Implementation of Fuzzy Color Extractor on NI myRIO Embedded Device

Conference Paper · September 2014

DOI: 10.1109/MFI.2014.6997638

7 authors, including:

David Oswald
California State University, Bakersfield
5 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Wei Li
California State University, Bakersfield
157 PUBLICATIONS **2,000** CITATIONS
SEE PROFILE

Linwei Niu
West Virginia State University
32 PUBLICATIONS **315** CITATIONS
SEE PROFILE

Jin Zhang
Chinese Academy of Sciences, China, Shenyang
11 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Implementation of Fuzzy Color Extractor on NI myRIO Embedded Device

David Oswald, Wei Li, Linwei Niu, Jin Zhang, Yan Li, Jiangchen Yu, Fuchun Sun

***Abstract*— A fuzzy color extractor is defined by fuzzy rules to extract certain colors based on a color pattern (color seed). In order to extract certain colors from images in real-time, this paper attempts to implement a fuzzy color extractor on NI myRIO embedded device, which features the Xilinx Zynq-7010 all-programmable system including a dual-core ARM® Cortex™-A9 real-time processor and an Artix-7 FPGA with customizable I/O.**

To implement the fuzzy color extractor on NI myRIO, the fuzzy logic module, real-time module, and FPGA module of LabVIEW 2013 are required. LabVIEW is a system-design platform and development environment for a visual programming language. First, we define the fuzzy membership functions and fuzzy rules for the fuzzy color extractor using the Fuzzy Logic module. Second, we implement the fuzzy color extractor using Real-Time module. Finally, we implement the fuzzy color extractor a DaNI robot to test the color extraction performance under varying illumination conditions.

***Index Terms*—Fuzzy color extractor, embedded system, myRIO, LabVIEW, DaNI robot.**

I. Introduction

Image segmentation is a fundamental issue in image processing and computer vision. Especially, color images can provide much more information than gray-level, so algorithms for color image segmentation have attracted more and more attention. A number of image segmentation techniques have been proposed in past decades [1]-[6]. However, most of the proposed techniques provide a crisp segmentation of images, where each pixel is classified into a unique subset. This classification may not reflect an understanding of images by human very well, as the work [1] stated "the image segmentation problem is basically one of psychophysical perception, and therefore not susceptible to a purely analytical solution". Probably, one of the difficulties in color image segmentation is how to define color conception, since an understanding of colors in an image varies by different observers.

David Oswald is with the Department of Computer & Electrical Engineering and Computer Science, California State University, Bakersfield, CA 93311, USA.

Wei Li is with Robotics State Key Laboratory, Shenyang Institute of Automation, Chinese Academy of Sciences Shenyang, Liaoning 110016, China, and Department of Computer & Electrical Engineering and Computer Science, California State University, Bakersfield, CA 93311, USA. (phone: +661-654-6747; fax: +661-654-6960; Email: wli@csub.edu). W. Li is the author to whom correspondence should be addressed.

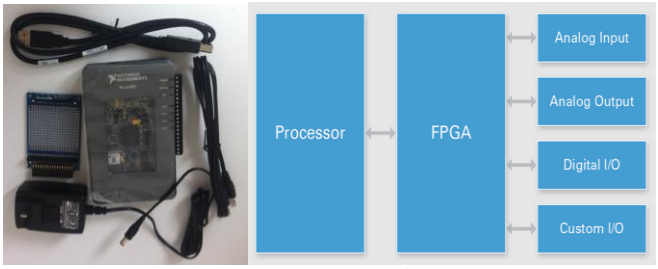
Linwei Niu is with Department of Mathematics & Computer Science, West Virginia State University, WV 25112, USA.

Jing Zhang is with Robotics State Key Laboratory, Shenyang Institute of Automation, Chinese Academy of Sciences Shenyang, Liaoning 110016, China.

Yan Li is with Robotics State Key Laboratory, Shenyang Institute of Automation,Chinese Academy of Sciences Shenyang, Liaoning 110016, China.

Jianchen Yu is with Robotics State Key Laboratory, Shenyang Institute of Automation, Chinese Academy of Sciences Shenyang, Liaoning 110016, China.

Fuchun Sun is with is with Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.



(a) (b)

Fig. 1. (a) NI myRIO embedded system package, including NI myRIO, MXP connector, accessories, power supply, and cables. (b) The NI myRIO (reconfigurable I/O) architecture. It is based on four components: a processor, a reconfigurable FPGA, inputs and outputs, and graphical design software.

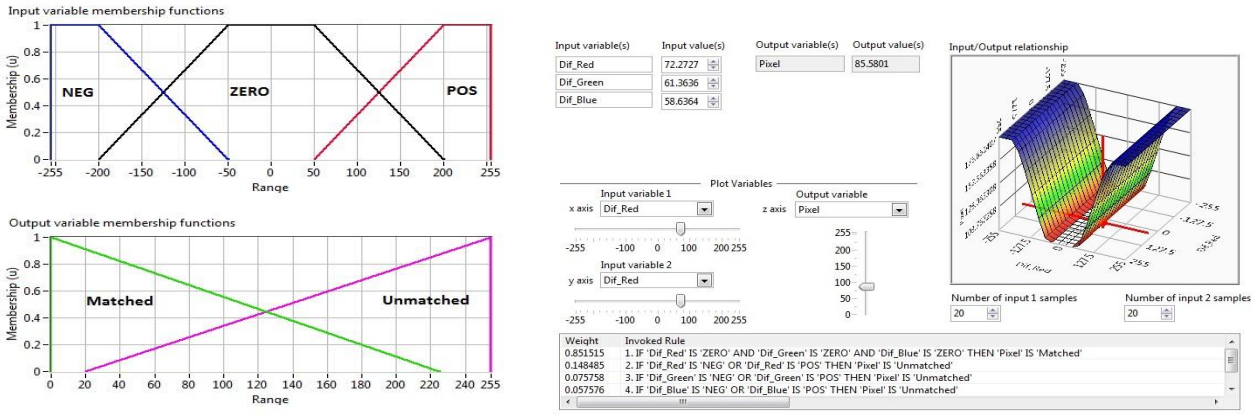
Color based tracking is an area seeing great interest in robot manipulation and control. In robotics, vision cannot be considered an isolated component, but it is instead a part of a system resulting in an action. Thus, in robotics the vision research should include consideration of the control of the system, in other words, the entire perception-action loop. In this case, computational time for color detection is considerable.

Considering uncertainty and ambiguity in many real applications, we propose an approach to segmenting a color image based on a fuzzy color extractor (FCE) [6][8]. The FCE is directly extended from the fuzzy grey-level extractor, which is applied to extract the landmarks on roads for autonomous vehicle navigation [9]-[11]. The FCE is described by the following rules:

If *the color components of a pixel closely match a reference color pattern*
Then *extract this pixel*
Else *do not extract this pixel*

In order to extract certain colors from images in real-time, this paper attempts to implement the fuzzy color extractor on NI myRIO embedded device, which features the Xilinx Zynq-7010 all-programmable system including a dual-core ARM® Cortex™-A9 real-time processor and an Artix-7 FPGA with customizable I/O.

To implement the fuzzy color extractor on NI myRIO, the fuzzy logic, real-time, and FPGA modules of LabVIEW 2013 are required. LabVIEW is a system-design platform and development environment for a visual programming language. First, we define the fuzzy membership functions and fuzzy rules for the fuzzy color extractor using the fuzzy logic module. Second, we implement the fuzzy color extractor using real-time module. Next, we evaluate the color extraction performance under varying light conditions. Finally, we test the



(a) (b)

Fig. 2. (a) Membership functions of input and output variables defined for the FCE using *FL New Fuzzy System.vi* in LabVIEW 2013. (b) Fuzzy rules defined for the FCE and the weights of the invoked rules.

implemented fuzzy extractor on a DaNI robot for color detection.

II. Fuzzy Color Extractor

The partitioning of a color image into some sub-images that represent different objects is called segmentation. In many color-based vision applications, however, colors in an image are ambiguous due to environment uncertainty. The work in [x] proposes the fuzzy color extractor to deal with such uncertainty. In this study, colors of an image are described in the RGB space, where colors are represented by their red, green, and blue components in an orthogonal Cartesian space. The color of each pixel $p(m, n)$ denoted by $p(m, n)_{RGB}$ is processed to separate its red, green, and blue components ($p(m, n)_R$, $p(m, n)_G$, $p(m, n)_B$). The FCE extracts a cluster of colors based on a defined color pattern (CP or CP_{RGB}) or a seed. The CP_{RGB} is either directly defined by its RGB components (CP_R, CP_G, CP_B) or determined by a pixel in the image. The color component differences between $p(m, n)_{RGB}$ and CP_{RGB} are calculated as follows:

$$\begin{cases} \mathrm{dif}(m,n)_R = p(m,n)_R - CP_R \\ \mathrm{dif}(m,n)_G = p(m,n)_G - CP_G, \quad 0 \le m \le M,\ 0 \le m \le N, \\ \mathrm{dif}(m,n)_B = p(m,n)_B - CP_B \end{cases} \quad (1)$$

where M and N indicate the size of an array which holds the image. The following fuzzy rules are applied to $\mathrm{dif}(m, n)_R$, $\mathrm{dif}(m, n)_G$, and $\mathrm{dif}(m, n)_B$:

If $\mathrm{dif}(m, n)_R$ *and* $\mathrm{dif}(m, n)_G$ *and* $\mathrm{dif}(m, n)_B$ *are* **`Zero`**
Then $p(m, n)$ *is* **`Matched`**
If $\mathrm{dif}(m, n)_R$ *or* $\mathrm{dif}(m, n)_G$ *or* $\mathrm{dif}(m, n)_B$ *is* **`Negative`** *or* **`Positive`**
Then $p(m, n)$ *is* **`Unmatched`**

Both the rules indicate that the pixel, $p(m, n)$, belongs to the object to be extracted, if the Euclidean distances between $p(m, n)_{RGB}$ and CP_{RGB} along the three axes in RGB coordinate system are small enough; otherwise, $p(m, n)$ does not belong to the object. We use the Fuzzy Logic module of LabVIEW 2013 to define membership functions for the FCE. The top figure in Fig. 1(a) shows the membership functions ($\mu_N(x)$, $\mu_Z(x)$, $\mu_P(x)$) for the input fuzzy variables (**`Negative`**, **`Zero`**, **`Positive`**)

$$\mu_N(x) = \begin{cases} 1 & -255 \le x < -\alpha_2 \\ \dfrac{(x+\alpha_1)}{(\alpha_1-\alpha_2)} & -\alpha_2 \le x < -\alpha_1 \\ 0 & -\alpha_1 \le x \le 255 \end{cases}, \quad (2)$$

$$\mu_Z(x) = \begin{cases} 0 & -255 \le x < -\alpha_2 \\ \dfrac{(x+\alpha_2)}{(\alpha_2-\alpha_1)} & -\alpha_2 \le x < -\alpha_1 \\ 1 & -\alpha_1 \le x < \alpha_1 \\ \dfrac{(x-\alpha_1)}{(\alpha_2-\alpha_1)} & \alpha_1 \le x < \alpha_2 \\ 0 & \alpha_2 \le x \le 255 \end{cases}, \quad (3)$$

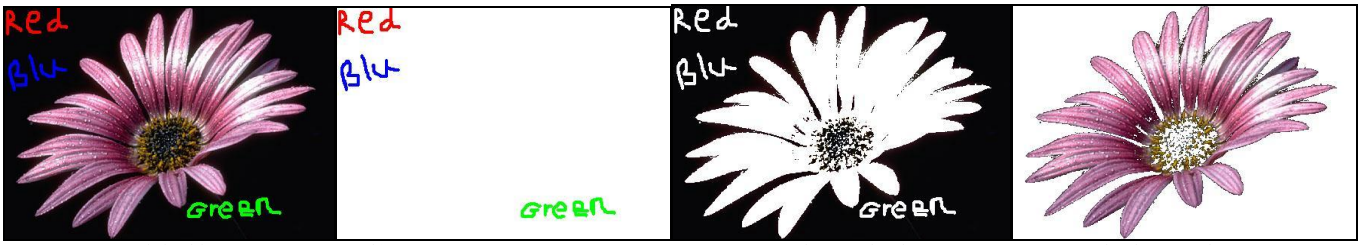
$$\mu_P(x) = \begin{cases} 0 & -255 \le x < \alpha_1 \\ \dfrac{(x-\alpha_1)}{(\alpha_2-\alpha_1)} & \alpha_1 \le x < \alpha_2 \\ 1 & \alpha_2 \le x \le 255 \end{cases}, \quad (4)$$

where $\alpha_1 = 50$ and $\alpha_2 = 200$ for this study. The bottom figure in Fig. 1(a) shows the membership functions ($\mu_M(x)$, $\mu_U(x)$) for the output fuzzy variables (**`Matched`**, **`Unmatched`**)

$$\mu_M(x) = \begin{cases} \dfrac{(\rho_M - x)}{\rho_M} & 0 \le x \le \rho_M \\ 0 & \rho_M \le x \le 255 \end{cases}, \quad (5)$$

$$\mu_U(x) = \begin{cases} 0 & 0 \le x \le \rho_U \\ \dfrac{(x-\rho_U)}{(255-\rho_U)} & \rho_U \le x \le 255 \end{cases}, \quad (6)$$

where $\rho_M = 235$ and $\rho_M + \rho_U = 255$. Based on $\mathrm{dif}(m, n)_R$, $\mathrm{dif}(m, n)_G$, and $\mathrm{dif}(m, n)_B$, the fuzzy rules produce the weight w_m for **`Matched`** and the weight w_u for **`Unmatched`** by



(a) Original image. (b) "RED," "BLUE," "GREEN." (c) Background color. (d) Flower colors.

Fig. 3: Extract colors from the flower image by the fuzzy color extractor

$$\begin{aligned} w_{\mathrm{m}} &= \min\{w_{\mathrm{m}(R)}, w_{\mathrm{m}(G)}, w_{\mathrm{m}(B)}\} \\ w_{\mathrm{u}} &= \max\{w_{\mathrm{m}(R)}, w_{\mathrm{m}(G)}, w_{\mathrm{m}(B)}\} \end{aligned} . \quad (7)$$

Fig. 1(b) shows the weights of invoked fuzzy rules in the output domain while w_{m} and w_{u} cutting $\mu_{\mathbf{M}}(x)$ and $\mu_{\mathbf{U}}(x)$. A crisp output value, $\Delta\rho_{\mathrm{F}}$, is calculated by the centroid defuzzification method

$$\Delta\rho_{\mathrm{F}} = \frac{\int \mu_{\mathrm{out}}(x)xdx}{\mu_{\mathrm{out}}(x)dx} , \quad (8)$$

where $\mu_{\mathrm{out}}(x)$ represents the envelope function of the areas cut by w_{m} and w_{u} in fuzzy output domain. If $\Delta\rho_{\mathrm{F}} < \varepsilon$, $p(m, n)$ is extracted; otherwise, $p(m, n)$ is not extracted, where ε is a threshold. The FCE can be understood as a mapping operator between Euclidean distances $\{\mathrm{dif}(m, n)_{\mathrm{R}}, \mathrm{dif}(m, n)_{\mathrm{G}}, \mathrm{dif}(m, n)_{\mathrm{B}}\}$ in the RGB space and a difference $\Delta\rho_{\mathrm{F}}$ in the intensity space under a fuzzy metric.

We take the images in Fig. 3 as examples to demonstrate how to extract some interesting colors from the original image –"flower" by the FCE. First, we define three color patterns: (255, 0, 0)–red, (0, 255, 0)–green, and (0, 0, 255)–blue, to extract three words "RED", "GREEN", and "BLUE" from–"flower". Based on the three CPs, the FCE extracts the three words from the image –"flower" as shown in Fig. 3(b). Meanwhile, we use a color pattern $p(0, 0)_{\mathrm{RGB}}$ with color components (6, 6, 8) to remove the background from the image as shown in Fig. 3(c). The background color can also be extracted by the black color pattern (0, 0, 0). After we extract the words "RED," "GREEN," and "BLUE," and the background, we obtain the color components of the flower petal, as shown in Fig. 3(d).

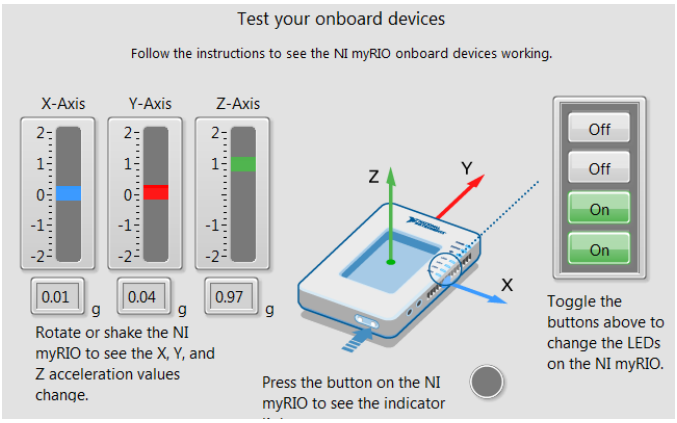


Fig. 4. Configure and test the three embedded accelerometers align with X-Axis, Y-Axis, and Z-Axis embedded in NI myRIO.

III. CONFIGURE NI MYRIO

In order to run the FCE in real-time, we implement the FCE model on an NI myRIO. Fig. 1(b) shows the LabVIEW reconfigurable I/O (RIO) architecture, which is based on four components: a processor, a reconfigurable FPGA, inputs and outputs, and graphical design software. The NI myRIO places a dual-core ARM® Cortex™-A9 real-time processor and an Artix-7 FPGA. It includes the Xilinx Zynq system, 10 channels analog inputs, 6 channel analog outputs, 40 digital I/O lines, 3 embedded accelerometers, a WIFI adapter, and the additional analog input and output available through 3.5 mm Audio Jacks. Combined, these components provide the ability to rapidly create custom hardware circuitry with high-performance I/O and unprecedented flexibility in system timing control. LabVIEW 2013 offers the LabVIEW for myRIO programming environment providing access to wizards, customized palette views, and other resources useful for developing applications for the myRIO target.

In order to test the myRIO, we establish a communication between the myRIO and its host computer through a WIFI access point or a USB cable and run the myRIO wizard to check the firmware on the myRIO and to test LEDs on the myRIO and the X-Axis, Y-Axis, and Z-Axis accelerometers embedded in the myRIO, which indicate if the myRIO I/Os function or not. Fig. 4 shows that we can rotate or shake the NI myRIO to see the X, Y, Z acceleration values change. We can use NI Measurement & Automation Explorer (MAX) to check the status of myRIO. If we can see the myRIO ID under *Remote Systems* folder in MAX, the myRIO configuration is accomplished and it is ready to use.

After testing the myRIO, we start the LabVIEW myRIO programming environment to specify a myRIO project for developing our algorithms using LabVIEW VIs. The myRIO ID with its IP address should appears in the project. The *vi.lib* and *instr.lib* libraries list under the *Dependencies* folder for users to develop their own programs. For this study, we need the Dynamic-Link Libraries *nivision.dll* and *niimaqdx.dll*. The *nivision.dll* library acquires, saves, and displays images from thousands of different cameras, works with all NI frame grabbers, NI vision systems, and NI Smart Cameras. The *niimaqdx.dll* library works with USB3 Vision, GigE Vision, IP (Ethernet), and IEEE 1394 devices. We use the *niimaqdx.dll* library to connecting a Logitech webcam, an AXIS network camera, or a Built-in iSight camera along with a laptop computer. For our study, we also need the Dynamic-Link

library *NiFpqaLv.dll* which most often has a description of LV-FPGA Adapter DLL for FPGA Interface C API. All Dynamic-Link libraries can be loaded and executed in any running process. The *Build Specifications* folder includes build configurations for source distributions and other types of builds available in LabVIEW toolkits and modules. Using the LabVIEW RIO architecture approach, we can take advantage of LabVIEW system design software to offer NI myRIO users a spectrum of programming complexity.

IV. Implementation of the FCE using LabVIEW

Fig. 5 shows the diagram of *FuzzColorExtractorLV.vi* for implementing the FCE with four sections: acquiring image, creating membership functions and fuzzy rules for the FCE, processing image using parallel computation, and displaying the segmented images.

The *acquiring image* section first uses the *Open Camera.vi* from *niimaqdx.dll* to open a camera, to query the camera for its capabilities, to load a camera configuration file, and to create a unique reference to the camera. The *acquiring image* section also uses *IMAQ create.vi* to create a temporary location for an image. *IMAQ create.vi* specifies an image boarder size, an image type, such as RGB(U32) or HSL(U32), and an image name. Then, the *acquiring image* section uses *snap.vi* from *niimaqdx.dll* to configure, start, acquire, and unconfigure a snap acquisition. Use a snap for low-speed or single-capture where ease of programming is essential.

LabVIEW offers two ways to design a fuzzy system. The first way to design a fuzzy system is to use *Fuzzy System Designer*. By opening a *Fuzzy System Designer* window under *Control Design and Simulation* from Tools menu, we can define input and output fuzzy variables, specify fuzzy rules and chose a defuzzification method. The designed fuzzy system is saved in a .fs file. The second way is to use the *Fuzzy Logic Palette*. *FL New Fuzzy System.vi* in the palette creates a fuzzy system. *FL Fuzzy Controller.vi* implements a fuzzy controller in a fuzzy system to be specified. *FL SAVE Fuzzy System.vi* saves a designed fuzzy system to a .fs file. *FL Load Fuzzy System.vi* loads a designed fuzzy system from a .fs file. We can specify fuzzy variables under *Variables Palette*, and define membership functions and fuzzy rules under *Membership* and *Rules Palettes*.

For implementing the FCE, we use *FL New Fuzzy System.vi* to design a fuzzy controller with three inputs and two outputs. Fig. 1(a) shows their membership functions defined according

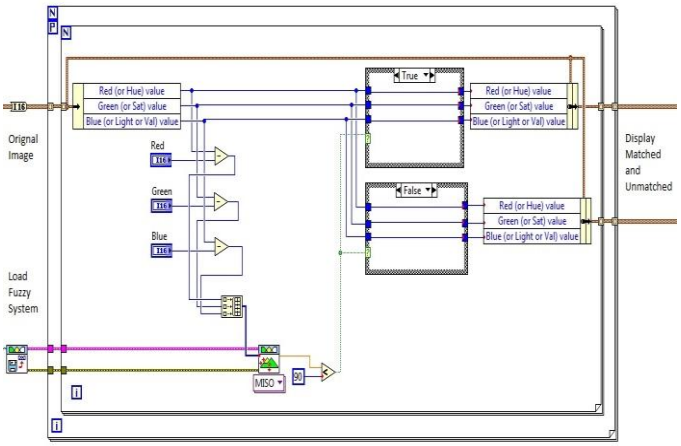
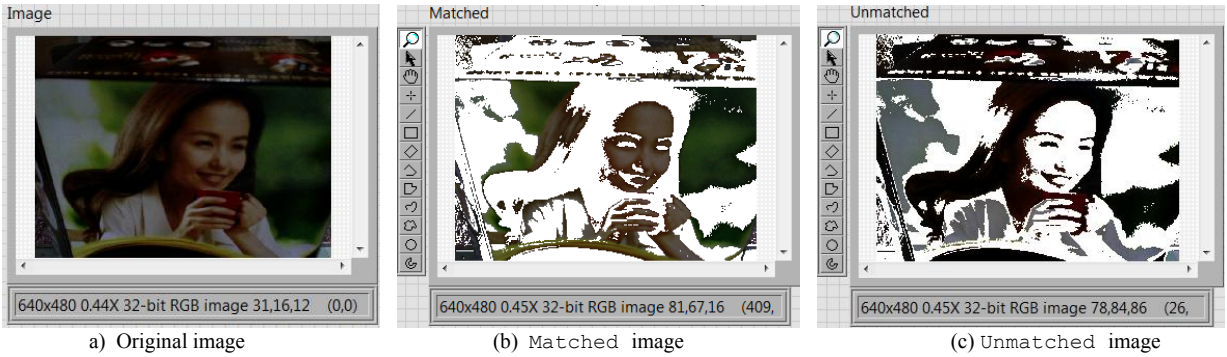


Fig. 5. The diagram of *LVFuzzColorExtractor.vi* includes four sections: acquiring image, creating membership functions and fuzzy rules for the FCE, processing image using parallel computation, and displaying the segmented images.

to Eqs (2)-(6). We specify the four fuzzy rules and chose the centroid method for defuzzification. *FL New Fuzzy System.vi* is able to visualize the designed fuzzy controller. Fig. 1(b) shows the FCE control surface, the output from the FCE $\Delta\rho_F$ =85.6 and the weights of the invoked rules when dif$(m, n)_R$=72.3, dif$(m, n)_G$=61.4, and dif$(m, n)_B$=58.6.

FuzzColorExtractorLV.vi uses double for-loops to process each pixel in an image taken by a camera. Please note that the outside loop uses a parallel computing structure to speed up image processing due to dual-core ARM® Cortex™-A9 real-time processor. An *unbundle cluster by name* VI decomposes colors of the original image into red, green, and blue components whose differences from a color pattern are calculated. As discussed above, the inputs to the FCE are dif$(m, n)_R$, dif$(m, n)_G$, and dif$(m, n)_B$, and the output from the RGB is $\Delta\rho_F$. The threshold, ε=90, is chosen to classify each pixel. If $\Delta\rho_F$ associated with a pixel is greater than ε, the pixel is classified into a **`Matched`** image and, otherwise, into an **`Unmatched`** image.

After classifying each pixel, *FuzzColorExtractorLV.vi* uses *IMAQ ColorValueToInteger.vi* to convert the two **`Matched`** and **`Unmatched`** clusters composed of three color components in mode (R, G, B) into colors encoded in the form of an



a) Original image (b) `Matched` image (c) `Unmatched` image

Fig. 6: Original image is taken by a Built-in iSight camera along with a MAC Pro laptop and is processed based on a color pattern chosen in the area with "light green" color on the image.

(a) DaNI robot (b) An image taken by the network camera

Fig. 7. The DaNI robot platform is made by National Instruments with an open architecture mobile robot. An AXIS M1011 network camera mounted on the DaNI robot sends images around the environment to the host computer through a wireless router.

unsigned 32-bit integer. For displaying the **`Matched`** and **`Unmatched`** images, *IMAQ ArrayToColorImage.vi* needs to create them from 2D arrays representing color pixels in unsigned 32-bit integer.

Fig. 6(a) shows an image taken under very poor illumination condition by a Built-in iSight camera along with a MAC Pro laptop. The image resolution is 640x480 with RGB (32-bit format). The color pattern CP_{RGB}=(46, 67, 29) is selected from "light green" areas on the original image to segment the image into a **`Matched`** image and an **`Unmatched`** image, as shown in Fig. 6(b) and Fig. 6(c), respectively. Evidently, the dominant colors in the "light green" areas are different from the pure green color with the RGB components (0, 255, 0) due to the pure illumination condition.

Because the dual-core ARM Cortex-A9 processor on NI myRIO runs the NI Linux Real-Time OS, users can also choose to program the processor in C or C++ using the Eclipse IDE.

V. Color Extraction via a DaNI Robot

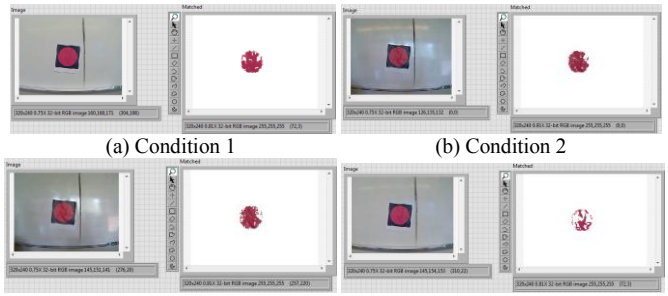
This section discusses applying the FCE for color detection via a DaNI robot made by National Instruments, as shown in Fig. 7(a). For color tracking, we mount an AXIS M1011 network camera mounted on the DaNI robot that sends images around the environment to the host computer through a wireless router. Fig. 7(b) shows an image taken by the camera.

The DaNI robot platform uses the sbRIO-9632 embedded control and acquisition device which integrates a 400 MHz industrial processor, a user-reconfigurable field-programmable gate array (a 2M gate Xilinx Spartan FPGA), and I/O on a single printed circuit board. The DaNI robot features 110 3.3V (5V tolerant/TTL compatible) digital I/O lines, 32 single-ended/16 differential 16-bit analog input channels at 250 kS/s, and four 16-bit analog output channels at 100 kS/s. It also has three connectors for expansion I/O using board-level NI C Series I/O modules. The sbRIO-9632 includes a 19 to 30 VDC power supply input range, 128 MB of DRAM for embedded operation, and 256 MB of nonvolatile memory for storing programs and data logging. Both the sbRIO-9632 and myRIO have the same LabVIEW reconfigurable I/O (RIO) architecture, so we can almost directly deploy the FCE algorithm developed for the myRIO on the sbRIO-9632.

Color as a distinct feature is widely used for objects representation and tracking as the objects can be represented by their dominant colors. However, color-based tracking is often influenced by illumination variation. Our experiments are to test the robustness of the FCE for extracting the dominant colors of objects to be tracked under different illumination conditions.

In our test environment, there are two rows of lights on the ceiling. Varying illumination conditions include 4 cases. Condition 1: all lights turn on. Condition 2: the first row of lights turns on, but the second row of lights turns off. Condition 3: the first row of lights turns off, but the second row of lights turns on. Condition 4: both the rows of lights turn off. We configure the network camera to snap a capture with a resolution of 320x240 (32-bit RGB).

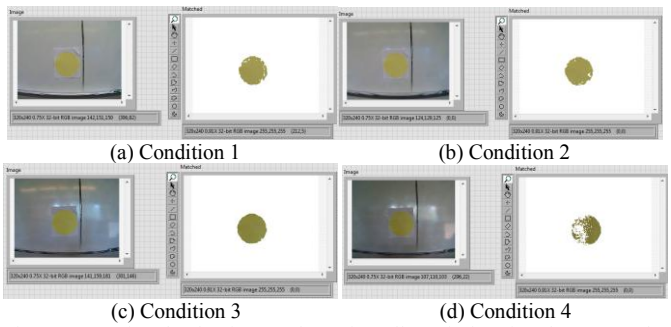
We conduct a number of experiments to evaluate the robustness of the FCE for extracting the dominant colors based on a given color pattern. Fig. 8 shows the experiments on extracting dominant colors of a red circle taken under the 4 illumination conditions. The experimental procedure is described as follows. We select the color pattern CP_{RGB}=(200, 0, 0) on the red circle in the original image taken under Condition 1. Then, the FCE uses the color pattern to extract the dominant colors of the red circle in the images taken under all the illumination conditions. Fig. 8(a) shows the original image (the left one) taken by the network camera under Condition 1 and, based on the color pattern CP_{RGB}=(200, 0, 0), the `Matched` image (the right one) representing the dominant colors of the red circle extracted by the FCE. Fig. 8(b)-(d) show the original images (the left ones) taken by the network camera when the



(a) Condition 1 (b) Condition 2

(c) Condition 3 (d) Condition 4

Fig. 8: Extracting the dominant colors of a red circle using the FCE under four illumination conditions. Condition 1: all lights turn on. Condition 2: the first row of lights turns on, but the second row of lights turns off. Condition 3: the first row of lights turns off, but the second row of lights turns on. Condition 4: both the rows of lights turn off.



(a) Condition 1 (b) Condition 2

(c) Condition 3 (d) Condition 4

Fig. 9: Extracting the dominant colors of a yellow circle using the FCE under four illumination conditions. Condition 1: all lights turn on. Condition 2: the first row of lights turns on, but the second row of lights turns off. Condition 3: the first row of lights turns off, but the second row of lights turns on. Condition 4: both the rows of lights turn off.

environment illumination switches to Conditions 2, 3, or 4, and based on the same color pattern CP_{RGB}=(200, 0, 0), the `Matched` images (the right ones) representing the dominant colors of the red circle extracted by the FCE, respectively. Although the pixel numbers of the dominant colors of the red circle are inference due to changing the illumination conditions, the sizes of the dominant colors are almost the same, which are important for us to recognize the red circle. Fig. 9(a)-(d) show another example of the experiments on detecting dominant colors of a yellow circle under the 4 illumination conditions. It can be seen that, based on the color pattern CP_{RGB}=(200, 200, 0) selected from the yellow circle in the original image, as shown in the left one in Fig. 9(a), the FCE extracts the dominant colors of the orange circle under all the illumination conditions.

VI. Conclusions

There are two major issues in developing algorithms for color-based tracking via a robot. First, the most color extraction algorithms are greatly influenced by the illumination conditions. We propose the FCE-based color extraction algorithm to deal with the environment uncertainty and illumination variations. Second, in order to extract certain colors from robot tracking in real-time, we implement a fuzzy color extractor on NI myRIO embedded device, which features the Xilinx Zynq-7010 all-programmable system including a dual-core ARM® Cortex™-A9 real-time processor and an Artix-7 FPGA with customizable I/O.

How to select the color patterns plays an important role in achieving a good performance of color extraction [12]. In our further research, we will investigate adaptive algorithms for determining the color pattern and report the experiments on color tracking via a DaNI robot.

References

[1] K. S. Fu, and J. K. Mui, "A survey on image segmentation," *Pattern Recognition*, 13, 3-16 (1981).

[2] S. K. Pal et al., "A review on image segmentation techniques", *Pattern Recognition*, 29, 1277-1294 (1993)

[3] H. D. Cheng et al., "Color image segmentation: Advances & Prospects," *Pattern Recognition*, 34, 2257-2281, 2001.

[4] H. Q. Liu, L. C. Jiao, and F. Zhao, "Non-local spatial spectral clustering for image segmentation," *Neurocomputing,* 74(1), 461-471, 2010.

[5] D. MúJica-Vargas, F. J. Gallegos-Funes, and A. J. Rosales-Silva, "A fuzzy clustering algorithm with spatial robust estimation constraint for noisy color image segmentation," *Pattern Recognition Letters*, 34(4), 400-413, 2013.

[6] K. N. Plataniotis, and A. N. Venetsanopoulos, [*Color image processing and applications]*, Springer, 2000.

[7] W. Li, Y. Li, and J. Zhang, "Fuzzy color extractor based algorithm for segmenting an odor source in near shore ocean conditions," *FUZZ-IEEE* 2256-2261, 2008.

[8] G. Zhao, Y. Li, G. Chen, and Q. Meng, and W. Li, "A fuzzy-logic based approach to color segmentation," Proc. SPIE 8739, Sensors and Systems for Space Applications VI, 873910, 2013,

[9] W. Li, G. Lu, and Y. Wang, "Recognizing white line markings for vision-guided vehicle navigation by fuzzy reasoning," *Pattern recognition letters*, 18(8), 771-780, 1997.

[10] W. Li, X. Jiang, and Y. Wang, "Road recognition for vision navigation of an autonomous vehicle by fuzzy reasoning," *Fuzzy Sets and Systems,* 93(3), 275-280, 1998.

[11] W. Li, F. M. Wahl, J. Z. Zhou, et al., "Vision-based behavior control of autonomous systems by fuzzy reasoning," *Sensor Based Intelligent Robots*, Springer Berlin Heidelberg, 311-325, 1999.

[12] R. Adams, and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6), 641-647, 1994.