AUTOMATIC VHDL GENERATION FOR SOLVING ROTATION AND SCALE-INVARIANT TEMPLATE MATCHING IN FPGA

Henrique P. A. Nobre and Hae Yong Kim

Escola Politécnica, Universidade de São Paulo, Brazil
hennobre@provida.org.br and hae@lps.usp.br

ABSTRACT

Template matching is a classical problem in computer vision. It consists in detecting the presence of a given template in a digital image. This task becomes considerably more complex with the invariance to rotation, scale, translation, brightness and contrast (RSTBC). A novel RSTBC-invariant robust template matching algorithm named Ciratefi was recently proposed. However, its execution in a conventional computer takes several seconds. Moreover, the implementation of its general version in hardware is difficult, because there are many adjustable parameters. This paper proposes a software that automatically generates compilable Hardware Description Logic (VHDL) modules that implements Ciratefi in Field Programmable Gate Array (FPGA) devices. The proposed solution accelerates the time to process a frame from 7s (in a 3GHz PC) to 1.06ms. This excellent performance (more than the required for a real-time system) may lead to cost-effective high-performance co-processing computer vision systems.

1. INTRODUCTION

Image processing and computer vision are becoming popular in many areas and more and more powerful but time-consuming new algorithms are being developed. These algorithms require intensive use of mathematical calculations, such as averaging, interpolation and correlation. FPGA (Field-Programmable Gate Array) seems to be an adequate hardware system for many image processing algorithms, because FPGAs have the capability of making thousands of tasks in a single clock. Authors in [1] mention advantages of FPGAs for image processing and authors in [2] compare performance of FPGA systems. Paper [3] exemplifies high performance of FPGA architectures, where the authors implemented a string matching algorithm with execution time 8-340 times faster than the implementation in a Pentium 4 computer 3.5 GHz.

Parallel processing is particularly important in image processing algorithms where the same series of operations must be repeated for each pixel. Authors in [4] developed an object-tracking method based on a real time vision module. In this method, real-time image-processing performance is achieved by a parallel implementation in a multiprocessor, DSP-based system. A more complex system was proposed in [5], where the authors implemented highly parallel architecture for real-time object recognition using signal processing and FPGA technologies.

A novel template matching, invariant to rotation, scale, translation, brightness and contrast (RSTBC), named Ciratefi was recently proposed [6, 7]. The implementation of its general version in hardware is difficult, because there are many adjustable parameters that require hardware modifications. Figure 1 demonstrates the matching result of the algorithm. This algorithm repeats the same series of operations for each pixel, what makes it good for hardware implementation. However, the hardware implementation of the general version of Ciratefi is not straightforward, because it must be flexible to changes in the size of the template and the range of template scales. Therefore, we propose a software that, given the parameters, automatically generates compilables VHDL modules that implements Ciratefi in FPGA. The generated VHDL modules are highly optimized and pipelined. The performance of our implementation is optimal, because it classifies one pixel (as matching or non-matching) per clock.

Ciratefi consists of three cascaded filters, named Cifi, Rafi and Tefi. The first two filters are time-consuming and they are the objects of this hardware implementation. The last one is usually fast even in software and we have decided to let it as a software implementation. Therefore, we present in this paper a FPGA system that implements Cifi and Rafi. However, until now, we have completed only the implementation of Cifi. The implementation of Rafi is quite similar and we intend to finalize it soon. The proposed solution has accelerated Cifi 5000 times, from 7s to process a frame (in a 3GHz PC) to 1.06ms (in FPGA).

In the literature, there are some other implementations of template matching in FPGA. Hegel et al. [8] present a binary image template matching algorithm in FPGA. This technique is neither rotation nor scale-invariant. Shen et al. [9] present a rotation-invariant template matching based on circular projections. However, it is not scale-invariant.

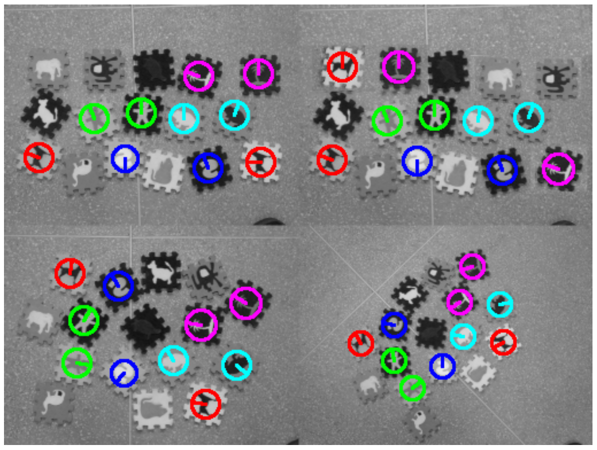


Fig. 1. Output of Ciratefi, where the matching positions, angles and scales are depicted as circles with pointers

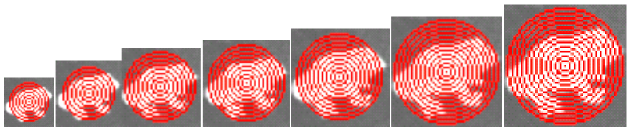


Fig. 2. The circular projections different scales

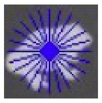


Fig. 3. The radial projection at the selected scale

2. CIRATEFI

The objective of the Ciratefi algorithm is to find a grayscale query image Q in a larger image "to analyze" A, invariant to rotation, scaling, translation, brightness and contrast. We present below a brief description of Ciratefi. The readers are referred to [6] for further details.

Ciratefi consists of three cascaded filters. Each filter successively excludes pixels that have no chance of matching the template.

2.1. First Filter: Cifi

The first filter, called Cifi (Circular Sampling Filter), computes the average grayscales of images A and Q on circles (figure 2), and uses them to classify some pixels of A as "first grade candidate pixels" for matching. This filter also determines a "probable scale factor" for each candidate pixel. To accomplish this, Cifi makes successive correlations between the 2-D matrix C_Q of average values on circles of Q in several scale factors (figure 2) and the 3-D matrix C_A that contains, for every pixel (x,y) in A, a vector of the average values of the circles centered at (x,y). That is, C_A is obtained processing all the pixels of A according to the equation:

$$C_A[x, y, r] = \frac{1}{2\pi r}\int_0^{2\pi} B(x + r\cos\theta, y + r\sin\theta)d\theta \qquad (1)$$

Cifi uses matrices C_Q and C_A to detect the correlation coefficient at the best matching scale for each pixel (x,y). A pixel (x,y) is classified as a first grade candidate pixel if the best correlation is larger than some defined threshold.

2.2. Second Filter: Rafi

The second filter, called Rafi (Radial Sampling Filter), computes, for each first grade candidate pixel (x,y), the projections of images A and Q on radial lines (figure 3) with the radius given by the scale factor computed by Cifi. Rafi upgrades the first grade candidate pixels that have chance of matching the template to the second grade. It makes successive correlations between the two sets of projections using circular shifting. It also computes the "probable rotation angle" for each second grade candidate pixel.

2.3. Third Filter: Tefi

The third filter, called Tefi (Template Matching Filter), is a conventional brightness and contrast-invariant template matching applied to the second grade pixels, using the scales and angles determined respectively by Cifi and Rafi. It makes use of the correlation coefficient to evaluate how well the template Q matches each second grade candidate pixel.

2.4. Performance of Ciratefi

The interesting points of Ciratefi are its roughness and accuracy when compared to others algorithms [6]. However, it takes several seconds to compute the matching positions. For the worst case tested (A with 465×338 pixels, Q with 52×51 pixels, 6 scales and 36 angles) the complete Ciratefi algorithm took 22s, divided as follows:

- The first filter – Cifi, took 2.5s for the 3D matrix calculation and 4.5s for the correlation process, with total of 7s.
- The second filter – Rafi, took 13s to output its result.
- The final filter – Rafi, was the fastest and took about 1s to output its result.

These times were obtained using a 3GHz Pentium 4.

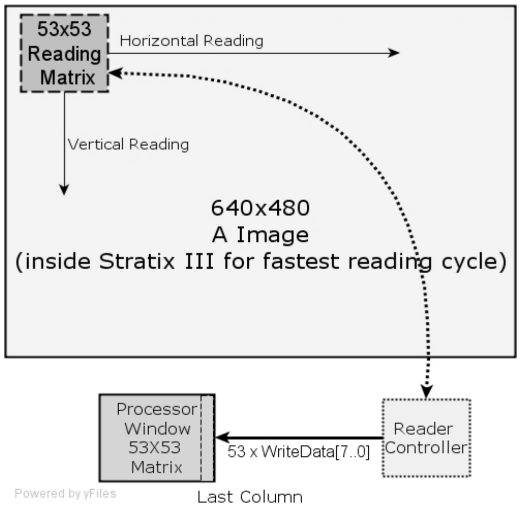


Fig. 4. Architecture of the reading process

3. OBJECTIVES

Our objective is to propose and demonstrate a methodology to design a FPGA system capable of executing in real time, very time-consuming image processing algorithms originally developed for PCs. This strategy was used to design a hardware system (FPGA) that applies to Cifi, the first of the three Ciratefi filters. The solution for Rafi, the second filter, is very similar. Tefi, the third filter is very fast, and it does not make part of the proposed innovation.

The proposed system should be: flexible enough to be used by other similar processing algorithms; independently of the FPGA vendors; and easily adaptable for different input parameters maintaining its performance. In this case, the parameters are the size of the query image *Q* (that is, the size of the window matrix); the number of circles (where the average grayscales are computed); and the number of different scales.

4. FPGA IMPLEMENTATION

To attend both flexibility and performance requirements, we decided to use the high level programming language C to automatically create all the VHDL files needed in every development steps.

The only third-part VHDL codes used in this project are those for computing the square root and the division. All the generated VHDL files can be implemented in any FPGA (depending naturally of the resources of the device).

In this work, we chose to work with Altera devices using its tools for synthesis, router and timing analysis. For simulation, we used ModelSim from MentorGraphics that permits to explore the parallel behavior and analyze the results of the designed VHDL hardware.

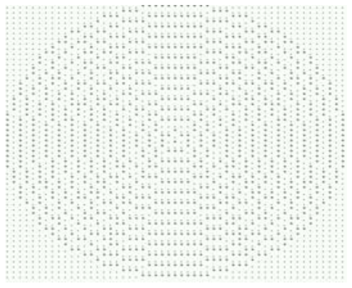


Fig. 5. The window and the circle coordinates calculated by program in C to generate the CWP for Cifi

The Configurable Window Processor

Our system makes use of a hardware module that we named “Configurable Window Processor” (CWP). Another paper [3] uses the same expression, with a very different meaning: The 7×7 CWP proposed in [3] is configurable in the sense it can have different functionalities; in our case, the CWP size can be configured from 23×23 to 53×53 pixels. Even these limits can be easily changed.

The objective of our CWP module is to calculate the sum and averages of the pixels in each circle (Cifi filter) or radial line (Rafi filter), for every pixel of image *A*. To avoid accessing external memories, we chose a FPGA capable to store a full 640×480 grayscale image *A*: the EP3SL340H1152C3 device that has an internal memory of 16 Mbits. Some other papers (for example, [10]) also use the internal memory of the FPGA to store the analyzed image.

Figure 4 represents the hardware architecture to read pixels from image *A* to the CWP. A whole column is written in CWP in every clock cycle. Pixels are written in CWP in just one way, from right to left. This choice simplifies and minimizes the logical size of the module. A similar approach was used in [11], where three small processing windows works in parallel so that the pixels used by a CWP is passed and used by the next one.

The coordinates of the pixels to be averaged are automatically calculated by the VHDL generator (implemented in C). Figure 5 shows a plot of the calculated circles coordinates for the following parameters: CWP of size 53×53 and two pixels distance between two neighboring circles. With this approach, we use neither trigonometric functions in hardware as [12, 13] nor lookup tables as [9] to calculate circles coordinates. As consequence, while COrdinate Rotation Digital Computer (CORDIC) [12, 13] spends some iterations (clock cycles) to calculate each result, our implementation spend one clock cycle per result. The lookup table solution [13] requires much memory resources as the size of the images grows and would be impractical in our high resolution non-simplified images.

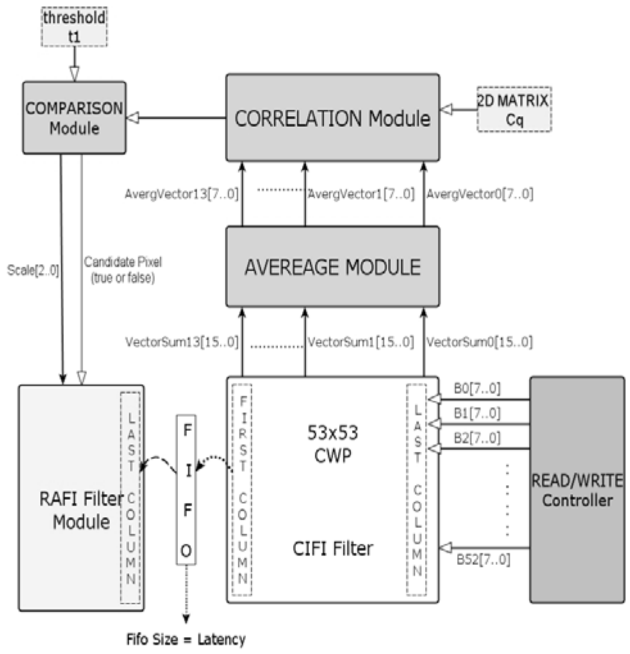


Fig. 6. Interconnections between the two filters

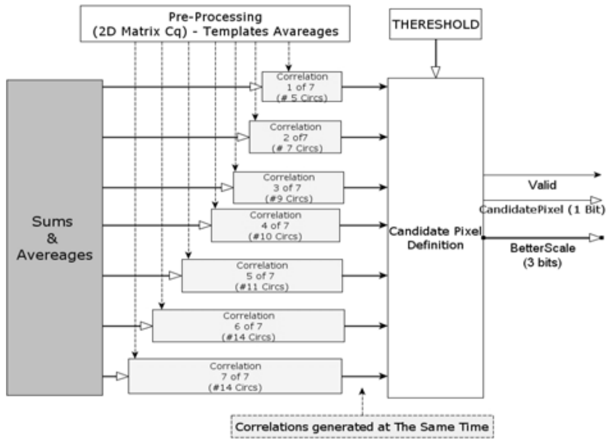

Pre-Processing (2D Matrix Cq) - Templates Avareages
THERESHOLD
Sums & Avereages
Correlation 1 of 7 (# 5 Circs)
Correlation 2 of7 (# 7 Circs)
Correlation 3 of 7 (#9 Circs)
Correlation 4 of 7 (#10 Circs)
Correlation 5 of 7 (#11 Circs)
Correlation 6 of 7 (#14 Circs)
Correlation 7 of 7 (#14 Circs)
Candidate Pixel Definition
Valid
CandidatePixel (1 Bit)
BetterScale (3 bits)
Correlations generated at The Same Time

Fig. 7. Parallel architecture for correlation computation

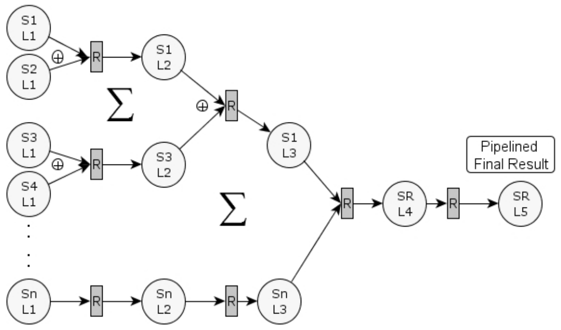


Fig. 8. Pipelined sum's tree

4.1. System Architecture

Figure 6 depicts the different modules of the Cifi hardware architecture. The same figure also demonstrates the interconnections between Cifi and Rafi filters. Implementation of the Rafi filter is left to a second phase.

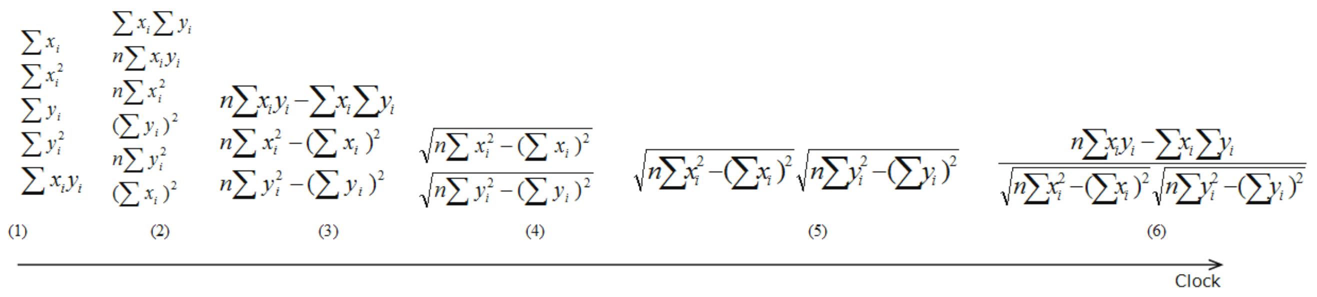
In the proposed approach, the first column of the CWP module (in Cifi filter) is passed to the next filter (Rafi filter) after the first correlation. Thus, it is not necessary for the Rafi filter to wait the end of Cifi to start processing the data. Both filters process simultaneously one pixel at every clock cycle, with an initial latency of 53 clocks (the resolution of the larger template scale). That is, at every clock, one pixel is categorized as a "candidate pixel" or "not candidate". In the former case, the most suitable scale value (Cifi) and the most suitable rotation angle (Rafi) are also computed.

Following the data path, after the computation of the sums of the grayscales in circles in CWP, the average module divides each sum by the number of pixels. Then, using the pre-calculated C_q matrix, the correlations are calculated and the largest correlation is chosen. Finally, we compare the result of the largest correlation with a given threshold to infer if the pixel has some probability of matching the given template. The probable scale is the scale that yielded the best correlation.

To maximize the performance, we pipelined the intermediate results in order to classify one pixel in every clock cycle (after the latency of the system for the initials data). This is why we implemented as many correlation modules as scale factors in the system. Figure 7 depicts this, where one correlation module is instantiated for each template scale. The work [4] uses a similar solution to make many correlations simultaneously. The number of scale factors is configurable. Our program in C generates modules with up to 7 different scale factors.

4.2. Mathematical Calculations in Pipeline

The four mainly used operations in our system are: sum, multiplication, division and square root. The CWP module calculates the average of the sums of grayscales of the pixels in each circle, for all pixels in A (except at tiny image borders). As the high-end FPGAs have multipliers implemented within the device, we multiply the sums by the inverses of the numbers of pixels in each circle to calculate the average grayscales of the circles. Given the parameters, the software in C calculates the coordinates of the pixels in each circle, generating the pipelined sum's tree (Figure 8) and the final division (that we compute by a multiplication) to finally generate the average.



$\sum x_i$ $\sum x_i^2$ $\sum y_i$ $\sum y_i^2$ $\sum x_i y_i$
$\sum x_i \sum y_i$ $n\sum x_i y_i$ $n\sum x_i^2$ $(\sum y_i)^2$ $n\sum y_i^2$ $(\sum x_i)^2$
$n\sum x_i y_i - \sum x_i \sum y_i$ $n\sum x_i^2 - (\sum x_i)^2$ $n\sum y_i^2 - (\sum y_i)^2$
$\sqrt{n\sum x_i^2 - (\sum x_i)^2}$ $\sqrt{n\sum y_i^2 - (\sum y_i)^2}$
$\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}$
$\frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$
(1)
(2)
(3)
(4)
(5)
(6)
Clock

Fig. 9. Intermediary steps to compute correlation coefficients.

For the computation of the correlation coefficient between two vectors ***x*** and ***y*** we use equation:

$$Corr(x,y) = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}, \qquad (2)$$

Figure 9 depicts how the equation (2) is divided into smaller equations to compute the correlation coefficients through pipeline trees. The results are stored as a 16 bits fixed-point variables with all the 16 bits for the fraction part.

5. SIMULATIONS

Until now, we have implemented and analyzed only Cifi, the first of the Ciratefi filters. The implementation of Rafi (the second filter) is similar, and is subject of a future work. However, the hardware infra-structured of the system with both filters is already designed (figure 6). We have compared the outputs of the software implementation of Cifi (that uses float-point variables) with the fixed-point FPGA implementation from ModelSim simulations, and verified that the two are quite similar. Figure 10 depicts the simulation architecture that permits to validate the VHDL hardware using real images.

5.1. Timing Analysis

In this section, we present the timing analysis for each individual module, based on the Altera Stratix III device EP3SL340H1152C3. In Table 1, we present the resource usage and the performance for the worst case (the largest template with 53×53 resolution, the maximum quantity of circles 14, and the maximum number of scales 7). We also present the data for the largest and the smallest correlation module. All frequency informations were obtained with the Classical Timing Analyzer tool (Slow Model Analysis) on the Altera Quartus II software.

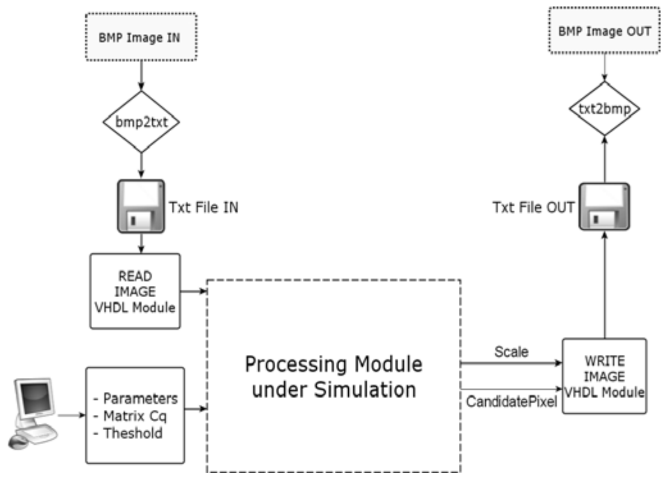


Fig. 10. Simulation Architecture involving the input and output images

After the system latency, the proposed architecture will classify one pixel in each clock as “candidate for matching” or “not candidate”, and will output the probable scale and angle. The processing time for Cifi (that searches a 640×480 image for a 53×53 template) is 1.06ms for the FPGA running at 258 MHz, to be compared with 7s in a 3GHz Pentium.

Table 1. Performance of each module

Module	Maximum Frequency (Mhz)	Size in Logical Elements	Latency
CWP (Sums and Averages)	380	31.258 (12%)	9
Correlations	258	21.538 (1%)	75

Device: EP3SL340H1152C3

6. CONCLUSION

In this paper, we have designed an FPGA system that implements a novel rotation and scale-invariant template matching named Ciratefi. We have actually implemented and tested only the first filter, however the implementation of the second filter is quite similar. To achieve both flexibility and high performance, we created a C language program that automatically generates the VHDL for different parameters. The proposed system has the optimal performance, classifying one pixel as matching or non-matching per clock cycle, and takes 1.06ms to process a frame, 5000 times faster than the software implementation. Seemingly, the same strategy can be used to implement other Image Processing and Computer Vision algorithms.

7. REFERENCES

[1] C. Torres-Huitzil, M. Arias-Estrada, "Real-time image processing with a compact FPGA-based systolic architecture," *Real-Time Imaging* 2004 (177-187).

[2] T. Kean, A. Duncan, "A 800 Mpixel/sec Reconfigurable Image Correlator on XC6216," *Proceedings of FPL'97*, pp. 382-391, 1997.

[3] P. D. Michailidis, K. G. Margaritis. "A programmable array processor architecture for flexible approximate string matching algorithms," *J. Parallel Distrib. Comput., 131 – 141, 2007.*

[4] J. Ferruz, A. Ollero "Integrated real-time vision system for vehicle control in non-structured environments," *Engineering Applications of Artificial Intelligence* 215-236, 2000 .

[5] M. Sen, I. Corretjer, F. Haim, S. Saha, J. Schlessman, S. Bhattacharyya and W. Wolf. "A parallel algorithm for real-time object recognition," *Pattern Recognition 1917–1931, 2002.*

[6] H. Y. Kim and S. A. Araújo, "Grayscale Template-Matching Invariant to Rotation, Scale, Translation, Brightness and Contrast," *IEEE Pacific-Rim Symposium on Image and Video Technology*, Lecture Notes in Computer Science, vol. 4872, pp. 100-113, 2007.

[7] S. A Araújo, H. Y Kim, "Rotation, scale and translation-invariant segmentation-free grayscale shape recognition using mathematical morphology," In: *Int. Symposium on Mathematical Morphology*, 2007.

[8] S. Hezel, A. Kugel, R. Manner, D. M. Gavrila "FPGA-based Template Matching using Distance Transforms," *Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2008

[9] M. Shen, H. Song, W. Sheng, Z. Liu "Fast Correlation Tracking Method based on Circular Projection" Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel / Distributed Computing 235-238, 2007.

[10] A. Lindoso, L. Entrena "High performance FPGA-based image correlation," *J. Real-Time Image Proc. 2:223–233, 2007.*

[11] C. Torres-Huitzil, M. Arias-Estrada, "FPGA-Based Configurable Systolic Architecture for Window-Based Image Processing," *Journal on Applied Signal Processing* 1024–1034, 2005.

[12] V. Bonato, "Proposta de uma arquitetura de hardware em FPGA implementada para SLAM com multi-câmeras aplicada à robótica móvel," Ph.D. Thesis, Universidade de São Paulo, Brazil, 2008.

[13] M. Ma, A. van Genderen, P. Beukelman "A Sign Bit Only Phase Normalization for Rotation and Scale Invariant Template Matching," Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing, ProRisc, pp. 641-646, 2005.