Volume 5, Issue 2, February 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Image Processing on DSP Environment Using OpenCV

Sharmila B*, Karalan N, Nedumaran D
Central Instrumentation and Service Laboratory,
University of Madras, Chennai, India

Abstract— This paper presents image conversion, including C++ implementation, memory constraints, and floating-point support to integrate in the accelerators or co-processors environment. A new function of converting an image file to data file (.dat) using OpenCV to optimize image processing systems on the TMS320C6713 DSK digital signal processor (DSP) has been developed. The image of all formats and all sizes can be converted efficiently to suite the end user requirements. The function so created supports both 8-bit (uint8) and 16-bit (uint16) raw pixel depth formats for uploading the image data in the DSP environment. The converted image is then loaded to the DSP and a contrast stretching filter is applied to enhance the contrast of the image. The results obtained are studied for possible implementation in real-time applications.

Keywords— DSP, OpenCV, Contrast Stretching, Image Conversion

I. INTRODUCTION

Image processing [1] is a multi-disciplinary subject, which has potential applications in many areas of science, engineering and technology, viz., biomedical, space, military, vision, biometrics, etc. Image processing helps to enhance an image or to extract hidden information from an image. In this paper, image conversion including C++ implementation, memory constraints and floating-point support to integrate with different embedded platforms like DSP, was carried out. In embedded system image processing [2], the input image, such as video frame or photograph, gives an output which can be either an image or one of the associated characteristics of that image. The image-processing techniques consider the image as a two-dimensional signal and apply some of the standard signal-processing methods on it. Most of the advanced embedded platforms process the images in hex file format. In embedded environment, the system's limited power and size must be used efficiently to achieve real-time processing [3]. Real-time systems, based on DSPs for enhanced full-featured image recognition system without any requirement of additional sensor, have been developed efficiently [4].

At present, many dedicated software like MATLAB, Vivado, OpenCV, etc., are available for embedded system programming. Among these languages, OpenCV (Open Computer Vision) [5, 6], a well documented and vibrant open source C++ library growing with new computing technology applications, was designed and originally developed by Intel and now supported by Willow Garage, has a huge computational efficiency with a strong focus on real-time applications. It is a free source library for both commercial and non-commercial uses, which provides a simple-to-use computer vision infrastructure that helps build fairly sophisticated vision and real time image processing systems. OpenCV is written in optimized C and can take advantage of multi-core processors [7]. Implementation of such useful library on an embedded processor like DSP makes it possible to achieve real-time image processing without any bottlenecks in the software part.

Our paper is organized as follows. In Section 2, some of the work already carried out in this field is presented. Section 3 explains the steps and method used for conversion of an image to .dat file with OpenCV interface. Section 4 gives a detailed view about the contrast stretching using a linear contrast stretching transformation. In Section 5, the implementation, results, and their performance evaluation were described. Section 6 deals with results and discussions.

II. RELATED WORK

The current I/O routines for basic image conversion which works with 8-bit gray scale BMP and TIFF images under C were discussed in [8]. Miano illustrates the complete instruction set to read/ write various 2D image formats [9] for the coding process. In MATLAB, the '*imread*' [10] function is used for importing the graphics data file into the workspace. In this work, conversion of an image into a data file was carried out using *endiness architecture*. The memory storage was handled by two different types of endiness, namely Big endian and Little endian [11, 12]. The types of endianess are differentiated by the order in which the converted sequential bytes are stored in the memory [13]. Considering OpenCV, it is an vibrant open source environment which can be adopted with different API's based on the user's requirements and also allows the usage of many new API for different hardware modules like GPU, Microsoft Kinect Fusion, Mobile devices, DSP, FPGA, etc., for real-time parallel processing of high resolution 4-channel imaging [14]. Throne and Grasset [15] carried out a comparative study between Numpy/scipy library, OpenCV phython wrapper library and native C implemented OpenCV library and evaluated them based on their performance and usability. Xie and Lu [16] explain most of the OpenCV image processing tools available with an elaborate theory of '*IplImage Data Structure*'. Pant et al.

carried out an encryption of an image using a method that combines existing Arnold transformation [17] with a defined pointer to transverse image data. The OpenCV platform allows transformation of a pixel in an image from one format to the other using CUDA on GPU in a concurrent method rather than a sequential method [18]. Recently, real-time processing of images is required for embedded applications such as automotive applications, robotics, entertainment, etc. The embedded image acquisition systems [19, 20] have advantages of small volume and high stability. The strong signal processing ability of DSP systems gave a solid foundation for processing image information in real-time. To realize real-time processing of image recognition on such systems, we need optimized libraries for embedded processors. Some of the basic theories of digital image processing were discussed [21-24]. Occurrence of low quality digital images due to low lightning condition is still a common problem. These low quality images can be enhanced effectively using a contrast stretching algorithm. Implementing such an algorithm in an embedded processor like beagle board was carried out for giving a high quality image from a low quality image [25].

Implementation of such vibrant open source library on an embedded processor like DSP makes it possible to achieve real-time image recognition. To make use of the library functions of OpenCV, the libraries (.dll files) of OpenCV must be configured first into any compiler. In this work, it was done into the Microsoft Visual Studio Professional 2010.

III. DATA CONVERSION APPLICATION IN DSK

To load an image (which is to be processed) into the DSP kit, first we have to convert the image into the hex (.dat) format as the DSP processors in particular the Texas Instrument based TMS320C6713 DSK accepts an image in hex (.dat) format only.

A. Steps in Data Conversion

The function SAVE_CCS_DATA_FILE was created for image conversion. It supports both 8-bit (uint8) and 16-bit (uint16) raw pixel depth formats for uploading the image data in the DSP environment. The OpenCV library function *highgui* is mainly used for this conversion. The following steps were followed for the conversion of the image:

***Step* 1**: The image to be converted is loaded into the workspace in a given name and format. If the file was unable to open, then the error message, *Failed to open the file,* will be displayed.

Step* 2*: In the next step the raw image was resized according to the requirement. If the given values are 0 and 0, the image will not be resized.

Step* 3*: Then the program checks the loaded image for its pixel depth. If the image loaded is not of 8-bit or 16-bit, then the program displays the error message, *Only 8-bit and 16-bit images are supported*.

Step* 4*: In the next step, the program writes the encoding format of the file (1651 1 0 0 0). The above encoding format shows that the data are in little endian format.

Step* 5*: According to the above mentioned format, the program starts converting an image into hexadecimal numbers. Then the converted hexadecimal image was stored in a specified location.

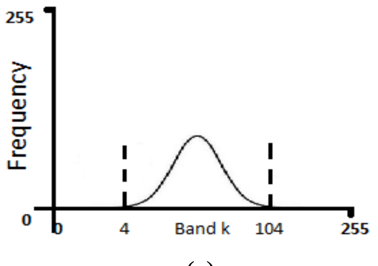
In the final output, each line of the converted image contains a single word (32 bits) of data. If it is 8-bit pixel depth, each line in the output text file contains four pixels worth of data and the 16-bit pixel depth data has each line in the output text file with two pixels worth of data.

IV. CONTRAST STRETCHING

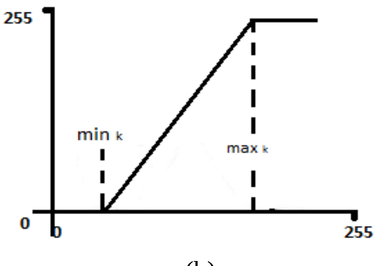
Contrast is the difference in visual properties which make an object (or its representation in an image) distinguishable from other objects and the background. In visual perception of the real world, contrast is determined by the difference in the colour and brightness of the object and the other objects within the same field of view. Contrast generally refers to the difference in luminance or grey level values in an image and is an important characteristic. It can be defined as the ratio of the maximum intensity to the minimum intensity over an image [1]. Contrast stretching is a common technique used for better visualization of an image by evenly distributing the intensity histogram values. It is mostly used in fields like computer vision, medical imaging, automation using visual processing, etc. The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a *specific* application. The word specific is important, because it establishes at the outset that the techniques are oriented to the problem. Thus, for example, a method that is quite useful for enhancing X-ray images may not necessarily be the best approach for enhancing pictures of Mars transmitted by a space probe. Regardless of the method used, however, image enhancement [25] is one of the most interesting and visually appealing areas of image processing. Contrast ratio [1, 26] has a strong bearing on the resolving power and detectability of an image. Larger this ratio, more easy it is to interpret the image. For images with low contrast in greyscale, the contrast stretching is done along with histogram equalization [28-30].

A. Linear Contrast Stretching

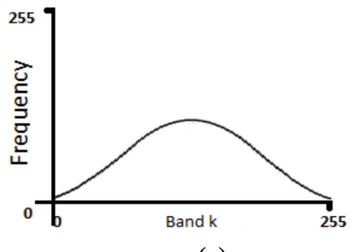
This method has the simplest contrast stretching algorithm. A linear relationship is followed between the grey values in the original and modified images [27]. The raw image is first converted into grey scale and the histogram of the image is found. From the histogram, a density number in the low range of the original histogram is assigned to extremely black and a value at the high end is assigned to extremely white, while the remaining pixel values will be distributed linearly between these extremes. Fig. 1 shows the graphical representation of linear contrast stretching operation [31]. Optimal contrast and colour variation were provided by the small range grey values in each band which was stretched to the full brightness range of the output image.



(a)



(b)



(c)

Fig. 1 (a) Frequency of Histogram of low contrast image, (b) Linear contrast stretch transformation, (c) Frequency of Histogram of min-max linear contrast stretched image sample line graph using colours which contrast well.

B. Non-Linear Contrast Stretching

In this type, a non-linear transformation [32] is followed by the input and output values of contrast stretching. The general form of the non-linear contrast enhancement is defined by $y = f(x)$, where x is the input and y is the output. Non-linear contrast enhancement techniques were used for enhancing the colour contrast between the subclasses and nearly classes of a main class. A type of non linear contrast stretch involves scaling the input data logarithmically. This enhancement has the greatest impact on the brightness values found in the darker part of the histogram. It could be reversed to enhance values in the brighter part of histogram by scaling the input data using an inverse log function. Histogram equalization [33, 34] is another non-linear contrast enhancement technique. In this technique, the histogram of the original image is redistributed to produce a uniform population density. This is obtained by grouping certain adjacent grey values. Thus, the number of grey levels in the enhanced image is less than that of the grey levels in the original image. A major disadvantage of non-linear contrast stretching is that each value of the input image has more than one output value, hence making the objects in accordance with the original image, there is a lose in correct relative brightness value.

C. Algorithm for Contrast Stretching

***Step* 1**. A histogram for loaded hex format grey scale image was taken and stored in an array.
***Step* 2**. For each pixel value, the histogram or PDF (probability density function) is computed.
***Step* 3**. In the next step CDF (Cumulative Distribution Function) is taken and stored in an array.
***Step* 4**. Two bins corresponding to lower percentile and upper percentile were calculated.
***Step* 5**. Finally, a linear scaling function is applied with respect to the lower and upper percentile values.

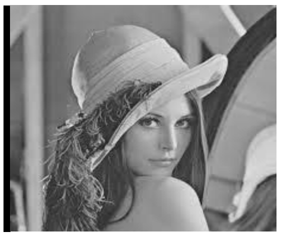
V. IMPLEMENTATION

The CCStudio file format dictates that each line contains a single word (32 bits) of data. So, to use the save_ccs_data_file to generate a data file (.dat), the string *uint8* should be passed in as the second argument. Since each word consists of four bytes (32 bits), each line in the output text file contains four pixels worth of data.

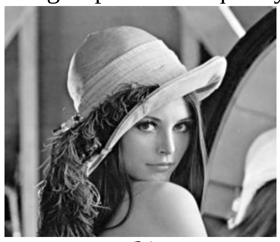
The function assumes little-endian ordering, i.e., bytes are numbered from right to left within a word, with the most significant bytes in a word having a larger address. The C6713 architecture is unique in that it is bi-endian, meaning that depending on the build options it can be either little endian or big endian. In order to run the save_ccs_data_file correctly, the build options for the project must be set to little endian (which is the default setting). The function save_ccs_data_file was created using OpenCV 2.4.7 which was configured with Visual Studio 2010 Professional. The converted image (.dat) is then loaded to a Texas Instrument based TMS320C6713 DSK DSP Kit using the Code Compresser Studio 507849-6001A.

VI. RESULTS AND DISCUSSIONS

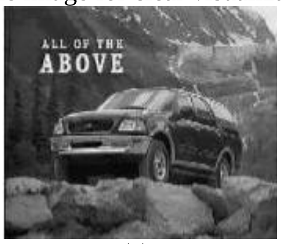
In this work, we implemented the save_ccs_data_file function and tested it on various images for data conversion and contrast stretching. The snap shot of the input raw image and its corresponding output image are shown in Fig. 2. The function developed in OpenCV provides a tool for easily experiencing the benefits of DSP application in image processing. In the case of the MATLAB, the code is lengthy and consumes more time for the same conversion. The total time taken for the conversion in OpenCV falls less than 3 s. Further, the data conversion and contrast stretching implemented in the TMS320C6713 DSP Starter Kit clearly indicated that the data conversion reduces the time of conversion and the contrast stretching improves the quality of the image for clear visualization of the input image.



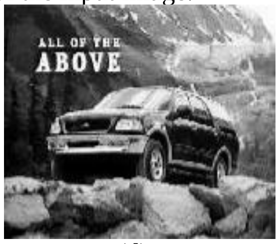
(a)



(b)



(c)



(d)

Fig. 2 Raw images (a, c) and their contrast stretched images (b, d).

In MATLAB, the resize of an image must be done before the conversion. But in our program, we can define the output size of an image in the same function. The program so developed can be directly ported in the CCS environment, whereas in MATLAB the same is done using SimuLink and then the code is ported into CCS, which will reduce the compilation time of the application program.

We have attempted to mitigate image conversion, to support integrated accelerators or co-processors. Since OpenCV is the open source free software for academic and commercial use, the program developed in this software environment supports various imaging applications very efficiently at no cost. Images of all formats and all sizes were tested and it was found that the program converted the image file formats efficiently in less than 3 s. The created function supports both 8-bit (uint8) and 16-bit (uint16) raw pixel depth formats for uploading the image data in the DSP environment.

A function generating .cmd and .pjt files in OpenCV will be developed to perform all these operations automatically without user intervention in CCS.

ACKNOWLEDGMENT

The authors are grateful to the Department of Science and Technology for the award of a DST- INSPIRE Fellowship to carry out this research work.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed., Prentice Hall, New Jersey, 2002, pp. 1-70.

[2] R. Kamal, *Embedded systems*. Boston: McGraw-Hill Higher Education, 2008, pp. 1-54.

[3] M. Humenberger, C. Zinner, M. Weber, W. Kubinger and M. Vincze, “A fast stereo matching algorithm suitable for embedded real-time systems”, *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1180-1202, 2010.

[4] A. Clemens, L. Florian and B. Horst, “Real-Time License Plate Recognition on an Embedded DSP-Platform”, in *Computer Vision and Pattern Recognition, 2007, CVPR “07,* Minneapolis, MN, 2007, pp. 1-8.

[5] G. Bradski and A. Kaehler, *Learning OpenCV,* Sebastopol, CA: O”Reilly, 2008, pp. 1-87.

[6] Opencv.org, “DOCUMENTATION | OpenCV”, 2015. [Online]. Available at: http://opencv.org/documentation.html. [Accessed: 17- Feb- 2015].

[7] G. Kornaros, *Multi-core embedded systems,* Boca Raton, FL: CRC Press/Taylor & Francis, 2010, pp. 1-63.

[8] D. Phillips, *Image processing in C,* Lawrence, Kans.: R & D Publications, 1994, pp. 7-25, 33-45.

[9] J. Miano, *Compressed image file formats,* Reading, Mass.: Addison Wesley, 1999, pp. 1-35.

[10] *MATLABÃ‚Â® Data Import and Export,* MATLAB Inc., 2014, pp. (1-1) – (1-30), (5-1) – (5-22).

[11] Support.microsoft.com, “Explanation of Big Endian and Little Endian Architecture”, 2006. [Online]. Available: http://support.microsoft.com/kb/102025.

[12] M. Arora, *The art of hardware architecture,* New York: Springer Science+Business Media, LLC, 2012, pp. 155-161.

[13] “Endianness and ARMÃ‚Â® Processors”, *Application notes by Asset*, pp. 1-6, 2013.

[14] K. Pulli, A. Baksheev, K. Kornyakov and V. Eruhimov, “Real-time computer vision with OpenCV”, *Commun. ACM*, vol. 55, no. 6, pp. 61-69, 2012.

[15] T. Brain and G. Raphael, “Python for Prototyping Computer Vision Applications”, in *NZCSRSC 2010*, Wellington, New Zealand, 2010, pp. 12-15.

[16] G. Xie and W. Lu, “Image Edge Detection Based On Opencv”, *IJEEE*, vol. 1, no. 2, pp. 104-106, 2013.

[17] A. Pant, A. Arora, S. Kumar and R. Arora, “Sophisticated Image Encryption Using OpenCV”, *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 1, 2012.

[18] E. Olmedo, J. Calleja, A. Benitez and M. Medina, “Point to point processing of digital images using parallel computing”, *International Journal of Computer Science Issues*, vol. 9, no. 3, pp. 1-10, 2012.

[19] Z. Jianjun and Z. Jianhong, “Research on embedded digital image recognition system based on ARM-DSP”, in *Computer Science and Information Technology, 2009. ICCSIT 2009,* Beijing, 2009, pp. 524 - 527.

[20] H. Jiang and X. Su, “The Embedded Image Acquisition System Based on the ARM”, *JCIT*, vol. 8, no. 9, pp. 845-852, 2013.

[21] S. Brahmbhatt, *Practical OpenCV,* Berkeley, California: Apress, 2013, pp. 1-41.

[22] A. Chavan and S. K. Yogamani, “Real-time DSP implementation of Pedestrian Detection algorithm using HOG feature”, in *ITS Telecommunications (ITST), 2012,* Taipei, 2012, pp. 352 - 355.

[23] P. K. Aby, A. Jose, L. D. Dinu and J. John, “Implementation and optimization of embedded Face Detection system”, in *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN),* Thuckafay, 2011, pp. 250-253.

[24] P. Poudel and M. Shirvaikar , “Optimization of computer vision algorithms for real time platforms”, in *System Theory (SSST), 2010 42nd Southeastern Symposium*, Tyler, TX, 2010, pp. 51-55.

[25] H. Reddy and S. K, “Color Image Enhancement for Contrast Stretching by Using Begale Board”, *International Journal of VLSI and Embedded Systems-IJVES,* vol. 05, no. 03238, pp. 769-772, 2014.

[26] E. Peli, “Contrast in complex images”, *J. Opt. Soc. Am. A*, vol. 7, no. 10, pp. 2032-2040, 1990.

[27] Patel. A. N. and S. Singh, *Remote Sensing Principles and Applications*, 2nd ed. Jodhpur: Scientific Publishers, 1999, pp. 70-73.

[28] Yeong-Taeg Kim, “Contrast enhancement using brightness preserving bi-histogram equalization”, *IEEE Transactions on Consumer Electronics*, vol. 43, no. 1, pp. 1-8, 1997.
[29] J. Shin and R. Park, “Histogram-Based Locality-Preserving Contrast Enhancement”, *IEEE Signal Process. Lett.*, pp. 1-1, 2015.
[30] S. Gupta and Y. Kaur, “Review of Different Local and Global Contrast Enhancement Techniques for a Digital Image”, *International Journal of Computer Applications*, vol. 100, no. 18, pp. 18-23, 2014.
[31] T. Lillesand and R. Kiefer, *Remote sensing and image interpretation*, New York: Wiley, 1979.
[32] S. Saleh Al-amri, K. N. V. and K. S. D., “Linear and Non-linear Contrast Enhancement Image”, *International Journal of Computer Science and Network Security*, vol. 10, no. 02, pp. 139-143, 2010.
[33] A. Raju, G. S. Dwarakish and D. V. Reddy, “A Comparative Analysis of Histogram Equalization based Techniques for Contrast Enhancement and Brightness Preserving”, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 6, no. 5, pp. 353-366, 2013.
[34] R. Garg, B. Mittal and S. Garg, “Histogram Equalization Techniques for Image Enhancement”, *International Journal of Electronics & Communication Technology*, vol. 02, no. 01, pp. 107-111, 2011.