Analysis of Mojette Transform Implementation on Reconfigurable Hardware

József Vásárhelyi, Péter Serfőző
Department of Automation,
University of Miskolc,
Miskolc-Egyetemváros, Miskolc, Hungary,
{vajo, serfozo1}@mazsola.iit.uni-miskolc.hu

Abstract: **In the last ten years there were intensive researches to find new methods for image processing. One of these methods is the Mojette transform (MOT). This transform is an exact discrete Radon transform defined for a set of specific projections. The MOT is used mainly in image processing applications. There were several implementation using personal computers [8] and [9]. There are some applications where real time computing is a must, but the implementations before referred can not accomplish this requirement. There is analysed the Mojette algorithm implementation for images of 256x256 pixels. There is also analysed the implementation of Mojette transform and Inverse Mojette transform (IMOT) in Field Programmable Gate Arrays using reconfigurable platform. The paper tries to outline the development work in order to create an embedded reconfigurable hardware based on Xilinx ML310 board [13].**

Index Terms: **Field Programmable Gate Arrays (FPGAs) image processing, Mojette transform, embedded systems, run-time reconfiguration.**

I. 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). 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 [10]. Inscribing invisible marks (watermarking) into an image has different applications such as copyright, steganography or data integrity checking. Many 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 [11], internet distributed data bases [4], encoding, multimedia error correction [12], 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 and video streams require run-time processing of the frames. Only specialized hardware or embedded systems with real-time operating systems can ensure this.

II. DIRECT MOJETTE TRANSFORM

The main idea behind the Mojette transformation (similarly to the Radon transformation) is to calculate a group of projections on an image block. [3].

The Mojette transform (MOT) (see [5], [6] and [7]) projects the original digital 2D image:

$$F = \{ F(i,j);\ i = 1, \ldots, N;\ j = 1, \ldots, M\} \qquad (1.)$$

onto a set of K discrete 1D projections with:

$$M = \{ M_k(l);\ k = 1, \ldots, K;\ l = 1, \ldots, l_K \} \qquad (2.)$$

MOT is an exact discrete Radon transform defined for a set $S = \{(p_k, q_k),\ k = 1,\ldots, K\}$ specific projections angles:

$$M_K(l) = proj(p_k, q_k, b_l) = \\ = \sum_{(i,j)\in L} F(i,j)\delta(b_l - iq_k - jp_k) \qquad (3.)$$

where proj (p_k, q_k, b_l) defines the projection lines p_k, q_k; $\delta(x)$ is the Dirac delta with the form:

$$\delta(x) = \begin{cases} 1, if\_x = 0 \\ 0, if\_x = 1 \end{cases} \qquad (4.)$$

and

$$L = \{(i,j); b_l - iq_k - jp_k = 0\} \qquad (5.)$$

is a digital bin in the direction θ_k and on set b_l.

So the projection operator sums up all pixels value which centres are intersected by the discrete projection line *l*. The restriction of angle θ_k leads both to a different sampling and to a different number of bins in each projection (p_k, q_k).

Dagstuhl Seminar Proceedings 06141
Dynamically Reconfigurable Architectures
http://drops.dagstuhl.de/opus/volltexte/2006/746

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

$$n_i = (N-1)\left|p_i\right| + (M-1)\left|q_i\right| + 1 \qquad (6.)$$

The direct MOT is depicted in Fig. 1 for a 3x3 pixel image and for 3x3 pixel unary image. The set of three directions *S = {(-1, 1), (1, 1), (1, 0)}*, giving the thirteen bins for integer valued (*.pgm files – Portable Grey Map format file with 256 grey tones) and unary image.

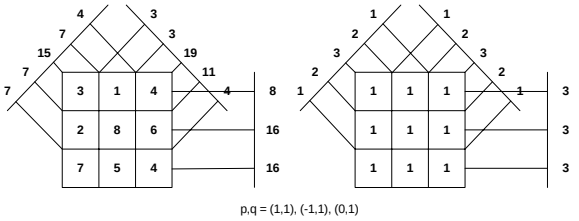

Fig. 1. The direct Mojette transform for real valued (left) and unary (right) image

The MOT can be either performed by directly addition of the image grey values or by additio-modulus (to preserve the magnitude of the computed spectral coefficients). For binary images operation XOR may be used (see [5], [6] and [7]).

III. INVERSE MOJETTE TRANSFORM

The inverse Mojette transform (IMOT) back-projects the bins of the different projections onto the reconstructed image. That means a single pixel bin correspondence must be found at each iteration in order to reconstruct a pixel value. When it has been done, this pixel value is subtracted from each projection. The Inverse Mojette Transform means to iterate this process until the image is completely reconstructed. For the reconstruction one need two projection sets, one is the MOT of the image and the other is the MOT of a unary image (each pixel value is 1). Fig. 2 shows the first step of the reconstruction process.

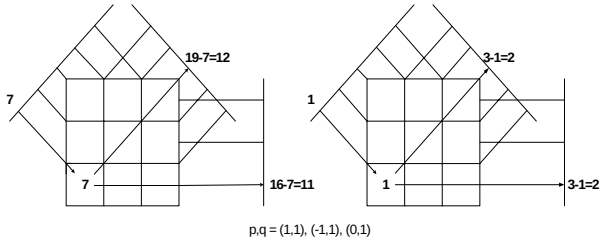

Fig. 2. Inverse Mojette Transform: (left) reconstructed image, (right) correspondence finding in unary image

A. The 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 one 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 (compute the Inverse Mojette Transform). The result was given by Katz in 1979 for rectangular image [5].

According to Katz's lemma [5], an image F of dimension NxM pixels can be reconstructed with the set of K projections $M = \{M_k\}$ if

$$N \le P_K = \sum_{k=1}^{K} \left|p_k\right| \text{ or } M \le Q_K = \sum_{k=1}^{K} \left|q_k\right| \qquad (7.)$$

were *N x M* is the image size, p_k is the direction coordinate of x axis (*N*), q_k is the direction of coordinate y axis (*M*). This means that for the reconstruction one need as many projections as the sum of the absolute value of the projection coordinates have to be equal or greater then the dimension of the image in the same direction. There is no need that both sums satisfy the criteria, it is sufficient if one of the sums does.

The order of complexity of the MOT is linear both in the number of projections and the number of pixels. In order to obtain the same complexity for the IMOT, a specific list of single pixel-bin correspondence has to be created and managed [6, 7].

The main characteristic of the MOT is the redundancy, which can be measured by the following parameter:

$$R = \frac{\sum bins}{\sum pixels} - 1 \qquad (8.)$$

Due to the ill-posed nature of the IMOT important degradation occurs when the bins have been slightly modified during the transfer (i.e. before the IMOT process start). [5], [6] and [7].

IV. "MOTIMOT" CO-PROCESSOR IMPLEMENTATION

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 (subject of later analyses) real time operating system kernel.

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.).

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 co-processors can be implemented in an ASIC chip or in Field Programmable Gate Array (FPGA). The implementation is motivated by this two solutions as in this way the algorithm is implemented in parallel processes and using relatively low frequencies (100-300MHz) and can be obtained very high processing speeds.

As stated in [15] real time image processing needs very high computing performances with relatively low power dissipation, which condition can not accomplished by PC and fixed point DSP. Also the ASIC solution is not suitable for an academic research. While using FPGA the advantage is of using embedded on-chip processors with on-chip memory and on chip system bus. For this reason and for the flexibility the Virtex IIP FPGA chip will be used. During the design process the ML310 board is used. This board is a Xilinx embedded system board and for details see [14].

Just to summarize the ML310 board hardware possibilities some characteristics are mentioned: the Virtex-II Pro™ FPGA with the embedded PPC405, the ML310 board features the following:

- Virtex-II Pro™ XC2VP30 FPGA
- ATX form factor motherboard
- 256 MB DDR DIMM
- System ACE™ CF controller
- 512 MB Compact Flash card
- Onboard 10/100 Ethernet network interface card (NIC)
- RS-232 mini-cable
- Standard JTAG connectivity
- ALi South Bridge Super I/O controller (parallel and serial ports, USB, etc.)
- ATX power supply

The System ACE™ CF controller is the power on configuration manager with multiple configuration possibilities. System ACE supports multiple bitstream management, FPGA microprocessor cores, and system reconfiguration/update over a network. They also provide an easily scalable and reusable configuration platform, a built in interface to a system processor, and the ability to centralize configuration for the entire system, simplifying debug and minimizing board space.

The main advantage using FPGAs is the flexibility of development tools, the hardware-software co-design solution, and hardware in the loop simulation.

Starting from a 256x256 pixel sized greyscale image this requires 64KB memory. In order to process in parallel all the projection lines (in the case of the direct MOT) one need as many 64KB sized memory blocks (i.e. Block RAM) as many projection lines we have for this image size. The bin vectors are stored in the external memory in a so called “MOT memory file”. The dimension of the MOT memory file is given by the product between the image slices, the number of bins ni (see eq. 6.) and the maximum size of a bin. , which is given below:

$$\lambda_i = \min(N \bmod |p_i|, M \bmod |q_i|) \quad (9.)$$

$$bin\_size = \max[\{\lambda_i | i = 1, w] \quad (10.)$$

$$\max(bit\_nr) = C + (bin\_size \bmod 2 + x) \quad (11.)$$

where

$$x = \begin{cases} 0 \; if \; bin\_size = 2k, k \in Z^+ \\ 1 \; if \; bin\_size = 2k+1, k \in Z^+ \end{cases}$$

$$mem_{file\_size} = n_{slices} \cdot \sum_i n_i \cdot \max(bit\_nr), \quad (12.)$$

where N, M, p_i, q_i were specified in eq. (6), λ_i is the maximum number of pixels contained by a bin in the i^{th} projection line, w is the number of projection lines, C is the colour depth of the original images (8 bits for a *.pgm), bin_size is the maximum number of pixels contained by a bin for any projection line, n_i is the number of bins in the projection line i and n_{slices} is the number of image slices. An image slice is obtained by the division of the image by an even number.

The division of the original image in slices is motivated by the fact that during the transmission of the MOT file this can be corrupted. From the corrupted MOT file the image can not be reconstructed without damaged area. While applying the Mojette transform to slices the effect of the MOT corruption is diminished. Also the memory necessary to calculate the MOT and IMOT is smaller in the case of slices. Processing the whole image would result in the need for reconfiguration

in order to calculate all the projections, because to load the 256x256 image in the embedded memory need 256KB only for 4 projection lines (the XC2VP30 has 1.7Mb Block RAM ≈ 212KB). For this reason the 256x256 pixel image is divided in 4 slices (128x128).

While processing the IMOT (for the same image size) one needs to read the MOT memory file from the external memory and to reconstruct the image. For the reconstruction one need to generate the MOT file of the same size unary image with the same projection lines. From eq. (10) the bin size means the maximum pixel numbers contained by a bin and also defines the number of bits needed for the unary bins of the unary MOT file. During the back projection (IMOT) the bins which contain only one pixel value are substituted to their corresponding position. These pixel values contained by other bins (in other projections) also and those bin values have to be decreased with the current pixel value. To calculate the positions of the bins in the projections and to calculate the correspondence in the image of a single pixel bin the unary image is needed. It results when the value of a single pixel bin is substituted and the other bin values are decreased, the changes also have to be validated in the unary MOT file. The unary MOT file contain unary bins. These bins contain not only the current pixel number contained by the bin in the MOT file but the corresponding position in the image of these pixels to.

Since there is no need to process at the same time the MOT and the IMOT there is the possibility to activate only one co-processor at a time by using partial run-time reconfiguration. The reconfiguration is activated by the embedded Power PC processor using the ICAP interface, since the ML310 board based system is intended to be an independent working platform dedicated for Mojette transform.

The proposed MOT/IMOT hardware is strongly dependent by the ML310 board structure given in [13]. This dependency constrain only the communication channels to the devices from which the images are received, but do not impose limits to the co-processor implementations. Fig. 3 shows the internal block schema of a MOT/IMOT computing hardware. In the figure the MOT/IMOT is represented as a single block, since the implementation area needed for one of them will be used to implement the other one, this may explain why partial reconfiguration is needed.

Fig. 3 summarise only the main parts of the image processing system. This are: the PowerPC as the main processing unit, the MOT unit and the IMOT unit. Both MOT and IMOT processors are connected to the PowerPC via the internal PLB bus, because their work is running under the main processor control.

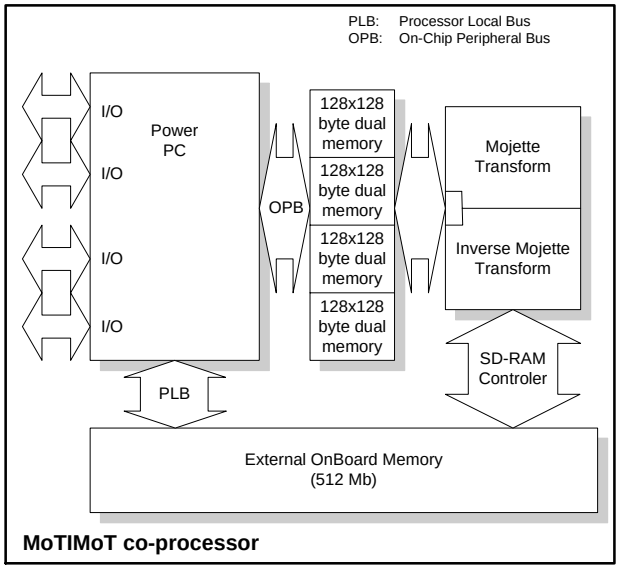

Fig. 3. MoTIMoT Processor Block Scheme

The image processing algorithms are memory costly therefore the onboard external memory will be used as a dual memory for temporary store the image while MOT is applied and for the image reconstruction while IMOT is applied.

The PowerPC is the embedded hard core of the Virtex II chip and provides the tasks which are needed by the system. Also the main processor will supply the data to the MOT and receive data from IMOT units by transferring the 128x128 pixel size image slices from and to the external memory.

A. *MOT Block Scheme*

Fig. 4 shows the block scheme of the MOT computing unit. The dual memory unit with the size of 4x128x128 byte enables to decrease the I/O operation between the onboard external memory and the MOT processor. The 128x128 image window is loaded in the dual memory 4 times, as 4 projection lines are processed in parallel at once. In this way the quarter of the Mojette memory file (MoTF) is computed at one instance. The SD-RAM controller is responsible for the organisation of MoTF structure. Each record in the file is a projection line of the transform. A part of the image, a 128x128 pixel size window will be transformed, so the transform will be applied to the original image for four times. The advantage of this method compared to other implementations [8] and [9] is the parallel computing of

each projection line, which results in the increase of processing speed on lower working frequencies (100MHz). Another advantage as was mentioned is the diminution of damages in the picture when the bins get crashed. Also as mentioned when applying the transform on 256x256 pixel size image and a bin get crashed, the reconstructed image contain only distortions in the columns of the bin and in the close neighbourhood. Using only a 128x128 pixel size slice of image, the vertical size of the damaged bin area is decreased by 50%.

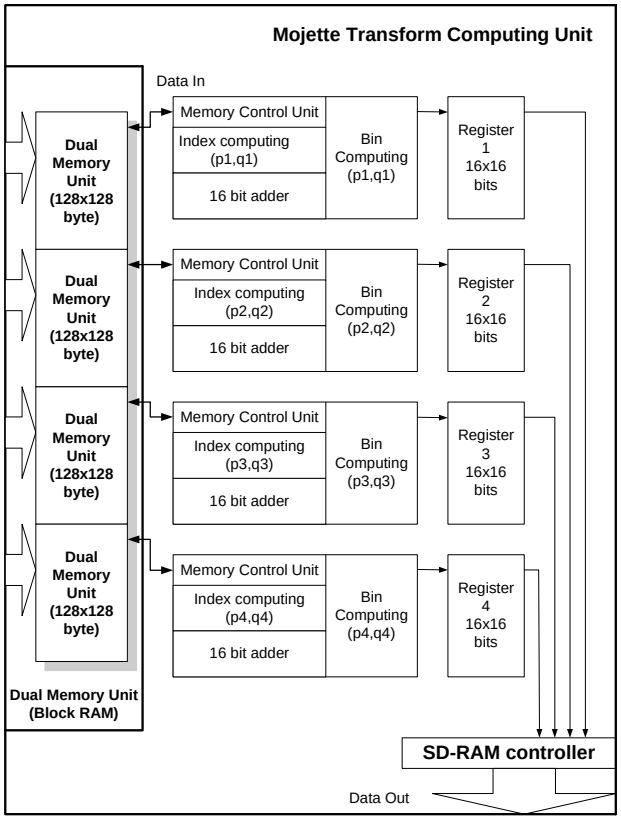

Fig. 4. MOT Computing Unit

The index computing units has a starting address (increased by a counter) and computes the memory address of the corresponding pixels. After that the pixel values are summed up by the 10-bit adder, the output result the bin (ten bits long).

The code sequence calculating the bins of a projection (p=1, q=32, im_size=127) is shown in Fig. 5. The result is stored in a 16x16 bits size register where the lower 10 bits are the bins, and the higher 4bits are the unary bins (ubins) inserted as correspondence information. When the registers are full, the bins are sent to the external memory for storage.

```
for (i=0; i<=im_size+3p; i++)
      {
      for (j=0; j<=q-1; j++)
            {
            if (i= =0) do
                  bin[i+j]=im[i,j];
            if (i = =1) do
                  bin[i+j]=im[i,j]+im[i-p, j+q];
            if (i = =2) do
                  bin[i+j]=im[i,j]+im[i-p, j+q]
                              +im[i-2p,j+2q];
            if ((i = =3) && (i<=im_size) do
                  bin[i+j]=im[i,j]+im[i-p, j+q]
                  +im[i-2p,j+2q] +im[i-3p,j+3q];
            if (i= =im_size+p) do
                  bin[i+j]= im[i-p, j+q]+im[i-2p,j+2q]
                        +im[i-3p,j+3q];
            if (i= =im_size+2p) do
                  bin[i+j]= im[i-2p,j+2q] +im[i-3p,j+3q];
            if (i= =im_size+3p) do
                  bin[i+j]= im[i-3p,j+3q];

          }
}
```

Fig. 5. Code sequence of projection line computation

B. *IMOT Block Scheme*

The inverse Mojette after configuration becomes active and waits from the central processor the enable the start of the IMOT processing. The quarter of the MoTF file read from the SD-RAM is loaded in a temporary on chip memory, this is necessary because the memory MOT file have to be accessed as a whole and in parallel for all the back projections.

The IMOT computing co-processor is somehow similar to the MOT one, but instead of the bin storage registers is need for a temporary memory. The size of this memory (divided in 4 blocks) results from eq. (6) completed with the corresponding ubin file, as was described in the previous section (A).

During the back projection process when a single pixel bin is found, the pixel value is replaced into the blank image and is subtracted from the corresponding bins of each projection record of the MOT file. The reconstructed image is stored in the dual memory. The reconstruction needs a register access arbiter in order to compute the pixel position size. This results from the bin position in the MOT file and the corresponding ubin value. The arbiter decides which of the reconstruction module can access the temporary registers. The pixel reconstruction unit will use (the analogue of the bin computing unit) value contained in the temporary storage register to reconstruct its pixel and to update the bin values.

V. SIMULATION RESULTS

The simulations were made on PC hardware environment using a portable grey map 256x256 image without transmission and no bit-corrupted errors.

Fig. 6 *a*. shows the original image and Fig. 6 *b* shows the reconstructed image without any errors.

a) original image; *b) reconstructed image*

Fig. 6. Simulation result of original and MOT/IMOT transformed image

The simulation proved the correctness of the implemented algorithms and the functionality of the proposed hardware.

VI. CONCLUSIONS

The paper outlined the hardware requirements for the implementation of Mojette direct and inverse transformation in the embedded system using FPGA. Original contribution is the calculation of the dimension of MoTF, the definition and analysis of the hardware structure as a whole, with the simulation results.

Future work is needed to finalise the implementation of the co-processors as a whole with run-time reconfiguration.

AKNOWLEDGMENT

The authors gratefully acknowledge the donations of *Xilinx Inc.* and *Celoxica Inc.* which made possible the start of this research.

The research is part of the Hungarian-Research Project GVOP 3.1.1-2004-05-0333/3.0.

REFERENCES

[1] Hartung, F., Kutter, M.: *Multimedia Watermarking Techniques*, Proc. IEEE, vol. 87, No7, 1999.

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

[3] 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.

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

[5] Katz, M.: *Questions of uniqueness and resolution in reconstruction from projections*. Springer Verlag, Berlin, 1977. pp.

[6] 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.

[7] 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

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

[9] Szoboszlai P.: *Digital Watermarking with Mojette and Wawelet transforms*, Diploma work, University of Miskolc, Department of Automation, 2006, pp. 58.

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

[11] Guédon J-P., Normand N.**:** Spline Mojette Transform Application in tomography and communication, In EUSIPCO, sep. 2002.

[12] 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.

[13] Xilinx, *ML310 User Guide*, pp73, http:\\xilinx.com

[14] Bobda C., Huebner M., Niyonkuru A, Blodget B., Majer M., Ahmadinia A.,: *Designing Partial and Dynamically Reconfigurable Applications on Xilinx Virtex-II FPGAs using Handel-C,* "http://www.celoxica.com/cup/registered_users/community/default.asp", pp. 16.

[15] Amilcar do Carmo L., Heithecker S., Rolf E., *FlexFilm An Image Processor for Digital Film Processing,* Dagstuhl Seminar on Dynamically Reconfigurable Architectures, 2-7.04.2006. http://www.dagstuhl.de/06141/Materials/

[16] Heithecker S., Rolf E., *FlexFilm: A Quality of Service Capable Scheduling SDRAM Controller***,** Dagstuhl Seminar on Dynamically Reconfigurable Architectures, 2-7.04.2006. http://www.dagstuhl.de/06141/Materials/