Realization of Different Algorithms Using Raspberry Pi for Real-Time Image Processing Application

Mrutyunjaya Sahani and Mihir Narayan Mohanty

Abstract As automated system is an efficient one which decreases the malfunctions and error, industries having more demand to adopt such systems day by day. To enhance this capability, image processing-based system has a major role. But to design an embedded system with the facilities, it is difficult and challengeable task. Another issue for the use interfacing of such systems should be simple so that the system can be user friendly. To develop this system with various facilities, this piece of work has been attempted. Hence in this paper, authors showcase the attempt to realize the algorithms based on image processing for different applications. The work is developed in Raspberry pi development board with python 2.7.3 and OpenCV 2.3.1 platform. The result shows for the fully automated without any user intervention. The prime focus is on python image library (PIL) in addition to numpy, scipy, and matplotlib form a powerful platform for scientific computing.

Keywords ARM1176JZF-S · Image processing · Raspberry pi · Python image library · Raspbian-Wheezy

1 Introduction

Real-time image processing techniques are important not only in terms of improving productivity, but also in reducing operator errors associated with visual feedback delay. For implementing the different image processing algorithm, we have used the Raspberry Pi (shown in Fig. 1) which is a single-board computer developed by Cambridge University. The Pi has been extremely popular among

M. Sahani (✉) · M.N. Mohanty
ITER, Siksha 'O' Anusandhan University, Odisha, India
e-mail: mrutyunjayasahani@soauniversity.ac.in

M.N. Mohanty
e-mail: mihirmohanty@soauniversity.ac.in

L.C. Jain et al. (eds.), *Intelligent Computing, Communication and Devices*,
Advances in Intelligent Systems and Computing 309,
DOI 10.1007/978-81-322-2009-1_53

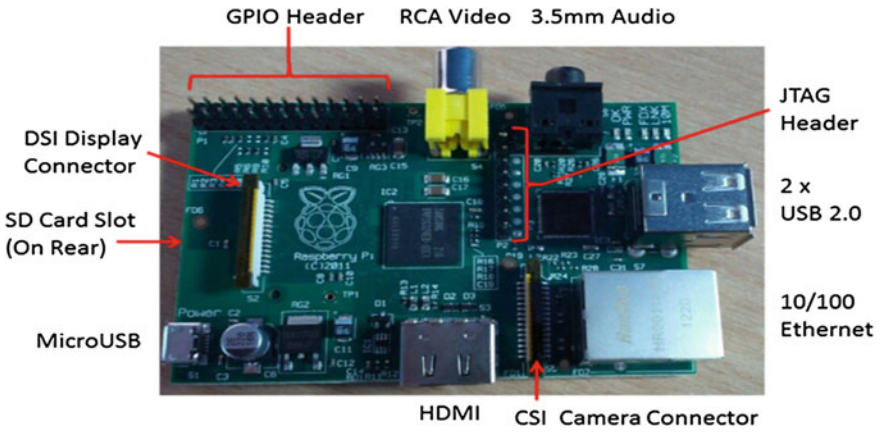

Fig. 1 Raspberry pi board (model B)

the academic fraternity due to its low cost. The model B of the Pi ships with 512 Mb of RAM, 2 USB ports, and an Ethernet port. It packs an ARM1176JZF-S 700 MHz processor, Video Core IV GPU into the Broadcom BCM2835 System on Chip which is cheap, powerful, and also low on power. The Pi has HDMI support and has an SD card slot for booting up due to lack of BIOS and a persistent memory [1]. Python is one of the primary languages supported by the Pi; hence, we have used the python image library (PIL) which adds powerful image processing functions to the Python interpreter. There are many different operating systems supported by the Raspberry Pi. We have chosen Raspbian-Wheezy which can be downloaded from the University of Cambridge website. Raspbian-Wheezy is a freely available version of Debian Linux which has been customized to run on the Pi. It has been designed to make use of the Pi's floating point hardware architecture, thus enhancing performance. The program designed by me is completely automatic, and there is no need for user intervention. The program displays the processed image on the screen for comparison.

This chapter is organized as follows. Section 1 introduces the work, and Sect. 2 deals with the system configuration. Section 3 proposes the implementation method that follows the result. Finally, Sect. 4 concludes this piece of contribution.

2 System Configuration

The basic hardware requirements are a Raspberry Pi board (model B), a USB hub, a USB keyboard, and mouse. An HDMI-enabled display device is also required along with a micro-USB charger to power the board. An SD card is required to boot the Pi.

2.1 Preparation of SD Card for Raspberry Pi

Raspberry Pi lacks a very important feature, i.e., BIOS. So the Pi always needs a SD card loaded with a snapshot of an operating system to boot up. After downloading the image, we must prepare a snapshot of the image on the SD card for the Pi to boot [2].

2.2 Network Configuration

Network configuration should be done for the Pi to enable installation of different packages, dependencies and to regularly update and upgrade the operating system.

2.3 List of Packages

Following packages are to be installed for implementing the proposed model. Installation commands have been listed below.

(a) sudo apt-get install camorama
(b) sudo apt-get install python-dev
(c) sudo apt-get install libjpeg62-dev libpng12-dev
(d) sudo apt-get install python-matplotlib
(e) sudo apt-get install python-numpy python-scipy
(f) sudo apt-get install python-imaging python-tk

2.4 OpenCV Installation

OpenCV is used to implement different algorithms in the python environment to make the system automatic [3]. Before final installation of OpenCV, following dependencies have to be installed.

1. Menu → Accessories → LX Terminal → this opens a terminal window in RPi graphical interface. Then, we use following commands.

   (a) sudo apt-get install build-essential
   (b) sudo apt-get install cmake
   (c) sudo apt-get install pkg-config
   (d) sudo apt-get install libpng12-0 libpng12-dev libpng++dev libpng3
   (e) sudo apt-get install zlib1g-dbg zlib1g zlib1g-dev
   (f) sudo apt-get install libpnglite-dev libpngwriter0-dev libpngwriter0c2

(g) sudo apt-get install pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools
(h) sudo apt-get install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs
(i) sudo apt-get install ffmpeg libavcodec-dev libavcodec52 libavformat52 libavformat-dev
(j) sudo apt-get install libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev
(k) sudo apt-get install libxine1-ffmpeg libxine-dev libxine1-bin
(l) sudo apt-get install libunicap2 libunicap2-dev
(m) sudo apt-get install libdc1394-22-dev libdc1394-22 libdc1394-utils
(n) sudo apt-get install libv4l-0 libv4l-dev
(o) sudo apt-get install python-numpy
(p) sudo apt-get install libpython2.7 python-dev python2.7-dev
(q) sudo apt-get install libgtk2.0-dev pkg-config

2. OpenCV was downloaded from the Internet using the following command. Wgethttp://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.3.1/OpenCV2.3.1a.tar.bz2/download.
3. Then, we make, install, and configure the OpenCV installed by using following commands.

(a) cmake -D CMAKE_BUILD_TYPE=RELEASE –D MAKE_INSTALL_PREFIX = "/usr/local/lib"–D BUILD_NEW_PYTHON_SUPPORT=ON –D BUILD_ EXAMPLES = ON
(b) sudo make
(c) sudo make install

Now OpenCV-2.3.1 is installed and ready to run on the Raspberry Pi. The installation takes a very long time to complete.

3 Proposed Method for System Realization

3.1 Geometrical Transformation

Geometrical transformation refers to modification of the spatial relationship of the pixels in an image [4]. Some or all of the pixels in the original image are mapped to a new coordinate in the modified image. Mapping is usually of two types, forward mapping and inverse mapping. Inverse mapping is more efficient than forward mapping and produces better results. Affine transforms are one of the most commonly used transforms and can be used to achieve cropping, flipping, and rotation as shown in Fig. 2. Affine transforms can be achieved by applying the following technique:

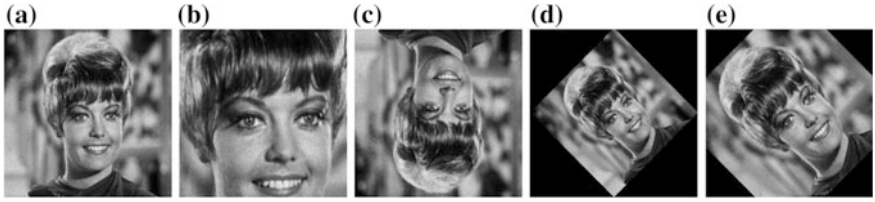

Fig. 2 **a** Original image. **b** Magnified image. **c** Inverted image. **d** Rotation with resizing. **e** Rotation without resizing

$$\begin{bmatrix} x' \\ y\prime \\ 1 \end{bmatrix} = \begin{bmatrix} m1 & m2 & m3 \\ n1 & n2 & n3 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Here, $x'$and y' are the modified pixels locations, and x and y are the original image locations. By altering the values of the constants in the middle matrix, the image can be rotated, flipped, and cropped [5]. If we wish to perform multiple operations, then the resultant matrix is the product of all the required matrices. For determining the intensity levels, we use interpolation techniques. Usually, bilinear is preferred over the nearest neighbor approach of interpolation.

3.2 Smoothing and Gray-level Slicing

Smoothing operation is usually applied for removal of noise and blurring. Smoothing tends to replace the pixel values with the average of the values in the neighborhood of that pixel. They are also known as low-pass filters or averaging filters. Random noise usually consists of sharp changes in intensity levels, so this filter is a great tool for noise reduction, but since edges are also represented by sharp changes in intensity values so they have the undesirable effect of blurring the edges.

For image enhancement, some applications may need us to focus on certain intensity levels instead of the entire range to highlight the region of interest. There are usually two approaches in this case. In the first approach, the intensity levels within certain desired range are enhanced as required and the remaining intensity levels are suppressed. In the second approach, the intensity levels within the desired range are enhanced and remaining intensity levels are left as it is shown in Fig. 3.

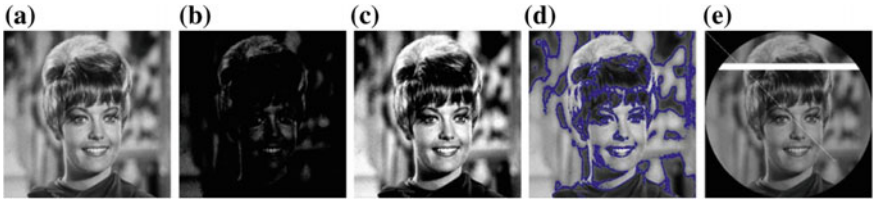

Fig. 3 **a** Original image. **b** Darkened image. **c** Brightened image. **d** Contour plot. **e** Sliced image

3.3 Sharpening and Denoising

Sharpening is done to enhance details and edges in an image. Among the various options available, we will restrict ourselves to sharpening by Laplacian operator. The matrix A represents the Laplacian mask considering only the horizontal and vertical directions. The matrix B represents the Laplacian mask considering horizontal, vertical as well as diagonal directions.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In case of median filter, we take a mask of the desired size and apply it to each pixel in the image. On applying the mask, we replace the original intensity value by the median (by arranging in ascending order and selecting the middle value) of the neighborhood intensity values. For denoising applications, median filter has greater advantage over box filter or Gaussian filter since it tends to retain the sharpness and contrast of the image after processing it which is shown in Fig. 4. Median filter is particularly very effective in case of salt and pepper noise.

3.4 Edge Detection

Industry is using edge detection technique to automate the process for increasing efficiency by eliminating defects or malfunctioning of any system. An edge may be

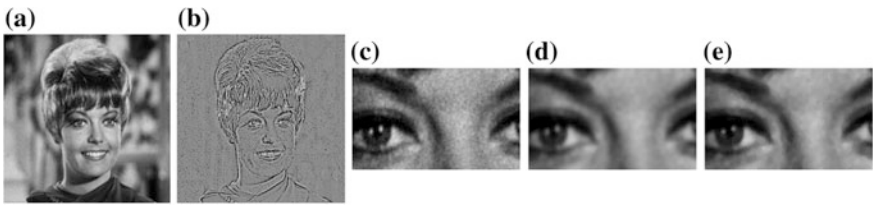

Fig. 4 **a** Original image. **b** Sharpened image. **c** Noisy image. **d** Gaussian filter. **e** Median filter

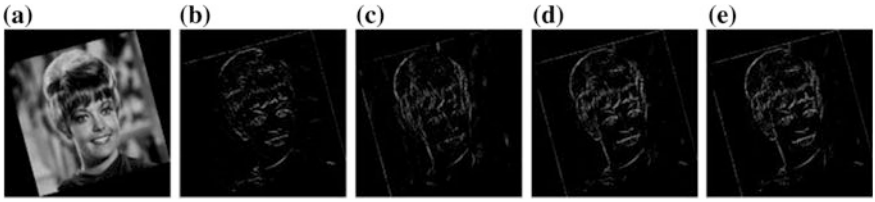

Fig. 5 **a** Original image. **b** Sobel along *x*. **c** Sobel along *y*. **d** Sobel. **e** Canny

defined as a high-frequency component in an image $f(x, y)$ or a set of continuous pixels in an image in which there is a considerable change in some physical aspect of the image. Such discontinuities in intensity values of an image are detected by using first- and second-order derivatives as shown in Fig. 5. There is several edge detection algorithms [2] implemented in digital image processing like Sobel, Prewitt, Roberts, and Canny [6], but all of them can be categorized into two categories, gradient and Laplacian.

4 Conclusion

Matplotlib, Numpy, Scipy, PIL, and OpenCV were installed successfully on Raspberry Pi development board. It was found that the algorithm developed for the Raspberry Pi executes successfully for different image processing applications and gives very colorful images. Therefore, we conclude that the Raspberry Pi module can easily replace with a host processor for any kind of real-time image processing applications.

References

1. Schmidt, M.: Pragmatic Raspberry Pi, 1st edn. The Pragmatic Programmers, LLC., USA (2012)
2. Tank, J.K., Patel, V.: Edge detection using different algorithms in raspberry pi. Int. J. Sci. Res. Dev. **1**(4), 984–986 (2013)
3. Solem, J.F.: Programming Computer Vision with Python, 1st edn. O'Reilly Media Inc., USA (2012)
4. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Pearson Prentice Hall, New Jersy (2008)
5. Vlasic, D.: Geometric image transformation. Internet: http://groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture07/Slide01.html. 30 Oct 2013
6. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-8**(6), 679–698 (1986)