Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 148190, 16 pages
doi:10.1155/2012/148190

Research Article

An FPGA-Based Omnidirectional Vision Sensor for Motion Detection on Mobile Robots

Jones Y. Mori,[1] Janier Arias-Garcia,[1] Camilo Sánchez-Ferreira,[1] Daniel M. Muñoz,[2] Carlos H. Llanos,[1] and J. M. S. T. Motta[1]

[1] *Faculty of Technology, University of Brasilia, 70910-900, Brasilia, DF, Brazil*
[2] *Faculty of Gama, University of Brasilia, 72405-610, Brasilia, DF, Brazil*

Correspondence should be addressed to Jones Y. Mori, jonesyudi@unb.br

Received 20 February 2012; Accepted 4 April 2012

Academic Editor: Alisson Brito

This work presents the development of an integrated hardware/software sensor system for moving object detection and distance calculation, based on background subtraction algorithm. The sensor comprises a catadioptric system composed by a camera and a convex mirror that reflects the environment to the camera from all directions, obtaining a panoramic view. The sensor is used as an omnidirectional vision system, allowing for localization and navigation tasks of mobile robots. Several image processing operations such as filtering, segmentation and morphology have been included in the processing architecture. For achieving distance measurement, an algorithm to determine the center of mass of a detected object was implemented. The overall architecture has been mapped onto a commercial low-cost FPGA device, using a hardware/software co-design approach, which comprises a Nios II embedded microprocessor and specific image processing blocks, which have been implemented in hardware. The background subtraction algorithm was also used to calibrate the system, allowing for accurate results. Synthesis results show that the system can achieve a throughput of 26.6 processed frames per second and the performance analysis pointed out that the overall architecture achieves a speedup factor of 13.78 in comparison with a PC-based solution running on the real-time operating system xPC Target.

1. Introduction

Scientists predict that robots will play an important role in the future. In this scenario, robots will be able to assist humans in many tasks as domestic labors, elderly people care, cleaning, vehicles operation, and surveillance. Animals have mechanisms to interact with the environment provided by natural evolution. They are able to sense the surrounding environment and to move according to a defined objective, contouring obstacles and performing a dynamic path planning. In the robotic field, one of the major challenges is providing robots with sensorial and rational capabilities, allowing them to assist, and possibly substitute, humans in some activities requiring special skills.

Autonomous mobile robot navigation considers the execution of three stages: (a) mapping, (b) localization, and (c) decision making. The first stage uses information from sensors for creating a map of the environment. The second one relates the map with the sensor information, allowing the robot to self-localization in the environment. The third stage considers the path-planning problem [1].

Different kinds of sensors can be used for providing environment information to the mobile robot. Such sensors are classified in two main groups: (a) interoceptive and (b) exteroceptive. The interoceptive sensors perform internal robot parameters measurements without environment dependence. Encoders, gyroscopes, and accelerometers are some examples of interoceptive sensors. On the other hand, exteroceptive sensors perform external measurements, for instance, ultrasound, radar and infrared positioning systems as well as cameras, GPS and magnetometers. In humans, the vision sense is the one which provides more quantity of information about the environment. Through the sensorial fusion (provided by our stereo vision system) we are able to estimate efficiently the localization of surrounding objects.

The use of cameras jointly with image processing algorithms for implementing sensors (e.g., distance, movement, color, and presence sensors) is suitable solution for mobile robotic applications. Additionally, cameras with embedded image processing issues are the foundations of computer vision area. Catadioptric systems are realizations of omnidirectional vision, being mainly based on specially shaped mirrors (e.g., spherical, hyperbolic, parabolic, etc.) that reflect the environment to the camera from all directions, obtaining a panoramic view. Thus, these systems can provide information from a larger area than other vision sensors [2].

The task of processing the acquired images depends on the objective of the process itself. A common problem in mobile robotics is the localization of moving objects around the robot. For that, different methodologies can be used, such as motion detection, trajectory estimation, and tracking. Since the motion detection approach makes use of simple and easily implemented algorithms, this technique is suitable for real-time embedded applications. A common technique for implementing the motion detection is the background subtraction, in which an image is acquired at the beginning of the measurement process, and then each new image is subtracted pixel by pixel from the background. This technique is largely used in surveillance systems, since it acts as an automatic intrusion detection algorithm. In robotics, the localization of the differences between the background and the new frame provides the position estimation of the moving objects around the robot.

On the other hand, distance sensors are important for solving mobile robotic localization problems (namely, local and global localization tasks), and an important issue is the use of cameras for these tasks, providing (in real time) the robot with information about the distance to an obstacle. To accomplish this, the development of a mapping process among the actual scenario and the captured image is fundamental. This aspect introduces the calibration problem, which comprises the estimation of metrological values such as accuracy and precision (that are related to systematic and random errors, resp.), apart from the calculation of calibrations curves.

Otherwise, taking into account performance points, autonomous mobile robots must be able to acquire images from the environment, processing the information and making a decision in a short period of time. In order to avoid failures, autonomous mobile robots must perform the decision process as quickly as possible. This real-time constraints require the use of high-performance computational platforms for implementing image processing algorithms. In this context, the high computational cost of the involved algorithms is the main drawback, specifically when performing operations with high accuracy and high performance.

Common robotic platforms are based on desktop solutions executing complex algorithms for robot navigation. However, desktop platforms are not tailored for embedded applications with portability and low-power consumption requirements. Field programmable gate arrays (FPGAs) are a suitable solution for implementing image processing algorithms with a high performance. FPGAs allow the involved algorithms to be mapped directly in hardware in a parallel way. In addition, FPGAs allow software RISC processors to be implemented in order to execute parts of the algorithms with low performance requirements.

In [3] the authors proposed the development of a distance sensor based on an 800×480 pixels camera connected to an FPGA, a spatial convolution filter (for edge enhancement), a hardware architecture for estimating the distance of real objects and a touch-screen display as user interface (the camera image was addressed to the screen). In that system the screen was capable to detect the coordinates of a touched point, being used for calculating the distance (in pixels) from the robot to a defined object. In this approach the calibration parameters (errors and calibration curves) were calculated by comparing the actual distances, in a particular scenario, and the pixel distance in the screen.

The main contribution of this work is the design of an integrated hardware/software sensor system for both moving object detection and distance calculation, based on background subtraction algorithm. In this approach the calibration problem has been also treated, validating a proposed calibration process, which is suitable for this kind of application. Several image processing operations as filtering, segmentation, and erosion have been implemented in the architecture. The object's position was determined by computing its center of mass coordinates. The movement detection technique was also used to automate the omnidirectional vision system (achieved in a catadioptric implementation). As a result the uncertainties related to point objects in the touchscreen were eliminated.

The proposed pipeline image processing algorithm was mapped onto a Cyclone II FPGA device. Also, in this device, a NiosII soft processor was implemented for computing the distance and orientation as well as a simple user interface. Execution time comparisons among the proposed hardware architecture and a C-code implementation show that the hardware solution speeds up by 13.78 times a pc-based solution running in an Intel Pentium IV processor at 2.2 GHz, with 2.0 GB RAM and using a real-time operating system (the xPC Target OS from MathWorks).

The remainder of this paper is organized as follows. Section 2 outlines some computational vision techniques. Section 3 presents the related work. Section 4 describes the system and the calibration procedure. Section 5 shows the FPGA hardware and software implementations, and, before concluding, Section 6 presents synthesis, validation results, and a performance analysis.

2. Background

The use of cameras is a common solution for mobile robotics applications. Different works are related to the extraction of features from images. Monocular systems are able to provide only 2D information of the environment in front of the camera. In this case, in order to extract depth information it may be necessary to analyse vanishing points or to use

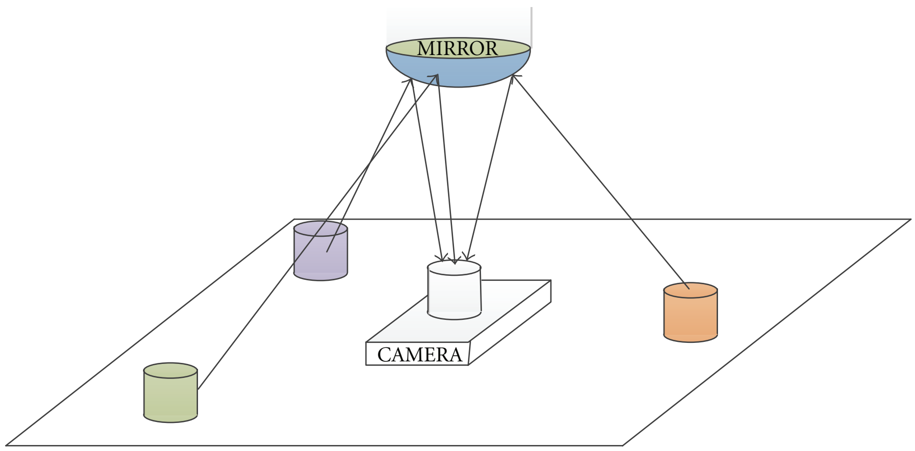


Figure 1: Catadioptric system.

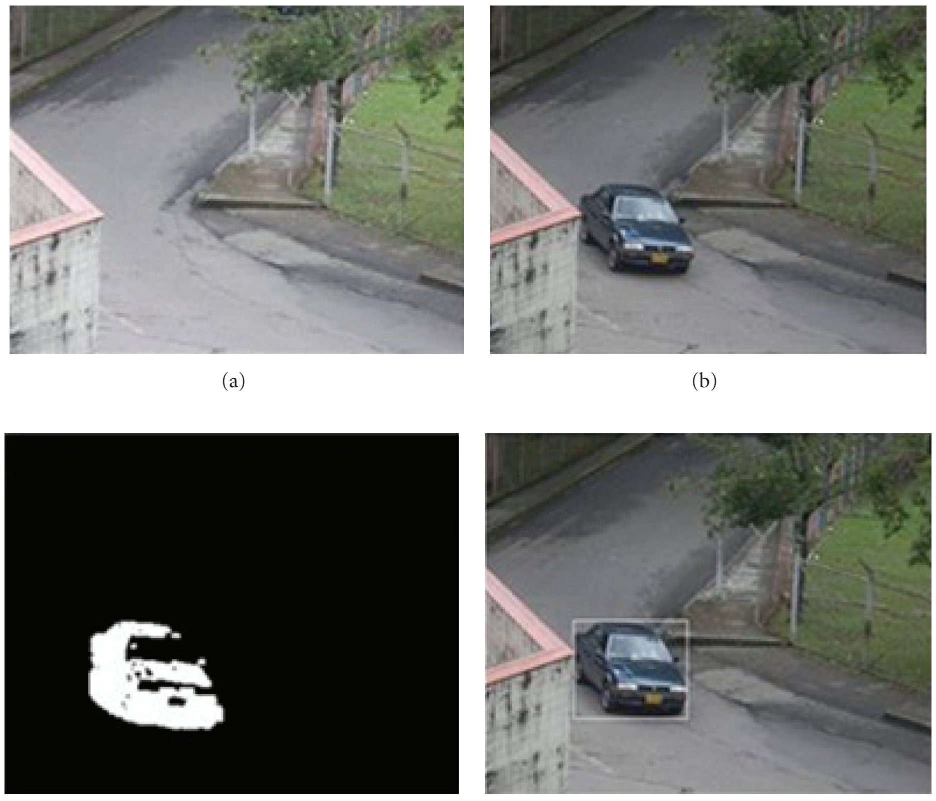


Figure 2: Image chain of the background subtraction algorithm: (a) background image; (b) new image; (c) image subtraction; (d) overlap of new image and object's position.

perspective models with known object shapes. The use of a pair of cameras (stereo systems) allows the depth to be estimated through the epipolar geometry. Once the objects have been identified in each camera, it is possible to compute the depth of the image using simple geometric techniques. However, similar to monocular systems, stereo cameras provide information only in front of the cameras. In order to obtain information about the surrounding environment it is necessary to turn the system 360 degrees acquiring images in all the directions [4].

Omnidirectional vision systems are a suitable alternative to view the surrounding environment from one single image. Such a system can be built in two ways: (a) using panoramic lenses, for instance, fisheye lenses and (b) using catadioptric systems. Cameras equipped with fisheye lenses acquire images directly from the environment. On the other hand,

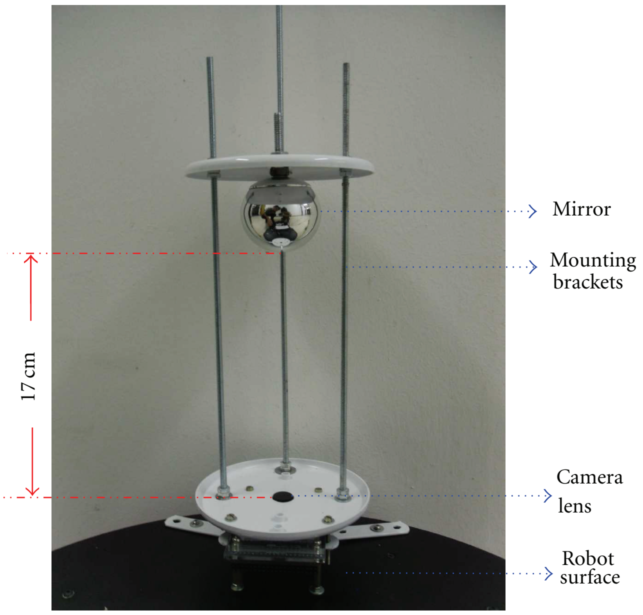


Figure 3: Main components of the proposed catadioptric system.

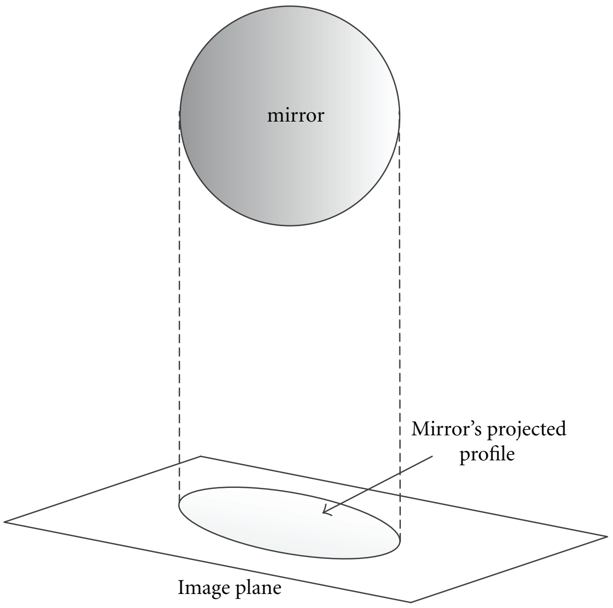


Figure 4: Mirror's profile projected over image plane.

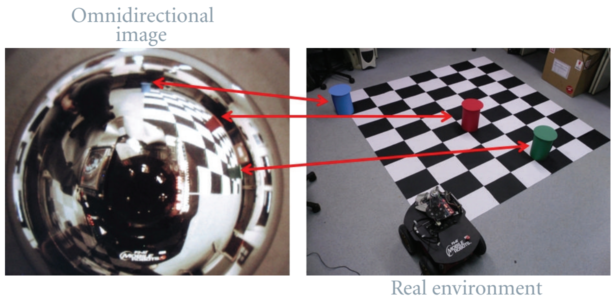


Figure 5: Correspondences among the omnidirectional image and the real environment.

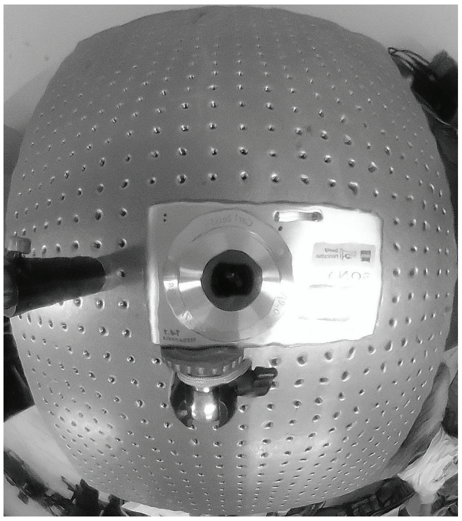


Figure 6: The catadioptric system mounted over a calibration board.

catadioptric systems capture the image of the environment reflected on a special geometry mirror [5].

Figure 1 shows the principles of operation of a catadioptric system. It is composed of a camera and a convex mirror. The acquired image characteristics depend on the geometry of the mirror. Thus, knowing the geometry, reflection equations can be used in order to determine the environment geometric characteristics. Commonly, omnidirectional vision systems make use of hyperbolic, parabolic, spherical, and conical mirrors [5, 6].

In our previous work [3] the object's coordinates in the image were determined using a touch screen. Thus, by touching over the object in the image, the distance in pixels was calculated. Although this procedure is easily performed, its precision depends on several factors such as parallax effect, accuracy, and lighting variations. In order to minimize the human factor, in this work we make use of an identification method based on a background subtraction algorithm which allows the object to be automatically located and identified.

Figure 2 shows an image chain demonstrating the response of the background subtraction algorithm. In this method an image without objects is acquired (background image), and then each new image (with objects) is subtracted pixel by pixel from the background. After the subtraction (Figure 2(c)), the coordinates of the center of mass are determined using (1) and used as the object location. It is important to note that the system can determine the position of only a single object in the image, since all other moving objects will be considered noise and disregarded.

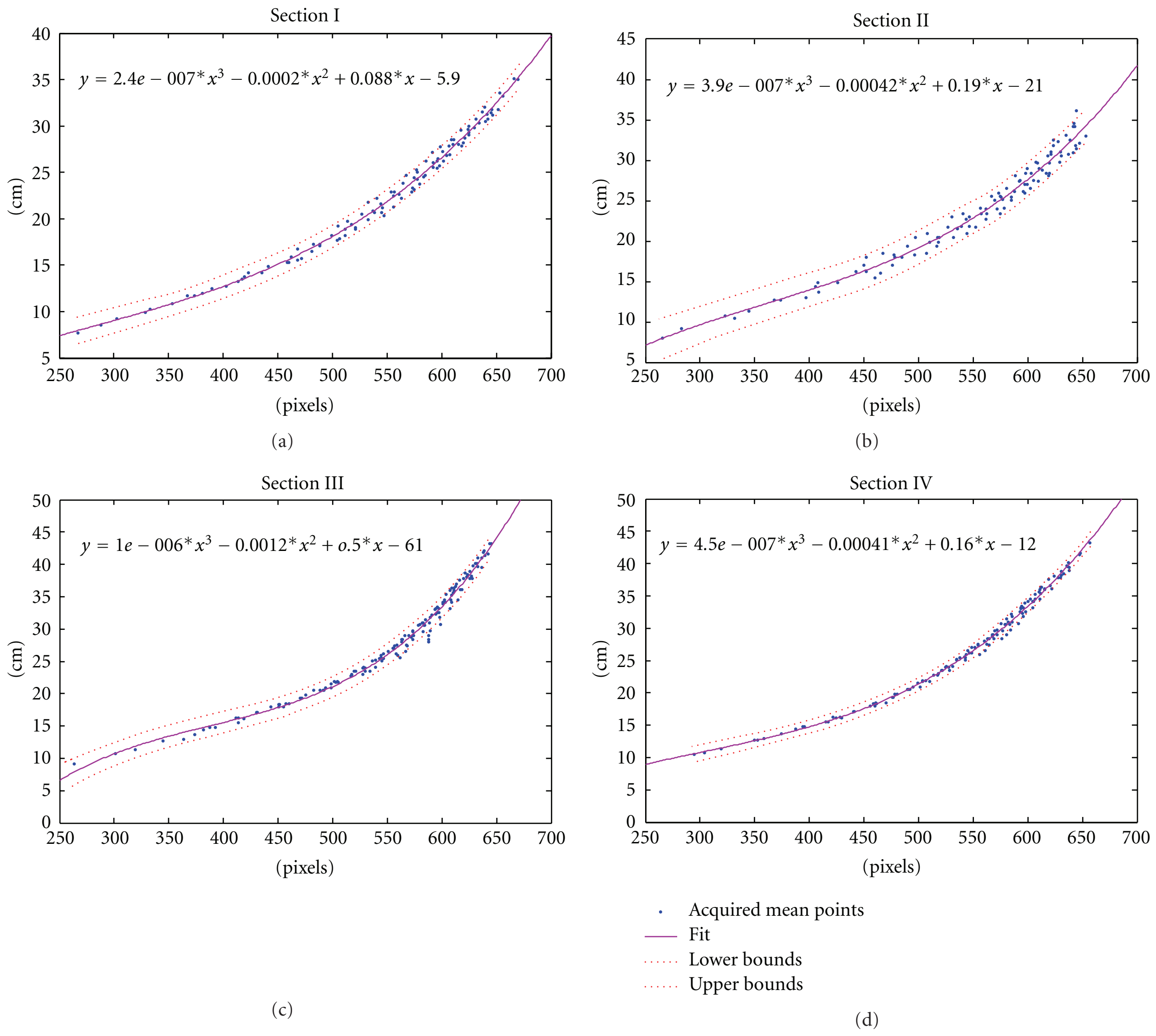


Figure 7: Polynomial fitting for distance estimation for sections I, II, III and IV.

Otherwise, the center of mass will be determined considering the distributed mass of all objects:

$$C(x \cdot y) = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (i, j) \cdot B(i, j)}{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} B(i, j)}. \quad (1)$$

The construction of catadioptric systems is a complex task taking into account that there are a lot of of geometrical uncertainties that must be precisely determined in order to assure the necessary accuracy of the system. In this context, it is essential to apply a calibration process to the catadioptric system for determining the errors (namely, systematic and random ones) related to defects in the catadioptric systems (e.g., mirror defects).

3. Related Works

Several works have been developed using FPGAs for speeding up image processing tasks, mainly for embedded systems applications with real-time constraints. In [7], a biological inspired architecture for motion estimation by optical flow was implemented. This approach is suitable for implementation in both FPGA and ASIC devices, achieving a processing rate of 177 frames per second (128×96 pixels). Also, in [8] an FPGA implementation of an embedded motion estimation sensor (that uses an optical flow algorithm achieving 15 frames per second for images with 640×480 pixels size) is proposed.

A vision system for visual feedback applied to control a mechanical system was proposed in [9]. In this approach a matched filter by correlation is used, which also determines the object's center of mass each 4,51 ms for small images (256×256 pixels). In [10], a system for image

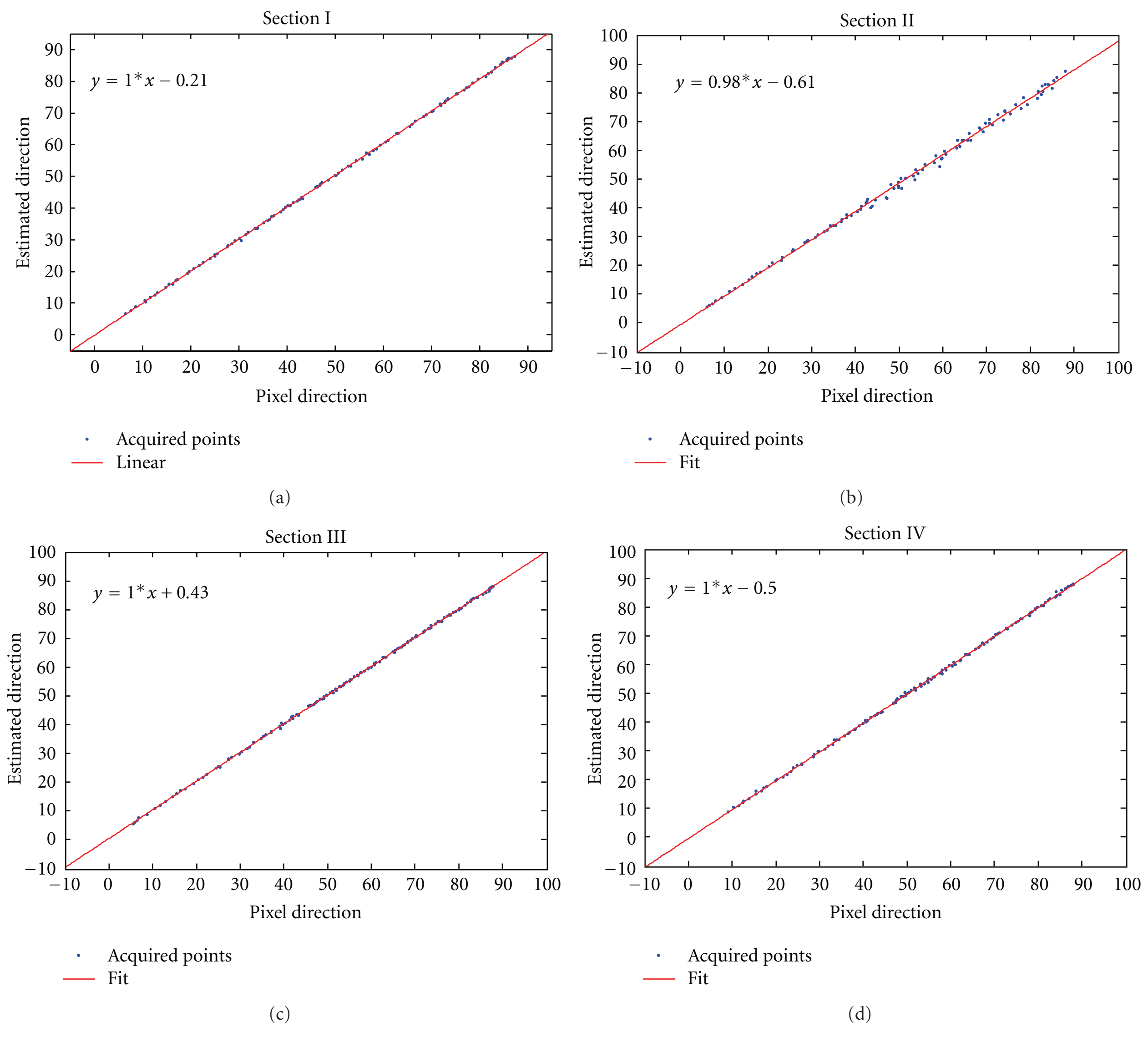


Figure 8: Polynomial fitting for direction estimation for sections I, II, III, and IV.

filtering and motion estimation using SAD (sum of absolute differences) is implemented using a systolic architecture suitable for estimating motion each 5 ms in images with 640×480 pixels.

The design and implementation of robust real-time visual servoing control, with an FPGA-based image coprocessor for a rotary inverted pendulum, are presented in [11]. In this approach the position of the pendulum is measured with a machine vision system whose image processing algorithms are pipelined and implemented on a FPGA device for achieving real-time constraints. Furthermore, it uses an edge enhancement algorithm to determine the center of mass of the detected object, reaching a throughput of 580 (128×101 pixels) processed frames per second.

In [12, 13] an FPGA-based video processing for surveillance systems is described. Reference [13] shows an FPGA implementation for real-time background subtraction. The implemented architecture reaches a performance of 32,8 frames per second with 1024×1024 images. In [12] a pipeline architecture for multimodal background generation algorithm is described, for colour video stream and moving objects segmentation based on brightness, colour, and textural information. In the later case, the overall throughput was about 25 frames per second, with a resolution of 720×576 pixels.

An implementation in a PC of an omnidirectional sensor for mobile robot navigation was presented in [14]. In this approach, a catadioptric system was calibrated by placing landmarks in the environment, using a polynomial interpolation to characterize the system. Additionally, the same uses an edge enhancement technique to create a polar map of the surrounding environment.

Reference [15] introduces an implementation of a framework for image processing speedup by using reconfigurable devices. Some of the most common image preprocessing algorithms were implemented achieving a high throughput.

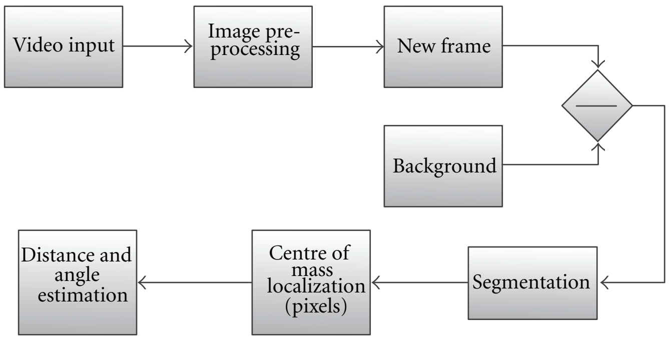

Video input
Image pre-processing
New frame
Background
Segmentation
Centre of mass localization (pixels)
Distance and angle estimation

Figure 9: Flowchart of the general hardware architecture.

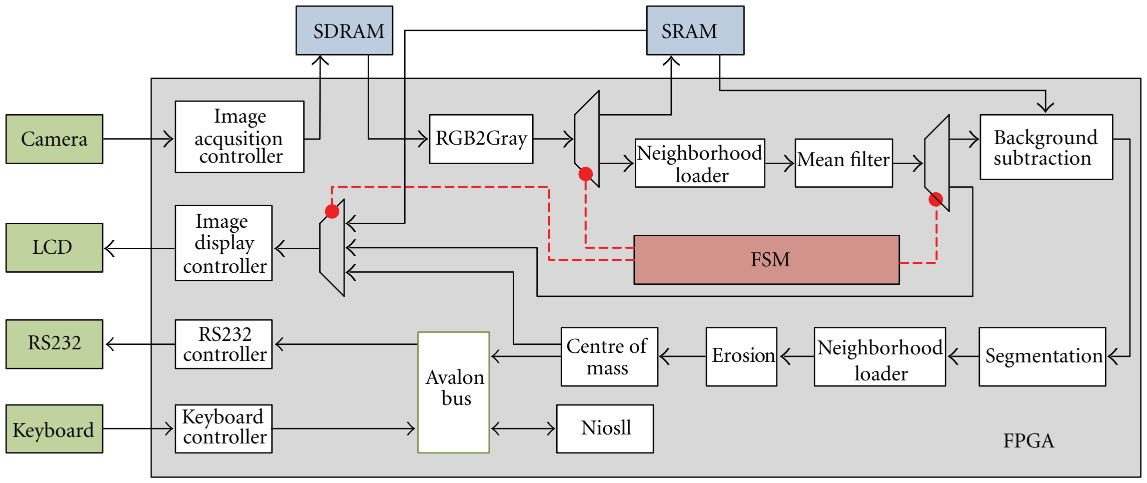


Figure 10: Hardware implementation of the image acquisition block.

Otherwise, [16] uses dynamic reconfiguration for color recognition and optical flow computation in FPGA, in which a throughput of 30 frames per second for 160 × 120 pixels images is achieved.

An approach that uses lookup tables for avoiding complex computations was proposed in [17], in which an architecture for real-time rectification of catadioptric images was implemented in FPGAs. This work has a good throughput but requires a large amount of memory block for storing all the distances, which were calculated off line.

In [18] an FPGA-based architecture (which calculates pixel by pixel the undistorted image from a polar frame) is proposed, thus providing a plane image as output. In that case, a pipeline architecture is used to organize the processing stages, achieving a throughput of one pixel per clock cycle. An FPGA for image reconstruction was used in [19]; however, differently from [17, 18], where the images are generated by an omnidirectional mirror, the system processes the images from a camera with a fisheye lens, although, in this case, the authors do not present a description of the architecture for the reconstruction process.

An architecture using a mixed FPGA/DSP to obtain large speedup factor for high-resolution images was presented in [20], while in [21] an embedded Nios II processor that makes use of several hardware coprocessors for image filtering and tasks related to the autoadjustment of the camera focal length was described.

In [6] a complete procedure for catadioptric systems calibration using line projections is presented. In this case, the system achieves a high accuracy for paraboloid mirrors. An approach that uses calibration patterns to determine the response of hyperbolic sections on a nonrevolute hyperbolic mirror was presented in [22]. The simple idea that the external and internal boundaries of the mirror can be used as a 3D calibration pattern was proposed in [23], allowing for a high-speed self-calibration procedure. [24] which developed a complete generic camera calibration procedure by overlapping calibration grids simultaneously. It is suitable for calibrating many kinds of cameras such as fisheye lens, catadioptric cameras with spherical and hyperbolic mirrors, and also multicamera setups.

Some of the cited works use FPGAs for accelerating omnidirectional vision processing and have mainly focused on image undistortion/reconstruction/rectification tasks. However, in mobile robot applications (such as localization, navigation, and multiagent robotics), it is not always necessarily a complete image reconstruction, but the correct, appropriated, and fast measurement of distances between the robot and the different environment objects. In this context, the main contribution of this work is to provide the robot

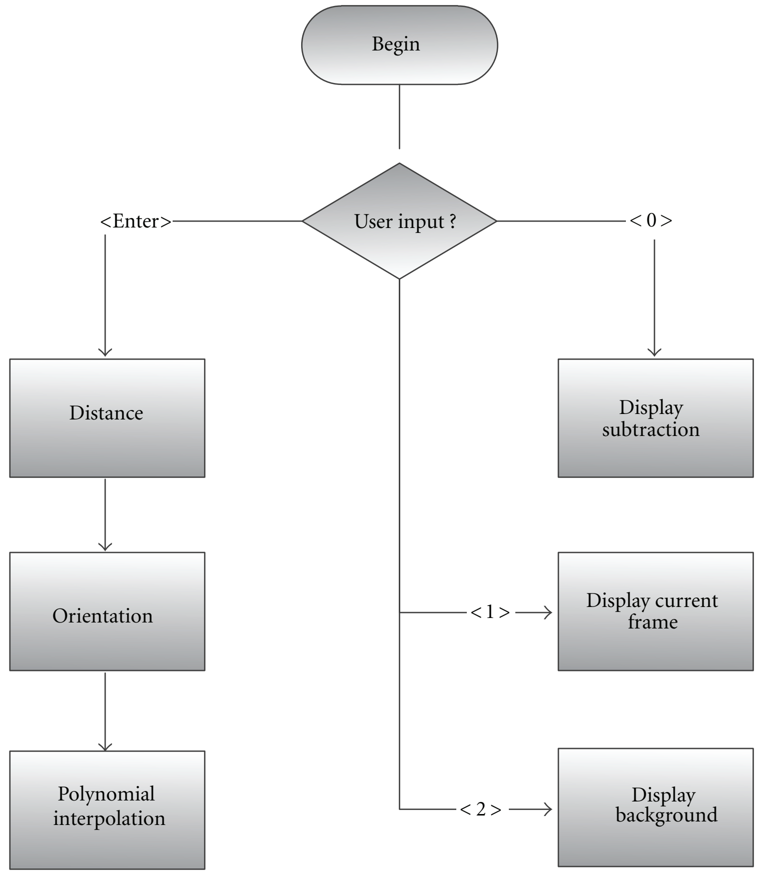


Figure 11: Flowchart of the NiosII coprocessor.

with a suitable and low-cost system for measuring distances automatically from the robot to the detected objects, using an appropriated image resolution (800 × 480 pixels) and achieving real-time characteristics.

4. Development of the Catadioptric System

This section describes the mechanical design of the catadioptric system as well as the calibration procedure description and its analysis.

4.1. The Proposed Catadioptric System. A catadioptric system allows the camera to capture reflected images in the mirror, obtaining a panoramic view. Figure 3 shows the system developed and its main components: (a) a convex mirror, (b) the mounting brackets, and (c) a CMOS camera with a maximum resolution of 2592 × 1944 pixels. In order to perform numerical comparisons with related works we have used a resolution of 800 × 480 pixels, which is appropriated for mobile robotics applications. The distance between the camera and the vertex of the mirror is approximately 17 cm.

In this case it is desirable that the center of the mirror is placed in a vertical line from the center of the camera lens. To achieve this, once mounted the system, an image was acquired and analyzed in order to determine (in pixels) the coordinates of the circle projected by the mirror over

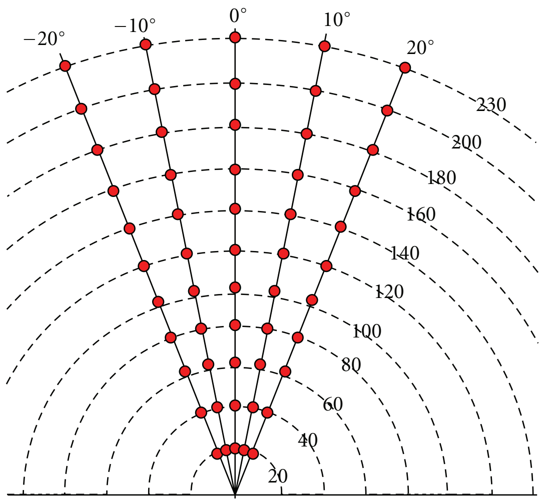


Figure 12: Calibration positions for a particular section in the scene.

the image plane (Figure 4 shows this idea). Otherwise, in this approach, the mirror's projection was assumed to be circular.

For mobile robot applications, the catadioptric system provides a panoramic image in which the robot occupies

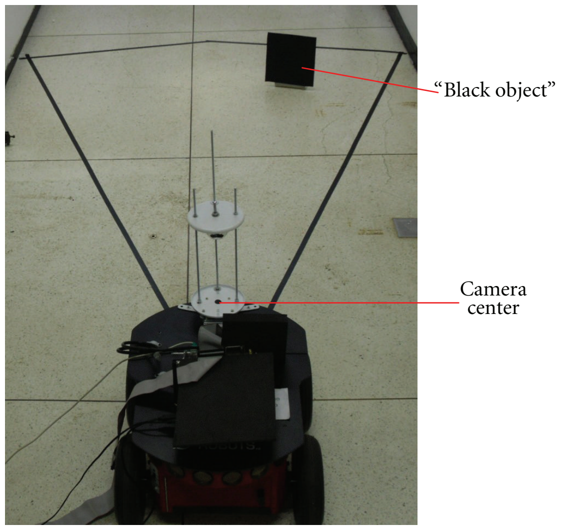


Figure 13: The overall system and the calibration environment.

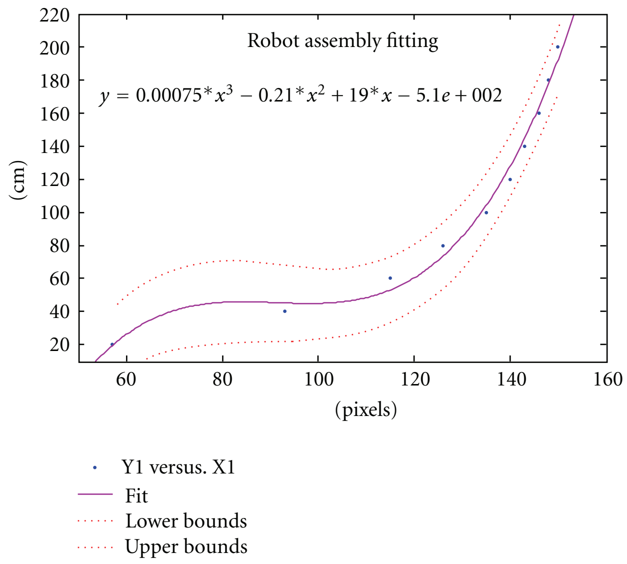


Figure 14: Polynomial interpolation functions.

the center of the image. Figure 5 shows the omnidirectional image captured and the respective environment, in which the robot is positioned with some objects around it, with arrows indicating the correspondence among objects and the image.

The omnidirectional vision system provides a panoramic image, which can be processed in order to extract the distance in pixels between the center of the mirror and the identified object. Once the system has been calibrated, the actual distance can be computed by using a mathematical model (explained in the following section). Additionally, by using omnidirectional vision it is possible to estimate the direction of the surrounding objects, providing to the robot a polar representation of the environment. We assume that the robot only detects objects within its viewing area, which corresponds to a circle with a radius of 2.0 meters. Objects outside this area are reflected on the mirror border, and then large distortions are produced. Additional distortions in the acquired image can be produced by small errors in the optical geometry of the vision system.

Table 1: RMSE for each section.

Section	Distance RMSE	Direction RMSE
I	0.9284	0.244
II	1.038	1.110
III	0.8353	0.273
IV	0.5398	0.388

4.2. Calibration Process. The quality of the data obtained from an omnidirectional vision system depends directly on several constructive parameters such as the optical geometry, curvature of the convex mirror, and quality of mirror's surface. For one to use the equations of the mirror surface profile (and afterward modeling the light reflection), the geometrical parameters of the system must be precisely characterized. However, in this work we have used a convex mirror with unknown geometry, and then it is not possible to use light reflection equations.

The proposed calibration process allows the whole vision system to be characterized providing a fitting function relating the distance in pixels with distances in world coordinates. The calibration procedure was performed by associating objects placed at measured distances with the distances estimated in the image in pixels, obtaining a polynomial fitting associated to a particular section of the convex mirror.

As in our previous work [3], the image was divided into four sections, and the same calibration procedure was executed to each one. In order to provide a better demonstration of the correctness of the calibration procedure, in this work a calibration board and a 14 megapixels camera have been used. The mounted system is depicted in Figure 6. The board used has a separation of 2,54 cm between each hole (in both horizontal and vertical directions). The catadioptric mount was positioned approximately in the center of the board, and the images are shown in Figure 6.

In this image, the holes were detected and their pixel coordinates were determined. The image was divided in four sections, and for each section the real distance and the distance in pixels were associated with a polynomial fitting. In order to allow the robot to identify the position of any object in the surrounding environment, the system was also calibrated to estimate the direction of the detected object. Figures 7 and 8 show the polynomial fitting for estimating both distances and directions, respectively, for each section of the mirror.

Table 1 shows the RMSE (root mean-square error) values of the fittings which characterize the quality of the obtained models. It can be observed that the polynomial fittings have a low value of the RMSE, which means that the calibration data can be well modeled by the polynomials.

In order to validate the precision of the measurement system, a new image (with the same assembly of Figure 6)

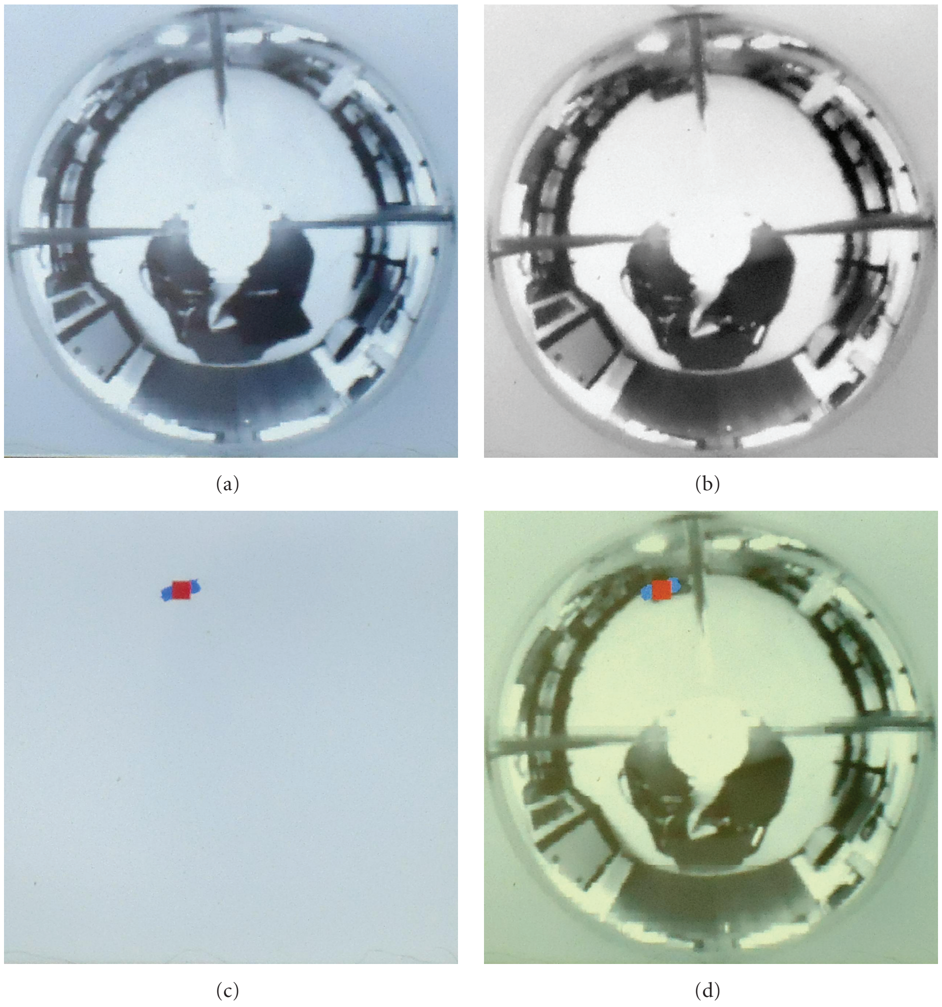


Figure 15: Results of the image processing chain: (a) background, (b) new frame, (c) subtraction, and (d) overlap.

was shot, and five points were picked from each section. By using the polynomial fittings (used like calibration curves), their actual positions were estimated. Table 2 shows the actual and the estimated positions of the points (distance and direction).

The validation results (shown in Table 2) demonstrate that both the calibration procedure and the polynomial fitting can be effectively used to relate distances/directions in the image (in pixels) with the actual distances/directions values.

5. FPGA Implementation

The proposed algorithms for image processing were implemented in hardware, using both VHDL and Verilog hardware description languages. Figure 9 shows the general architecture for motion detection, which is composed of several hardware components (blocks) connected in a pipeline way. The first processing step receives from the camera an RGB 800 × 480 pixels image with a resolution of 8 bits per color channel. At the second step, namely, *image processing*, a gray-scale image is obtained and a *mean* filter is applied for eliminating noise. At the third step, the background subtraction is performed. To do this the background image has been previously stored in an SRAM. At the fourth step, a thresholding algorithm is applied for segmentation, obtaining a binary image (only one bit per pixel). Additionally, the obtained image is eroded in order to minimize noise. At the fifth stage the center of mass is computed, and, finally, at the sixth stage the object position (distance and orientation) is computed.

5.1. Image Acquisition and Color Conversion. The system uses a CMOS camera which provides synchronism and data signals in an RAW format. A color conversion process is performed by calculating the RGB data from RAW ones and storing it in an external SDRAM (see Figures 5 and 6).

5.2. Mean Filter Implementation. After the pixel conversion from RAW to RGB format, a gray-scale transformation is applied. Afterward, a *neighborhood loader* block provides a 3 × 3 neighborhood to a *mean filter*, which eliminates

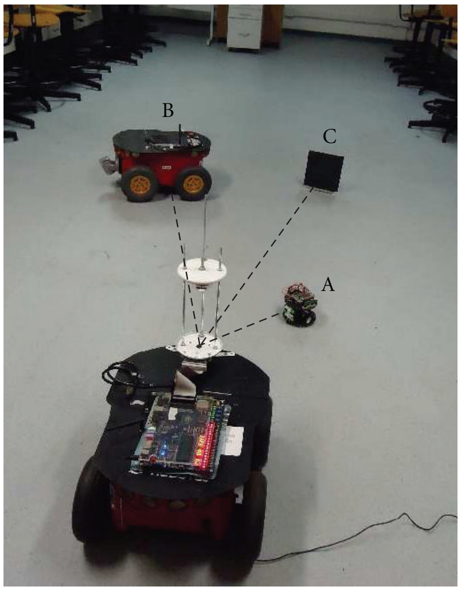


Figure 16: Objects used for testing the overall system.

high-frequency noises. The neighborhood loader operation requires an initial latency of 1603 clock cycles (69.62 μs) [25]. A convolution operation was used to implement the mean filter (see Figures 5 and 6), which is performed in one clock cycle by multiplying the mask with the neighborhood and then yielding the sum of the products, after an initial latency. More details on this implementation and the convolution architecture can be found in [25, 26].

5.3. The Background Storage and Image Subtraction. The background image is stored in an external 512 Kbyte SRAM memory (chip ISSI IS61LV25616AL) of the DE2 development kit. Once the mean filter is performed, the subtraction between the current frame and background is computed, providing one output pixel per clock cycle. Afterward, the absolute value of each pixel is calculated. Finally, the segmentation operation is performed by a simple thresholding operation (see Figures 5 and 6).

5.4. The Erosion Operation. The erosion computation is based on logic operations between the pixel of a binary image and a structuring element as shown in (2). The erosion block receives nine pixels from the neighborhood loader (f_i), as well as the structuring element (K_i) (a square mask was used like structuring element). Therefore, the e_i values are calculated in the first equation. Afterward, they are used in the next equation in order to perform a complete erosion operation. Both steps are performed in one clock cycle. In this work we have used a neighborhood of nine elements; therefore, $i = 1, \dots, 9$:

$$\begin{gathered} e_i = \overline{K}_i \cdot \overline{f}_i + \overline{K}_i \cdot f_i + K_i \cdot f_i, \\ \text{Erosion} = e_1 \cdot e_2 \cdot e_3 \cdot e_4 \cdot e_5 \cdot e_6 \cdot e_7 \cdot e_8 \cdot e_9. \end{gathered} \tag{2}$$

Table 2: Validation points.

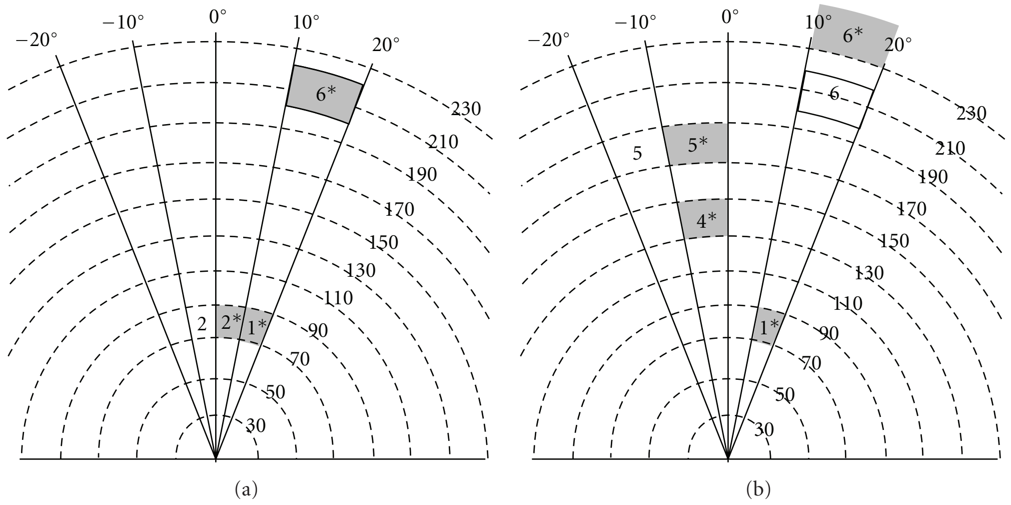
Section	Actual distance (cm)	Estimated distance (cm)	Actual direction (°)	Estimated direction (°)
I	20.5	19.6	7.6	7.2
I	11.3	12.1	69.5	68.4
I	17.0	18.0	66.2	67.4
I	21.7	23.3	71.4	72.7
I	28.4	29.4	65.8	65.0
II	28.4	25.8	10.8	11.5
II	27.4	25.2	22.8	23.6
II	27.0	25.7	50.9	53.6
II	30.6	29.6	50.2	50.9
II	26.2	24.8	30.5	31.8
III	17.9	18.0	8.8	7.9
III	28.4	31.8	10.8	9.8
III	29.7	31.9	20.9	19.4
III	34.5	35.9	37.3	35.7
III	43.3	41.5	51.1	48.9
IV	15.5	16.1	10.3	10.2
IV	23.4	24.4	13.2	13.6
IV	29.6	30.3	32.3	32.8
IV	35.9	36.0	46.5	46.6
IV	43.3	43.7	51.1	51.8

5.5. The Center of Mass Calculation. In this work only the detection of a single object is performed at a time. In order to calculate the center of mass, $C(x, y)$, (1) was used, where $B(i, j)$ is a binary image and i and j are the positions of pixels on the image. Since the algorithm has to explore the overall image, the center of mass is calculated at each frame.

5.6. Distance and Orientation Estimation. In our previous work [3] the actual and pixel distances were computed using several floating-point arithmetic libraries [27]. However, a large consumption of hardware resources was observed, specially for embedded applications. In this work we have chosen an embedded software implementation for these computations allowing for cost reduction in logic area.

A Nios II soft processor (from Altera) has been used in order to execute the following tasks: (a) receiving commands from the user through a PS2 keyboard, (b) calculating the distance in pixels and orientation values, (c) calculating the actual distance using a polynomial function obtained from the calibration data (see Figure 14), and (d) sending to the host (PC or robot computer) both the estimated distance and orientation values. The communication with the PC is done through a RS232 communication standard. The image processing architecture and a keyboard are connected to the Nios II using the Avalon Bus from Altera, as shown in Figure 10.

Figure 11 shows the flowchart algorithm implemented in the NiosII processor. The processor receives the user input



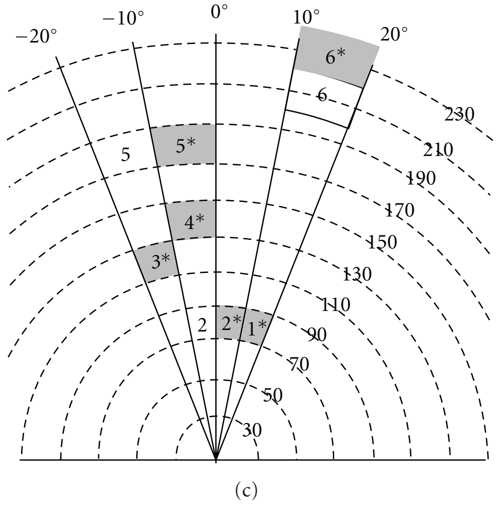


Figure 17: Estimation of distance and orientation. (a) object A, (b) object B, and (c) object C. White grids are the real positions and gray grids are the estimated position.

commands from a keyboard, allowing the selection of the following options.

(i) Send to the host the distance and orientation values. In this case the NiosII receives from the hardware architecture the center of mass coordinates $C(x, y)$ and computes the euclidean distance in pixels and the orientation θ using the atan() function. Finally, the polynomial interpolation is used to estimate the actual distance R, sending the polar coordinates (R, θ) to the host.

(ii) Capture a new background image. This option allows the user to upgrade the background image in the SRAM memory.

(iii) Subtract the background from current image. This option allows the user to manually execute the subtraction image step.

(iv) Display the subtraction result. This option allows the user to show the image substation result in the display.

(v) Display the current image. This option allows the user to address the current camera image to the display.

(vi) Display the background image. This option allows the user to see the stored background image in the display.

The last five commands make use of the Avalon bus to send the respective commands to the FSM (finite state machine) which controls the data flow and the LCD. The FSM was implemented in hardware for controlling the overall system operation according to the user inputs. The same controls several multiplexers for addressing the incoming pixels to the SRAM/SDRAM memories and the hardware modules previously described.

In our approach both the Euclidean distance as well as the arctangent function need to be computed (but only once time at each frame). In this case, although von Neumann-based architectures have serious restrictions (such as the NiosII case) for real-time image processing, specially for

Table 3: Synthesis results (chip ep2c35f672c6).

Implemented core	LC 33216	MB 483840	DSP18×18 35	Freq. 250 MHz
Entire Architecture	9953 (30.0%)	359352 (74.3%)	8 (22.8%)	10.2
Image Acquisition	2161 (6.5%)	57400 (11.86%)	0 (0%)	45.31
Gray scale Conversion	498 (1.5%)	384 (0.08%)	1 (1%)	45.31
Neighborhood Load	681 (2.05%)	16256 (3.36%)	0 (0%)	104.12
Spatial Convolution	1853 (5.0%)	16256 (3.36%)	3 (5%)	104.12
Background Subtraction	16 (0.05%)	0 (0%)	0 (0%)	250
Segmentation	15 (0.05%)	0 (0%)	0 (0%)	250
Erosion	722 (2.17%)	0 (0%)	0 (0%)	321.3
Center of mass	2521 (7.59%)	0 (0%)	0 (0%)	10.2
NiosII	3615 (10.9%)	285696 (59.0%)	4 (6%)	250.0

Table 4: Latency of the motion detection architecture.

Implemented core	Latency
Background Subtraction	2 clk
Segmentation	1 clk
Erostion	1603 clk
Center of mass	384.001 (1 frame)
Entire architecture	385.607

embedded systems, the software implementation of these tasks attends the real-time constraints in this work.

6. Results

The proposed architectures for processing the images from the omnidirectional vision system were effectively implemented in a Cyclone II FPGA device using the Quartus II development tool.

6.1. Synthesis Results. Table 3 presents the synthesis results of the overall architecture and its main components. The cost in logic area is presented in terms of logic cells (LCs), memory bits (MBs), and embedded DSP 18 × 18 blocks consumption. The performance of the architectures is presented in MHz.

It can be observed that the entire architecture consumes 30% of logic cells and around 74% of the memory bits. The image acquisition block requires the largest number of memory bits due to the fact that the camera provides pixels in an RAW format; therefore, this block needs to store several rows in order to convert the pixels to an RGB format. The neighborhood loader is composed of two line buffers, which are used for providing the 3 × 3 pixels for performing the neighborhood operations [25]. As described in Section 5, the spatial convolution block makes use of a neighborhood loader module leading to more memory bits consumption. As expected, the NiosII implementation requires a large

Table 5: Distance calibration data using mobile object detection.

Distance (cm)	Mean distance (pixels)	σ (pixels)
20	56.6	3.3
40	93.4	7.3
60	114.6	1.3
80	126.2	7.3
100	134.6	7.3
120	139.8	1.7
140	143.0	1.0
160	146.2	2.2
180	147.8	1.7
200	149.6	1.3
230	152.2	1.4

Table 6: Direction calibration data using mobile object detection.

Angle (°)	Mean angle (°)	σ (°)
−20	−24.8	4.8
−10	−14.4	11.6
0	−1.3	1.1
10	9.3	4.4
20	21.1	4.5

Table 7: Validation results (using the moving object detection technique) distance is given in cm.

Real distance	Estimated distance and error					
	A	e_A	B	e_B	C	e_C
$p_1 = 70$	70	0%	80	14%	83	19%
$p_2 = 74$	70	5%	—	—	83	12%
$p_3 = 115$	—	—	—	—	107	7%
$p_4 = 137$	—	—	132	4%	132	4%
$p_5 = 170$	—	—	170	0%	170	0%
$p_6 = 200$	200	0%	248	24%	235	17%

Table 8: Validation results for orientation in degrees (°), using mobile object detection.

Angle (°)	A	B	C
$p_1 = 20$	20	19	20
$p_2 = -1$	9	—	9
$p_3 = -18$	—	—	−11
$p_4 = -8$	—	−4	−3
$p_5 = -14$	—	−8	−7
$p_6 = 11$	16	15	16

Table 9: Performance comparison.

Author	Year	Main algorithm	Image resolution	Frames per second	Megapixels per second
[10]	2009	Sum of absolute differences (SAD)	640 × 480	200	61.44
[13]	2012	Background subtraction	1024 × 1024	32	33.55
[9]	2006	Correlation	256 × 256	221	14.48
[12]	2011	Background subtraction	720 × 576	25	10.37
This work	*2012*	*Background subtraction*	*800 × 480*	*26*	*9.98*
[11]	2011	Edge enhancement	128 × 101	580	7.50
[8]	2008	Optical flow	640 × 480	15	4.61
[7]	2008	Optical flow	128 × 96	177	2.17
[15]	2007	Optical flow	160 × 120	30	0.58

amount of memory bits for storing the program memory and its hardware architecture.

The background subtraction block is based on a pixel by pixel operation. The segmentation block operates using a comparator and a multiplexer. The erosion block is based on simple logic operations. Therefore, these blocks have a small hardware resources consumption.

It can be observed that the entire architecture operates at a maximum frequency of 10.2 MHz. Taking into account that the system is based on a pipeline architecture and the images resolution is 800 × 480 pixels, the system achieves a performance of 26.6 frames per second.

It is important to point out that the selected FPGA chip is not the largest device from the Cyclone II family; therefore, one can expect a performance improvement when using modern FPGA devices with more hardware resources.

Since the proposed system is based on a pipeline architecture, it is necessary for several clock cycles (*latency time*) before computing the first result (center of mass). According to Table 4 the latency of the proposed architecture is around 385.607 clock cycles. Note that the center of mass has the largest latency due to the fact that the entire image must be analyzed for computing the area.

6.2. The Calibration of the Overall System Using the Moving Object Detection Technique (Figure 13). The process of using a polynomial fitting for associating distances in pixels with actual distances was validated in Section 4. However, in real environments, mobile robots commonly operate with large uncertainties associated to the odometry sensors and nonholonomic restrictions. Therefore, we have again calibrated the catadioptric system including the mobile robot, using a precision lower than shown in Section 4. To do that, the moving object detection technique implemented in the FPGA has been used for calibration tasks. In this case, several objects were introduced in the scene, and the detection system automatically defined the distance between the robot and the detected object. The calculated distances can be compared with the actual ones in the scene, yielding new polynomial functions.

For this case, the mirror was divided into sections of 40° (nine sections cover the whole mirror surface). In the scene, a grid of radial distances were chosen to vary from 20 cm to 230 cm around all sections in a circle. Figure 11 shows the calibration positions for a particular section in the scene. A black object was positioned over the red dots. The calibration environment is shown in Figure 12, in which the robot, the catadioptric system, and the object are presented. It is important to point out that the environment does not suffer from natural light variations; only artificial illumination was used, and the white floor allows for a high contrast with the black object.

Tables 5 and 6 show a statistical analysis of the experimental data. According to Figure 4, each object located in the scene had its projected position on the image calculated (in pixels) for 5 different orientations in the chosen section, in the form of mean distance and its standard deviation, summing up 55 positions in a section.

As expected in Section 5, the interpolation function has a monotonically increasing behavior. It is important to notice that, as an effect of the reflection characteristics of this system, for greater distances the objects appear smaller than in common acquisition systems (e.g., nonomnidirectional ones). That occurs due to the reflection angle for objects placed far away from the mirror. In this case, it can be observed that the variance values for each distance do not follow the monotonic behavior. This is because the error in distance estimation is not a function of the distance but is mainly determined by mirror's surface quality.

The direction of the detected object is determined by calculating the arctangent function using the estimated coordinates of the center of mass (of the detected object). Notice that the uncertainty in direction estimation is related to the uncertainty in object detection. To calculate the direction, the values in pixels have been used; therefore, both the distance estimation and direction estimation are independent tasks.

Figure 14 depicts the behavior of the pixel/centimeter transformation by using a polynomial interpolation for each one of the regions.

6.3. Validation Results. In order to demonstrate the system running in an actual scene, Figure 15 shows an example of the processing chain, in which Figure 15(a) represents the background image. Figure 15(b) shows an object around the mobile robot. Figure 15(c) depicts the background subtraction and the position of the center of mass. Finally, Figure 15(d) overlaps the current image and the detected center of mass.

Several experiments have been performed so as to evaluate the accuracy of the implemented system. To do that, three different objects (namely, *A*, *B*, and *C*) which corresponds to (a) a small cylindrical robot, (b) a pioneer mobile robot, and (c) the calibration object, respectively, were placed at different positions in front of the robot. Figure 16 depicts the objects used for testing the overall system. The distances and orientation between the center of the camera and the objects have been previously measured (the actual values) in the arena. Additionally, both the estimated distances and orientation were sent to the host (via RS 232 interface) and compared with the actual values.

Table 7 presents the location results and respective errors for each object. It can be observed that the proposed architecture achieves more accurate results when detecting the object *A*. It can be explained because the size and shape of the cylindrical robot produce a small shadow, leading to a better accurate. As expected, large approximation errors were achieved for large distances. For instance, results for localization of objects *B* and *C* show the largest errors (around 24% and 17%, resp.). This fact is explained given that the spherical mirror produces large distortions to the light rays reflected from the uppermost surface of the mirror. This distortion is produced by a compression effect, as the farther the object is, the smaller is its projection on the mirror surface.

Table 8 presents the orientation estimation for each object. Figure 17 uses occupation grids in a polar graph form for summarizing the achieved results of distance and direction estimation. The gray grid represents the estimated position of the object. As expected, the system produces large errors for large distances (see point 6 for objects *B* and *C*).

One can conclude that the omnidirectional system performs better for estimating distances than orientation. Notice that the calibration data (see Table 8) show large errors for orientation values. As explained in Section 4 the system requires a large contrast between background and objects. Therefore, when the system operates with a low contrast, some errors in the object borders are introduced. However, these errors can be overcome by using more efficient techniques for motion segmentation (e.g., optical flow).

6.4. Performance Analysis. Additionally, the same algorithm for motion detection was implemented in a PC, running at 2.2 GHz, 2.0 GB RAM using a real-time xPC Target OS from MathWorks. The average elapsed time for processing a 10×10 pixel image was around 138.1 μs, (value of the average TET (task execution time)). Thus, an output pixel is processed in 1.381 μs. Therefore, the proposed hardware architecture, operating at 10.2 MHz, achieves a speedup factor of 13.78 in comparison with the real-time software solution.

As cited in Section 3, several works have been developed to solve the problem of object's position estimation for real-time applications. Table 9 shows the comparison among some works and our approach. Each proposal uses different algorithms and image resolutions, leading to difficulty in the comparison task. Therefore, we have used the overall throughput (megapixels per second) as a comparison metric.

It is important to note that all systems listed in Table 9 have as output the estimated position of an object in the image. In this case, all listed systems have implemented FPGA-based hardware architectures to process the image and determine object's position.

7. Conclusions

This work has presented a FPGA-based omnidirectional vision system for mobile robotic applications. It takes advantage of a pipeline approach for processing the polar image, using a background subtraction algorithm. The overall latency of the motion detection architecture is 385.607 clock cycles, and after this latency, the system has a throughput of 26 frames per second (running at 10.2 MHz). The proposed architecture is suitable for robot localization, allowing to compute the distance between the robot and the surrounding objects.

The architectures were described in VHDL and Verilog and successfully implemented in a Cyclone II FPGA device. Synthesis results have demonstrated that the proposed hardware achieves an operational frequency around 10.2 MHz. In addition, the pipelined architecture allows the image to process one pixel per clock cycle after an initial delay. This fact demonstrated acceleration of 13, 78 times in comparison with the same algorithm in C using a xPC Target OS from MathWorks implementation running on a common desktop platform.

This work has also addressed several calibration problems related to omnidirectional vision systems (based on a catadioptric implementation) of mobile robotic applications, especially the application of a technique for detection of mobile objects for mirror calibration tasks. Experimental results show that the results are consistent with the expected ones.

Concerning our future work we intend to analyze the power consumption of the proposed architecture. It is an important issue in order to validate the effectiveness of the implemented algorithms for real-time image processing in portable applications.

Acknowledgments

The authors would like to thank CAPES Foundation and National Council of Scientific and Technological Development of Brazil-CNPq (processes 133501/2010-8 and 142033/2008-1) for the financial support of this work. Special thanks are to Altera Corp. for providing Quartus II Licenses and to DHW Engenharia e Representação for the partnership.

References

[1] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, MIT Press, Cambridge, Mass, USA, 2004.

[2] L. Spacek and C. Burbridge, "Instantaneous robot self-localization and motion estimation with omnidirectional

vision," *Robotics and Autonomous Systems*, vol. 55, no. 9, pp. 667–674, 2007.

[3] J. Yudi Mori, D. Mũoz Arboleda, J. N. Arias Garcia, C. Llanos Quintero, and J. Motta, "FPGA-based image processing for omnidirectional vision on mobile robots," in *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design*, pp. 113–118, João Pessoa, Brazil, 2011.

[4] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[5] K. Daniilidis and C. Geyer, "Omnidirectional vision: theory and algorithms," in *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 1, pp. 89–96, 2000.

[6] C. Geyer and K. Daniilidis, "Catadioptric camera calibration," in *Proceedings of the 17th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 398–404, September 1999.

[7] G. Botella, M. Rodriguez, A. Garca, and E. Ros, "Neuromorphic configurable architecture for robust motion estimation," *International Journal of Reconfigurable Computing*, vol. 2008, Article ID 428265, 9 pages, 2008.

[8] Z. Wei, D. Lee, N. Brent, J. Archibald, and B. Edwards, "FPGA-based embedded motion estimation sensor," *International Journal of Reconfigurable Computing*, vol. 2008, Article ID 636145, 9 pages, 2008.

[9] K. Shimizu and S. Hirai, "CMOS+FPGA vision system for visual feedback of mechanical systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, pp. 2060–2065, May 2006.

[10] G. Saldaña-González and M. Arias-Estrada, "FPGA based acceleration for image processing applications," in *Image Processing*, 2009.

[11] Y. Tu and M. Ho, "Design and implementation of robust visual servoing control of an inverted pendulum with an FPGAbased image co-processor," *Mechatronics*, vol. 21, no. 7, pp. 1170–1182, 2011.

[12] T. Kryjak and M. Gorgoń, "Real-time implementation of moving object detection in video surveillance systems using FPGA," *Computer Science*, vol. 12, pp. 149–162, 2011.

[13] R. Rodriguez-Gomez, E. Fernandez-Sanchez, J. Diaz, and E. Ros, "FPGA implementation for real-time background subtraction based on horprasert model," *Sensors*, vol. 12, pp. 585–611, 2012.

[14] R. Chojecki and B. Siemiatkowska, "Mobile robot navigation based on omnidirectional sensor," in *Proceedings of the European Conference on Mobile Robots (ECMR '03)*, pp. 101–106, Radziejowice, Poland, September 2003.

[15] M. A. Vega-Rodríguez, A. Gómez-Iglesias, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez, "Reconfigurable computing system for image processing via the internet," *Microprocessors and Microsystems* , vol. 31, pp. 498–515, 2007.

[16] F. Nava, D. Sciuto, M. D. Santambrogio et al., "Applying dynamic reconfiguration in the mobile robotics domain: a case study on computer vision algorithms," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, no. 3, 2011.

[17] L. Chen, M. Zhang, B. Wang, Z. Xiong, and G. Cheng, "Real-time FPGA-based panoramic unrolling of high-resolution catadioptric omnidirectional images," in *Proceedings of the International Conference on Measuring Technology and Mechatronics Automation (ICMTMA '09)*, pp. 502–505, Hunan, China, April 2009.

[18] A. Gardel, A. Hernández, R. Miota, I. Bravo, and R. Mateos, "Correction of omnidirectional camera images using reconfigurable hardware," in *Proceedings of the 32nd Annual Conference on IEEE Industrial Electronics*, pp. 3403–3407, Paris, France, November 2006.

[19] B. Zhang, Z. Qi, J. Zhu, and Z. Cao, "Omnidirection image restoration based on spherical perspective projection," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*, pp. 922–925, Macao, China, December 2008.

[20] T. Shu-ren, Z. Mao-jun, X. Zhi-hui, L. Le, and C. L. Dong, "Design and implementation of high-resolution omnidirectional vision system," *Chinese Journal of Video Engineering*, vol. 10, no. 1, pp. 1–6, 2008.

[21] A. Maeder, H. Bistry, and J. Zhang, "Towards intelligent autonomous vision systems—smart image processing for robotic applications," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '07)*, pp. 1081–1086, Sanya, China, December 2007.

[22] R. Benosman, E. Deforas, and J. Devars, "A new catadioptric sensor for the panoramic vision of mobile robots," in *Proceedings of the IEEE Workshop on Omnidirectional Vision*, pp. 112–116, 2000.

[23] J. Fabrizio, J.-P. Tarel, and R. Benosman, "Calibration of panoramic catadioptric sensors made easier," in *Proceedings of the 3rd Workshop on Omnidirectional Vision*, pp. 45–52, 2002.

[24] S. Ramalingam, P. Sturm, and S. K. Lodha, "Towards complete generic camera calibration," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 1093–1098, June 2005.

[25] J. Y. Mori, C. Sánchez-Ferreira, D. M. Munoz, C. H. Llanos, and P. Berger, "An unified approach for convolution-based image filtering on reconfigurable systems," in *Proceedings of the 7th Southern Conference on Programmable Logic (SPL '11)*, pp. 63–68, Crdoba, Argentina, April 2011.

[26] J. Mori, *Implementação de técnicas de processamento de imagens no domínio espacial em sistemas reconfiguráveis*, M.S. thesis, Universidade de Brasília, Brasília, Brazil, 2010.

[27] D. M. Muñoz, D. F. Sanchez, C. H. Llanos, and M. Ayala-Rincón, "Tradeoff of FPGA design of a floating-point library for arithmetic operators," *Journal of Integrated Circuits and Systems*, vol. 5, no. 1, pp. 42–52, 2010.