparrow}), f(x^{\uparrow},y^{\downarrow}), f(x^{\uparrow},y^{\uparrow})\}$$

and

$$\overline{f}(x,y) \leq \max\{f(x^{\downarrow},y^{\downarrow}), f(x^{\downarrow},y^{\uparrow}), f(x^{\uparrow},y^{\downarrow}), f(x^{\uparrow},y^{\uparrow})\}.$$

That means that Bilinear Interpolation does not include "new" details in the image. Finally, note that a priori $\overline{f}$ does not belong necessarily to L. However, this can be solved just by considering the integer part of the value obtained.

[1]Reduction of images can be defined similarly, but in such a case $R_W < 1$ and $R_H < 1$

Fig. 1. 15 × enlarged section of Lena Image by using Bilinear interpolation.

Fig. 2. 15 × enlarged section of Lena Image by using Bilinear interpolation over image sharpened using Laplace

C. Bilinear Interpolation with Sharpening

The enlarged image obtained by just applying Bilinear Interpolation is not considered a good enlargement. The reason is the blurriness appearing in the image (see Figure 5 in Section V). Usually, to correct this feature it is used a sharpening processing, which can be applied before or after interpolate the image. The goal of sharpening is to amplify edges of images and keep unaffected homogeneous areas. The most common procedures to detect edges of images are those based on gradient operators, e.g. Sobel [9], Prewitt [9] or Laplace [10]. Thus, by mean of a gradient operator ∇ we can define the Sharpening of and image $f\colon W \times H \to L$ by a weight $\gamma \in [0,1]$ as a new image $f^S\colon W \times H \to L$ that assigns to each $(x,y) \in W \times H$ the value:

$$f^S(x,y) = f(x,y) + \gamma \cdot \nabla f(x,y).$$

For image sharpening, one gradient operator has a dominant position namely, the Laplace gradient operator [10] represented by his mask

$$M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \quad (2)$$

The advantage of using Laplace operator for sharpening is due to the computational complexity. Although Sharpening processing can be done before and/or after bilinear interpolation, the most common is to apply it like a pre-processing. The reasons is that after Bilinear Interpolation the complexity of computing gradient operators increases considerably. Thus, the Bilinear Interpolation usually is applied to f^S instead of f. Figures 1 and 2 show the difference between applying Bilinear Interpolation with and without sharpening preprocessing.

III. Image Represented by a Fuzzy Function

In this section we present a fuzzification procedure to represent the uncertainty related to intensities of pixels. For the sake of presentation, we begin by recalling some basic

notion of triangular fuzzy numbers. Then, we describe the fuzzification procedure, a method for its computation and a sharpening processing for images based on such fuzzification.

A. Triangular Fuzzy Numbers.

Let us recall the notion of a fuzzy set.

Definition 1: A fuzzy set is a pair $A = (\mathscr{U}, \mu_A)$ where $\mathscr{U}$ is a set (called the *universe*) and μ a mapping from $\mathscr{U}$ to the unit interval $[0,1]$ (called the membership function).

Note that the membership function of a fuzzy set comprehends somehow the universe as well. By this reason, the universe is usually omitted and fuzzy sets are identified with their membership functions. We will consider also the pointwise ordering between fuzzy sets; i.e. given two fuzzy set A and B we say that $A \leq B$ if and only if $\mu(A) \leq \mu(B)$.

When the universe $\mathscr{U} = \mathbb{R}$ we can define a special kind of fuzzy sets: the fuzzy numbers.

Definition 2: Let $A = (\mathbb{R}, \mu_A)$ be a fuzzy set. We say that A is a fuzzy number if

- A is normal (i.e. there exists $x \in \mathbb{R}$ such that $\mu_A(x) = 1$)
- A is convex (i.e. the set $A_\alpha = \{x \in \mathbb{R} \mid \mu_A(x) \geq \alpha\}$ is a closed interval for all $\alpha \in (0,1]$)
- the support of A is bounded (i.e. $supp(A) = \{x \in \mathbb{R} \mid \mu_A(x) \neq 0\}$ is an interval).

Among the family of fuzzy numbers we consider only a particular family: the triangular fuzzy number. Specifically, given $a,b,c \in \mathbb{R}$ such that $a \leq x \leq c$, the triangular fuzzy number (a,b,c) is the fuzzy set determined by the membership function:

$$\mu_{(a,b,c)}(x) = \begin{cases} \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ \frac{x-c}{b-c} & \text{if } b < x \leq c \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Note that actually a triangular fuzzy number is a fuzzy number (normality, convexity and bounded). Moreover, in this paper we will consider the usual ordering between fuzzy sets described above for fuzzy numbers. In particular, two triangular fuzzy numbers (a_1,b_1,c_1) and (a_2,b_2,c_2) satisfy $(a_1,b_1,c_1) \leq (a_2,b_2,c_2)$ if and only if $a_1 \geq a_2, b_1 = b_2$ and $c_1 \leq c_2$.

B. Fuzzification of an Image

The fuzzification of an image $f\colon W \times H \to L$ is a procedure divided in two steps:

- First, we consider the symmetric window of length $\delta > 0$ defined by:

$$\omega^{\delta}_{x,y} = \{f(x_i,y_i) \in L \mid \delta \geq |x - x_i|,\ \delta \geq |y - y_i|\}.$$

- And second, given $(x,y) \in W \times H$, we define $f^{F_\delta}(x,y)$ as the triangular fuzzy number given by the triple:

$$(\min(\omega^{\delta}_{x,y}), f(x,y), \max(\omega^{\delta}_{x,y}))$$

We will write $\omega^{\delta}_{x,y} = \omega_{x,y}$ and $f^{F_\delta} = f^F$ if the size of the windows need not to be specified. Note also that after the manipulation of the fuzzified image, the triple of the fuzzy number associated to (x,y) can be different from $(\min(\omega_{x,y}), f(x,y), \max(\omega_{x,y}))$. For such a reason we will use also the notation $(\min_{xy}, cen_{x,y}, \max_{x,y})$ to refer to the fuzzy number $f^F(x,y)$.

The fuzzification is based on the idea that the gray intensity assigned to one pixel has an inherent uncertain. This uncertainty can be due to different reasons, for instance the pixel represents an area which is not necessarily uniform and then, the intensity assigned is an average of the intensities in such area; or because the focus of the image is not well adapted and then, the intensities of the surroundings interfere the real intensity of the pixel; etc.

Let us talk now about the computational cost of the fuzzification procedure. Note that by considering the smallest window (i.e. $\delta = 1$, or equivalently windows 3×3) we need to do, a priori, 16 comparisons for each pixel to obtain the respective fuzzy set (8 to obtain $\min(\omega_{x,y})$ and 8 to obtain $\max(\omega_{x,y})$). Note also that the number of comparisons increase considerably by increasing the size of windows. Summarizing and roughly speaking, that means that the naive approach above has a hard computational cost. We can reduce the computational cost considerably by following a specific strategy: to compute the fuzzification of several pixel in parallel. In this way, we avoid to compare many times the same set of pixels. Let us consider windows 3×3 (i.e. $\delta = 1$) and four adjacent pixels forming a square 2×2. The goal is to calculate the maximum and minimum associated to such four pixels simultaneously. By following the naive approach, the pairwise intersection of the four respective windows are compared twice and the four original pixel, four times. A better strategy would be to do local comparisons in each pairwise intersection and finally, unify them to achieve the final result. Let us be more specific. Let us consider the Figure 3, where the four adjacent pixels are marked by the number (1) and individually by WN (west-north), EN (east-north), ES (east-south) and WS (west-south). Let us denote by (x) the set of pixels marked by the number $x \in \{1,\ldots,9\}$ in the Figure 3. Then, the windows 3×3 centered in WN, EN, ES and WS can be discomposed by:

- $\omega_{WN} = (1) \cup (2) \cup (3) \cup (6)$,
- $\omega_{EN} = (1) \cup (3) \cup (4) \cup (7)$,
- $\omega_{ES} = (1) \cup (4) \cup (5) \cup (8)$,
- $\omega_{WS} = (1) \cup (2) \cup (5) \cup (9)$,

respectively.

6	3	3	7
2	1_{WN}	1_{EN}	4
2	1_{WS}	1_{ES}	4
9	5	5	8

Fig. 3. Overlaping of four adjacent windows 3×3

Thus, we determine the values $\min(\omega_x)$ and $\max(\omega_x)$ with $x \in \{WN, EN, ES, WS\}$ as follows:

- Firstly, we calculate the minimum and maximum of (1). That requires 4 comparisons, one to order the two pixel above, other to order the two pixels below and two to compare both minimums and maximums.
- Secondly, we calculate the minimum and maximum of the sets (2), (3), (4) and (5). Note that we only need 1 comparison to compute each pair of maximums and minimums, as each set has only two elements.
- Finally, each pair of values $\min(\omega_x)$ and $\max(\omega_x)$, with $x \in \{WN, EN, ES, WS\}$, can be computed by mean of 6 comparisons by following the decomposition given above for $\omega_{WN}, \omega_{EN}, \omega_{ES}$ and ω_{WS}.

Summarizing, with the procedure above we compute the fuzzy number associated to four pixels simultaneously by using a total amount of 32 comparisons instead of the 64 required by the naive approach. That is exactly a 50% of reduction in comparisons. We can extend the procedure above for arbitrary sizes of windows, by considering different decompositions.

C. Gradient and Sharpening Modification

The idea behind gradient operators is the measurement of variation of a function in the surroundings of a point; in our the variation of intensities in the surroundings of a pixel. In fuzzification, this variation can be measured just be measuring the size of the supports. Specifically, we define the fuzzy gradient of a pixel $(x,y) \in W \times H$ in a fuzzy image $f\colon W \times H \to L$ as the value given by:

$$\nabla f^F(x,y) = min_{x,y} - max_{x,y}$$

The main advantage of using this fuzzy gradient is the speed in the computation with respect to classical methods, for details see [6]. Another differential characteristic is that the result of our gradient is a little bit more blurrier than with standard gradients. Just a note, in some applications, blurring process is sometimes applied before gradient operator. It is worth to mention that this gradient operator is not novel in the Literature. In [11], [**?**] authors define a family of gradients operators aimed to detect (binary) edges of an image and ∇f^F coincides with one of them.

The surrounding of a pixel is obviously connected with the shape of the fuzzy set associated to it. If the image is blur, the intensities surrounding a pixel are uniformly distributed and thus, the shape of the membership function of the fuzzy set looks like an isosceles triangle. On the other hand, if the image is sharp, the shape of the membership function looks like a right triangle. Therefore, we can sharp or blur an image by transforming the shapes of the fuzzy numbers assigned to pixels. Specifically, given a fuzzy image f^F, we can blur an image by moving $cen_{x,y}$ near to the center of mass in the support, i.e. to $M_{x,y} = 0.5(\min_{x,y} + \max_{x,y})$. Dually, we can sharp an image by moving $cen_{x,y}$ away to the center of mass in the support. Formally, given a value $\gamma \in [0,1]$, the sharpening of f^F assigns to each pixel (x,y) the fuzzy number $(\min_{x,y}, SH(cen_{x,y}), \max_{x,y})$ where

$$SH(cen_{x,y}) = inf(255, sup(0, cen_{x,y} + \gamma(cen_{x,y} - M_{x,y}))).$$

IV. Bilinear Interpolation over Fuzzified Images.

In this section we begin by rewriting the Bilinear Interpolation formula in terms of fuzzy numbers by means of fuzzy arithmetic. Let us recall that any operation $\star$ between real numbers can be extended to fuzzy numbers as follows: let A and B be two fuzzy numbers, then $A \star B$ is defined as:

$$A \star B(z) = \sup_{z = x \star y} \{\min\{\mu_A(x), \mu_B(y)\}\} \quad \forall z \in \mathbb{R}.$$

Matrix arithmetic for Fuzzy Numbers is straightforwardly extended from the generalization above. Now, let us note that the position of a *"new pixel"* (i.e. of those in $(\overline{W} \times \overline{H}) \setminus (W \times H)$) is clearly determined; in other words, it is a crisp notion. Therefore, the coordinates of the vector used to represent the position of $x \in W$ (analogous for $y \in H$) are given by the characteristic functions:

$$\chi_{\frac{(x^{\uparrow} - x)}{R}}(z) = \begin{cases} 1 & \text{If } z = {}^{(x^{\uparrow} - x)}/_{R} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\chi_{\frac{(x - x^{\downarrow})}{R}}(z) = \begin{cases} 1 & \text{If } z = {}^{(x - x^{\downarrow})}/_{R} \\ 0 & \text{otherwise} \end{cases}.$$

Now we define the fuzzy Bilinear Interpolation as follows.

Definition 3: Let $f\colon W \times H \to L$ be an image, the fuzzy bilinear interpolation of f, denoted by $\overline{f^F}(x,y)$, is given by the formula:

$$\begin{pmatrix} \chi_{\frac{(x^{\uparrow} - x)}{R}} & \chi_{\frac{(x - x^{\downarrow})}{R}} \end{pmatrix} \begin{pmatrix} f^F(x^{\downarrow}, y^{\downarrow}) & f^F(x^{\downarrow}, y^{\uparrow}) \\ f^F(x^{\uparrow}, y^{\downarrow}) & f^F(x^{\uparrow}, y^{\uparrow}) \end{pmatrix} \begin{pmatrix} \chi_{\frac{(y^{\uparrow} - y)}{R}} \\ \chi_{\frac{(y - y^{\downarrow})}{R}} \end{pmatrix}. \quad (4)$$

The computation of the fuzzy set above has a priori a high cost. However, thanks to the triangular fuzzy number structure of f^F, the computation can be reduced significantly as the following results states.

Lemma 1: Let $f\colon W \times H \to L$ be an image. Let $\omega_{\min}\colon W \times H \to L$ and $\omega_{\max}\colon W \times H \to L$ be the mappings given by:

$$\omega_{\min}(x,y) = \min(\omega_{x,y}) \quad \text{and} \quad \omega_{\max}(x,y) = \max(\omega_{x,y}),$$

respectively. Then, $\overline{f^F}(x,y)$ is the fuzzy triangular number given by the triple:

$$\overline{f^F}(x,y) = (\overline{\omega_{\min}}(x,y), \overline{f}(x,y), \overline{\omega_{\max}}(x,y)),$$

where $\overline{\omega_{\min}}(x,y), \overline{f}(x,y)$ and $\overline{\omega_{\max}}(x,y)$ denotes the Bilinear Interpolation of $(\omega_{\min}(x,y), f(x,y)$ and $\omega_{\max}(x,y)$, respectively.

Proof: The proof comes straightforwardly from the fact that for all $\alpha \in \mathbb{R}$ (with characteristic function χ_α) and for all pair of triangular numbers (a_1, b_1, c_1) and (a_2, b_3, c_2) we have the following equalities:

- $\chi_\alpha \cdot (a_1, b_1, c_1) = (\alpha \cdot a_1, \alpha \cdot b_1, \alpha \cdot c_1)$
- $(a_1, b_1, c_1) + (a_2, b_3, c_2) = (a_1 + a_2, b_1 + b_2, c_1 + c_2)$.

■

The following lemma shows that, as in the case of bilinear interpolation, the values assigned to pixels of the original image remain the same.

Lemma 2: Let $f\colon W\times H\to L$ be an image. Then, $\overline{f^F}(x,y)=f^F(x,y)$ for all $x\in W$ and $y\in H$.

Proof: The proof comes from a simple calculus from the formula (4).Specifically, if $x\in W$ and $y\in H$ then $x\in\{x^\uparrow,x^\downarrow\}$ and $y\in\{y^\uparrow,y^\downarrow\}$. Then, if we assume $x=x^\uparrow$ and $y=y^\uparrow$ (the rest of cases are proved similarly) we have

$$\begin{aligned}\overline{f^F}(x^\uparrow,y^\uparrow)&=\begin{pmatrix}\chi_0 & \chi_1\end{pmatrix}\begin{pmatrix}f^F(x^\downarrow,y^\downarrow) & f^F(x^\downarrow,y^\uparrow)\\ f^F(x^\uparrow,y^\downarrow) & f^F(x^\uparrow,y^\uparrow)\end{pmatrix}\begin{pmatrix}\chi_0\\ \chi_1\end{pmatrix}\\&=f^F(x^\uparrow,y^\uparrow)\end{aligned}$$

■

In the rest of this section we study the relationship between the fuzzification and the bilinear interpolation in terms of conmutativity. In other words, the relationship between fuzzify and then interpolate (denoted by $\overline{f^F}$) and interpolate and then fuzzify (denoted by $\overline{f}^F$). The following result shows that the Bilinear Interpolation procedure somehow commutes with the fuzzification in the pixels of the original image. For the sake of presentation we consider the following notation hereafter: we denote by ω windows related to the original image $f\colon W\times H\to L$ and by $\overline{\omega}$ windows related to the "crisp" interpolated image $\overline{f}\colon\overline{W}\times\overline{H}\to L$

Theorem 1: Let $f\colon W\times H\to L$ be an image and $R,\delta\in\mathbb{N}$. Then, $\overline{f^{F_\delta}}(x,y)=\overline{f}^{F_{R\cdot\delta}}(x,y)$ for all $x\in W$ and $y\in H$.

Proof: Note that to prove the result is enough to show that, for all $(x,y)\in W\times H$, $\min\omega^\delta_{x,y}=\min\overline{\omega}^{R\cdot\delta}_{x,y}$ and $\max\omega^\delta_{x,y}=\max\overline{\omega}^{R\cdot\delta}_{x,y}$ since, by Lemma 2 we have $\overline{f}^{F_{R\cdot\delta}}(x,y)=f^{F_\delta}(x,y)=\overline{f^{F_\delta}}(x,y)$. Let $(x,y)\in W\times H$. By the properties of Bilinear Interpolation we know that

$$\begin{aligned}\omega^\delta_{x,y}=&\{f(x_i,y_i)\mid\delta\geq|x-x_i|,\ \delta\geq|y-y_i|\}\\&\subseteq\{\overline{f}(x_i,y_i)\mid R\cdot\delta\geq|x-x_i|,\ R\cdot\delta\geq|y-y_i|\}=\overline{\omega}^{R\cdot\delta}_{x,y}.\end{aligned}$$

Note that in the latter set the ordering is related to $\overline{W}\times\overline{H}$ whereas in the former to $W\times H$. Then,

$$\min\omega^\delta_{x,y}\geq\min\overline{\omega}^{R\cdot\delta}_{x,y}\quad\text{and}\quad\max\omega^\delta_{x,y}\leq\max\overline{\omega}^{R\cdot\delta}_{x,y}.$$

So, if we prove the converse inequalities, we finish the proof. Let us show that $\min\omega^\delta_{x,y}\leq\min\overline{\omega}^{R\cdot\delta}_{x,y}$ (the other inequality is proved analogously). Let $\overline{f}(\overline{x},\overline{y})\in\overline{\omega}^{R\cdot\delta}_{x,y}$ with $(\overline{x},\overline{y})\in\overline{W}\times\overline{H}$. Note that, by the definition of the domain $\overline{W}\times\overline{H}$, necessarily $f(\overline{x}^\downarrow,\overline{y}^\downarrow),f(\overline{x}^\downarrow,\overline{y}^\uparrow),f(\overline{x}^\uparrow,\overline{y}^\downarrow),f(\overline{x}^\uparrow,\overline{y}^\uparrow)\in\omega^\delta_{x,y}$ Let us consider without loss of generality that $f(x^\downarrow,y^\downarrow)=\min\{f(x^\downarrow,y^\downarrow),f(x^\downarrow,y^\uparrow),f(x^\uparrow,y^\downarrow),f(x^\uparrow,y^\uparrow)\}$. Then,

$$\begin{aligned}\overline{f}(\overline{x},\overline{y})=&\begin{pmatrix}\frac{x^\uparrow-x}{R} & \frac{x-x^\downarrow}{R}\end{pmatrix}\begin{pmatrix}f(x^\downarrow,y^\downarrow) & f(x^\downarrow,y^\uparrow)\\ f(x^\uparrow,y^\downarrow) & f(x^\uparrow,y^\uparrow)\end{pmatrix}\begin{pmatrix}\frac{y^\uparrow-y}{R}\\ \frac{y-y^\downarrow}{R}\end{pmatrix}\\=&\frac{y^\uparrow-y}{R}\left(\frac{x^\uparrow-x}{R}\cdot f(x^\downarrow,y^\downarrow)+\frac{x-x^\downarrow}{R}\cdot f(x^\downarrow,y^\uparrow)\right)\\&+\frac{y-y^\downarrow}{R}\left(\frac{x^\uparrow-x}{R}\cdot f(x^\uparrow,y^\downarrow)+\frac{x-x^\downarrow}{R}\cdot f(x^\uparrow,y^\uparrow)\right)\\\geq&\frac{y^\uparrow-y}{R}\left(\frac{x^\uparrow-x}{R}\cdot f(x^\downarrow,y^\downarrow)+\frac{x-x^\downarrow}{R}\cdot f(x^\downarrow,y^\downarrow)\right)\\&+\frac{y-y^\downarrow}{R}\left(\frac{x^\uparrow-x}{R}\cdot f(x^\downarrow,y^\downarrow)+\frac{x-x^\downarrow}{R}\cdot f(x^\downarrow,y^\downarrow)\right)\\=&f(x^\downarrow,y^\downarrow)\in\omega^\delta_{x,y}.\end{aligned}$$

In other words $\overline{f}(\overline{x},\overline{y})\geq\min\omega^\delta_{x,y}$ for all $(\overline{x},\overline{y})\in\overline{\omega}^{R\cdot\delta}_{x,y}$ or equivalently, $\min\omega^\delta_{x,y}\leq\min\overline{\omega}^{R\cdot\delta}_{x,y}$. ■

It is not hard to check that the equality in the previous theorem does not hold for new pixels, i.e. for pixels in $(\overline{W}\setminus W)\times(\overline{H}\setminus H)$. However, for those pixel we can establish another relationship, specifically fuzzifying and then interpolating gives an image greater than or equal to the one given by interpolating and then fuzzifying. But firstly let us consider the following lemma.

Lemma 3: Let $f\colon W\times H\to L$ be an image and let $(x,y),(\overline{x},\overline{y})\in\overline{W}\times\overline{H}$. If $\overline{f}(x,y)\in\overline{\omega}^{R\cdot\delta}_{\overline{x},\overline{y}}$ then:

- $\overline{f}(x^\downarrow,y^\downarrow)\in\omega^\delta_{\overline{x}^\downarrow,\overline{y}^\downarrow}$,
- $\overline{f}(x^\downarrow,y^\uparrow)\in\omega^\delta_{\overline{x}^\downarrow,\overline{y}^\uparrow}$,
- $\overline{f}(x^\uparrow,y^\downarrow)\in\omega^\delta_{\overline{x}^\uparrow,\overline{y}^\downarrow}$,
- $\overline{f}(x^\uparrow,y^\uparrow)\in\omega^\delta_{\overline{x}^\uparrow,\overline{y}^\uparrow}$.

Proof: We prove only the former item since the others are proved similarly. Note that any $x\in\overline{W}$ (analogously for $y\in\overline{H}$) can be written as $x=\alpha_x+\beta_x\cdot R$ with $\alpha_x,\beta_x\in\mathbb{N}$ and $\alpha_x<R$. Since elements in W are allocated into $\overline{W}$ in positions represented by multiples of R, $\alpha_x=0$ is equivalent to say $x\in W$ (analogously for $y\in H$). Moreover, by definition of $x^\downarrow$ and $x^\uparrow$ we know that $x^\downarrow=\beta_x\cdot R$ and $x^\uparrow=(\beta_x+1)\cdot R$ for all $x\in\overline{W}$.

Let $\overline{f}(x,y)\in\overline{\omega}^{R\cdot\delta}_{\overline{x},\overline{y}}$ and let us assume without loss of generality that $\overline{x}\leq x$ and $\overline{y}\leq y$. Note that by the notation described in the previous paragraph, $x^\downarrow-\overline{x}^\downarrow=(\beta_x-\beta_{\overline{x}})\cdot R$ and $y^\downarrow-\overline{y}^\downarrow=(\beta_y-\beta_{\overline{y}})\cdot R$. Thus, if we prove that $(\beta_x-\beta_{\overline{x}})\cdot R\leq\delta\cdot R$ and $(\beta_y-\beta_{\overline{y}})\cdot R\leq\delta\cdot R$ we finish the proof, since in such a case $\overline{f}(x^\downarrow,y^\downarrow)\in\omega^\delta_{\overline{x}^\downarrow,\overline{y}^\downarrow}$[2]. As $x-\overline{x}\leq\delta\cdot R$ we have

$$\alpha_x-\alpha_{\overline{x}}+(\beta_x-\beta_{\overline{x}})\cdot R\leq\delta\cdot R.$$

Note that as $\alpha_x,\alpha_{\overline{x}}<R$, then $-R<\alpha_x-\alpha_{\overline{x}}<R$, and therefore:

$$(\beta_x-\beta_{\overline{x}})\cdot R\leq\delta\cdot R-\alpha_x+\alpha_{\overline{x}}<(\delta+1)\cdot R.$$

Finally, taking into account that $(\beta_x-\beta_{\overline{x}})\cdot R$ is a multiple of R, the upper bound above can be improved by considering

[2]Note that a distance δ between two pixels in W and H corresponds to a distance $R\cdot\delta$ in $\overline{W}$ and $\overline{H}$

the greatest multiple of R lesser than $(\delta+1)\cdot R$; in other words $(\beta_x-\beta_{\overline{x}})\cdot R\leq\delta\cdot R$. The proof of $(\beta_y-\beta_{\overline{y}})\cdot R\leq\delta\cdot R$ is analogous. ■

Theorem 2: Let $f\colon W\times H\to L$ be an image. Then, $\overline{f^{F_\delta}}(x,y)\geq\overline{f}^{F_{R\cdot\delta}}(x,y)$ for all $x\in\overline{W}$ and $y\in\overline{H}$.

Proof: By Lemma 1 we know that $\overline{f^{F_\delta}}(x,y)=(\overline{\omega_{\min}}(x,y),\overline{f}(x,y),\overline{\omega_{\max}}(x,y))$ for all $(x,y)\in\overline{W}\times\overline{H}$. Therefore, if we show that $\overline{\omega_{\min}}(x,y)\leq\min\overline{\omega}^{R\cdot\delta}_{x,y}$ and $\overline{\omega_{\max}}(x,y)\geq\max\overline{\omega}^{R\cdot\delta}_{x,y}$ we finish the proof,

To prove $\overline{\omega_{\min}}(x,y)\leq\min\overline{\omega}^{R\cdot\delta}_{x,y}$ let us show that for all $\overline{f}(x_*,y_*)\in\overline{\omega}^{R\cdot\delta}_{x,y}$ we have $\overline{f}(x_*,y_*)\geq\overline{\omega_{\min}}(x,y)$. By Lemma 3, we obtain straightforwardly that:

- $\overline{f}(x^\downarrow_*,y^\downarrow_*)\geq\omega_{\min}(x^\downarrow,y^\downarrow)$,
- $\overline{f}(x^\downarrow_*,y^\uparrow_*)\geq\omega_{\min}(x^\downarrow,y^\uparrow)$,
- $\overline{f}(x^\uparrow_*,y^\downarrow_*)\geq\omega_{\min}(x^\uparrow,y^\downarrow)$,
- $\overline{f}(x^\uparrow_*,y^\uparrow_*)\geq\omega_{\min}(x^\uparrow,y^\uparrow)$.

So, by the formula (1) of the bilinear interpolation we have:

$$\begin{aligned}\overline{\omega_{\min}}(x,y)&=\begin{pmatrix}\frac{x^\uparrow-x}{R} & \frac{x-x^\downarrow}{R}\end{pmatrix}\begin{pmatrix}\omega_{\min}(x^\downarrow,y^\downarrow) & \omega_{\min}(x^\downarrow,y^\uparrow)\\ \omega_{\min}(x^\uparrow,y^\downarrow) & \omega_{\min}(x^\uparrow,y^\uparrow)\end{pmatrix}\begin{pmatrix}\frac{y^\uparrow-y}{R}\\ \frac{y-y^\downarrow}{R}\end{pmatrix}\\
&=\frac{y^\uparrow-y}{R}\left(\frac{x^\uparrow-x}{R}\cdot\omega_{\min}(x^\downarrow,y^\downarrow)+\frac{x-x^\downarrow}{R}\cdot\omega_{\min}(x^\downarrow,y^\uparrow)\right)\\
&+\frac{y-y^\downarrow}{R}\left(\frac{x^\uparrow-x}{R}\cdot\omega_{\min}(x^\uparrow,y^\downarrow)+\frac{x-x^\downarrow}{R}\cdot\omega_{\min}(x^\uparrow,y^\uparrow)\right)\\
&\leq\frac{y^\uparrow-y}{R}\left(\frac{x^\uparrow-x}{R}\cdot\overline{f}(x^\downarrow_*,y^\downarrow_*)+\frac{x-x^\downarrow}{R}\cdot f(x^\downarrow_*,y^\uparrow_*)\right)\\
&+\frac{y-y^\downarrow}{R}\left(\frac{x^\uparrow-x}{R}\cdot f(x^\uparrow_*,y^\downarrow_*)+\frac{x-x^\downarrow}{R}\cdot(x^\uparrow_*,y^\uparrow_*)\right)\\
&=\overline{f}(x_*,y_*).\end{aligned}$$

Therefore, $\overline{\omega_{\min}}(x,y)\leq\min\overline{\omega}^{R\cdot\delta}_{x,y}$. The inequality $\overline{\omega_{\max}}(x,y)\geq\max\overline{\omega}^{R\cdot\delta}_{x,y}$ is proved similarly. ■

The following corollary shows that the gradient value in the original image is greater than in the bilinear interpolated one. That explains why the interpolated image looks blurrer than the original.

Corollary 1: Let $f\colon W\times H\to L$ be an image. Then,

$$\nabla f^{F_\delta}(x,y)\geq\nabla\overline{f^{F_\delta}}(x,y)\geq\nabla\overline{f}^{F_{R\cdot\delta}}(x,y)\geq\nabla\overline{f}^{F_\delta}(x,y)$$

for all $x\in W,y\in H$.

V. Enlargement Image Experiments

In this section we present results of applying Fuzzy Bilinear Interpolation for image enlargement. As in the case of enlargement via "crisp" Bilinear Interpolation, the image obtained look blur. For this reason, the algorithm used in this section is similar to the one described in Section II-C, i.e. a combination of Sharpening and Bilinear Interpolation. Thus, we have five possibilities for enhancement, namely:

- Bilinear Interpolation (BiIlinear)
- Bilinear Interpolation with a Laplace based sharpening pre-processing (Bilinear + Laplace pre)
- Bilinear Interpolation with a Laplace sharpening post-processing (Bilinear + Laplace post)
- Fuzzy Bilinear Interpolation with a Fuzzy sharpening pre-processing (FBI)
- Fuzzy Bilinear Interpolation with a Fuzzy sharpening post-processing (FBI)

Two remarks. Note that the defuzzification of a Fuzzy Bilinear Interpolated image coincides with its Bilinear Interpolation. This is because firstly, the central element assigned to every pixel by $\overline{f^F}$ coincides with $\overline{f}$ and secondly, the natural defuzzification on triangular fuzzy numbers consists in taking the unique element such that its membership value is 1; i.e. the central element. By this reason has no sense to represent the Fuzzy Bilinear Interpolation and Bilinear Interpolation for the same image. Note also that the enlargement depending on sharpening require a sharp-parameter γ.

Let us begin by comparing the five enlargement procedures through a simple signal $f=\{102,221,32,221,127,221,0,127,195\}$. Figure 4 shows in detail the result for a section of the input data with $R=20$ and $\gamma=1$ for all sharpening algorithm. From this graph we can conclude that:

- the five procedures are definitely different.
- In the case of crisp sharpening, the sharpening after Bilinear Interpolation change slightly the image interpolated. However, the sharpening pre-processing has a considerable effect.
- In the case of Fuzzy sharpening, both sharpening pre-processing and sharpening post-processing have a important influence in the final result.

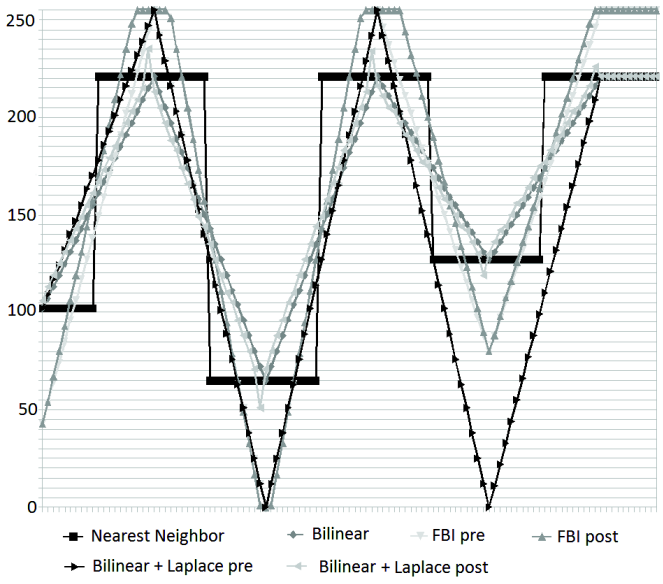

Fig. 4. Illustration of enlargement result

Table I shows computation times for the procedure for $R = 2$. Because of FBI differs from Bilinear Interpolation only in the sharpening processing, the procedure concerning with sharpening pre-pocesing is computed only partially, specifically only the central value of $\overline{f^F}$. The image resolution for testing is 1920×1080px (high resolution suppress measure errors). Each algorithm was run 5 times and in the table is included the average time. The result shows that Fuzzy Interpolation with sharpening preprocessing needs about 20% more of computation time than the Bilinear Interpolation, but it is about 15% faster than Bilinear Interpolation extended by Sharpening pre-processing and, about 170% faster than Bilinear Interpolation extended by Sharpening post-processing.

TABLE I. IMAGE ENLARGEMENT COMPUTATION TIME FOR $R = 2$

Algorithm	Computation time [ms]
Bilinear	413
FBI pre	503
FBI post	1847
Bilinear + Laplace pre	581
Bilinear + Laplace post	1359

Since Sharpening preprocessing depends only on the size of the input image, for big values of R such procedures should have similar time complexity. Table II shows the computation time for those procedures with big R. Note that the time is almost the same for all three algorithms, actually the difference can be explained by a measure error.

TABLE II. IMAGE ENLARGEMENT COMPUTATION TIME FOR DIFFERENT R

	Computation time [ms]		
R	Bil	Bil Lap pre	FIB
2	47	63	47
5	281	296	297
10	1063	1078	1109
20	4297	4297	4312
30	9719	9734	9735

Figure 5 shows enlargement of Lena image with $R = 2$. It is obvious, that standard Bilinear Interpolation creates a little blur image. The version with sharpening post-processing look more natural, without hallo effect around edges. Sharpening before enlargement in provides output over-sharped (the decrease of γ might help).

Figure 6 shows enlargement of image consisting only of white and black pixels. The original image is enlarged recursively four times with $R = 2$. This kind of enlargement is chosen to magnify misbehaviors, errors etc., since this process accumulate them and make they more visible. The nearest neighbor algorithm was taken also as reference. In the case of Bilinear Interpolation, the output is a blur copy of the original. The last two outputs were created with sharpening pre-processing to show sharpening misbehaviors. In the case of Bilinear Interpolation with Laplace sharpening appear something similar to *"crosses"* at the beginning and at the end of

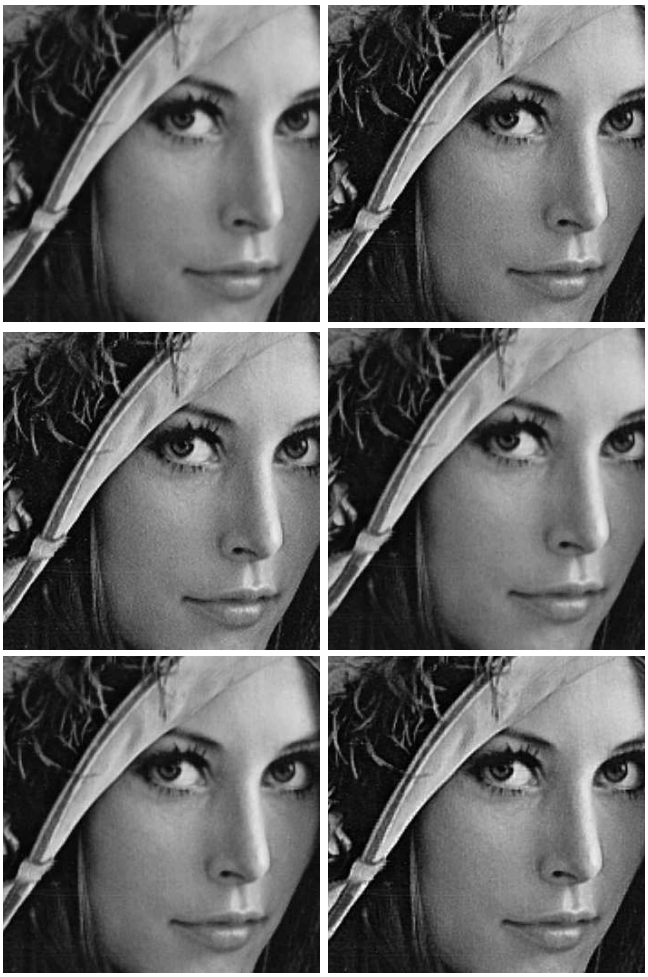

Fig. 5. Example of enlargement with $R = 2$ over Lena image. From left to right, from top to botton: Bilinear; Bilinear with Laplace pre, $\gamma = 0.4$; Bilinear with Laplace post, $\gamma = 0.4$; FBI post $\gamma = 0.4$; FBI pre $\gamma = 0.4$; FBI pre $\gamma = 1.0$

lines. These crosses are less visible in the case of FBI. In our opinion, the crosses are due to the Laplace mask (see Formula (2)), where the non-zero weight has cross-shape.

To end this section, we provide some examples with noisy images. It is important to take into account that Sharpening processing is very sensitive to noise. Figure 7 shows an enlargement preprocessing of a noised image. The image shows that FBI is to noise much more robust than Bilinear with pre- and post- sharpening. The reason is that Fuzzy sharpening lie in values $\min \omega_{x,y}$ and $\max \omega_{x,y}$ - see formula (III-C). It means that if a new noise value o is in the original image but satisfying $\min(\omega_{x,y}) \leq o \leq \max(\omega_{x,y})$ and $o \neq cen(\omega_{x,y})$ the noise not affect the original result and the noise is fully suppressed.

VI. CONCLUSION

In this paper we have recalled the notion of Bilinear Interpolation in image processing for enlargement. We have extended Bilinear Interpolation with a fuzzification processing and we have shown that the relationship between the result obtained by considering the fuzzification procedure like a preprocessing or like a post-processing.

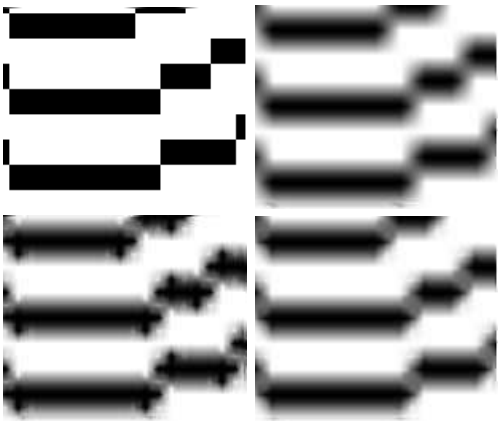

Fig. 6. Example of enlargement with $R = 2$, fourth times. On the top line is nearest neighbor in the left side, Bilinear on the right one. Bottom line includes Bilinear Interpolation with Laplace pre and $\gamma = 0.5$; on the right is FIBI $\gamma = 0.5$

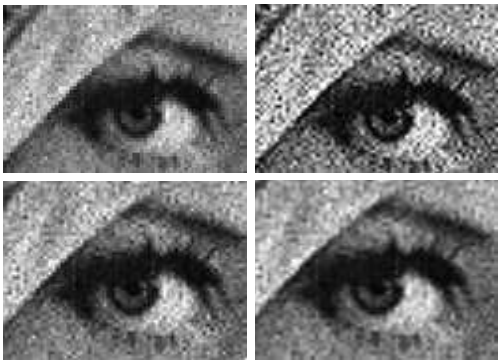

Fig. 7. Example of enlargement of noised image. From top to bottom, from left to right: original image enlargement with nearest neighbor; Bilinear with Laplace pre; bilinear with Laplace post; FBI. All $\gamma = 0.5$

We have shown that the main disadvantage of Bilinear Interpolation is the blurriness. We used Laplace gradient operator to sharp the image. We have applied sharpening as a pre and post processing. The experiment shows that enlarging image with sharpening processing gives better results but with an increase in the computation time. In the parallel, we propose gradient detection over image represented by a fuzzy function and modify them with the same idea as in standard case to image sharpening. We have shown that the output image can be compared with standard approach with sharpening, but the computation time is in our case lesser.

Image interpolation can be applied to many other different tasks from rescaling, as image shrinking, image rotation or image reconstruction [12]. In the future work we should research theoretically and experimentally how to applied the Fuzzy Bilinear Interpolation to such procedures.

ACKNOWLEDGMENT

Additional support was provided by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Spanish Ministry of Science by the project TIN2012-39353-C04-04. This work is also co-support by University of Ostrava SGS project.

REFERENCES

[1] K.-T. Chang, *Introduction to geographic information systems*. McGraw-Hill, 2008.

[2] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, Dec 1981.

[3] C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of Applied Meteorology*, vol. 18, no. 8, pp. 1016–1022, 2015/04/23 1979.

[4] J. Kopf and D. Lischinski, "Depixelizing pixel art," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 99:1–99:8, Jul. 2011.

[5] J.-L. Chen, J.-Y. Chang, and K.-L. Shieh, "2-d discrete signal interpolation and its image resampling application using fuzzy rule-based inference," *Fuzzy Sets and Systems*, vol. 114, no. 2, pp. 225 – 238, 2000.

[6] N. Madrid and P. Hurtik, "Lane departure warning for mobile devices based on a fuzzy representation of images," *Fuzzy Sets and Systems*, vol. (submitted), 2015.

[7] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *IEEE Conference on Computer Vision and Pattern Recognition. CVPR '07.*, June 2007, pp. 1–8.

[8] Y. Li, Y. Zhou, J. Yan, Z. Niu, and J. Yang, "Visual saliency based on conditional entropy," in *Computer Vision – ACCV 2009*, ser. Lecture Notes in Computer Science, H. Zha, R.-i. Taniguchi, and S. Maybank, Eds. Springer Berlin Heidelberg, 2010, vol. 5994, pp. 246–257.

[9] S. Saluja and A. K. S. and, "A study of edge-detection methods and sonu agrawal," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 1, 2013.

[10] T. Ma, L. Li, S. Ji, X. Wang, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "Optimized laplacian image sharpening algorithm based on graphic processing unit," *Physica A: Statistical Mechanics and its Applications*, vol. 416, no. 0, pp. 400 – 410, 2014.

[11] H. Bustince, E. Barrenechea, M. Pagola, and J. Fernandez, "Interval-valued fuzzy sets constructed from matrices: Application to edge detection," *Fuzzy Sets and Systems*, vol. 160, no. 13, pp. 1819 – 1840, 2009, theme: Information Processing and Applications.

[12] I. Perfilieva and P. Vlasanek, "Image reconstruction by means of f-transform," *Knowledge-Based Systems*, vol. 70, no. 0, pp. 55 – 63, 2014.

P. Hurtik and I. Perfilieva. Image compression methodology based on fuzzy transform using block similarity. In *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*. Atlantis Press, 2013.

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.

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

© 2013. The authors - Published by Atlantis Press

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:

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

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 $D = 20$.

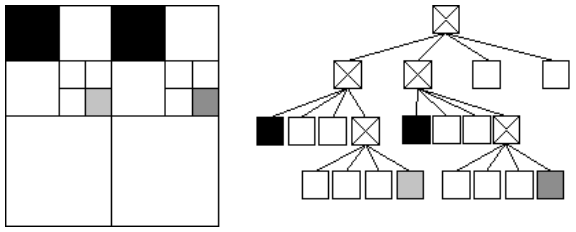

Figure 1: Left: Example of an original image to compress. Right: Image divided into quad-tree structure. Blocks with cross represent decomposed block.

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

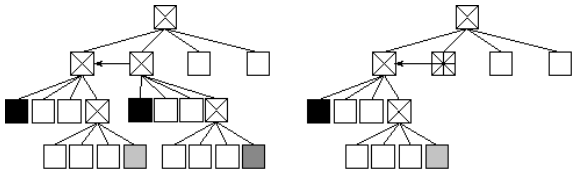

Figure 2: Left: Quad-tree structure with marked reference to similar block. Right: Updated quad-tree without blocks redundant in similarity. The block with star represent block with reference to another one.

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

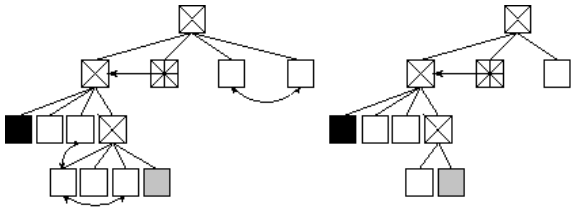

Figure 3: Left: Quad-tree structure with marked blocks to join. Right: updated Quad-tree structure.

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

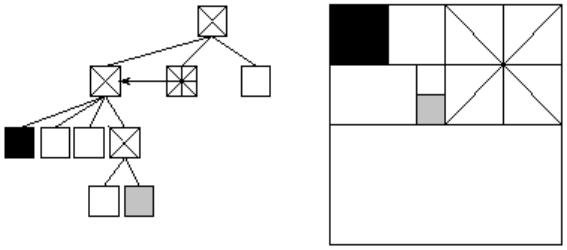

Figure 4: Left: Final quad-tree structure. Right: the quad-tree structure visualized as an image.

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.

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

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

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.

P. Hurtik, I. Perfilieva, and P. Hodakova. Fuzzy transform theory in the view of image registration application. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 143–152. Springer, 2014.

Fuzzy Transform Theory in the View of Image Registration Application

Petr Hurtík, Irina Perfilieva, and Petra Hodáková

University of Ostrava, Centre of Excellence IT4Innovations,
Institute for Research and Applications of Fuzzy Modeling,
30. dubna 22, 701 03 Ostrava 1, Czech Republic
{petr.hurtik,irina.perfilieva,petra.hodakova}@osu.cz

Abstract. In this paper, the application of the fuzzy transforms of the zero degree (F^0-transform) and of the first degree (F^1-transform) to the image registration is demonstrated. The main idea is to use only one technique (F-transform generally) to solve various tasks of the image registration. The F^1-transform is used for an extraction of feature points in edge detection step. The correspondence between the feature points in two images is obtained by the image similarity algorithm based on the F^0-transform. Then, the shift vector for corresponding corners is computed, and by the image fusion algorithm, the final image is created.

Keywords: image registration, feature detection, edge detection, image similarity, image fusion.

1 Introduction

In computer graphics, interactions between the machine and the real worlds are basically ensured by the image processing. One of the tasks is to represent data for computer processing to be similar to the humen eye vision as much as possible. Therefore, this task is very popular in developing soft-computing methods. It became a common practice that soft computing methods work with uncertain information and can achieve better result than methods based on crisps information.

One of the effective soft computing methods is fuzzy transform (F-transform for short) developed by Irina Perfilieva. The main theoretical preliminaries were described in [1] [2]. The F-transform is a technique that performs a transformation of an original universe of functions into a universe of their "skeleton models". Each component of the resulting skeleton model is a weighted local mean of the original function over an area covered by a corresponding basic function. The F-transform consists of two steps: direct and inverse transform. This method proved to be very general and powerful in many applications. Particularly, image compression [3] [4], where the user can control the strength and the quality of compression by choosing the number of components used in F-transform. Another application is image fusion [7] [8], where several damaged images are fused in one image which then has better quality than all the particular images. Image reduction and interpolation [5] is another application where

A. Laurent et al. (Eds.): IPMU 2014, Part II, CCIS 443, pp. 143–152, 2014.
© Springer International Publishing Switzerland 2014

the direct F-transform can reduce (shrink) the original image and the inverse F-transform can be used as an interpolation method. The F-transform of a higher degree (F^s-transform, $s \geq 1$) [10] can approximate the original function even better. Moreover, the F^1-transform can approximate the partial derivatives of the original function and therefore, it can be used in edge detection to compute the image gradient [9].

The task of the image registration is to match up two or more images. There are several examples where the image registration is used - images taken by different sensors, in different time, from different positions, with different size, etc. One of the most natural applications is to match several images of landscape which are partially overlapped into one large image. There exists a lot of methods how to register images [6], most of them consist of four basic steps: detect important features in each image; match the features from all images; find a suitable mapping function which describes image shift, rotation, etc.; interpolate images and fuse their overlaps.

This contribution, we demonstrate the use of the F-transform technique for all those steps: the F^1-transform for the gradient detection and for the feature points extraction; the F^0-transform for image similarity measures and for the image fusion.

2 Fuzzy Transform

2.1 Generalized Fuzzy Partitions

A *generalized fuzzy partition* appeared in [10] in connection with the notion of the higher-degree F-transform. Its even weaker version was implicitly introduced in [3] for the purpose of meeting the requirements of image compression. We summarize both these notions and propose the following definition.

Definition 1. *Let $[a, b]$ be an interval on the real line $\mathbb{R}$, $n > 2$, and let $x_1, \ldots, x_n$ be nodes such that $a \leq x_1 < \ldots < x_n \leq b$. Let $[a, b]$ be covered by the intervals $[x_k - h'_k, x_k + h''_k] \subseteq [a, b]$, $k = 1, \ldots, n$, such that their left and right margins $h'_k, h''_k \geq 0$ fulfill $h'_k + h''_k > 0$.*

We say that fuzzy sets $A_1, \ldots, A_n : [a, b] \to [0, 1]$ constitute a generalized fuzzy partition *of $[a, b]$ (with nodes $x_1, \ldots, x_n$ and margins h'_k, h''_k, $k = 1, \ldots, n$), if for every $k = 1, \ldots, n$, the following three conditions are fulfilled:*

1. *(locality) — $A_k(x) > 0$ if $x \in (x_k - h'_k, x_k + h''_k)$, and $A_k(x) = 0$ if $x \in [a, b] \setminus (x_k - h'_k, x_k + h''_k)$;*
2. *(continuity) — A_k is continuous on $[x_k - h'_k, x_k + h''_k]$;*
3. *(covering) — for $x \in [a, b]$, $\sum_{k=1}^{n} A_k(x) > 0$.*
4. *(monotonicity) — $A_k(x)$, for $k = 2, \ldots, n$, strictly increases on $[x_k - h'_k, x_k]$ and $A_k(x)$, for $k = 1, \ldots, n-1$, strictly decreases on $[x_k, x_k + h''_k]$;*

An (h, h', h'')-*uniform* generalized fuzzy partition of $[a, b]$ is defined for equidistant nodes $x_k = a + h(k-1)$, $k = 1, \ldots, n$, where $h = (b-a)/(n-1)$; $h', h'' > h/2$ and two additional properties are satisfied:

4. $A_k(x) = A_{k-1}(x-h)$ for all $k = 2, \ldots, n-1$ and $x \in [x_k, x_{k+1}]$, and $A_{k+1}(x) = A_k(x-h)$ for all $k = 2, \ldots, n-1$ and $x \in [x_k, x_{k+1}]$.
5. $h'_1 = h''_n = 0$, $h''_1 = h'_2 = \ldots = h''_{n-1} = h'_n = h'$ and for all $k = 2, \ldots, n-1$ and all $x \in [0, h']$, $A_k(x_k - x) = A_k(x_k + x)$.

An (h, h')-uniform generalized fuzzy partition of $[a, b]$ can also be defined using the *generating function* $A_0 : [-1, 1] \to [0, 1]$, which is assumed to be *even*[1], continuous and positive everywhere except for on boundaries, where it vanishes. Then, basic functions A_k of an (h, h')-uniform generalized fuzzy partition are shifted copies of A_0 in the sense that

$$A_1(x) = \begin{cases} A_0\left(\frac{x-x_1}{h'}\right), & x \in [x_1, x_1 + h'], \\ 0, & otherwise, \end{cases}$$

and for $k = 2, \ldots, n-1$,

$$A_k(x) = \begin{cases} A_0\left(\frac{x-x_k}{h'}\right), & x \in [x_k - h', x_k + h'], \\ 0, & otherwise. \end{cases}, \tag{1}$$

$$A_n(x) = \begin{cases} A_0\left(\frac{x-x_n}{h'}\right), & x \in [x_n - h', x_n], \\ 0, & otherwise, \end{cases}$$

2.2 F^0-transform

The direct and inverse F^0transform (originally just as 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 [2] 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^0-transform of f can be introduced in a manner analogous to the case of a function of one variable.

Definition 2. *Let a function f be given at points $(p_i, q_j) \in [a, b] \times [c, d]$, for which $i = 1, \ldots, N$ and $j = 1, \ldots, M$, and $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$, where $n < N$ and $m < M$, be basic functions that form generalized fuzzy partitions*

[1] The function $A_0 : [-1, 1] \to \mathbb{R}$ is even if for all $x \in [0, 1]$, $A_0(-x) = A_0(x)$.

of $[a,b]$ *and* $[c,d]$ *respectively. We say that the* $n \times m$*-matrix of real numbers* $\mathbf{F}[f] = (F_{kl})_{nm}$ *is the discrete* F^0*-transform of* f *with respect to* $A_1, \ldots, A_n$ *and* $B_1, \ldots, B_m$ *if*

$$F_{kl} = \frac{\sum_{j=1}^{M} \sum_{i=1}^{N} f(p_i, q_j) A_k(p_i) B_l(q_j)}{\sum_{j=1}^{M} \sum_{i=1}^{N} A_k(p_i) B_l(q_j)} \tag{2}$$

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

The inverse F^0-transform of a discrete function f of two variables is defined as follows.

Definition 3. *Let* $A_1, \ldots, A_n$ *and* $B_1, \ldots, B_m$ *be basic functions that form generalized fuzzy partitions of* $[a,b]$ *and* $[c,d]$*, respectively. Let function* f *be defined on the set of points* $(p_i, q_j) \in P \times Q$ *where* $P = \{p_1, \ldots, p_N\} \subseteq [a,b]$, $Q = \{q_1, \ldots, q_M\} \subseteq [c,d]$ *and both sets* P *and* Q *are sufficiently dense with respect to corresponding partitions, i.e* $\forall k, l\ \exists i, j;\ A_k(p_i) B_l(p_j) > 0$*. Moreover, let* $\mathbf{F}[f] = (F_{kl})_{nm}$ *be the discrete* F^0*-transform of* f *w.r.t.* $A_1, \ldots, A_n$ *and* $B_1, \ldots, B_m$*. Then, the function* $\hat{f} : P \times Q \to \mathbb{R}$ *represented by*

$$\hat{f}(p_i, q_j) = \frac{\sum_{k=1}^{n} \sum_{l=1}^{m} F_{kl} A_k(p_i) B_l(q_j)}{\sum_{k=1}^{n} \sum_{l=1}^{m} A_k(p_i) B_l(q_j)} \tag{3}$$

is called the inverse F^0-transform *of* f.

2.3 $\mathbf{F^1}$-transform

We can generalize the F-transform with constant components to the F^1-transform with linear components. The latter are orthogonal projections of an original function f onto a linear subspace of functions with the basis of polynomials $P_k^0 = 1, P_k^1 = (x - x_k)$. We say that the n-tuple

$$F^1[f] = [F_1^1, \ldots, F_n^1] \tag{4}$$

is the F^1*-transform* of f w.r.t. $A_1, \ldots, A_n$ where the $k-$th component F_k^1 is defined by

$$F_k^1 = c_{k,0} P_k^0 + c_{k,1} P_k^1,\ k = 1, \ldots, n. \tag{5}$$

For the h-uniform fuzzy partition and the triangular-shaped basic functions we can compute the coefficients $c_{k,0}, c_{k,1}$ for each $k = 1, \ldots, n$ as follows

$$c_{k,0} = \frac{1}{h} \sum_{i=1}^{N} f(p_i) A_k(p_i), \tag{6}$$

$$c_{k,1} = \frac{12}{h^3} \sum_{i=1}^{N} f(p_i)(p_i - x_k) A_k(p_i). \tag{7}$$

It can be shown that the coefficient $c_{k,0}$ is equal to the F-transform component $F_k,\ k = 1, \ldots, n$. The next theorem shows the important property of the coefficient $c_{k,1}$ which will be useful for the proposed edge detection technique. The theorem is formulated for the continuous version of the F^1-transform.

Theorem 1. *Let $A_1, \ldots, A_n$, be an h-uniform partition of $[a, b]$, let functions f and A_k, $k = 1, \ldots, n$ be four times continuously differentiable on $[a, b]$, and let $F^1[f] = [F_1^1, \ldots, F_n^1])$ be the F^1-transform of f with respect to $A_1, \ldots, A_n$,. Then, for every $k = 1, \ldots, n$, the following estimation holds true:*

$$c_{k,1} = f'(x_k) + O(h). \tag{8}$$

We refer to [10] for a proof of **Theorem 1** and for a detailed description of the F^1-transform.

3 Image Registration

This section focuses on applications of the F-transforms theory into image registration. The developed method is divided into four steps: feature extraction; feature matching; image mapping; image fusion.

3.1 Feature Extraction

Let us remark that there exist many algorithms for feature extraction, the most used are FAST, ORB or SIFT [11]. In this contribution, we understand the problem of feature extraction as a procedure that selects small corner areas in the image. According to the accepted terminology, we call the latter *point features*. Extracted point features in a reference and sensed images should be detected on the similar places even if the sensed image is rotated, resized or has different intensity. We propose an original technique of point features detection using the first degree F-transform (F^1-transform) adopted from [9].

By **Theorem 1**, coefficients $c_{k,1}$ of the F^1-transform give us a vector whose components approximate the first derivative of the original function at certain nodes. We use these coefficients as components of the inverse F-transform and we get the approximation of the first derivative of the original image function in each pixel.

Let triangular fuzzy sets $A_1, \ldots, A_n$ establish a fuzzy partition of $[1, N]$ and triangular $B_1, \ldots, B_m$ do the same for $[1, M]$. Let $x_1, \ldots, x_n \in [1, N]$, $h_x = x_{k+1} - x_k$, $k = 1, \ldots, n$ and $y_1, \ldots, y_m \in [1, M]$, $h_y = y_{l+1} - y_l$, $l = 1, \ldots, m$ be nodes on $[1, N]$, $[1, M]$ respectively. Then we can determine the approximation of the first derivative for each $(p_i, p_j) \in D$ in the horizontal direction

$$G_x(p_i, p_j) \approx \sum_{k=1}^{n} \sum_{l=1}^{m} c_{k,1}(y_l) A_k(p_i) B_l(p_j) \tag{9}$$

and in the vertical direction

$$G_y(p_i, p_j) \approx \sum_{k=1}^{n} \sum_{l=1}^{m} c_{l,1}(x_k) A_k(p_i) B_l(p_j) \tag{10}$$

as the inverse F-transform of the image function u where the coefficients $c_{k,1}(y_l)$, $c_{l,1}(x_k)$, $k = 1, \ldots, n$, $l = 1, \ldots, m$ are given by the F^1-transform

$$c_{k,1}(y_l) = \frac{12}{h_x^3} \sum_{i=1}^{N} f(p_i, y_l)(p_i - x_k)A_k(p_i), \tag{11}$$

$$c_{l,1}(x_k) = \frac{12}{h_y^3} \sum_{j=1}^{M} f(x_k, p_j)(p_j - y_l)B_l(p_j). \tag{12}$$

Then, the gradient magnitude G of an edge at point (p_i, p_j) is computed as

$$G(p_i, p_j) = \sqrt{G_x(p_i, p_j)^2 + G_y(p_i, p_j)^2} \tag{13}$$

and the gradient angle Θ is determined by

$$\Theta(p_i, p_j) = \arctan \frac{G_y(p_i, p_j)}{G_x(p_i, p_j)} \tag{14}$$

where for simplicity in according to [9] the gradient angle will be quantized by: $\Theta^Q : \Theta \to \{0, 45, 90, 135\}$.

Definition 4. *We say that a corner is a set of neighboring pixels (we call them corner points) where at least three different quantized angles show up. The center of gravity of a corner is called a feature point.*

Many corner points can be found in an image. It may happen that corner points are close to each other, and in this case, we have to choose only one of them. We modify computer graphic *flood fill* algorithm to detect clusters of close corner points and then compute centers of gravity of each cluster. These centers constitute the set of point features.

Image 1 simply demonstrate comparison of proposed algorithm with SIFT [11]. Two top images are original; two middle images were firstly vertically flipped, processed by both algorithms and then flipped back for comparison. Two bottom images were lighten and then processed by both algorithms. The result show that both algorithms works correctly for this simplest image modification. The feature points detection should hold invariance for difficult cases such as scale transformation or rotation. The research of rotation and scale invariance of the proposed algorithm deserve future work.

3.2 Feature Matching

In this step, a correspondence between the point features detected in the reference and sensed images is established. As a main technique (among various similarity measures or spatial relationships) we propose to measure similarity by a (inverse) distance between F^0-transform components of various levels.

In more details, the lowest (first) level is comprised by the F^0-transform components of image f and corresponds to the discretization given by the respective

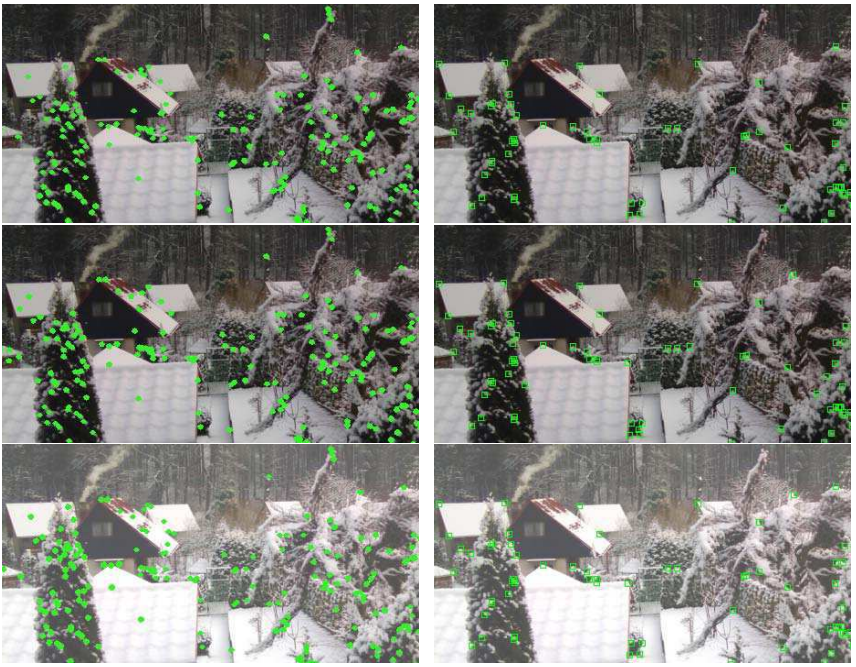

Fig. 1. Left: inpainted circles by SIFT. Right: inpainted squares by the own algorithm.

fuzzy partition of the domain. This first level $F^{(1)}[f]$ is given by the F^0-transform of f so that

$$F^{(1)}[f] = F[f] = (F_{11}, ..., F_{nm}). \tag{15}$$

The vector of the F^0-transform components $(F_{11}, ..., F_{nm})$ is a linear representation of a respective matrix of components. This first level serves as a new image for the F^0-transform components of the second level and so on. For a higher level ℓ we propose the following recursive formula:

$$F^{(\ell)}[f] = F[F^{(\ell-1)}] = (F_{11}^{(\ell-1)}, ..., F_{n_{(\ell-1)}m_{(\ell-1)}}^{(\ell-1)}). \tag{16}$$

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

The F^0-transform based similarity S of two image functions $f, g \in \mathcal{I}$ is proposed to be as follows:

$$S(f,g) = 1 - |F^{fin} - G^{fin}| \cdot \frac{\sum_{k=1}^{n}\sum_{l=1}^{m}|F_{kl}^{(1)} - G_{kl}^{(1)}|}{nm} \tag{17}$$

where F^{fin}, G^{fin} are the top F^0-transform components of f and g, and $F_{kl}^{(1)}, G_{kl}^{(1)}$, $k = 1, \ldots, n$, $l = 1, \ldots, m$ are the first level F^0-transform components of f and g, respectively. The justification that S is a similarity measure with respect to the product t-norm was given in [4].

We propose the following procedure in order to establishes a feature matching between two point features:

- choose two point features P and Q from the reference and sensed images respectively,
- create square areas S_P and S_Q of the same size around P and Q as center points,
- compute similarity $S(S_P, S_Q)$ according to (17),
- establish a feature matching between P and Q, if there is no point R in the sensed image such that $S(S_P, S_Q) < S(S_P, S_R)$.

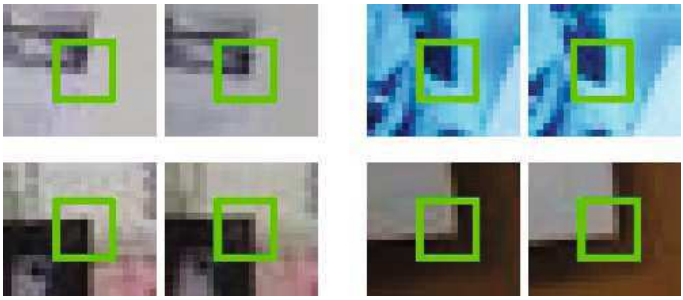

Fig. 2. Green inpainted square illustrates an area around a point feature. Every couple of image fragments illustrates detected matching between corresponding point features.

The Figure 2 demonstrates feature matching in different images. Because of images can be obtained in different time, different light conditions or some noise can be in images, it is necessary to create similarity measure which will be robust enough to these changes. The used F^0-transform based similarity approximate image function with user defined accuracy defined by value of h parameter - see approximation theorem 2 in [2]. This property allow to compare images even if there are changes between them.

3.3 Image Mapping

The goal of the image mapping step is to find a shift between two images in x and y axis. The image mapping should determine a perspective distortion and then transform images in such a way that they would fit each other. For simplicity of demonstration, we show on Fig. 3 the result of registration of seven images where detected points features were shifted, but not interpolated. The shift between every two images is computed as an average shift between all pairs of matched points features. Figure 4 demonstrates result of existing software called AutoStitch published in [12]. It is obvious that the proposed approach is on the right way, but should be improved.

3.4 Image Fusion

Image fusion is the last step of our registration algorithm. It is applied to each overlapping part of input images with the purpose to extract the best representative pixels. The detailed description of used image fusion that has been applied in the proposed registration algorithm is in [8]. Figure 3 shows the final result of the proposed algorithm of image registration that uses seven input images. The result image is not perfect, there are lot of artifacts inside. The reason why the artifacts are there is because of mapping function without flexible grid cannot work with perspective distortion properly. Therefore some existing algorithm for an landscape composition can achieve better result.

Fig. 3. Registered and fused image of seven images

Fig. 4. Same result obtained by AutoStitch [12]

4 Conclusion

In the paper we apply the techniques of the F^0-transform and the F^1-transform to the problem of image registration. The technique of F^0-transform is used in computation of image similarity and that of the F^1-transform is a part of the gradient detection algorithm. The proposed theory is applied to image registration problem where F^1-transform edge detection is a base of detecting important areas (corners) in image. In order to find a correspondence between corner areas we used a newly proposed image similarity algorithm that helps in computation

of shift vectors between images. Finally, overlapped part of images are inputs of image fusion algorithm that is based on F^0-transform. As a final result panorama image is created. Measuring of computation time show that the process of detection of feature points and map them is lower than one second without any perfect programming skill. The paper demonstrate how the unique technique of F-transform can be successfully used in various algorithms that comprise the multi-step problem of registration. The future work will be focused on the last step: to find image mapping function where the F-transform can be used (again) for operation of an image interpolation. Without these missing part the result is now worse than result obtained by existing approach called AutoStitch.

Acknowledgement. This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070). This work was also supported by SGS14/PrF/2013 project and "SGS/PrF/2014 – Vyzkum a aplikace technik soft-computingu ve zpracovani obrazu" project.

References

1. Perfilieva, I.: Fuzzy transforms: Theory and applications. Fuzzy Sets and Systems 157, 993–1023 (2006)
2. Perfilieva, I., De Baets, B.: Fuzzy Transform of Monotonous Functions with Applications to Image Processing. Information Sciences 180, 3304–3315 (2010)
3. Hurtik, P., Perfilieva, I.: Image compression methodology based on fuzzy transform. In: Herrero, Á., Snášel, V., Abraham, A., Zelinka, I., Baruque, B., Quintián, H., Calvo, J.L., Sedano, J., Corchado, E. (eds.) Int. Joint Conf. CISIS'12-ICEUTE'12-SOCO'12. AISC, vol. 189, pp. 525–532. Springer, Heidelberg (2013)
4. Hurtik, P., Perfilieva, I.: Image compression methodology based on fuzzy transform using block similarity. In: 8th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2013). Atlantis Press (2013)
5. Hurtik, P., Perfilieva, I.: Image Reduction/Enlargement Methods Based on the F-Transform, pp. 3–10. European Centre for Soft Computing, Asturias (2013)
6. Zitova, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing 21(11), 977–1000 (2003)
7. Perfilieva, I., Daňková, M.: Image fusion on the basis of fuzzy transforms. In: Proc. 8th Int. FLINS Conf., Madrid, pp. 471–476 (2008)
8. Vajgl, M., Perfilieva, I., Hod'kov, P.: Advanced F-Transform-Based Image Fusion. Advances in Fuzzy Systems 2012 (2012)
9. Perfilieva, I., Hodáková, P., Hurtík, P.: F^1-transform edge detector inspired by Canny's algorithm. In: Greco, S., Bouchon-Meunier, B., Coletti, G., Fedrizzi, M., Matarazzo, B., Yager, R.R. (eds.) IPMU 2012, Part I. CCIS, vol. 297, pp. 230–239. Springer, Heidelberg (2012)
10. Perfilieva, I., Daňková, M., Bede, B.: Towards F-transform of a higher degree. Fuzzy Sets and Systems 180, 3–19 (2011)
11. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60 (2004)
12. Brown, M., Lowe, D.: Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision 74(1), 59–73 (2007)

P. Hodakova, I. Perfilieva and P. Hurtik. F-transform and its Extension as Tool for Big Data Processing. *Proc. of the 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2014)*, 374–383, 2014.

F-transform and Its Extension as Tool for Big Data Processing

Petra Hodáková, Irina Perfilieva, and Petr Hurtík

University of Ostrava,
Institute for Research and Applications of Fuzzy Modeling,
Centre of Excellence IT4Innovations,
30. dubna 22, 701 03 Ostrava 1,
Czech Republic
{Petra.Hodakova,Irina.Perfilieva,Petr.Hurtik}@osu.cz

Abstract. In this contribution, the extension of F-transform to F^s-transform for functions of two variables is introduced. The F^s-transform components are characterized as orthogonal projections, and some of their properties are discussed. The aim of this study is to present the possibility of using the technique of F^1-transform in big data processing and to suggest a good searching mechanism for large databases.

Keywords: F-transform, F^s-transform, big data.

1 Introduction

In this paper, we focus on the F-transform technique and its extension to the F-transform of a higher degree (F^s-transform). The aim is to describe the formal conception of this technique, demonstrate some of its properties and introduce its application to big data processing.

"Big data" refers to data that exceeds the processing capacity of conventional database systems, e.g., geographical data, medical data, social networks, and banking transactions. The data are generally of high volume, high velocity or high variety. The benefit gained from the ability to process large amounts of data is that we can create effective models and use them for databases of a custom size.

It is not feasible to handle large databases with classical analytic tools. We are not able to process every item of data in a reasonable amount of time. Therefore, we are searching for an alternative way to obtain the desired information from these data.

The F-transform is a technique that was developed as tool for a fuzzy modeling [1]. Similar to conventional integral transforms (the Fourier and Laplace transforms, for example), the F-transform performs a transformation of an original universe of functions into a universe of their "skeleton models". Each component of the resulting skeleton model is a weighted local mean of the original function over an area covered by a corresponding basic function. The F-transform is a

A. Laurent et al. (Eds.): IPMU 2014, Part III, CCIS 444, pp. 374–383, 2014.
© Springer International Publishing Switzerland 2014

simplified representation of the original function, and it can be used instead of the original function to make further computations easier.

Initially, the F-transform was introduced for functions of one or two variables. This method proved to be very general and powerful in many applications. The F-transform of functions of two variables shows great potential in applications involving image processing, particularly, image compression [2], image fusion [3], and edge detection [4], [5].

Generalization of the ordinary F-transform to the F-transform of a higher degree in the case of functions of one variable was introduced in [6]. An extension of the F-transform of the first degree (F^1-transform) to functions of many variables was introduced in [7]. Many interesting properties and results have been proven in those studies .

The aim of this contribution is to introduce the F-transform of a higher degree (F^s-transform) for functions of two variables and to show how this technique can be successfully applied for searching for patterns in big data records.

The paper is organized as follows: Section 2 recalls the basic tenets of the fuzzy partition and introduces a particular Hilbert space. In Section 3, the F^s-transform of functions of two variables is introduced. The inverse F^s-transform is established in Section 4. In Section 5, an illustrative application of F^1-transform to big data is presented. Finally, conclusions and comments are provided in Section 6.

2 Preliminaries

In this section, we briefly recall the basic tenets of the fuzzy partition and introduce a particular Hilbert space.

2.1 Generalized Fuzzy Partition

Let us recall the concept of a generalized fuzzy partition [8].

Definition 1. *Let $x_0, x_1, \ldots, x_n, x_{n+1} \in [a,b]$ be fixed nodes such that $a = x_0 \leq x_1 < \ldots < x_n \leq x_{n+1} = b$, $n \geq 2$. We say that the fuzzy sets $A_1, \ldots, A_n : [a,b] \to [0,1]$ constitute a generalized fuzzy partition of $[a,b]$ if for every $k = 1, \ldots, n$ there exists $h_k^{'}, h_k^{''} \geq 0$ such that $h_k^{'} + h_k^{''} > 0$, $[x_k - h_k^{'}, x_k + h_k^{''}] \subseteq [a,b]$ and the following conditions are fulfilled:*

1. *(locality) – $A_k(x) > 0$ if $x \in (x_k - h_k^{'}, x_k + h_k^{''})$ and $A_k(x) = 0$ if $x \in [a,b] \setminus [x_k - h_k^{'}, x_k + h_k^{''}]$;*
2. *(continuity) – A_k is continuous on $[x_k - h_k^{'}, x_k + h_k^{''}]$;*
3. *(covering) – for $x \in [a,b]$, $\sum_{k=1}^{n} A_k(x) > 0$.*

By the *locality* and *continuity*, it follows that $\int_a^b A_k(x)dx > 0$.

If the nodes $x_0, x_1, \ldots, x_n, x_{n+1}$ are equidistant, i.e., for all $k = 1, \ldots, n+1$, $x_k = x_{k-1} + h$, where $h = (b-a)/(n+1)$, $h^{'} > h/2$ and two additional properties are satisfied,

6. $h_1^{'} = h_1^{''} = h_2^{'} = \ldots = h_{n-1}^{''} = h_n^{'} = h_n^{''} = h^{'}$, and $A_k(x_k - x) = A_k(x_k + x)$ for all $x \in [0, h^{'}]$, $k = 1, \ldots, n$;
7. $A_k(x) = A_{k-1}(x - h)$ and $A_{k+1}(x) = A_k(x - h)$ for all $x \in [x_k, x_{k+1}]$, $k = 2, \ldots, n-1$;

then the fuzzy partition is called an $(h, h^{'})$-*uniform generalized fuzzy partition.*

Remark 1. A fuzzy partition is called a Ruspini partition if

$$\sum_{k=1}^{n} A_k(x) = 1, \ x \in [a, b].$$

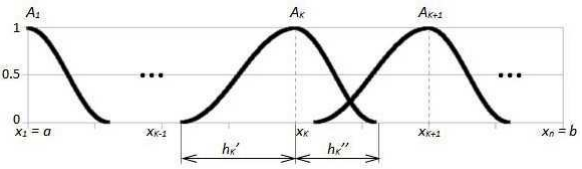

Fig. 1. Example of a generalized fuzzy partition of $[a, b]$

The concept of generalized fuzzy partition can be easily extended to the universe $D = [a, b] \times [c, d]$. We assume that $[a, b]$ is partitioned by $A_1, \ldots, A_n$ and that $[c, d]$ is partitioned by $B_1, \ldots, B_m$, according to **Definition 1**. Then, the Cartesian product $[a, b] \times [c, d]$ is partitioned by the Cartesian product of corresponding partitions where a basic function $A_k \times B_l$ is equal to the product $A_k \cdot B_l$, $k = 1, \ldots, n$, $l = 1, \ldots, m$.

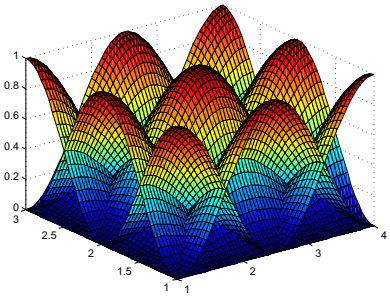

Fig. 2. Example of a cosine-shaped fuzzy partition of $[a, b] \times [c, d]$

For the remainder of this paper, we fix the notation related to fuzzy partitions of the universe $D = [a, b] \times [c, d]$.

2.2 Spaces $L_2(A_k)$, $L_2(A_k) \times L_2(B_l)$

Throughout the following subsections, we fix integers k, l from $\{1, \ldots, n\}$, $\{1, \ldots, m\}$, respectively.

Let $L_2(A_k)$ be a Hilbert space of square-integrable functions $f : [x_{k-1}, x_{k+1}] \to \mathbb{R}$ with the inner product $\langle f, g \rangle_k$ given by

$$\langle f, g \rangle_k = \int_{x_{k-1}}^{x_{k+1}} f(x) g(x) A_k(x) dx. \tag{1}$$

Analogously, the same holds for the space $L_2(B_l)$.

Then, the Hilbert space $L_2(A_k) \times L_2(B_l)$ of functions of two variables $f : [x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}] \to \mathbb{R}$ is given by the Cartesian product of the respective spaces $L_2(A_k)$ and $L_2(B_l)$. The inner product is defined analogously to (1).

Remark 2. The functions $f, g \in L_2(A_k) \times L_2(B_l)$ are *orthogonal* in $L_2(A_k) \times L_2(B_l)$ if

$$\langle f, g \rangle_{kl} = 0.$$

In the sequel, by $L_2([a, b] \times [c, d])$, we denote a set of functions $f : [a, b] \times [c, d] \to \mathbb{R}$ such that for all $k = 1, \ldots, n$, $l = 1, \ldots, m$, $f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]} \in L_2(A_k) \times L_2(B_l)$, where $f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]}$ is the restriction of f on $[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]$.

2.3 Subspaces $L_2^p(A_k)$, $L_2^s(A_k \times B_l)$

Let space $L_2^p(A_k)$, $p \geq 0$, ($L_2^r(B_l)$, $r \geq 0$) be a closed linear subspace of $L_2(A_k)$ ($L_2(B_l)$) with the orthogonal basis given by polynomials

$$\{P_k^i(x)\}_{i=0,\ldots,p}, \ (\{Q_l^j(y)\}_{j=0,\ldots,r}),$$

where p (r) denotes a degree of polynomials and orthogonality is considered in the sense of (2).

Then, we can introduce space $L_2^s(A_k \times B_l)$, $s \geq 0$ as a closed linear subspace of $L_2(A_k) \times L_2(B_l)$ with the basis given by orthogonal polynomials

$$\{S_{kl}^{ij}(x, y)\}_{i=0,\ldots,p;\ j=0,\ldots,r;\ i+j \leq s} = \{P_k^i(x) \cdot Q_l^j(y)\}_{i=0,\ldots,p;\ j=0,\ldots,r;\ i+j \leq s}. \tag{2}$$

Remark 3. Let us remark that the space $L_2^s(A_k \times B_l)$ is not the same as the Cartesian product $L_2^p(A_k) \times L_2^r(B_l)$; the difference is in using fewer of the possible combinations of orthogonal basis polynomials. Therefore, $s \leq (p+1)(r+1)$. In the case where $s = (p+1)(r+1)$, the space $L_2^s(A_k \times B_l)$ coincides with $L_2^p(A_k) \times L_2^r(B_l)$.

In point of fact, s is the maximal degree of products $P_k^i(x) Q_l^j(y)$ such that $i + j \leq s$. For example, the basis of the space $L_2^1(A_k \times B_l)$ is established by the following polynomials

$$\underbrace{P_k^0(x) Q_l^0(y)}_{S_{kl}^{00}(x,y)}, \ \underbrace{P_k^1(x) Q_l^0(y)}_{S_{kl}^{10}(x,y)}, \underbrace{P_k^0(x) Q_l^1(y)}_{S_{kl}^{01}(x,y)}. \tag{3}$$

The following lemma characterizes the orthogonal projection of a function $f \in L_2([a,b] \times [c,d])$ or the best approximation of f in the space $L_2^s(A_k \times B_l)$.

Lemma 1. *Let $f \in L_2([a,b] \times [c,d])$ and let $L_2^s(A_k \times B_l)$ be a closed linear subspace of $L_2(A_k) \times L_2(B_l)$, as specified above. Then, the orthogonal projection F_{kl}^s of f on $L_2^s(A_k \times B_l)$, $s \geq 0$, is equal to*

$$F_{kl}^s = \sum_{0 \leq i+j \leq s} c_{kl}^{ij} S_{kl}^{ij} \tag{4}$$

where

$$c_{kl}^{ij} = \frac{\langle f, S_{kl}^{ij} \rangle_{kl}}{\langle S_{kl}^{ij}, S_{kl}^{ij} \rangle_{kl}} = \frac{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} f(x,y) S_{kl}^{ij}(x,y) A_k(x) B_l(y) dx\ dy}{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} (S_{kl}^{ij}(x,y))^2 A_k(x) B_l(y) dx\ dy}. \tag{5}$$

3 Direct F^s-transform

Now let $f \in L_2([a,b] \times [c,d])$ and let $L_2^s(A_k \times B_l)$, $s \geq 0$ be a space with the basis given by

$$\{S_{kl}^{ij}(x,y)\}_{i=0,\ldots,p;\ j=0,\ldots,r;\ i+j \leq s}.$$

In the following, we define the direct F^s-transform of the given function f.

Definition 2. *Let $f \in L_2([a,b] \times [c,d])$. Let F_{kl}^s, $s \geq 0$ be the orthogonal projection of $f|_{[x_{k-1},x_{k+1}] \times [y_{l-1},y_{l+1}]}$ on $L_2^s(A_k \times B_l)$, $k = 1, \ldots, n$, $l = 1, \ldots, m$. We say that $(n \times m)$ matrix $\mathbf{F_{nm}^s}[f]$ is the direct F^s-transform of f with respect to $A_1, \ldots, A_n$, $B_1, \ldots, B_m$, where*

$$\mathbf{F_{nm}^s}[f] = \begin{pmatrix} F_{11}^s & \ldots & F_{1m}^s \\ \vdots & \vdots & \vdots \\ F_{n1}^s & \ldots & F_{nm}^s \end{pmatrix}. \tag{6}$$

F_{kl}^s, $k = 1, \ldots, n$, $l = 1, \ldots, m$ is called the F^s-transform component.

By **Lemma 1**, the F^s-transform components have the representation given by (4).

We will briefly recall the main properties of the F^s-transform, $s \geq 0$.

(A) The F^s-transform of f, $s \geq 0$, is an image of a linear mapping from $L_2([a,b] \times [c,d])$ to $L_2^s(A_1 \times B_1) \times \ldots \times L_2^s(A_n \times B_m)$ where, for all functions $f, g, h \in L_2([a,b] \times [c,d])$ such that $f = \alpha g + \beta h$, where $\alpha, \beta \in \mathbb{R}$, the following holds:

$$\mathbf{F_{nm}^s}[f] = \alpha \mathbf{F_{nm}^s}[g] + \beta \mathbf{F_{nm}^s}[h]. \tag{7}$$

(B) Let $f \in L_2([a,b] \times [c,d])$. The kl-th component of the F^s-transform, $s \geq 0$, of the given function f gives the minimum to the function

$$c_{kl}^{00}, \ldots, c_{kl}^{ij} = \int_a^b \int_c^d (f(x,y) - \sum_{i+j \leq s} c_{kl}^{ij} S_{kl}^{ij})^2 A_k(x) B_l(y) dx\ dy, \tag{8}$$

Therefore, F_{kl}^s is the best approximation of f in $L_2^s(A_k \times B_l)$, $k = 1, \ldots, n, l = 1, \ldots, m$.

(C) Let f be a polynomial of degree $t \leq s$. Then, any F^s-transform component F^s_{kl}, $s \geq 0$, $k = 1, \ldots, n$, $l = 1, \ldots, m$ coincides with f on $[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]$.

(D) Every F^s-transform component F^s_{kl}, $s \geq 1$, $k = 1, \ldots, n$, $l = 1, \ldots, m$, fulfills the following recurrent equation:

$$F^s_{kl} = F^{s-1}_{kl} + \sum_{i+j=s} c^{ij}_{kl} S^{ij}_{kl}. \tag{9}$$

The following lemma describes the relationship between the F^0-transform and F^s-transform components.

Lemma 2. *Let* $\mathbf{F^s_{nm}}[f] = (F^s_{11}, \ldots, F^s_{nm})$, *where* F^s_{kl}, $k = 1, \ldots, n$, $l = 1, \ldots, m$ $s \geq 0$ *is given by (4), be the* F^s*-transform of* f *with respect to the given partition* $\{A_k \times B_l\}$, $k = 1, \ldots, n$, $l = 1, \ldots, m$. *Then,* $(c^{00}_{11}, \ldots, c^{00}_{nm})$ *is the* F^0*-transform of* f *with respect to* $A_k \times B_l$, $k = 1, \ldots, n$, $l = 1, \ldots, m$.

The proof is analogous to that of the case of functions of one variable given in [6].

Any F^s-transform component $F^0_{kl}, F^1_{kl}, \ldots, F^s_{kl}$, $k = 1, \ldots, n$, $l = 1, \ldots, m$ $s \geq 0$, can approximate the original function $f \in L_2([a, b] \times [c, d])$ restricted to $[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]$. The following lemma says that the quality of approximation increases with the degree of the polynomial.

Lemma 3. *Let the polynomials* F^s_{kl}, F^{s+1}_{kl}, $k = 1, \ldots, n$, $l = 1, \ldots, m$ $s \geq 0$, *be the orthogonal projections of* $f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]}$ *on the subspaces* $L^s_2(A_k \times B_l)$ *and* $L^{s+1}_2(A_k \times B_l)$, *respectively. Then,*

$$\| f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]} - F^{s+1}_{kl} \|_{kl} \leq \| f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]} - F^s_{kl} \|_{kl} . \tag{10}$$

The proof is analogous to that of the case of functions of one variable given in [6].

3.1 Direct F^1-transform

In this section, we assume that $s = 1$ and give more details to the F^1-transform and its components.

The F^1-transform components F^1_{kl}, $k = 1, \ldots, n, l = 1, \ldots, m$, are in the form of linear polynomials

$$F^1_{kl} = c^{00}_{kl} + c^{10}_{kl}(x - x_k) + c^{01}_{kl}(y - y_l), \tag{11}$$

where the coefficients are given by

$$c^{00}_{kl} = \frac{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} f(x, y) A_k(x) B_l(y) dx \; dy}{(\int_{x_{k-1}}^{x_{k+1}} A_k(x) dx)(\int_{y_{l-1}}^{y_{l+1}} B_l(y) dy)}, \tag{12}$$

$$c_{kl}^{10} = \frac{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} f(x,y)(x-x_k)A_k(x)B_l(y)dx\ dy}{(\int_{x_{k-1}}^{x_{k+1}} (x-x_k)^2 A_k(x)dx)(\int_{y_{l-1}}^{y_{l+1}} B_l(y)dy)}, \tag{13}$$

$$c_{kl}^{01} = \frac{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} f(x,y)(y-y_l)A_k(x)B_l(y)dx\ dy}{(\int_{x_{k-1}}^{x_{k+1}} A_k(x)dx)(\int_{y_{l-1}}^{y_{l+1}} (y-y_l)^2 B_l(y)dy)}. \tag{14}$$

Lemma 4. *Let $f \in L_2([a,b]\times[c,d])$ and $\{A_k \times B_l\}$, $k=1,\dots,n$, $l=1,\dots,m$ be an (h,h')-uniform generalized fuzzy partition of $[a,b]\times[c,d]$. Moreover, let functions f, A_k, B_l be four times continuously differentiable on $[a,b]\times[c,d]$. Then, for every k,l, the following holds:*

$$c_{kl}^{00} = f(x_k,y_l) + O(h^2), \quad c_{kl}^{10} = \frac{\partial f}{\partial x}(x_k,y_l) + O(h), \quad c_{kl}^{01} = \frac{\partial f}{\partial y}(x_k,y_l) + O(h).$$

The proof can be found in [7].

4 Inverse F^s-transform

The inverse F^s-transform of the original function f is defined as a linear combination of basic functions and F^s-transform components.

Definition 3. *Let $\mathbf{F_{nm}^s}[f] = (F_{kl}^s)$, $k=1,\dots,n$, $l=1,\dots,m$ be the F^s-transform of given function $f \in L_2([a,b]\times[c,d])$. We say that the function $\hat{f}^s : [a,b]\times[c,d] \to \mathbb{R}$ represented by*

$$\hat{f}^s(x,y) = \frac{\sum_{k=1}^n \sum_{l=1}^m F_{kl}^s A_k(x)B_l(y)}{\sum_{k=1}^n \sum_{l=1}^m A_k(x)B_l(y)}, \quad x\in[a,b],\ y\in[c,d], \tag{15}$$

is the inverse F^s-transform of the function f.

Remark 4. From **Definition 3** and property (9) of the F^s-transform components, the recurrent formula below easily follows:

$$\hat{f}^s(x,y) = \hat{f^{s-1}}(x,y) + \frac{\sum_{k=1}^n \sum_{l=1}^m \sum_{i+j=s} c_{kl}^{ij} S_{kl}^{ij} A_k(x)B_l(y)}{\sum_{k=1}^n \sum_{l=1}^m A_k(x)B_l(y)}. \tag{16}$$

In the following theorem, we show that the inverse F^s-transform approximates an original function, and we estimate the quality of the approximation. Based on **Lemma 3**, the quality of the approximation increases with the increase of s.

Theorem 1. *Let $\{A_k \times B_l\}$, $k=1,\dots,n$, $l=1,\dots,m$ be an h-uniform fuzzy partition (with the Ruspini condition) of $[a,b]\times[c,d]$, and let $\hat{f}^s$ be the inverse F^s-transform of f with respect to the given partition. Moreover, let functions f, A_k, and B_l be four times continuously differentiable on $[a,b]\times[c,d]$. Then, for all $(x,y)\in[a,b]\times[c,d]$, the following estimation holds true:*

$$\int_a^b \int_c^d |f(x,y) - \hat{f}^s(x,y)|dxdy \le O(h). \tag{17}$$

5 Illustrative Application

In this section, we present an illustrative application of F^1-transform to big data. The application shows how an effective searching mechanism in large databases can be constructed on the basis of F^1-transform. The detailed characterization is as follows.

Let o be a sound (voice) signal represented by the function $o : T \to V$, where $T = \{0, ..., t_{max}\}$ is a discrete set of regular time moments and V is a certain range.

Assume that the signal o is an example of a very large record (big data), i.e., $t_{max} = 220 \times 60$ sec, sampled at every second. We are given a small sound pattern o^P that is represented by $o^P : T^P \to V$, where T^P is a set of time moments measured in seconds such that $T^P = \{1, \ldots, 6\}$. The goal is to find occurrences of the pattern o^P in the given sound signal o in a reasonable amount of time. See the illustrative example on Fig. 3.

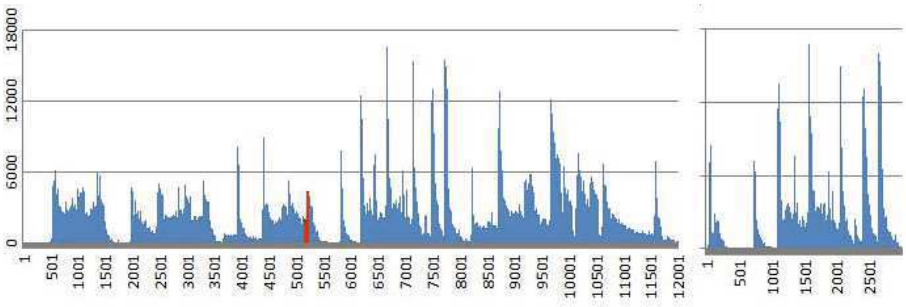

Fig. 3. An extraction of a big sound signal o (*Left*) and a small sound pattern o^P (*Right*). The red mark indicates the first occurrence of the recognized pattern in the given signal.

The naive approach is to make a sliding comparison between the values of the pattern o^P and the values in o. For this comparison, the following measure of closeness can be used:

$$\sum_{j=0}^{|T^P|} |o(t+j) - o^P(j)|, \ t \in T. \tag{18}$$

This method is very computationally complex and time consuming.

Our approach is as follows. We apply the direct F^1-transform to the record of the sound signal o and to the pattern o^P, and we obtain their vectors of components $\mathbf{F^1_n}[o]$, $\mathbf{F^1_m}[o^P]$, respectively. The dimensions of $\mathbf{F^1_n}[o]$, $\mathbf{F^1_m}[o^P]$ are significantly less than the dimensions of the original data o, o^P. Instead of comparing all values of o^P and o, we suggest to compare the components of F^1-transform $\mathbf{F^1_n}[o]$ and $\mathbf{F^1_m}[o^P]$.

Finally, the algorithm is realized by the following steps:

S 1: Read data o and compute $\mathbf{F^1_n}[o] = (F^1_1, \ldots, F^1_n)_o$ w.r.t. the (h, h')-uniform generalized fuzzy partition. This step is realized just once and can be stored independently.

S 2: Read data o^P and compute $\mathbf{F^1_m}[o^P] = (F^1_1, \ldots, F^1_m)_{o^P}$ w.r.t. the same fuzzy partition.

S 3: Compute the measure of closeness (18) between components of $\mathbf{F^1_n}[o]$ and $\mathbf{F^1_m}[o^P]$. The pattern is recognized if the closeness is less than a predefined threshold of tolerance.

Experiment

For our experiment, we took a record of a sound signal, o with $t_{max} = 220 \times 60$ sec, and a record of a small sound pattern, o^P with $t_{max} \approx 6.4$ sec. The records were taken unprofessionally. In fact, the sounds were both part of a piece of music recorded by a microphone integrated in a notebook computer. Therefore, the records are full of noise (because of the microphone, surroundings, etc.) and they may, for example, differ in volume of the sounds.

We applied the naive approach to these records and obtained the following results:

- $5.463 \cdot 10^{12}$ computations,
- run time ≈ 11 h.

Then, we applied the approach based on F^1-transform and obtained the following results:

- $1.081 \cdot 10^7$ computations,
- run time ≈ 0.008 s.

In this experiment, we used a fuzzy partition with triangular shaped fuzzy sets and tested the fuzzy partition for $h = 1 * 10^n,\ n = 2, 4, 6, 8$. The optimal length for the experiment demonstrated above is for $n = 4$. For the larger $n = 6, 8$, multiple false occurrences of the searched pattern were found. A possible solution is to make the algorithm hierarchical, i.e., use the larger h at the beginning and then use the smaller h for the detected results. This approach can achieve extremely fast recognition by making the algorithm sequentially more accurate.

Remark 5. The task discussed here is usually solved as speech recognition, where words are individually separated and then each of them is compared with words in a database. The most similar words are then found as results. The speech recognition is mostly solved by neural networks. Comparing different speech recognition approaches will be our future task.

From a different point of view, this task can be discussed as an example of reducing the dimension of big data. The sub-sampling algorithm is often used in this task. We tried to apply this algorithm to the task above, and, for comparison, we used the same reduction of dimensions as in the F^1-transform algorithm. The sub-sampling algorithm failed; it did not find the searched pattern correctly.

The example demonstrates the effectiveness of the technique of F^1-transform in big data processing. A good searching mechanism for large databases can be developed on the basis of this technique. Similar applications can be developed in the area of image processing, where searching of patterns is a very popular problem. The F^s-transform, $s > 1$, technique can be efficiently applied as well. This will be the focus of our future research.

6 Conclusion

In this paper, we presented the technique of F-transform and our vision of its application in big data processing. We discussed the extension to the F^s-transform, $s \geq 1$, for functions of two variables. We characterized the components of the F^s-transform as orthogonal projections and demonstrated some of their properties. Finally, we introduced an illustrative application of using the F^1-transform in searching for a pattern in a large record of sound signals.

Acknowledgments. The research was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and SGS18/PRF/2014 (Research of the F-transform method and applications in image processing based on soft-computing).

References

1. Perfilieva, I.: Fuzzy transforms: Theory and applications. Fuzzy Sets and Systems 157, 993–1023 (2006)
2. Di Martino, F., Loia, V., Perfilieva, I., Sessa, S.: An image coding/decoding method based on direct and inverse fuzzy transforms. International Journal of Appr. Reasoning 48, 110–131 (2008)
3. Vajgl, M., Perfilieva, I., Hodáková, P.: Advanced F-transform-based image fusion. Advances in Fuzzy Systems (2012)
4. Daňková, M., Hodáková, P., Perfilieva, I., Vajgl, M.: Edge detection using F-transform. In: Proc. of the ISDA 2011, Spain, pp. 672–677 (2011)
5. Perfilieva, I., Hodáková, P., Hurtík, P.: F^1-transform edge detector inspired by canny's algorithm. In: Greco, S., Bouchon-Meunier, B., Coletti, G., Fedrizzi, M., Matarazzo, B., Yager, R.R. (eds.) IPMU 2012, Part I. CCIS, vol. 297, pp. 230–239. Springer, Heidelberg (2012)
6. Perfilieva, I., Daňková, M., Bede, B.: Towards a higher degree F-transform. Fuzzy Sets and Systems 180, 3–19 (2011)
7. Perfilieva, I., Hodáková, P.: F^1-transform of functions of two variables. In: Proc. EUSFLAT 2013, Advances in Intelligent Systems Research, Milano, Italy, pp. 547–553 (2013)
8. Stefanini, L.: F-transform with parametric generalized fuzzy partitions. Fuzzy Sets and Systems 180, 98–120 (2011)

P. Hurtik, P. Hodakova, I. Perfilieva, M. Liberts, and J. Asmuss. Network Attack Detection and Classification by the F-transform. In *The 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015),* 2015.

Network Attack Detection and Classification by the F-transform

Petr Hurtik*, Petra Hodakova*, Irina Perfilieva*†, Mārtiņš Liberts,† and Jūlija Asmuss‡
*Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, Czech Republic
petr.hurtik; petra.hodakova; irina.perfilieva@osu.cz
†Institute of Mathematics and Computer Science, University of Latvia
pm90015@lu.lv, irina.perfilieva@osu.cz
‡Institute of Telecommunication, Riga Technical University, Latvia
julija.asmuss@rtu.lv

***Abstract*—We solve the problem of network attack detection and classification. We discuss the way of generation and simulation of an artificial network traffic data. We propose an efficient algorithm for data classification that is based on the F-transform technique. The algorithm successfully passed all tests and moreover, it showed ability to perform classification in an on-line regime.**

I. Introduction

Understanding the nature of network traffic is essential for proper design and implementation in computer networks and network services. The research community has devoted a great effort to the study of new traffic characterization and classification methods (see, e.g., [1]). The ability to accurately classify and identify network traffic is a central issue for many network operation and research topics including traffic engineering, monitoring, pricing as well as anomaly detecting. The definition of what is anomalous traffic or unwanted traffic is ill-defined and greatly varies among networks. Anomalies (such as *Denial-of-Service (DoS)* attacks, for example) may cause significant changes in a network's traffic levels.

Distributed Denial-of-Service (DDoS) attacks constitute one of the major threats and one of the hardest security problems in today's Internet. There are mainly three categories of DDoS attacks (see, e.g., [2], [3]): volume attacks called flood attacks, protocol attacks and logical attacks. Our paper mainly focuses on flood attacks. The most common DDoS flood attacks target the computer networks bandwidth or connectivity. Bandwidth attacks flood the network with such a high volume of traffic that all available network resources are consumed and legitimate user requests cannot get through, resulting in degraded productivity. Connectivity attacks consume actual server connection resources and the network can no longer process legitimate requests. In this context, traffic volume analysis is required as a method for anomaly detection.

A good structural approach to the DDoS problem, classification of DDoS attacks and DDoS defence schemes was given in [4]. Although there are many monitoring mechanisms against DDoS attacks (see, e.g., [2], [5], [6], [7]), they still need relatively high computational time to identify an attack from a normal packet flow. These mechanisms are usually designed to work on the routing level and are adapted to interfaces of intermediate routers. In order to detect attacks the data from a router includes parameters that indicate a risk of anomalies. Once an attack is identified by the router the immediate response is to determine the attack source and to block its traffic. The blocking part is usually performed under manual control of the administrator of the network.

In this paper, we develop a DDoS detection method based on an on-line processing of a corresponding to traffic time series. Our method is based on the technique of the higher order F-transform that well recommended itself in analysis of time series (see, e.g., [8], [9]). Our research focuses on the ability of the F-transform technique to make a reduction of traffic data and by this, help in classification of computational resources. This approach aims to solve the following major tasks:

- reduce the amount of traffic data and by this, achieve effective use of data analysis techniques, both in time and space;
- extract the relationship between traffic data in order to characterize network traffic, identify patterns and detect anomalies.

The paper is organized as follows. Section II introduces traffic representation and generation tools. Section III contains preliminaries on F-transform as a technique for traffic data compression. Section IV presents a mechanism for a classification of traffic data based on F-transform. Section V shows and discusses experiments. Section VI concludes the results.

II. Preliminaries

A. Network traffic models

Let z be a network traffic (traffic for short) function, indicating by $z(t)$ - the volume of traffic at time $t \geq 0$. This function represents aggregated traffic (it consists of arrival packets of all connections at the input of a server).

We work with traffic time series $z(t_i)$, $i = 0,1,2,\dots$, where $t_{i-1} < t_i$, $i = 0,1,2,\dots$. Without confusions, we use $z(t)$ and $z(t_i)$ to represent the traffic function (in the continuous case and the discrete case, respectively).

We suppose that classes of traffic time series are given by several representative functions, which form a reference database. We consider classes of normal traffic and anomalous

or unwanted traffic and solve the problem of classification (pattern recognition) for a new traffic time series. It is important to identify new traffic class or classify this traffic as unknown.

When studying a traffic classification technique with real traces, it is important to have a baseline for traffic classification that will be used as a reference. Because it is very difficult to obtain a dataset that is representative of real network activities and contains both normal and attack traffic, another option (i.e., to generate the traffic that will be analyzed) has been proposed.

Many research papers that deal with network traffic modeling issues start with a traffic analysis before proposing an approach. We will utilize the obtained results and knowledge of the most common types of network traffic. Network traffic modeling studies often aggregate all connections together into a single flow, and it is known (see, e.g., [10]) that such aggregated traffic:

- has the property of self-similarity;
- exhibits long-range dependence (LRD) correlations.

Experimental investigations of many authors (see, e.g., [11], [12], [13]) show that fractional Gaussian noise (FGN) can be used for modeling aggregate traffic functions for various types of traffic (TCP, UDP, IP and others). The FGN model was first introduced by Mandelbrot and Van Ness in [14], and now it is widely used in network traffic modeling for its simplicity and mathematical attractiveness.

B. Data generation

In our research, network traffic data are generated using FGN. FGN is the increment process of fractional Brownian motion

$$x_H(t) = B_H(t+1) - B_H(t)$$

where $B_H(t)$ is a fractional Brownian motion at the moment t with Hurst exponent H. FGN is know to be good representation of the network traffic.

The mean and the variance of simulated traffic data are controlled by two time dependent parameters $\alpha(t)$ and $\beta(t)$

$$y_H(t) = \alpha(t) + \beta(t) \cdot x_H(t).$$

The amount of traffic in real networks is non-negative. All generated negative values are changed to zero. The simulated traffic data are represented as

$$z_H(t) = \begin{cases} y_H(t) & if \quad y_H(t) \geq 0 \\ 0 & if \quad y_H(t) < 0 \end{cases}$$

We assume the attack on network is represented as change of level (the mean) of traffic data. The attack can be described as a piecewise linear function of time. Attack data can be added as additional component on traffic data.

Nine classes of traffic data are defined, we call them reference database (see Table I). Examples of network traffic from each class are shown in Figure 1. Traffic data in examples are generated with Hurst exponent $H = 0.7$. The value $H = 0.7$ is in the same time default value for the Hurst exponent at the fgnSim function from the fArma package.

TABLE I. CLASSIFICATION OF NETWORK TRAFFIC

Process with	No attack	Rapid attack	Slow attack
Constant trend	1-A	1-B	1-C
Growing trend	2-A	2-B	2-C
Declining trend	3-A	3-B	3-C

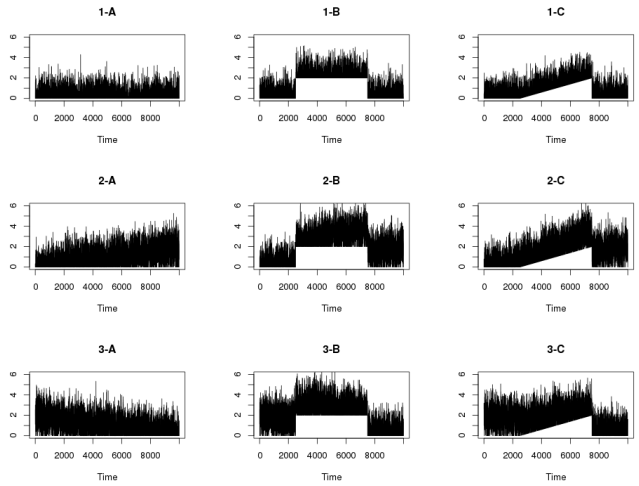

Fig. 1. Generated traffic data with/without attack patterns.

Data are generated using the R software [15]. FGN is generated using the R package *fArma* [16].

C. Problem formulation

Let $z(t)$ be a traffic function (time series) $z : T \to V$ where $T = \{1, ..., t_{max}\}$ is a discrete set of regular time moments and V is a certain range. More specifically, we distinguish three types, to which z may belong:

- $\boldsymbol{z^{ref}}$ - a reference database which consists of 9 classes of reference data (see Table I and Figure 1). More specifically, the reference database contains at least one representative for each class of data, i.e., $\boldsymbol{z^{ref}} = \{z_1^{ref}, ..., z_\ell^{ref}\}$ where $\ell \geq 9$. The latter are given by $z_i^{ref} : T^{ref} \to V$ where $T^{ref} = \{1, \ldots, t^{ref}\}$, $i = 1, \ldots, \ell$.
- z^{in} - input traffic time series given by $z^{in} : T^{in} \to V$ where $T^{in} = \{1, \ldots, \infty\}$
- z^P - extractions (patterns) of input traffic time series given by $z^P : T^P \to V$, $T^P = \{1, \ldots, t^{ref}\}$.

The goal *is to develop an algorithm which will provide an on-line processing of the input traffic time series z^{in}, with the indication of the attack, if found.*

Remark *The algorithm can be used in an application for monitoring actual server state, and in case of incoming attack, the output of the algorithm can be the action to store and save users data.*

III. Classification of data based on F-transform

The main idea of our approach is to apply the technique of F-transform to the function z^P and to the functions z_i^{ref}, $i = 1,\ldots,\ell$ representing the reference database. We obtain their reduced representations with smaller lengths. Then we compare these reduced representations and classify the actual data with respect to the defined nine classes (Table I).

The detailed description of our approach together with the algorithm is presented in Section IV. Below, we describe the formal concept of the F^s-transform, $s \geq 0$.

A. F^s-transform, $s \geq 0$, for functions of one variable

In this section, we assume that the reader is familiar with the main concept of the ordinary F-transform [17]. The F-transform with constant components can be extended to the F-transform of a higher degree - the F^s-transform, $s \geq 0$ - with s-degree polynomial components [18]. With respect to this extension, the original F-transform can be called the F^0-transform.

Generally, the F-transform depends on a chosen fuzzy partition. In the following, we recall the definition of generalized fuzzy partition [19]. Then we introduce a particular Hilbert space as a background for the further definition of the F^s-transform, $s \geq 0$.

1) Generalized Fuzzy Partition:

Definition 1: Let $[a,b]$ be a universe and $p_0, p_1, \ldots, p_n, p_{n+1} \in [a,b]$ be fixed nodes such that $a = p_0 \leq p_1 < \ldots < p_n \leq p_{n+1} = b$, $n \geq 2$. The fuzzy sets (*basic functions*) $A_1,\ldots,A_n : [a,b] \to [0,1]$ constitute a generalized fuzzy partition of $[a,b]$ if for every $k = 1,\ldots,n$, there exist $h_k^{'}, h_k^{''} \geq 0$ such that $h_k^{'} + h_k^{''} > 0$, $[p_k - h_k^{'}, p_k + h_k^{''}] \subseteq [a,b]$ and the following conditions are fulfilled:

1) (locality) – $A_k(t) > 0$ if $t \in (p_k - h_k^{'}, p_k + h_k^{''})$ and $A_k(t) = 0$ if $t \in [a,b] \setminus (p_k - h_k^{'}, p_k + h_k^{''})$;
2) (continuity) – A_k is continuous on $[p_k - h_k^{'}, p_k + h_k^{''}]$;
3) (covering) – for $t \in [a,b]$, $\sum_{k=1}^{n} A_k(t) > 0$.

By the *locality* and *continuity* conditions, it follows that

$$\int_a^b A_k(t)dt > 0.$$

If the nodes $p_0 = p_1, p_2, \ldots, p_{n-1}, p_n = p_{n+1}$ are h-equidistant, i.e., for all $k = 1,\ldots,n+1$, $p_k = p_{k-1} + h$, where $h = (b-a)/(n+1)$, $h^{'} > h/2$ and two additional properties are satisfied:

4. $h_1^{'} = h_n^{''} = 0$, $h_1^{''} = h_2^{'} = \cdots = h_{n-1}^{''} = h_n^{'} = h^{'}$, and $A_k(p_k - t) = A_k(p_k + t)$ for all $t \in [0, h^{'}]$, $k = 2,\ldots,n-1$;
5. $A_k(t) = A_{k-1}(t-h)$ and $A_{k+1}(t) = A_k(t-h)$ for all $t \in [p_k, p_{k+1}]$, $k = 2,\ldots,n-1$;

then the fuzzy partition is called $(h,h^{'})$-*uniform generalized fuzzy partition*. If moreover, $h = h'$, then we say that a fuzzy partition is h-uniform.

2) Space $L_2(A_k)$, subspace $L_2^s(A_k)$: Let interval $[a,b]$ be a universe and $\{A_k \mid k = 1,\ldots,n\}$ be an $(h,h^{'})$-uniform generalized fuzzy partition of $[a,b]$. Throughout this section, we fix one A_k from the chosen fuzzy partition.

Let $L_2(A_k)$ be a Hilbert space of square-integrable functions $f : [p_{k-1}, p_{k+1}] \to \mathbb{R}$ with the inner product $\langle f,g \rangle_k$ given by

$$\langle f,g \rangle_k = \int_{p_{k-1}}^{p_{k+1}} f(t)g(t)A_k(t)dt. \tag{1}$$

Remark 1: The functions $f, g \in L_2(A_k)$ are *orthogonal* in $L_2(A_k)$ if

$$\langle f,g \rangle_k = 0. \tag{2}$$

In the sequel, by $L_2([a,b])$, we denote a set of functions $f : [a,b] \to \mathbb{R}$ such that for all $k = 1,\ldots,n$, $f|_{[p_{k-1},p_{k+1}]} \in L_2(A_k)$, where $f|_{[p_{k-1},p_{k+1}]}$ is the restriction of f on $[p_{k-1}, p_{k+1}]$.

Moreover, let space $L_2^s(A_k)$, $s \geq 0$, be a closed linear subspace of $L_2(A_k)$ with the orthogonal basis given by polynomials

$$\{S_k^i(t)\}_{i=0,\ldots,s},$$

where i denotes a degree of polynomials and the orthogonality is considered in the sense of (2). For instance, $L_2^1(A_k)$ is a linear subspace of $L_2(A_k)$ with the orthogonal basis given by polynomials:

$$S_k^0(t) = 1,\ S_k^1(t) = t - p_k.$$

The following lemma characterizes the orthogonal projection of a function $f \in L_2([a,b])$ or the best approximation of f in the space $L_2^s(A_k)$.

Lemma 1: Let $f \in L_2([a,b])$ and let $L_2^s(A_k)$ be a closed linear subspace of $L_2(A_k)$, as specified above. Then, the orthogonal projection F_k^s of $f|_{[p_{k-1},p_{k+1}]}$ on $L_2^s(A_k)$, $s \geq 0$, is equal to

$$F_k^s(t) = \sum_{i=1}^{s} c_k^i S_k^i(t) \tag{3}$$

where

$$c_k^i = \frac{\langle f, S_k^i \rangle_k}{\langle S_k^i, S_k^i \rangle_k} = \frac{\int_{p_{k-1}}^{p_{k+1}} f(t)S_k^i(t)A_k(t)dt}{\int_{p_{k-1}}^{p_{k+1}} (S_k^i(t))^2 A_k(t)dt}. \tag{4}$$

The proof can be found in [18].

B. Direct F^s-transform, $s \geq 0$

Let $[a,b]$ be the universe and let $\{A_k \mid k = 1,\ldots,n\}$ be the $(h,h^{'})$-uniform generalized fuzzy partition of $[a,b]$. Moreover, let $f \in L_2([a,b])$ and let $L_2^s(A_k)$, $s \geq 0$, $k = 1,\ldots,n$, be a space with the basis given by

$$\{S_k^i(t)\}_{i=1,\ldots,s}.$$

In the following, we define the direct F^s-transform of the given function f.

Definition 2: Let $f \in L_2([a,b])$. Let F_k^s, $s \geq 0$ be the orthogonal projection of $f|_{[p_{k-1},p_{k+1}]}$ on $L_2^s(A_k)$, $k = 1,\ldots,n$ given by (3). We say that the n-dimensional vector $\mathbf{F_n^s}[f] = (F_1^s,\ldots,F_n^s)$ is the direct F^s-transform of f with respect to $\{A_k \mid k = 1,\ldots,n\}$, where F_k^s, $k = 1,\ldots,n$ is called the F^s-transform component.

1) Direct F^1-transform: In this section, we briefly introduce the (direct) F^1-transform of functions from $L_2([a,b])$. The technique of F^1-transform will be used in the application part of this contribution.

Let $L_2^1(A_k) \subseteq L_2(A_k)$ be a linear span of the set consisting of two orthogonal polynomials

$$S_k^0(t) = 1,\ S_k^1(t) = t - p_k, \tag{5}$$

where A_k from the chosen fuzzy partition is assumed to be symmetrical with respect to the node p_k, $k = 1,\ldots,n$.

Analogous to the general F^s-transform, $s \geq 0$, in the following, we introduce the F^1-transform with the components in the form of linear polynomials.

Definition 3: Let $f \in L_2([a,b])$ and let F_k^1 be the orthogonal projection of $f|_{[p_{k-1},p_{k+1}]}$ on subspace $L_2^1(A_k)$, $k = 1,\ldots,n$, with the basis given by (5).

The n-dimensional vector $\mathbf{F_n^1}[f] = (F_k^1)$, $k = 1,\ldots,n$ is the F^1-transform of f with respect to $\{A_k \mid k = 1,\ldots,n\}$, and F_k^1 is the corresponding F^1-transform component represented by

$$F_k^1(t) = c_k^0 + c_k^1(t - p_k), \tag{6}$$

where the coefficients c_k^0, c_k^1 are given by (4).

The following theorem presents approximations of the function f and its first derivative f' using the coefficients of the F^1-transform and estimates the qualities of the approximations.

Theorem 1: Let $f \in L_2([a,b])$ and $\{A_k \mid k = 1,\ldots,n\}$, $n \geq 2$, be an (h,h')-uniform generalized fuzzy partition of $[a,b]$. Let $\mathbf{F_n^1}[f] = (F_1^1,\ldots,F_n^1)$ where $F_k^1(t) = c_k^0 + c_k^1(t - p_k)$, $k = 1,\ldots,n$, be the F^1-transform of f with respect to the given partition.

- Let functions f, A_k, $k = 1,\ldots,n$ be twice continuously differentiable on $[a,b]$, then for every k:

$$c_k^0 = f(p_k) + O(h'^2).$$

- Let functions f, A_k, $k = 1,\ldots,n$, be four times continuously differentiable on $[a,b]$, then for every k:

$$c_k^1 = f'(p_k) + O(h'^2).$$

The proof is analogous to the case of h-uniform fuzzy partition introduced in [18].

2) Discrete case of F^1-transform: Let us consider the discrete case when the original function is known only at some discrete points. The data used in experiments in this paper are represented by discrete elements.

The discrete (direct) F^1-transform is defined as follows.

Definition 4: Let a function $f : [a,b] \to \mathbb{R}$ be defined at discrete points $P = \{(t_i) \mid i = 1,\ldots,N\}$. Let $\{A_k \mid k = 1,\ldots,n\}$ be a fuzzy partition of $[a,b]$ where $p_0, p_1, \ldots, p_{n+1} \in [a,b]$ are fixed nodes. Suppose that the set P is sufficiently dense with respect to the chosen partition, i.e.,

$$\forall k\ \exists i;\ A_k(t_i) > 0.$$

We say that the n-dimensional vector $\mathbf{F_n^1}[f] = (F_1^1,\ldots,F_n^1)$ is the discrete F^1-transform of f with respect to the chosen partition if for all $k = 1,\ldots,n$ and $t_i \in P$

$$F_k^1(t_i) = c_k^0 + c_k^1(t_i - p_k), \tag{7}$$

where the coefficients c_k^0, c_k^1 are given as follows

$$c_k^0 = \frac{\sum_{i=1}^N f(t_i)A_k(t_i)}{\sum_{i=1}^N A_k(t_i)},$$

$$c_k^1 = \frac{\sum_{i=1}^N f(t_i)(t_i - p_k)A_k(t_i)}{\sum_{i=1}^N (t_i - p_k)^2 A_k(t_i)}.$$

The alternative way for definition of the discrete higher degree F-transform can be found in [20].

IV. Proposed algorithm of classification

As we mentioned above, we work with three types of data functions: $\boldsymbol{z^{ref}} = \{z_1^{ref},\ldots,z_\ell^{ref}\}$ - reference database, z^{in} - input traffic time series, z^P - an extraction of input traffic time series which is actually processed. We apply the direct F^1-transform to these functions and compare their components. The goal is to classify the actual z^P into one of the nine classes (see Table I) in each predefined period of time.

We use the F^1-transform because it is sensitive to local changes in the analyzed string (opposite to the F^0 transform that produces only average values and by this, does not react on the linear structure of a string).

The idea of our approach is as follows. At first, we apply the F^1-transform to the reference functions z_i^{ref}, $i = 1,\ldots,\ell$ and obtain $\mathbf{F_n^1}[z_i^{ref}] = (F_1^{1,i},\ldots,F_n^{1,i})$, $i = 1,\ldots,\ell$. Then we extract the "first" $z_{t_0}^P$ from the function z^{in} starting from $z^{in}(t_0)$ and finishing at $z^{in}(t_0 + t^{ref} - 1)$. We apply the F^1-transform to $z_{t_0}^P$ and obtain $\mathbf{F_n^1}[z_{t_0}^P] = (F_1^{1,t_0},\ldots,F_n^{1,t_0})$. Then we compare $\mathbf{F_n^1}[z_i^{ref}]$ and $\mathbf{F_n^1}[z_{t_0}^P]$ in order to classify $z_{t_0}^P$ into one of the nine classes. The comparison is done by computation a measure of closeness between coefficients $c^1_{k,z_{t_0}^P}$ and $c^1_{k,z_i^{ref}}$, $k = 1,\ldots,n$, for all reference classes $i = 1,\ldots,\ell$. The following measure of closeness is used:

$$Cl(c^1_{k,z_{t_0}^P}, c^1_{k,z_i^{ref}}) = \sum_{k=1}^{n} |c^1_{k,z_{t_0}^P} - c^1_{k,z_i^{ref}}|. \tag{8}$$

The function $z_{t_0}^P$ is classified based on the minimal closeness $Cl(c^1_{k,z_{t_0}^P}, c^1_{k,z_i^{ref}})$. Moreover, if this minimal closeness is higher than a predefined threshold θ (i.e., $z_{t_0}^p$ is not enough similar to any class), the previous classification is taken as a result.

In the next step (iteration), $z_{t_0}^P$ is modified in the following way. The first h values of $z_{t_0}^P$ are removed and the new unprocessed h values from z^{in} are added to the end of $z_{t_0}^P$. Therefore, we obtain $z_{t_{0+h}}^P = (z^{in}(t_0 + h),\ldots,z^{in}(t_0 + t^{ref} + h))$. We compute one more F^1-transform component F_n^{1,t_0+h} and compose the new vector of components as follows: $(F_2^{1,t_0},\ldots,F_n^{1,t_0},F_n^{1,t_0+h})$. This vector is again compared with the components of the reference database and the process continues iteratively.

Let us stress, that the parameter h represents the distance between nodes from the chosen generalized fuzzy partition. It influences the number of basic functions in the fuzzy partition and therefore, the number of F^1-transform components. In our case, all basic functions from the fuzzy partition are triangular fuzzy sets with the parameters $(h,h') = (h,2h)$.

The Figure 2 illustrates this process. The red parts denote unprocessed data, i.e, the data for which the F^1-transform components have to be computed. The green parts denote the already processed data, i.e., the F^1-transform components stored from the previous iteration.

The algorithm consists of four inputs ($\boldsymbol{z^{ref}}$, z^{in}, h, θ), four preprocessing steps (marked with prefix *P*) which are executed only once, five standard steps (marked with prefix *S*), and one output vector O. Let us briefly recall the inputs:

- $\boldsymbol{z^{ref}}$ - the reference database consisting of reference time series (at least one for each class),
- z^{in} - infinite time series monitoring network traffic,
- h - parameter of the chosen fuzzy partition,
- θ - the user-defined threshold delimiting the maximal value of closeness.

The output is the vector O of classifications, i.e. at each step *S 2* the particular classification of $z^P_{t_0}$ is stored to a next position in O and printed immediately to the user.

The particular steps of the algorithm are as follows:

Inputs: $\boldsymbol{z^{ref}}$, z^{in}, h, θ.

P 1: Compute $\mathbf{F^1_n}[z_i^{ref}] = (F_1^{1,i},\ldots,F_n^{1,i})$, $i = 1,\ldots,\ell$ w.r.t. the $(h,2h)$-uniform generalized fuzzy partition.

P 2: $t_0 := 1$

P 3: Create an extraction $z^P_{t_0}$ from z^{in} as $z^P_{t_0} = (z^{in}(t_0),\ldots,z^{in}(t_0+t^{ref}-1))$.

P 4: Compute $\mathbf{F^1_n}[z^P_{t_0}] = (F_1^{1,t_0},\ldots,F_n^{1,t_0})$ w.r.t. the same $(h,2h)$-uniform generalized fuzzy partition.

S 1: Compute the closeness $Cl(c^1_{k,z_i^{ref}}, c^1_{k,z^P_{t_0}})$, $k = 1,\ldots,n$, $i = 1,\ldots,\ell$.

S 2: Classify $z^P_{t_0}$ according to the value of closeness Cl and threshold θ. Add the classification into the output vector O.

S 3: Create $z^P_{t_0+h}$ as $z^P_{t_0+h} = (z^{in}(t_0+h),\ldots,z^{in}(t_0+t^{ref}+h))$.

S 4: Compute F_n^{1,t_0+h} and compose the vector of F^1-transform components as follows: $(F_2^{1,t_0},\ldots,F_n^{1,t_0},F_n^{1,t_0+h})$.

S 5: $t_0 := t_0+h$; go to *S 1*.

Outputs: O

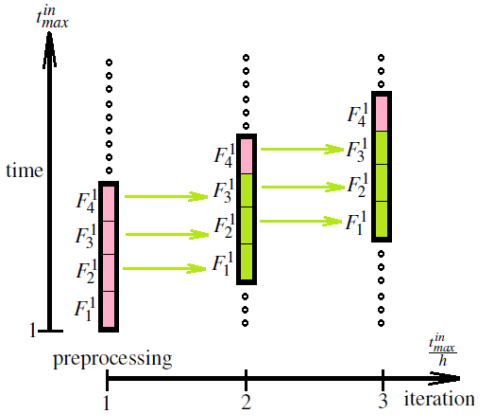

Fig. 2. Illustration of the iteration process for modifying z^p.

TABLE II. INFLUENCE OF CHOSEN h

h	compute time [ms]	success rate [%]
100	0.052	100.0
1000	0.015	100.0
5000	0.060	24.1

V. EXPERIMENTS AND RESULTS

For our tests, we create the database $\boldsymbol{z^{ref}}$ with the help of FGN model and obtain 27 reference time series z_i^{ref}. Among these reference time series we have 3 representatives of each class. Then using the same model, we create our input z^{in} in such a way that it contains 900 sequentially-ordered time series (100 for each class). The length of each time series in $\boldsymbol{z^{ref}}$ is equal to 10000, i.e., $t^{ref} = 10000$.

The Table II shows the influence of the chosen parameter h. The lower h, the less values are used to compute the new F^1-transform component. On the other hand, the lower h, the more components for one z^p we obtain, and therefore, the more computations of closeness have to be computed.

From the results in the Table II, we can observe that for $h = 1000$ the best computation time was achieved (0.015ms). The computation time of the preprocessing steps (*P1 - P4*) is not taken into account because these steps are executed only once. The algorithm made 17961 classifications, all of them successful.

Let us emphasize that for $h = 5000$, the low success rate was obtained. In the latter case, we do not have the sufficient number of F^1-transform components for one z^p, and therefore, the algorithm does not work properly.

VI. CONCLUSION

The goal of this work was to classify the input traffic time series into one of the classes which simulated the real network traffic situations. We defined nine classes - three for the case of no attack, three for the rapid attack and three for the slow attack. We designed the algorithm for classification based on the F-transform theory. We recalled the theoretical background of the F-transform in order to use this technique for data

aggregation. The proposed algorithm achieved computation time 0.015ms per one classification with overall success rate 100 %. The next research can be focused on a comparison with other existing methods. Moreover, we intend to modify the implemented application into the version which can be moved on to real network servers and by this, we want to analyze the behavior of the application for the real traffic classification.

ACKNOWLEDGMENT

This work has been supported by the European Social Fund within the project 2013/0024/1DP/1.1.1.2.0/13/APIA/VIAA/045. Additional support was provided by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070). This work was also co-supported by SGS project of the University of Ostrava.

REFERENCES

[1] M. Mellia, A. Pescapč, and S. L., "Traffic classification and its applications to modern networks," *Computer Networks*, vol. 53, pp. 759–760, 2009.

[2] M. Li, "Change trend of averaged hurst parameter of traffic under DDoS flood attacks," *Computers & Security*, vol. 25, pp. 213–220, 2006.

[3] Computer Emergency Response Team (CERT-EU), *DDoS overview and incident response guide*, 2014. [Online]. Available: http://cert.europa.eu/static/WhitePapers/

[4] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, pp. 643–666, 2004.

[5] B. Xiao, W. Chen, and Y. He, "A novel approach to detecting ddos attacks at an early stage," *The Journal of Supercomputing*, vol. 36, pp. 235–248, 2006.

[6] K. Lee, J. Kim, K. Kwon, J. Han, and S. Kim, "DDoS attack detection method using cluster analysis," *Expert Systems with Applications*, vol. 34, pp. 1659–1665, 2008.

[7] Z. Xia, S. Lu, J. Li, and J. Tang, "Enhancing DDoS flood attack detection via intelligent fuzzy logic," *Informatica*, vol. 34, pp. 497–507, 2010.

[8] V. Novák, M. Štěpnička, A. Dvořák, I. Perfilieva, V. Pavliska, and L. Vavříčková, "Analysis of seasonal time series using fuzzy approach," *International Journal of General Systems*, vol. 39, pp. 305–328, 2010.

[9] I. Perfilieva, V. Novák, and V. Pavliska, "The use of higher-order F-transform in time series analysis," in *World Congress IFSA 2011 and AFSS 2011*, Surabaya, Indonesia, 2011, pp. 2211–2216.

[10] W. Willinger, M. Taqqu, W. Leland, and D. Wilson, "Self-similarity in high-speed packet traffic: analysis and modeling of Ethernet traffic measurements," *Statistical Science*, vol. 10, pp. 67–85, 1995.

[11] I. Norros, "On the use of fractional Brownian motion in the theory of connectionless networks," *Selected Areas in Communications*, vol. 13, pp. 953–962, 1995.

[12] V. Paxson, "Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic," *Computer Communication Review*, vol. 27, pp. 5–18, 1997.

[13] M. Li, C.-H. Chi, and D. Long, "Fractional Gaussian noise: a tool of characterizing traffic for detection purpose," in *Lecture Notes in Computer Science*, 2004, vol. 3309, pp. 94–103.

[14] B. Mandelbro and J. Van Ness, "Fractional Brownian motions, fractional noises and applications," *SIAM Review*, vol. 10, pp. 422–437, 1968.

[15] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: http://www.R-project.org/

[16] D. Wuertz, *fArma: ARMA Time Series Modelling*, 2013. [Online]. Available: http://CRAN.R-project.org/package=fArma

[17] I. Perfilieva, "Fuzzy transforms: Theory and applications," *Fuzzy Sets and Systems*, vol. 157, pp. 993–1004, 2006.

[18] I. Perfilieva, M. Daňková, and B. Bede, "Towards a higher degree F-transform," *Fuzzy Sets and Systems*, vol. 180, pp. 3–19, 2011.

[19] I. Perfilieva, "F-transform," in *Handbook of Computational Intelligente*, J. Kacprzyk and W. Pedrycz, Eds. Berlin, Heidelberg: Springer, 2014, p. in press.

[20] M. Holčapek and T. Tichy, "Discrete fuzzy transform of higher degree," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, July 2014, pp. 604–611.

P. Hurtik and P. Hodakova. FTIP: Tool for image plagiarism detection. *Proceedings of 2015 Seventh International Conference of Soft Computing and Pattern Recognition (SoCPaR 2015)*, 42–47, IEEE, 2015.

FTIP: Tool for Image Plagiarism Detection

Petr Hurtik, Petra Hodakova
University of Ostrava, Centre of Excellence IT4Innovations, Institute for Research and Applications of Fuzzy Modeling,
30. dubna 22, 701 03 Ostrava 1, Czech Republic,
Petr.Hurtik@osu.cz, Petra.Hodakova@osu.cz

***Abstract*—The goal of this paper is to introduce a task of image plagiarism detection. More specifically, we propose a method of searching for a plagiarized image in a database. The main requirements for searching in the database are computational speed and success rate. The proposed method is based on the technique of F-transform, particularly F^s-transform, $s \geq 0$. This technique significantly reduces the domain dimension and therefore, it speeds up the whole process. We present several experiments and measurements which prove the speed and accuracy of our method. We also introduce examples to demonstrate an ability of using this method in many applications.**

***Keywords*-Image plagiarism; Pattern searching; F-transform; F^s-transform; Image retrieval**

I. Introduction

People generally plagiarize documents because it is faster, easier and results are more fluent. In [1] the plagiarism of a text is defined by copying the text directly from a source without any citation. In [2] Bouville discussed the question "What is a text plagiarism?" more extensively. He specified several plagiarism areas: "Pseudo-Plagiarism; Vague definition; Self-plagiarism; Plagiarism of secondary sources". For that, Bouville stated a Hexham's definition: "Academic plagiarism occurs when a writer repeatedly uses more than four words from a printed source without the use of quotation marks and a precise reference to the original source."

If the plagiarism is considered in the sense of the Hexham's definition, the plagiarism detection is indeed a straightforward task. Source documents are compared with reference documents in a database and all the matches are searched [3]. In several past decades, a lot of algorithms or systems for text plagiarism were developed [4], [5]. Moreover, in a real world, the number of words is finite and authors incline to use commonly used phrases such as "on the other hand", etc. Therefore, the plagiarism detection systems have to deal with so called non-detections or false positives [6].

From the background research, it is obvious that the plagiarism problem is well-defined, well-studied, or even more or less solved. On the other hand, almost everything is aimed to the *text plagiarism*. The same importance should be given to a *media plagiarism*, e.g. music plagiarism, image/video plagiarism, etc. But for example in the case of music, the task of plagiarism is not straightforward. The decision about plagiarism cannot be given a-priori and it has to be created by human consensus, see for example [7].

In this paper, we focus on the image plagiarism detection. The motivation came from a plagiarism system used for checking of thesisses plagiarism. By this system only texts are processed but we think that image plagiarism should be inspected as well. To deal with the image plagiarism detection, we propose the following definition: *"Image plagiarism is a process when an image or its part is copied and used without any reference to its source."* Another problem is when the image plagiarism is considered in the sense of using images which were forbidden to copy by their authors. In this paper, we do not consider the latter case.

Although the image plagiarism is not as explored as the text plagiarism, the idea of searching for a source image in a huge database of images is fairly common. The most naive method is based on a comparison of all pixels of the searched image with all pixels of all the images in the database. In the case when we search only for a part of the image in the database, even all positions of the searched part in all the images in the database have to be inspected. Therefore, the naive method is very time expensive.

There are other methods/systems solving this task more efficiently, for example *Image Querying Database* [8] or *Image Retrieval* [9]. Unfortunately, the global aim of the image retrieval task is to search for the image in its broader sense, i.e., by taking into account colors, shapes, textures, segmented parts, etc. The result is an image which is somehow *visually similar* but not exactly *the same*, see for example [10]. One of easy-to-use application of image retrieval is called Google Images[1] where the searched image is uploaded and then, it is automatically searched in internet. Many possibilities of Google Images system are described in Ph.D. thesis [11].

The goal of this paper is to present a general method of searching for images or their cropped parts in huge databases. This method can be used as a core of an image plagiarism system. The proposed searching method is based on the technique of F-transform, specifically, the F-transform of a higher degree (F^s-transform, $s \geq 0$). Generally, the F-transform performs a transformation of an original universe of functions into a universe of their "skeleton models" and therefore, it gives us a simplified representation of the original function. This F-transform representation approximates the original function and its dimension is significantly reduced with respect to the dimension of the original function. Therefore, *it is easier and faster to process the F-transform representation instead of the original function*.

The F-transform was originally introduced in [12] and

[1] $http://images.google.com$

generalized to the F^s-transform, $s \geq 0$ in [13], [14]. This technique was successfully used in one dimensional searching task, specifically, in string searching mechanism [15]. Generally, the principle of our searching mechanism can be used in more sophisticated applications where pattern searching is needed.

The structure of the paper is as follows: Section II presents the theoretical background of the algorithm - the F^s-transform, $s \geq 0$. The algorithm itself is designed in Section III. Tested data and their structure are described in Section IV together with experiments and results. Interesting remarks, open questions and future work are discussed in Section V. Finally, conclusions are summarized in Section VI.

II. Preliminaries: F^s-transform, $s \geq 0$

Before we will present the mechanism of searching for images based on the F-transform, let us introduce the technique of F-transform of a higher degree for functions of two variables.

We assume that the reader is familiar with the main concept of the ordinary F-transform [12]. In this section, we extend the F-transform with constant components to the F^s-transform, $s \geq 0$, with s-degree polynomial components. First, we recall the basic tenets of the fuzzy partition [12] and introduce a particular Hilbert space [14].

A. Fuzzy partition

Let us first introduce the notion of fuzzy partition for interval $[a, b]$ and then extend it to $[a, b] \times [c, d]$.

Definition 1: Let $x_1, \ldots, x_n \in [a, b]$ be fixed nodes such that $a = x_1 < \ldots < x_n = b$ and $n \geq 3$. Fuzzy sets $A_1, \ldots, A_n : [a, b] \to [0, 1]$ identified with their membership functions defined on $[a, b]$ establish a fuzzy partition of $[a, b]$ if they fulfill the following conditions for $k = 1, \ldots, n$:

1) $A_k(x_k) = 1$;
2) $A_k(x) = 0$ if $x \in [a, b] \setminus (x_{k-1}, x_{k+1})$ and we set $x_0 = a,\ x_{n+1} = b$;
3) $A_k(x)$ is continuous on $[x_{k-1}, x_{k+1}]$;
4) $A_k(x)$ for $k = 2, \ldots, n$ strictly increases on $[x_{k-1}, x_k]$ and for $k = 1, \ldots, n-1$ strictly decreases on $[x_k, x_{k+1}]$.

The membership functions $A_1, \ldots, A_n$ are called *basic functions*. A point $x \in [a, b]$ is *covered* by the basic function A_k if $A_k(x) > 0$.

If the nodes $x_1, \ldots, x_n$ are *h-equidistant*, i.e., for all $k = 2, \ldots, n,\ x_k = x_{k-1} + h$, where

$$h = (b - a)/(n - 1), \tag{1}$$

and hold two additional properties for $k = 2, \ldots, n - 1$:

5) $A_k(x_k - x) = A_k(x_k + x)$ for all $x \in [0, h]$;
6) $A_k(x) = A_{k-1}(x - h)$ and $A_{k+1}(x) = A_k(x - h)$ for all $x \in [x_k, x_{k+1}]$;

then the fuzzy partition $A_1, \ldots, A_n$ is *h-uniform*.

Moreover, the fuzzy partition is called *Ruspini partition* if the *Ruspini condition* $\sum_{k=1}^{n} A_k(x) = 1$ holds for all $x \in [a, b]$.

The concept of fuzzy partition can be easily extended to the universe $[a, b] \times [c, d]$. We assume that $[a, b]$ is partitioned by $A_1, \ldots, A_n$ and $[c, d]$ is partitioned by $B_1, \ldots, B_m$. Then, the Cartesian product $[a, b] \times [c, d]$ is partitioned by the Cartesian product of corresponding partitions where a basic function $A_k \times B_l$ is equal to the product $A_k \cdot B_l$, $k = 1, \ldots, n$, $l = 1, \ldots, m$.

Remark 1: Let us remark that the fuzzy partition $\{A_k \times B_l \mid k = 1, \ldots, n, l = 1, \ldots, m\}$ of $[a, b] \times [c, d]$ is called *h-uniform* if $A_1, \ldots, A_n$ establish the h-uniform fuzzy partition of $[a, b]$ and $B_1, \ldots, B_m$ establish the h-uniform fuzzy partition of $[c, d]$.

B. Spaces $L_2(A_k \times B_l)$ and $L_2^s(A_k \times B_l)$

Let us assume a rectangle $[a, b] \times [c, d]$ and fix a fuzzy partition $\{A_k \times B_l \mid k = 1, \ldots, n, l = 1, \ldots, m\}$, $n, m \geq 2$ of this rectangle. Let k, l be fixed integers from $\{1, \ldots, n\}, \{1, \ldots, m\}$ respectively, and let $L_2(A_k \times B_l)$ be a set of square-integrable functions $f : [x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}] \to \mathbb{R},\ k = 1, \ldots, n;\ l = 1, \ldots, m$, on their domain.

Let $\langle f, g \rangle_{kl}$ be an inner product of functions f and g in $L_2(A_k \times B_l)$ defined as follows

$$\langle f, g \rangle_{kl} = \int_{x_{k-1}}^{x_{k+1}} \int_{y_{l-1}}^{y_{l+1}} f(x, y) g(x, y) A_k(x) B_l(y) dx dy$$

and let

$$\|f\|_{kl} = \sqrt{\langle f, f \rangle_{kl}}$$

be a corresponding norm. Then $L_2(A_k \times B_l)$ is a Hilbert space.

In the sequel, we denote by $L_2([a, b] \times [c, d])$ a set of functions $f : [a, b] \times [c, d] \to \mathbb{R}$ such that for all $k = 1, \ldots, n,\ l = 1, \ldots, m,\ f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]} \in L_2(A_k \times B_l)$, where $f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]}$ is the restriction of f on $[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]$.

Moreover, let $L_2^s(A_k \times B_l)$, $s \geq 0$ be a closed linear subspace of $L_2(A_k \times B_l)$ with the basis given by orthogonal polynomials

$$\{S_{kl}^{ij}(x, y)\}_{i+j \leq s},$$

where s denotes the maximal degree of polynomials and the orthogonality is considered in the usual sense with respect to the inner product.

The following Lemma characterizes the orthogonal projection of a function $f \in L_2([a, b] \times [c, d])$ or the best approximation of f in the space $L_2^s(A_k \times B_l)$.

Lemma 1: Let $f \in L_2([a, b] \times [c, d])$ and let $L_2^s(A_k \times B_l)$ be a closed linear subspace of $L_2(A_k \times B_l)$, as specified above. Then, the orthogonal projection F_{kl}^s of $f|_{[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]}$ on $L_2^s(A_k \times B_l)$, $s \geq 0$, is equal to

$$F_{kl}^s = \sum_{0 \leq i+j \leq s} c_{kl}^{ij} S_{kl}^{ij}, \tag{2}$$

where

$$c_{kl}^{ij} = \frac{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} f(x, y) S_{kl}^{ij}(x, y) A_k(x) B_l(y) dx dy}{\int_{y_{l-1}}^{y_{l+1}} \int_{x_{k-1}}^{x_{k+1}} (S_{kl}^{ij}(x, y))^2 A_k(x) B_l(y) dx dy}. \tag{3}$$

The proof is constructed analogously to the case of functions of one variable [13].

C. Direct F^s-transform, $s \geq 0$

The direct F^s-transform of the given function f is defined as follows [14].

Definition 2: Let $f \in L_2([a,b] \times [c,d])$ and let $\{A_k \times B_l \mid k = 1,\dots,n, l = 1,\dots,m\}$ be the fuzzy partition of $[a,b] \times [c,d]$. Moreover, let F^s_{kl}, $s \geq 0$ be the orthogonal projection of $f|_{[x_{k-1},x_{k+1}]\times[y_{l-1},y_{l+1}]}$ on $L^s_2(A_k \times B_l)$, $k = 1,\dots,n$, $l = 1,\dots,m$. We say that $(n \times m)$ matrix $\mathbf{F^s_{nm}}[f] = (F^s_{kl})_{k=1,\dots,n;l=1,\dots,m}$ is the direct F^s-transform of f with respect to $\{A_k \times B_l \mid k = 1,\dots,n, l = 1,\dots,m\}$, where F^s_{kl} is called the F^s-transform component.

By *Lemma 1*, the F^s-transform components have the representation given by (2).

The following Lemma estimates the quality of the local approximations of the original function f by the F^s-transform components.

Lemma 2: Let $n, m \geq 2$ and let functions f, A_k, B_l, $k = 1,\dots,n$, $l = 1,\dots,m$, be $(s+1)$-times continuously differentiable on $[a,b] \times [c,d]$. Moreover, let $\{A_k \times B_l \mid k = 1,\dots,n, l = 1,\dots,m\}$ establish an h-uniform fuzzy partition of $[a,b]\times[c,d]$ and let $\mathbf{F^s_{nm}}[f] = (F^s_{kl})_{k=1,\dots,n;l=1,\dots,m}$, $s \geq 1$, be the direct F^s-transform of f with respect to the given partition where

$$F^s_{kl} = \sum_{0 \leq i+j \leq s} c^{ij}_{kl} S^{ij}_{kl}.$$

Then for every $(x,y) \in [a,b]\times[c,d]$ and for every $s \geq 1$ there exist k, l such that the following holds true

$$|F^s_{kl}(x,y) - f(x,y)| \leq C_s \cdot (h)^{s+1},$$

where $C_s \to 0$ for $s \to \infty$.

The proof can be found in [14].

1) Discrete (direct) F^1-transform: In this contribution, we work with images represented by discrete array of pixels. Therefore, in this section, we briefly present the F^1-transform for functions of two variables where the values of the original function are known only at discrete points. The F^1-transform is later used in the application part.

The discrete (direct) F^1-transform is defined as follows.

Definition 3: Let a function $f : [a,b] \times [c,d] \to \mathbb{R}$ be defined at discrete points $P = \{(p_i, q_j) \in [a,b] \times [c,d] \mid i = 1,\dots,N, j = 1,\dots,M\}$. Let $\{A_k \times B_l \mid k = 1,\dots,n,\ l = 1,\dots,m\}$ be a fuzzy partition of of $[a,b] \times [c,d]$. Suppose that the set P is sufficiently dense with respect to the chosen partition, i.e.,

$$\forall k,l\ \exists i,j;\ A_k(p_i)B_l(q_j) > 0.$$

We say that $(n\times m)$ matrix $\mathbf{F^1_{nm}}[f] = (F^1_{kl})_{k=1,\dots,n;l=1,\dots,m}$ is the discrete direct F^1-transform of f with respect to $\{A_k\times B_l \mid k = 1,\dots,n, l = 1,\dots,m\}$, where

$$F^1_{kl}(p_i,q_j) = c^{00}_{kl} + c^{01}_{kl}(p_i - x_k) + c^{10}_{kl}(q_j - y_l), \qquad (4)$$

and the coefficients c^{00}_{kl}, c^{01}_{kl} and c^{10}_{kl} are given as follows

$$c^{00}_{kl} = \frac{\sum_{i=1}^N \sum_{j=1}^M f(p_i,q_j)A_k(p_i)B_l(q_j)}{\sum_{i=1}^N \sum_{j=1}^M A_k(p_i)B_l(q_j)},$$

$$c^{01}_{kl} = \frac{\sum_{i=1}^N \sum_{j=1}^M f(p_i,q_j)(p_i - x_k)A_k(p_i)B_l(q_j)}{\sum_{i=1}^N \sum_{j=1}^M (p_i - x_k)^2 A_k(p_i)B_l(q_j)}.$$

$$c^{10}_{kl} = \frac{\sum_{i=1}^N \sum_{j=1}^M f(p_i,q_j)(q_j - y_l)A_k(p_i)B_l(q_j)}{\sum_{i=1}^N \sum_{j=1}^M (q_j - y_l)^2 A_k(p_i)B_l(q_j)}.$$

More details can be found in [14]. The alternative way for definition of the discrete higher degree F-transform can be found in [16].

III. Image plagiarism detection based on F^1-transform

In this section, we propose an algorithm $FTIP$ for the image plagiarism detection which is based on the theory given above. The idea is to take one source image (or its cropped part) and search for this image in the given database of images. The goal is to find and mark the same corresponding image in the database.

The idea of the proposed algorithm came from a string searching algorithm [15] where the suitability of the F^1-transform to one-dimensional pattern searching was presented. We follow the principles from [15] and extend it to two-dimensional case. Searching for the two-dimensional source image in the database of images can be also seen as a task of pattern searching.

The main idea is to represent the source image and the images in the database by discrete functions and apply the F^1-transform to these functions in order to obtain their simplified representations in the form of matrices of F^1-transform components. Searching for the source image in the database is then basen on a comparison of those matrices of the F^1-transform components by computing distances. The algorithm $FTIP$ is described in more details below.

A. $FTIP$: algorithm design

Let we have a large database of images $\mathbf{I_D} = [f_1, f_2, ..., f_d]$. Assume that the image is described as a discrete real function $f : [1,N]\times[1,M] \to \mathbb{R}$ defined on the $N \times M$ array of pixels.

Particular steps of the algorithm $FTIP$ are following:

In: Database of images $\mathbf{I_D}$, source image (pattern) f_p, partition parameter h, threshold θ.

P1: For each $f_i \in \mathbf{I_D}$, $i = 1,\dots,d$ compute $\mathbf{F^1_{nm}}[f_i] = (F^{1,i}_{11},\dots,F^{1,i}_{nm})$ w.r.t. the h-uniform fuzzy partition.

S1: Compute $\mathbf{F^1_{nm}}[f_p] = (F^{1,p}_{11},\dots,F^{1,p}_{nm})$ w.r.t. the same h-uniform fuzzy partition.

S2: For each $f_i \in \mathbf{I_D}$, $i = 1,\dots,d$ compute the distance $Dist_i(\mathbf{F^1_{nm}}[f_i], \mathbf{F^1_{nm}}[f_p])$ between the F^1-transform components of f_i and f_p.

Out: (a) The image $f_j \in \mathbf{I_D}$ such that

$$Dist_j = \min_{i=\{1,\dots,d\}} \{Dist_i\} \text{ and } Dist_j < \theta.$$

(b) The source image is not found in the database if the minimal distance $Dist_j \geq \theta$.

There are many approaches how to compute the distances. Based on an evaluation in [17], in our approach we use

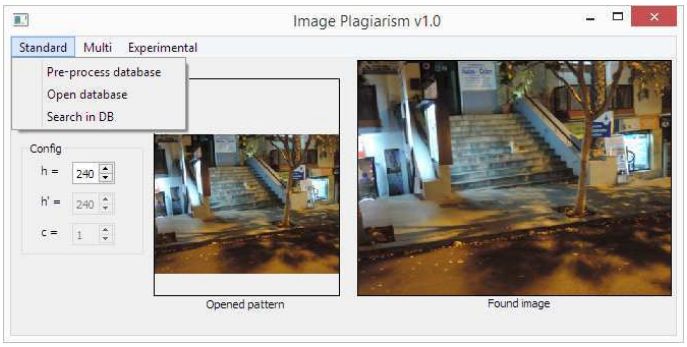

Fig. 1. The graphic interface of the implemented application.

Minkowsky distance with a parameter $p = 1$ which is known as Manhattan distance or City-blocks distance.

Remark 2: From the algorithm above it is obvious that the process actually searches for *similar* images. The similarity between the pattern and the database images is represented by the distance function. The usage of this idea is really wide, practically in all applications which use pattern searching, texture searching, etc. In this paper, we focus only on searching for *the same* images.

IV. Experiments

In this section, we present experiments and results of our searching algorithm. The most important aspects are the processing time and the success rate.

The algorithm was implemented for OS Windows and the user interface is shown on Figure 1. The implemented application can be freely downloaded[2]. Let us remark that all the experiments were tested on a non-parallelized application and a standard-power notebook.

At first we used images from the well-known database BSDS500[3]. This database contains images with a resolution only 481×321px (or 321×481px). To be more general, we created our own database composed from hand-made real-life images taken by a mobile phone or a compact camera. Our database contains images with different resolutions: from small (the smallest is 127×175px) to large (the largest is 4000×3000px).

A. Tested data, Preprocessing

For experiments presented in this paper, we created three different databases of images. Their specifications are as follows:

- Test DB1: 600 images with different but mainly low resolutions. Physical size on a hard disk (png and jpg graphic compression format): 127MB. Points in total: $4.752 \cdot 10^8$.
- Test DB2: 800 images: 600 from DB1 and 200 new higher-resolution images. Physical size on a hard disk: 485MB, Points in total: $1.46 \cdot 10^9$.

[2] *graphicwg.irafm.osu.cz/storage/plagiarism_image_v1_0.rar*
[3] *www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/*

TABLE I
Reduced sizes of the preprocessed databases with respect to the chosen h

h	DB1 [MB]	DB2 [MB]	DB3 [MB]
20	10.767	33.587	254.845
40	2.599	8.220	63.717
60	1.124	3.628	27.887
80	0.604	1.990	15.454
100	0.383	1.233	9.776

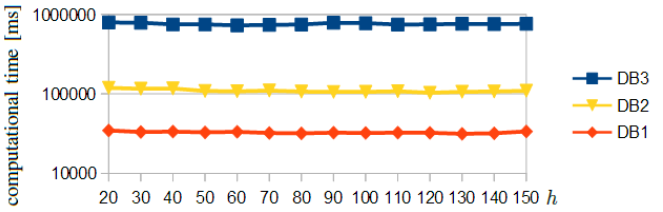

Fig. 2. Influence of the chosen h to the comp. time [ms] of step *P1*.

- Test DB3: 1292 images with high-resolutions. Physical size on a hard disk (jpg graphic compression format): 4076MB. Points in total: $1.11 \cdot 10^{10}$.

Each database was preprocessed (step *P1* in the algorithm above) for several different settings of the F^1-transform. It means, all the images in the databases were processed by the F^1-transform with several values of the input parameter h and then represented by the F^1-transform components. Let us remark, *the bigger value of h, the less number of the F^1-transform components* and vice versa.

The preprocessing step is very important because it rapidly reduces the sizes of the databases. Table I shows the reduction of sizes of the databases based on the chosen parameter h. For example, we can observe that by using the F^1-transform with $h = 100$, the database DB3 can be stored with a size which is more than $400\times$ smaller than the original size. Let us remark that the preprocessed database is stored as a .txt file without any compression. By using a rar compression method, we obtain even smaller size – using the same example, we obtain $1100\times$ smaller size of the database than the original. This suitability of using the F-transform for image reduction and image compression was already proven in previous works, see for example [18], [19].

Another interesting fact from this preprocessing step is that *the value of the input parameter h does not influence the preprocessing computational time*, see graph on Figure 2.

Remark 3: The very important fact is that the preprocessing step is computed only once. Moreover, the database can be easily updated by adding new images. Updating means that only the F^1-transform components of the new images are computed, re-computation of the whole database is not necessary.

B. Searching for non-cropped images

The first experiment is about searching for non-cropped images (patterns) in the databases DB1, DB2 and DB3. We

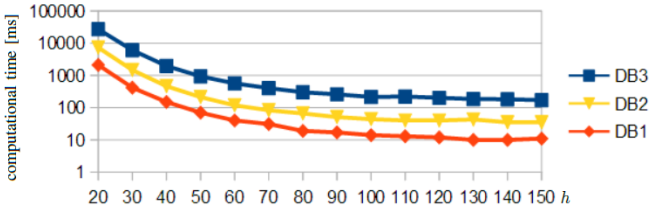

Fig. 3. Influence of the chosen h on the comp. time [ms] of steps *S1*, *S2*.

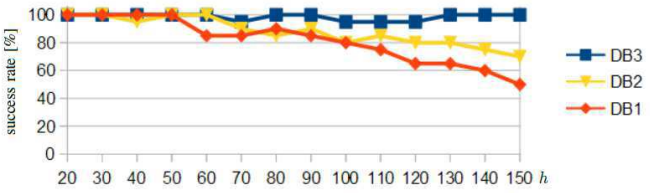

Fig. 4. Influence of the chosen h on the success rate [%] of searching for cropped pattern images.

randomly selected 20 images from each database to be used as the patterns. Let us remark that these patterns (images) occur in the databases in the same form, i.e., there are no modifications. The goal is to find and mark each pattern in the database in the reasonable time and with the high success rate.

The success rate is expressed by a percentage value of successfully found patterns with respect to all searched patterns. In this experiment we achieved the 100 % success rate, i.e., all the pattern images were found correctly.

The interesting point is the influence of the chosen parameter h on the computational time of this phase (steps *S1*, *S2*), see graph on Figure 3. It can be observed that the bigger h leads to the shorter computational time. As we mentioned above, the bigger value of h, the less number of the F^1-transform components. On one hand, the value of h does not influence the computational time of the preprocessing (see graph on Figure 2), on the other hand, the less number of the components leads to the less computations of distances and comparisons.

C. Searching for cropped images

The second experiment is about searching for the cropped images (patterns) in the databases DB1, DB2 and DB3. We again randomly selected 20 images from each database and modified them to represent the cropped patterns. It means that the patterns are small parts cut out from the original images and they differ in size (they are substantially smaller). The goal is to find and mark the image (and its exact area) in the database which contains the searched pattern.

First, we applied the algorithm to this task in the same way as it was done in the previous task. The result of this experiment was good but we did not achieve 100 % success rate, i.e., not all the pattern images were found correctly. The explanation is following. The success rate is dependent on the chosen parameter h (see graph on Figure 4) and on the cropped pattern image. In the previous case, the pattern image and the images in the database had the same size. Therefore, the components of all images were in the same positions and the comparison was straightforward. In the case of the cropped pattern images, the components of the pattern image can be shifted with respect to the components of the original image. The comparison of the pattern image and the images in the database is obtained by computing distances between components. In the case of the cropped images, the components do not cover the same areas of the image as the corresponding components in the original image (the components are computed from a different partition position) and therefore, the distances between them are non-zero. This can leads to an incorrect matching, i.e., the algorithm finds and marks the image with smallest distances but the found image is not the correct one.

This problem can be solved by using the smaller value of h. By this we obtain more components and compute more distances. Therefore, the computational time increases. For example, the average computational time of searching for one cropped pattern in DB3 varies from approximately 2min for $h = 20$ to 140ms for $h = 150$. The further theoretical background for the explanation and solution of this problem (i.e., how much is the distance influenced by the chosen partition, how much should be the parameter h decreased in order to avoid incorrect matching) is out of the scope of this paper. For lack of limited space, we briefly present the main idea of the proposed improvement, the detailed description will be presented in the next paper.

Searching for cropped images - improvement: The main idea how to achieve 100 % success rate is following. The algorithm described in Section III-A is executed three times for decreasing input parameter h. We proved that the best option is to use $\langle h, h/2, h/4\rangle$. Then for each image we compute a cumulative distance as a sum of the distances from the three runs. The output image is the one with the minimal cumulative distance.

Using this improvement, we finally achieved the 100 % success rate but we increased the computational complexity. For example, the computational time is approximately 700ms for h chosen as $\langle 240, 120, 60\rangle$. Let us remark that the improved algorithm works reliable even for very small cropped patterns, see for example Figure 5.

V. Discussion, Future work

The preprocessing step (representation of the databases by the F^1-transform components) can take several minutes. There are several possibilities how to reduce the computational time of this step.

1) *Parallelization:* The problem can be easily parallelized, each CPU core can compute components for different images separately.
2) *Computer:* The computational time was measured on a notebook with a weak power. Usage of a fast desktop

Fig. 5. *Left:* The cropped pattern image. *Right:* The found image with the exact area marked by the red rectangle (found in the database of 1292 images).

computer can rapidly reduce the computational time.

3) *GPU:* The F^1-transform can be implemented in the form of convolution (see for example [20]). The convolution matrix can be computed by GPU and therefore the computational time can be significantly reduced.

Let us recall that we focused on two aspects: computational time and success rate. Finally, we achieved in all cases 100 % success rate. In the case of cropped pattern images, the success rate 100 % was achieved at the expense of higher computational time (dependent on the chosen h). On the other hand, in the case of non-cropped pattern images, we achieved 100 % success rate in all cases. It means that the algorithm is successful even for the maximal h in the ultra-fast computational time.

We investigated basic possibilities of the image plagiarism task. During the exploration process we found out several sub-tasks to be investigated in *the future work*:

1) The input parameter θ in the algorithm is now user-dependent. Based on our preliminary investigation, the threshold θ depends on differences between pattern components and can be computed automatically.
2) We started with searching for the same images or the cropped images. Another important task is searching for different-sized images and it requires a deeper study.
3) We intend a further investigation of image retrieval methods and their possible combination or comparison with our approach.

VI. Conclusion

In this paper, we focused on searching for plagiarized images in databases. We proposed the searching algorithm $FTIP$ based on the F^1-transform technique which was mainly used in the preprocessing step. The preprocessing step was proven to be very important: it is computed only once and it reduces the domain dimension. Therefore, this step significantly speeds up the whole process of the image searching.

To demonstrate the power of our searching method, we created three different databases (different sizes and resolutions). We tested two problems of searching for plagiarized images: searching for same-size images (non-cropped images) and searching for cropped images. *We showed that the proposed algorithm is very fast (approximately 100ms) with the* 100% *success rate for the non-cropped images.* We demonstrated that the 100% success rate can be achieved also for the cropped images with a slight improvement of the original algorithm.

Acknowledgment

This work was supported by the European Regional Development Fund in the project of IT4Innovations Center of Excellence (CZ.1.05/1.1.00/02.0070, VP6 and by the IT4I XS project number LQ1602.

References

[1] H. R. Fowler, J. E. Aaron *et al.*, *The little, brown handbook*. Pearson Longman, 2007.
[2] M. Bouville, "Plagiarism: Words and ideas," *Science and Engineering Ethics*, vol. 14, no. 3, pp. 311–322, 2008.
[3] J. Kasprzak, M. Brandejs, and M. Kripac, "Finding plagiarism by evaluating document similarities," in *Proc. SEPLN*, vol. 9, 2009, pp. 24–28.
[4] E. Stamatatos, "Intrinsic plagiarism detection using character n-gram profiles," *threshold*, vol. 2, pp. 1–500, 2009.
[5] A. Si, H. V. Leong, and R. W. Lau, "Check: a document plagiarism detection system," in *Proceedings of the 1997 ACM symposium on Applied computing*. ACM, 1997, pp. 70–77.
[6] S. Butakov and V. Scherbinin, "The toolbox for local and global plagiarism detection," *Computers & Education*, vol. 52, no. 4, pp. 781–788, 2009.
[7] D. Müllensiefen and M. Pendzich, "Court decisions on music plagiarism and the predictive value of similarity algorithms," *Musicae Scientiae*, vol. 13, no. 1 suppl, pp. 257–295, 2009.
[8] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 277–286.
[9] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.
[10] Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: Current techniques, promising directions, and open issues," *Journal of visual communication and image representation*, vol. 10, no. 1, pp. 39–62, 1999.
[11] L. Van Heerden *et al.*, "Detecting internet visual plagiarism in higher education photography with google search by image: proposed upload methods and system evaluation," Ph.D. dissertation, Bloemfontein: Central University of Technology, Free State, 2014.
[12] I. Perfilieva, "Fuzzy transforms: Theory and applications," *Fuzzy Sets and Systems*, vol. 157, pp. 993–1023, 2006.
[13] I. Perfilieva, M. Daňková, and B. Bede, "Towards a higher degree F-transform," *Fuzzy Sets and Systems*, vol. 180, pp. 3–19, 2011.
[14] P. Hodáková, *Fuzzy (F-)transform of functions of two variables and its application in image processing*. Ostrava: University of Ostrava, 2014. [Online]. Available: http://irafm.osu.cz/f/PhD_theses/Hodakova.pdf
[15] P. Hurtik, P. Hodakova, and I. Perfilieva, "Fast string searching mechanism," in *Proceedings of the IFSA-EUSFLAT 2015*. EUSFLAT.
[16] M. Holcapek and T. Tichy, "Discrete fuzzy transform of higher degree," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2014.
[17] D. Zhang and G. Lu, "Evaluation of similarity measurement for image retrieval," in *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, vol. 2. IEEE, 2003, pp. 928–931.
[18] P. Hurtik and I. Perfilieva, "Image reduction/enlargement methods based on the f-transform," *European Centre for Soft Computing, Asturias*, pp. 3–10, 2013.
[19] ——, "Image compression methodology based on fuzzy transform using block similarity," in *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*. Atlantis Press, 2013.
[20] P. Vlašánek and I. Perfilieva, "F-transform and discrete convolution," in *Proceedings of the IFSA-EUSFLAT 2015*. EUSFLAT.

P. Hurtik, M. Vajgl, and M. Burda. Jewelry Stones Classification: Case Study. *Proceedings of 2015 Seventh International Conference of Soft Computing and Pattern Recognition (SoCPaR 2015)*, 205–210, IEEE, 2015.

Jewelry Stones Classification: Case Study

Petr Hurtik, Marek Vajgl, Michal Burda
University of Ostrava, Centre of Excellence IT4Innovations, Institute for Research and Applications of Fuzzy Modeling,
30. dubna 22, 701 03 Ostrava 1, Czech Republic,
Petr.Hurtik@osu.cz, Marek.Vajgl@osu.cz, Michal.Burda@osu.cz

***Abstract*—The paper introduces a real-life industrial problem: a jewelry stones classification. The stones are represented by their camera images. The goal of the contract was to evaluate stones into two (or more) specified classes according to their quality. Given requirements include very high processing speed and success rate of the classification. The goal of this paper is to publish a report of this contract and show a way how this task can be solved. In this paper we aim to usage of machine learning with respect to the image processing. We also design own learning and classification algorithm and answer the question if there is a place for a new machine learning algorithm. As an output of this paper a benchmark of the proposed algorithm with 81 state-of-the-art machine learning methods is presented.**

***Keywords*-image classification; machine learning; image processing; perceptron separation tree**

I. Introduction

A machine learning is an approach in which classical human decision is replaced by automated one using data-driven prediction or decision. In this work we will look at the machine learning from the two points of the view: a) what is it used for, what is the target area of the usage; b) what is it based on, which tools and approaches were used to make the system working.

The first view, usage, covers a wide range of applications: there are challenges in a text recognition [1], sound recognition [2], some industrial products quality evaluation [3], human face detection and recognition [4] and others. All previously mentioned approaches share the same aspect: they are designed only for the one specific task.

The second view, a kind of the algorithm, is as same wide as the first case. Let us mention: Neural Networks [5] and their actual derivation Deep Learning [5], Trees and their derivatives Forests (Random Forests etc.) [6], Support Vector Machines [7], Regression Models [8], Boosted Models [4] and many other including their combinations.

The final mentioned solution is a pair of an task and an algorithm. Globally, any of the mentioned algorithm is significantly better than other; solving some problem require expert knowledge to decide which algorithm is appropriate for the corresponding specific problem.

In this paper, we will focus on a quality evaluation of industrial products. We are dealing with jewelry stones represented by their pictures taken by a single shot of a camera. By quality evaluation we mean classification of a stone into predefined classes corresponding to the stone damage, i.e., good stone or stone damaged by scratches, insufficient polishing etc. The classes are given by expert in the field of stone classification.

The motivation arises from the fact that increased production rate and the requirement of the testing of the quality of the resulting products causes that it is no longer possible to evaluate quality only by human experts. In this case, automatic testing needs to be taken into an account, as the mass of industrial production of the products is growing together with customer needs and improved technical processes. The mentioned algorithm has to deal with only one stone size and shape. However, for the different size etc. the new learning can be executed using here presented algorithm.

The aim of this paper is to describe our solution of the problem to take an unknown-quality jewelry stone and classify it into one of the predefined classes (with respect to an human expert classification) as fast as is possible with the maximal success rate. The classification will be connected with own-designed learning algorithm. The second goal of the paper is to show a comparison with state-of-the-art algorithms and evaluate their computation speed and the success rate. This presentation can be used also as a general handbook presenting how the image classification problem can be solved – it shows how the data can be accessed, how the preprocessing can be done, or to answer the question if its better to use some well-known learning algorithm or design an own one.

The structure of this paper is as follows: at first, we describe parts of project with corresponding images in Section II. The processing of the obtained images is explained in Section III. This section also gives an overview about features extraction. As the features are used in learning and classification part, the proposed learning algorithm is described in Section IV. Finally, the comparison of the proposed algorithm with state-of-the-art can be found in Section V. At the end, the conclusion is established in Section VI.

II. Project Milestones, Overview of Used Images

In this section we describe input images according to the project milestones from the beginning to the current state. For purpose of this paper we define an image as a gray-scale function $f : \{1, 2, \ldots, W\} \times \{1, 2, \ldots, H\} \rightarrow \{0, 1, \ldots, 255\}$, where W (resp. H) is the considered width (resp. height).

A. Phase I: based on one manually obtained image

At the beginning of the project, the idea was to make one image per stone. Then, preprocessing of the image was realized in order to obtain a stone separated from background. Image of the separated stone was used to a defect detection. The detection has to be realized according to a damage type,

such as "this is a small scratch", "this is a set of scratches" etc. Figure 1 illustrates two images of two different stones: good one and damaged one. The mentioned process was established and published in [9]. From Figure 1 can be also observed that the classification into good and bad class is very trivial (e.g., using average brightness). Unfortunately, the process of image capturing was handled manually. Moreover, the processing speed and recognition rate did not fulfill required time and qualitative limits and therefore this approach has been abandoned.

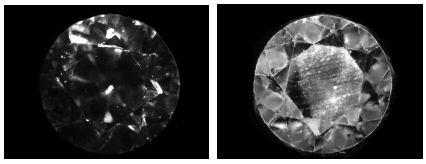

Fig. 1. Phase I stone images. Left: good stone. Right: damaged stone.

B. Phase II: based on four automatically obtained images

The reflections on the stone are dependent on a light and a camera position. The idea is to create several images of one stone with different lights' positions and fuse them into one image in order to reduce these reflections. The final image is composed from four ones using image fusion [10]. In this part of contract, the stone is classified into one of six predefined classes – one representing good stone, five representing particular damage types. We also defined own learning / classification algorithm [3]. This learning algorithm is also the base for the new version described in Section IV. The algorithm worked with $1000 \times 500px$ images (cropped manually to $320 \times 320px$) and achieved about 20 classifications per seconds including loading and processing image, handling dll calls etc. The obtained success rate of the classification was 98% in case of two classes and 86% in case of six classes respectively. Figure 2 shows how the images from the second phase look like. The classification rate was evaluated as success, the improvement of the classification speed become point of the next phase.

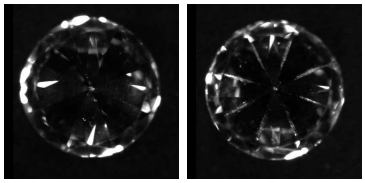

Fig. 2. Phase II stone images. Left: good stone. Right: damaged stone.

C. Phase III: solution aimed at high processing speed

In the third phase of the contract, processing speed become much more important parameter. Therefore we return back to capturing one image per stone. Moreover, using faster camera with lower resolution $500 \times 500px$, the quality of stone image decreased. Figure 3 illustrates the obtained images. To achieve required classification speed (hundreds classifications per second) only two groups (good vs. damaged stones) are determined. This paper is aimed at the solution of problem of image classification over images taken in this phase.

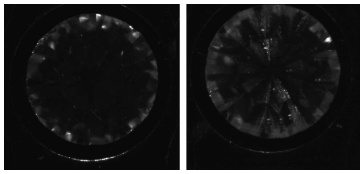

Fig. 3. Phase III stone images. Left: good stone. Right: damaged stone.

III. Image processing

In this section we introduce image processing methods used in order to follow up two goals. The first goal is to modify images to achieve higher classification speed. The second one is to modify images somehow to achieve higher success rate of classification. Let us remark, the two methods presented bellow are only small piece of tested ones such as application of mathematical morphology, non-linear filtering, textures detection etc.

A. Decreasing of an image size

One of the easiest way to increase the computation speed is to decrease an image resolution. We combine two methods: removing useless pixels and reducing the image size.

By removing useless pixels we mean preserving stone and removing everything else. At first, stone center coordinates (x', y') are searched as $x' = \bar{\mathbf{X}}$ where $\bar{\mathbf{X}}$ represents mean value of $\mathbf{X} = \{x \in f | f(x, y) > \tau\}$ for all combinations of y. By τ we mean minimal intensity (such as $\tau = 10$ etc.) to be considered. The same process is realized for searching of y'. Then, using externally known stone radius r the new image is created as $f' = \{(x, y) \in f |\ |x - x'| \leq r, |y - y'| \leq r\}$.

The next step is image reduction that is formally defined as $f \rightarrow f'$ where f' is new function with width W' and height H' such that $W' < W$ and $H' < H$. Due to requirements we chose the fastest reduction algorithm, Nearest Neighbor [11].

Figure 4 shows input and output after the removing useless pixels and image reduction. The reduced image resolution is approx. $10\times$ lower.

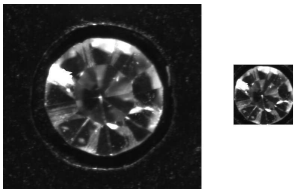

Fig. 4. On the left the original image, on the right side the reduced image used for feature extraction. The example is for damaged stone.

B. Reflection suppressing

In the image, the light reflections are presented as areas where intensities are near or equal the maximal value. These

reflections can affect feature values used in the learning part. However these reflections cannot be separated by a simple threshold value as stone defects can be represented by intensities near to the maximal one too. Therefore, simple thresholding can erase important information.

To suppress reflections we use modified stacked flood fill. Let us define pair of thresholds T_1 and T_2. At first, using stacked flood fill, we detect areas with high intensities. If an particular area volume is higher than T_1 (to preserve small scratches) and lower than T_2 (to preserve big damages), the area is erased. Figure 5 shows the output of the algorithm. The stacked flood fill was chosen due to its low computation cost. Let us remark that originally we experimented with mathematical morphology [9]. However, due to high computational cost this approach was no more used.

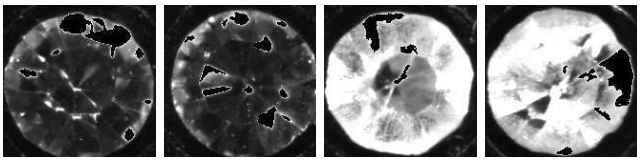

Fig. 5. Example of reflections suppression from two good stones (the left ones) and highly-damaged stones (the right ones).

C. Feature detection

There are several ways how features can be extracted:

- use as much general descriptors as possible[1],
- use feature points (e.g, corner detection [12]),
- use descriptors based on an expert knowledge,
- reduce image in the size (e.g., using F-transform [13]) into a few elements (pixels, components etc.) and use these elements as features.

Our algorithm has to work in accordance to a human expert. Therefore, we decided to define features according to the expert knowledge. The list of chosen basic features is as follows:

1) average intensity,
2) average intensity of non zero pixels,
3) sum of intensities on a centering circle,
4) difference between intensities on centering circles,
5) ratio between mean of intensities and mean of intensities after removing zero pixels,
6) standard deviation of intensities inside centering circle,
7) percentile value,
8) power of intensities inside centering circle,
9) average value of intensities inside centering circle,
10) average gradient value.

From these *basic* features and their combinations we implement 37 variants of features.

IV. Learning / classification algorithm

In this section we give an explanation of the proposed learning / classification methods working over defined features. We will call them *PBDT*. The method has two historical version: the first one, called *PBDT1*, was designed to reach high computation speed sufficient for the project. The second one, current algorithm called *PBDT2*, extends the original one to obtain higher success rate.

[1]https://code.google.com/p/jfeaturelib/

A. The original learning algorithm - PBDT1 [3]

We chose a decision tree as a basis for the algorithm as it is easy to interpret. The used tree terminology is following: each part of the tree is called *node*. the topmost node of the tree is called *root*. Nodes without child nodes are *leafs*. The motivation was that the project starts with human expert's advises such as *stone is good or damaged*; *the damage is inside, or outside*; *if the damage is inside, the type is on the top or bottom*; etc. It is very natural to put these rules into tree and create fixed tree topology. Let us remark that the standard way how a tree structure is composed is using measurement of an entropy [14].

As the decision trees are mainly suitable for text–labeled data, firstly we have to replace the text decision (good vs damaged etc.) by some numerical decision. For a decision, a feature value is used together with its corresponding threshold (e.g., "under/above 0.5").

$PBDT1$ learning is a process which assigns pair of feature with its threshold to each non-leaf node. Let T be the set of training images for particular non-leaf node. According to the classification expected at this node, let S_1 and S_2 represent two non-empty disjoint sets of training images such that $S_1 \cup S_2 = T$, where S_1 contains images classified as first group following left sub–node of the current node in the tree, S_2 contains images of second group following right sub–node of the current node in the tree. For example, as root node represents classification for good/damaged stones, set S_1 represents images of good stones, S_2 represents images of damaged stones. The learning algorithm has to, for given n features C_i, $i = 1, ..., n$, find a feature C_i and such threshold τ_i, which provides the best separation of the training set T into known S_1 and S_2 according to the intention of the tree node being trained.

Let $c_i(t)$ be a value of i^{th} feature for image t. For each image $t \in T$ a vector $\boldsymbol{c}_t = (c_1(t), \ldots, c_n(t))$ of feature values is computed. Let $S_j^{C_i}$ $(j = \{1, 2\})$ is a vector of i^th feature values for all stones in S_j.

Now, for all n features C_i we seek for τ_i in the non-leaf node. At the beginning of the algorithm, τ_i is initialized to a random value of $S_1^{C_i} \cup S_2^{C_i}$. After that, τ_i is modified iteratively by a separation algorithm to minimize classification error ω_i, which equals to the number of incorrectly classified stone images. Finally, for the node being processed, a pair $\langle C_i, \tau_i \rangle$ with the minimal classification error ω_i is selected. *The output of the learning process is a tree topology, where each non-leaf node has assigned a pair* $\langle C_i, \tau_i \rangle$.

Remark 1: The classification error ω_i is defined as an absolute number of incorrect classifications.

Separation algorithm: Given $S_1^{C_i}, S_2^{C_i}$, the goal of the separation algorithm is to find τ_i that divides them into correct

classification classes to obtain minimum ω_i. In [3], the possible relations of $S_1^{C_i}, S_2^{C_i}$ are determined.

In general, separation algorithm should satisfy the following conditions: the solution is found in a finite time; the solution is optimal, i.e., there does not exist any other solution ω_i' satisfying $\omega_i' < \omega_i$. In order to these, we propose following separation algorithm executed for all $i \in \{1, ..., n\}$ consisting of two preprocessing steps $P1$, $P2$ and five $S1...S5$ processing steps.

$In:$ $S_1^{C_i}$, $S_2^{C_i}$
$P1$: $\tau_i =$ random item from $S_1^{C_i} \cup S_2^{C_i}$;
$P2$: $p_i =$ standard deviation of $S_1^{C_i} \cup S_2^{C_i}$;
$S1$: $\forall e \in S_1^{C_i}$ do: if $e > \tau_i$ then $\tau_i = \tau_i - p_i$;
$S2$: $\forall e \in S_2^{C_i}$ do: if $e < \tau_i$ then $\tau_i = \tau_i + p_i$;
$S3$: compute error ω_i of separation; break if $\omega_i = 0$;
$S4$: set $p_i = 0.5p_i$;
$S5$: if τ_i changed from the previous iteration by more than p_i, go to S1;
Out: value τ_i and separation error ω_i.

The algorithm runs twice for each non-leaf node and each feature C_i. In the second run, $S_1^{C_i}$ and $S_2^{C_i}$ are swapped. The combination $\langle C_i, \tau_i \rangle$ with the lowest ω_i is selected.

Classification: Once the decision tree is learned, its usage for unknown images in order to classify the grinding defect is quite straightforward. Every non-leaf node of the tree is assigned with a pair $\langle C_i, \tau_i \rangle$, i.e., a feature C_i and a separation threshold τ_i, and every leaf node is assigned with a resulting class. The algorithm starts from a root node. For an image t to be classified, a feature value $c_i(t)$ of the actual node is computed. If $c_i(t) < \tau_i$, the algorithm continues on the left child node, otherwise right child node is selected. The tree is descended this way until a leaf node is achieved. The walk through the tree describes the classification of the damage type.

Similar approaches: As it is obvious from the description, the main idea of the learning algorithm is a binary decision tree. But there are several differences in our implementation. The first one, topology: in decision tree, the topology is created automatically, e.g., using entropy. This technique can be seen in so called *ID3* algorithm [15]. In our case, the topology is created manually using expert knowledge. The second difference is determination of τ_i value for C_i. In our case the own separation algorithm has the same property as a special case of general perceptron with one input and one output. The technique of hybrid tree with perceptrons is known as *Perceptron Decision Tree* [16]. The difference is in the special form of perceptron we use: only one input, only one output.

B. Improved version of the original algorithm - PBDT2

The algorithm described above excels in computation speed. At first, it uses only one feature, so only one feature value needs to be calculated during classification. Next, classification means to compare only one feature value with the given threshold. The comparison can be taken at no computational cost. On the other hand, the usage of only one feature means that *there must exists feature explaining label*. As this the proposition is not fulfilled generally, we improved the original approach by a non-leaf node preprocessing.

The preprocessing is executed before original learning, for each non-leaf node for given $S_1 \cup S_2 = T$ and $S_1^{C_i}$, $S_2^{C_i}$ and consists of the six following processing steps $S1...S6$:

In: $S_1^{C_i}$, $S_2^{C_i}$, $k = 1$
$S1$: $\tau_{i,1}^k = \max(S_1^{C_i})$; $\tau_{i,2}^k = \min(S_2^{C_i})$;
$S2$: $S_1^{C_i^k} = \{e \in S_1^{C_i} | e < \tau_{i,2}^k\}$;
$S_2^{C_i^k} = \{e \in S_2^{C_i} | e > \tau_{i,1}^k\}$
$S3$: find C_i^k which holds $\min(|S_1^{C_i^k}| + |S_2^{C_i^k}|)$ and store triplet $\langle C_i^k, \tau_{i,1}^k \tau_{i,2}^k \rangle$;
$S4$: for all n features delete from $S_1^{C_n}$ all values in $S_1^{C_i^k}$; the same for $S_2^{C_n}$ delete all values in $S_2^{C_i^k}$;
$S5$: $k = k + 1$;
$S6$: if $|S_1^{C_i}| + |S_2^{C_i}| > 0$ and $|S_1^{C_i^k}| + |S_2^{C_i^k}| > 0$ go to $S1$;
Out: modified $S_1^{C_i}$, $S_2^{C_i}$, k triplets $\langle C_i^k, \tau_{i,1}^k \tau_{i,2}^k \rangle$.

The outputs of this processing are modified sets $S_1^{C_i}$, $S_2^{C_i}$. Then, $PBDT1$ is executed over these sets.

Classification: The classification is straightforward and uses values obtained during iterations: given k triplets $\langle C_i^k, \tau_{i,1}^k, \tau_{i,2}^k \rangle$, the image feature value $c_i^k(t)$ for C_i^k is computed. If $c_i^k(t) < \tau_{i,1}^k$, algorithm continues on the left child node. If $c_i^k(t) > \tau_{i,2}^k$, it continues on the right child node. If all k triplets are processed and no decision is made, execute $PBDT1$ with $PBDT1$ learned pair $\langle C_i, \tau_i \rangle$.

V. Experiments

For the experiments we design test set including 1144 stone images. The set consists of 564 good stones and 580 damaged stones. For each stone in the training set a full vector of 37 features was precomputed. The test set was designed using all available stone images. For the comparison with $PBDT2$ we chose 81 state-of-the-art methods from R package $carret$ and $PBDT1$.

A. Success rate

The experiment was executed using leave-one-out method. I.e., one stone was left out of the learning set, an algorithm was trained on the rest 1143 stones and then, classification for one left stone was computed. The process was repeated until all stones in the training set were processed as left-out ones (1144 iterations for our testing set). The final success rate is computed as a mean value of the each iteration classification.

Let us emphasize several interested results illustrated in Table I. Method *kknn* [17] achieved 100 % success rate. To simulate real conditions, the test set includes four stones with incorrect label, so the 100 % success rate leads to assumptions that $kknn$ was lucky and does not work properly.

The nice results were obtained by the boosted method *AdaBoost.M1* [18]; fuzzy methods *FRBCS.CHI/FRBCS.W* [19]; the improved version of *C4.5* tree, *C5.0* [20]; random forests methods *parRF* and *rf* [21]; and support vector based *svmPoly* [22].

TABLE I
MEASURED ACCURACY FOR TESTED METHODS

rank	method	accuracy	rank	method	accuracy
1	kknn	1.00000	41	lda2	0.97290
2	AdaBoost.M1	0.99563	44	gamboost	0.97203
2	FRBCS.CHI	0.99563	45	**PBDT1**	0.97115
4	FRBCS.W	0.99388	46	bstSm	0.97028
5	C5.0Cost	0.99126	46	gcvEarth	0.97028
5	C5.0	0.99126	46	rocc	0.97028
7	parRF	0.98951	49	bagFDAGCV	0.96941
8	gbm	0.98864	49	nb	0.96941
8	rf	0.98864	51	rFerns	0.96853
8	RRFglobal	0.98864	52	lssvmRadial	0.96766
8	svmPoly	0.98864	53	stepQDA	0.96591
12	Boruta	0.98776	54	glmboost	0.96416
12	RRF	0.98776	55	RFlda	0.96066
12	treebag	0.98776	55	spls	0.96066
15	AdaBag	0.98514	57	knn	0.95892
15	pcaNNet	0.98514	58	partDSA	0.95629
17	LogitBoost	0.98427	59	Mlda	0.95542
18	bstTree	0.98339	60	lvq	0.95455
18	nodeHarvest	0.98339	61	xyf	0.93444
20	evtree	0.98252	62	pda2	0.92570
20	gamLoess	0.98252	63	sda	0.92133
20	rpart2	0.98252	64	bdk	0.91871
23	rpartCost	0.98164	65	kernelpls	0.91783
24	bayesglm	0.98077	65	pls	0.91783
25	svmRadialCost	0.97990	65	simpls	0.91783
25	svmRadialWeights	0.97990	65	widekernelpls	0.91783
27	blackboost	0.97902	69	PenalizedLDA	0.91608
27	rpart	0.97902	70	stepLDA	0.90472
29	C5.0Rules	0.97815	71	ada	0.89519
30	ctree	0.97727	72	bstLs	0.86801
30	svmRadial	0.97727	73	elm	0.86626
32	ctree2	0.97640	74	CSimca	0.84003
32	C5.0Tree	0.97640	75	RSimca	0.71503
32	glmnet	0.97640	76	avNNet	0.59003
32	svmLinear	0.97640	77	mlp	0.53059
36	**PBDT2**	0.97552	78	nnet	0.52622
37	glm	0.97465	79	mlpWeightDecay	0.52360
38	bagEarth	0.97378	80	rbf	0.50699
38	bagEarthGCV	0.97378	81	rbfDDA	0.50524
38	pda	0.97378	82	oblique.tree	0.49301
41	cforest	0.97290	82	protoclass	0.49301
41	lda	0.97290			

TABLE II
MEASURED COMPUTATION TIME FOR TESTED METHODS [MS]

method	learn	classify	method	learn	classify
PBDT1	2	0.000001	rf	968	0.01104
PBDT2	17	0.000002	RRFglobal	1143	0.01300
sda	510	0.00098	bdk	358	0.01349
bstLs	1131	0.00115	xyf	354	0.01388
rpart	342	0.00115	ctree2	331	0.01410
rpartCost	326	0.00115	ctree	337	0.01459
glm	365	0.00137	gamLoess	7965	0.01502
rpart2	328	0.00137	treebag	845	0.01945
gamboost	1526	0.00137	svmRadial	380	0.01956
gbm	437	0.00148	svmRadialCost	388	0.02021
bayesglm	604	0.00153	svmRadialWeights	346	0.02082
lvq	338	0.00158	blackboost	1104	0.02136
glmnet	501	0.00202	svmPoly	414	0.03300
glmboost	336	0.00208	RFlda	306	0.03682
lda2	263	0.00279	Mlda	348	0.03726
nnet	323	0.00284	kknn	483	0.04797
lda	296	0.00290	knn	299	0.05365
nodeHarvest	4696	0.00322	C5.0Tree	339	0.06971
PenalizedLDA	333	0.00339	C5.0Rules	338	0.07015
pda	316	0.00344	C5.0	386	0.07108
pda2	270	0.00344	C5.0Cost	332	0.07113
kernelpls	351	0.00361	LogitBoost	322	0.07643
simpls	327	0.00361	rocc	410	0.09184
widekernelpls	333	0.00361	bstTree	1240	0.11287
evtree	2114	0.00366	RSimca	1017	0.16073
pls	312	0.00366	CSimca	536	0.16231
elm	332	0.00388	ada	1335	0.16537
svmLinear	348	0.00426	bagEarth	1774	0.19313
lssvmRadial	1191	0.00437	AdaBag	7185	0.19946
gcvEarth	435	0.00453	AdaBoost.M1	7327	0.21105
pcaNNet	329	0.00470	bagEarthGCV	2668	0.21198
spls	376	0.00486	bagFDAGCV	2351	0.33233
oblique.tree	540	0.00503	protoclass	1493	0.67734
rFerns	323	0.00514	mlp	2644	1.50535
partDSA	7727	0.00601	mlpWeightDecay	2310	1.50683
parRF	630	0.00694	rbf	2834	1.73853
bstSm	8391	0.00798	rbfDDA	3111	1.89510
stepQDA	4415	0.00841	nb	351	3.83976
avNNet	324	0.00847	cforest	1120	5.26415
stepLDA	4484	0.00858	FRBCS.CHI	2320	629.532
Boruta	132638	0.01016	FRBCS.W	2322	630.545
RRF	1987	0.01087			

The result shows that the original proposed algorithm 45^{th} place was improved to 36^{th} place by a new version. Although the 36^{th} place does not look like as a success, our approach loses only about 2% to the top algorithms.

B. Computation time

At first, we tested learning time. According to the requirements, the learning time is not crucial, if it is in the matter of minutes. Table II shows learning times for all mentioned algorithms. Because of speed of the separation algorithm of $PBDT$, the proposed algorithms are the fastest of all mentioned. To prove that our algorithm is really fast, we also test dependency of number of images in the learn set on learn time. Table III shows that the learn time is linear and can be easily estimated accordingly to the number of used images.

From the classification speed point of view, the proposed algorithms are again the fastest. Possible explanation is that used known algorithms were tested using R framework. On the other side, this reflects real case, when an user take implemented algorithm, without its rewriting. Remark: In case

TABLE III
DEPENDENCY OF IMAGES IN LEARN SET ON LEARNING TIME FOR PBDT2

learning set size [images]	learn–only time [ms]	read+preprocess+learn time [ms]
200	4.0	12,625
400	6.5	25,172
600	9.0	36,578
800	11.5	51,828

of further operations are included, as calling our dll from external app, feature values computation, image reading and image resizes, the classification time is approx 2.2ms per stone. Moreover, if time of reflections removing is included, the final time per classification is approx 27ms. Let us remark that the source code for reflection suppressing was not optimized. Without taking into account this reflection removing, our classification algorithm is able to classify approx 450 stones per second on non-parallelized version running on a standard-powered notebook. We can expect that using existing algorithm, their computation time can be much more bigger. The reason is that our algorithm does not use all features, while other algorithms can.

VI. CONCLUSION

In this paper the process of solving real–life project of classification of jewelry stones with high classification rate and success rate was described.

At first, we show the way of capturing images and justify why the decreased-quality images are processed instead of the original ones. Then, we introduce image preprocessing in order to achieve lower computation cost and remove artifacts to create more robust solution. For the preprocessed image we extract image features based on a human expert knowledge. To process these features we recall own learning algorithm similar to perceptron decision tree. Furthermore, the original approach was improved in order to achieve better success rate. Finally, we design the test consisting of classification of 1144 images and we presented a comparison with 81 state-of-the-art learning/classification methods. In our opinion, the comparison is also useful as an quick overview of well-known methods.

Taking into account success rate of classification and computation time of learning and classification part, we can evaluate $PBDT2$ algorithm as one of the best. The presented solution of the project successfully fulfilled requirements and proposed solution is prepared to be tested in the industrial production.

The report can be concluded as follows: usage of some well-known machine algorithm for solving of a general task is a good choice. But, if the task includes some special requirements (e.g., processing speed is more important than success rate), the design of an own, or redesign of a know algorithm can be justified.

VII. ACKNOWLEDGMENT

This work was supported by the European Regional Development Fund in the project of IT4Innovations Center of Excellence (CZ.1.05/1.1.00/02.0070, VP6 and by the IT4I XS project number LQ1602.

REFERENCES

[1] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[2] P. Rani, C. Liu, N. Sarkar, and E. Vanman, "An empirical study of machine learning techniques for affect recognition in human–robot interaction," *Pattern Analysis and Applications*, vol. 9, no. 1, pp. 58–69, 2006.

[3] P. Hurtik, M. Burda, and I. Perfilieva, "An image recognition approach to classification of jewelry stone defects," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint IFSA World Congress*. IEEE, 2013, pp. 727–732.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–511.

[5] C. Clark and A. Storkey, "Training deep convolutional neural networks to play go," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1766–1774.

[6] V. Y. Kulkarni and P. K. Sinha, "Random forest classifiers: a survey and future research directions," *nternational Journal of Advanced Computing*, vol. 36, no. 1, pp. 1144–1153, 2013.

[7] G. Mountrakis, J. Im, and C. Ogole, "Support vector machines in remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 3, pp. 247–259, 2011.

[8] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[9] I. Perfilieva, P. Hodakova, M. Vajgl, and M. Dankova, "Classification of damages on jewelry stones: Preprocessing," in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, 2013.

[10] M. Vajgl and I. Perfilieva, "Improved f-transform based image fusion," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2014, pp. 153–162.

[11] P. Miklos, "Image interpolation techniques," in *2nd Siberian-Hungarian Joint Symposium On Intelligent Systems*, 2004.

[12] P. Hurtík, I. Perfilieva, and P. Hodáková, "Fuzzy transform theory in the view of image registration application," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2014, pp. 143–152.

[13] P. Hurtik and I. Perfilieva, "Image reduction/enlargement methods based on the f-transform," *European Centre for Soft Computing, Asturias*, pp. 3–10, 2013.

[14] C. Burch, "A survey of machine learning," Tech. report, Pennsylvania Governor's School for the Sciences, 2001. 4, Tech. Rep.

[15] R. L. De Mántaras, "A distance-based attribute selection measure for decision tree induction," *Machine Learning*, vol. 6, no. 1, pp. 81–92, 1991.

[16] P. E. Utgoff, "Perceptron trees: A case study in hybrid concept representations," *Connection Science*, vol. 1, no. 4, pp. 377–391, 1989.

[17] K. S. . K. Hechenbichler, *kknn: Weighted k-Nearest Neighbors*, 2014, r package version 1.2-5. [Online]. Available: http://CRAN.R-project.org/package=kknn

[18] E. Alfaro, M. Gámez, and N. García, "adabag: An R package for classification with boosting and bagging," *Journal of Statistical Software*, vol. 54, no. 2, pp. 1–35, 2013. [Online]. Available: http://www.jstatsoft.org/v54/i02/

[19] L. S. Riza, C. Bergmeir, F. Herrera, and J. M. Benítez, "frbs: Fuzzy rule-based systems for classification and regression in R," *Journal of Statistical Software*, vol. 65, no. 6, pp. 1–30, 2015. [Online]. Available: http://www.jstatsoft.org/v65/i06/

[20] M. Kuhn, S. Weston, N. Coulter, and M. C. C. code for C5.0 by R. Quinlan, *C50: C5.0 Decision Trees and Rule-Based Models*, 2015, r package version 0.1.0-24. [Online]. Available: http://CRAN.R-project.org/package=C50

[21] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: http://CRAN.R-project.org/doc/Rnews/

[22] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, "kernlab – an S4 package for kernel methods in R," *Journal of Statistical Software*, vol. 11, no. 9, pp. 1–20, 2004. [Online]. Available: http://www.jstatsoft.org/v11/i09/

P. Hurtik, M. Vajgl and N. Madrid. Enhancement of Night Movies Using Fuzzy Representation of Images. *IEEE World Congress on Computational Intelligence (IEEE WCCI)*, 2016, in press.

Enhancement of Night Movies Using Fuzzy Representation of Images

Petr Hurtik*, Marek Vajgl*, Nicolás Madrid‡
*Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, Centre of Excellence IT4Innovations
Czech Republic
petr.hurtik@osu.cz, marek.vajgl@osu.cz
†Departamento de Matemática Aplicada, University of Málaga
Spain
nmadrid@ctima.uma.es

***Abstract*—A cheap personal camera is a perfect device that allows us to test algorithms for video enhancement. In this paper we present a cascade of filters based on a fuzzy representation of images. This representation tries to capture the uncertainty underlying in the intensity of a pixel by means of a fuzzy set. The cascade of filters is compared with the corresponding standard filters: blurring, sharpening and image averaging. Besides, experimentally we show that our approach provides similar results with a significant reduction of the computational time.**

I. Introduction

In this paper we analyze videos obtained from dashboards cameras in order to develop appropriate algorithms in time and accuracy. We have chosen images obtained from cheap cameras and during the night to represent the necessity of incorporating control cameras in certain drastic circumstances (e.g., with bad light conditions, vibration, movements, etc).

The use of security and dashboard cameras has become popular in the last years due to the facility to acquire them (because of prices and offers) and because are increasingly required for insurance companies as evidence proofs. The quality of cheap cameras is very low, specially at night, when random (Gaussian) noise appears in the movie. Therefore, a good preprocessing of such videos require the elimination of such a noise. There exists many methods for denoising a movie (see for instance [1], [2], or [3]). Instead of using them singly, an usual procedure is to use them in a cascade [4] to filter the movie. In this work, we propose a cascade of filters to enhance a night movie recorded from a cheap dashboards camera.

The set of filters proposed are defined on a fuzzy representation of images. This fuzzy representation is based on the *visual salience* [5], that states that the color intensity of a pixel perceived by the human eye depends on its surrounding. This representation has shown to be useful to detect lanes in a road in real time [6] or to modify the size of an image [7]. In both works the computational speed is considered crucial. This paper extends our original work into the new proposed application and propose a way, how images equality can be measured and further aggregated.

The structure of the paper is the following. We start in Section II by recalling some standard filters in image processing. Then, in Section III we present our fuzzification procedure and the filters based on it. Subsequently, in Section IV we present some test to show that our approach improves the computational time of standards filters with comparable results. Finally in Section V some conclusions are given.

II. Movie improvement by using standard methods

In this section, we recall two kinds of filters commonly used in image and video processing for blurring and sharpening. But first things first, an image is formally defined as a function $f : D \to L$. The domain D determines the size of an image and their elements are called pixels. The domain is usually defined as a bounded set of the form $D = W \times H \subseteq \mathbb{N}^2$, where W and H are called the width and height of the image, respectively. The range L varies depending on the image considered so, if the image is black and white, $L = \{0, 1\}$; if it is a gray-scale image then $L = \{0, ..., m\}$ where m is the maximal considered intensity; and if it is a RGB color image, L can be considered as a subset of $\mathbb{N}^3$ where each triple (a, b, c) in it represents the intensity of red, green and blue of a pixel, respectively. Note that those possible ranges are only few examples of those that can be used to define images (e.g. Yuv, CMYK, or CIELab).

The goal of blurring an image is to remove its noise by making it "smoother". Usually these filters are defined by using masks and convolutions. A mask M of radius r is a matrix of size $(2r + 1) \times (2r + 1)$ where the position of its elements are given according to the relative position from the central element. So, elements of M are denoted by $M(i, j)$ with $-r \leq i \leq r$ and $-r \leq j \leq r$. Given a mask M of radius r and an image f, the blur of f by M is the result of the following convolution:

$$M \otimes f = \sum_{j=-r}^{r} \sum_{k=-r}^{r} M(j, k) f(x - j, y - k). \qquad (1)$$

The choice of the mask is relevant for the final result. As we expect Gaussian noise in our images, we use Gaussian distribution (see (2) in [8]) in M for suppressing this type of noise.

The opposite process to image blurring is called image sharpening. The process of sharpening should change intensities of pixels around edges but preserve homogeneous

areas. Therefore sharpening is usually connected to a gradient operator ∇f for detecting edges. Generally, we can sharp the image by using the following operation:

$$f(x,y) + \gamma \cdot \nabla f(x,y), \tag{2}$$

where $\gamma \in [0,1]$ is a constant controlling strength of the sharpening. The most commonly used gradient operators are Sobel, Prewitt and Laplace [9]. Due to the low computational complexity, the Laplace operator is often used for the sharpening. Note that the Laplace gradient operator can be written by using the convolutions associated to the mask

$$M_L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3}$$

In other words, $\nabla_L f(x,y) = M_L \otimes f$.

Using the previous two filters we can eliminate noise from the image and to sharp it. These two processes improve the quality of the image for human eyes. But in this approach we are considering movies, or equivalently, finite sequences of images (called frames), i.e., a movie is an ordered set $\mathbf{f} = \{f_1, ..., f_\ell\}$ where each f_i is the image. Therefore we can use specifics filters for movies. By assuming firstly that noise in one frame is independent from the noise in another frame and secondly, that two successive frames should be similar, we propose to aggregate preceding and succeeding frames for noise removal. Specifically, given a frame $f_i \in \mathbf{f}$ we consider k preceding frames and k succeeding frames. As objects in the movie change their position, a direct aggregation with the set above would blur the image and eliminate edges. By this reason for each pixel we consider only those images in the set $\{f_{i-k}, ..., f_i, ..., f_{i+k}\}$ such that $|f_j(x,y) - f_i(x,y)| < T$ for a given threshold T. We denote such a set by Ω_i. Thus we consider the aggregation:

$$\frac{1}{||\Omega_i||} \sum_{f_j \in \Omega_i} f_j(x,y). \tag{4}$$

This aggregation can be considered as a simple generalization of those aggregations defined on the unit interval $[0,1]$ [10].

From the filters defined above (blurring, sharpening and aggregation) we can define an algorithm for movie enhancing by using them in a sequence. Thus, we have:

In: Movie, sharpening strength, blurring mask, aggregation threshold.
Pre1: Decode movie into frames.
A1: Blur images using (1) and given blurring mask.
A2: Join images using (4) and given threshold.
A3: Blur or sharp the joined image pixels.
Post1: Encode filtered frames back to the movie.
Out: Filtered movie.

In the algorithm, the step *Pre1* denotes the preprocessing where a movie is converted into a set of frames. *Post1* denotes the postprocessing step where output frames are encoding into a movie. An example of an external library that may be used for decoding/encoding in steps *Pre1* and *Post1* is ffmpeg[1]. In step *A1* blurring suppresses potentially extreme noisy pixels. In step *A2* the aggregation improves the quality of the original input. In step *A3* the resulting frame is either blurred or sharpened to improve final human perception feeling. The decision is taken by using a gradient operator. Specifically, if $\nabla f(x,y)$ is low (so we can assume (x,y) is not in a edge) we apply again blurring. Otherwise (if $\nabla f(x,y)$ is high and then (x,y) is possibly an edge) a sharpening process is applied. Steps *A1* - *A3* are processed separately for each red, green and blue color component, if we consider RGB color model.

III. Filters based on Fuzzy representation of images

Most of the standard approaches consider the intensity of pixels without taking into account their surrounding. In this paper we consider the fuzzification of images presented in [7], [6]. This fuzzification is based on the idea that the gray intensity assigned to one pixel has an inherent uncertainty. This uncertainty can be due to different reasons, e.g.: each pixel represents an area that is not necessarily uniform and, the intensity assigned is then an average of the intensities in that area; or the focus of the image is not well adapted and the intensities of the surroundings interfere with the real intensity of the pixel; etc. At any rate, it is assumed that such uncertainty can be approximated by considering the surrounding pixels and by a triangular fuzzy set [11].

The consideration of *triangular fuzzy sets* instead of general ones provides an improvement of computational speed. This is because they can be identified with 3-tuple of numbers. Specifically, given $a, b, c \in \mathbb{N}$ such that $a \leq x \leq c$, we can define a triangular membership as:

$$A(x) = \begin{cases} \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ \frac{x-c}{b-c} & \text{if } b < x \leq c \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Note that, conversely, every triangular membership function can be associated with a triple of values.

An image, where each pixel value is represented by a triangular fuzzy set is called *Image Represented By a Fuzzy Function*, *Fuzzy Image*, or *IRFF* for short. Let us remark that the Fuzzy Image term is not new in literature. Usually, as in [12], the Fuzzy Image is defined as a mapping with range the unit interval $[0,1]$ and the fuzzification consists in a normalization. Note that our approach differs from this latter approach. By contrast, our approach is closely related to those named Interval Valued Images [13], [14] but with an essential difference: the consideration of a third value.

A. Fuzzification and Defuzzification of an image

The fuzzification procedure is defined as follows. To each pixel $(x,y) \in D$ of image $f\colon D \to L$ we assign a triangular fuzzy set $f^F(x,y)$ on the universe L, i.e., the universe is the set of intensities of the image. As we have said above, the

[1] http://www.ffmpeg.org

purpose of this fuzzy set is to represent a relationship between the intensity of one pixels and its surrounding. So, we consider *"neighborhood pixels"* by symmetric windows ω of length $\delta > 0$ around the pixel coordinate (x, y), i.e.:

$$\omega_{x,y} = \{f(x_i, y_i) \in L \mid \delta \geq |x - x_i|,\ \delta \geq |y - y_i|\}.$$

From $\omega_{x,y}$ we define the associated fuzzy set $f^F(x, y)$ as a triple:

$$(\min(\omega_{x,y}), f(x, y), \max(\omega_{x,y})).$$

For the sake of simplicity, we substitute $f(x, y)$ by $\text{cen}(\omega_{x,y})$ and the notation of the triplet as:

$$(\min(\omega_{x,y}), \text{cen}(\omega_{x,y}), \max(\omega_{x,y})).$$

The interval $[\min(\omega_{x,y}), \max(\omega_{x,y})]$ is called the support of the fuzzy set $f^F(x, y)$.

This representation of images can be related to a well known human eye behavior called *visual salience* [5]. For humans, images are perceived as complex structures where an interaction occurs between the intensities of different pixels. Specifically, the gray level perceived in one pixel depends on the pixels in its neighborhood.

The defuzzification procedure can be defined in various ways. One desirable property that can be required on them is that the iteration of fuzzification and defuzzification does not modify the original image. In this way perhaps the most natural defuzzification consists in taking the central element of the triple $(\min(\omega_{x,y}), \text{cen}(\omega_{x,y}), \max(\omega_{x,y}))$. But others procedures can be considered, for instance to take the greatest or least element in the 1-cut (note that these latter defuzzifications may require a normalization of $f^F(x, y)$.

B. Sharpening and blurring

A smooth image varies gradually and little by little in every direction. That mean that the fuzzy representation of a smooth image satisfies that the value of $f(x, y)$ is close to the center of the interval $[min(\omega_{x,y}), max(\omega_{x,y})]$. The converse also holds. That is, given an fuzzy image f^F, if we move the center $\text{cen}(\omega_{x,y})$ closer to $0.5(min(\omega_{x,y}) + max(\omega_{x,y}))$ for all $(x, y) \in D$ then, the image is blurred.

To allow degrees of blurring, we consider $\gamma \in [0, 1]$ and the distance $R_{x,y}$ between the center of gravity and the center

$$R_{x,y} = \text{cen}(\omega_{x,y}) - 0.5(min(\omega_{x,y}) + max(\omega_{x,y})),$$

from this is obvious that $R_{x,y} \in [-m/2, m/2]$, where m is maximal considered intensity. Thus, the fuzzy blurring of fuzzy image f^F is defined as the fuzzy image

$$b^F(x, y) = (min(\omega_{x,y}), cen(\omega_{x,y}) - \gamma R_{x,y}, max(\omega_{x,y})), \tag{6}$$

for all $(x, y) \in D$.

The case of considering sharp images is exactly the opposite case than the one for smooth images. In other words, the value $f(x, y)$ for a sharp image should be close of one of the extremes of the interval $[min(\omega_{x,y}), max(\omega_{x,y})]$. As above, the converse also holds. So we can sharp the image just by moving the center $\text{cen}(\omega_{x,y})$ to one of the borders of $[min(\omega_{x,y}), max(\omega_{x,y})]$. So, given $\gamma \in [0, 1]$, the fuzzy sharpened image of a fuzzy image f^F is defined as

$$s^F(x, y) = (min(\omega_{x,y}), cen(\omega_{x,y}) + \gamma R_{x,y}, max(\omega_{x,y})), \tag{7}$$

for all $(x, y) \in D$.

As in the case of blurring, the value $\gamma \in [0, 1]$ is a parameter that determines a degree of the sharpening. Thus, in both tasks, the greater the value γ, the more blurred/sharped is the image. Note that $\gamma = 0$ means no modification of the image in both cases.

C. Gradient magnitude

On the one hand, the use of gradients in image processing is important because they are able to measure the variability of intensities and then, to estimate where edges are. On the other hand, to measure the variability of intensities around one pixel we do not need to use convolutions or derivatives: it is enough to measure the difference of intensities in the surrounding of pixels. Note that in our approach it can be straightforwardly done by measuring the length of the support of $f^F(x, y)$. In other words, the fuzzy gradient of a fuzzy image f^F is defined as:

$$\nabla f^F(x, y) = \max(\omega_{x,y}) - \min(\omega_{x,y}). \tag{8}$$

Note that $\nabla f^F(x, y)$ is not a fuzzy image, but a standard one. Secondly, note that the greater the support, the grater the value of $\nabla f^F(x, y)$. Thirdly, this gradient coincides with the gradient commonly used in interval valued image processing [13], [14].

D. Equality measure

In section II we showed that by aggregating consecutive frames we can remove noise from movies. In the aggregation used we restricted to images with similar intensities, i.e., by considering only those images such that $|f_j(x, y) - f_i(x, y)| < T$ for a given threshold T. In the case of fuzzy images such a simple formula is inapplicable because the subtraction is not well defined in general between fuzzy sets. In the literature is common the use of similarity measures to compare two fuzzy sets [15], [16]. One of the most usual ways to define a similarity measure is by means of an implication $\to$ (i.e. an operator from $L \times L$ to L antitonic in the first component and monotonic in the second) and by defining:

$$a \leftrightarrow b = \inf\{a \to b, b \to a\}$$

It is worth to mention that the implication considered is usually residuated [11]. Then, given two fuzzy sets A and B, we define the measure of similarity:

$$E(A, B) = \inf_{v \in L} \{A(v) \leftrightarrow B(v)\}. \tag{9}$$

For the sake of computation complexity, we have chosen the Gödel residuated implication to define similarity measures. That is, the implication $\to_G$ is defined as

$$a \to_G b = \begin{cases} 1 & \text{if } b \geq a \\ b & \text{otherwise} \end{cases}$$

and thus, $\leftrightarrow$ is

$$a \leftrightarrow_G b = \begin{cases} 1 & \text{if } a = b \\ \inf\{a, b\} & \text{else} \end{cases}.$$

The computation advantage of considering the similarity measure associated to Gödel implication resides in checking just the cases where $E_G(f_1^F(x,y), f_2^F(x,y)) > 0$ for two fuzzy images $f_1^F(x,y)$ and $f_2^F(x,y)$. Such cases can be reduced to compare only the support of both fuzzy sets as follows:

$$E_G(f_1^F(x,y), f_2^F(x,y)) > 0 \text{ iff } \begin{cases} \max(\omega_{1,x,y}) \geq \min(\omega_{2,x,y}) \\ \text{AND} \\ \min(\omega_{1,x,y}) \leq \max(\omega_{2,x,y}) \end{cases}$$

E. Image aggregation

The aggregation of images is done in a similar way than in Section II. So we define a fuzzy movie as a finite sequences of fuzzy images, i.e., a fuzzy movie is an ordered set $\mathbf{f}^F = \{f_1^F, ..., f_\ell^F\}$ where each f_i^F is the fuzzy image. Then, given a $f_i^F \in \mathbf{f}$ we consider k preceding and k succeeding fuzzy images. Moreover, from such a set we retrieve only those fuzzy images f_j^F such that $E_G(f_1^F(x,y), f_2^F(x,y)) > 0$. We denote such a set by Ω_i^F.

Now, the aggregation is done component-wise in each triple $(\min(\omega_{x,y}), \text{cen}(\omega_{x,y}), \max(\omega_{x,y}))$. That is, the aggregation is the fuzzy image such that:

$$\begin{aligned} \min(\omega_{x,y}) &= \frac{1}{||\Omega_i^F||} \sum_{f_j^F \in \Omega_i^F} \min(\omega_{j,x,y}), \\ \text{cen}(\omega_{x,y}) &= \frac{1}{||\Omega_i^F||} \sum_{f_j^F \in \Omega_i^F} \text{cen}(\omega_{j,x,y}), \\ \max(\omega_{x,y}) &= \frac{1}{||\Omega_i^F||} \sum_{f_j^F \in \Omega_i^F} \max(\omega_{j,x,y}). \end{aligned} \tag{10}$$

To say the same with another words, image aggregation is element by element. I.e., for each image element (fuzzy number, instead of pixel) a set Ω_i^F of images satisfying element equality condition is taken and the elements are aggregated. The consequence is, each image element (fuzzy number) can be aggregated from different number of image elements.

F. Algorithm

Finally, we can define our fuzzy version of the algorithm given in Section II:

In: Movie, sharpening strength, blurring strength.
Pre1: Decode movie into frames.
Pre2: Convert frames into Fuzzy Images.
A1: Blur Fuzzy Images using (6).
A2: Aggregate Fuzzy Images using (10).
A3: Blur / sharp Fuzzy Images sets using (6) / (7).
Post1: Convert Fuzzy Images into standard image representation.
Post2: Encode filtered frames back to the movie.
Out: Filtered movie.

The algorithm is basically the same than in Section II, actually the decoding/encoding in steps *Pre1* and *Post2* uses the same library and the decision in step *A3* is done in the same way, but by considering the gradient operator given in equation (8). The main difference is in steps $Pre2$ and $Post2$ where the fuzzification and defuzzification are applied. It is important to mention that the fuzzification can be done by computing several fuzzy sets at the same time. This reduces significantly the computational time since it allows to reduce at least 50% the number of comparisons (see [6] for more details). Note: As same as in the case of standard processing, the steps *Pre2* - *Post1* has to be processed separately for red, green and blue color components (if we consider RGB model).

IV. Experiments

To test both algorithms, we have used two different short real-life movies (530 and 690 frames). Figure 1 shows two screenshots from these two movies. The frame resolution in both cases is 640×480px. In order to perform a fair comparison between the standard and fuzzy version, we have used in the implementation the same programming language, coding style, compiler and have been tested on the same low-powered notebook without parallelism.

The average computation time is 204ms per frame for the standard version and 142ms for the fuzzy version. The computation time includes reading images, processing and saving them into disc. The movie-to-frame coding/decoding operation is not included, because it depends on the used movie compression. The reason that the fuzzy version takes 70% computation time of the standard approach can be explained as follows:

- On the one hand, the fuzzy algorithm requires two more steps than the standard one. However these two processes do not require excessive computational cost. In fact, defuzzification is done just by taking the element $\text{cen}(\omega_{x,y})$ in $f^F(x,y)$. Therefore defuzzification takes almost no computation time. In the case of the fuzzification, as is based on comparisons, the computation can be done by computing at the same time several fuzzy sets for different pixels. A detailed explanation of this computation can be found in [6]. Therefore, the computational cost of fuzzification and defuzzification is acceptable.
- On the other hand, once the fuzzification is applied, fuzzy filters only require three values per pixel in decrement of the 9 or 25 elements (depending on the size of the mask) required by standard approaches based in convolutions. Moreover, in any filter-based convolution all the elements in the mask must be operated by multiplications and sums, whereas fuzzy blurring and sharpening only require one product and one sum. This fact makes much faster the filters based on fuzzy images than those based on convolutions.

In summary, the speed of the execution of filters highly makes up for the computational time spent in the fuzzification.

Fig. 1. Example screens of the two movies.

The comparison of computation times for the three basic operations (blurring, sharpening and gradient magnitude computation) for masks of sizes 3×3 and 5×5 pixels is shown in Table I. In the table we denote to Image Represented by a Fuzzy Function as IRFF. To offer a good comparison we show two cases of IRFF: the first one shows computation time of a particular operation only. The second case includes computation time of fuzzification and defuzzification procedures as well.

Figure 2 shows outputs of one of the movies used for testing. The top row shows one randomly selected frame of the processed movie. The second and the third row contains three columns with a zoom on the red rectangle drawn in the top image:

- the left column refers to the original (unprocessed) image,
- the middle column refers to the image processed by standard algorithm described in II,
- the right column refers to the image processed by newly proposed algorithm based on Fuzzy Images described in III.

TABLE I
COMPUTATION TIMES [MS]

Operation	3×3	5×5
Blurring		
Gauss core	29	70
IRFF blur only	5	5
IRFF with FUZZ and DEF	21	26
Sharpening		
Laplace	24	44
IRFF sharp only	5	5
IRFF with FUZZ and DEF	19	27
Gradient magnitude		
Sobel	40	118
IRFF gradient only	3	3
IRFF with FUZZ and DEF	22	24

Second row shows simply the output of the cut-out zoomed part after the execution of its respective algorithm. Third row illustrates the difference between the movie frame shown and the following one. Dark values mean low difference between the intensity of pixels in different frames whereas lighter pixels represent a high difference. Let us remark that images in the third row have been enhanced to emphasize the results. It can be observed that both, the standard and the fuzzy algorithms filter the noise significantly (the areas are darker in the second and third column than in the first one) while the moving object (car in the top right corner of cropped image) is preserved. The original movies can be downloaded from [2] and [3] with its corresponding version of standard methods filtering [4] and [5] and fuzzy methods filtering [6] and [7].

V. CONCLUSION

In this paper we deal with the problem of denoising and enhancing movies captured by a dashboard camera. On the one hand, the movie was decoded into frames and then processed using standard computer graphic methods.

On the other hand, we have recalled a novel representation of images by using fuzzy set. Specifically we have assigned to each pixel a fuzzy set to represents the uncertainty of its intensity. We have shown that information codified by such fuzzy sets can be used for image blurring, sharpening and gradient magnitude. Moreover, we have shown that the computational speed of these filters is quite good since the use of convolutions is no required. Furthermore this paper extends our original approach by a way how an image equality can be measured. The equality measure is used in the second extension of our original work, image aggregation.

In our tests, we have presented results applied on two movies. *The outputs of standard and fuzzy algorithms are*

[2] http://graphicwg.irafm.osu.cz/movies/v.wmv
[3] http://graphicwg.irafm.osu.cz/movies/r.wmv
[4] http://graphicwg.irafm.osu.cz/movies/out-v-s.mp4
[5] http://graphicwg.irafm.osu.cz/movies/out-r-s.mp4
[6] http://graphicwg.irafm.osu.cz/movies/out-v-f.mp4
[7] http://graphicwg.irafm.osu.cz/movies/out-r-f.mp4

Fig. 2. Output of filtering. Top: original movie frame. Middle from left: original, standard algorithm output, fuzzy algorithm output. Bottom images show difference with movie frame and preceding frame with the same ordering as middle ones.

comparable, subjectively of the same-quality and both significantly reduce noise from the original movie. However, the approach based on the fuzzy functions representation is faster and therefore can substitute commonly used methods. We show that the fuzzy enhancing algorithm needs only 70% computation time of the standard algorithm.

The application can be freely downloaded from url[8]. The movies and their filtered version using both algorithms (standard one and the newly proposed) are available at given web addresses.

We conclude the paper with claim that the proposed Fuzzy Image can be propagated into other algorithms (color enhancing etc.) with possible benefit of increasing computation speed.

[8]http://graphicwg.irafm.osu.cz/storage/movie_denoise.zip

ACKNOWLEDGMENT

This research was partially supported by the NPU II project LQ1602 "IT4Innovations excellence in science" provided by MŠMT.

REFERENCES

[1] A. Buades, B. Coll, and J.-M. Morel, "Nonlocal image and movie denoising," *International journal of computer vision*, vol. 76, no. 2, pp. 123–139, 2008.

[2] S. Esakkirajan, T. Veerakumar, A. N. Subramanyam, and P. C. Chand, "Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter," *Signal Processing Letters, IEEE*, vol. 18, no. 5, pp. 287–290, 2011.

[3] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[4] M. P. H. Yawalkar and M. P. Pusdekar, "A review on low light video enhancement using image processing technique."

[5] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[6] N. Madrid and P. Hurtik, "Lane departure warning for mobile devices based on a fuzzy representation of image," *Fuzzy Sets and Systems*, 2015.

[7] P. Hurtik and N. Madrid, "Bilinear interpolation over fuzzified images: enlargement," in *The 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015)*, 2015.

[8] M. Basu, "Gaussian-based edge-detection methods-a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 32, no. 3, pp. 252–260, 2002.

[9] T. Ma, L. Li, S. Ji, X. Wang, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "Optimized laplacian image sharpening algorithm based on graphic processing unit," *Physica A: Statistical Mechanics and its Applications*, vol. 416, pp. 400–410, 2014.

[10] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation functions: A guide for practitioners*. Springer, 2007, vol. 221.

[11] V. Novák, I. Perfilieva, and J. Mockor, *Mathematical principles of fuzzy logic*. Springer Science & Business Media, 2012, vol. 517.

[12] H. R. Tizhoosh, "Fuzzy image processing: potentials and state of the art," in *IIZUKA*, vol. 98, 1998, pp. 16–20.

[13] A. Jurio, D. Paternain, Lopez-Molina, H. Bustince, R. Mesiar, and G. Beliakov, "A construction method of interval-valued fuzzy sets for image processing," in *Advances in Type-2 Fuzzy Logic Systems (T2FUZZ)*, 2011, pp. 16–22.

[14] H. Bustince, E. Barrenechea, M. Pagola, and J. Fernandez, "Interval-valued fuzzy sets constructed from matrices: Application to edge detection," *Fuzzy Sets and Systems*, vol. 160, no. 13, pp. 1819–1840, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.fss.2008.08.005

[15] V. Novák, P. Hurtík, H. Habiballa, and M. Štepnička, "Recognition of damaged letters based on mathematical fuzzy logic analysis," *Journal of Applied Logic*, vol. 13, no. 2, pp. 94–104, 2015.

[16] N. Madrid, M. Ojeda-Aciego, and I. Perfilieva, "f-inclusion indexes between fuzzy sets," in *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15), Gijón, Spain., June 30, 2015.*, 2015.