SPLASSH: Open source software for camera-based high-speed, multispectral in-vivo optical image acquisition

Ryan Sun,[1,2] Matthew B. Bouchard,[1,3] and Elizabeth M. C. Hillman[1]

[1]*Laboratory for Functional Optical Imaging, Departments of Biomedical Engineering and Radiology, Columbia University, New York, NY 10027*
[2]*rs2878@columbia.edu*
[3]*mb2936@columbia.edu*
http://www.bme.columbia.edu/~hillman

Abstract: Camera-based in-vivo optical imaging can provide detailed images of living tissue that reveal structure, function, and disease. High-speed, high resolution imaging can reveal dynamic events such as changes in blood flow and responses to stimulation. Despite these benefits, commercially available scientific cameras rarely include software that is suitable for in-vivo imaging applications, making this highly versatile form of optical imaging challenging and time-consuming to implement. To address this issue, we have developed a novel, open-source software package to control high-speed, multispectral optical imaging systems. The software integrates a number of modular functions through a custom graphical user interface (GUI) and provides extensive control over a wide range of inexpensive IEEE 1394 Firewire cameras. Multispectral illumination can be incorporated through the use of off-the-shelf light emitting diodes which the software synchronizes to image acquisition via a programmed microcontroller, allowing arbitrary high-speed illumination sequences. The complete software suite is available for free download. Here we describe the software's framework and provide details to guide users with development of this and similar software.

OCIS codes: (170.0170) Medical optics and biotechnology; (110.4234) Multispectral and hyperspectral imaging; (170.2655) Functional monitoring and imaging; (170.6280) Spectroscopy, fluorescence, and luminescence.

References and links

1. A. J. Blood, N. Pouratian, and A. W. Toga, "Temporally staggered forelimb stimulation modulates barrel cortex optical intrinsic signal responses to whisker stimulation," J. Neurophysiol. **88**(1), 422–437 (2002).
2. I. Vanzetta, R. Hildesheim, and A. Grinvald, "Compartment-resolved imaging of activity-dependent dynamics of cortical blood volume and oximetry," J. Neurosci. **25**(9), 2233–2244 (2005).
3. D. Shoham, D. E. Glaser, A. Arieli, T. Kenet, C. Wijnbergen, Y. Toledo, R. Hildesheim, and A. Grinvald, "Imaging cortical dynamics at high spatial and temporal resolution with novel blue voltage-sensitive dyes," Neuron **24**(4), 791–802 (1999).
4. G. Themelis, J. S. Yoo, K. S. Soh, R. Schulz, and V. Ntziachristos, "Real-time intraoperative fluorescence imaging system using light-absorption correction," J. Biomed. Opt. **14**(6), 064012 (2009).
5. S. L. Troyan, V. Kianzad, S. L. Gibbs-Strauss, S. Gioux, A. Matsui, R. Oketokoun, L. Ngo, A. Khamene, F. Azar, and J. V. Frangioni, "The FLARE intraoperative near-infrared fluorescence imaging system: a first-in-human clinical trial in breast cancer sentinel lymph node mapping," Ann. Surg. Oncol. **16**(10), 2943–2952 (2009).
6. M. S. Rahman, N. Ingole, D. Roblyer, V. Stepanek, R. Richards-Kortum, A. Gillenwater, S. Shastri, and P. Chaturvedi, "Evaluation of a low-cost, portable imaging system for early detection of oral cancer," Head Neck Oncol **2**(1), 10 (2010).
7. J. Lawlor, D. W. Fletcher-Holmes, A. R. Harvey, and A. I. McNaught, "In Vivo Hyperspectral Imaging of Human Retina and Optic Disc," Invest. Ophthalmol. Vis. Sci. **34**, 4350 (2002).
8. K. Svanberg, I. Wang, S. Colleen, I. Idvall, C. Ingvar, R. Rydell, D. Jocham, H. Diddens, S. Bown, G. Gregory, S. Montán, S. Andersson-Engels, and S. Svanberg, "Clinical multi-colour fluorescence imaging of malignant tumours--initial experience," Acta Radiol. **39**(1), 2–9 (1998).

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

9. C. W. Zemlin, O. Bernus, A. Matiukas, C. J. Hyatt, and A. M. Pertsov, "Extracting intramural wavefront orientation from optical upstroke shapes in whole hearts," Biophys. J. **95**(2), 942–950 (2008).
10. E. M. Hillman, and A. Moore, "All-optical anatomical co-registration for molecular imaging of small animals using dynamic contrast," Nat. Photonics **1**(9), 526–530 (2007).
11. M. Rudin, and R. Weissleder, "Molecular imaging in drug discovery and development," Nat. Rev. Drug Discov. **2**(2), 123–131 (2003).
12. R. M. Levenson, and J. R. Mansfield, "Multispectral imaging in biology and medicine: slices of life," Cytometry A **69**(8), 748–758 (2006).
13. M. B. Bouchard, B. R. Chen, S. A. Burgess, and E. M. C. Hillman, "Ultra-fast multispectral optical imaging of cortical oxygenation, blood flow, and intracellular calcium dynamics," Opt. Express **17**(18), 15670–15678 (2009).
14. R. Sun, M. B. Bouchard, S. A. Burgess, A. J. Radosevich, and E. M. C. Hillman, "A Low-Cost, Portable System for High-Speed Multispectral Optical Imaging," in *OSA Biomedical Optics*(Optical Society of America, Miami, FL, 2010).
15. E. M. C. Hillman, "Hillman Lab: Research: Instrumentation," (Laboratory for Functional Optical Imaging, 2010), *http://www.bme.columbia.edu/~hillman/Instrumentation.html*, Accessed May 25, 2010, 2010.
16. O. 1394-Trade, "Frequently Asked Questions," (1394 Trade Organization, 2008), *http://www.1394ta.org/consumers/FAQ.html*, Accessed May 25, 2010, 2010.
17. Microsoft, "Performance of 1394 devices may decrease after you install Windows XP Service Pack 2," (Microsoft, 2006), *http://support.microsoft.com/kb/885222*, Accessed May 25, 2010, 2010.
18. Microsoft, "1394 Bus Driver in Windows 7," (Microsoft, 2009), *http://www.microsoft.com/whdc/connect/1394_Windows7.mspx*, Accessed May 25, 2010, 2010.
19. National Instruments, "Ring Acquisitions," (National Instruments, 2007), *http://zone.ni.com/devzone/cda/tut/p/id/4001*, Accessed May 25, 2010, 2010.
20. Advanced Micro Devices, "ATI Stream Technology," (2010), *http://www.amd.com/US/PRODUCTS/TECHNOLOGIES/STREAM-TECHNOLOGY/Pages/stream-technology.aspx*, Accessed July 23, 2010.
21. Q. Fang, and D. A. Boas, "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units," Opt. Express **17**(22), 20178–20190 (2009).
22. Khronos Group, "OpenCL," (2010), *http://www.khronos.org/opencl/*, Accessed July 23, 2010.
23. Microsoft, "DirectCompute PDC HOL - Home - " (2008), *http://code.msdn.microsoft.com/directcomputehol*, Accessed July 23, 2010.

1. Introduction

One of the simplest forms of optical imaging involves illuminating tissue with light and capturing wide-field images using a camera. Images acquired in this manner can provide detailed information about the structure and function of living tissues via contrast from endogenous and exogenous absorbers and fluorophores. High-speed imaging can capture dynamic events such as changes in blood flow or responses to stimulation. Multispectral imaging can provide spectroscopic information to allow quantitative analysis of multiple absorbers and/or fluorophores in parallel.

Camera-based optical imaging has been widely employed for neuroscience research, where imaging of 'Optical Intrinsic Signals' (OIS) corresponding to changes in cortical hemoglobin concentrations have proven valuable for mapping regions of activation [1,2]. Fluorescence imaging of voltage sensitive dyes has also been extensively used for neuroscience studies [3]. Beyond these applications, wide-field multispectral and fluorescence imaging can be valuable for clinical applications such as characterizing the structure and function of a broad range of tissues including cervix, retina, oral mucosa, gastric tissues, the heart, internal organs during surgery and skin [4–9], and for basic research applications such as small animal molecular imaging [10–12].

Despite the simplicity of camera-based optical imaging, commercial in-vivo imaging systems are restricted in their utility for more than their specifically designed application, and typically cost in excess of $100,000. While researchers can purchase their own cameras and construct their own imaging systems, commercially available cameras are typically available with either user-oriented software packages optimized for in-vitro microscopy (e.g. point-and-click single frame acquisition, or slow time-lapse image acquisition routines), or manufacturers provide only software development kits (SDKs) that leave the user to construct their own control software.

For in-vivo imaging studies, a camera-based system requires a number of design parameters that differ from those characteristic of microscopy-oriented systems, specifically:

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

(1) The ability to acquire 'runs', or 'sets', of data at high frame rates for periods of between 10 and 60 seconds. The start of each run must usually be triggered either internally by the camera or externally via an outside trigger timed precisely to stimulus or other experimental events.

(2) The ability to acquire multiple 'runs' of data in sequence without additional user input.

(3) The option to save acquired images in a standard image format, suitable for analysis by custom or commercial software analysis packages.

(4) A simple graphical user interface (GUI) capable of providing intuitive and rapid control over major camera parameters (e.g. ability to set spatial binning, frame rates, regions of interest, etc.) and image acquisition routines.

(5) The flexibility to image both fluorescence and absorption contrast, as well as the capability to switch quickly between the two during an experiment.

(6) The capacity to operate with multiple light sources for multispectral and/or fluorescence image acquisition.

(7) Be compatible with physically adaptable hardware which can be arbitrarily positioned around living tissues/animals during imaging.

(8) Be compatible with a wide range of imaging optics from widefield lenses to high optical power microscope objectives.

(9) Require relatively low monetary investment and be simple to implement.

We recently introduced a new approach to high-speed in-vivo multispectral and fluorescence optical imaging [13] and have moreover developed a second generation, compact, portable, low-cost, modular design [14]. To control our second generation system we developed open-source GUI software programmed in C which we have named SPLASSH (SPectral Light Acquisition Software for Signaling and Hemodynamics). SPLASSH is novel software capable of controlling a wide range of IEEE 1394 Firewire standards compliant cameras as part of a simple to assemble high-speed multispectral and fluorescence in-vivo optical imaging system. Here, we detail the modular software architecture which comprises SPLASSH and show how our system provides a simple yet highly flexible framework for high-speed multispectral optical imaging. By releasing SPLASSH's source code to the research community we hope to make high-speed multispectral and fluorescence imaging more accessible to a larger number of dynamic imaging applications, and foster further development of the software package. This paper also provides valuable information for those wishing to develop their own optical imaging systems and control software. Links to download the software source code for SPLASSH can be found on our Laboratory's website [15].

2. High-speed, portable, low-cost multispectral optical imaging system

Our SPLASSH software package can be used to control a wide range of Firewire cameras, and has functionality to also control synchronized light sources and external triggers. Below, we provide a generalized overview of the hardware required to construct a representative system, and explain how each component interfaces with the SPLASSH software package.

At the core of our system are four principal elements: an IEEE 1394 Firewire camera, a camera control computer, a low-cost programmable microcontroller, and one or more modulatable light source(s) Fig. 1. The monochrome camera can be CMOS or CCD, and must be capable of outputting a TTL signal indicating exposure of a given frame in real time (the 'Camera Exposure' signal, black line, in Fig. 1). The camera control computer connects to the camera and sets image acquisition properties (spatial binning, frame rate, etc.), coordinates

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

image acquisition routines, and saves acquired images. The programmable microcontroller, connected to the camera control computer, receives this 'Camera Exposure' signal and uses it to generate digital signals that control the system's light sources. Light sources can be modulatable high power light emitting diodes (LEDs) or other sources such as laser diodes.

Using a programmable microcontroller allows the system's light sources to be modulated in synchrony with each image frame, which provides a range of benefits. Firstly, at high frame rates inexpensive cameras without a high-speed shutter can suffer from a streaking artifact due to exposure of the imaging chip during data read-out. Using the 'Camera Exposure' signal, the microcontroller can precisely strobe the light source such that it only illuminates the sample during image acquisition and not during read-out. This in turn also reduces photodamage of the sample. Further, this functionality allows multiple different light sources to be strobed in sequence, allowing acquisition of high-speed multispectral or interlaced reflectance and fluorescence images (see Section 3.4 for more details on multispectral illumination protocols). Since the microcontroller derives these illumination drive signals from the camera in real-time, image acquisition can proceed at, or close to, the maximum frame rate of the free-running camera, rather than requiring the camera to be triggered from a signal derived from either the computer or a wavelength-switching element such as a filter wheel. By outsourcing the function of switching the light sources to the programmable microcontroller, the camera control computer can focus on grabbing and writing data to disk at its maximum rate. Programming of the microcontroller is integrated into the SPLASSH control software such that it sets up the microcontroller to generate a defined strobe order at the start of every new image acquisition sequence. This means that strobe sequences can be readily altered as needed within an experiment.

The microcontroller used in our system is widely available and costs less than $100. If modulated light sources are not required, the SPLASSH software is also able to acquire images without light source strobing. In this case, the light sources must be operated in continuous wave (CW) mode and the microcontroller does not need to be included in the system.

With appropriate choice of camera lens, illumination wavelengths, and optical emission filters, this generic optical imaging platform can be easily modified to accommodate a broad range of research protocols. For example, a static excitation-blocking emission filter can be placed in front of the camera to allow interlaced acquisition of fluorescence and reflectance images, as long as reflectance wavelengths are passed by the emission filter. For imaging that is precisely timed to external events, the camera's external trigger input (red line in Fig. 1) can be enabled using the SPLASSH software. This TTL signal instructs the camera when to begin acquisition of a preprogrammed image acquisition sequence.

The SPLASSH software package interfaces with hardware components through a set of modular functions which control every aspect of the system and are accessed from a single GUI. SPLASSH communicates directly with Firewire cameras over the Firewire bus to manipulate camera properties (e.g. frame rate, exposure time, spatial binning mode, etc.). It communicates with the microcontroller over a USB RS-232 serial port to set the illumination strobe sequences. Within the camera control computer, SPLASSH coordinates all events during image acquisition from pre-allocating space in the computer's memory to providing file saving information for organizing acquired images into a logical folder structure.

We have currently implemented this system with an AVT Pike F-032B Firewire 1394b camera, Arduino Diecimila microcontroller (SparkFun Electronics), a HP EliteBook 8510p laptop PC, a Firewire 1394b ExpressCard (EC1394B2, StarTech), and LEDs from Thorlabs (MxLED series with LEDD1 controllers) for a total cost of approximately $5,000 USD. With only four main components, this setup easily fits into a backpack and can be assembled relatively quickly (< 10 minutes). This system is capable of streaming images to disk at the Pike F-032B camera's full 480 x 640 (height x width) pixel resolution and 14-bit resolution at 90 frames per second. Large increases in frame rate are possible with spatial binning and/or reduced regions of interest. At these high frame rates, the rate limiting factor is no longer the frame rate of the camera, but the amount of data that can be passed from the camera and saved

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

to the computer. With this concern in mind, it is relatively simple to implement this system design with SPLASSH running on a desktop computer. Desktop computers typically offer faster internal data transfer rates and faster hard drive write speeds at the same price point as laptops, making these systems faster and more economical for a loss of portability.

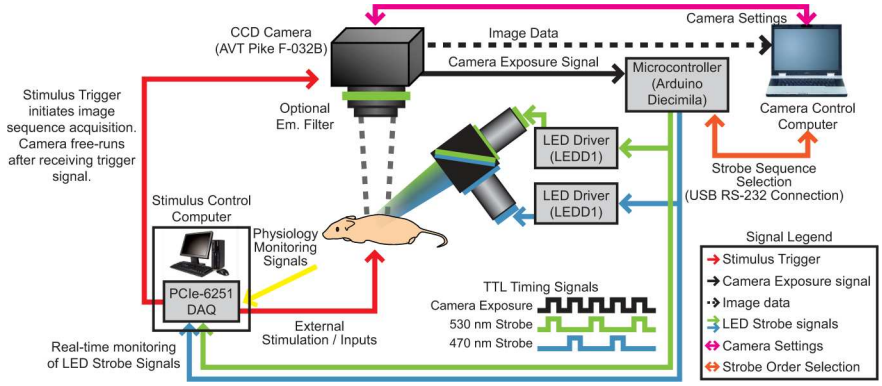

Fig. 1. High-speed, low-cost, portable multispectral optical imaging system. The system consists of four principal elements: an IEEE 1394 Firewire camera, a camera control computer, a programmable microcontroller, and modulatable light sources (LEDs are shown here). Strobed illumination is driven via the microcontroller and derived from the TTL 'Camera Exposure Signal' generated by the camera. Data from the camera is streamed to the computer's hard disk. All components of the system are modular and can be replaced with compatible pieces of hardware (for example, the LEDs can be replaced with laser diodes). An optional 'stimulus control computer' is also shown which can be used to drive external triggers or stimuli and/or log experimental parameters such as animal physiology.

3. SPectral Light Acquisition Software for Signaling and Hemodynamics

The SPLASSH software package integrates all aspects of image acquisition into a single GUI. The GUI was composed using the C programming language and was written within the National Instruments LabWindows/CVI environment. To interface with external hardware (e.g. the camera and the programmable microcontroller), SPLASSH uses a set of commercially available hardware interface drivers (Advanced Analysis, NI-VISA, NI-IMAQ, NI-IMAQdx, IMAQ Vision, Toolbox, Utility, Formatted I/O, CVI Run Time Engine, User Interface, and ANSI C, and C Standard I/O, all from National Instruments). SPLASSH was written for PCs running Windows operating systems. The open source Arduino compiler software for programming microcontroller strobe sequences can be found at www.arduino.cc.

Below we provide an overview of the major features of the SPLASSH software package. First, we discuss a significant legacy issue connected with Windows operating systems and high-speed IEEE Firewire cameras.

3.1 Windows operating systems and high-speed Firewire 1394b

Most IEEE 1394 Firewire cameras currently operate within one of two different data rate ranges: 1394a is the data rate standard most commonly referred to as "Firewire" and can be configured to transfer data at one of three rates: 100, 200, or 400 Megabits per second (Mbit/s) (Firewire 100, 200, and 400 or S100, S200, and S400, respectively). 1394b is a newer Firewire standard and can operate up to 800 Mbit/s, as well as any of the data rates available under the 1394a standard [16].

SPLASSH has been extensively tested on the Microsoft Windows XP operating system running on personal computers and is capable of operating Firewire cameras at any of the standard data rates. However, for SPLASSH to be able to control cameras at data rates above S100 on computers running Windows XP Service Pack 2 or Service Pack 3 (SP2 or SP3) a 'SP1 1394 driver downgrade' must be performed. This downgrade replaces the SP2 or SP3 versions of Windows XP system files used for Firewire interfaces with their Service Pack 1

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

versions. Windows XP SP1 is the only version of the Windows XP operating system where greater than S100 Firewire data rate transfers are possible [17]. This Firewire driver 'downgrade' is not available for Windows Vista systems. However, Windows 7 has been developed and released with native 1394b support and does not require a driver downgrade [18]. Windows XP is still viewed as the most stable platform for SPLASSH and the one that will provide the smoothest user experience, although those who prefer to use Windows 7 should not experience major inconveniences. Windows Vista will not support SPLASSH at Firewire data rates above S100.

The second hardware issue to address for SPLASSH to operate cameras above the S100 data rate is the camera control computer's Firewire connection. Some PCs ship with built-in Firewire ports which are typically 1394a six pin connections. For desktop computers without built-in Firewire connections, a large number of hardware manufacturers produce 1394a and 1394b (nine pin connections) cards with multiple bus connection standards (PCI, PCI-X, and PCI-Express among the most common for PCs). For maximum performance it is recommended to utilize the fastest Firewire standard which is compatible with the bus connection on a user's motherboard. SPLASSH is compatible with both types of Firewire connections, but data throughput may be decreased if a slower connection is used. Firewire adapter cards for laptops are also available either via PCMCIA or the faster ExpressCard ports. Our laptop did not come with a built-in Firewire connection. A 2 port 1394b 34-pin Express Card was used to provide a high-speed Firewire bus connection.

During system initialization, SPLASSH is able to detect and set the camera's data rate to those speeds offered by the control computer's Firewire connection. Typically it is advisable to select the fastest available rate to allow for the most rapid transfer of images from the camera to the camera control computer's hard drive; however it is not required.

3.2 SPLASSH control GUI interface

The SPLASSH GUI was designed to integrate modular functions into a seamless interface which allows the user to execute a range of image acquisition routines (Fig. 2). The modularity of SPLASSH's underlying routines enables developers to add or subtract functionality as their experiment requirements demand. The following provides a short summary, referenced to the panels of the GUI indicated in Fig. 2 of SPLASSH's modular functions. Detailed comments on each subroutine can be found in the source code:

(1) Available image acquisition routines include 'Single Grab', 'Preview', and 'Movie'. 'Single Grab' acquires, saves, and displays one image at the designated camera settings. 'Movie' saves a pre-determined number of images to the camera control computer's hard disk (see Section 3.3 for a discussion on the 'Movie' acquisition routine). 'Preview' displays real-time, sample movie data which is not saved to disk, can be analyzed in real-time, and is typically used during alignment procedures. Multispectral illumination can be provided during both 'Movie' and 'Preview' acquisitions. Acquired images are displayed in their own pop-up windows, separate from the control GUI.

(2) The available camera attributes can be controlled from the diagnostic panel. SPLASSH automatically detects and enumerates all of the available attributes in this panel such as Brightness and Gain during system start up.

(3) The desired shutter speed of the camera can be input in units of either frame rate or exposure time. The other attribute updates automatically, unless an invalid parameter is entered, in which case both reset to defaults (see Section 3.5 for a discussion of camera specific attributes).

(4) Multispectral illumination of up to three wavelengths strobed sequentially is currently supported (see Section 3.4 for a discussion of strobe sequencing with a programmable microcontroller). Light sources can be toggled on and off for 'Preview' acquisitions under CW illumination or set to strobe in selected sequences

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

during ‘Movie’ and ‘Preview’ acquisitions. SPLASSH does not let image acquisition proceed when invalid strobe sequences are entered. SPLASSH is able to operate without multispectral illumination and does not require connection to a programmable microcontroller if desired.

(5) Multiple file types are supported, although best performance is obtained when images are saved as uncompressed TIFFs. An option to switch between camera-supported pixel bit-depths is provided. An interface to control in-camera spatial binning is provided. Spatial binning averages neighboring pixels into one large pixel for increased signal-to-noise but at the cost of lower spatial resolution (see Section 3.5 for a discussion of camera specific attributes).

(6) For experiments where external triggering of multiple ‘runs’ is necessary, it is possible to set the length of acquisition and number of ‘runs’ beforehand.

(7) If desired, a sample image can be displayed after a predetermined amount of time during ‘Movie’ acquisitions to offer interim data diagnostics.

(8) Images can be saved to any desired directory under any name. In case the user does not increment the image filename between acquisitions, SPLASSH appends an increasing number to the filename to designate its order and avoid unintended file overwriting.

(9) Acquisition cannot proceed unless all status indicators in this panel are green. A variety of error messages are included in all routines to prompt users should they make common mistakes when configuring the system. Status updates are also included to indicate successful manipulation of the system.

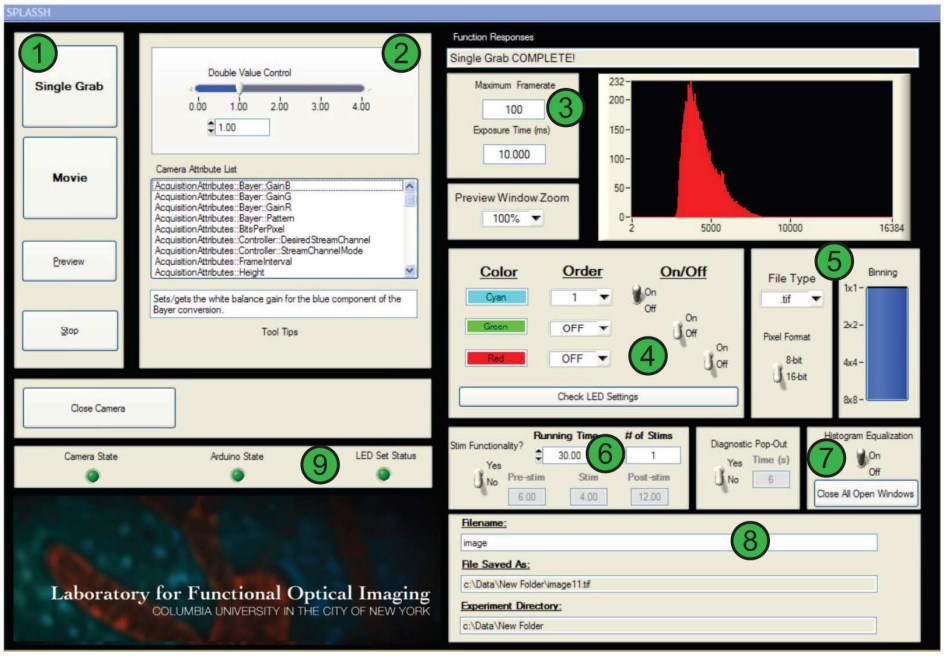

Fig. 2. The SPLASSH Graphical User Interface provides a single window from which to control all aspects of multispectral and fluorescence imaging acquisition. The GUI is organized into panels which contain buttons that access similar types of high-level routines (see Section 3.2 for descriptions). Acquired images are displayed in their own pop-up windows, separate from the control GUI. When initializing the program, SPLASSH dynamically communicates with the camera to determine its capabilities and available camera properties. SPLASSH then updates the corresponding GUI interfaces to reflect these values (e.g. the number of available binning modes).

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

During ‘Single Grab’ and ‘Movie’ image acquisitions an ‘Information File’, which contains a list of the system parameters selected for the given acquisition, is generated and saved to disk in a text file format before and after each acquisition. Each Information File is uniquely named to match the image filename and contains all relevant system status information for simple record-keeping. The Information File is saved before image acquisition begins in order to provide a record of the image acquisition in case of a system crash during image acquisition. During post-acquisition image analysis, the Information File can be read by analysis software to inform it of the parameters of the image acquisition (e.g. frame rate, acquisition length, number of acquired frames, etc.).

Although the routines called by each of the GUI interfaces are interconnected, it is important to remember that they are very modular in nature. Each routine calls subfunctions which largely execute three types of actions: system preparation, acquisition, and error-checking. Separating the high-level routines into sub-functions is designed to limit system-wide errors and provide concise information about faults that do occur. In addition, we hope that this particular architecture will make it easier for the research community to augment SPLASSH’s capabilities piece by piece.

3.3 High-speed image acquisition routines

The most important function performed by SPLASSH is to orchestrate the series of events necessary to perform high-speed ‘Movie’ acquisitions. During ‘Movie’ acquisitions, a series of images are acquired at a fixed rate and stored on the camera control computer’s hard disk. This function is typically referred to as ‘streaming images to disk.’ While many commercially available microscope camera software packages offer ‘Movie’ acquisition routines, they are not designed for the large quantities of data characteristic of long runs or high-speed, repeated, triggered acquisitions. These software packages commonly store acquired images in the computer’s RAM during acquisition before writing the images to disk after the acquisition is complete. For in-vivo imaging studies, it is desirable for the interval between frames to be on the order of 0.1 – 0.01 s, (frame rates of 10 – 100 frames per second) acquired over 10 – 60 s making storage of acquired images in RAM during acquisition not feasible.

We have therefore implemented a more efficient streaming to disk routine in SPLASSH which performs the least number of steps necessary to ‘grab’ the image data from the camera, pass it through the camera control computer, and write the image data to the hard disk. In addition, SPLASSH uses a simple file storage scheme which makes it easy to locate and manipulate saved images and allows continuous data acquisition limited only by hard disk storage space.

The major events of this routine are illustrated in Fig. 3. Upon user selection of a ‘Movie’ acquisition, SPLASSH first determines if the system has been successfully initialized and then locks the GUI buttons to prevent the user from inadvertently accessing another routine during image acquisition. SPLASSH next prepares the necessary information within the camera control computer for streaming images to disk: it allocates memory in the computer’s RAM for a ring buffer [19], determines a unique filename based on the user defined filename in the GUI (panel 8, Fig. 2), and creates folders to save the incoming image data. If the external trigger mode has been selected, the camera’s 1394 Trigger attribute is modified to reflect this. Next the total number of frames is calculated from the user defined acquisition length multiplied by the chosen image acquisition rate. When performing multispectral imaging, single reference images are acquired with each of the light sources selected for the ‘Movie’ acquisition. These images provide a reference for how the sample appeared at the beginning of the ‘Movie’ acquisition and allows the user to ensure that all have good signal and / or are not saturated. The last event to occur before image acquisition begins is the saving of the pre-acquisition Information File. Image acquisition then begins upon receipt of the image acquisition start trigger. If the trigger is internal to the camera, image acquisition begins immediately. Otherwise the system waits until an external trigger (falling edge TTL signal) is received on the Camera’s external trigger port (Camera Input 1 on the AVT Pike F-032B, for example). During image acquisition, drivers provided by the National Instruments software

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

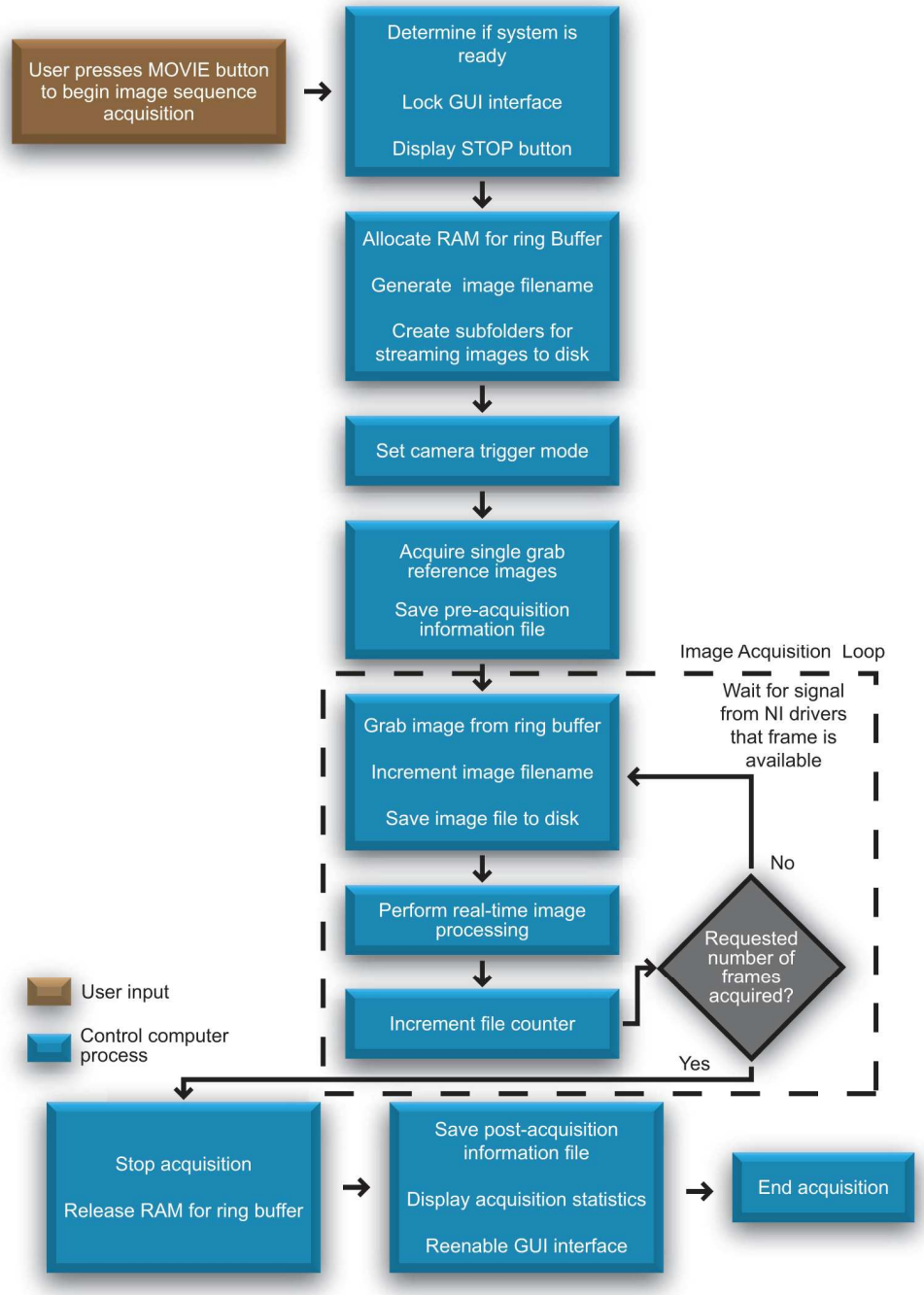

Fig. 3. 'Movie' Image Acquisition Routine Flowchart details the major subroutines called during a 'Movie' image sequence acquisition. Before the 'Movie' acquisition can begin, SPLASSH executes a series of preparatory subroutines. After the camera begins image acquisition, SPLASSH enters the image acquisition loop in which frames are grabbed from the ring buffer, the image filename is incremented to reflect the number of the image acquired in the sequence, and the image is saved to the camera control computer's hard disk. Once the number of requested frames is acquired (determined by the frame rate multiplied by the length of the movie) the image acquisition loop ends. To end the 'Movie' acquisition, SPLASSH executes a series of closing subroutines that prepare the system for the next acquisition.

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

'grab' images off the camera as they become available and place them within the ring buffer. The drivers then signal SPLASSH that an image is available. When this signal is received, the SPLASSH software executes the following four actions:

(1) When an image is available the image is 'grabbed' from the Ring Buffer and placed into a single image buffer, a number reflecting the number of the image acquired in the sequence is appended to the filename, and the image buffer is then saved to disk.

(2) Real-time processing of the images can be performed here (e.g. real-time display, average image intensity, etc.). Real-time processing during 'Movie' acquisitions has not been implemented in SPLASSH currently.

(3) The image file counter, which reflects the number of images that have been acquired during the current 'Movie' acquisition, is incremented.

(4) The image file counter is compared to the total number of images that need to be acquired to complete the acquisition. If the image file counter is below this number, the system waits for the NI drivers to signal that the next frame is available. If the image file counter matches the number of frames which need to be acquired, SPLASSH stops the image acquisition routine.

Upon stopping the image acquisition, SPLASSH frees the memory allocated for the ring buffer, saves the final version of the Information File (which reflects a successful image acquisition), displays statistics on the image acquisition routine just completed (its total length of time and the number of frames missed, if any), and re-enables the GUI interface.

The rate at which SPLASSH can stream images to hard disk is determined by the Firewire data rate, the speed of the camera control computer's RAM, internal bus data transfer speeds, and the write speed of the hard disk. The faster these data transfers can occur, the faster the rate at which SPLASSH can stream images to disk. The length of time over which the system can be used to acquire images is limited only by the amount of available hard drive storage space.

'Preview' acquisitions largely follow the same flow chart used to describe the 'Movie' acquisition routine. However, during 'Preview' acquisitions, images and 'Information Files' (see Section 3.2) are not saved to the hard disk. The same looping structure used to write images to disk is instead used to display images on the camera control computer's monitor in real-time. The additional processing time made available by not streaming images to disk can be used for more computationally intensive real-time image processing routines. A real-time histogramming routine is currently implemented in SPLASSH which provides a real-time histogram of the live image stream displayed within the GUI (top right panel, Fig. 2).

3.4 Multispectral illumination driven with a programmable microcontroller

Multispectral imaging is typically achieved by acquiring a series of images with different illumination wavelengths, by using a broadband light source and selectively filtering the wavelengths of light reaching the camera, or by using a Bayer-mask type color camera. The latter approach is inflexible, does not permit interlaced fluorescence and reflectance imaging, and typically provides worse signal to noise due to reduced pixel area. For approaches that require mechanical switching of filters, multispectral acquisition speed is hindered by both the time required for switching and the need to trigger camera image acquisition in synchrony with wavelength switching rather than letting the camera free-run at its maximum speed.

To overcome these limitations, SPLASSH combines a microcontroller with modulatable light sources to form a unique digital alternative to the filter wheel or other mechanical methods of selecting wavelengths [13]. The sequence of events for multispectral imaging is illustrated in Fig. 4. The process begins when a user selects a preprogrammed illumination pattern from the GUI. If an invalid sequence has been selected, no illumination is set and the user must start again (error path in Fig. 4). If an acceptable order is input, the sequence is

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

passed to the microcontroller via the USB RS-232 bus, and the microcontroller analyzes the information to determine how it will loop through the light sources in the desired order. The

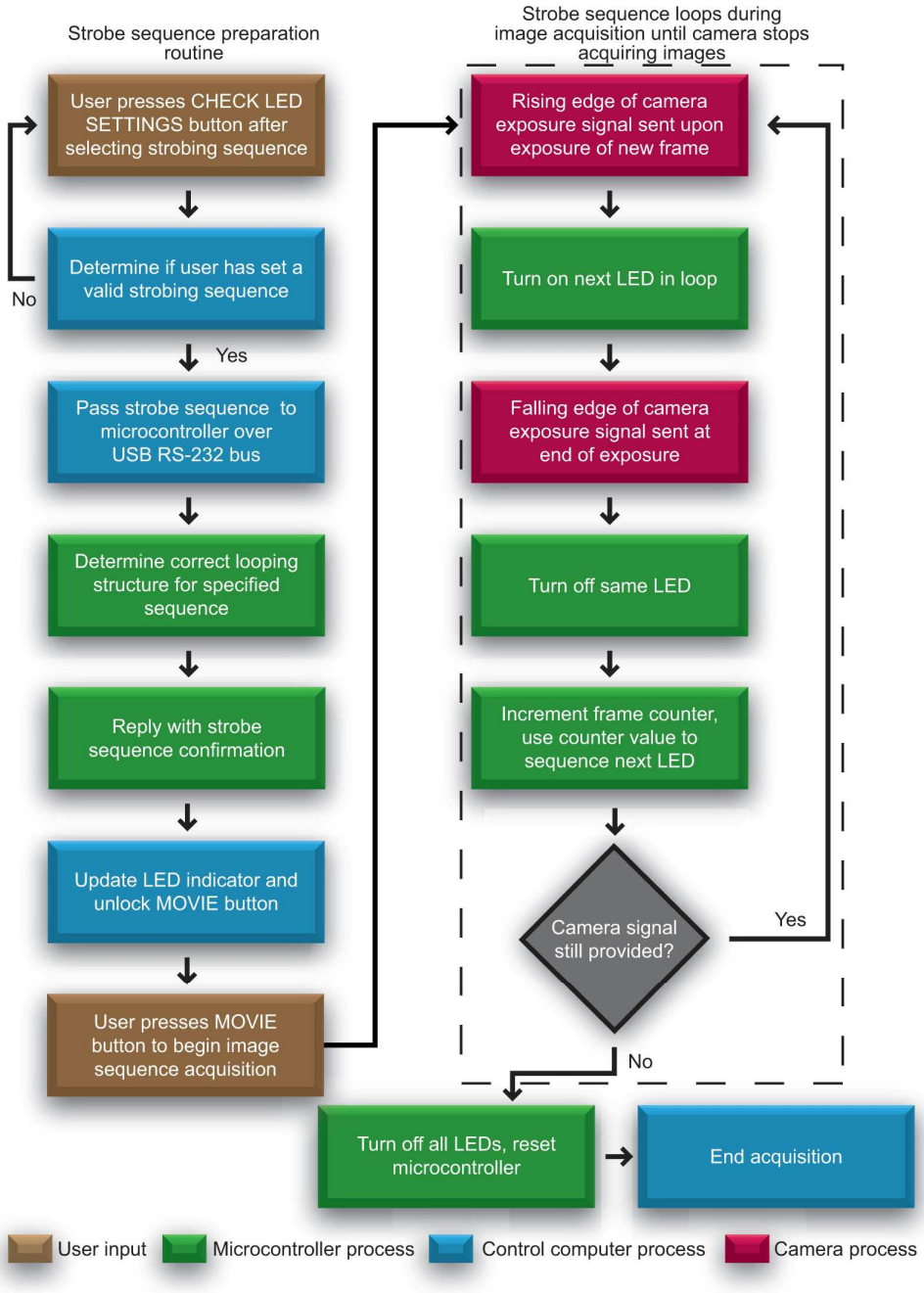

Fig. 4. Multispectral illumination with a microcontroller flowchart. Multispectral illumination is provided through the interface of a microcontroller, control computer, and camera. Once the user inputs a desired strobing sequence from the GUI, the control computer checks the validity of the pattern and instructs the microcontroller to use a preprogrammed strobe sequence. If the process is successful, the user is allowed to begin image acquisition. During acquisition the microcontroller is driven by the camera to turn the LEDs on/off. A frame counter provides the next LED in the sequence. When acquisition ends, the microcontroller is reset.

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

microcontroller then passes the information it has received back to the control computer to verify the parameters. After receiving this confirmation, the control computer enables the "Movie" option, and the user can now choose to begin their acquisition. At this point the system enters a loop which continues until the control computer registers that it is time to terminate the acquisition (dotted black box Fig. 4).

During image acquisition, the camera outputs a TTL signal indicating exposure of every frame ('Camera Exposure Signal', Fig. 2). When the microcontroller reads the rising edge of this signal it turns on a selected light source via one of the microcontroller's digital outputs. The subsequent falling edge of the 'Camera Exposure Signal' causes the microcontroller to turn off that same light source. A frame counter is then incremented in the microcontroller software to direct it to the next light source in the pre-programmed sequence. Each acquisition loops through this same series of events until the control computer stops directing the camera to acquire images. In the absence of a Camera Exposure Signal, the microcontroller turns off all of the light sources and the acquisition ends.

The illumination strobe sequences run by the programmable microcontroller must be preprogrammed into the microcontroller before using the SPLASSH GUI. This involves composing code in the microcontroller's native software development environment and 'burning' the code directly to the microcontroller's memory (in the case of Arduino microcontrollers, the software is freely available on www.arduino.cc and consists of a C-type language with a small number of functions). The strobe sequence code only needs to be 'burned' to the microcontroller's onboard memory once. The code will persist in the microcontroller's memory until the microcontroller chip is damaged or new code is 'burned' in place of it. Arbitrary strobe sequences can thus be devised and preprogrammed by any developer of the SPLASSH source code. Currently, we have developed microcontroller code and the SPLASSH GUI for serial strobe sequences of up to three light sources in arbitrary orders (panel 4, Fig. 2). For users of SPLASSH who wish to create new strobe sequences, composition of microcontroller strobe code as well as modification of the SPLASSH GUI to allow for selection of the new strobe sequences is necessary.

Light sources compatible with SPLASSH must be modulatable via a TTL input. Numerous commercial LED and laser diode drivers are available that incorporate this functionality, allowing plug-and-play system set up (LEDD1x and LDCxxxC series, Thorlabs, Model 8500 series, Newport). If custom light sources are to be used, it is important to ensure that response-times are fast relative to frame exposure times, and that light sources do not exhibit significant signal fluctuations or decays while being strobed. For quantitative spectroscopic or fluorescence imaging, it may be necessary to refine LED spectra using appropriate band-pass filters.

3.5 Camera specific parameters

To maximize the flexibility and versatility of SPLASSH, there exist some camera specific control parameters for which we cannot provide hard-coded, universal solutions. The most notable feature that presents this challenge is configuring "Spatial Binning" (spatial binning averages neighboring pixels into one large pixel for increased signal-to-noise at the cost of lower spatial resolution). Individual Firewire cameras natively support different sets of IIDC 1394 defined spatial binning modes, making it difficult to create a binning interface that could offer universal control over all IEEE 1394 cameras. For full SPLASSH functionality, it is therefore necessary to configure specific C pre-processor macros in the SPLASSH source code to reflect the spatial binning modes offered by the chosen camera. Within the SPLASSH code, these pre-processor macros are clearly defined at the top of the code and are set by default to manage the Pike F-032B from Allied Vision Technologies. These macros can easily be changed to conform to the specifications of different cameras, which are typically provided in the camera's manual. Each process and function in SPLASSH related to spatial binning employs these macros instead of hard-coded values, such that modifying the value of the macro once will carry through to the rest of the program. Alternatively, if spatial binning functionality is not necessary, this panel can be left alone with no modifications at all.

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010

A second feature which suffers from this same hurdle is the Exposure Time/Frame rate setting. Camera clock speeds and initial offset times are not standard across cameras, which means the calculations of frame rate based on the IEEE 1394 'Shutter' attribute will not necessarily be correct across all cameras. Again, there are clearly defined pre-processor macros in place to resolve this problem, and the defaults are set for the Pike F-032B. It is important to note that modifying these macros is a simple task and requires minimal knowledge of C or the LabWindows/CVI environment. As long as the correct parameters for the user's camera of choice are known, the macros can be changed in a matter of minutes by simply changing the value of the pre-processor macros which are located at the top of the SPLASSH source code. The advantage of this approach is a system which is versatile enough to run a variety of IEEE 1394 cameras and perform a wide range of different research protocols with minimal modifications required of the source code to accommodate different Firewire cameras.

4. Conclusion

We have provided a comprehensive description of our SPLASSH software for control of high-speed multispectral in-vivo optical imaging systems. We have explained details of implementation that will be valuable both to users of SPLASSH, and those developing similar software for camera-based in-vivo optical imaging. The modular design of SPLASSH makes it compatible with a wide variety of Firewire cameras and light sources, providing a versatile platform for a large number of research applications. SPLASSH is also capable of providing high-speed, multispectral illumination when implemented with a programmable microcontroller and modulatable light sources. We have made SPLASSH available for free download as open-source code.

The software package described here built upon earlier implementations which were developed using the Matlab Image Acquisition Toolbox. We transitioned to the LabWindows/CVI Integrated Development Environment (IDE) since programs compiled from C code are better able to handle the large amount of data generated during high-speed image acquisition. LabWindows/CVI provided this increase in system performance while maintaining a user-friendly development environment and intuitive programming. In addition, the image acquisition libraries provided by National Instruments (IMAQ, IMAQdx, NI Vision, etc.) provide access to a wider range of Firewire cameras. One area that we hope to investigate further in future implementations of SPLASSH is the use of Graphics Processing Units (GPUs), which are intrinsically optimized to perform the high speed computations necessary for modern computer display. Many major video card manufacturers and operating systems developers have recently begun to release Software Development Kits (SDKs) and Application Programming Interfaces (APIs) that allow direct programming of these previously inaccessible GPU processing and memory units. This development makes the GPU's massively parallel processors available for the first time [20–23]. While GPUs could not perform all of the image acquisition and data streaming functions executed by SPLASSH, harnessing their power could allow much improved real-time image processing, analysis, and online display of data. Thanks to the modular nature of the SPLASSH software, the necessary GPU access code could be easily added to enhance such post-processing functions.

Acknowledgements

We would like to thank Andrew Radosevich, Keith Yeager, and Richard Levenson for their contributions to this work. Funding was provided by the following National Institutes of Health (NIH) grants: (NINDS) R21NS053684 and R01NS063226, (NCI) U54CA126513, the Human Frontier Science Program, and The Wallace Coulter Foundation. M. Bouchard was supported by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program and the National Science Foundation (NSF) Graduate Research Fellowship Program. * Ryan Sun and Matthew B. Bouchard contributed equally to this work.

#129234 - $15.00 USD Received 1 Jun 2010; revised 28 Jul 2010; accepted 28 Jul 2010; published 2 Aug 2010