Image Processing Test Bench for Robot Vision Using C#

Sudip Chakraborty [1] & P. S. Aithal [2]
[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2]ViceChancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Area of the Paper: Robotics
Type of the Paper: Experiment-based Research
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.5806319
Google Scholar Citation: IJCSBE

How to Cite this Paper:
Sudip Chakraborty & Aithal, P. S., (2021). Image Processing Test Bench for Robot Vision Using C#. *International Journal of Case Studies in Business, IT, and Education (IJCSBE), 5*(2), 366-374. DOI: https://doi.org/10.5281/zenodo.5806319

International Journal of Case Studies in Business, IT and Education (IJCSBE)
A Refereed International Journal of Srinivas University, India.

Crossref DOI : https://doi.org/10.47992/IJCSBE.2581.6942.0141

Image Processing Test Bench for Robot Vision Using C#

Sudip Chakraborty [1] & P. S. Aithal [2]
[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2]ViceChancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *Nowadays, image processing is a well-known technological term. In some of the industries, it has practical needs. It is an essential tool for the process and robotic industry. Various popular frameworks and libraries are available to process the image. The OpenCV is one of the best and popular libraries for image processing. It was originally written in C++ by Intel. Now various wrappers are available to implement into the different programming languages. The OpenCvSharp is the wrapper of OpenCV. Those who are familiar with C# can use it. The new researcher who wants to integrate image processing into their project takes some time for setup, function writing, and integration. Here we created a test bench application for Image processing demonstration. It has been made with some usual function to process the image. It was created using visual studio 2022 and OpenCvSharp wrapper in C# language. The researcher can learn about various image processing algorithms without writing any code or giving little bits of effort. The complete project is available on GitHub. Anyone can download, experiment, and integrate into their project without any issue.*

Design/Methodology/Approach: *We created a GUI (Graphical User Interface) based C# application. Using Nuget Package manager, installed two OpenCV wrapper packages. To invoke several functions, we add some buttons, and for changing the method's parameter, we integrate some text boxes. We created some abstraction layers Between the OpenCvSharp wrapper and GUI. We made our custom module as portable as possible so that our researchers could easily incorporate it into their project.*

Findings/result: *This unique image processing test bench is designed for new researchers trying to integrate image processing capability into their research work. It can take still images or moving images through the connected webcam, automatically sending the various commands and promptly observing the result.*

Originality/Value: *This test bench has been arranged uniquely for the researcher. It might have some value to their research work. The unique feature like automatic trigger can help them send the series of commands without repeatedly typing or pressing the button to see the result.*

Paper Type: *Experiment-based Research*

Keywords: OpenCvSharp, OpenCV in C#, Image Processing, Robotics Vision, test Bench for Image processing, Digital Image processing.

1. INTRODUCTION :

Using image processing, we can do a lot. Now in several fields, we are adding value. The modern surveillance camera captures the image and processes it intelligently. It can detect fire incidents, suspicious activity. The driverless car is entirely dependent on image processing. Taking the picture from the camera, process and get lots of information about the road. The robotic arm detects and grasps the object using a depth camera. There is more list like feature extraction, face detection, forecasting, optical character recognition, finger-print detection, optical sorting, microscope imaging, and many more. This technology is grasping more areas day by day. For All of the above, we need to process the image in real-time or almost real-time, depending on the application nature. For that, we need an efficient framework to process the image.

The most popular framework is OpenCV. It is written in the C++ programming language. For other than C++, users use the wrapper. For the C# programmer, Emgu CV, Aforge, OpenCvSharp Wrappers are popular. The other variant has pros and cons. In our research project, we will use the OpenCvSharp library. The Image processing framework is significantly larger. It requires lots of effort to grasp and integrate it into our research work. So for our robot researcher, we provide a test bench to understand the algorithm very easily. Here we create a GUI-based test bench application, and the most relevant functions can be accessed through the GUI elements. For our purpose, we made some custom classes. OpenCV.cs, UtilOpenCv.cs etc. These classes can be integrated into different projects. Our robot researcher can use it [1-8].

2. RELATED WORKS :

Feng Ran et al., in their paper, presents an implementation of the Qt4 test bench. The implementation process combines the build of compiler environments, the kernel transplant, the production of the root filesystem, drivers installation, and the process of Qt/Embedded transplant supporting touch screen [9]. In their paper, R. D. C. Santos et al. proposed an image processing-based test bench that monitors the performance of the cloud's resources to gather valuable statistics and determine which aid should be used for a specific task to reduce costs and to run time [10]. M. Morelli et al. proposed an approach in which a functional model of the controls is matched to a model of the execution platform through an intermediate mapping model representing the software tasks and communication messages. They describe a robotic car test bench used to show the application of the methodology. The test bench has enough functional complexity and a distributed implementation to justify the creation of architecture models while requiring a moderate cost and effort for its construction by the interested researchers [11]. S. V. Vityazev, in their paper, proposed a test bench for signal processing module testing. Essential components of the bench, their functionality, and development issues are discussed. Using the test bench for educational purposes [12]. G. N. Chaple et al. proposed Image processing applications in real-time embedded systems. Their Edge detection algorithms are written with the help of hardware descriptive language VHDL and focused on edge detection of grayscale images [13]. D. Gergelyi et al. investigated the possibility of the usage of the DMD based digital light processing (DLP) technology on a High-speed vision system [14]. V. Pashaei et al. proposed a complete low-cost open-source portable ultrasound test bench for biomedical imaging applications. The test bench is a programmable 64-channel system with a modular design easily updated with improved hardware and software for wearable and implantable medical ultrasound research [15]. Zheng Shenghua, in their paper, proposed an arbitrary waveform generator for synthetic aperture radar (SAR) testbench application. This waveform generator uses high-speed multi-DSPs to accelerate the processing to obtain a near real-time baseband echo signal of SAR. A DDS-based frequency synthesizer shifts the call to the operating band. [16]. S. J. Weddell and J. R. proposed a novel method to measure phase modulation using wavefront tilt. The optical testbench is for the study of inverse problems, such as atmospheric tomography and deconvolution from wavefront sensing [17]. A. Coghe et al., in their paper, describe first the general layout of the test plant. with a brief description of all the updated required instrumentation [18].

3. OBJECTIVES :

The objective of the research work is to provide information to the robot researcher working with image processing. To minimize the setup time of their research work, we created a test bench application. We provide a GUI control interface for their understanding and ease of operation. Through it, image editing and transformation commands can be sent. Also, there is an autonomous command sending interface available. It helps to send a sequence of commands without typing repeatedly. After experimenting and studying the code, the researcher can integrate it into their research work.

4. APPROACH AND METHODOLOGY :

Figure 1 depicts the architecture of the research work. The main coordinator class is “frm_main.cs”. This class is responsible for interacting with the User Interface element. It is the manager class. It interacts with the other class and invokes the other class object on a need basis. On the right-hand side of the picture is OpenCV-related parts. The “Open_CV.cs” and “utilOpenCV.cs” is our created class. These two classes are the operation's primary interaction function—the “System.CodeDom” and

System. Management object is responsible for getting the list of available Webcam connected with the device. The primary thing is openCvSharp3-AnyCPU and OpenCVSharp object.

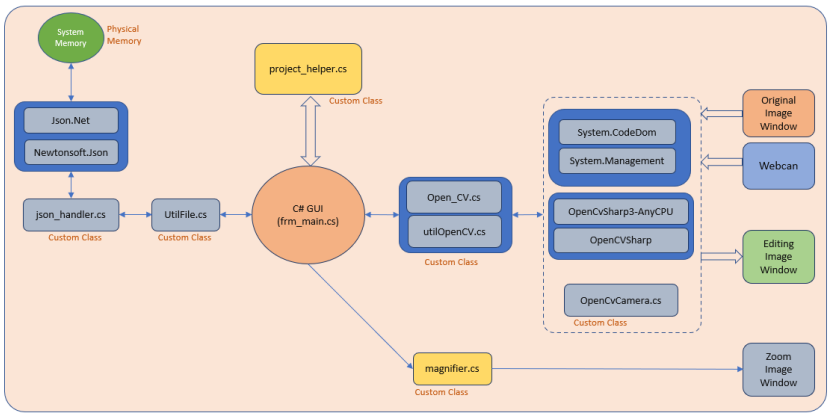

Fig. 1 : Architecture of the research work

The OpenCvCamera.cs is used to read the moving picture from the webcam. Whatever we edit, the image is displayed inside the "Editing image window." To see the zooming image, we provide a separate picture box. When the mouse moves on the edited image box, the particular portion is zoomed and visible inside the zoomed picture box. The "project_helper.cs" class provides functionality that helps the main module. All system settings are saved upon application exits. There is also a button "Save Config" to save the configuration manually. We keep the file in .json format. We created "json_handler.cs," which holds the configuration settings in JSON format.

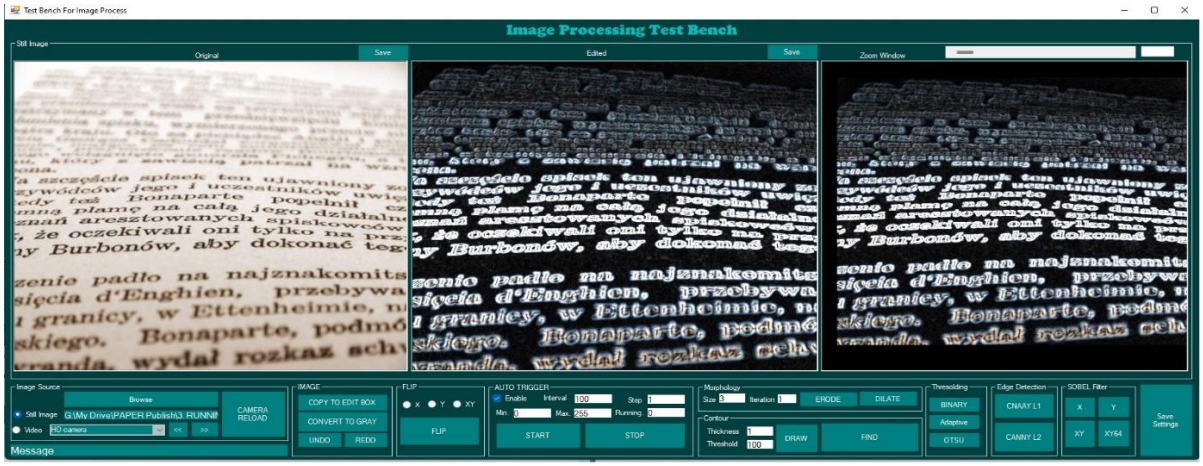

Fig. 2 : The application Interface

Figure 2 depicts the application interface of several GUI applications to invoke the image processing function. Let us introduce the Application element one by one.

Image Source: The top left is the Image source group box depicted in figure 3. The two radio buttons are available. The one is for still images, and the other is for video images or moving images. For still images, press the browse button. One dialog box will appear. Select an image from a location, and the image will display inside the still image original picture box. For motion images, select the "Video" radio button first. From the combo box, we can choose one of them. The web camera image is displayed In the original picture box. The "<<" and ">>" button is used to switch to the previous and next camera. When the application starts, it read the number of connected cameras and lists them out into the combo box. If the camera is connected after the application starts, it can not populate into the combo box. So we provide a button "CAMERA RELOAD." We can update the camera list into the combo box by pressing the button. Whatever still or motion image and path is selected is saved automatically when the application is closed. Alternatively, by pressing the "Save Settings" button, save the settings manually.

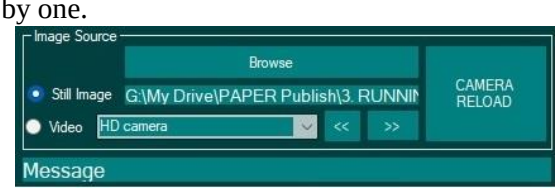

Fig. 3 : Image source

IMAGE: Figure 4 depicts the image-related control. There are four buttons available inside the group box. The Top One is the "COPY TO EDIT BOX" button. Using this button, we can copy the image from the original box and clone it into the edit picture box used to apply the various transformations. The next one is the "CONVERT TO GRAY" button. Pressing the button, we can convert the colored image into grayscale. Most image processing algorithms work on the gray image because it is efficient and faster. Here "UNDO" and "REDO" button is not implemented. The researcher can write their own.

Fig. 4 : Image

FLIP: This group box is used to flip the image is depicted in figure 5. If the source image is not correctly oriented, we can change the orientation. The first radio button, "X," is for exposure on X direction. The second radio button, "Y," is used to flip in the "Y" direction, and the third one is used to convert X and Y direction at a single shot.

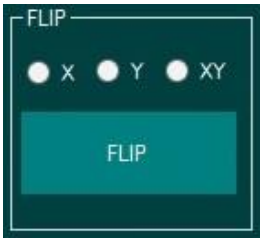

Fig. 5 : Flip

Figure 6 depicts the morphological operation. We can apply to erode and dilate function on running editing images. The dilation process adds the pixels to the boundaries of objects. At the same time, erosion removes pixels on object boundaries. In the "Morphology" group box, In the size textbox is the size of the kernel. The iteration textbox, how many times the process will repeat. After selecting this parameter, we need to press the "ERODE" or "DILATE" button.

Fig. 6 : Morphological Operation

Thresholding: Thresholding techniques are the most popular for segmentation. A set of suitable thresholds must be determined first, and then the image can be segmented by comparing the pixel properties with these thresholds. The process of detecting object contours based on the computation of a binary image in which pixel values are set to one, in correspondence to a video intensity value of the corresponding pixel position in the analyzed image more significant than the threshold, and zero if the video intensity is lower than the threshold. The point can be constant for all Images or adaptive and spatially dependent. Figure 7 depicts the group box, which provides buttons for three thresholding types discussed below.

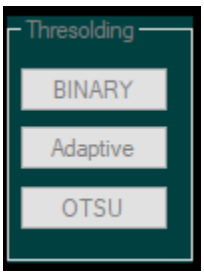

Fig. 7 : Theresholding

Binary Threshold: This thresholding is a technique in OpenCV, which is the assignment of pixel values about the threshold value provided. In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is 0. Otherwise, it is set to a maximum value (generally 255)

Adaptive Threshold: This thresholding is the method where the threshold value is calculated for smaller regions, and therefore, there will be different threshold values for other areas. In OpenCV, we can perform an Adaptive threshold operation on an image using the adaptive threshold method() of the utilOpenCv.cs class.

OTSU Threshold: This thresholding procedure came from Nobuyuki Otsu. It performs automatic image thresholding. In the simplest form, the algorithm returns a single intensity threshold that separates pixels into two classes, foreground and background.

Auto Trigger: In the image processing research, we apply an algorithm and see the result. If we are not satisfied with the result, we use the same algorithm with a different value. This procedure takes time to experiment. So here we provide a feature, "AUTO TRIGGER." In this group box depicted in figure 4.8, one has to fill up the box before sending the command (applicable for limited command only). The first is the "Enable" check box. If the check box is not enabled, the command will

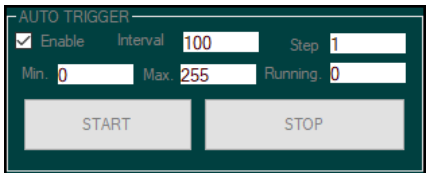

Fig. 8 : Auto Trigger

apply only once, and the timer will auto-disable. The command will run indefinitely if the checkbox is checked until the stop button is pressed.

The start button is to start the auto-trigger timer. The "interval" textbox sets the time to repeat. It is in millisecond format. Let say we apply "1000," which means the command will use after a one-second interval. The "step" text box is for increment per step. Every time this value will add with the previous. The "Min" value text box is used for a minimum algorithm parameter value, whether the "Max." Value is for the maximum parameter value. The "Running" textbox is used to display for running parameter value. Moreover, the "START" and "STOP" buttons start and stop the timer.

5. EXPERIMENT :

Let us work on some experiments. Here, we use the visual studio 2022 community edition. We need to download the project from GitHub. The link is available in recommend section. Opening the project inside the visual studio and building the project, we will observe the list of errors depicted in figure 9. The errors are due to the absence of OpenCV packages.

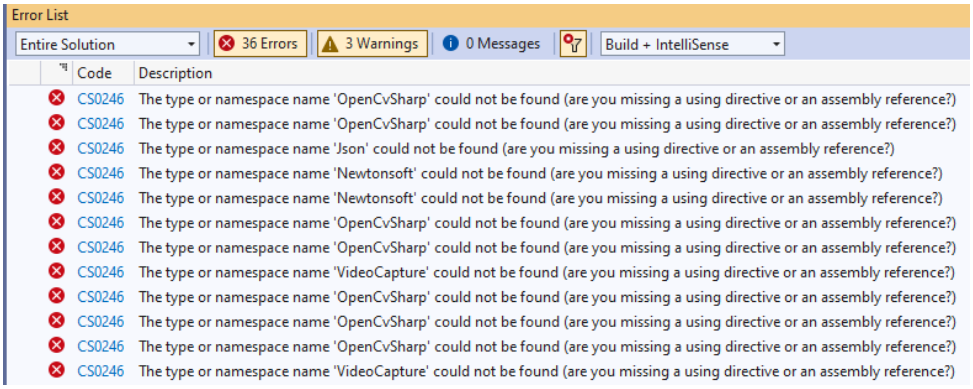

Fig. 9 : List of Error

Install the below packages from the web using Nuget package manager, which is available under the project menu, or right-click on the project, select "Manage NuGet Packages..". In the "Browse" tab, type the name, and from the right side, click on the "Install" button.

OpenCVSharp by OpenCVSharp (Description: OpenCVSharp package)

OpenCvSharp3-AnyCPU by shimat (Description: OpenCV wrapper for .NET Framework)

Now, we can build the project. The build should be successfully done. Else the application will not run. After opening the application, the windows look like figure 4.2. Check once the settings.json file is inside the bin folder else will not run properly. Now for the experiment, may follow the below sequence.

(1) In the middle, three picture box present. The first is the original image, the center is the editing window, and the last is the zoom window. After opening the application, the image source selection is available in the bottom left group box. We can Select "Still Image" or "Video" images using the radio button. For still images, select using the "Browse" button. For motion Images, choose from the combo box.

(2) We need to transfer the image into Edit windows to edit the image. From the "Image" group box, Press the "COPY TO EDIT BOX" button—most of the algorithms work on gray image mode. So, convert the image into grayscale by pressing the "CONVERT TO GRAY" button.

(3) If the image orientation is not proper, then we can change the image orientation. There are three types of direction. The select radio button "X" to flip the image on X-axis. Select the "Y" radio button around the "Y" axis. Moreover, for both directions, select the "XY" radio button. Then press the "FLIP" button.

(4) Then according to our need, we can process Morphology, Contour, Thresholding, Edge detection, and SOBEL filter.

(5) Some algorithm has an Auto Trigger facility to eliminate the repetition work overhead.
(6) For the AUTO trigger, check to enable button, set interval, step, min, max, and running value. Then press the start button to start the timer. For any time to stop, press the "STOP" button.
(7) To save various settings in the application, press the "Save settings" button.

6. RECOMMENDATIONS :

- To download the project, https://github.com/sudipchakraborty/Image-Processing-Test-Bench-Using-C-Sharp
- This research work is based on the Udemy Online course "Computer Vision Fundamentals with OpenCV and C#. https://www.udemy.com/course/image-processing-fundamentals-with-opencv-and-csharp/. It is a good course for OpenCvSharp. We can study the System to understand various algorithms and its parameter.
- Image processing is a vast area, and The few primary functions are included here. The researcher can integrate more features on their need.
- This application is not entirely bug-free. We debugged what we got at experiment time. May persists several bugs, which can be debugged under more tests.
- We developed core functions only. The researcher can add some additional functionality for better usability.
    - To make it robust, error Handling may be reviewed.
    - We provide X, Y, and XY flip functions using a button press. We can implement using check change events.
    - We provided UNDO and REDO Button. We didn't write a function inside the button. The researcher can implement it if required. When applying any effect on the edit image box. We can keep a copy, then use it. Creating an image list, we can push the current image before applying the image. When we press undo, it will POP the image and display it inside the image box. The same thing can happen for the REDO button. The image index will PUSH and POP to the image box.
    - This is not production-grade code. In most places, there is no exception handle. The researcher can add the try-catch block to avoid exception occurrence.
    - We provided a Zoom window, and the primary example is provided. For specific requirements, the researcher can be customized for their research purpose.
    - We can implement a sequential command runner to send the image transformation command through Datagrid or text box. The researcher can provide the program save and load option.
    - The Auto trigger can be extended for most of the functions.
    - Auto trigger decrement can be implemented.

Any application-related suggestions are appreciated.

7. CONCLUSION :

Through this research work, we observe how we can create an interface for the Image processing test bench. First, we started an application in Microsoft visual studio 2020 using the C# language. We add the OpenCV library to work together. Through the GUI control, we invoke the function according to our needs. The two classes played a significant role, the primary and helper classes. Our robot researcher, working with image processing for process control and robotic vision, can get some reference information from this research work.

REFERENCES :

[1] Chakraborty, S., & Aithal, P. S. (2021). Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 29-37.
Google Scholar

[2] Chakraborty, S., & Aithal, P. S. (2021). A Custom Robotic ARM in CoppeliaSim. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 38-50.
Google Scholar

[3] Chakraborty, S., & Aithal, P. S. (2021). An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim. *International Journal of Case Studies in Business, IT, and Education (IJCSBE), 5*(1), 78-87.
Google Scholar

[4] Chakraborty, S., & Aithal, P. S. (2021). Demonstration of Modbus Protocol for Robot Communication Using C#. *International Journal of Applied Engineering and Management Letters (IJAEML), 5*(2), 119-131.
Google Scholar

[5] Chakraborty, S., & Aithal, P. S. (2021). Demonstration of Drawing by Robotic Arm using RoboDK and C. *International Journal of Applied Engineering and Management Letters (IJAEML), 5*(1), 153-158.
Google Scholar

[6] Chakraborty, S., & Aithal, P. S. (2021). Forward and Inverse Kinematics Demonstration using RoboDK and C#. *International Journal of Applied Engineering and Management Letters (IJAEML), 5*(1), 97-105.
Google Scholar

[7] Chakraborty, S. & Aithal, P. S. (2021). Terminal++ for Robot Researcher Using C#. *International Journal of Applied Engineering and Management Letters (IJAEML), 5*(2), 175-182. DOI: https://doi.org/10.5281/zenodo.5773848.
Google Scholar

[8] Chakraborty, Sudip, & Aithal, P. S., (2021). ABB IRB 120-30.6 Build Procedure in RoboDK. *International Journal of Management, Technology, and Social Sciences (IJMTS), 6*(2), 256-264. DOI: https://doi.org/10.5281/zenodo.5782759
Google Scholar CrossRef/DOI

[9] Feng Ran, Tao Wang, and Sijie Ran, (2012). Implementation of the Qt4 test bench based on Arm9. IEEE Symposium on Electrical & Electronics Engineering (EEESYM), 198-201, DOI: 10.1109/EEESym.2012.6258623.
Google Scholar

[10] Santos, R. D. C. and Apduhan, B. O. (2013). An Image Processing-based Test Bench for Performance Evaluation in Hybrid Clouds. 13th International Conference on Computational Science and Its Applications, 2013, pp. 33-38, DOI: 10.1109/ICCSA.2013.54.
Google Scholar

[11] Morelli, M., Moro, F., Rizano, T., Fontanelli, D., Palopoli, L. and Di Natale, M. (2013). A robotic vehicle testbench for the application of MBD-MDE development technologies. IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1-4, DOI: 10.1109/ETFA.2013.6648147.
Google Scholar

[12] Vityazev, S. V. (2012). Test bench for signal processing modules examination and efficiency rating. 5th European DSP Education and Research Conference (EDERC), pp. 213-216, DOI: 10.1109/EDERC.2012.6532257.
Google Scholar

[13] Chaple, G. N. Daruwala, R. D. and Gofane, M. S. (2015). Comparisons of Robert, Prewitt, Sobel operator based edge detection methods for real-time uses on FPGA. International Conference on Technologies for Sustainable Development (ICTSD), pp. 1-4, DOI: 10.1109/ICTSD.2015.7095920.
Google Scholar

[14] Gergelyi, D. and Földesy, P. (2010). Digital Micromirror Device (DMD) projector-based test bench for vision chips. 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010), pp. 1-4, DOI: 10.1109/CNNA.2010.5430248.
Google Scholar

[15] Pashaei, V. Roman, A. and Mandal, S. (2018). Live Demonstration: An Open-Source Test-Bench for Autonomous Ultrasound Imaging. IEEE Biomedical Circuits and Systems Conference (BioCAS), 2018, pp. 1-1, DOI: 10.1109/BIOCAS.2018.8584728.
Google Scholar

[16] Zheng Shenghua, Xu Dazhuan, Jin Xueming, Zhang Shishan, and Zhang Hongrong, (2005). An arbitrary waveform generator for SAR testbench application. Asia-Pacific Microwave Conference Proceedings, pp. 3-12, DOI: 10.1109/APMC.2005.1606359.
Google Scholar

[17] Weddell, S. J. and Howe, J. R. (2010). An optical testbench and atmospheric turbulence emulator for astronomical image restoration. 25th International Conference of Image and Vision Computing New Zealand, pp. 1-6, DOI: 10.1109/IVCNZ.2010.6148808.
Google Scholar

[18] Coghe, A. Ferri, L., Ghezzi, U., Pasini U, and Solero, G. (1996). Test bench for industrial burners: a diagnostic study for combustion control and process regulation. IECEC 96. Proceedings of the 31st Intersociety Energy Conversion Engineering Conference, *3*(1), pp. 2062-2067. DOI: 10.1109/IECEC.1996.553437.
Google Scholar

****