Implementation of CT and IHT Processors for Invariant Object Recognition System

Ján TURÁN[1], *Ľuboš OVSENÍK*[1], *Martin BENČA*[1], *Ján TURÁN, Jr.*[2]

[1] Dept. of Electronics and Multimedia Telecommun., Univ. of Technology, Park Komenského 3, 04021 Košice, Slovakia
[2] 3D People, GmbH, Keiser Passage D-72766, Reutlingen, Germany

jan.turan@tuke.sk, lubos.ovsenik@tuke.sk

Abstract. *This paper presents PDL or ASIC implementation of key modules of invariant object recognition system based on the combination of the Incremental Hough transform (IHT), correlation and rapid transform (RT). The invariant object recognition system was represented partially in C++ language for general-purpose processor on personal computer and partially described in VHDL code for implementation in PLD or ASIC.*

Keywords

Invariant object recognition, feature extraction, Hough and Rapid transform, PLD and ASIC implementation.

1. Introduction

The field of invariant image recognition is widely increasing part of image processing area. Various complex techniques must be used in order to recognize translated, rotated and scaled input objects. The recognition system (Fig. 1) can be partitioned into modules such as pre-processing (sensing, segmentation), invariant feature extraction and classification. The invariant feature extraction of recognized image always plays a central role. Invariant feature extractor has a big CPU time nature, being normally implemented in software on general-purpose processor (GPP). Custom implementations in hardware allow faster or real-time processing [1], [2].

This paper proposes hardware (PLD or ASIC) implementation of key modules of invariant object recognition system (IHTP – Incremental Hough Transform Processor and CTP – Creatin Transform Processor) aiming the integration of GPP with programmable logic devices (PLD). The system comprises of pre-processing, invariant feature extraction, and classification tasks. The GPP is used when sequential processing is required or complicated floating-point arithmetic is needed. The PLD is used for parallel and bit level operations as well as simple floating-point arithmetic operations. This improves the performance of the system when compared to pure GPP solution, reduces the processing time and cost when compared to pure application-specific integrated circuit (ASIC) implementation. So, the hardware/software system can represent a good trade-off between performance and cost [3].

2. Invariant Object Recognition System and Its Hardware Implementation

The invariant object recognition system (Fig. 1.) and its software realization aspects are described in [4]. Invariant object recognition system consists of three subsystems:

- Digital image pre-processing system,
- Invariant feature extractor,
- Classifier.

An input image from a CCD camera is converted to a binary image using a pre-processing system. The pre-processing system includes the following operations: noise filtering, digitalization, edge detection, tresholding, thinning, and segmentation. The feature extractor has three stages. The first stage implements the Incremental Hough transform (IHT) to map the original image to the Hough parameter space. The second stage provides translation and scale invariant capabilities by correlation operations. The third stage is an implementation of translation invariant transform (RT) used to provide rotation invariant features. As classifier, a simple Euclidean distance classifier is used.

The hardware implementation was developed in two versions [7] (Fig. 2).

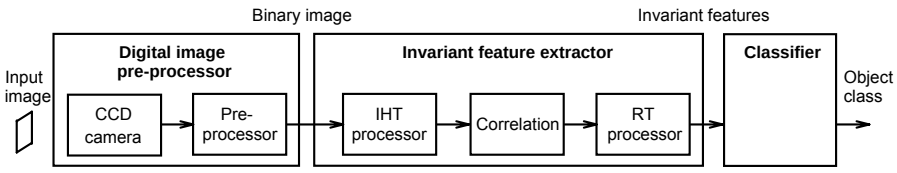

Fig. 1. The block scheme of the proposed invariant object recognition system.

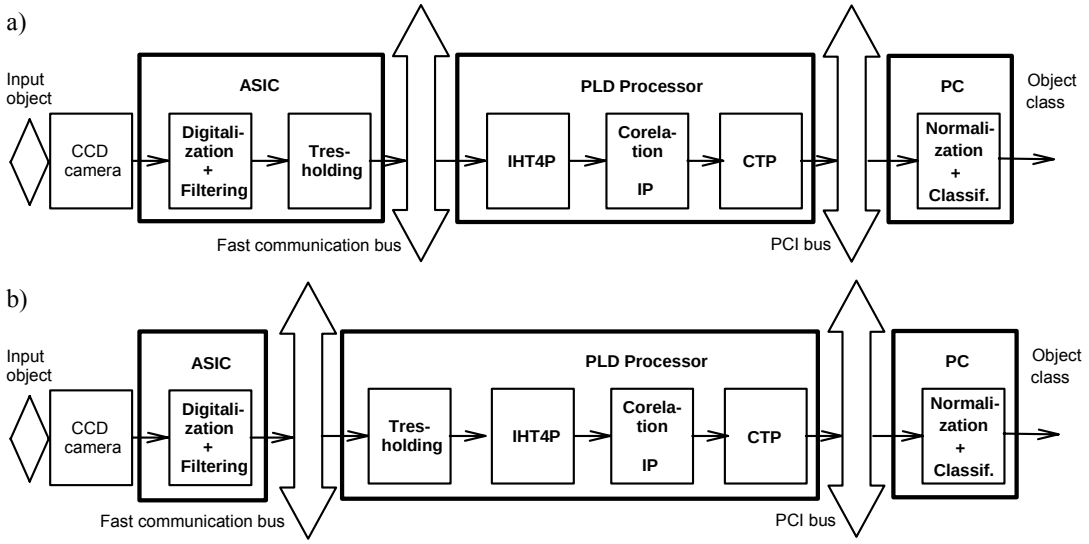

Fig. 2. The block scheme of invariant object recognition system: ASIC version a) and PLD version b).

2.1 ASIC Concept

A CCD camera obtains an input image. Pre-processing operations such as noise filtering, digitalization, edge detection, tresholding and thinning are done by a specialized ASIC chipset [5]. A communication of a pre-processing unit with a PLD processor is realized by a fast communication bus controlled by PLD. Parametrisable IHT processor maps the binary image to Hough parameter space. The correlation unit is a standard function available in an Intellectual Property core (IP core) program from various vendors [6]. Parametrisable CT processor provides rotation invariant features that are proposed via fast PCI bus to PC. Scale invariance is done in normalization unit in PC as well as the final classification task. The block scheme of ASIC version of invariant object recognition system can be seen in Fig. 2a.

2.2 PLD Concept

A CCD camera obtains an input image. Noise filtering and digitalization are done by a specialized ASIC chipset. Pre-processing operations such as edge detection, tresholding and thinning are done in PLD by IP core function. Parametrisable IHT processor maps the binary image to Hough parameter space. The correlation unit is a standard function available in IP core program. Parametrisable CT processor provides rotation invariant features that are proposed via fast PCI bus to PC. Scale invariance is done in normalization unit in PC as well as the final classification task. The block scheme of PLD version of invariant object recognition system can be seen in Fig. 2b.

3. CT Processor

3.1 Transforms from the Class CT

The class of fast translation invariant transforms (CT) was introduced in 1977 [8]. The class CT results as an extension of the original RT [9] using instead of $(a+b)$ and $|a-b|$ operators, any pair of commutative operators $f_1(a,b)$ and $f_2(a,b)$.

Transform / Operators	${}^{N}RT$	${}^{N}QT$	${}^{N}NT$	${}^{N}MT$
$f_1(a,b)$	a + b	a + b	max{a,b}	a v b
$f_2(a,b)$	|a – b|	$(a-b)^2$	min{a,b}	a ^ b

Tab. 1. Transforms from the class CT.

While the set of all commutative operators is infinite, the class CT may contain infinite number of transforms. Overview of all transform used in this paper is shown in Tab. 1.

There are several algorithms for RT computation [10], [11]. The algorithms differ from each other by ordering of computed spectral coefficients, and by sequencing of operations in transform steps. B-algorithm is usually used in CT processors because of identical operations that are performed in each transform step [10].

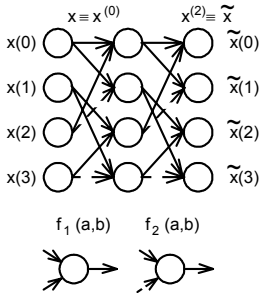

Fig. 3. Signal graph for CT (for N=4).

The principle of CT calculation (B-algorithm) for 1D digital signal (a vector with 4 elements) is depicted as a signal graph in Fig. 3. For the field of input data (numbered from 0 to N-1) $n=\log_2 N$ transform steps are needed. The transformed data are computed using the following formulas

$$x^{(r)}(2i) = f_1\left(x^{(r-1)}(i), x^{(r-1)}(i+N/2)\right)$$
$$x^{(r)}(2i+1) = f_2\left(x^{(r-1)}(i), x^{(r-1)}(i+N/2)\right) \quad (1)$$

where i=0,...,N/2-1; r is the transform step, $x^{(r)}$ is the data of r transform step [10].

3.2 Parametrisable CTP Design

A CT processor (CTP) can be defined as a computational unit for performing of transforms from the class CT. There are many possible architectures of CTP in relations to an algorithm, a used degree of parallelism [10], [11]. A CTP can be classified by complexity, speed and resource cost. There are four architectures of CTP, which compute transforms from the class CT with the help of B-algorithm, described in [10]: serial-recirculate CTP, parallel-recirculate CTP, parallel-cascade CTP, parallel-pipelined CTP.

In the following we give results of the development work related to a design of two new parametrisable CTPs with the possibility of changing a width of input elements and transform type (RT, QT, NT and MT). The CTPs were described in VHDL code for programmable logic devices and tested with help of simulation tool of a design system for programmable logic devices.

3.2.1 Serial-Recirculate CTP

Input data are shifted to a serial input register, which is composed of two parts RIB and RIA. When the register is fulfilled with the elements of the vector $\mathbf{x}^{(r)}$, then there are the elements $x^{(r)}(N/2)$ and $x^{(r)}(0)$ at the input of a functional block (FB). FB computes operations f_1 and f_2 according to the coding and then sequentially computes all the elements from the next transform step $\mathbf{x}^{(r+1)}$, which are sequentially shifted to the output register RO. After processing all the data from the registers RIB and RIA, i.e. after the register RO is fulfilled, data are shifted to RIB and RIA through switch RO and the whole cycle of the transform is repeated. After n transform steps CT spectral coefficients [10] are in the RO.

3.2.2 Parallel-Recirculate CTP

This architecture of CTP enables parallel computation of CT spectral coefficients so that we realize technically one cascade only and the output data from this cascade recirculate n-times [10].

3.2.3 Parallel-Cascade CTP

This CTP consists of the one register field (RF) and according to B-algorithm connected n fields of functional blocks (FBF). The advantage of the using B-algorithm is the possible unification of all the connections of the used elements, because all the connections among FBF are the same [10].

3.2.4 Parallel-Pipelined CTP

If we add one RF after each FBF to the architecture of 1D parallel-cascade CTP we obtain architecture of 1D parallel-pipelined CTP. This CTP is suitable for high-speed computations of any transform of the class CT in cases where there is need of sequentially computing the transform for many vectors, which may be the case of 2D transforms [10].

3.2.5 Architecture of Parametrisable Serial CTP for PLD

Architecture of the new serial CTP is similar to general-purpose processors (PC processors). There are ALU, address counter unit, memory of data and control unit, implemented in the architecture (Fig. 4). Each transform step N_{ALU} consists of 10 sub-steps. This architecture is economic from the point of view of number of used resources, but generally with a limited speed of computing. The architecture uses a memory block of size of 2N, because of sharing of memory and using the memory partially as internal registers and partially as a memory. For i=0,2,4,..., n (even numbers) a memory region from address 0 up to N-1 serves as an input memory location, where S is the number of transform steps; n=log$_2$(N). The memory region of N up to 2N-1 is an output memory. For i=1, 3, 5,...,n-1 (odd numbers) the memory region from address N up to 2N-1 is an input memory location, and the location from 0 up to N-1 is an output memory location.

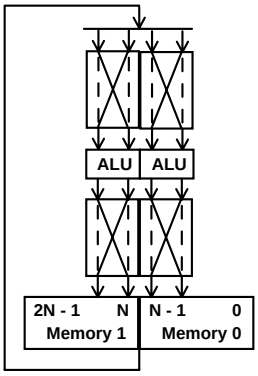

Fig. 4. Architecture of serial CT processor.

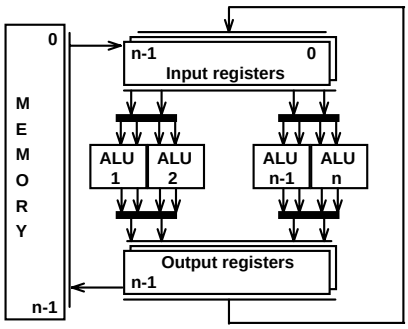

Fig. 5. Architecture of parallel CT processor.

Memory complexity of the CTP can be generally described in the form

$$M = W(M_{MEMio} + M_{ALU}) + M_x \tag{2}$$

where M is the number of register resources of the processor, W is the width of input elements, M_{MEMio} composes input, output registers and memory block, M_{ALU} are internal

registers of arithmetic-logic units (ALUs) and M_x contains registers for control part of the processor. Then

$$M = 2W(N+2) + M_x . \quad (3)$$

The total time of computation includes initialization steps, that are necessary for each transformation step, preparing of operands and writing of results

$$T = t_{per}(N_{init} + N_{comp}) \quad (4)$$

where t_{per} is the period of the main processor clock, N_{init} is the number of initialization steps, and N_{comp} is the number of computational steps. Then

$$T = t_{per}(2 + N_{ALU} \log_2 N) \quad (5)$$

where N_{ALU} is 10, because each computational step consists of 10 sub-steps.

3.2.6 Architecture of Parametrisable Parallel CTP for PLD

This architecture uses besides a memory with input elements also $2N$ registers and internal register of ALU. This approach differs from the previously design that at the beginning of computing all input elements are stored into registers. Then these elements are read concurrently because it is possible to access all of these registers at the same time. This feature increases the speed of computing at the cost of increasing of memory requirements (Fig. 5).

Memory complexity of processor can be expressed as

$$M = W(M_{io} + M_{ALU} + M_{MEM}) + M_x \quad (6)$$

where M is the number of register resources of the processor, W is the width of input elements, M_{io} composes input and output registers, M_{ALU} are internal registers of ALUs, M_{MEM} is the main memory block, and M_x contains registers for the control part of the processor. Then

$$M = 2W(N_{ALU} + N) + M_x . \quad (7)$$

The total time of computation for the parallel architecture depends on the number of ALUs and memory blocks

$$T = t_{per}(N_{init} + N_{op} + N_{comp}) \quad (8)$$

where t_{per} is the period of the main processor clock, N_{init} is the number of initialization steps, N_{op} is the number of steps for preparation (reading) of operands into registers from a memory, and N_{comp} is the number of calculation steps. It follows

$$T = t_{per}\left(1 + \frac{2N}{N_{MEM}} + \frac{N \log_2 N}{N_{ALU}}\right) \quad (9)$$

where N_{ALU} is 2, because each computational step consists of 2 sub-steps: reading of operands, writing of results; N is the number of input elements, N_{ALU} is the number of ALUs.

3.3 Experiments and Results

3.3.1 CT Transforms Computed on General-Purpose Processor

Implementation of CTP on a general-purpose processor (PC) is straightforward. We have used C++ compiler from Borland Company. Because of measuring of time event being possible only on frame of milliseconds, the program repeats computing of CT transform cyclically. We have changed input elements for each run to prevent a situation that compiler does optimization of computing. We have accomplished experiments on two general-purpose processors of different generation and frequency for better comparison of results. The first one is older Intel 386/33MHz (P1), the second one is Intel Pentium 233 MHz (P2). P1 processor has similar working frequency as PLD, the comparing results can be interesting. P2 processor presents a relatively powerful general-purpose processor.

3.3.2 CT Transforms Computed on CTP for PLD Implementation

Each of the described transforms (RT, NT, QT and MT) can be applied to both mentioned architectures. For comparing of the speed and the number of logic resources for each of described transforms we have done experiments with serial CTP for N=8 to 512 and W=16. It follows from the results that the quickest transform is MT, then RT and NT have quite similar nature, only NT consumes less logic resources than RT. The QT is the slowest and the most resources demanding transform. MT transform can be applied only to integer number. For real numbers, we have used 32-bit wide IEEE single precision floating-point number standard with 24 bits of a mantissa, 8 bits of an exponent and a sign. The NT real number transform is more accurate than transforms for integer number, which is compensating also with bigger demand for logic resources [12].

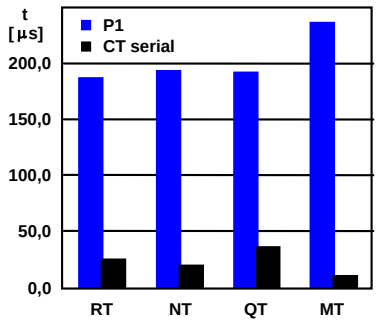

Fig. 6. Dependence of computing time of transform for N=32, W=16, P1 vs. serial CTP.

3.3.3 Comparison of Results

Results of experiments show that the serial CT processor is approximately 10 times, 10 times, 5 times, 20 times faster than P1 for RT, NT, QT and MT, respectively. For W=16, N=32, the computing times are 24.3µs, 19.3µs, 35.5µs, 8.9µs for RT, NT, QT and MT, resp. (Fig. 6).

Results of experiments for parallel CT processors show approximately the same time of computing for all transforms with comparing to powerful general-purpose processor P2. For W=16, N=32 computing time is 3.9 ms, 2.16ms, 2.17ms, 1.22m for RT, NT, QT, MT, resp. (Fig. 7).

All timings are valid for Altera CPLD family FLEX-10K. The t_{per} value is PLD type dependent, because it implies particular types of PLD's resources, PLD synthesis tool features as well as programming technique in VHDL [12].

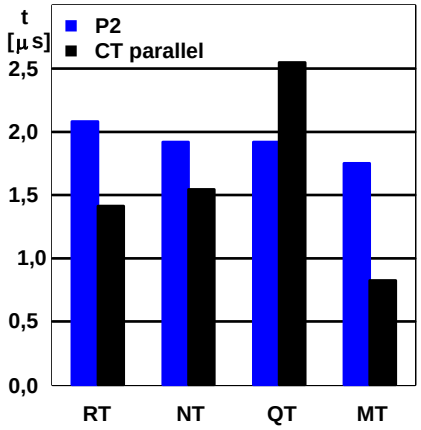

Fig. 7. Dependence of computing time on transform for N=32, W=16, P2 vs. parallel CTP.

4. IHT Processor

4.1 Incremental Hough Transform (IHT)

Using the Hough transform each image point (x,y) is mapped to the curve expressed in the r-θ plane

$$r = x\cos\theta + y\sin\theta \tag{10}$$

where r is the distance from the original and θ represents an angle between the normal and the x-axes (Fig. 8). Both r and θ axes have to be quantized and hence a two-dimensional accumulator array must be constructed in the r-θ plane [13], [14]. The equation (10) is applied to each point in the image and the contents of all the cells in the parameter space that the corresponding curve passes through are incremented.

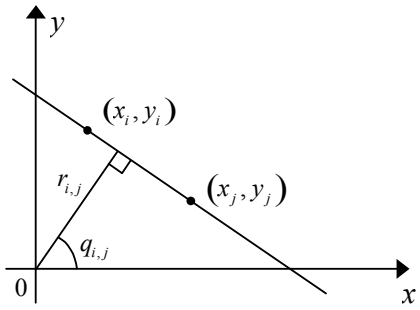

Fig. 8. Relationship between (x,y) and (r,θ).

The Hough transform method described above is known in literature as the Standard Hugh Transform (SHT) [13], [14]. Using direct implementation of SHT in PLD leads to slow and large multipliers and look-up table utilization. To solve this problem, the incremental HT (IHT2) [15], [16], [17] was utilized.

Let's assume, p is a natural number, q is a number of independent sub-ranges of IHT, $q=2^p$. If q divides K with zero remainder, then a modification of IHT – IHTq with q independent sub-ranges exists. Maximal number of sub-ranges of IHT modification is K. For every index n is IHT of point calculated as initial point, which leads to SHT calculation.

Generally a modification of IHT can be expressed as:

$$\begin{aligned} r_{\left(\frac{K}{q}d\right)+n+1} &= r_{\left(\frac{K}{q}d\right)+n} + \varepsilon r_{\left(\frac{K}{q}\left(d+\frac{N_d}{2}\right)\right)+n}, \\ r_{\left(\frac{K}{q}d\right)+n+1} &= r_{\left(\frac{K}{q}d\right)+n} - \varepsilon r_{\left(\frac{K}{q}\left(d+\frac{N_d}{2}\right)\right)+n} \end{aligned} \tag{11}$$

for $\frac{K}{q}d < \frac{K}{2}$, respectively $\frac{K}{q}d \geq \frac{K}{2}$

where $0 \leq n < K/q$, q is the number of independent sub-ranges of IHT, d is the index actual sub-ranges, and N_d is maximal index d. Initial points can be expressed as:

$$\begin{aligned} & r_0 = x; r_{\left(\frac{N_d K}{2q}\right)} = y \\ & r_{\left(\frac{K}{q}d\right)} = x\cos\left(\frac{\pi}{q}d\right) + y\sin\left(\frac{\pi}{q}d\right) \end{aligned} \tag{12}$$

The benefits of the proposed algorithm for q=4 (IHT4) are increasing of parallelism of computing by factor of 2 comparing to IHT2 (q=2) and by factor of 4 comparing to SHT.

4.2 Parametrisable IHTP Design

Hardware implementation of the core of IHT4 processor can be seen in architecture in Fig. 9. In the first transform step the initial values for each section of K are calculated (12). Then the actual values with help of the previous values are calculated in each transform step. If the value of ε is carefully selected as $1/2m$, multiplication from (11) can be realized with help of shift registers only. This simplification leads to effective and fast realization of IHT processor, which can be for IHT4 expressed as:

$$\begin{aligned} \mathrm{Re}\, g_0 &= MUX_0 + rot(MUX_2, m) \\ \mathrm{Re}\, g_1 &= MUX_1 + rot(MUX_3, m) \\ \mathrm{Re}\, g_2 &= MUX_2 + rot(MUX_0, m) \\ \mathrm{Re}\, g_3 &= MUX_3 + rot(MUX_1, m) \end{aligned} \tag{13}$$

IHT processor (IHT2 and IHT4) are parametrisable, i.e. it is possible to modify a format of representation of the float-point number, width of a mantissa or an exponent of a processed numbers as well as an image dimension.

4.3 IHTP Implementation

A comparison of difference between SHT and IHT2, respectively SHT and IHT4 can be seen in Fig. 10, using the parameters:

$$P_1 = \frac{\sum_n |f_{SHT}(n) - f_{IHT2}(n)|}{n} = 0{,}07$$
$$P_2 = \frac{\sum_n |f_{SHT}(n) - f_{IHT4}(n)|}{n} = 0{,}03 \qquad (14)$$

where $f_{SHT}(n)$, $f_{IHT2}(n)$ and $f_{IHT4}(n)$are the values computed with SHT, IHT2, IHT4, respectively at the index n. P1, resp. P2 is addition of difference of SHT and IHT2, SHT and IHT4, respectively for K=100.

It follows from Fig. 10 and experimental results that the accuracy of IHT4 is 50% better than IHT2 for the same start conditions.

It follows from the experimental results that the total time of computing on general-purpose processor (Intel Pentium 300 MHz) for IHT2 and IHT4 is 25% from the computing time of SHT for K=100. The computing time of IHT2 and INT4 is the same because of architecture of general-purpose processor with the only one ALU.

With the help of floating-point representation of real numbers it is possible to decrease the used system resources. Besides of standard IEEE type of floating-point representation (float and double) it is possible to apply also a custom type of real number representation. An exponent defines the scale of number representation and a mantissa defines an accuracy of number representation.

It follows from the experimental results that it is possible to decrease system resources for IHT2 and IHT4 algorithm by 50% by decreasing of width of exponent and mantissa of number representation.

The inaccuracy of real number representation is increased by decreasing of width of a mantissa and an exponent. From the experimental results it follows that, i.e. for IHT2 processor at number format *"m*10*e*8*"* consumes only 53% of resources used at IEEE number format *"m*23*e*8*"*. The maximal difference of computed value is 1,6% in this case. For IHT4 at *"m*9*e*8*"* number format the processor consumes only 50% of resources used at IEEE format. The maximal difference is 1,6%.

With the help of modification of number format it is possible to design an IHT processor with a sufficient accuracy and with a lower usage of system resources. The rest of unused resources – logic units, flip-flop memory blocks can be utilized for another part of the developed invariant object recognition system.

A modification of number representation format affects the computing time as well. It follows from the experimental results that for the custom format *"m*9*e*8*"* the computing time is only 40% of the computing time for the standard IEEE format with single precision, i.e. *"m*23*e*8*"*.

5. Conclusion

We have presented the implementation of key modules (CTP and IHTP) of the invariant object recognition system based on the combination of the incremental Hough transform, correlation and rapid transform. The system was represented partially in C++ language for GPP for PC and partially in VHDL code for implementation in PLD or in ASIC. We are now performing additional experiments for further examination of the system capability. In future we will focus also on a possibility of finding optimal approaches for hardware implementation of the proposed system.

Acknowledgements

The authors are thanking for the financial support from the COST 276 and COST 292 grant and VEGA grant No. 1/0381/03.

References

[1] DAVIES, E. R. *Machine Vision: Theory, Algorithms, Practicabilities.* London: Academic Press, 1990.

[2] UMBAUGH, S. E. *Computer Vision and Image Processing: A Practical Approach Using CVIP-tools.* Prentice Hall, 1998.

[3] STAMOULIS, I., FORD, N., DUNNET, G. J., WHITE, M., LISTER, P. F. *WHDL Methodologies for Effective Implementation on FPGA Devices and Subsequent Transition to ASIC Technology.* Centre for VLSUI and Computer Graphic, Brighton, 1998.

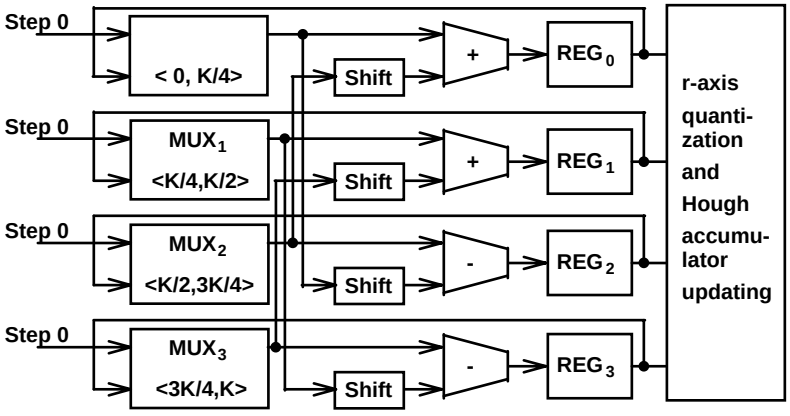

Fig. 9. Architecture of IHT4 processor.

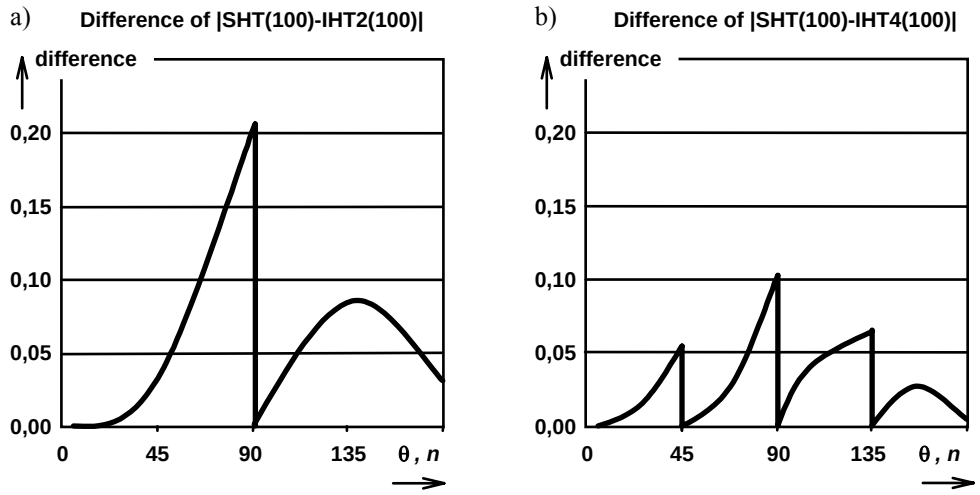

Fig. 10. Comparison of difference of SHT and IHT2 a), resp. SHT and IHT4 b) for K=100.

[4] TURÁN, J., FAZEKAS, K., FARKAŠ, P., ŠIŠKOVIČOVÁ, D. Invariant Feature Extraction Based on the Hough Transform.In *Proc. IWSSIP 2001*(invited paper), Bucharest, 2001, pp. 39-42.

[5] MOINI, A. *Vision Chips*. Kluwer Academic Publishers, 2000.

[6] FANNING, J. *Literature Survey of Present State of FPGA's.* Department of Instrumentation and Analytical Science, UMIST, Manchester, 1999.

[7] TURÁN, J., BENČA, M., FARKAŠ, P. Hardware Implementation of Key Modules of Invariant Object Recognition System Based on Hough and Rapid Transform. In *Proc. ICCVG 2002,* Zakopane (Poland), 2002, pp. 764-770.

[8] WAGH, M. D., KANETKAR, S. V. A Class of Translation Invariant Transform. *IEEE Trans,* vol. ASSP-25, no. 3, 1977, pp. 203-205.

[9] REIBOECK, H., BRODY, T. P. A Transformation with Invariance Under Cyclic Permutation for Application in Pattern Recognition. *Inf. and Control*, vol. 15, 1969, pp. 130-154.

[10] TURÁN, J. *Fast Translation Invariant Transforms and their Applications*. Košice: Elfa, 1999.

[11] LOHWEG, V. Ein Beitrag zur effektiven Implementierung adaptiver Spektraltransformationen in applikations-spezifische integrierte Schaltkreise. *Dissertationsschrift*, TU Chemnitz, Germany, 2003.

[12] Altera Digital Library 2001, version 2, version 3.

[13] HOUGH, P. V. C. Method and Means for Recognizing Complex Pattern. *U.S. Patent 3069654*, 1962.

[14] DUDA, R. O., HART, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm. of ACM,* vol. 15, no. 1, 1972.

[15] KOSHIMIZU, H., NUMADA, M. FIHT2 Algorithm: a fast incremental Hough Transform. *IEICE Trans*., 74 (10), October 1991, pp. 7-14.

[16] TAGZOUT, S., ACHOUR, K., DJEKOUNE, O. Hough Transform Algorithm for FPGA Implementation. *Signal Processing*, 84 (2001), pp. 1295-1301.

[17] BESSALAH, H., ALIN, F., SEDDIKI, S. Implementation of the Hough Transform by the On-line Mode. In *Proc. VIProm Com 2002,* Zadar (Croatia), 2002, pp. 167-171.

About Authors...

Ján TURÁN (Prof., Ing., RNDr., DrSc.) was born in Šahy, Slovakia. He received Ing. (MSc.) degree in physical engineering with honours from the CTU Prague, Czech Republic, in 1974, and RNDr. (MSc.) degree in experimental physics with honours from the UK Prague, Czech Republic, in 1980. He received a CSc. (PhD.) and DrSc. degrees in radioelectronics from the TU Košice, in 1983, and 1992, respectively. Since March 1979, he has been at the TU Košice as Professor for electronics and information technology. His research interests include digital signal processing and fiber optics, communication and sensing.

Ľuboš OVSENÍK (Ing., PhD.) was born in Považská Bystrica, Slovakia. He received his Ing. (MSc.) degree in 1990 from the TU Košice. He received a PhD. degree in electronics from the TU Košice, Slovakia, in 2002. Since February 1997, he has been at the TU Košice as Assistant professor for electronics and information technology. His general research interests include optoelectronic, digital signal processing, photonics, fiber optic communications.

Martin BENČA (Ing., PhD.) was born in Prešov, Slovakia. He studied computer science and informatics at the TU Košice. His research interests include multimedia signal processing and programmable logic devices.

Ján TURÁN, Jr. (Ing.) was born in Košice, Slovakia. He studied computer science and informatics at the TU Košice. His research interests include multimedia signal processing and programmable logic devices.