

Digital Image Processing Analysis using Matlab

Ahmed Abdullah¹, Wahid Palash², Ashiqur Rahman³,
Md. Kobirul Islam⁴, Shakh Md. Alimuzjaman Alim⁵

¹(B.Sc in EEE, American International University-Bangladesh, Bangladesh)

²(M.Sc in ICT, Bangladesh University of Engineering Technology, Bangladesh)

³(M.Sc in Information Technology (IT), Jahangirnagar University, Bangladesh)

⁴(CSE, Royal University of Dhaka, Bangladesh)

⁵(EEE, Royal University of Dhaka, Bangladesh)

ABSTRACT: *The intelligent analysis of video data is currently in wide demand because a video is a major source of sensory data in our lives. Text is a prominent and direct source of information in video, while the recent surveys of text detection and recognition in imagery focus mainly on text extraction from scene images. Here, this paper presents a comprehensive survey of text detection, tracking, and recognition in video with three major contributions. First, a generic framework is proposed for video text extraction that uniformly describes detection, tracking, recognition, and their relations and interactions. Second, within this framework, a variety of methods, systems, and evaluation protocols of video text extraction are summarized, compared, and analyzed. Existing text tracking techniques, tracking-based detection and recognition techniques are specifically highlighted. Third, related applications, prominent challenges, and future directions for video text extraction (especially from scene videos and web videos) are also thoroughly discussed. To this aim, a supervised DNN is trained to project the input samples into a discriminative feature space, in which the blur type can be easily classified. Then, for each blur type, the proposed GRNN estimates the blur parameters with very high accuracy. Experiments demonstrate the effectiveness of the proposed method in several tasks with better or competitive results compared with the state of the art on two standard image data sets.*

Keywords – Matlab, Image Processing, Web video, Image Resolution, 3D Scans

I. IMAGE PROCESSING

In imaging science, Image Processing is processing of images using mathematical operations by using any form of signal processing for which the input is an image, a series of images, or a video, such as a photograph or video frame [1]; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Images are also processed as three-dimensional signals where the third-dimension being time or the z-axis. Image processing usually refers to digital image processing, but optical and analog image processing also are possible. This article is about general techniques that apply to all of them. The acquisition of images (producing the input image in the first place) is referred to as imaging. Closely related to image processing are computer graphics and computer vision. In computer graphics, images are manually *made* from physical models of objects, environments, and lighting, instead of being acquired (via imaging devices such as cameras) from *natural* scenes, as in most animated movies. Computer vision, on the other hand, is often considered *high-level* image processing out of which a machine/computer/software intends to decipher the physical contents of an image or a sequence of images (e.g., videos or 3D full-body magnetic resonance scans). In modern sciences and technologies, images also gain much broader scopes due to the ever growing importance of scientific visualization (of often large-scale complex scientific/experimental data). Examples include microarray data in genetic research, or real-time multi-asset portfolio trading in finance. It is among rapidly growing technologies today [2], with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too. Image processing basically includes the following three steps.

- Importing the image with optical scanner or by digital photography [3].
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.

- Output is the last stage in which result can be altered image or report that is based on image analysis.

II. ALGORITHM

Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing [4]. The three general phases that all types of data have to undergo while using digital technique are Pre- processing, enhancement and display, information extraction.

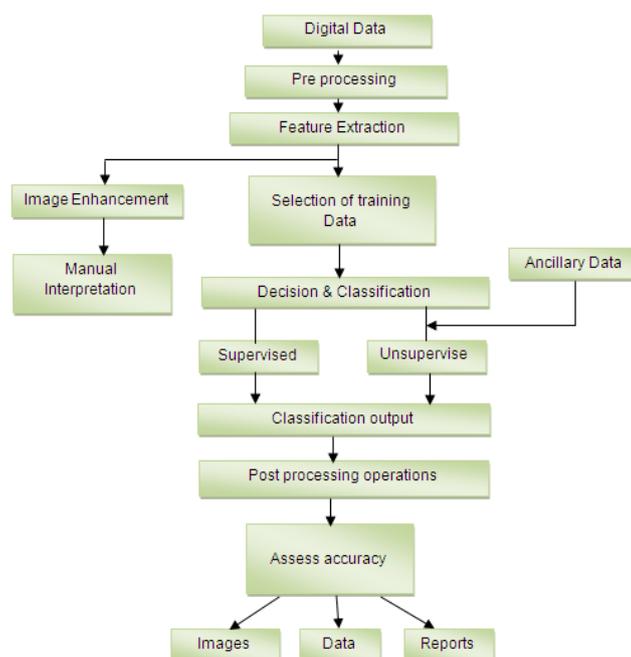


Fig 1: Flow chart for operations.

III. COLOR TRANSFORM

Multiple methods of representing color data exist. Whilst RGB is most widely used for capture and display, it is not always the best for image processing, since it is a perceptually non-uniform representation. This means that if we change the RGB values by a fixed amount, the observed difference depends on the original RGB values. One way of observing this is to mix the output of standardized colored lights to generate a color, then alter the brightness of the input until an observer just notices a change in the light's color [5]. The original color and the color of the just noticeable difference can be plotted. By making measurements systematically over the whole color space, we can generate a MacAdam diagram. The points represent the original color, the ellipses the just noticeable difference contours. It is also possible to categorize color spaces as being device dependent or device independent. Device dependent spaces are used in the broadcast and printing industry [6], largely for convenience. The most widely used spaces are YIQ, YCr C and HSV. Conversion between the spaces is by using simple functions. E.g.

>> $YIQ = \text{rgb2ntsc}(RGB);$

Device independent spaces are used because the device dependent spaces include subjective definitions. The CIE defined a standardized color space in 1931. It specifies three color sources, called X, Y and Z. All visible colors can be generated by a linear combination of these. The X, Y and Z values can be normalized, to sum to 1. The color's represented by the normalized x and y values can be plotted – as in the MacAdam diagram. Conversion of data between color spaces is a two stage process. A color transformation structure is first defined, e.g. to convert from RGB to XYZ:

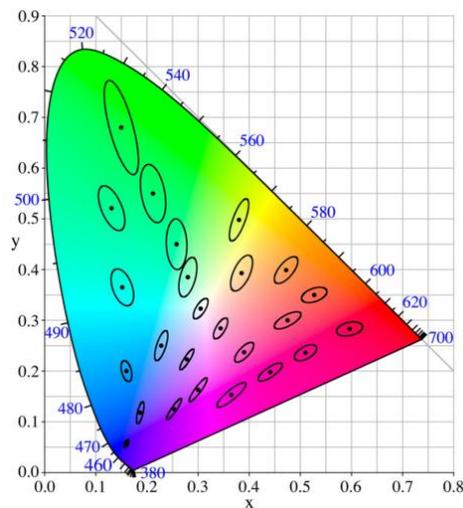


Fig 2: Color Transform

IV. FUNCTIONS

<code>imread</code>	Read image from graphics file
<code>imwrite</code>	Write image to graphics file
<code>imfinfo</code>	Information about graphics file
<code>nitfinfo</code>	Read metadata from National Imagery Transmission Format (NITF) file
<code>nitfread</code>	Read image from NITF file
<code>dpxinfo</code>	Read metadata from DPX file
<code>dpxread</code>	Read DPX image
<code>analyze75info</code>	Read metadata from header file of Analyze 7.5 data set
<code>analyze75read</code>	Read image data from image file of Analyze 7.5 data set
<code>interfileinfo</code>	Read metadata from Interfile file
<code>interfileread</code>	Read images in Interfile format
<code>hdrread</code>	Read high dynamic range (HDR) image
<code>hdrwrite</code>	Write Radiance high dynamic range (HDR) image file
<code>makehdr</code>	Create high dynamic range image
<code>tonemap</code>	Render high dynamic range image for viewing
<code>hdrread</code>	Read high dynamic range (HDR) image

V. IMAGE REPRESENTATION

There are five types of images in MATLAB.

1. **Grayscale.** A grayscale image M pixels tall and N pixels wide is represented as a matrix of double datatype of size $M \times N$. Element values (e.g., `MyImage(m,n)`) denote the pixel grayscale intensities in $[0,1]$ with 0=black and 1=white [7].
2. **Truecolor RGB.** A truecolor red-green-blue (RGB) image is represented as a three-dimensional $M \times N \times 3$ double matrix. Each pixel has red, green, blue components along the third dimension with values in $[0,1]$, for example, the color components of pixel (m,n) are `MyImage(m,n,1)` = red, `MyImage(m,n,2)` = green, `MyImage(m,n,3)` = blue.
3. **Indexed.** Indexed (paletted) images are represented with an index matrix of size $M \times N$ and a colormap matrix of size $K \times 3$. The colormap holds all colors used in the image and the index matrix represents the pixels by referring to colors in the colormap. For example, if the 22nd color is magenta `MyColormap(22,:) = [1,0,1]`, then `MyImage(m,n) = 22` is a magenta-colored pixel.
4. **Binary.** A binary image is represented by an $M \times N$ logical matrix where pixel values are 1 (true) or 0 (false).
5. **uint8.** This type uses less memory and some operations compute faster than with double types. For simplicity, this tutorial does not discuss uint8 further.

Grayscale is usually the preferred format for image processing. In cases requiring color, an RGB color image can be decomposed and handled as three separate grayscale images. Indexed images must be converted to grayscale or RGB for most operations.

VI. CODE

```

vid=videoinput('winvideo',1,'YUY2_640x480');
set(vid,'ReturnedColorSpace','rgb');
                                triggerconfig(vid,'manual');

%Capture one frame per trigger
set(vid,'FramesPerTrigger',1 );
set(vid,'TriggerRepeat', Inf);
start(vid); %start video
aa=1;
%Infinite while loop
Out=[];
while(1)
% preview(vid)
trigger(vid);
%Get Image
im=getdata(vid,1);
imshow(im);
hold on
if aa == 5
red=im(:,1);
Green=im(:,2);
Blue=im(:,3);
Out(:,1)=red;
Out(:,2)=Green;
Out(:,3)=Blue;
Out=uint8(Out);
end
if aa > 5
red=im(:,1);
Green=im(:,2);
Blue=im(:,3);
red1=Out(:,1);
Green1=Out(:,2);
Blue1=Out(:,3);
z1 = imabsdiff(red,red1); %get absolute difference between both images
z2 = imabsdiff(Green,Green1); %get absolute difference between both images
z3 = imabsdiff(Blue,Blue1); %get absolute difference between both images
zz1= sum(z1,1); %calculate SAD
zz2= sum(z2,1); %calculate SAD
zz3= sum(z3,1); %calculate SAD
zzz1= sum(zz1)/ 307200;
zzz2= sum(zz2)/ 307200;
zzz3= sum(zz3)/ 307200;
Final=(zzz1+zzz2+zzz3)/3
disp(Final);
end
aa=aa+1;
disp(aa);
if aa == 100
break
end
end
stop(vid),delete(vid),clear vid;

```

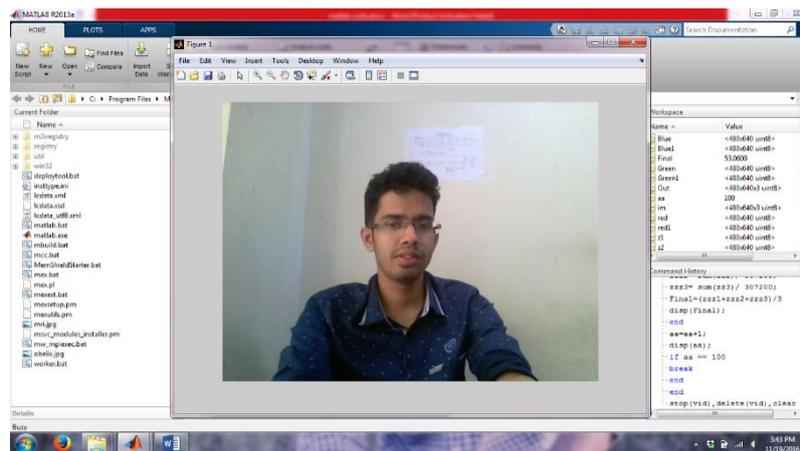


Fig 3: Sample

VII. CONCLUSION

This paper completes the description of the MATLAB-based applications for image processing and image quality assessment developed at MMTG, FEE, CTU in Prague started in the first part with the description of their function. In this part some concrete examples of use were shown and every application was practically presented. The future work could be aimed to extend the set of applications to cover even more areas of image and videoprocessing. Authors are currently working on development of new applications for quality assessment of videosequences and stereoscopic images. In case of interest about the test use of presented applications please contact the main authors at CTU in Prague.

REFERENCES

- [1] KRASULA, L., KLÍMA, M., ROGARD, E., JEANBLANC, E. MATLAB-based applications for image processing and image quality assessment – Part I: Software description. *Radio engineering*, 2011, vol. 20, no.4, p.1009 – 1015.
- [2] BREMOND, R., TAREL, J.-P., DUMONT, E., HAUTIERE, N. Vision models for image quality assessment: One is not enough. *Journal of Electronic Imaging*, October – December 2010, vol. 19, no. 4, p. 043004-1 to 043004-14.
- [3] SHEIKH, H. R., BOVIK, A. C., VECIANA, G. D. An information fidelity criterion for Image Quality Assessment using natural scene statistics. *IEEE Transactions on Image Processing*, December 2005, vol. 14, no. 12, p. 2117-2128.
- [4] SHEIKH, H. R., BOVIK, A. C. Image information and visual quality. *IEEE Transaction on Image Processing*, February 2006, vol. 15, no. 2. [Online]. Available at: <http://live.ece.utexas.edu/research/Quality/index.htm>
- [5] MeTriX MuX Visual Quality Assessment Package. [Online]. Available at: http://foulard.ece.cornell.edu/gaubatz/metrix_mux/
- [6] SPRLJAN, N. MATLAB Quality Assessment Toolbox. [Online]. Available at: <http://www.sprljan.com/nikola/matlab.html>
- [7] MURTHY, A. V., KARAM, L. J. A MATLAB-based framework for image and video quality evaluation. In *Second International Workshop on Quality of Multimedia Experience (QoMEX)*. Trondheim (Norway), 2010, p. 242.