Mojette Transform FPGA Implementation and Analysis

József Vásárhelyi[1], Péter Szoboszlai[2], Jan Turán[3] Péter Serfözö[4]

[1] University of Miskolc, Department of Automation, Miskolc, Hungary, vajo@mazsola.iit.uni-miskolc.hu,
[2] Magyar Telekom NyRt., Budapest, Hungary szoboszlai.peter@telekom.hu
[3] Technical University of Kosice, Department of Electronics and Multimedia Communications jan.turan@tuke.sk
[4] Ericsson Hungary Ltd., Budapest, Hungary serfozo@ericsson.com

Abstract: In the last ten years there were intensive researches to find new methods for handling internet image processing and distributed databases. One of the many methods is the Mojette transform (MOT). The MOT is used mainly in image processing applications but can be used also for distributed databases also. There were several implementation using personal computer, but these application were implemented only for locally stored images. The MOT implemented under different operating systems and for different processors presented in helps the performance analyses of the MOT and also let to conclude the need for reconfigurable hardware implementation. There is also analyzed the implementation of Mojette transform and Inverse Mojette transform (IMOT) the implementation in Field Programmable Gate Arrays using reconfigurable platform. The paper tries to conclude the necessity for hardware implementation for real time processing. The paper outline the development work in order to create an embedded reconfigurable hardware based on FPGA board.

Keywords: distributed databases, Mojette transform, FPGA, embedded systems

1 Introduction

The “word” Mojette comes from France (Poitiers), where - in old French - it describes the class of white beans. These white beans where used to teach children to start computing basics of arithmetic (addition and subtraction). They were also used for computing the number of victories for card games like „Aluette” still played with middle age Spanish cards. Mojette is the name of the transform to remember first that when only adds are invoked, the computations can be easily made, second that by sharing the information pot each player will get a part of it. Guédon named the transform Mojette after the analogy of beans and bins. The *bins* contain the sum of pixel values of the respective projection line [8.].

Inscribing invisible marks (watermarking) into an image has different applications such as copyright, steganography or data integrity checking. Several different techniques have been employed for the last years in different spaces (Fourier, wavelet, Mojette domains, etc.) [1-10]. The applications come from different field such as computer tomography, internet distributed data bases [3.], encoding, multimedia error correction [9.], etc.

Until today the aim of the researches was software implementation of the Mojette and inverse Mojette transform. This kind of implementation can be used only on still images, because they can not process the data in real-time. Motion pictures, video streams and distributed data bases require run-time processing of the frames. Only specialized hardware or embedded systems with real-time operating systems can ensure this.

This paper presents one implementation method to implement the Mojette transform in Field Programmable Gate Array (FPGA).

1 Mojette Transform

The Mojette transform is published at first time in 1995 0. The Mojette transform is an exact discrete Radon transform. The Mojette Transform (MoT) has been introduced and used for the last thirteen years [8.]. The main idea behind the Mojette Transform (similarly to the Radon transform) is to calculate a group of projections on an image block. Since the Mojette transform is appeared in image processing several variations and applications are published. This section describes the Mojette transform with the different versions and applications and gives an introspection into the current state of the "art".

1.1.1 Direct Mojette Transform

The Mojette Transform has two basic types. The MoT on integers [2.], [3.] [5.] and the MoT on binary images [12.]. At first let's speak about the MoT on integers. The continuous Radon transform was given by

$$R_f(x,y) = p_\Theta(t) = \\ = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\delta(t - x\cos\Theta + y\sin\Theta)dxdy, \qquad (1)$$

When a discrete pixel grid is considered, the function *f(x,y)* in (1) is replaced by *f(k, l)*. Since functions *cosΘ* and *sinΘ* are giving pure real values, (k* *cosΘ* + l* *sinΘ)* is of elliptical form and the possibility to equally sample variable t is to use of the form *tanΘ=p/q*. To avoid ambiguities, only integer couples (p,q) with

GCD(p,q)=1 give acceptable angles (i.e. exact discrete angles). Thus, the MoT denoted M is described by

$$Mf(k,l) = proj(p,q,m) = \\ = \sum_{k}\sum_{l} f(k,l)\Delta(m - qk + pl), \tag{2}$$

where Δ is the Kronecker function. This is a generic definition of the MoT showing the linearity of the transform. For a given (p, q) corresponding to the angle *θ*, the value of the *bin m* is simply the sum of the pixels which centers are on the line defined by:

$$m = ql - pl \quad and \quad \Delta(m) = \begin{cases} 1\ if\ m = 0 \\ 0\ if\ m \neq 0 \end{cases}, \tag{3}$$

For an image block *{f(k,l), k = 1, 2, ..., K, l = 1, 2, ..., L}* and a group of different angles θ_i which are so chosen that $tg(\theta_i) = q_i/p_i$, where q_i and p_i are two mutually prime integers, every projection of index *i* is a vector of various size element. This element called *bin* and it is equal to the sum of values of pixels along the straight line corresponding to θ_i (Figure. **1).** The major difference with the classical Radon transform is that the *bin* spacing on a projection is a function of the projection angle θ_i (it is different for all projections).

A direct consequence of this sampling is that the number of bins of the projection indexed by (p, q) depends on both the projection angle and the shape of the region (the block dimensions).

For a projection defined by θ_i, the number of bins n_i can be calculated by

$$n_i = (K-1)|q_i| + (L-1)|p_i| + 1, \tag{4}$$

Therefore for one group of projections (p_i, q_i), i = 1, 2, …, *I* the number of bins is:

$$N_b = (K-1)\sum_{i=1}^{I}|q_i| + (L-1)\sum_{i=1}^{I}|p_i| + 1, \tag{5}$$

where *I* is the number of projections. The order of complexity of the direct MoT is obviously *O*(*IN*) i.e. linear in the number of pixels N_p and linear in the number of projections l. When N_b is greater than the number of image pixels $N_p(N_p = KL)$, the MoT corresponding to this group of projections results in a data redundancy:

$$R = \frac{N_b}{N}, \tag{6}$$

1.1.2 Inverse Mojette Transform

Computing Inverse Mojette Transform (IMoT), bins are back-projected. A single pixel-*bin* correspondence must be found at each iteration cycle in order to reconstruct a pixel value. When it has been done, this pixel value is substituted in the adequate coordinate of a blank image and subtracted from the corresponding bins in each projection of the original image. To find this single pixel-*bin* correspondence examination of the projections of unary image is necessary. The *bin* values in those projections show the number of pixels which are stored in the given *bin*. Therefore the modification of the unary projections has to be done parallel. The IMoT is iterating this process until the image is completely reconstructed. For the reconstruction both projection sets needed (one is the MoT of the image and the other is the MoT of a unary image where each pixel value is 1). Figure. 2 shows the first step of the reconstruction process.

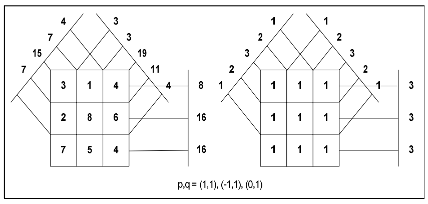

Figure. 1. Mojette Transform on integers

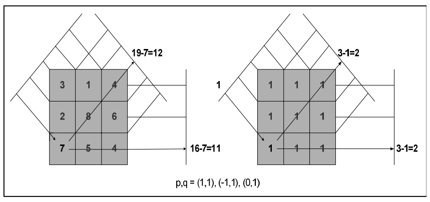

Figure. 2. Inverse Mojette Transform – first step

1.1.3 Condition of Reconstruction

The key of the transform with regard to the controlled redundancy is that the number of computed projections can be larger than we need for the reconstruction (inverse transform). Thus, we can control a first step of redundancy with the number of projections. The question is then to know how many projections and which projections give an adequate set to reconstruct the image (i.e. compute the Inverse Mojette Transform). The result for rectangular image was given by Katz in [4.].

According to Katz's Lemma [4.] an image of dimension KxL can be reconstructed with the *Mf* set of directions (p_i, q_i), i=1, 2, …, I if

$$K \le P_i = \sum_{i=1}^{I} |p_i| \quad or \quad L \le Q_i = \sum_{i=1}^{I} |q_i|, \tag{7}$$

where K and L are the dimensions of the image p_i the horizontal projection coordinate (x axis) and q_i the vertical projection coordinate (y axis).

The condition of reconstruction ensures that the number of *bins* (N_b) is greater than the number of pixels (N_p) therefore the number of equations will be larger than the number of unknown variables during the reconstruction process. At the same time the condition of reconstruction answers the question whether is any projection in the projection set which is unnecessary to reconstruct the original image.

MoT for binary image of simply binary data is defined with the operations „AND" and „XOR" respectively instead of the multiplication „*" and addition „+" [12.]

3 Low Power – Low Complexity Hw Design Methodology For Selected Algorithm- MOT AND IMOT

Power consumption is a very important question of our days. Many energy saving technologies and techniques were developed in different fields. The spread of portable computers and communication devices inspires the developers to design low power consumption devices (CPUs, storage devices, etc.). In the computer technology there are several options when low power consumption devices are needed (DSPs, μPs). One of these options is using field programmable gate arrays (FPGAs).

The "MOTIMOT" co-processor denotes the hardware implementation of the direct and inverse Mojette transform as co-processing elements of an embedded processor.

To construct a new hardware for a special task is difficult and expensive both in time and costs. Estimating the hardware and software needs to implement a MOTIMOT processing system there is necessary to map the tasks what such a system should implement. These tasks are not limited to the direct and inverse Mojette transformation, but also posses embedded computer functions with or without (depends on the application) real time operating system kernel. Figure 3 shows the functional block scheme of the MOTIMOT co-processor. Both of the Virtex II PRO and Virtex IV FPGAs (FPGAs used for implementation) contain an embedded powerPC 405 RISC processor (PPC). This general purpose processor can manage the MOTIMOT hardware, its driver and the incoming and outgoing data. The MOT and IMOT blocks are connected to the PPC via the processor local bus (PLB).

Probably calculation of MOT or IMOT of an image is not necessary in the same time, therefore only one of the configuration files is loaded into the device. At this point we can use the advantage of partial run-time reconfiguration.

The images are received in real-time thru an Ethernet connection or from a digital camera connected to the platform via an USB port, after processing the image or the frames these are returned to the sender or sent to a client (PC, PDA, etc.). On the motherboard there is an external onboard memory with the size of 512 MB. In our days this resource is available on almost any mobile devices. There are several types of cellular phones which contain place for a micro secure digital card.

3.1 Low Complexity System Design for Stepping Window Method

Figure 4 shows the symbolic logic circuitry of MOT on a 4x4 image. The input registers ($IR_{x,y}$, x = 1, .., 4; y = 1, .., 4) contain the values of pixels of the image while the output registers ($OR_{i,j}$, i = 1, .., 3; j = 1, .., max_*bin*_number$_i$) contain the values of bins. Three projection lines (1,1; -1,1; 1,2) gives an adequate set of bins and the set of projections meets the reconstruction criteria. The Mojette operator is the XOR logic operator and the different colour (greenscale) of the line means separate the three projection lines and their bins. This figure (Figure 4) represents the MOT of a 4x4 pixel size image. This size do not meet with real image sizes of coarse, even so the 4x4 pixel size window is usable in the stepping window method. The register size depends on the width of input data (byte, word, double word) but notice that the larger the data width the more hardware resources are required.

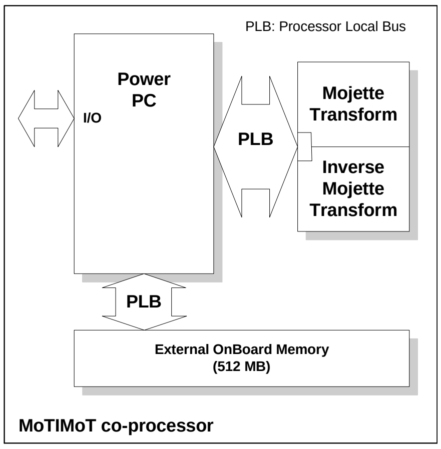

Figure 3. Block schema of MOTIMOT co-processor

The symbolized logic circuit of the reconstruction (IMOT) is depicted in

Figure 5. The pixels can calculated in two ways. At first every pixel value calculable from *bin* values only and secondly a pixel value is calculable from *bin* values and the already calculated pixel values. The second version gives more simple (low-complexity) hardware. In

Figure 5 the input registers (IR) contain the *bin*-values, the output registers (OR) contain the original pixel-values, while the IMOT operator is the XOR as in the MOT was. The number of registers is the same in both cases (MOT and IMOT). As a matter of fact the number of input registers can be smaller then the number of bins because of the redundancy.

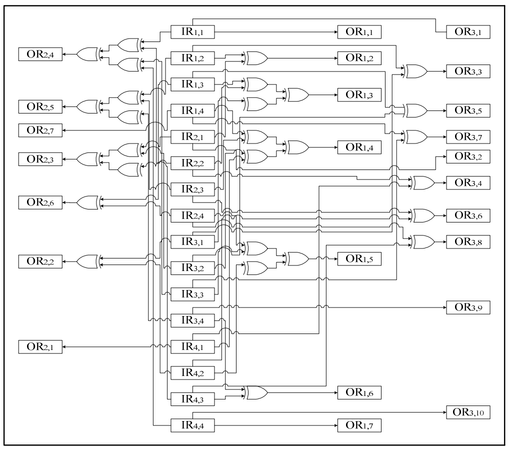

Figure 4. Logical model of the MOT on a 4x4 image

The single pixel-*bin* correspondences (most of projection lines contain some) give the original pixel value without any calculation this is called zero level. Other bins contain more pixel values. The number of XOR operations need to be performed on them to get the original pixel value is the number of its level. In this case there are five levels from zero to four. In software solution it means a cycle ("for") from zero to five and every cycle contains another cycle ("for") from one to n_{bl} where n_{bl} is the number of bins on the same level. If the window is chosen larger the number of level will increase with it. Using the already calculated pixel values the complexity of the whole system will increase also of course. However increasing of complexity will be slower then when only the *bin* values are used in the calculations.

Definitely there is a window size limit (WSL) for every FPGA chip. The WSL is the block size what the given FPGA can process in parallel. The WSL depends on the number of logic cells in the given chip and other resources. To enlarge the window size above to the WSL more FPGA chips must be applied. Next figure (Figure 6) shows a MOT/IMOT computing system. The data source means image or other types of data pre-processing before the MOT is not necessary. Post-processing after the MOT can be any kind of lossless compression method to decrease the redundancy. Data sink can be a storage device (eg.: PC). In the decoding process the data source is the file of projection lines. Pre-processing is necessary (uncompress the files). Post processing is not necessary here. The data sink in this case is the user's application.

As it is shown in Figure 6 when larger window size must chosen then one need to multiply the WSL hardware. The sub-picture limited by the broken line shows a multi-chip MOT/IMOT system. The distribution and merge of data (by the multiplexer and demultiplexer units) also managed by an FPGA chip or if the hardware resources of the FPGA chip allows it, can managed by the on-chip general purpose processor (eg.: the PPC in the Xilinx Virtex II Pro FPGA). A four FPGA solution of the stepping window method for MOT is depicted in Figure 7. The sub-window size is fit to the possibilities of the given FPGA and can not be larger then the WSL.

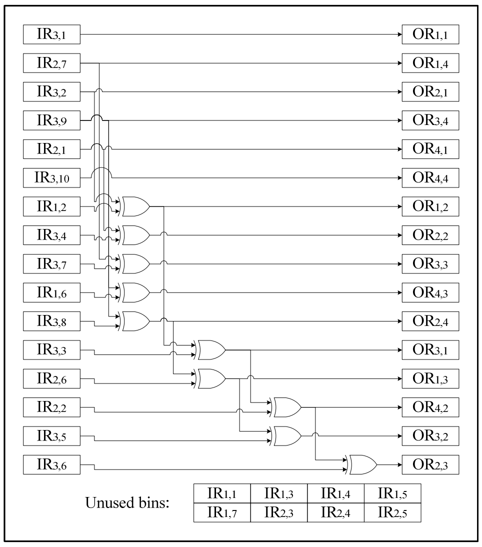

Figure 5. Symbolized logic circuitry of a 4x4 size image reconstruction

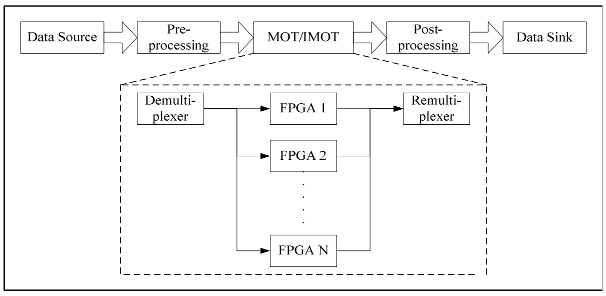

Figure 6. MOT/IMOT system block-schema

With this method the window size (area) equal to the window size limit (area) multiplied by four (if n FPGAs are in use of course the WSL will be multiplied by n). It results shorter computing time but larger power consumption. To find the balance between the power consumption and speed, it is necessary to known where the algorithm of the Mojette transform will be used.

3.1.1 Low Complexity System Design for Sliding Window Method

The sliding window method differs to the stepping window method in its basic. At the stepping window method as it is suggested by its name, there are no common pixels of the windows in two neighbor steps. Contrarily the sliding window method moves the window only with one row or column (depends on the direction of processing) forward. It means most of the pixels are common in two windows, which are neighbors of each other.

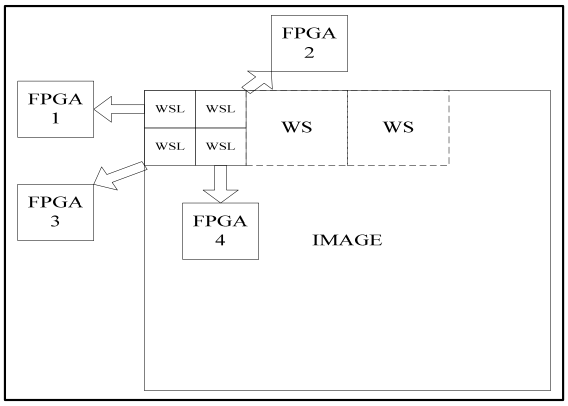

Figure 7. Stepping window method with four FPGAs

Another difference compared to the stepping window method that while at the stepping window method the MOT/IMOT computing is one single step, the MOT/IMOT computing has two parts at the sliding window method. First part is to calculate the final value of the given *bins*/pixels and calculate the next temporal value of the other *bins*/pixels in the window and second part is to move the data: write out the final values, move the temporal values into the corresponding registers and read in new data.

Figure 8 shows the symbolic logic circuitry of the sliding window method where $p_i = \{1, -1, 3, -3\}$ and $q_i = 1$. The virtual image size is defined by P and Q where Q = 4 and P = file size/Q. This means that the size of sliding window is chosen independently the file size. The size of sliding window is given by the hardware resources (number of logic circuits, memory, etc.) of the given FPGA.

Note if q is chosen larger (q=2) the logic circuitry and the size of sliding window (number of registers) will be multiplied by the new q value. When larger window size is required then the WSL, more FPGAs can be used. Four FPGAs using system is depicted in Figure 9. Each of the FPGAs contain the same logic circuitry and the size of WSL is the same but the virtual WSL of the whole system is four times larger then the WSL of an individual FPGA. Every FPGA processes a four word width data stream transparently, so the whole system processes a sixteen word width data stream. The projection coordinate q is equal to one in every projection of every FPGA, therefore the corresponding projection lines are merged into one line. Four FPGAs and four projection lines give sixteen projection lines. The merge of the mentioned projections will result four *bin* vectors where the resultant q coordinate is equal to four ($q_r = 4$). This way the permanently processed data width is multiplied by four. The IMOT computing system will reconstruct the original data whether it contains only one FPGA (a larger one) or four FPGAs.

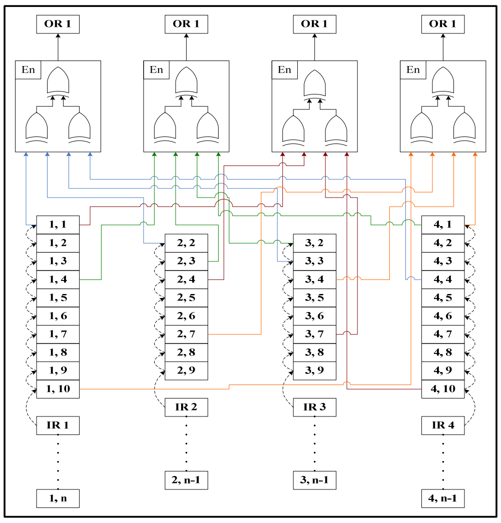

Figure 8. Symbolic logic circuitry for sliding window method

Figure 10 shows the merge process after the MOT computing. Every FPGA has four outputs (four *bin* vectors). As it is depicted in the above mentioned figure the corresponding *bin* vectors of the FPGAs are merged into one *bin* vector.

3.1.2 IMOT

The logic circuitry of the IMOT compared with the MOT shows that the reconstruction process is more complicated. There are single-pixel *bin* correspondences which result pixel values very simply but the bins of every other projection line need correction. It is necessary, because the *bin* value corrections generate new single-pixel *bin* correspondences and ensures the continuity of the reconstruction process.

The symbolic logic circuitry of the IMOT computing SLW co-processor is depicted in Figure 11. The picture does not show the total system only a part of it. In the image Pi (i=1..8) means the reconstructed pixel values, PLi (i=1..4) are the projection lines, while the rectangles above them represents the bins of the projection lines. In the image "tr" means temporary register which are necessary for the second step, the *bin* value correction. The outputs of Unit 1 are the reconstructed pixel values and it has an enable input. Unit 2 gives the *bin* values after the correction and works with the same enable signal as the Unit 1. In the image (Figure 11) the bit correction unit is depicted only for one projection line but the other three units are very similar. After the *bin* correction the new values are stored in temporary registers.

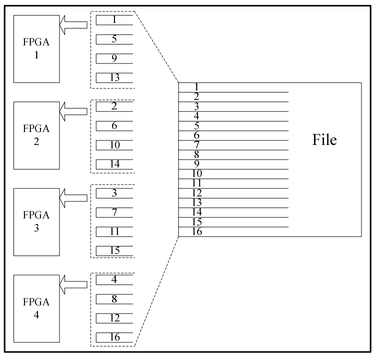

Figure 9. Sliding Window Method with four FPGAs

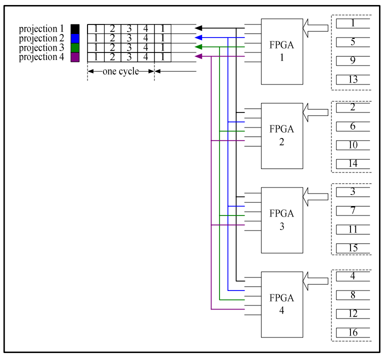

Figure 10. Merge process

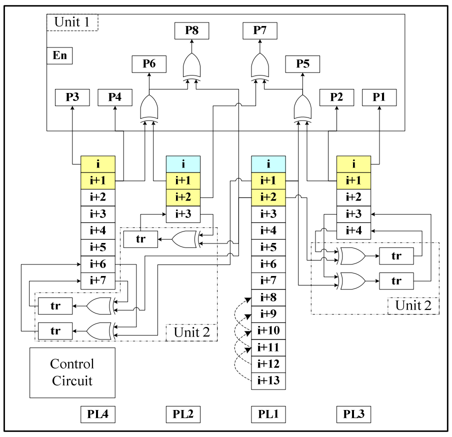

Figure 11. Symbolic logic circuitry for IMOT (SLW)

The MOT and IMOT functions are implemented as separate hardware co-processors of the main processor. This is possible in two ways. The main processor can be an off-chip or an on-chip solution. The implemented algorithms are realized as separate co-processors and they work either in parallel or using run-time reconfiguration (This method was not tested yet) using relatively low working frequencies (100-300MHz). This way can be obtained very high processing speeds.

Conclusions

There was presented the hardware implementation of MOT and IMOT in FPGAs using parallel implementation of the Mojette algorithm.

For complex 256x256 size images the MOT/IMOT can be implemented using multiple FPGA chips.

Acknowledgement

This work was partially supported from the grants VEGA No. 1/0034/09, project COST ICO802 and Centrum of Excellence FEI TUKE.

References

[1.] Turán, J. Fast Translation Invariant Transforms and Their Applications. ELFA, Kosice, 1999.

[2.] Normand, N., Guedon, J. P.: La transformee Mojette: une representation recordante pour l'image, Comptes Rendus Academie des Sciences de Paris, Theoretical Comp. SCI. Section, 1998, pp. 124 – 127

[3.] Guedon, J. P., Parrein, B., Normand, N.: *Internet Distributed Image Databases*. Int. Comp. Aided Eng., Vol. 8, 2001, pp. 205 – 214

[4.] Katz, M.: Questions of uniqueness and resolution in reconstruction from projections. Lecture Notes in Biomathematics, (Vol. 26), Springer-Verlag, Berlin, 1979

[5.] Autrusseau, F., Guedon, J. P.: Image Watermarking for Copyright Protection and Data Hiding via the Mojette Transform, Proceedings of SPIE, VOL. 4675, 2002, pp. 378 – 386

[6.] Turán, J., Ovsenik, L., Benca, M., Turán, J. Jr: Implementation of CT and IHT Processors for invariant Object Recognition System, Radioengineering, Vol. 13, No 4 2004. dec. page 65-71

[7.] Autrusseau F.: Modélisation psychovisuelle pour le tatouage des images, Ph. D. Dissertation, Nantes, France, 2002., pp.

[8.] Guédon J-P., Normand N.**:** The Mojette Transform: The First Ten Years, Proceedings of DGCI 2005, LNCS 3429, 2005, pp. 79-91

[9.] Parrein B., Normand N., Guédon J-P.**:** Multimedia Forward Error Correcting Codes For Wireless Lan, Annals of Telecommunications (3-4) 448-463 March-April, 2003

[10.] P. Serfőző, J. Vásárhelyi, Analysis of Mojette Transform Implementation on Field Programmable Gate Array, Proceedings of the 7th International Symposium of Hugarian Researchers on Computational Intelligence, Budapest, 2006. pp. 255-267, ISBN 963-7154-54-X

[11.] P. Serfőző, J. Vásárhelyi Szoboszlai, P.; Turan, J.; Performance requirements of the Mojette transform for internet distributed databases and image processing, IEEE OPTIM 2008. 11th International Conference on 22-24 May 2008 Page(s):87 - 92

[12.] Wu,J.-Olivier,C.-Chatellier,C.-Poussard,A-M.: Redundant Transformation on the Binary Data and Its Applications to Image Coding and Transmisssion, IEEE APCCAS 2000, Dec 2000, 763-766