IMPLEMENTATION OF VARIOUS MATLAB COMMANDS USING IN DIFFERENT IMAGE PROCESSING TECHNIQUES

RAGHU KUMAR LINGAMALLU[1], K. NAVEEN KUMAR[2]
ASST PROFESSOR, PH.D SCHOLAR[1], PROFESSOR PH.D[2]
COMPUTER SCIENCE ENGINEERING[1], ELECTRONICS & COMMUNICATION ENGINEERING[2]
O.P.J.S UNIVERSTY, RAJASTHAN[1], ST. PETER'S ENGINEERING COLLEGE[2]

Abstract: Different methods employed for object detection are widely exploited covering application areas such as traffic monitoring, video surveillance and capturing various human activities and motion. The traditional methods that have earlier been proposed for detection are found to be beneficial if the detected object is properly identified. Moreover, minimizing the effect of dynamic changes as well as development of the algorithm which is robust of intensity variation is a challenging task. So this paper emphasizes on enhancement followed by detection which provide an ease for identification of distant objects. The task of detection was performed on a video using simple detectors and developing an approach for proper segmentation of moving objects. Moving body was detected from a video having a frame rate of 25 frames per second, total bit rate of 234 kbps and having 160x112 as frame width and height. Further operation of enhancement and detection was processed on MATLAB R2013b tool.

***Index Terms*-GREYSCALE, RGB IMAGE**

Introduction:

Matlab is a data analysis and visualization tool which has been designed with powerful support for matrices and matrix operations. As well as this, Matlab has excellent graphics capabilities, and its own powerful programming language. One of the reasons that Matlab has become such an important tool is through the use of sets of Matlab programs designed to support a particular task. These sets of programs are called toolboxes, and the particular toolbox of interest to us is the image processing toolbox.

Rather than give a description of all of Matlab's capabilities, we shall restrict ourselves to just those aspects concerned with handling of images. We shall introduce functions, commands and techniques as required. A Matlab function is a keyword which accepts various parameters, and produces some sort of output: for example a matrix, a string, a graph or figure. Examples of such functions are sin, imread, imclose. There are many functions in Matlab, and as we shall see, it is very easy (and sometimes necessary) to write our own. A command is a particular use of a function. Examples of commands might be

```
>> sin(pi/3)
>> c=imread('cameraman.tif');
>> a=imclose(b);
```

1. Images and Matlab:

Images may be considered as matrices whose elements are the pixel values of the image. In this chapter we shall investigate how the matrix capabilities of Matlab allow us to investigate images and their properties.

2. Greyscale images

Suppose you are sitting at your computer and have started Matlab. You will have a Matlab command window open, and in it the Matlab prompt.

```
>>
```

ready to receive commands. Type in the command window

```
>> w=imread('wombats.tif');
```

This takes the grey values of all the pixels in the greyscale image wombats.tif and puts them all into a matrix w. This matrix w is now a Matlab variable, and so we can perform various matrix operations on it. In general the imread function reads the pixel values from an image file, and returns a matrix of all the pixel values.

Two things to note about this command:

1. It ends in a semicolon; this has the effect of not displaying the results of the command to the screen. As the result of this particular command is a matrix of size 256x256, or with 65536 elements, we don't really want all its values displayed.

2. The name wombats.tif is given in single quote marks. Without them, Matlab would assume that wombats.tif was the name of a variable, rather than the name of a file. Now we can display this matrix as a grayscale image:

```
>> figure,imshow(w),pixval on
```

3 RGB Images:

Matlab handles 24-bit RGB images in much the same way as greyscale. We can save the color values to a matrix and view the result:

```
>> a=imread('autumn.tif');
>> figure,imshow(a),pixval on
```

Note now that the pixel values now consist of a list of three values, giving the red, green and blue components of the color of the given pixel. An important deference between this type of image and a grayscale image can be seen by the command.

```
>> size(a)
```

Which returns three values: the number of rows, columns, and "pages" of a, which is a three dimensional matrix, also called a multidimensional array. Matlab can handle arrays of any dimension, and 'a 'is an example. We can think of a as being a stack of three matrices, each of the same size. To obtain any of the RGB values at a given location, we use similar indexing methods to above. For example

```
>> a(100,200,2)
```

Returns the second colour value (green) at the pixel in row 100 and column 200. If we want all the colour values at that point, we can use

```
>> a(100,200,1:3)
```

4. Image Display:

We look more deeply at the use of the imshow() function, and how spatial resolution and quantization can affect the display and appearance of an image. In particular, we look at image quality, and how that may be affected by various images attributes. Quality is of course a highly subjective matter: no two people will agree precisely as to the quality of different images. However, for human vision in general, images are preferred to be sharp and detailed. This is a consequence of two properties of an image: its spatial resolution, and its quantization.

The imshow ( ) function:

Grey scale images

We have seen that if x is a matrix of type uint8, then the command

Imshow(x)

Will display x as an image. This is reasonable, since the data type uint8 restricts values to be integers between 0 and 255. However, not all image matrices come so nicely bundled up into this data type, and lots of Matlab image processing commands produces output matrices which are of type double. We have two choices with a matrix of this type:

1. Convert to type uint8 and then display,

2. Display the matrix directly.

The second option is possible because imshow will display a matrix of type double as a greyscale image as long as the matrix elements are between 0 and 1. Suppose we take an image and convert it to type double:

```
>> c=imread('caribou.tif');
>> cd=double(c);
>> imshow(c),figure,imshow(cd)
```

REFERENCES:

[1] M. Andriluka, S. Roth, and B. Schiele, “Pictorial structures revisited:People detection and articulated pose estimation,” in *Proc. IEEE Conf.Comput. Vis. Pattern Recog.*, 2009, pp. 1014–1021.

[2] M. Everingham, L. Van Gool, C. K.Williams, J.Winn, and A. Zisserman,“The pascal visual object classes (VOC) challenge,” *Int. J. Comput. Vis.*,vol. 88, no. 2, pp. 303–338, 2010.

[3] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, “Progressive searchspace reduction for human pose estimation,” in *Proc. IEEE Conf. Comput.Vis. Pattern Recog.*, 2008, pp. 1–8.

[4] M. P. Kumar, A. Zisserman, and P. H. Torr, “Efficient discriminativelearning of parts-based models,” in *Proc. IEEE 12th Int. Conf. Comput.Vis.*, 2009, pp. 552–559.

[5] V. Delaitre, I. Laptev, and J. Sivic, “Recognizing human actions in stillimages: A study of bag-of-features and part-based representations,” in*Proc. IEEE Brit. Mach. Vis. Conf.*, 2010.

[6] A. Gupta, A. Kembhavi, and L. S. Davis, “Observing human-object interactions:Using spatial and functional compatibility for recognition,” *IEEETrans. Pattern Anal. Mach. Intell.*, vol. 31, no. 10, pp. 1775–1789, Oct.2009.

[7] B. Yao and L. Fei-Fei, “Grouplet: A structured image representation forrecognizing human and object interactions,” in *Proc. IEEE Conf. Comput.Vis. Pattern Recog.*, 2010, pp. 9–16.

[8] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman, “Longterm arm and hand tracking for continuous sign language TV broadcasts,”in *Proc. 19th Brit. Mach. Vis. Conf.*, 2008, pp. 1105–1114.

[9] A. Farhadi and D. Forsyth, “Aligning ASL for statistical translation usinga discriminative word model,” in *Proc. IEEE Comput. Soc. Conf. Comput.Vis. Pattern Recog.*, 2006, pp. 1471–1476.

[10] L. Zhao and L. S. Davis, “Iterative figure-ground discrimination,” in *Proc.17th Int. Conf. Pattern Recog.*, 2004, pp. 67–70.