https://cejsr.academicjournal.io

Estimation of Pulmonary Signal Using a Digital Camera

Zainab Abd Alhussain Muhammd
Department of Medical Instrumentation Technical Engineering,
Ministry of Higher Education and Scientific Research, Middle Technical University

ABSTRACT

This work aims to facilitate detection of IC information, including serial number and company by reading the IC printed numbers from the manufacture using an image processing program. This project proposes a low-cost system for extracting IC information using a digital camera. The experiments were conducted on a DIP ICs at a different distance of 2 m and MATLAB® system were then compared with the data sheet of it. The experimental results show a promising performance in comparison with the data sheet from the manufacture, with low error rate.

Introduction

Most of the reviewed remote measurement technologies described in the previous chapter are promising for extracting physiological signs within the respiratory signal from several ROIs. However, each remote measurement technology has advantages and disadvantages under different assumptions, leading to a number of challenges to be considered. These challenges are summarized as the issues associated with an increased number of applied ROIs, possible biological effects, the cost of the proposed system, noise artefact removal and the detection range. Computer system–based camera imaging has high potential for being a robust and feasible research design to extract physiological signs using different image and signal processing techniques, in which the above challenges are main considerations.

This chapter outlines the methodology and techniques used in this project to address the research questions by providing descriptions of the study's design, data collection procedures, data processing and analysis, software used and statistical methods applied.

Research Design

A key task of the research presented in this thesis was to formulate image and signal processing techniques for extracting and measuring physiological signs from the physiological and physical variations caused by respiratory activity, using a computer system based on camera imaging. Figure 3-1 displays the main block diagram that needed to be achieved for a robust remote measurement system.

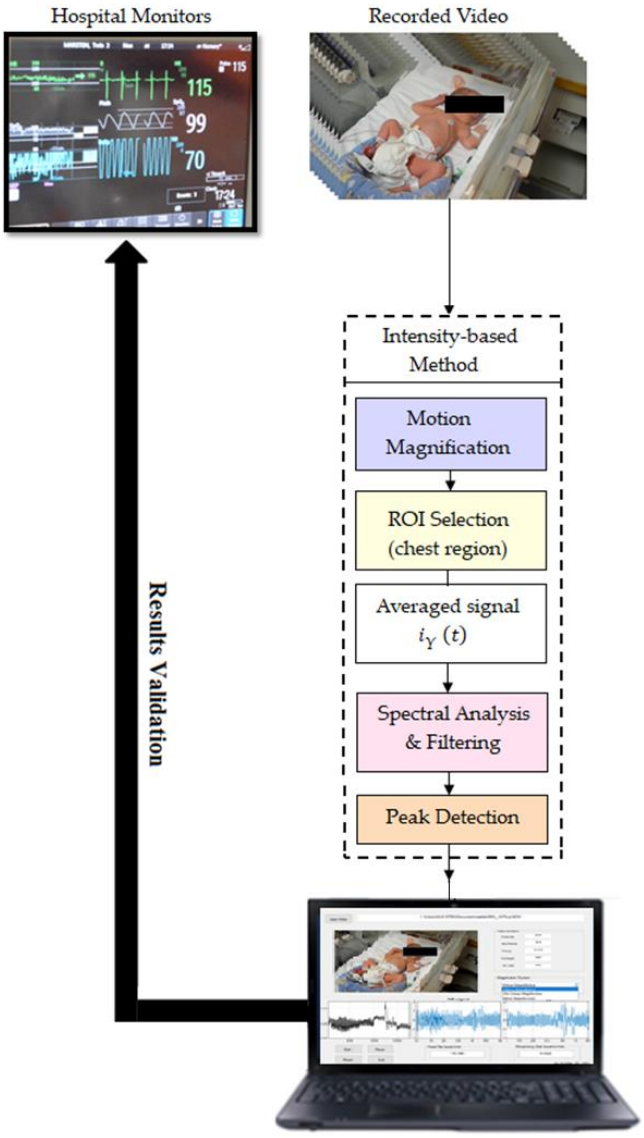


Figure 3-1: Schematic diagram of the proposed monitoring system.

Data Collection

Research Ethics

The research procedure described in this project adhered to the ethical tenets of the Declaration of Helsinki (Finland 1964). The protocol was approved by local ethics committees and a written informed consent form for all infant was obtained from their parents after a full explanation of the research procedures before commencing the experiment.

Participants

A group of 5 subjects (males and females, <1 year of age) with different ages and different skin tones participated in the research experiment.

Image Data Acquisition

To take a clear picture, we use JINGOU portable USB microscope which consists of Image CMOS Sensor and Controller High Speed DSP.

Still Image Capture Resolution Standard 640 * 480, Max 1600 * 1200 and Frame Rate 30 f / s under

600 LUX Brightness and Magnification Range 100X-200X, 50X-400X, 50X-500X, 50X-600X, 800X, 1000X.

Built-in 8 White-light LED and adjustable illumination ensure the magnified images are clear and bright and Brightness Control Manual adjustment.

JINGOU portable USB microscope

Validation

For validation purposes, the ECG monitor (Phillips IntelliVue ECG monitor) in the hospital was used to compare with the results with those obtained by the proposed monitoring system.

Data Processing and Analysis

Image Processing Techniques

Various image and video processing techniques have been applied in this project to analyze and extract useful features of interest from image data. These techniques included a description of color space conversion procedures, image quality assessment, image magnification, face detection and feature extraction, and frame subtraction.

Image Processing

Image Processing Toolbox™ provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing.

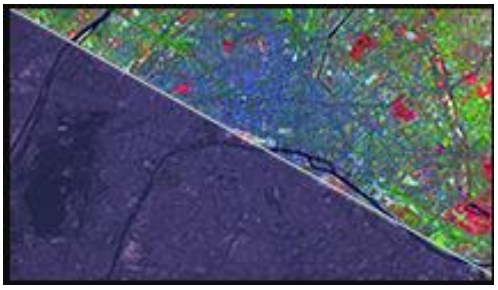
Image Processing Toolbox apps let you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large data sets. Visualization functions and apps let you explore images, 3D volumes, and videos; adjust contrast; create histograms; and manipulate regions of interest (ROIs).

Image Preprocessing

Increase the signal-to-noise ratio and accentuate image features using custom or predefined filters.

Image Enhancement

Increase the signal-to-noise ratio and accentuate image features by modifying the colors or intensities of an image. Perform convolution and correlation, remove noise, adjust contrast, and remap the dynamic range.



Enhancing Multispectral Color Composite Images

Image Deblurring

Correct blurring caused by out-of-focus optics, movement by the camera or the subject during image capture, atmospheric conditions, short exposure time, and other factors.

Deblurring Images Using the Blind Deconvolution Algorithm

Image Analysis

Extract meaningful information from images, such as finding shapes, counting objects, identifying colors, or measuring object properties.

Edge Detection

Identify object boundaries in an image using pre-built algorithms. These algorithms include the Sobel, Prewitt, Roberts, Canny, and Laplacian of Gaussian methods.

Image Region Analysis

Calculate the properties of regions in images, such as area, centroid, and orientation. Use the Image Region Analysis App to automatically count, sort, and remove regions based on properties.

Hough Transform, Statistical Functions, and Color Space Conversions

Find line segments, line endpoints, and circles. Statistical functions let you analyze the characteristics of an image. Color-space conversion accurately represents color independently from devices.

(IC) Integrated circuit design

ICs consist of miniaturized electronic components built into an electrical network on a monolithic semiconductor substrate by photolithography.

IC design can be divided into the broad categories of digital and analog IC design. Digital IC design is to produce components such as microprocessors, FPGAs, memories (RAM, ROM, and flash) and digital ASICs. Digital design focuses on logical correctness, maximizing circuit density, and placing circuits so that clock and timing signals are routed efficiently. Analog IC design also has specializations in power IC design and RF IC design. Analog IC design is used in the design of op-amps, linear regulators, phase locked loops, oscillators and active filters. Analog design is more concerned with the physics of the semiconductor devices such as gain, matching, power dissipation, and resistance. Fidelity of analog signal amplification and filtering is usually critical and as a result, analog ICs use larger area active devices than digital designs and are usually less dense in circuitry.

Software

In this project, the following software and tools were used:

Matlab

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

MATLAB is also a programming language. Like other computer programming languages, MATLAB has some decision making structures for control of command execution. These decision making or control flow structures include for loops, while loops, and if-else-end constructions. Control flow structures are often used in script M-files and function M-files. By creating a file with the extension .m, we can easily write and run programs. We do not need to compile the program since MATLAB is an interpretative (not compiled) language. MATLAB has thousand of functions, and you can add your own using m-files. MATLAB provides several tools that can be used to control the flow of a program (script or function). In a simple program as shown in the previous Chapter, the commands are executed one after the other. Here we introduce the flow control structure that make possible to skip commands or to execute specific group of commands.

A Graphical User Interface (GUI)

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. The GUI should behave in an

understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on a pushbutton, the GUI should initiate the action described on the label of the button. This chapter introduces the basic elements of the MATLAB GUIs. The chapter does not contain a complete description of components or GUI features, but it does provide the basics required to create functional GUIs for your programs.

How a Graphical User Interface Works

A graphical user interface provides the user with a familiar environment in which to work. This environment contains pushbuttons, toggle buttons, lists, menus, text boxes, and so forth, all of which are already familiar to the user, so that he or she can concentrate on using the application rather than on the mechanics involved in doing things. However, GUIs are harder for the programmer because a GUI-based program must be prepared for mouse clicks (or possibly keyboard input) for any GUI element at any time. Such inputs are known as events, and a program that responds to events is said to be event driven. The three principal elements required to create a MATLAB Graphical User Interface are :

1. Components. Each item on a MATLAB GUI (pushbuttons, labels, edit boxes, etc.) is a graphical component. The types of components include graphical controls (pushbuttons, edit boxes, lists, sliders, etc.), static elements (frames and text strings), menus, and axes. Graphical controls and static elements are created by the function uicontrol, and menus are created by the functions uimenu and uicontextmenu. Axes, which are used to display graphical data, are created by the function axes.
2. Figures. The components of a GUI must be arranged within a figure, which is a window on the computer screen. In the past, figures have been created automatically whenever we have plotted data. However, empty figures can be created with the function figure and can be used to hold any combination of components.
3. Callbacks. Finally, there must be some way to perform an action if a user clicks a mouse on a button or types information on a keyboard. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back. There must be a callback to implement the function of each graphical component on the GUI.

Creating and Displaying a Graphical User Interface

MATLAB GUIs are created using a tool called guide, the GUI Development Environment. This tool allows a programmer to layout the GUI, selecting and aligning the GUI components to be placed in it. Once the components are in place, the programmer can edit their properties: name, color, size, font, text to display, and so forth. When guide saves the GUI, it creates working program including skeleton functions that the programmer can modify to implement the behavior of the GUI. When guide is executed, it creates the Layout Editor . The large white area with grid lines is the layout area, where a programmer can layout the GUI. The Layout Editor window has a palate of GUI components along the left side of the layout area. A user can create any number of GUI components by first clicking on the desired component, and then dragging its outline in the layout area. The top of the window has a toolbar with a series of useful tools that allow the user to distribute and align GUI components, modify the properties of GUI components, add menus to GUIs, and so on. The basic steps required to create a MATLAB GUI are:

Graphical User Interface Components

This section summarizes the basic characteristics of common graphical user interface components. It

describes how to create and use each component, as well as the types of events each component can generate. The components discussed in this section are

- Text Fields
- Edit Boxes
- Frames
- Pushbuttons
- Toggle Buttons
- Checkboxes
- Radio Buttons
- Popup Menus
- List Boxes
- Slide

Text Fields: A text-field is a graphical object that displays a text string. You can specify how the text is aligned in the display area by setting the horizontal alignment property. By default, text fields are horizontally centered. A text field is created by creating a uicontrol whose style property is 'edit'. A text field may be added to a GUI by using the text tool in the Layout Editor. Text fields do not create callbacks, but the value displayed in the text field can be updated in a callback function by changing the text field's String property,

Edit Boxes: An edit box is a graphical object that allows a user to enter a text string. The edit box generates a callback when the user presses the Enter key after typing a string into the box. An edit box is created by creating a control whose style property is 'edit'. An edit box may be added to a GUI by using the edit box tool in the Layout Editor. 'Edit Box' and a text field named 'TextBox' .When a user types a string into the edit box, it automatically calls the function EditBox_Callback, This function locates the edit box using the handles structure and recovers the string typed by the user.

Frames: A frame is a graphical object that displays a rectangle on the GUI. You can use frames to draw boxes around groups of logically related objects. For example, a frame is used to group the radio buttons together. A frame is created by creating a control whose style property is 'frame'. A frame maybe added to a GUI by using the frame tool in the Layout Editor. Frames do not generate callbacks.

Pushbuttons: A pushbutton is a component that a user can click on to trigger a specific action. The pushbutton generates a callback when the user clicks the mouse on it. A pushbutton is created by creating a control whose style property is 'pushbutton'. A pushbutton may be added to a GUI by using the pushbutton tool in the Layout Editor. Function My First GUI illustrated the use of pushbuttons.

Toggle Buttons: A toggle button is a type of button that has two states: on (depressed) and off (not depressed). A toggle button switches between these two states whenever the mouse clicks on it, and it generates a callback each time. The 'Value' property of the toggle button is set to max (usually 1) when the button is on, and min (usually 0) when the button is off. A toggle button is created by creating a control whose style property is toggle button. A toggle button may be added to a GUI by using the toggle button tool in the Layout Editor. When a user clicks on the toggle button, it automatically calls the function Toggle Button _Callback. This function locates the toggle button using the handles structure and recovers its state from the' Value' property. Then, the function locates the text field and displays the state in the text field.

Checkboxes and Radio Buttons: Checkboxes and radio buttons are essentially identical to toggle buttons except that they have different shapes. Like toggle buttons, checkboxes and radio buttons have two states: on and off. They switch between these two states whenever the mouse clicks on them, generating a callback each time. The 'Value' property of the checkbox or radio button is set to max (usually 1) when they are on, and min (usually 0) when they are off. A checkbox is created by creating a control whose style property is 'checkbox', and a radio button is created by creating a control whose style property is 'radiobutton'. A checkbox may be added to a GUI by using the checkbox tool in the Layout Editor, and a radio button may be added to a GUI by using the radio button tool in the Layout Editor. Checkboxes are traditionally used to display on/off options, and groups of radio buttons are traditionally used to select among mutually exclusive options. The GUI in this figure creates three radio buttons, labeled "Option 1," "Option 2," and "Option 3." Each radio button uses the same callback function, but with a separate parameter.

Popup Menus: Popup menus are graphical objects that allow a user to select one of a mutually exclusive list of options. The list of options that the user can select among is specified by a cell array of strings, and the 'Value' property indicates which of the strings is currently selected. A popup menu may be added to a GUI by using the popup menu tool in the Layout Editor.

List Boxes: List boxes are graphical objects that display many lines of text and allow a user to select one or more of those lines. If there are more lines of text than can fit in the list box, a scroll bar will be created to allow the user to scroll up and down within the list box. The lines of text that the user can select among are specified by a cell array of strings, and the' Value' property indicates which of the strings are currently selected. A list box is created by creating a uicontrol whose style property is 'listbox'. A list box may be added to a GUI by using the listbox tool in the Layout Editor. List boxes can be used to select a single item from a selection of possible choices. In normal GUI usage, a single mouse click on a list item selects that item but does not cause an action to occur. Instead, the action waits on some external trigger, such as a pushbutton. However, a mouse double-click causes an action to happen immediately. Single-click and double-click events can be distinguished using the SelectionType property of the figure in which the clicks occurred. A single mouse click will place the string 'normal' in the SelectionType property, and a double mouse click will place the string' open' in the SelectionType property.

Sliders: Sliders are graphical objects that allow a user to select values from a continuous range between a specified minimum value and a specified maximum value by moving a bar with a mouse. The 'Value' property of the slider is set to a value between min and max depending on the position of the slider. A slider is created by creating a control whose style property is 'slider'. A slider may be added to a GUI by using the slider tool in the Layout Editor. The 'Min' property for this slider is set to zero, and the 'Max' property is set to ten. When a user drags the slider, it automatically calls the function Slider1_Callback. This function gets the value of the slider from the 'Value' property and displays the value in the text field. with the slider at some intermediate position in its range.

References:

1. How to Read an IC Part Number. *Kevin Mason* **2017**. [Sciencing]
2. Strehle, E.M.; Gray, W.K.; Gopisetti, S.; Richardson, J.; McGuire, J.; Malone, S. Can home monitoring reduce mortality in infants at increased risk of sudden infant death syndrome? A systematic review. *Acta Paediatr.* **2012**, *101*, 8–13. [Google Scholar] [CrossRef] [PubMed]
3. Red Nose. *National Scientific Advisory Group (NSAG). Information Statement: Home Monitoring*; National SIDS Council of Australia: Melbourne, Australia, 2016; Available online: https://rednose.com.au/downloads/Home_Monitoring-Safe_Sleeping-Information_Statement.pdf (accessed on 25 December 2016).

4. Min, S.D.; Yoon, D.J.; Yoon, S.W.; Yun, Y.H.; Lee, M. A study on a non-contacting respiration signal monitoring system using Doppler ultrasound. *Med. Biol. Eng. Comput.* **2007**, *45*, 1113–1119. [Google Scholar] [CrossRef] [PubMed]
5. Cheng, H.-D.; Jiang, X. H.; Sun, Y.; Wang, J., Color image segmentation: advances and prospects. *Pattern recognition* **2001,** 34, (12), 2259-2281.
6. Shih, F. Y.; Cheng, S., Automatic seeded region growing for color image segmentation. *Image and vision computing* **2005,** 23, (10), 877-886.
7. Wu, X., YIQ vector quantization in a new color palette architecture. *IEEE Transactions on Image Processing* **1996,** 5, (2), 321-329.
8. Fialka, O.; Cadik, M., FFT and convolution performance in image filtering on GPU. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, IEEE: 2006; pp 609-614.
9. Asiminoael, L.; Blaabjerg, F.; Hansen, S., Detection is key-Harmonic detection methods for active power filter applications. *IEEE Industry Applications Magazine* **2007,** 13, (4), 22-33.
10. Banerjee, A.; Dhar, A. S.; Banerjee, S., FPGA realization of a CORDIC based FFT processor for biomedical signal processing. *Microprocessors and Microsystems* **2001,** 25, (3), 131-142.
11. Rehman, A.; Mustafa, M.; Javaid, N.; Qasim, U.; Khan, Z., Analytical survey of wearable sensors. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2012 Seventh International Conference on*, IEEE: 2012; pp 408-413.
12. Bland, J. M.; Altman, D. G., Statistical methods for assessing agreement between two methods of clinical measurement. *International Journal of Nursing Studies* **2010,** 47, (8), 931-936.
13. K. Saladin, *Anatomy & Physiology: The Unity of Form and Function*, 4th ed., vol. 31, no. 1140. New York, NY: McGraw-Hill, 2007, p. 75.
14. R. Murthy and I. Pavlidis, "Noncontact measurement of breathing function," *IEEE Eng. Med. Biol. Mag.*, vol. 25, no. 3, pp. 57–67, 2006.
15. A. Alnaji, K. Gibson, S.-H. Lee, & J. Chahl, "Monitoring of cardiorespiratory signal: Principles of remote measurements and review of methods," *IEEE Access*, vol. 5, no. 17124272, pp. 15776–15790, 2017.
16. A. Alonso, J. Sasin, and N. Bottini, "Signs of respiratory disease," *Apoptosis*, vol. 85, no. 8, pp. 721–726, 2006.
17. P. Tripathy, S. Srivastava, and S. Singh, "A modified TLS-ESPRIT-based method for low-frequency mode identification in power systems utilizing synchrophasor measurements," *IEEE Trans. Power Syst.*, vol. 26, no. 2, pp. 719–727, 2011.
18. E. Billauer, Peakdet: peak detection using MATLAB, 2008, http://billauer.co.il/ peakdet.html.
19. O. Mazumder, A. S. Kundu, P. K. Lenka, and S. Bhaumik, "Multi-channel fusion based adaptive gait trajectory generation using wearable sensors," *J. Intell. Robotic Syst.*, vol. 86, no. 3–4, pp. 335–351, 2017.
20. J. Muraskas and K. Parsi, "The cost of saving the tiniest lives: NICUs versus prevention," *Virtual Mentor*, vol. 10, no. 10, pp. 655–658, 2008.
21. J. M. Bland and D. G. Altman, "Statistical methods for assessing agreement between two methods of clinical measurement," *Int. J. Nurs. Stud.*, vol. 47, no. 8, pp. 931–936, 2010.

22. J. M. Bland and D. Altman, “Statistical methods for assessing agreement between two methods of clinical measurement,” *Lancet*, vol. 327, no. 8476, pp. 307–310, 1986.

23. C. Brüser, C. H. Antink, T. Wartzek, M. Walter, and S. Leonhardt, “Ambient and unobtrusive cardiorespiratory monitoring techniques,” *IEEE Rev. Biomed. Eng.*, vol. 8, pp. 30–43, 2015.