Computational Stereo Camera System with Programmable Control Loop

Simon Heinzle[1] Pierre Greisen[1,2] David Gallup[3] Christine Chen[1] Daniel Saner[2]
Aljoscha Smolic[1] Andreas Burg[2] Wojciech Matusik[1] Markus Gross[1,2]

[1]Disney Research Zurich [2]ETH Zurich [3]University of North Carolina

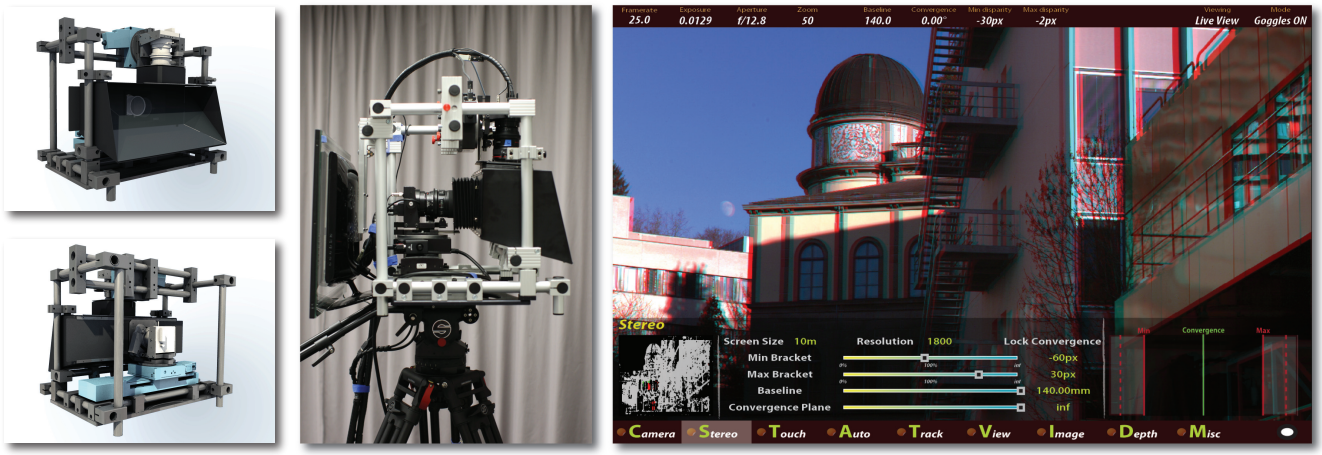

Figure 1: *Our custom beam-splitter stereo-camera design is comprised of motorized lenses, interaxial distance and convergence. A programmable high performance computational unit controls the motors. User input is performed using a stereoscopic touch screen.*

Abstract

Stereoscopic 3D has gained significant importance in the entertainment industry. However, production of high quality stereoscopic content is still a challenging art that requires mastering the complex interplay of human perception, 3D display properties, and artistic intent. In this paper, we present a computational stereo camera system that closes the control loop from capture and analysis to automatic adjustment of physical parameters. Intuitive interaction metaphors are developed that replace cumbersome handling of rig parameters using a touch screen interface with 3D visualization. Our system is designed to make stereoscopic 3D production as easy, intuitive, flexible, and reliable as possible. Captured signals are processed and analyzed in real-time on a stream processor. Stereoscopy and user settings define programmable control functionalities, which are executed in real-time on a control processor. Computational power and flexibility is enabled by a dedicated software and hardware architecture. We show that even traditionally difficult shots can be easily captured using our system.

CR Categories: I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Digital Cameras

Keywords: stereoscopy, camera system, programmable

1 Introduction

The entertainment industry is steadily moving towards stereoscopic 3D (S3D) movie production, and the number of movie titles released in S3D is continuously increasing. The production of stereoscopic movies, however, is more demanding than traditional movies, as S3D relies on a sensitive illusion created by projecting two different images to the viewer's eyes. It therefore requires proper attention to achieve a pleasant depth experience. Any imperfections, especially when accumulated over time, can cause wrong depth perception and adverse effects such as eye strain, fatigue, or even motion sickness. The main difficulty of S3D is the complex interplay of human perception, 3D display properties, and content composition. The last one of these especially represents the artistic intent to use depth as element of storytelling, which often stands in contrast to problems that can arise due to inconsistent depth cues. From a production perspective, this forms a highly complex and non-trivial problem for content creation, which has to satisfy all these technical, perceptual, and artistic aspects.

Unfortunately, shooting high-quality stereoscopic live video content remains an art that has been mastered only by a small group of individuals. More specifically, the difficulty arises from the fact that in addition to setting traditional camera parameters (such as zoom, shutter speed, aperture, and focus), camera interaxial distance and convergence have to be set correctly to create the intended depth effect. Adjusting all these parameters for complex dynamically changing scenes poses additional challenges. Furthermore, scene cuts and shot framing have to be handled appropriately in order to provide a perceptually pleasing experience. These problems become even more pronounced for live broadcast of stereo content, such as in sports applications. Capturing high-quality stereo 3D footage therefore requires very sophisticated equipment along with the craftsmanship of an experienced stereographer all of which makes the S3D production inherently difficult and expensive. The cost for S3D movie productions is estimated 10%-25% higher than for traditional productions [Mendiburu 2008].

We propose a *computational stereo camera system* that features a

closed control loop from analysis to automatic adjustments of the physical camera and rig properties. Our freely programmable architecture comprises a high-performance *computational unit* that analyzes the scene in real-time (e.g., by computing 3D structure or by tracking scene elements) and that implements knowledge from stereography to capture quality S3D video in our control loop algorithms. Since stereography is still a widely open field with a continuously evolving conception of S3D cinematography, we designed our camera architecture as a freely *reprogrammable* set of processing units. This enables us to utilize different algorithms for different scenes, shots, or artistic intentions. In addition, we support scripting of complex operations to develop and optimize shots within the actual movie production. Thus, some of the post-production is shifted back into the production cycle. In a live broadcast scenario scripts may be predefined and executed on demand.

For efficient camera operation, we devise a set of interaction metaphors that abstract the actual camera rig operations into intuitive gestures. The operator controls the camera using a multi-touch stereoscopic user interface. In addition, the interface enables monitoring the S3D content as well as the related stereo parameters instantly. In order to achieve real-time performance, we implemented our custom computational architecture combining FPGA, GPU, and CPU processing close to the sensor to achieve a low latency control loop. To summarize, the contributions of our paper are as follows:

- A computational stereo camera system for stereography and video analysis with a closed control loop model for automatic adjustment of the stereo and camera parameters,
- A programming environment for the computational unit of our stereoscopic camera rig for scripting of complex shots,
- A re-programmable control unit for adapting rig parameters to different shots, scenes, and user preferences,
- A multi-touch stereoscopic user interface including scene preview and intuitive interaction metaphors,
- An advanced system architecture combining FPGA, GPU, and CPU processing to provide a high-performance platform for full HD (1920x1080) video processing in real-time.

2 Related Work

Stereography. Production of high-quality S3D is a difficult art that has to consider and satisfy technical, perceptual, and artistic aspects at the same time [Mendiburu 2008]. In general, the displayed action should remain in a so called stereoscopic comfort zone related to the display [Howard and Rogers 2002; Hoffman et al. 2008; Jones et al. 2001; Sun and Holliman 2009]. Given a certain 3D scene, a stereographer adjusts the interaxial distance and convergence of the cameras to achieve this goal. There are software calculators [Maier 2010] that can compute parameters for a stereo camera-display combination off-line. Our system implements a re-programmable computational model for these stereographic rules that can be automatically executed and adjusted in real-time.

Commercial stereo rigs. Despite the importance of S3D productions, academic research on related camera systems is limited. However, industrial development in this area has recently gained significant momentum resulting in a variety of sophisticated solutions. Among those, the products developed by 3ality Digital [2011] are closest to our approach. 3ality provides fully motorized beam-splitter rigs, which allow accurate interactive control over all physical parameters including interaxial, convergence, and zoom. Very recently, 3ality presented a closed loop solution for automated camera control: their system is able to automatically adjust convergence and interaxial based on disparity analysis. To our knowledge, the system does not provide time-varying control algorithms, the possibility to script shots, and to extend the underlying control framework. Furthermore, no intuitive touch-based interaction metaphors are provided.

Stereoscopic scene analysis. A second group of products for stereo 3D production are software systems that analyze stereoscopic properties of the video stream in order to correct mismatches (color, key-stoning, vertical disparity). These systems often also include disparity estimation for visualization and monitoring. Sony [2011] provides a broadcasting box with basic video analysis and automatic correction functionality; however, their system requires the physical rig parameters to be tuned manually based on computed recommendations. Masaoka and colleagues [2006] present a research prototype of a stereoscopic system that computes depth of scene elements. The system can predict the quality of the S3D video based on scene layout and desired display parameters. The STereoscopic ANalyzer (STAN) is another similar system that computes disparities automatically to estimate misalignments and scene properties. Based on visual feedback, the user then adjusts the camera parameters manually [Zilly et al. 2009; Zilly et al. 2010a; Zilly et al. 2010b]. Mueller et al. [2008] describe an interactive WYSIWYG stereoscopic editing system that additionally uses visual markers as depth references for pre-planning live action 3D shots. Finally, Koppal and colleagues [2011] build a viewer-centric system that performs scene analysis and provides tools for shot planning. Although these systems allow for some basic correction and optimization of the stereo content, they only operate on the output of the camera system and therefore do not close the loop to control the rig directly.

Post-production. Some systems allow changing depth composition of S3D content after capture. Kawai et al. [2002] present an editing system that allows basic alignment and checks. Wang and Sawchuk [2008] describe a framework to warp and remap disparities to manipulate stereoscopic image pairs. Ocula [The Foundry 2010], an off-line post-production plug-in for Nuke, computes disparity maps to virtually change the interaxial distance. The non-linear disparity mapping algorithms described by Lang et al. [2010] can operate in real-time on the captured signal to correct errors and adjust depth composition. These systems are complementary to our system and can be used in sequence.

Multi-view cameras. The vast amount of research papers on multi-view or multi-lens camera systems mainly focus on the optimal capturing of data to perform operations like view synthesis or 3D reconstruction. None of these papers are focused on optimal stereo 3D production directly, implementing stereography in a computational model, and providing a corresponding programmable hardware-software solution. In that sense our work is unique, but shares a lot of basic vision components with these other systems such as color correction, rectification, or disparity estimation.

Programmable cameras. While there is a clear need for automated solutions, the full understanding on how to shoot the best S3D content is still evolving and not yet fully established. Furthermore, stereoscopy remains an art and therefore requires art directability. One fundamental property of our camera is its programmability. The most recent programmable framework for photographic cameras has been presented by Adams et al. [2010]. However, the platform does not ensure a constant flow of video data and is not able to process high-definition video in real-time. Furthermore, it is not designed for stereoscopic video streams.

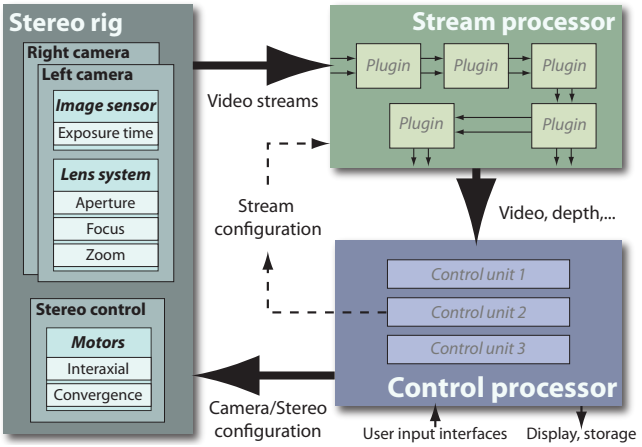

Figure 2: *The camera processing architecture is comprised of two cameras with motorized lens systems. The stereo parameters, interaxial distance and convergence, are motorized as well. Video data is sent to the stream processor, a modular plug-in-architecture. Multiple programmable sequential control programs then perform camera reconfiguration based on the incoming streams.*

3 Architecture Overview

Our architecture is composed of a motorized stereo camera set-up that is accompanied by a powerful processing architecture, as shown in Figure 2. The processing architecture is conceptually divided into stream processing and control processing. The stream processor receives the video streams from the cameras and performs the image processing and analysis for each frame. The control processor analyzes these results and evaluates control algorithms to re-adjust the camera parameters. To account for artistic control or changing requirements, both processors can be reprogrammed easily. In the following paragraphs we will describe all of the components in detail.

Motorized camera set-up. The system is designed for two cameras aligned along a common baseline. The convergence angle as well as the interaxial distance between the cameras are motorized. Furthermore, the optics of the system (aperture, focus, and zoom) are motorized as well. An illustration of the camera set-up is given in Figure 1. Changes to the camera parameters can be issued by the control processor in a precisely timed manner. This event system is described in more detail in Section 6.3.

Stream processor. The stream processor receives the stereo video and processes it to generate additional streams of data such as depth maps. More specifically, the video streams are distributed to a collection of virtual processing units that can be connected and configured arbitrarily. Dynamic reconfiguration is achieved by providing a modular plug-in architecture: all units share a common interface and any output stream can be connected to any input stream provided that the respective formats match. Furthermore, new virtual units can be easily created and added to extend the system. One example of a stream processor configuration contains Bayer-demosaicing, color correction, and disparity calculations. More details on the stream processor can be found in Section 6.1.

Control processor for real-time camera control. The video and additional data streams are then analyzed using control algorithms, presented in Section 4. Our controller changes camera parameters based on the current disparities and issues new events to the respective camera motors and the stream processing configuration. Users can tune and extend our controller or write completely different

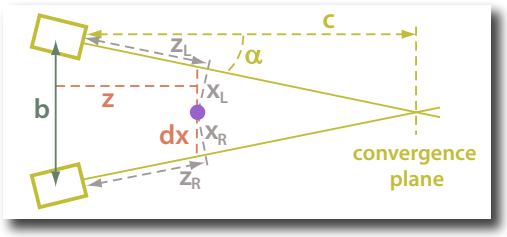

Figure 3: *Computation of screen space disparities.*

controllers to fit particular applications. Our implemented control algorithms are covered in Sections 4 and 5, the control processor is described in Section 6.2.

User interface and display. We propose to use a stereoscopic multi-touch interface that allows to toggle between both stereoscopic views and monoscopic previews, to control the camera parameters, and to adjust control algorithms using simple, intuitive interaction metaphors. More details on the user interface are presented in Section 5.

System performance. Dual, high resolution video streams at movie frame rates demand high data bandwidth as well as computational power. To process the video streams in real-time, we implement the architecture on a high-performance heterogeneous system composed of FPGA, GPU, and CPU components. High-level control algorithms are mainly computed on the CPU, while stream processing tasks are carried out on the FPGA and GPU. In addition to the high computational power, the interconnections between cameras and computational system are designed for low latency feedback. Our software and hardware implementation is described in more detail in Section 7.

4 Closing the Loop: Real-Time Camera Control

Satisfying the S3D comfort zone constraints in addition to setting traditional camera parameters requires a lot of manual tuning from the rig operator. To alleviate the currently cumbersome S3D acquisition process, we provide an approach that relieves the operator from manually setting all parameters. Our system provides a real-time controller capable of setting all (or a subset) of the camera parameters for each scene. Closing the loop between the camera output and the camera settings enables more automated acquisition and considerably simplifies the movie production process – the effort spent on time-consuming on-set adjustments and post-processing steps is minimized. Moreover, shots that are difficult to capture with traditional rigs can be handled much more easily. For instance, dynamic scenes that require simultaneous focus and convergence adjustments, typically very hard to achieve with a system without the control loop, can be captured using our setup.

We show the control loop in Figure 2: the stream processor extracts information from the scene, the control processor deduces appropriate actions from the gathered information and feeds it back to the camera system. The control variables are the stereo settings and the traditional camera settings. The feedback variables include, for example, screen disparities or object positions. Since most of the stereo limitations can be deduced from disparity values, we confine our discussion to disparity based control. Other control structures are easily realizable, as discussed later in Section 6.

4.1 Camera Parameters and Screen Disparities

The screen disparity of a given point in the scene refers to the distance between the two corresponding points in the images recorded

by the left and the right camera. The disparity is often the most important parameter for S3D depth perception and it is related to most comfort-zone constraints. We therefore consider it as the central parameter in the control loop. In the following we will show how to compute screen disparities with camera configuration parameters and scene depth information.

Our formulation of screen disparities assumes rectified and radially undistorted images as input. More specifically, we assume that both cameras have the same intrinsic parameters, the optical axes are coplanar, and the angles between the two optical axes to the baseline are equal up to orientation. For focal length f, the screen disparity d of an object can be described as:

$$d = -f\left(\frac{x_L}{z_L} - \frac{x_R}{z_R}\right), \tag{1}$$

where z_L, z_R are the depths of the object in left and right camera coordinates, and x_L, x_R are the signed distances from the two principal axes to the object (see Figure 3). For small convergence rotation angles α, the depths of the object can be approximated as $z_L \approx z_R \approx z$, where z is the distance from the object to the baseline. With the approximation $x_L - x_R \approx d_x$, the disparity can be expressed as

$$d \approx -f\frac{d_x}{z} = -f\left(\frac{b}{z} - 2\tan\alpha\right) = -f\left(\frac{b}{z} - \frac{b}{c}\right), \tag{2}$$

where b denotes the interaxial distance, α the convergence rotation angle, and c the depth of the convergence plane

$$c = \frac{b}{2\tan\alpha} \tag{3}$$

at which all objects yield a disparity of 0 pixels. Given the current camera parameters and the screen disparity d, the depth z of an object can be computed according to

$$z = \frac{bf}{-d + 2f\tan\alpha} = \frac{bf}{-d + fb/c}. \tag{4}$$

4.2 Disparity-based Camera Control

In order to control the parameters of the camera system to shoot visually pleasing 3D videos, we relate the disparities d for a given camera setting f, b, and α to the new disparities obtained with a modified setting f', b', and α':

$$d' = -f'\left(\frac{b'(-d + fb/c)}{bf} - \frac{b'}{c'}\right). \tag{5}$$

Using the convergence depth c as a more intuitive parameter instead of α, the following adjustment strategies follow directly from Equation 5.

Adjusting interaxial distance. In the simplest case we would like to respect the comfort zone constraints by adjusting the interaxial distance b while keeping the convergence plane fixed. More specifically, we would like to find the maximum interaxial distance b' (given fixed f and c) for which the screen space disparities do not exceed user-defined comfort zone limits $[d'_{min}, d'_{max}]$. If the current screen disparities lie in the range of $[d_{min}, d_{max}]$, the maximum allowed interaxial distance b' can be computed as

$$b' = \max\left(\min\left(\frac{d'_{min}}{d_{min}}, \frac{d'_{max}}{d_{max}}\right) b, 0\right). \tag{6}$$

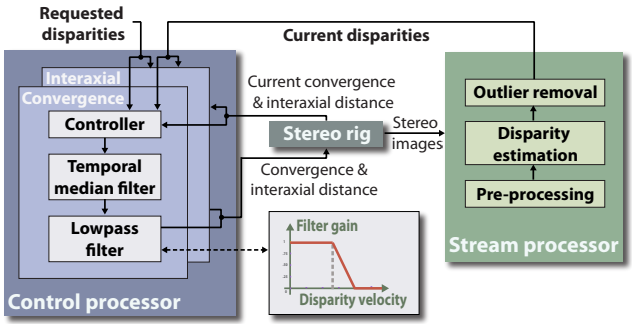

Figure 4: *Disparity based control loop. The controllers provide the next stereo settings based on the current settings and user defined inputs. A median filter reduces random, erroneous disparity jumps. A low-pass filter allows to tune the interaxial distance and convergence changes between very inert or very responsive. Controller and filters can be replaced to suit the application requirements.*

Adjusting interaxial distance and convergence plane. To fully utilize the target disparity range $[d'_{min}, d'_{max}]$ for a current disparity range $[d_{min}, d_{max}]$ we need to adjust both the interaxial distance and the convergence plane according to

$$b' = \frac{(d'_{max} - d'_{min})}{d_{max} - d_{min}} b, \tag{7}$$

$$c' = \frac{(d'_{max} - d'_{min}) bcf}{(d'_{min} d_{max} - d'_{max} d_{min})c + (d'_{max} - d'_{min})bf}. \tag{8}$$

4.3 Time-varying Changes and Control

So far we have discussed a controller that adapts interaxial distance and convergence for one specific moment in time. Now we consider how to extend this controller to time-varying scenes.

Median and low-pass filtering. Equations 7 and 8 can be used to directly control the interaxial distance and convergence plane. However, the controllers then immediately react to changes in disparities which makes them highly sensitive to errors in the disparity estimation. Moreover, to get smooth transitions, often slower reaction times are desired. We employ two filters to avoid sudden changes and to make our controllers tunable: first, a temporal median filter removes outliers in the controller output; then, a low-pass filter removes the remaining high-frequency components. A high median filter value makes the controller more robust against erroneous jumps in the disparities, but also increases the latency. The cut-off frequency of the low-pass filter determines the response time of the controller: a low cut-off frequency results in a very inert system, a high value results in a very responsive system. This approach enables to choose a tradeoff between disparity errors and transition errors: fast transitions may quickly compensate for spatial inconsistencies such as comfort zone violations, but also induce rapid temporal depth transitions which often lead to unpleasant viewer experiences. On the other hand, smooth transitions alleviate temporal depth jumps but fast appearing spatial errors remain visible for a longer time.

Alternative controllers and filters. In contrast to our filtered direct controller, classical feedback controllers such as proportional-integral-derivative (PID) controllers have no actual knowledge of the relation between feedback value and control signal. Although very robust against noise and model inaccuracies, PID controllers lack the ability to react fast, especially when the sampling rate is limited. In applications where the controller should anticipate the scene depth variations, a prediction filter could be plugged-in, such as extended Kalman filters or particle filters. A classical Kalman

filter is not suitable because the disparity noise is not necessarily Gaussian distributed (sporadic high outliers) and the model equations are non-linear.

Programmable control. Stereoscopic content creation has no unique solution due to varying scenes, applications, and user preferences. Moreover, for certain applications, it is useful to couple the stereo parameters to the lens and camera settings. Thus, the actual control behavior is dependent on the application. To combine flexibility with ease of use, we provide a programmable control framework that allows for defining arbitrary control functions as described in Section 6. We can specify a look-up table or a function that relates control parameters. In particular, we could define filter parameters versus time or as a function of disparities or disparity velocities. Also, instead of fixed disparity range requirements, we could define a function of user requirements and current scene information (e.g., depth histogram). Figure 4 provides an example of our controller together with programmable extensions.

4.4 Beyond Interaxial and Convergence Control

In the following, we discuss potential extensions to the disparity-based automatic interaxial distance and convergence plane controllers, and show how they would easily fit into our setup.

Depth of field. A different approach for handling disparities outside the comfort zone is to blur the image in the regions that have too large disparities. The blurring is obtained by narrowing the depth of field and focusing on the object or region within the comfort zone. Therefore, the controller sets aperture to achieve the desired depth of field, and then sets interaxial distance and convergence plane accordingly. The targeted disparity range could for instance be defined as a function of depth-of-field in a programmable controller.

Frame violations. Frame violations occur when an object with negative disparity (in front of the screen) intersects with the left or right image boundary. The so-called framing effect causes unpleasant stereoscopy due to two conflicting depth cues. Framing can be relieved by removing one of the two views in the vicinity of the left and right image boundaries (floating windows [Mendiburu 2008]). Frame violations could be analyzed by detecting large patches with negative disparities at the image boundaries and could be compensated by our system for e.g. live broadcasting applications.

Viewer-centric approaches. Viewer-centric approaches for stereoscopy such as [Masaoka et al. 2006] [Koppal et al. 2011] usually consider more variables in addition to the disparities captured and camera parameters used during filming. While screen size, distance to the viewer, or the human inter-ocular distance affect the possible comfort zone greatly, all related control parameters directly result from the measured disparities and camera parameters. Our framework could be directly extended to take viewer-centric variables into account as well.

5 Interactive Control

While the control algorithms presented in the previous section can be used to limit the disparity range automatically, the user often wants to be kept in the loop to account for artistic control. In addition to letting the user 'turn several knobs', we employ a control metaphor that is based on direct selection: using a multi-touch interface, the user can select points on the screen to set various parameters directly tied to scene content.

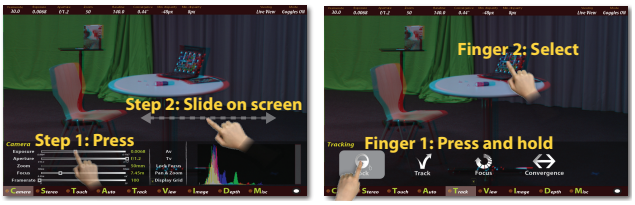

Figure 5: *Interactive control examples using our multi-touch UI.*

This selection metaphor then allows for intuitive parameter changes by selecting the objects of interest. We implemented several interactive controls using this scheme: refocusing, convergence based on touch, object tracking for follow focus and convergence, and intuitive depth of field selection.

5.1 Interactive control applications

Touch-based refocusing and convergence. Instead of calculating distances, the user can set focus and convergence plane onto a selected object. For the point of interest, window based matching in a region around the selected point is performed. The matching then returns the best reliable disparity patch which is used to calculate the depth of the object. With the depth of the object, the respective convergence plane and focus distance according to Equation 4 can be evaluated.

Tracking. Follow focus and follow convergence are traditionally hard problems that usually require highly trained operators to be performed well. In addition to touch-based refocusing/convergence, we therefore incorporate a template tracker [Babenko et al. 2009] into our framework. Using the same strategy to calculate the disparities as mentioned in the previous paragraph, our system can perform follow-focus and/or follow-convergence of a tracked object, an automatic operation that would not be possible without the computational control loop.

5.2 User Interface

Our multi-touch user interface (UI) displays the real time video stream in different viewing modes (red-cyan, disparity map, and shutter glass S3D) as well as associated scene information and camera parameters. Multiple menus allow for setting parameters or entering different operation modes. We propose the following gestures to enhance human-camera interaction: traditional *clicks*, *virtual sliders*, and *shift-button modes*. Users can perform standard interface operations with one finger *clicks* to toggle control buttons, open or close sub-menus, and drag slider thumbs.

Slider control can be tedious and inaccurate when using multi-touch displays, unless huge sliders are used. To support efficient control of smaller sliders more accurately we introduce *virtual sliders*. Users can click on a slider with one finger and subsequently use the whole screen space as invisible slider bar using a second finger. The user interface scales the finger movement and transfers it to relative shifts of the slider thumb. In this way users can operate in a larger and more comfortable space to control the slider value in a much finer scale.

In addition, we allow multi-finger touches to be used in a shift-button mode. This mode resembles the function of a shift key on a normal keyboard. While keeping the first finger pressed on a shift button, the second finger can be used to select screen objects e.g., to compute the convergence plane. Furthermore, the second finger can be used to perform consecutive actions as long as the first finger rests on the modifier button. We illustrate the user interface and our input gestures in Figure 5.

6 Programmable Control Implementation

Stereoscopic video production often employs a wide variety of different styles depending on the specific artistic intent. While some control algorithms, such as maintaining the comfort zone, play an important role in any S3D production, artistic control algorithms might change depending on the current shot. To accommodate this need for extensibility, we propose a programmable control architecture that allows implementing new control loop algorithms in addition to the algorithms presented in the previous two sections.

The control algorithms are conceptually decoupled from stream processing: while the control processor closes the loop by reconfiguring the cameras, the stream processor is responsible for generating the data needed to compute the control results. Furthermore, the control processor can issue events to the camera and motor controllers.

6.1 Stream Processor

The stream processor assembles available stream plug-ins to form a virtual device rack. The plug-ins are based on a simple interface to interchange video frame data and each plug-in is able to perform a specific processing step. Then, multiple plug-ins can be connected at run-time to form different types of processing pipelines. In order to facilitate these tasks, each plug-in defines its own set of input and output connectors. These connectors are associated with an image buffer which is annotated with dimensions and format. Furthermore, each plug-in is able to report its description and its list of functions (e.g., by invoking a run-time `help()` function). Any reported function can be called using the `runCommand("...")` function, without knowing the exact signatures at compile-time. Plug-in parameters can be set and retrieved using similar commands.

A central stream manager creates, configures, and executes the processing pipelines; it represents the virtual device rack and manages the assembly of virtual devices. The stream manager searches predefined locations for available plug-ins, creates a plug-in catalog and a list of associated methods for each plug-in. Based on the available plug-in catalog, a user program can define a desired pipeline by instantiating plug-ins and by specifying their interconnections. The central stream manager then 'compiles' the pipeline: in a first step a directed graph of all connected plug-ins is constructed to derive a processing order. Furthermore, illegal connections such as multiple sources connected to one single input connector as well as cyclic dependencies are detected and reported. In the next step, buffer formats for all interface connections are propagated to ensure that all data formats are correct.

At run-time, the stream manager issues processing requests in the correct order and manages data movement between individual plug-ins. This is especially important when using device-dependent languages such as OpenCL or CUDA, in order to avoid unnecessary transfers.

Using this concept, the base application does not depend on the available plug-ins, and it can be easily reconfigured during run-time to accommodate a specific scene being shot. Furthermore, this programming model facilitates an easy creation of arbitrary pipelines. More importantly, third party developers can adhere to this simple interface and provide additional plug-ins to extend the functionality of the current camera system.

6.2 Control Processor

Conceptually, the stream processor employs a uni-directional data flow model – it is only allowed to process the incoming stream.

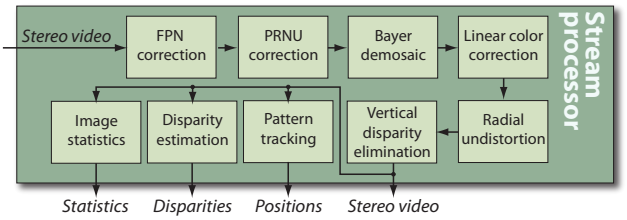

Figure 6: *Implemented stream processor configuration. After basic color processing, the stream is radially undistorted and vertical disparities are eliminated. Disparity estimation and pattern tracking are performed for subsequent control algorithms.*

The control processor is more general: it is able to reconfigure all camera settings as well as the stream processor.

The control units adhere to a simple interface that is very similar to the interface of the stream processing plug-ins. Control units can request buffer handles and parameter settings from the stream manager. Based on the available buffer information, a control unit evaluates its algorithms and can decide how to reconfigure the stream processing and the camera. The stream reconfiguration is directly performed using the stream manager. Camera reconfiguration is performed using *events* sent to an event queue. These events are described in the next section.

6.3 Event Scripting

The control processor issues *events* to set camera, lens, and stereo parameters in a precise and scriptable way. All events can be set either to a point relative to the current clock or relative to a future exposure starting time. In addition to the start time stamp, the control program can also prescribe a certain duration for the event to e.g. increase the interaxial very slowly. An example timed event that controls the camera aperture can be written as follows:

```
ApertureEvent *ae = new ApertureEvent();
ae->setAperture(1.4);    // set aperture to f/1.4
ae->setStartTime(0.03);  // start in 0.03s
ae->setDuration(0);      // as fast as possible
EventQueue::addEvent(ae);
```

The absolute-timed events can be helpful for immediate actions or to execute a pre-scripted sequence of commands. Alternatively, events can be timed relatively to an exposure event. Such events are useful to keep the cameras still during the exposure time and to change the parameters only during the read-out phase, if the physical changes are fast enough.

After a new event has been configured it is entered into the *event queue*. The event queue manages the timing and delegation of events to the appropriate hardware controllers, prohibiting direct control by the control programs. Events can be attributed with a priority, and the event queue resolves potential conflicts according to the assigned priorities. If the priority of a new event is smaller or equal to the priority of the currently executing event, the prior event will be canceled and replaced by the new one. We currently support event settings for focus, aperture, interaxial distance, convergence, zoom, exposure, and frame rate. However, other events can be implemented if the appropriate hardware controllers are present. Note that this concept is very similar to the *Device Action* mechanism of the FrankenCamera framework [Adams et al. 2010], but targeted for streaming video. Using events, camera settings for whole scenes can be pre-scripted and executed on demand.

6.4 Configuration Example

We discuss an example of a basic software configuration that employs both stream and control processor in the following. To realize the control algorithms from Section 4.2, we implemented the stream processing configuration as depicted in Figure 6. In the stream processing step, fixed pattern noise (FPN) reduction, pixel response non-uniformity (PRNU) correction, Bayer demosaicing, linear and non-linear color correction, and radial undistortion are performed as low-level image manipulation algorithms. Furthermore, the two images are rectified to avoid vertical disparities. Another plug-in estimates stereo disparities. Furthermore, template matching to estimate local screen disparities and perform object tracking is executed. Finally, the parameter controller analyzes the information computed by stream processing, evaluates the control algorithms presented in Section 4.2, and reconfigures the cameras on the fly.

7 Implementation

The proposed computational system is heterogeneous. It is implemented using FPGAs, a GPU, and a CPU. The stream processing plug-ins are distributed among different hardware units; the control algorithms mostly run on the CPU. This section presents details on the lower and higher level stream processing plug-ins, the implementation of the system as well as the specific hardware architecture.

7.1 Low-level Stream Processing Plug-ins

Image pre-processing. Low-level processing mainly encompasses the pre-processing steps from traditional image processing [Ramanath et al. 2005]. The synchronization of the two cameras is handled by simultaneously releasing a trigger pulse from the FPGA to both camera sensors with a configurable frame rate. As the employed camera does not correct for fixed pattern noise (FPN) and pixel response non-uniformity (PRNU), our framework lets us capture so-called black images for different exposure times and white images for the PRNU correction. Then, the correction is performed in a single plug-in on the FPGA. We employ a linear algorithm to extract the color image from the color filter array image. More specifically, we chose a linear 5x5 interpolation filter that is based on the Wiener filter [Malvar et al. 2004]. Next, the RGB values captured by the camera sensor need to be transformed to a well-defined colorimetric space such as sRGB. In order to accomplish this step, a linear transformation 3x3 matrix is estimated using a color checker chart. In addition, white balancing is performed in the same step. The color correction plug-in that executes on the FPGA applies this matrix transformation to the RGB values. Moreover, a gain correction plug-in can be added to enhance contrast.

Color matching and rectification. In addition to transforming the colors of the two cameras separately into calibrated color spaces, non-linear color shifts between the two cameras must be taken into account. Color matching is particularly important in mirror-rigs due to the dispersive behavior of the beam-splitter mirrors. We include programmable look-up tables (LUTs) on the FPGA that can realize arbitrary non-linear transformations on the individual color components in the RGB space or in the HSV space. One example for determining the LUTs is to separately equalize the histograms of the HSV channels of the two images. A more advanced method is described by Pitie et al. [2007]. Radial and tangential distortion correction is performed on the GPU according to Brown's distortion model [1966]. Next, a projective transformation is applied to both video streams on the GPU to rectify the images. Section 7.3 provides details on how to determine the projective transformations.

7.2 Disparity Estimation Plug-in

Our control algorithms rely heavily on robust disparity information. In order to achieve real-time performance using our FPGA-GPU-CPU architecture, we employ a local window method, similar to [Zach et al. 2004] with modifications to reduce the amount of outliers. First, the incoming images are progressively downscaled into an image pyramid. Then, starting on the lowest resolution, the algorithm returns the best match for each pixel in the first image along a line in the second image. The resulting disparity of the best match is propagated to the next higher resolution where it serves as offset to refine the match. As matching kernel we chose the normalized cross correlation (NCC) kernel in order to account for possible changes in offset and gain of the pixel intensities.

Local window based methods offer the best degree of parallelization and only require modest computational resources compared to more advanced disparity matching. However, these methods typically overextend object boundaries when using large kernel windows or low resolution images. Conversely, reducing the kernel radius or the amount of downscaling increases the amount of disparity estimation errors. For our automatic control architecture, we do not need dense per-pixel disparity values, but rather a robust histogram distribution of the disparities of a scene. Hence, we optimize our disparity estimation to yield minimal amount of outliers: the lowest resolution image in our pyramid architecture is downscaled 4 to 5 times, and matching is performed using large kernel sizes (15 to 20 pixels). These settings reduce the amount of noise greatly, but also reduce the ability to detect fine features. The disparity matching is performed from left-to-right and right-to-left, and a final consistency check ensures that the same disparities are found in both directions for additional outlier reduction. Moreover, we use a 5x5 spatial median filter and thresholding of the NCC matching costs to remove remaining outliers. The disparity estimation is implemented as a CUDA kernel. The resulting disparity maps are mostly free of outliers, see Figure 8 for obtained results.

7.3 Calibration

While eliminating vertical disparities is necessary for proper human stereopsis, it is also crucial for computational stereo algorithms to work. In order to successfully work with a stereo camera rig with motorized lenses and camera positions, high-quality calibration is indispensable. That is, the relative camera positions should differ only by a horizontal offset (interaxial distance) and the camera intrinsics must be known, in particular for varying focus and zoom settings.

Camera intrinsics. Our disparity calculation uses the camera focal length as input. Instead of relying on the lens markings, we perform an initial calibration [Bouguet 2010] for each camera-lens combination to compute its focal length, distortion coefficients, and principal points. For zoom lenses, we perform the calibration for a set of different focal lengths. Similarly to Fraser et al. [2006], we interpolate these results depending on the current motor positions.

Camera extrinsics. The relative positions of the two cameras are first coarsely aligned by manually changing the camera orientations on the rig. In practice, looking at the red-cyan stereo image quickly reveals coarse vertical disparities. The fine-tuning is done digitally in the rectification step: we estimate rectifying homographies from the epipolar geometry in real-time using feature matching and

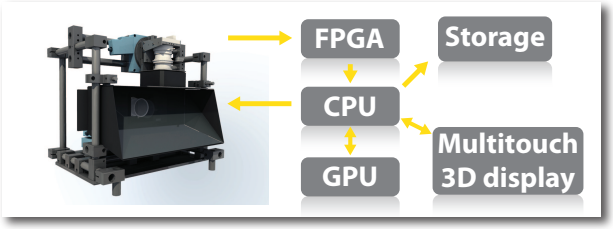

Figure 7: *Hardware implementation: The FPGA acquires the video stream and performs pre-processing. The CPU and GPU are used to implement high-level processing and control. The system includes a storage array as well as a multi-touch user interface.*

outlier removal tools. We use the approaches from Mallon et al. [Mallon and Whelan 2005] and Zilly et al. [Zilly et al. 2010a] that work well for feature-rich scenes with non-zero interaxial distance and radially undistorted image pairs.

7.4 Software Highlights

Plug-ins and scripting. The system uses shared library mechanisms to load plug-ins. A central stream manager performs the configuration and execution of plug-ins. LUA scripting is used for easy reconfigurability and extensibility, LUAbind is used to expose C++ classes to LUA. The UI can be reconfigured easily using this scripting language. Moreover, the UI can be connected to different stream and control processing plug-ins presented in Section 6.

Multi-threading. We employ multi-threading to optimize the overall system performance. The most important threads include a DMA thread for FPGA-to-host video transfer, a thread for video storage, a thread for stream processing, a control thread, and the event scheduler thread. Furthermore, all interfaces to external motors are started in separate threads in order to quickly react to events and to communicate efficiently with hardware controller boxes.

7.5 Hardware Components

We evaluate our proposed camera architecture using an experimental beam-splitter rig which allows for fine camera and mirror tilt adjustements. The system uses two Silicon Imaging SI-4000F color cameras with a resolution of 2048x2048. A Birger EF mount for Canon EF lenses is used to control focus and aperture electronically. Both cameras are mounted on high-precision rotary stages from Newport. Moreover, the right eye camera is mounted on a high-precision linear stage from Newport to control interaxial distance. The computational system uses a six-core CPU paired with an NVIDIA GTX480 GPU. We employ a PCI-Express board with ALTERA Stratix III FPGAs as a frame grabber to pre-process video streams. A hardware RAID array of eight solid-state hard drives is integrated in order to store the final video stream. A multi-touch overlay from PQLabs is attached on top of a 120Hz monitor for the user interface, and NVIDIA 3D vision shutter glasses can be used for live S3D preview. Figure 7 illustrates the system.

8 Prototype Performance and Limitations

Video preprocessing and storage. The FPGA is able to preprocess two 1080p60 video streams. However, the camera supports 30 frames per second only, and our storage system was designed for this data rate as well. The latency from camera to the display is approximately 3 frames, which is more than acceptable for the camera operator. The PCIe express throughput of the FPGA board is limited to 35 frames per second.

Disparity Estimation and Tracking. Disparity estimation and tracking are hard computer vision problems, and we employed rather simplistic approaches in favor of fast execution times. Robust feature detectors are a possible alternative approach to dense disparity estimation, but were not tested either in terms of execution speed or outlier performance.

System delay. Compared to the video pre-processing, the control loops require more complex processing to analyze the scene. Hence, the control loops are designed to operate independently of video display and recording and are performed at a lower frame rate and/or at a lower resolution. The dense disparity estimation operates on a 16x downsampled image for the coarsest level, and propagates the disparities up to a 2x downsampled image. Tracking uses a 4x downsampled image. When both algorithms are combined, the frame rate of the control loop in the current implementation reduces to 10Hz, while the frame rate of the recorded video stream is not affected.

Impact of system delay on control rate. The delay of the system dictates the update rate of the control parameters and hence should be small compared to temporal scene changes. While the current implementation does not allow fast changes, the overall performance could be significantly improved without a major redesign of the system, e.g., by implementing disparity estimation and tracking on the FPGA. However, in some situations, such as sports broadcasting using long zoom lenses, the images might change completely from frame to frame. The disparities will therefore also change completely and the automatic control algorithms will not be able to produce meaningful parameter changes. As a possible solution, additional cameras with wide angle lenses could be used to analyze the scene and subsequently to control the main cameras.

Degree of motorization. In our prototype, the misalignments of cameras and shift of optical axes during zoom can only be compensated using the digital rectification step. However, when the misalignments become more severe mechanical adjustments are necessary. Ideally, all degrees of freedom such as roll, tilt, pitch, and offset should be motorized to allow for pixel accurate camera positioning. Commercial camera rigs such as [3ality Digital 2011] often provide such motorization.

External motor control. The system is furthermore limited by the motor controls. For our hardware, the latency of any command for the linear and rotational stages is in the order of 20ms. Any motion command to the linear and rotational stages has a minimum execution time of 160ms. As a consequence, the overall execution time of the control loop is long, especially when combined with temporal disparity noise filtering.

Prototype rig. A further limitation of our system is the overall weight. In addition to a traditional camera, a processing box and multi-touch user interface is required.

8.1 Application Examples

We illustrate the potential of our system by using three examples: automatic interaxial distance and convergence plane control, touch-based refocusing and convergence, and subject tracking for follow-focus and follow-convergence shots. We refer to the accompanying video for the full results and we provide representative selected frames in the paper.

Automatic interaxial distance and convergence plane control is performed using data from the disparity estimation. Figure 8 shows results of the automatic controller. In the present implementation, the disparity estimation limits the frame rate and, in combination with the median filter to increase outlier robustness, leads to the

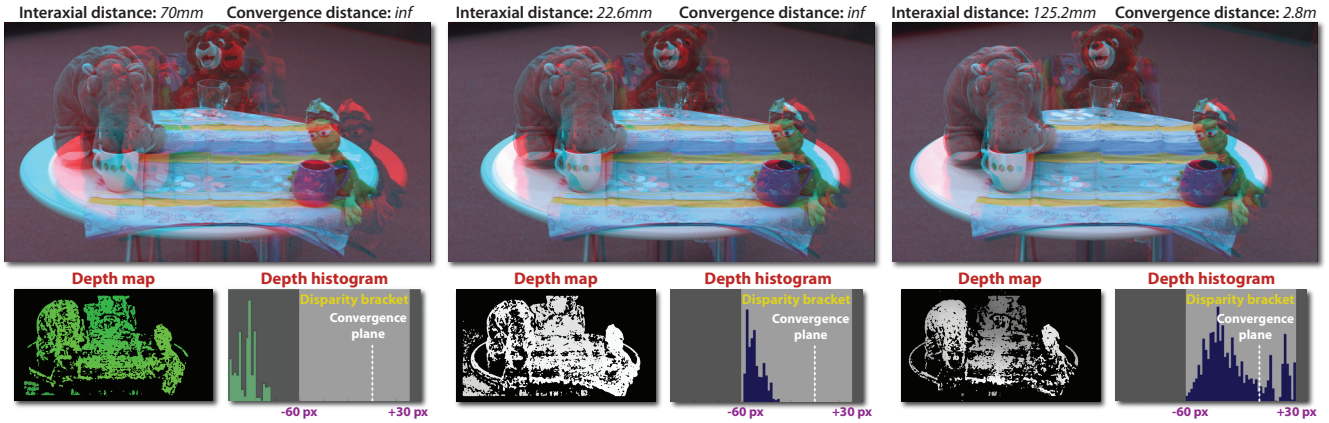

Figure 8: *Effect of different interaxial and convergence settings. The left image shows a close-up scene shot using a wide interaxial and parallel cameras, and produces huge disparities that violate the stereoscopic comfort zone. After applying the interaxial controller, the interaxial distance is reduced drastically to map the disparities into the comfort zone as shown in the center image. However, the overall depth range is not utilized fully. The image on the right shows the result after applying the interaxial distance and convergence plane controller, resulting in a more homogeneous disparity distribution due to the wide interaxial distance and near convergence setting.*

Figure 9: *Two examples of automatic control based on template tracking. Manual interaction is only required in the first frame (left column) to initialize the subject of interest. In the top row, the controller follows the convergence plane at the depth of the subject. In the bottom row, both convergence and focus are following the subject.*

latency visible in the video. For relatively static scenes or for higher frame rates the additional low-pass filter can be used to smooth out the camera-parameter adjustments.

Touch-based refocusing and convergence uses simple template matching similar to the disparity estimation to determine the distance to a selected object. To improve stability and accuracy of the object-based estimate, a larger matching radius is used. The operator can specify the duration of the transition using the event scripting to create smooth effects.

Subject tracking for follow-focus and follow-convergence works very well in practice, despite the slow update rate of the employed tracker algorithm. While the tracker can lose an object when the background is similar to the subject, it performed well for our applications. Figure 9 shows representative frames for two different sequences.

9 Conclusion

In order to address today's challenges in S3D production, we present a novel design of a computational stereo camera system, which closes the control loop from capture and analysis to automatic adjustment of physical parameters, such as interaxial and convergence. We develop intuitive interaction metaphors that automatically abstract and replace cumbersome handling of rig parameters. The main driving goal behind the design is to make S3D production for artists as intuitive and flexible as possible. In principle, the system can be combined with any motorized stereo camera rig. The architecture further is comprised of a configurable stream processor that efficiently performs video processing and analysis operations, a programmable control processor that implements control functionalities derived e.g. from best-practice rules of stereography or user input, and a user interface and display for intuitive interaction. Real-time performance and computational flexibility are enabled by the combination of FPGA, GPU, and CPU processing. Only such a design enables real-time closed loop control of physical camera parameters. The core of the control loop is a disparity-based implementation of knowledge from stereography combined with user settings, e.g. to allow for artistic depth composition of a scene. Intuitive interaction is enabled through metaphors via a touch screen with stereoscopic visualization. It allows monitoring and controlling all basic characteristics of the signals and functionalities of the system. Programmable control allows event scripting, e.g. for complex shots or for user preferences.

Our prototype system implements all basic functionalities of the concept and we demonstrate a number of compelling applications, including automatic disparity range adjustment via interaxial distance and convergence, touch-based refocusing and convergence, follow focus and convergence tracking. Such results are impossible or difficult to achieve with current systems. Our concept of a computational approach for a S3D camera system with the presented architecture and components is proven useful by these applications.

Future work will include optimization and extension of all components, including usage of other rigs and cameras, improvement of low-level image processing algorithms, optimization of the control loop implementation and parameters, design of further intuitive interaction metaphors, improvements to the user interface, as well as general improvements of the software (e.g., partitioning) and hardware. While our system automatically adjusts stereo parameters to limit disparities to a comfortable range, other more sophisticated concepts from stereography (e.g., framing violations, disparity gradients, flatness) still require manual interaction with the provided plug-ins. Such additional plug-ins are left for future work in order to achieve a fully automated stereographer. We also plan to test our prototype in professional productions and to incorporate feedback from S3D production professionals. The basic paradigm of our design, being an efficient computational vision system incorporating advanced image analysis and high-level concepts into a real-time closed control loop, easily extends to other application scenarios as well.

References

3ALITY DIGITAL, 2011. Beam splitter rigs and stereoscopic image processors. http://www.3alitydigital.com/.

ADAMS, A., TALVALA, E.-V., PARK, S. H., JACOBS, D. E., AJDIN, B., GELFAND, N., DOLSON, J., VAQUERO, D., BAEK, J., TICO, M., LENSCH, H. P. A., MATUSIK, W., PULLI, K., HOROWITZ, M., AND LEVOY, M. 2010. The Frankencamera: an experimental platform for computational photography. *ACM Transactions of Graphics 29*, 29:1–29:12.

BABENKO, B., YANG, M., AND BELONGIE, S. 2009. Visual tracking with online multiple instance learning. In *Conference on Computer Vision and Pattern Recognition*, IEEE, 983–990.

BOUGUET, J.-Y., 2010. Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

BROWN, D. C. 1966. Decentering distortion of lenses. *Photogrammatric Engineering 32*, 444–462.

FRASER, C. S., AND AL-AJLOUNI, S. 2006. Zoom-dependent camera calibration in digital close-range photogrammetry. *Photogrammetric Engineering & Remote Sensing 72*, 1017–1026.

HOFFMAN, D., GIRSHICK, A., AKELEY, K., AND BANKS, M. 2008. Vergence-accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision 8*, 3, 33.

HOWARD, I. P., AND ROGERS, B. J. 2002. *Seeing in Depth*. Oxford University Press.

JONES, G., LEE, D., HOLLIMAN, N., AND EZRA, D. 2001. Controlling perceived depth in stereoscopic images. SPIE, vol. 4297 of *Stereoscopic Displays and Virtual Reality Systems*, 42–53.

KAWAI, T., SHIBATA, T., INOUE, T., SAKAGUCHI, Y., OKABE, K., AND KUNO, Y. 2002. Development of software for editing of stereoscopic 3D movies. SPIE, vol. 4660 of *Stereoscopic Displays and Virtual Reality Systems*, 58–65.

KOPPAL, S., ZITNICK, C. L., COHEN, M., KANG, S. B., RESSLER, B., AND COLBURN, A. 2011. A viewer-centric editor for 3D movies. *IEEE Computer Graphics and Applications 31*, 1, 20 –35.

LANG, M., HORNUNG, A., WANG, O., POULAKOS, S., SMOLIC, A., AND GROSS, M. H. 2010. Nonlinear disparity mapping for stereoscopic 3D. *ACM Transactions on Graphics 29*, 4.

MAIER, F., 2010. Stereotec stereoscopic calculator. http://stereotec.com/.

MALLON, J., AND WHELAN, P. F. 2005. Projective rectification from the fundamental matrix. *Image Vision Computing 23*, 643–650.

MALVAR, H., HE, L.-W., AND CUTLER, R. 2004. High-quality linear interpolation for demosaicing of bayer-patterned color images. In *International Conference of Acoustic, Speech and Signal Processing*, IEEE.

MASAOKA, K., HANAZATO, A., EMOTO, M., YAMANOUE, H., NOJIRI, Y., AND OKANO, F. 2006. Spatial distortion prediction system for stereoscopic images. *Electronic Imaging 15*, 1, 013002:1–12.

MENDIBURU, B. 2008. *3D Movie Making - Stereoscopic Digital Cinema from Script to Screen*. Elsevier.

MUELLER, R., WARD, C., AND HUSAK, M. 2008. A systematized WYSIWYG pipeline for digital stereoscopic 3D filmmaking. SPIE, vol. 6803 of *Stereoscopic Displays and Applications*.

PITIÉ, F., KOKARAM, A., AND DAHYOT, R. 2007. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1-2, 123–137.

RAMANATH, R., SNYDER, W., YOO, Y., AND DREW, M. 2005. Color image processing pipeline. *Signal Processing Magazine, IEEE 22*, 1, 34–43.

SONY, 2011. MPE200 multi image processor and software. http://pro.sony.com/.

SUN, G., AND HOLLIMAN, N. 2009. Evaluating methods for controlling depth perception in stereoscopic cinematography. SPIE, vol. 7237 of *Stereoscopic Displays and Virtual Reality Systems*.

THE FOUNDRY, 2010. Ocula stereoscopic post-production plug-ins for Nuke. http://www.thefoundry.co.uk/.

WANG, C., AND SAWCHUK, A. A. 2008. Disparity manipulation for stereo images and video. SPIE, vol. 6803 of *Stereoscopic displays and applications*.

ZACH, C., KARNER, K., AND BISCHOF, H. 2004. Hierarchical disparity estimation with programmable 3D hardware. In *Short communications of WSCG*, 275–282.

ZILLY, F., EISERT, P., AND KAUFF, P. 2009. Real-time analysis and correction of stereoscopic HDTV sequences. In *Proceedings of Visual Media Production*.

ZILLY, F., MUELLER, M., EISERT, P., AND KAUFF, P. 2010. Joint estimation of epipolar geometry and rectification parameters using point correspondences for stereoscopic TV sequences. In *Proceedings of 3DPVT*.

ZILLY, F., MUELLER, M., EISERT, P., AND KAUFF, P. 2010. The stereoscopic analyzer - an image-based assistance tool for stereo shooting and 3D production. In *IEEE International Conference on Image Processing*.