IOSR Journal of Electrical And Electronics Engineering (IOSRJEEE)
ISSN : 2278-1676 Volume 2, Issue 4 (Sep.-Oct. 2012), PP 29-33
www.iosrjournals.org

Contrive and Effectuation of Active Distance Sensor Using MATLAB and GUIDE Package

Premnath Srikanthan[1], Vigneshwaran S K[2]
[1]*Department of Electronics and Instrumentation Engineering,*
[2]*R.M.D Engineering College, Kavaraipettai-601206, India*

Abstract: *This paper describes contrive and effectuation of active distance sensing device using MATLAB (Matrix Laboratory) and GUIDE (Graphical User Interface Design Environment) software packages. Image acquisition toolbox in MATLAB acquires images through image acquisition device such as a Laptop based webcam or a digital video camera. Image processing toolbox in MATLAB processes the images, detects the desired objects and filters the noise present in the images. The program in M-file computes the distance between the desired objects. GUIDE enables user interaction with the device. Hence it acts as a distance sensor. The proposed method can be used in many large scale applications such as to find the distance between the moving cars/aircrafts with a high resolution camera and comparatively low cost with other equipments.*
Key words: *MATLAB, GUIDE, M-file, Image Processing*

I. Introduction

MATLAB is a high level programming language which is used widely in the scientific and engineering communities for research and development purposes. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notations. Technical professionals in worldwide rely on MATLAB to speed up their research, reduce analysis and development time, minimize their project expenses, and produce effective solutions. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems in a simpler way.

The Image Acquisition Toolbox software [1] is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox supports a wide range of image acquisition parameters such as acquiring images through many types of image acquisition devices, from professional grade frame grabbers to USB-based webcams, viewing a preview of the live video stream, triggering acquisitions (includes external hardware triggers), configuring callback functions that execute when certain actions occur and on bringing the image data into the MATLAB workspace. We connect directly to our hardware in the tool and can preview and acquire image data. We can log the data to MATLAB in several formats, and also generate an AVI file, right from the tool. The Image Acquisition Tool integrates a preview/acquisition area with Acquisition Parameters so that we can change circumstances and see the changes dynamically applied to our image data.

The list of hardware that the Image Acquisition Toolbox software supports can change in each acquaintance, since hardware support is frequently added. [2] is the best place to check for the most up to date listing. We can extend the capabilities of the toolbox by writing our own M-files or by using the toolbox in combination with other toolbox such as the Image Processing Toolbox software [3]. The toolbox supports a wide range of image processing operations such as spatial image transformations, morphological operations, neighborhood and block operations, and linear filtering and filter design environment.

The framework of this paper is significant for several reasons. First, by extending the capabilities of MATLAB and GUIDE to low-cost microcontrollers and common distance sensors, our method overcomes two limitations, *viz.*, the lack of advanced icon-based software interface for efficient development of computational algorithms and the lack of a GUI to allow intuitive interaction. Second, in the video documents to be processed, there is no restriction on the viewed scene or on the object characteristics (expression, pose, etc.). Illumination conditions could vary depending on the nature of the video [7]. Third, the ability to interface and program webcam using the intuitive graphical programming environment of MATLAB provides the flexibility and versatility of equipping a wide array of undergraduate-level laboratories at an economical cost and allows use of Matlab Central which makes even an absolute beginner create an application quickly in the industries. Fourth, the program can run without MATLAB software by creating a standalone '.exe' file from 'M-file' and port it other computers using MCR (MATLAB Component Run time).

II. Concept Overview

Design and implementation of dynamic distance sensor can be divided into four blocks: image acquisition, processing of image, development of distance computing algorithm and creating GUI for display.

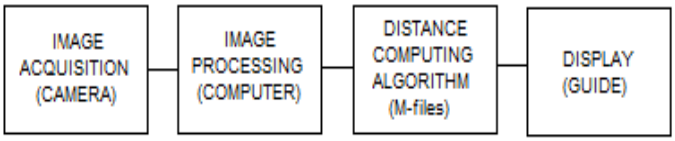


Fig. 1. Block diagram of design and implementation of dynamic distance sensor.

Image acquisition can be achieved by using a PC-based webcam or digital video camera. This device will capture the image, send it to the camera processor for further processing in computer. Its main function is to convert the light energy received into electrical signal.

Image processing involves conversion of RGB color images into gray scale images, setting of threshold levels, saturation of the features into binary images and setting of cut-off values to remove noise in the binary image.

Computational algorithm for sensing distance is done through the software program as M-files. MATLAB come with 'GUIDE' tool is similar to Visual Basic (VB) environment which is an event-driven programming language helps the users to implement graphic interfaces. It even generates a skeleton M-file (plus figure file with a '.fig' extension) which can further use to link other events through *callback* function (M-file which compute the distance between the objects).

III. Image Acquisition

Image acquisition can be achieved by using a PC-based webcam or digital video camera [5]. MATLAB has built –in adaptors to access the camera. The instruction ***imaqhwinfo*** returns the information about all the adaptors available in the system as follows:

ans = InstalledAdaptors: {'coreco' 'winvideo'}
MATLABVersion: '7.6 (R2008a)'
ToolboxName: 'Image Acquisition Toolbox'
ToolboxVersion: '3.1 (R2008a)'

The ***info=imaqhwinfo*** instruction returns information about a specific device accessible through a particular adaptor. The output of the ***info=imaqhwinfo('winvideo')*** instruction is:

info =
AdaptorDllName:'C:\Program Files\MATLAB\R2008a\toolbox\imaq\imaqadaptors\win32\mwwinvideoimaq.dll'
AdaptorDllVersion: '3.1 (R2008a)'
AdaptorName: 'winvideo'
DeviceIDs: {[1] [2]}
DeviceInfo: [1x2 struct]

The output of the ***dev_info=imaqhwinfo('winvideo',1)*** instruction is:

dev_info =
DefaultFormat: 'RGB24_640x480'
DeviceFileSupported: 0
DeviceName: 'Laptop Integrated Webcam'
DeviceID: 1
ObjectConstructor: 'videoinput('winvideo', 1)'
SupportedFormats: {1x19 cell}

Again, the ***obj=videoinput('adaptorname', deviceID,' format)*** instruction constructs a video input object where the *adaptorname* string specifies the name of the device adaptor that the object *obj* is associated d*eviceID* is the identifier of the device in numerical number. If *deviceID* is not specified, the first available *deviceID* is used. The *format* string specifies the video *format* for the object. If the format is not specified the device default format is used.

IV. IMAGE PROCESSING

A. Binary images (B&W)

A binary image is a logical array of 0's and 1's. Numeric-array images consisting of 0's and 1's are converted into binary, with '1' indicating white color (maximum intensity) and '0' indicating black color (minimum intensity).

B. RGB image

A red, green and blue (RGB) image is an MxNx3 array of color pixels, where each color pixel is a triplet corresponding to the red, green and blue components of an RGB image at a specific spatial location. An RGB image may be viewed as a stack of three gray-scale images that, when fed into the RGB color monitors, produce a color image. Eight bits are used to represent the pixel values of each component of the image. Thus an RGB image corresponds to 24 bits.

C. Setting RGB Threshold Values

In order to detect an object we need to find the threshold values of red, green and blue components in its image. Now red ball is of particular interest. The main technique is in the region of the red ball [5]:

- The threshold for red component (R_THRESHOLD) should be the least value of the red component found in the region of the red ball.
- The threshold for green component (G_THRESHOLD) should be the maximum value of the green component found in the region of the red ball.
- The threshold for blue component (B_THRESHOLD) should be the maximum value of the green component found in the region of the red ball.

The steps for setting RGB threshold values of the red ball follow:

- ✓ Take at least 10 snaps of the red ball at various angles
- ✓ Read each image.
- ✓ Display the image by using the function "imview".
- ✓ Note down the pixel values at the region of the red ball by moving the cursor at various places of the red ball.

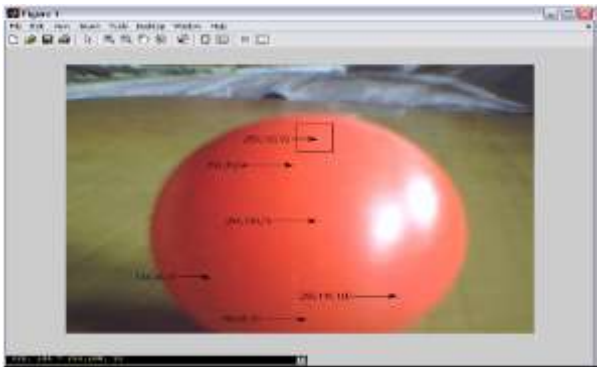


Fig. 2. Setting RGB threshold values.

- ✓ Here the least value of the red component we see is 198.So R_THRESHOLD can be taken as 195.
- ✓ Here the maximum value of the green component we see is 108. So G_THRESHOLD can be taken as 110.
- ✓ Here the maximum value of the blue component we see is 100.So B_THRESHOLD can be taken as 100

It is very essential that the threshold values are set before testing. Using this technique other than red ball other balls with green and blue colours can also be tracked. But for that we need to set the threshold values just as we did in case of the red ball.

D. Detection of the objects

Webcam capture a frame and store it in a variable, say, **rgb_image**. Extract the red, green and blue components of the images and store them in variables **fR, fG** and **fB** as follows:

```
fR = rgb_image (: , : , 1); %extracts the red component.
fG = rgb_image (: , : , 2); %extracts the green component.
fB = rgb_image (: , : , 3); %extracts the blue component.
```

Here **fR, fG** and **fB** are image matrices. Next, we need to find the red object in the image. (R_THRESHOLD=) 195, (G_THRESHOLD=) 108 and (B_THRESHOLD=) 100 are the numbers called threshold. Create a black and white (B&W) image array, I

```
I = ((fR>=195) & (fG<=108) & (fB>=100))          (1)
```

If the equation (1) satisfied, the pixel value of image is set to 1 else 0. Hence the B&W image is formed. Apart from the region of the red ball there are also some unwanted white regions in the images. These unwanted white regions are called noise. We need to filter these noisy parts before capturing the centre of the image as follows:

```
Se = strel ('disk', 20); %creates a flat, disk-shaped structuring element with radius 20.
B = imopen (I, Se); %morphological opening
Final = imclose (B,Se); %morphological closing
```

Morphological opening removes those regions of an object which cannot contain the structuring element; smoothes object contours, breaks thin connections and removes thin protrusions. Morphological closing also tends to smooth the contours of object besides joining narrow break and filling long, thin gulfs and holes smaller than the structuring element. Once we obtain the desired part, we can find the centre of the ball by computing all the connected components in a binary image.

[r, c] = find (L = = k) %k= 1,2,...n (2)

The above statement returns the row and column indices for all pixels belonging to the Kth object:

rbar = mean (r) (3)

cbar = mean (c) (4)

Equations (3) & (4) return the coordinates of the centre of mass. Even after filtering the image, the final image contains only one white region. But in case there is a computational fault due to excessive noise, we might have two connected components. A for loop from 1to n will ignore this, thus calculating the centre of mass for all objects. If there are no components in a frame, the control doesn't enter the loop and 'rbar' and 'cbar' remain initialised to zero.

V. DISTANCE SENSING AND GUIDE

Algorithm for detecting the objects returns the centre of mass for all objects. This program is written in a separate M-file. Knowing the coordinates of each object allows us find the distance between any two objects.

X = (cbar of object 1 – cbar of object 2)^2

Y = (rbar of object 1 – rbar of object 2)^2

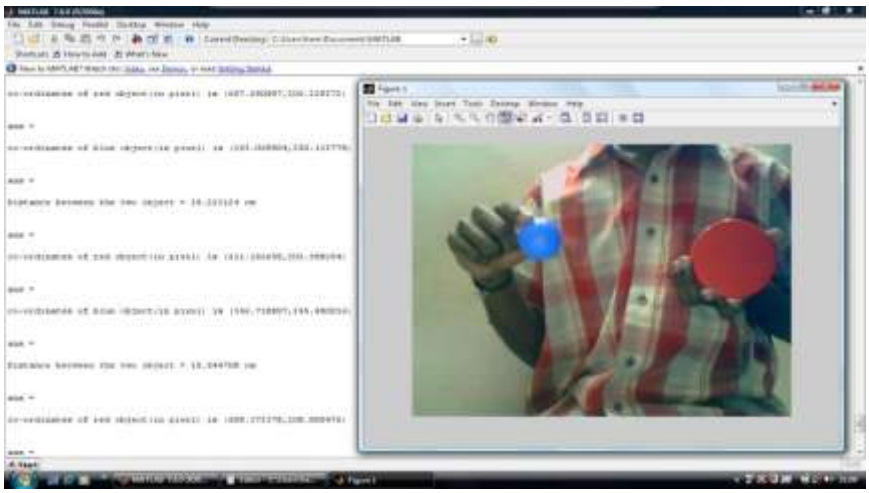
Distance_p = √(X + Y) (5)

Fig. 3. GUI of the dynamic distance sensor.

Since coordinates of the objects are in pixels, the distance is also in pixels. To convert the distance in pixels into distance in inches, we need to find ppi (pixel per inch) [6] value of the camera. This value depends on the camera resolution. Let N be ppi value of the camera, then distance in inches between the two objects is:

Distance_i = (Distance_p)/N (6)

The program for computing distance between two objects has been written in a separate M-file. GUI is created as shown



in Fig. 3 for the user interaction, which generates its own M-files [9], where we can edit it to call the functions for image processing and distance computing. The results of dynamic distance sensor using MATLAB and GUIDE can be interpreted from the figure shown below.

VI. CONCLUSION

This paper proposed an inexpensive method of designing a dynamic distance sensor using MATLAB and GUIDE. It can be successfully applied to broadcast videos of several hours and the problem of appearing/disappearing objects has been automatically handled by our tracking method. The MATLAB software and webcam interface has integrated the programming and tracking ability. The drawback of computing distance between the irregular objects can be overcome by this method.

References

[1] Matlab\help\getting started (image acquisition tool box).
[2] Online: http://www.mathworks.com/products/imaqe/related.html.
[3] Matlab\help\getting started (image processing tool box).
[4] Online: http://www.mathworks.com/products/imaqe/related.html.
[5] "Electronics For You" magazine, may 2009, Vol. 41 No.5.
[6] Online: http://www.wikipedia.org/pixel.
[7] Elise Arnaud, Brigitte Fauvet, Etienne M´emin, Patrick Bouthemy "A Robust Automatic Face Tracker Dedicated to broadcast video", IEEE 2005, 0-7803-9134-9/05/$20.00.
[8] Online: http://www.mathworks.com/support/books/. Mathworks Inc., "Matlab based books for use with matlab, simulink, toolboxes, and blocksets".
[9] Online: http://www.mathtools.com MathTools home page.
[10] Premnath Srikanthan , Chilambuchelvan A , "Design and Implementation of dynamic distance sensor using MATLAB and GUIDE" proceedings of International conference on Recent Development in Engineering and Technology, July 29, 2012 (conference style).