Computers and Electronics in Agriculture 84 (2012) 68–75

Contents lists available at SciVerse ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Development of software for spectral imaging data acquisition using LabVIEW

Weilin Wang [a], Changying Li [a,*], Ernest W. Tollner [b], Glen C. Rains [a]

[a] Biological and Agricultural Engineering, University of Georgia, Tifton, GA 31793, USA
[b] Biological and Agricultural Engineering, University of Georgia, Athens, GA 30602, USA

ARTICLE INFO

Article history:
Received 2 June 2011
Received in revised form 31 January 2012
Accepted 20 February 2012

Keywords:
Hyperspectral imaging
Software
Data acquisition
LabVIEW
LCTF
Finite state machine

ABSTRACT

Developing data acquisition software is a major challenge in integrating a spectral imaging system. This paper presents the design and implementation of a data acquisition program using LabVIEW for a liquid crystal tunable filter based spectral imaging system (900–1700 nm). The module-based program was designed in a three-tier structure. The image acquisition process, modelled by a finite state machine, was implemented in LabVIEW to control the spectral imaging system to collect hyperspectral or multispectral images. The collected spectral images were encoded in general format and could be further processed by other common spectral image analysis tools. In addition, the program could be used to observe band ratio images of the test object in real-time, collect spectral images after ensemble averaging, and select region of interest for spectral image acquisitions. This program is a useful data acquisition tool for the filter-based spectral imaging system. The design and implementation techniques described in this article could also be used to develop similar spectral image acquisition programs.

Published by Elsevier B.V.

1. Introduction

In the past decade, spectral imaging, including hyperspectral imaging (HSI) and multispectral imaging (MSI), has been widely used for non-destructive inspections in various areas such as remote sensing (Chang, 2007), biomedical imaging (Schwartz, 2005), food safety and quality (Sun, 2010), surveillance (Denman et al., 2010), and conservation (Fischer and Kakoulli, 2006). A spectral imaging system acquires the spectral and spatial information of the test object simultaneously and stores the collected information into a three dimensional data matrix (a spectral image). The large amount of information contained in spectral images enables researchers to investigate both the spectral and spatial characteristics of the test object more efficiently (Lu and Chen, 1998).

Liquid crystal tunable filter (LCTF) is an electronically tuned filter which selects a narrow band of light at a specific wavelength for transmission and blocks all others (Gat, 2000). The LCTF-based spectral imaging is an important branch in spectral imaging and has been widely employed for non-destructive sensing (Singh et al., 2010; Williams et al., 2009; Gebhart et al., 2007). It has several advantages in instantaneous imaging, such as wide field of view and adjustable camera exposure time in scanning (Wang et al., 2011).

An LCTF-based spectral imaging system is a complex integration of optical and electronic hardware components requiring sophisticated software (Gat, 2000). Currently, some commercial and open source software tools, such as ENVI (ITT Visual Information Solutions, Boulder, CO, USA) and MATLAB-based hyperspectral image analysis toolbox (HIAT) (Jimnez et al., 2011), are available for visualizing and analyzing spectral images. However, a general spectral image acquisition software package for the LCTF-based system is still not readily available. Some manufacturers of LCTF-based spectral imaging systems provide commercial data acquisition software packages. This kind of commercial software, however, is often limited by its lack of flexibility and extensibility, which are highly desirable features for research oriented applications. For instance, a small change of the illumination in a spectral imaging system would require a re-calibration of the system, which is often handled by software. As a result, many researchers and engineers have to rely on customized software to meet their specific requirements.

A well-designed architecture is a critical factor for the success of any data acquisition software (Bass et al., 2003). In modern software engineering, the architecture of a software program usually follows one or several design patterns. The finite state machine (FSM), which is often described as a “Moore machine”, is a design pattern for implementing complex decision-making algorithms (Wagner et al., 2006). The FSM pattern provides good support for both design and implementation phases in software development. It allows dynamic control of the system by defining the operation of the system to a number of states and providing flexible transitions between states. Due to its effectiveness and high flexibility, FSM is one of the most common software structures used for controlling systems (Wagner et al., 2006) and real-time applications (Williams, 2006).

* Corresponding author. Tel.: +1 229 386 3915; fax: +1 229 386 3958.
E-mail addresses: wweilin@uga.edu (W. Wang), cyli@uga.edu (C. Li), btollner@engr.uga.edu (E.W. Tollner), grains@uga.edu (G.C. Rains).

0168-1699/$ - see front matter Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.compag.2012.02.010

Although most publications of the spectral imaging applications described their software programs, only a few of them presented the design and the architecture of their data acquisition software. Lerner and Drake (1999) demonstrated the design of a LabVIEW program for a push-broom microscopy hyperspectral imaging system (385–750 nm), which can collect 242 spectra at each scanned line. However, the paper mainly presented the functions of the software rather than discussing the methodologies for designing the program. Yoon et al. (2011) briefly presented their software solution for real-time inspection of fecal and ingesta on poultry carcasses using line-scan hyperspectral imaging system. In their application, multi-thread programming and double buffering memory management were applied to increase the processing speed of their online inspection program. Overall, to our knowledge, there are few publications that discussed the general design and implementation methods of data acquisition software for the spectral imaging system, particularly for an LCTF-based spectral imaging system.

This article aimed to demonstrate the whole process of the design and implementation of a LabVIEW program, which integrated an LCTF-based spectral imaging system for data acquisition. Specific objectives of this work were to: (1) design a reliable and flexible architecture for the spectral image acquisition program, (2) implement an easy-to-use program to integrate the hardware devices of the system to acquire hyperspectral/multispectral images in the spectral region of 900–1700 nm, and (3) develop several useful functions, such as collecting ensemble averaging images and previewing band ratio images, to enhance the usability of the software.

2. Overview of the hardware

The system hardware mainly consisted of a spectral imager, illumination system, frame grabber, and computer. The key hardware component, the spectral imager, included a liquid crystal tunable filter (Model Varispec LNIR 20-HC-20, Cambridge Research & Instrumentation, MA, USA), an InGaAs camera (Model SU320KTS-1.7RT, Goodrich, Sensors Unlimited, Inc., USA), and a lens (Model SOLO 50, Goodrich, Sensors Unlimited, Inc., NJ, USA). The LCTF can be tuned over the spectral region from 850 to 1800 nm with 20 nm full-width at half-maximum (FWHM). The LCTF has a working aperture of 20 mm and takes 50–150 ms to tune its bandpass to a specific wavelength. The InGaAs camera can capture 12-bit grayscale images (320×256 pixel) with a maximum speed of 60 frames per second (fps). A frame grabber (NI PCI-1426, National Instruments, Austin, TX, USA) was used to control the camera and acquire image data. The lighting source was provided by four 35 watts quartz halogen lamps (Model S4121, Superior Lighting, Fort Lauderdale, FL, USA). A digital color CCD camera (LifeCam Cinema, Microsoft, Redmond, WA, USA) was applied to collect the complementary color information of the test object since the InGaAs camera collected images in the near-infrared range.

3. Software design

3.1. Design criteria

The software was designed to provide a user-friendly data acquisition program for researchers using the LCTF-based spectral imaging system. The major criterion was to ensure that the program can reliably acquire hyperspectral or multispectral images. Given the spectral imaging system would be used in different research applications, several other aspects were also considered in the design of this program:

Usability all required functions of the program can be performed under the stated conditions and the interface of the program is friendly to its targeted users.
Flexibility the program can be operated in flexible procedures for acquiring different types of spectral images.
Reusability modules can be further used by future programs with slight modification or no modification.
Extensibility new features can be added without significant changes to the architecture of the program.
Cost-efficiency the design of the software is easy to be implemented and the cost for developing, operating, and maintaining the program is minimized.

3.2. Programming language selection

Many programming languages have been used to develop HSI/MSI software, such as C++ (Evans et al., 1998; Yoon et al., 2011), Microsoft Visual Basic (Kim et al., 2001), and LabVIEW (Lerner and Drake, 1999; Martin et al., 2006). Selection of a programming platform depends upon many factors, such as the skill of the developer and available drivers of the hardware, etc. Among these programming languages, Laboratory Virtual Instrument Engineering Workbench (LabVIEW) has several advantages over other programming languages in terms of research use. A major advantage of LabVIEW is its rich graphic user interface (GUI) widgets and hardware drivers. For instance, for this spectral imaging system, the LabVIEW NI-IMAQdx toolset provides strong support for the serial communication between the computer and the InGaAs camera via the Camera Link interface. Moreover, LabVIEW is a graphic dataflow programming language based on virtual instruments (VIs), which are virtual representations of hardware equipment. Graphic programming allows programmers to implement programs by quickly dragging and dropping icons. This unique characteristic shortens the time required for software development, which met our criterion of cost-efficiency. Thus, the LabVIEW (v8.2, National Instruments, Austin, TX, USA) was chosen to develop the software in this study.

3.3. Software architecture

Overall, this module-based program was designed in a three-tier structure (Fig. 1). The top tier is the graphic user interface (GUI) using a set of LabVIEW user interface controls. The low level communication tier consists of a set of LabVIEW I/O functions (National Instruments, Austin, TX, USA) which send commands and receive data to/from the hardware devices. The middle tier (data collecting and processing) manipulates hardware and builds spectral images. The tier includes five modules: the InGaAs camera control, LCTF control, USB color camera control, spectral image reconstruction, and system configuration module. The core functions of each module are organized as independent virtual instruments (sub-VIs) to improve the reusability of the program. Details of these modules are introduced in the following sections.

3.4. Model for data acquisition

The data acquisition process was modeled by using an FSM to achieve high flexibility and extensibility. The FSM follows the design pattern recommended by National Instruments, which consists of while loops, case structures, shift registers, and transition codes. The overall data acquisition process has eight segments and each segment executes a set of processes as a batch to complete a relatively independent task. These segments and related program status are modeled as states (Fig. 2). A state can either be followed by another state, or wait for another user/system event, depending on history activities and current