Remotely Controlled Real-Time DSP Applications Through Customized GUIs based on LabVIEW

doi:10.3991/ijoe.v6s1.1390

A. Kalantzopoulos, D. Karageorgopoulos and E. Zigouris
University of Patras, Patras, Greece

***Abstract*—The purpose of this paper is to present an approach which could expand the features of Remote Laboratories focused on embedded Digital Signal Processing (DSP) systems. The proposed approach is based on a system which is designed and developed with LabVIEW and is called R-DSP Server. Exploiting this system, users are able to develop their own Graphical User Interfaces (GUIs), named Customized GUIs, for the remote control and validation of real-time DSP applications. These GUIs are tailored to the needs of each DSP application and can be implemented in any programming language. The rapid design of Customized GUIs using LabVIEW for the communication with the R-DSP Server is achieved utilizing an implemented set of functions, called R-DSP LabVIEW Toolkit.**

***Index Terms*—Remote Control, Digital Signal Processors, Graphical User Interfacing, LabVIEW.**

I. INTRODUCTION

In distance education, practical sessions are served from Remote Laboratories (RLs). Nowadays many RLs have been developed in different cognitive fields for the conduct of laboratory experiments from distance. For this reason the designers of each RL have implemented Graphical User Interfaces (GUIs) according to the needs and the particularities of each experiment.

Many RLs have been proposed in the field of embedded systems based on processors for Digital Signal Processing (DSP). These RLs support experiments from different areas such as digital signal and image processing, mechatronics and robotics [1-7]. Through carefully designed GUIs students are able to carry out experiments exploiting the laboratory equipment. However these RLs are able to serve a limited set of experiments, according to the features of their GUIs.

Especially in the areas of digital signal and image processing, the design of GUIs for the control and validation of advanced real-time DSP applications is very important. Many DSP applications such as graphical equalizers, dual tone multi frequency (DTMF) encoders/ decoders, communication and image processing applications are totally controlled through appropriate GUIs [8].

In order to provide users the ability to remotely control real-time DSP applications through Customized GUIs, a system which called R-DSP Server is proposed. Using the features of the R-DSP Server the users are able to utilize their own GUIs for the control and validation of their DSP applications from distance. This approach could expand RLs capabilities in order to support a larger set of experiments with minor modifications in their architecture. The Customized GUIs communicate with the R-DSP Server through TCP/IP protocol therefore they can be developed with any programming language. For these reasons the R-DSP Server advances the education of the students in the design and implementation of GUIs. For the rapid implementation of LabVIEW based Customized GUIs which interact with the R-DSP Server, a set of functions called R-DSP LabVIEW Toolkit is also developed. This toolkit simplifies the communication between LabVIEW and R-DSP Server in order to remotely control DSP development platforms.

II. R-DSP SERVER

The R-DSP Server is designed and developed with LabVIEW v8.6 and allows the remotely control of Code Composer Studio (CCS) Integrated Development Environment (IDE) and development platforms based on Texas Instrument (TI) DSP processors through Customized GUIs (Fig.1). The users are able to develop GUIs tailored to the needs of each DSP application. These Customized GUIs, which can be implemented with any programming language, control the R-DSP Server over TCP/IP through messages formatted according to the R-DSP Protocol.

The R-DSP Protocol is based on the MODBUS Master/Slave protocol and supports messages of variable length [9]. Each message consists of different fields that may contain data formatted as ASCII or binary. The first field includes a 2-byte number that specifies the length of the following message. The next fields are referred to the command with the required arguments, the data and the error description. At the end of each message a field called LRC (Longitudinal Redundancy Check) is used to make an additional verification of the proper delivery of the message.

According to this protocol the R-DSP Server is the Slave and expects messages from the Customized GUI which acts as Master. These messages contain all the necessary information for the execution of the appropriate procedure by the R-DSP Server. As the process is completed, the R-DSP Server sends a reply which includes the associated data and the possible errors.

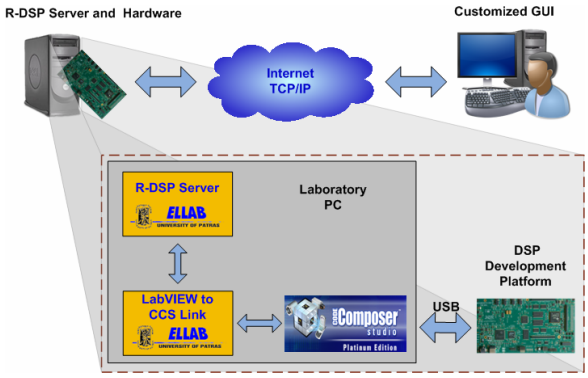

Figure 1. Remote control of DSP development platform through Customized GUI

The R-DSP Server controls the CCS and the development platform through a set of procedures such as the selection of the development platform, the opening and closing of CCS and the downloading to the DSP of the user's executable code. It also supports bidirectional data exchange between the Customized GUI and the DSP using two different scenarios. Data transfer can be performed either by direct access of the DSP memory or utilizing the RTDX technology. The R-DSP Server was implemented utilizing the features of the LabVIEW to CCS Link [10]. The LabVIEW to CCS Link is a LabVIEW toolkit that allows the control of both the CCS and DSP development platform programmatically.

III. R-DSP LABVIEW TOOLKIT

The R-DSP LabVIEW Toolkit is a new toolkit designed and developed with LabVIEW v8.6. It allows the communication between users' GUIs and the R-DSP Server in order to remotely control CCS and DSP development platforms. Using this toolkit, the user can easily and rapidly implement Customized GUIs that control and monitor real-time DSP applications from distance. The R-DSP LabVIEW Toolkit consists of functions that are called SubVIs. These SubVIs send messages to the R-DSP Server, based on the R-DSP Protocol, that activate the execution of the appropriate process. They also receive R-DSP Server replies which contain data associated with the process completion. The Customized GUI can manage and process data sent by R-DSP Server, giving the impression to the user that the DSP development platform is "directly connected" to his computer. These SubVIs are divided into three categories:

- TCP_Setup
- TCP_Automation
- TCP_Communication

The SubVIs of the TCP_Setup category undertake the remotely control of the CCS Setup utility allowing users to select the desired DSP development platform. The automated control of the CCS from distance is achieved through SubVIs of the TCP_Automation. This category consists of SubVIs which open, close the CCS, download to the DSP the executable code etc. The communication between the DSP development platform and the Customized GUI is realized through SubVIs of the TCP Communication category. The R-DSP LabVIEW Toolkit

TABLE I.
THE SUBVIS OF R-DSP LABVIEW TOOLKIT

TCP_Setup			
Icon	**Name**	**Icon**	**Name**
	CCS_Setup_Open_TCP.vi		CCS_Setup_Close_TCP.vi
	CCS_Setup_Clear_TCP.vi		CCS_Setup_Add_Board_TCP.vi
	CCS_Setup_Rename_Board_TCP.vi		CCS_Setup_Remove_Board_TCP.vi
	CCS_Setup_Rename_Processor_TCP.vi		CCS_Setup_Boards_&_Processors_TCP.vi
	CCS_Setup_Save_TCP.vi		
TCP_Automation			
Icon	**Name**	**Icon**	**Name**
	CCS_Open_TCP.vi		CCS_Close_TCP.vi
	CCS_Connect_TCP.vi		CCS_Disconnect_TCP.vi
	CCS_Reset_TCP.vi		CCS_Download_TCP.vi
	CCS_Run_TCP.vi		CCS_Halt_TCP.vi
	CCS_Restart_TCP.vi		CCS_Is_DSP_Running_TCP.vi
	CCS_RTDX_Enable_TCP.vi		CCS_RTDX_Disable_TCP.vi
TCP_Communication			
Icon	**Icon**	**Icon**	**Icon**
	RTDX_Channel_Status_TCP.vi		RTDX_Channel_Enable_TCP.vi
	RTDX_Channel_Disable_TCP.vi		RTDX_Read_TCP.vi
	RTDX_Write_TCP.vi		MEM_Get_Address_TCP.vi
	MEM_Read_TCP.vi		MEM_Write_TCP.vi

Figure 2. Remote control of DSP development platform through Customized GUI

and the R-DSP Server support both the direct access to the DSP memory and the RTDX Technology. The SubVIs of the R-DSP LabVIEW Toolkit are presented in Table I.

IV. USING THE R-DSP LABVIEW TOOLKIT

The development of GUIs that remotely control CCS and real-time DSP applications with LabVIEW, is considerably simplified and speeded up, using the R-DSP LabVIEW Toolkit. It is important to clarify that the GUIs developed using this toolkit do not create DSP executable code. Consequently the user must design and implement locally the DSP application and build the executable code utilizing the CCS. The following paragraphs present typical LabVIEW codes which called block diagrams and describe the features of the proposed toolkit.

A. TCP_Setup Category

The SubVIs of the TCP_Setup category can be used in case the R-DSP Server supports more than one DSP development platforms. In this case the user must configure the CCS to support the desired DSP development platform. A typical block diagram which configures from distance the CCS through the R-DSP Server to support the DSK C6713 development platform of Spectrum Digital is presented in Fig. 2. The CCS_Setup_Open_TCP.vi is placed in order to open the CCS Setup utility. The inputs "R-DSP Server IP" and "R-DSP Server Port" define the IP address and the port of the R-DSP Server. Then the CCS_Setup_Clear_TCP.vi is used to delete previous configurations of the CCS. The CCS_Setup_Add_Board_TCP.vi configures the CCS Setup utility to support the DSK C6713 development platform. Finally the CCS_Setup_Save_TCP.vi saves the current settings and the CCS_Setup_Close_TCP.vi closes the CCS Setup utility.

B. TCP_Automation Category

The Customized GUIs are able to remotely control the CCS and the DSP of the development platform exploiting SubVIs of the TCP_Automation category. Fig. 3 presents a typical block diagram for the automated control of the CCS and the DSP from distance. The inputs "R-DSP Server IP" and "R-DSP Server Port" of the CCS_Open_TCP.vi, define the IP address and port of the R-DSP Server. This SubVI establish the connection of the Customized GUI with the R-DSP Server and opens the CCS. The connection between the CCS and the development board is achieved through the SubVI CCS_Connect_TCP. The CCS sends a reset command to the DSP through CCS_Reset_TCP.vi. Then the executable code is transferred to the R-DSP and downloaded to the DSP using the TCP_Download_TCP.vi. The CCS_RTDX_Enable_TCP.vi enables RTDX interface for data exchange between the CCS and the DSP. The DSP starts the execution of the code after an appropriate command which is send by the CCS_Run_TCP.vi.

The main code of the GUI must be placed inside the While Loop. This structure may contain SubVIs of the TCP_Communication category. As the execution of While Loop is terminated, the CCS_Halt_TCP.vi halts the DSP. The CCS_RTDX_Disable_TCP.vi disables the RTDX interface and the CCS_Disconnect_TCP.vi disconnects the development board from CCS. Finally the CCS and the connection between the R-DSP Server and the Customized GUI are closed by the CCS_Close_TCP.vi

C. TCP_Communication Category

The TCP_Communication category contains SubVIs which are responsible for data exchange between the Customized GUI and the DSP by direct access of DSP memory or using the RTDX technology. These SubVIs are usually placed inside the While Loop of the typical block diagram which was presented in Fig. 3.

The read-write operations using the RTDX technology are demonstrated in Fig.4. The RTDX_Read_TCP.vi is configured to read from the "out_chan" RTDX Channel an array of 4-byte unsigned integers. The RTDX_Write_TCP.vi writes at the "input_chan" RTDX Channel one 4-byte floating point number.

The data communication by direct access of DSP memory is presented in Fig.5. The Mem_Get_Address_TCP.vi locates the address in DSP memory of the variables "message" and "Array". The variable "message" represents a string with a hundred characters length and is read from the MEM_Read_TCP.vi. The MEM_Write_TCP.vi is configured to write at the variable "Array" the content of the control "Data".

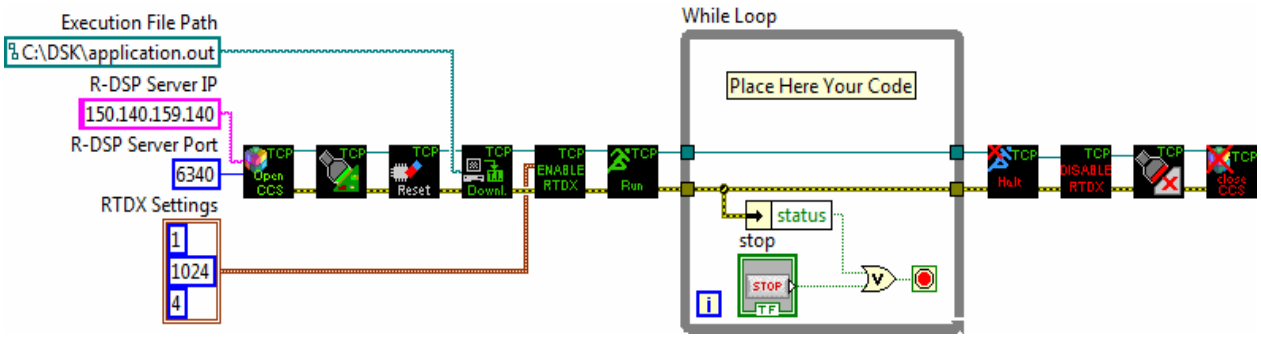

Figure 3. A typical block diagram for the remote control of CCS

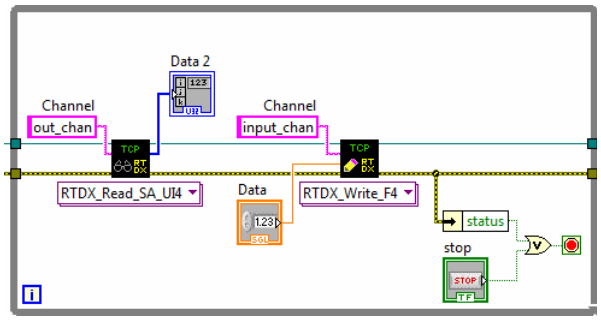

Figure 4. Read – write operations using RTDX technology

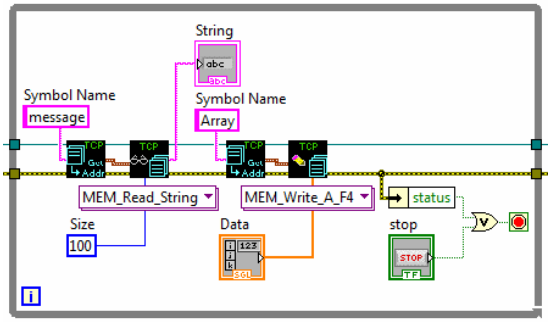

Figure 5. Read – write operations by direct access of DSP memory

V. APPLICATIONS

To demonstrate the features of the R-DSP Server and the R-DSP LabVIEW Toolkit, a real-time image processing application and a 10th band graphical equalizer are implemented. Each one of these applications is consisting of two parts, the Customized GUI and the DSP application. Through the Customized GUI the user is able to remotely control the required hardware and the DSP application which is implemented in the appropriate DSP development platform.

A. Real-Time Image Processing Application

The implemented real-time image processing application is presented in Fig. 6. The required hardware consists of the development platform DSK C6713, that is based on the TMS320C6713 DSP of TI, the daughter card DSKcam of Bitec and the OV7620 CMOS image sensor of OmniVision [11,12]. The DSK C6713 receives and processes images with 320x240 resolution, captured by the CMOS image sensor through DSKcam. The above hardware is located in Electronics Laboratory of Physics Department in Patras University.

The executable code of the real-time image processing application that is downloaded to the DSP, was developed in C language using CCS. According to the user settings, this application processes images which are captured by the CMOS image sensor. The presented implementation supports image processing algorithms such as sobel edge detection, histogram equalization and some basic filters [13]. The captured and the processed image are transferred to the R-DSP Server utilizing the RTDX technology.

The Customized GUI for the remote control of the real-time image processing application is developed using LabVIEW and the proposed R-DSP LabVIEW Toolkit (Fig.6). In the fields "Address" and "Port" the user must enter the IP Address and the Port of the R-DSP Server. After the connection establishment the executable code is transferred to the R-DSP Server and downloaded to the

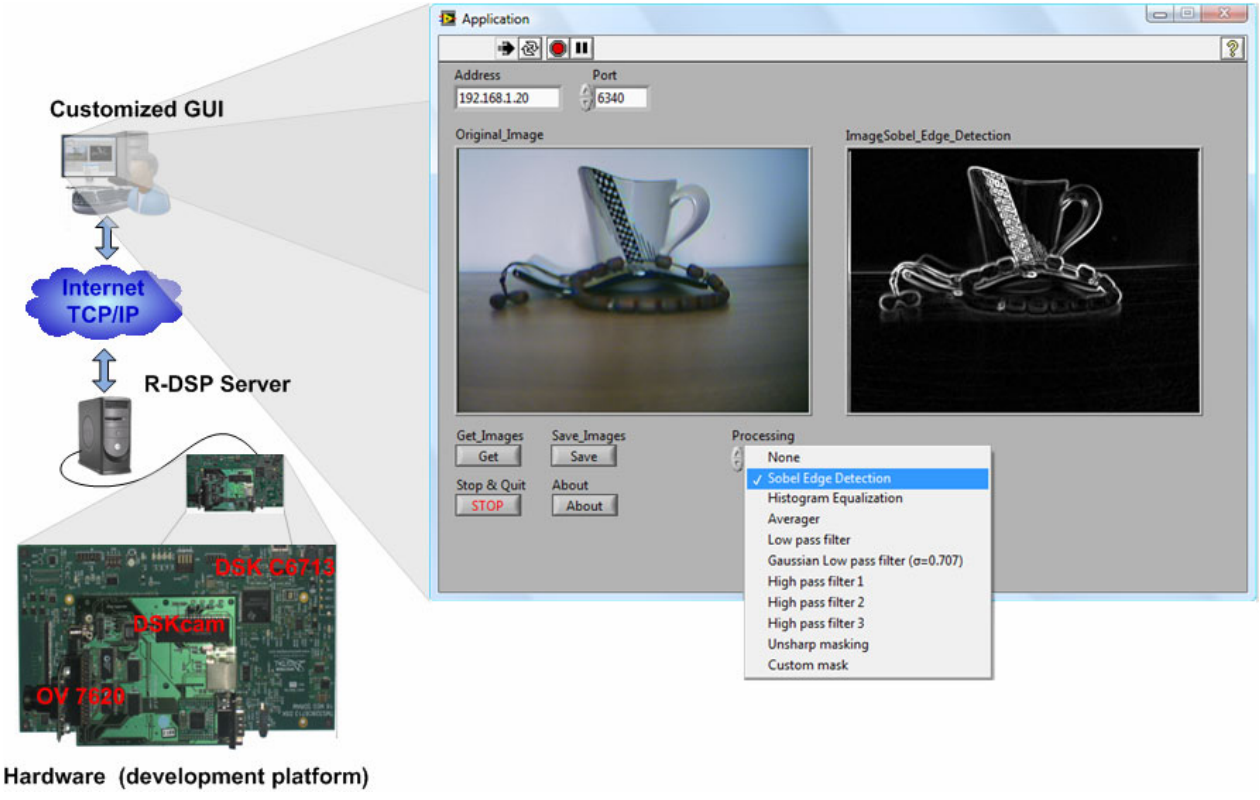

Figure 6. Remote control of real-time image processing application through Customized GUI

DSP. The user is able to select the desired image processing algorithm from the processing menu. By pressing the button "Get", the captured and the processed image are transferred to the GUI and presented in the appropriate indicators.

B. 10^{th} Band Graphical Equalizer

One of the most common applications on DSP systems, for educational purposes, is the graphical equalizer [8]. The Customized GUI which remotely controls a 10^{th} band graphical equalizer and is running on the DSK C6713 development board is presented in Fig. 7.

The DSP application which is implemented in C language using the CCS, simulates the operation of a 10^{th} band graphical equalizer. The input signal of the equalizer which consists of three sinusoidal signals with 125Hz, 1KHz and 5KHz frequencies, is generated by the DSP application. This signal is processed by the equalizer according to the values of the sliders in the Customized GUI. Finally the input and output signals are transferred to the Customized GUI through the R-DSP Server and they are displayed at the waveform indicators of the Customized GUI.

VI. CONCLUSIONS

In this paper the features of the R-DSP Server that enable the implementation of Customized GUIs to remotely control real-time DSP applications, were presented. The R-DSP Server was designed in such a way to be handled by Customized GUIs written in any programming language that supports TCP/IP Protocol. The incorporation of the R-DSP Server in RLs which are focused in the field of embedded systems based on processors for DSP, expands their features and increases the set of supported experiments. The easily and rapidly design of LabVIEW based Customized GUIs for the remote control of DSP applications through R-DSP Server is achieved using the R-DSP LabVIEW Toolkit. The potentials of the proposed toolkit and the R-DSP Server were demonstrated through the presented real-time image processing application and the 10^{th} band graphical equalizer. Future plans include the integration of the R-DSP Server features in Remote Laboratories, such as the R-DSP lab. In addition, there are plans for developing toolkits for other programming languages such as Matlab and C # that allow the fast implementation of GUIs to remotely control real-time DSP applications, utilizing the R-DSP server.

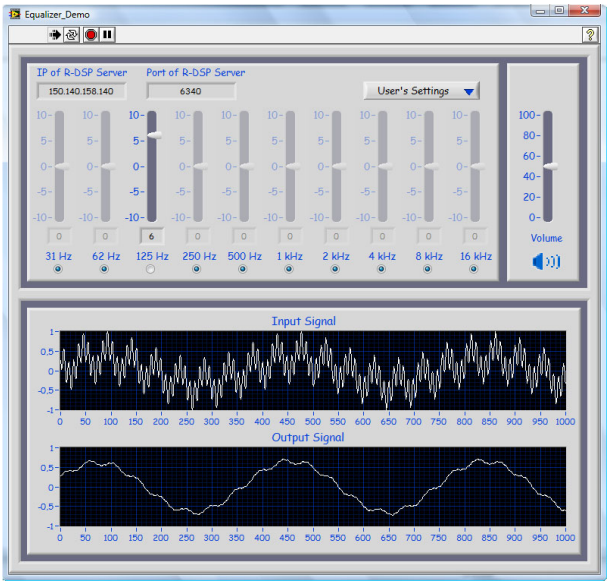

Figure 7. The Customized GUI for the remote control of a 10^{th} band graphical equalizer through

REFERENCES

[1] A. Tekin, F. Ata, M. Gökbulut, "Remote Control Laboratory for DSP-Controlled Induction Motor Drives", *Computer Applications in Engineering Education*, Available online in Wiley InterScience: 10.1002/cae.20440, April 2010.

[2] A. Kalantzopoulos, D. Markonis and E. Zigouris, "A Remote Laboratory for Real-Time Digital Image Processing on Embedded Systems", *International Journal of Online Engineering (i-JOE)*, Vol. 5, No. 4, pp. 24-29, November 2009.

[3] F.Barrero, S. Toral and S. Gallardo, "eDSPLab: Remote Laboratory for Experiments on DSP Applications", *Internet Research*, Vol. 18, No. 1, pp. 79-92, 2008. doi:10.1108/10662240810849603

[4] A. Kalantzopoulos, D. Karageorgopoulos and E. Zigouris, "A LabVIEW Based Remote DSP Laboratory", *International Journal of Online Engineering (iJOE)*, vol. 4, special issue 1: REV 2008, pp.36-44, July 2008.

[5] University of Patras, Physics Dept., Electronics Lab., Remote DSP Laboratory (R-DSP Lab), http://rdsplab.physics.upatras.gr.

[6] D. Hercog, B. Gergič, S. Uran and K. Jezernik, "A DSP-Based Remote Control Laboratory", *IEEE Transactions On Industrial Electronics*, vol. 54, No. 6, pp. 3057-3068, December 2007. doi:10.1109/TIE.2007.907009

[7] R. Šafarič, M. Truntič, D. Hercog and G. Pačnik, "Control and Robotics Remote Laboratory for Engineering Education", *International Journal of Online Engineering (i-JOE)*, Vol. 1, No. 1, pp. 1-8, November 2005.

[8] R. Chassaing and D. Reay, "Digital Signal Processing and Applications with the TSM320C6713 and TMS320C6416", 2nd Edition, John Wiley & Sons, 2008. doi:10.1002/9780470238141

[9] Modbus Specifications and Implementation Guides, http://www.modbus.org/specs.php.

[10] E.Zigouris, A.Kalantzopoulos and E. Vassalos, "LabVIEW to CCS Link for Automating Digital Signal & Image Processing Applications", *8th International Symposium on Signals, Circuits & Systems*, ISSCS 2007, pp. 445-448, Iasi, Romania, 12-13 July 2007.

[11] Spectrum Digital, "TMS320C6713 DSK, Technical Reference", 506735-0001 Rev.B, November 2003.

[12] BiTEC, "DSKcam, Users Manual", 2005.

[13] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", 3rd Edition, Prentice Hall, 2007.

AUTHORS

A. Kalantzopoulos is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: kalan@upatras.gr).

D. Karageorgopoulos is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: dimitriskarageorgopoulos@gmail.com).

E. Zigouris is with the Electronics Laboratory, Electronics and Computers Div., Department of Physics, University of Patras, Rio Patras, GR-26500 (e-mail: ez@physics.upatras.gr)

This article was modified from a presentation at the REV2010 conference in Stockholm, Sweden, June 2010. Submitted, March 27, 2010. Published as resubmitted by the authors on July 15, 2010.