FPGA Implementation of Log-polar Mapping

Wai Kit Wong, Chee Wee Choo, Chu Kiong Loo and Joo Peng Teh
Faculty of Engineering and Technology, Multimedia University, Jln Ayer Keroh Lama,
75450 Melaka, Malaysia. Tel:+60 06-2523044
Email:wkwong@mmu.edu.my;cwchoo@mmu.edu.my;ckloo@mmu.edu.my;teh_joo_peng@yahoo.com

***Abstract*-Log-polar or spatially-variant image representation is an important component of active vision system in tracking process for many robotic applications due to its data compression ability, faster sampling rates and hence, direct to faster image processing speed in machine vision system. In this paper, we try to implement log-polar mapping techniques on Xilinx FPGA (Field Programmable Gate Array) board for unwarping the omnidirectional images into panoramic images to present a wide angle of view by preserving fine output image quality in a higher data compression manner. Simulations are also run on MATLAB to find out the optimum mapping criterion. Some significant advantages of this new approach are: lighter processing system, lesser space utilization, cost saving, faster processing speed and faster reset time (boot time) compared to a laptop computer that uses MATLAB for doing the unwarping process.**

***Keywords* – log-polar mapping, FPGA implementation, robotic application, machine vision.**

I. INTRODUCTION

Tracking process is important for many robotic applications such as surveillance, navigation, motion estimation, vehicle stabilization, mosaic creation etc. The attempt to use robots in omnidirectional view's environment raised the need for better sampling and image process techniques to keep track of the changes occurring in those dynamic environments. In such a way, log-polar imaging geometry can play an important role.

Log-polar imaging geometry is a spatially variant image representation in which pixel separation increases linearly with distance from a central point [1]. It is like human eye visualization, whereby it distributes into fovea portion and periphery portion for a sampled image. Computational resources on region of interest will be concentrating on the fovea portion, whilst retaining low resolution information will be distributed in the periphery portion. In contrast to conventional Cartesian images, the log-polar images have data compression ability and allow faster sampling rates on vision systems without reducing the size of the field of view and the resolution on the fovea portion [2]. It has been noticed in [3] that the use of log-polar geometry also provides important algorithmic benefits. In [4] and [5], the authors shown that the use of log-polar images increase the size range of objects that can be tracked using simple translation model and also in planar structure model respectively. All these studies agree and provide the advantages of using log-polar imaging geometry in tracking process for robotic applications.

The present paper builds on the theoretical work of Fabio Berton [6] to explore the implementation of log-polar mapping on Xilinx FPGA (Field Programmable Gate Array) board for unwarping the omnidirectional images into panoramic images to present a wide angle of view by preserving fine output images qualities in a higher data compression manner. If a mobile robot is intended to be used for surveillance by using a single video camera plus a hyperbolical optical mirror for capturing an omnidirectional view, then the implementation provided in this paper can be embedded on the said robot. The same can be applied on navigation, motion estimation and other robot applications too. Some significant advantages of the implementation are: it is much lighter, utilizes less space, cheaper, faster processing speed and faster reset time (boot time) compared to a laptop computer that uses MATLAB for doing the unwarping process.

This paper is organized in the following way: Section II will briefly summarize the log-polar image geometry and the mapping techniques used to unwarp the omnidirectional images into panoramic images. Then in Section III, we will simulate log-polar mapping techniques in MATLAB to unwarp omnidirectional images into panoramic images. Simulation results run on MATLAB will be investigated and the optimum mapping criterion is determined here. In Section IV, we will briefly describe what is FPGA and the implementation of log-polar mapping on it. The performance of FPGA output based on some omnidirectional images will be commented and comparison will be done on weight, size, cost, processing speed and boot time with laptop computer that uses MATLAB for doing the unwarping process. Finally in Section V, we draw some conclusions and envision future developments.

II. LOG-POLAR MAPPING

Conventional images are usually acquired by using conventional cameras that have uniformly distributed rectangular arrays of sampling units. Due to the conventional cameras' architecture, most traditional robotic vision systems tend to use Cartesian representations to handle images. However, for human eye visualization, this is not the way images are acquired. Log-polar imaging geometry is a biological inspired approach to human vision whereby it transforms the original two dimensional captured image into a spatially variant (retina –like) representation.

In human eye, the retina deals with the reception and transmission of input photometric information to neuron

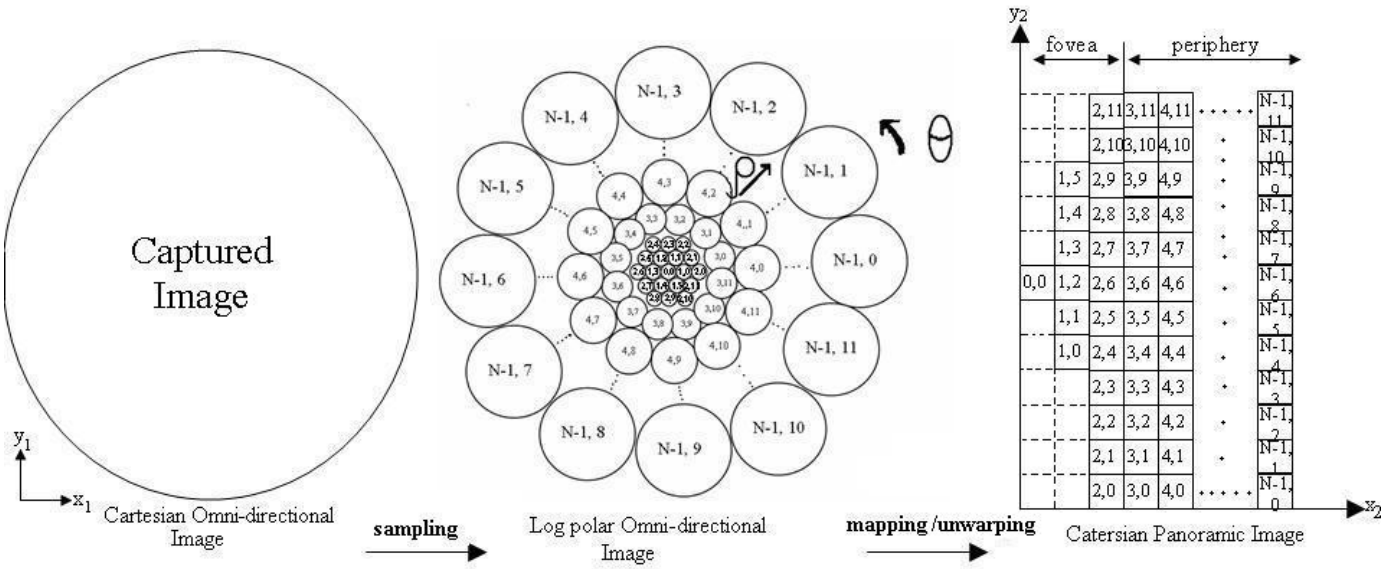


Figure 1. An example of log-polar mapping of a N rings retina containing 3 rings fovea. To simplify the figure, no overlapping was use. Note that periphery portion is log-polar.

layers located on the photoreceptive surface. Each individual neuron on the neuron layers will receives the outputs of a group of photoreceptive cells on an approximately circular area of the retina, so called the receptive field [7]. Generally, the Cartesian image in robotic vision can be resample to retina-like image through the use of a mask consisting of concentric rings of circular receptive fields whose centers are geometrically spaced from the center of the mask, inspiring the retina. The retina can be roughly divided into two distinct regions: fovea and periphery. The fovea region (central portion of the retina) is formed by approximately constant size receptive fields and organized in hexagonal form. In the periphery region, the receptive fields are circularly distributed with an area exponentially increasing as a function of the distance to the retina center. In bio-inspired retina image, overlapping may exist among the receptive fields in order to prevent some Cartesian areas from not being covered by the retina transformation (due to the circular geometry of the receptive fields) [7].

In robotics, there has been a trend to design and use true retina-like sensors (e.g. SVAVISCA log-polar camera [8] and CMOS foveated image sensor [9]) or simulate the log-polar images by software conversion (e.g. application to face detection and tracking [10], application to monocular active tracking [11], etc). In the software conversion of log-polar images, practitioners in pattern recognition usually named it as log-polar mapping. An example illustrating the log-polar mapping of a hypothetical N rings retina containing a 3 rings fovea is shown in Fig.1.

The log-polar mapping can be summarized as following: Initially, omnidirectional image is captured using a conventional camera and a hyperbolic mirror. The geometry of the captured omnidirectional image is in Cartesian form (x_1,y_1). Next, log-polar sampling is uses to sample the Cartesian omnidirectional image into log-polar form (ρ,θ) omnidirectional image. After that, the log-polar omnidirectional image is mapped to another Cartesian form (x_2,y_2) whereby in this process, the omnidirectional image is unwarped into a panoramic image. Since the panoramic image is in Cartesian form, subsequent image processing task will become much easier.

The center of pixel for log-polar sampling is described by [6]:

$$\rho(x_1, y_1) = \log_\lambda \frac{R}{r_o} \tag{1}$$

$$\theta(x_1, y_1) = \frac{N_\theta}{2\pi} \arctan \frac{y_1}{x_1} \tag{2}$$

The center of pixel for log-polar mapping is described by [6]:

$$x_2(\rho, \theta) = \lambda^\rho r_o \cos\left(\frac{2\pi\theta}{N_\theta}\right) \tag{3}$$

$$y_2(\rho, \theta) = \lambda^\rho r_o \sin\left(\frac{2\pi\theta}{N_\theta}\right) \tag{4}$$

where R is the distance between the given point and the center of mapping $= \sqrt{x_1^2 + y_1^2}$

r_o is a scaling factor which will define the size of the circle at $\rho(x_1, y_1) = 0$.

λ is the base of the algorithm,

$$\lambda = \frac{1 + \sin\frac{\pi}{N_\theta}}{1 - \sin\frac{\pi}{N_\theta}} \tag{5}$$

N_θ is the total number of pixels per ring in log-polar geometry.

The number of rings in the fovea region is given by [6]:

$$N_{fov} = \frac{\lambda}{\lambda - 1} \tag{6}$$

To sample the Cartesian pixels (x_1, y_1) into log polar pixels (ρ, θ), at each center point calculated using (1) and (2), the corresponding log-polar pixel (ρ_n, θ_n) is covering a region of Cartesian pixels with radius:

$$r_n = \lambda r_{n-1} \quad (7)$$

where n=0, 1, …, N-1. Fig. 2 shows the conventional circle sampling method of log-polar mapping [9, 12].

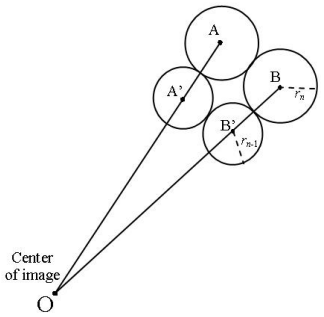


Figure. 2: Conventional circle sampling method for log polar image.

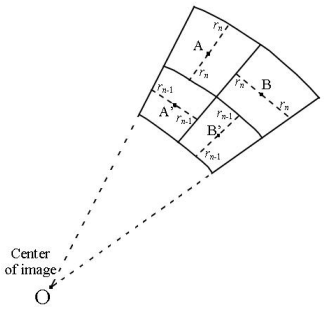


Figure 3: Sector sampling method for log-polar image.

One of the disadvantages of using circle sampling is that certain region of Cartesian pixels outsides sampling circles did not cover by any log-polar pixels. Therefore, some researchers [10, 13, 14] had come out with sector sampling method as shown in Fig. 3, which could maximize the coverage of Cartesian pixels for each log polar pixel. The region of Cartesian pixels covers by an individual log-polar pixel will has the same color intensity follow the respective original Cartesian center sampling point.

During unwarping process, the (ρ, θ) pixels will map to each corresponding (x_2, y_2) pixels as shown in Fig. 4.

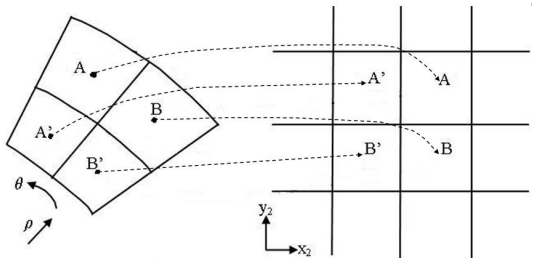


Figure 4. Unwarping process

III. MATLAB SIMULATION

In this section, we will perform computer simulation on log-polar mapping techniques. The simulation is run by using MATLAB version 7.0 on an Intel Core 2 Duo laptop computer with specifications: 1.83 GHz processor, 2 GB DDR2 RAM. The MATLAB program will first read in a Cartesian omnidirectional image as shown in Fig. 5. It is a 256 X 256 $(N_{x_1} \times N_{y_1})$ pixels sample image taken from [15]. Next the sample image is sampled into log-polar form using equation (1) and (2) in MATLAB program to calculate out the center of pixels for the log-polar geometry and perform sector sampling method to sample the log polar omnidirectional image. We consider 4 arbitrary cases of the log-polar samping:

(a) 256 X 256 $(N_{x_1} \times N_{y_1})$ Cartesian pixels into 256 X 256 $(N_\rho \times N_\theta)$ log-polar pixels.
(b) 256 X 256 Cartesian pixels into 256 X 128 log-polar pixels.
(c) 256 X 256 Cartesian pixels into 128 X 128 log-polar pixels.
(d) 256 X 256 Cartesian pixels into 128 X 64 log-polar pixels.

where N_{x_1} is the total number of pixels in x_1-axis in Cartesian omnidirectional image geometry.
N_{y_1} is the total number of pixels in y_1-axis in Cartesian omnidirectional image geometry.
N_ρ is the total number of pixels per radius in log-polar omnidirectional image geometry.
N_θ is the total number of pixels per ring in log-polar omnidirectional image geometry.

The scaling factor, r_o is set to a normalized value, which is 1. In each of the cases stated above, the base of logarithm, λ = 1.0248 for case (a), 1.0503 for case (b) and case (c), and 1.1032 for case (d). The number of rings for the fovea (N_{fov}) is calculated using (6), i.e. for case (a) $N_{fov} = 42$ rings, for case (b) and case (c) $N_{fov} = 21$ rings each, and in case (d), $N_{fov} = 11$ rings.

The corresponding log-polar omnidirectional images are shown in Fig. 6a, Fig. 6b, Fig. 6c, and Fig. 6d respectively.

Figure 5. 256 X 256 Cartesian omnidirectional image

After sampling, the log-polar omnidirectional image is mapped back into Cartesian coordinate geometry, with the total number of pixels in x_2-axis, $N_{x_2} = N_\rho$ and total number of pixels in y_2-axis, $N_{y_2} = N_\theta$ respectively. This is so called mapping/unwarping process whereby it is programmed using equation (3) and (4) in MATLAB to determine the center pixels for Cartesian geometry in (x_2, y_2) axis and then map the log-polar omnidirectional image into such Cartesian coordinate and output as panoramic image. For example, we

use the outputs of case (a) - (d) and mapped them into corresponding panoramic images as shown in Fig. 7a, Fig. 7b, Fig. 7c, and Fig. 7d respectively.

Figure 6a. 256 X 256 log-polar omnidirectional image

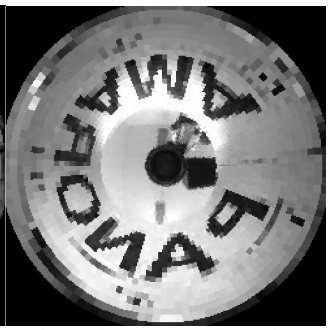


Figure 6b. 256 X 128 log-polar omnidirectional image

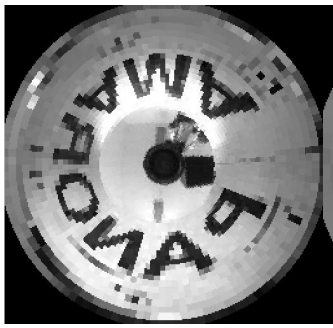


Figure 6c. 128 X 128 log-polar omnidirectional image

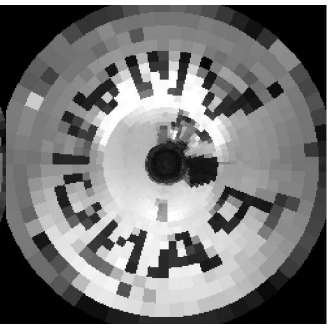
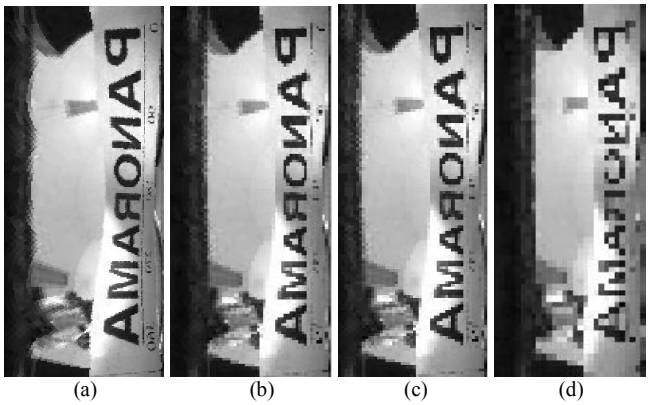


Figure 6d. 128 X 64 log-polar omnidirectional image



(a) (b) (c) (d)

Figure 7.(a) 256 X 256 Cartesian panoramic image, (b) 256 X 128 Cartesian panoramic image, (c) 128 X 128 Cartesian panoramic image, (d)128 X 64 Cartesian panoramic image

From the simulation results, we can observe that in case (a) mapping process (from Cartesian omnidirectional image sampling to log-polar omnidirectional image, then mapping into Cartesian panoramic image), which has the results as shown in Fig. 6a and Fig. 7a, is a full scale mapping with no data compression at all, since 256 X 256 omnidirectional Cartesian pixels are mapped back to 256 X 256 panoramic Cartesian pixels. Case (b) mapping process is by 2:1 reduction mapping scale, which means that 256 X 256 omnidirectional Cartesian pixels are mapped to half of the Cartesian pixels in panoramic view, with two fold data compression compare to full scale mapping image. In case (c) mapping process, the reduction mapping scale is by 4:1, with four fold of data compression whereas in case (d) mapping process, it is by 8:1 reduction mapping scale, i.e. 8 fold of data compression compare to full scale mapping.

Overall, case (a) mapping process is with the finest output resolution quality but no data compression. However, case (d) mapping process is with the highest data compression but poorest output resolution quality. Among the performance of case (b) and case (c) mapping process, we can observe in Fig. 7b and Fig. 7c that the resolution quality of output image in case (c) mapping process is close to that as in case (b) mapping process. However, case (c) mapping process has the advantage of one fold data compression more than that in case (b) mapping process. Hence, we decide to use case (c) mapping process (256 X 256 Cartesian omnidirectional pixels sampled into 128 X 128 log-polar omnidirectional pixels, then mapped into 128 X 128 Cartesian panoramic pixels) for FPGA implementation of log-polar mapping.

IV. FPGA IMPLEMENTATION

FPGA stands for field programmable gate array. It is a chip composed of array of configurable logic blocks. These logic blocks can be used as building blocks to implement any kind of functionality desired, from low-complexity state machines to complete microprocessors [16]. FPGA provides the ability to be reprogrammed to perform other algorithms when necessary. This flexibility therefore makes FPGA a great choice for implementing robotic applications' projects. The FPGA development board used in this paper is Xilinx Virtex 4 ML402 Development board as shown in Fig. 8 and the software development tools used are Xilinx ISE 8.2i and Xilinx EDK 8.2i. All the execution codes are written in C/C++ running on the Xilinx Microblaze Softcore Processor. Implementing log-polar mapping on FPGA is also a mean of achieving real-time high performance processing for the purpose of creating system on chip for robotic applications. This is important in the sense that it reduces space consumption and weight of robots carrying the chip or the development board. Besides, FPGA implementation reduces the cost and turn on time during start up. Table 1 shows the comparison of the FPGA board and laptop computer used in this work.

The average execution time required to produce panoramic output image of different dimensions (256X256, 256X128, 128X128, 128X64, 64X64) are measured for both FPGA and MATLAB using laptop computer. The time required to perform 10,000 counts of log-polar mapping by FPGA and by MATLAB using laptop computer are recorded. The average execution time for mapping a single image is calculated by dividing the total execution time for 10,000 mapping with total counts of mapping (10,000). The statistics are plotted in Fig. 9. From Fig. 9, we can observe that the processing speed required for FPGA is around 1.5 times faster than that in MATLAB using laptop computer.

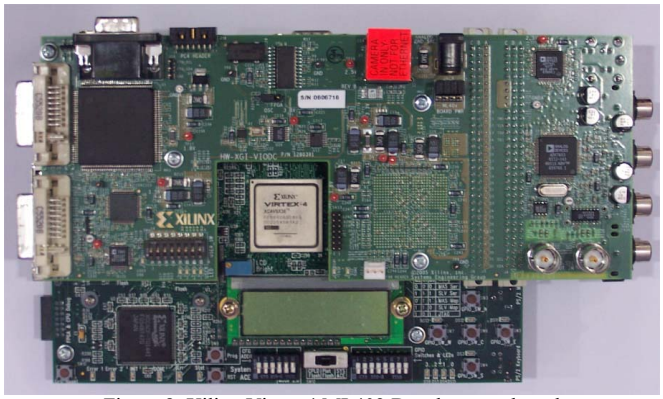


Figure 8. Xilinx Virtex 4 ML402 Development board

	Xilinx Virtex 4 ML402 Development Board	Laptop Computer (Intel core 2 Duo 1.83GHz Processor, 2GB DDR2 RAM)
Size (Volume)	(L)23.5x (W)13.5x(H)5.0 = 1586.25 cm^2	(L)34.0x (W)25.0x(H)4.0 = 3400 cm^2
Weight	0.47kg	2.65kg
Cost	~ US$600	~ US$1,200
Reset time (Boot time)	~ 2 s	~ 80 s

TABLE 1. Comparison of the FPGA board and laptop computer

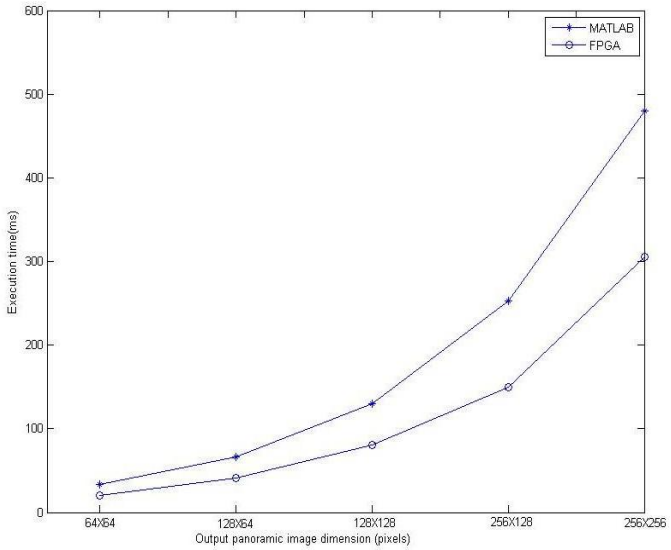


Figure 9. Execution time vs output image dimension

For real time applications in FPGA, we use MATLAB to calculate the geometry coordinates of Cartesian omnidirectional image in log-polar form and also the unwarped form of it (geometry coordinates of Cartesian panoramic image). This geometry coordinates $(x_1,y_1,\rho,\theta,x_2,y_2)$ are store as predefined lookup tables in terms of header files in FPGA. By doing this, the computational complexity could be greatly reduced and hence improved the efficiency in processing speed.

In Section III, we found that case (c) mapping process is the optimum process for FPGA implementation of log-polar mapping. Hence, we will input 256X256 Cartesian omnidirectional image into the FPGA, use FPGA to sample it into 128X128 log-polar omnidirectional image and also unwrap it to 128X128 Cartesian panoramic image. This will allow 4 fold of data compression with fine output resolution quality. We will consider 3 arbitrary sample images taken from [15], [17] and [18] to indicate different applications. These images are captured from a monitor screen output from the FPGA. The first image is shown in Fig. 10, it is the one we use in Section III for MATLAB simulation. The leftmost image is Cartesian omnidirectional image, the center is the log-polar omnidirectional image and the rightmost is the Cartesian panoramic image. The second image is shown in Fig. 11, it is for room monitoring. The third image is shown in Fig. 12, it is for lab monitoring.

Figure 10: image captured from a monitor screen output from the FPGA.

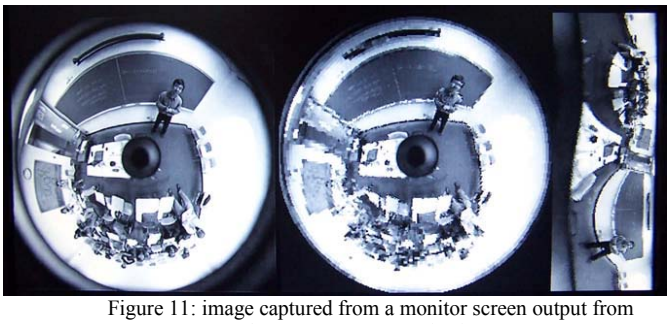


Figure 11: image captured from a monitor screen output from the FPGA (room monitoring).

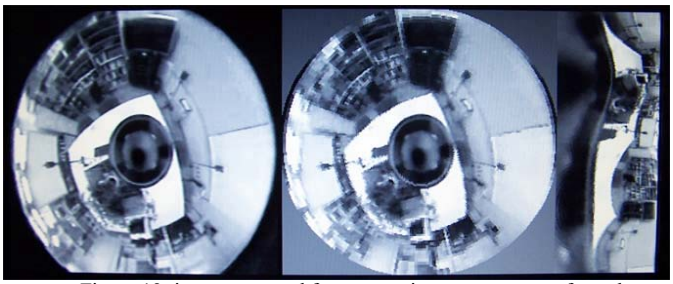


Figure 12: image captured from a monitor screen output from the FPGA (lab monitoring).

V. CONCLUSION

We have demonstrated the implementation of log-polar mapping on a Xilinx FPGA board. There are some significant advantages of this new approach. These include: lighter processing system, lesser space utilization, cost saving, faster processing speed and faster reset time (boot time) compared with a laptop computer that uses MATLAB for doing the unwarping process. Currently, we are using omnidirectional image sample from internet. In future, a video camera and a specific design hyperbolic mirror can be incorporate onto the FPGA system and this single integrated substrate can be embedded on a small mobile robot for surveillance, navigation or other robotic applications. Image processing tools can also be included to further process the panoramic image output by the FPGA. These topics will be addressed in future work.

REFERENCES

[1] C.F.R. Weiman and G. Chaikin, "Logarithmic spiral grids for image processing and display", *Computer Graphics and Image Processing*, 11, p.p. 197-226, 1979.

[2] C. Weiman, "Log-polar vision for mobile robot navigation", *Proc. Of Electronic Imaging Conferences*, p.p. 382-385, Boston, USA, Nov, 1990.

[3] C. Capurro, F. Panerai and G. Sandini, "Dynamic vergence using log-polar images", *IJCV*, 24(1): p.p. 79-94, Aug, 1997.

[4] A. Bernardino and J. Santos-Victor, "Binocular visual tracking: Integration of perception and control," *IEEE Trans. Robot. Automat.*, vol. 15 (6), pp. 1937–1958, Dec. 1999.

[5] A. Bernardino, J. Santos-Victor and G. Sandini, "Tracking planar structures with log-polar images, *Proc. Of Symp. On Intelligent Robotic Systems*, U.K, 2000 (also as VisLab TR 06/2000).

[6] F. Berton, A brief introduction to log-polar mapping, *Technical report*, LIRA-Lab, University of Genova, 10 Feb 2006.

[7] H. M. Gomes, R. B. Fisher, "Learning-based versus model-based log-polar feature extraction operators: a comparative study", *XVI Brazillian Symp. On Comp. Graphics and Image Process.*, p.p. 299-306, Oct 2003.

[8] LIRA Lab, Document on specification, *Tech. report*, Espirit Project n. 31951 – SVAVISCA- available at http://www.lira.dist.unige.it.

[9] R. Wodnicki, G. W. Roberts, and M. D. Levine, "A foveated image sensor in standard CMOS technology", *Custom Integrated Circuits Conf.* Santa Clara, p.p. 357-360, May 1995.

[10] F. Jurie, "A new log-polar mapping for space variant imaging: Application to face detection and tracking", *Pattern Recognition, Elsevier Science*, 32:55, p.p. 865-875, 1999.

[11] V. J. Traver, "Motion estimation algorithms in log-polar images and application to monocular active tracking", PhD thesis, Dep. Llenguatges I Sistemes Informatics, University Jaume I, Castellon (Spain), Sep, 2002.

[12] H. Araujo, J.M. Dias, "An Introduction to the log-polar mapping", *Proc. of 2nd Workshop on Cybernatic Vision,* (1996) p.p. 139-144.

[13] M. Bolduc and M.D. Levine, "A review of biologically-motivated space variant data reduction models for robotic vision", *Computer Vision and Image Understanding*, Vol. 69, No. 2, (February 1998) p.p. 170-184.

[14] C.G. Ho, R.C.D. Young and C.R. Chatwin, "Sensor geometry and sampling methods for space variant image processing", Pattern Analysis and Application, Springer Verlag, (2002) p.p. 369-384.

[15] http://cmp.felk.cvut.cz/demos/Omnivis/SphPanor/sphpan.gif.

[16] www.bdti.com/articles/dspdictionary.html

[17] www.cs.ccny.cuny.edu/~zhu/VirtualClassroom/circular_large.bmp

[18] www.ed.ams.eng/osaka-u.ac/jp/Sens_Omni_02.bmp