https://doi.org/10.2352/ISSN.2470-1173.2018.05.PMII-182

Lessons from design, construction, and use of various multicameras

Henry Dietz, Clark Demaree, Paul Eberhart, Chelsea Kuball, and Jong Yeu Wu;
Department of Electrical and Computer Engineering, University of Kentucky; Lexington, Kentucky

Abstract

A multicamera, array camera, cluster camera, or "supercamera" incorporates two or more component cameras in a single system that functions as a camera with superior performance or special capabilities. Many camera arrays have been built by many organizations, yet creating an effective multicamera has not become significantly easier. This paper attempts to provide some useful insights toward simplifying the design, construction, and use of multicameras. Nine multicameras our group built for diverse purposes between 1999 and 2017 are described in some detail, including four built during Summer 2017 using some of the proposed simplifications.

Introduction

Although individual cameras have been rapidly improving in their image quality and capabilities, multicameras offer scalable ways to handle larger and more complex image capture problems. Well known examples include many stereo/plenoptic cameras[1] and 360° capture rigs, but range from the groups of cameras used to capture "bullet time"[2] (as seen in the movie *The Matrix*) to DARPA's ARGUS-IS[3] which delivers 1.8GP using 368 component cameras. This paper suggests that creating and using a special-purpose array camera can be made much simpler, faster, and cheaper, by abstracting the design process into a small number of issues and establishing a pool of standard solutions to each.

The obvious problem that the designer of a multicamera needs to address is how to coordinate the actions of the individual component cameras. However, approximate synchronization of multiple cameras is not difficult. Issues involving programmable control, physical construction and optical alignment, and calibration and image processing generally dominate. Synchronization is not difficult because there is better infrastructure supporting it. Thus, the main goal in this paper is identifying and suggesting ways to create the key bits of missing infrastructure.

The multicameras

In February 1994, our research group built the world's first Linux PC cluster supercomputer – and demonstrated that it behaved like a tightly-coupled parallel computer by using it to display various real-time simulations on a video wall. Over the following years, we scaled our clusters and video walls from a 2x2 display array with just 1,280x960 pixel resolution driven by 4 PCs to a 4x4 display array with 6,400x4,800 pixels driven by 32 PCs in 1996. Our video wall library made it possible to display images on the wall and even to interactively pan and zoom them, and we had some interesting images including a 200MP stitched image of Io (the Jovian moon as photographed by NASA), but had no way to capture live images that would show-off the video wall. Thus, we began experimenting with digital cameras.

Figure 1. *Olympus D-340R and Nikon 950, both with 185° lenses*

Even the photos we took in February 1994 of the first Linux PC cluster were captured electronically – using a video camera with a frame grabber – and we soon began experimenting with a variety of cameras directly producing digital images. Throughout the late 1990s, we performed many experiments using a variety of frame grabbers, stand-alone digital still cameras, and the parallel-port-tethered Connectix Color QuickCam 2. However, none of these cameras had resolution and other properties that could produce compelling content for viewing on a high-resolution video wall. The obvious answer was to try to use multiple cameras together as one higher-performance camera capable of doing things an individual digital camera couldn't do. In effect, we began building camera clusters to provide interesting image data to manipulate on our Linux PC cluster supercomputers.

Autonomous 360-degree capture system, 1999

By 1999, the resolution of higher-end consumer digital cameras, such as the Olympus D-340R and Nikon 950 shown in Figure 1, bumped to beyond a megapixel, making them more cost-effective than large arrays of low-end cameras. However, cost of enough of these high-end cameras to match resolution of our video walls would still have been too high – so we looked for a way to make compelling images clustering just a few cameras. Thus, we decided to build a 360° camera.

Clustering cameras requires support for some form of communication or tethered control. The Nikon Coolpix 900 allowed for RS232C tethered capture and, using the FC-E8 fisheye converter lens, it captured 185° circular images so that just two cam-

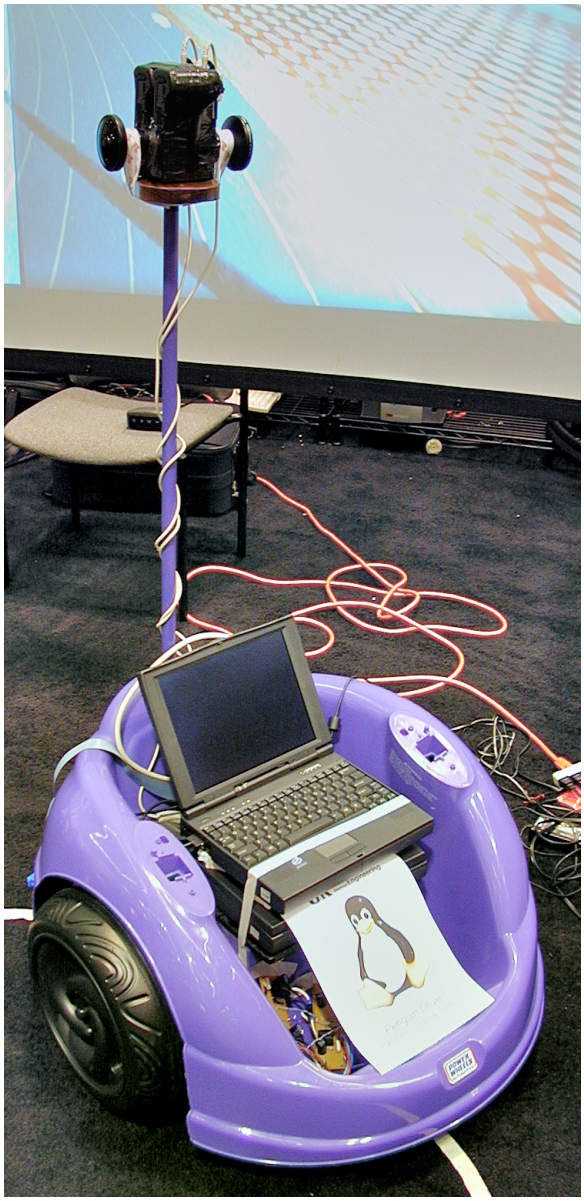

Figure 2. *Autonomous 360° capture system*

eras, mounted back-to-back, could be capable of capturing a full 360° view. Appropriate software to stitch the images was not available at that time, but we were able to write our own based on some of the early work posted by Helmut Dersch in his development of Panorama Tools[4]. Thus, we decided to build an autonomous 360° capture vehicle that could wander the exhibit floor at IEEE/ACM SC99 sending pairs of images back to a cluster supercomputer in our research exhibit via 802.11 wifi. The cluster supercomputer would then stitch the views and allow a user to pan and zoom through the 360° views on a 4-projector video wall.

Although Nikon had provided us with a 900 to test in 1998, they advised us to wait for the upcoming release of the 950, which upgraded the sensor from 1.3MP to 2MP. At a total cost of about $5000, we purchased two of the first 950s and FC-E8 adapters for them. Our initial tests with the 950s involved holding the two cameras (with FC-E8 converters) back-to-back, which failed because the minimum distance between the lenses was about 10". Allowing the lenses to be offset, by the camera bodies being held side-by-side, essentially halved that separation and produced fewer stitch errors. However, the biggest problem was that the tether support that worked well in the 900 was "temporarily" not implemented in the 950! In frustration, we bought a pair of 1.3MP Olympus D-340R with their 185° fisheye adapters; their image quality was poorer, but their RS232C tethering worked. Nikon eventually got their RS232C tether support working, so we built and demonstrated two autonomous 360° systems in our research exhibit at IEEE/ACM SC99.

The most cost-effective way to build the autonomous platforms was by modifying riding toys. Controlling steering of a fire truck was difficult and it had a wide turn radius; a Power Wheels Wild Thing (shown in Figure 2 with the Nikons mounted) was bulkier, but easier to control and it could pivot in place. The cameras on both vehicles were mounted on vertical posts bringing them up to a reasonable height. A pair of laptop computers were mounted where the child would have sat and the laptops controlled the movement of the vehicle, while performing synchronized tethered captures from the pair of cameras and uploading (via 802.11 wireless) the images to the cluster driving the video wall. Each update took about 45 seconds, primarily because it took nearly that long to transfer an image via RS232C, but the stitching and pan and zoom rendering on the PC cluster worked in real time. The autonomous control was mostly just following a path marked by white tape on the floor, but the system worked well.

As larger displays moved to being implemented by single video projectors and camera resolution quickly exceeded that of a single projector, we continued to modify and tether various cameras, but for several years did not see a need for more multicameras per se. For example, we did some work using computer-controlled mounting of a single camera to create gigapixel-scale stitched still images. We also investigated various methods for multi-spectral imaging.

FireScape

The FireScape project began in 2006 when our group was approached about the possibility of creating a cheaper alternative to a thermal imaging camera for firefighters navigating through a burning building. There had been some confusion about the NIR imaging work we had done, mistaking it for true IR imaging. However, as we investigated the problem more deeply, we quickly realized that the real need wasn't for an IR camera (which typically required dedicating one firefighter to use of the camera), but for sensor fusion that could improve situational awareness for each firefighter.

FireScape was to be a low-cost helmet-mounted aid for firefighters in burning buildings. It would combine data from a variety of sensors, including sensors for depth (using an active ultrasonic sensor), temperature (using a non-contact IR thermometer), accelerometers, electronic compass, and multiple cameras sensi-

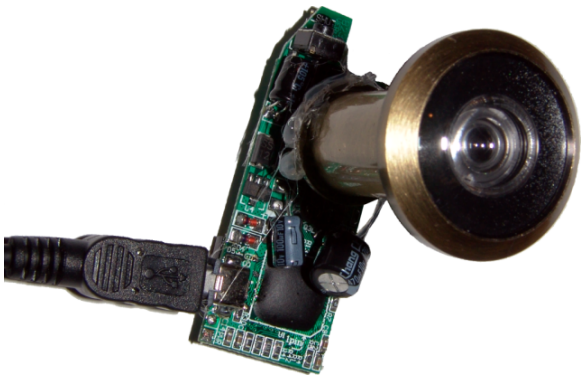

Figure 3. *FireScape Prototype Camera Module*

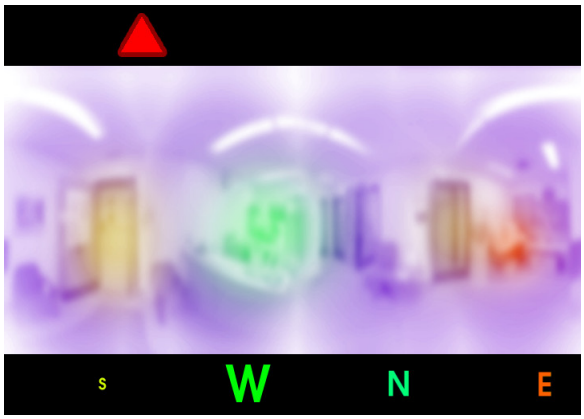

Figure 4. *FireScape Sample View*

tive to both visible and NIR light. A small display placed just outside of the viewing area of the protective goggles would allow the firefighter to easily see this information at a glance without interfering with the already narrow optical view through the goggles. Although construction of the complete FireScape system was not funded, the sensors and camera array were prototyped.

The FireScape camera array consisted of three USB cameras mounted around the helmet to provide a stitched 360° view in both visible and near-infrared. We quickly realized that prototype camera modules could be extracted from cheap USB webcams, but appropriately wide-angle lenses were not cheaply available. To obtain a 360° view around the firefighter, each camera needed to cover a view angle greater than 120°. Firefighters also often need to crawl through sections of a burning building, so it is important that the cameras still see in front of the crawling firefighter – which means we want as close to a 360x180 stitched view as possible. Worse still, there is a high probability that these lenses could be exposed to temperatures as high as the flash-over temperature; if such heat were to melt or disintegrate the lens, not only could the camera fail, but the camera itself could catch fire causing potential injury.

Our surprising prototype solution was based on the module shown in Figure 3. The camera was the single-board system extracted from a cheap USB webcam, and included its relatively narrow-angle plastic lens. However, we realized that door peepholes are essentially cheap fisheye converters that are fire rated: there are fire safety codes that require door peepholes to be able to withstand extreme exposure to fire and heat without catastrophic failure. A door peephole may fail, but they are required not to fail in a way that lets the fire penetrate behind it. Once a failed peephole has cooled, it easily can be replaced by unscrewing it and screwing in a new one.

From a control perspective, one would expect that synchronization of these USB cameras would be problematic – especially with three cameras sharing a USB controller. However, we realized that the key is not synchronization, but being able to immediately judge the quality of the image data, including temporal validity. Thus, as shown in Figure 4, sensor data are "painted" onto the display as they are read. Visible and NIR images update the gray scale value of each pixel in the image, and thermal data update the hue of each pixel. The temperature scale is mapped non-linearly, so that distinct color changes mark thresholds that are interesting to the firefighter.

Sensors may not cover the full 360° of view surrounding the firefighter. Areas that are not covered remain in the display, but are aged by reducing their contrast and/or color saturation over time. Areas that are not updated because a sensor has not passed over them – or because a sensor has failed – eventually become gray until they are updated. Areas will also age more quickly if the inertial sensor detects that the firefighter is moving and the data are outdated. Aging, rather than blanking gaps in sensor coverage, allows the firefighter to see what was in the gaps, but gives an indication of how reliable that data is.

Compass directions 'N', 'E', 'S', 'W' are shown below the main display image to help orient the firefighter. Color of each of the letters indicates the temperature of the hottest object sensed in that general direction, and the size of the letter indicates the distance to the nearest object. Above the display an arrow points in a direction the firefighter would likely be able to exit the building.

AVA (Ambient Virtual Assistant, 2008)

AVA (Ambient Virtual Assistant)[5] was a scalable wide-area surveillance and "smart space" system which was prototyped with 23 industrial firewire cameras and 40 microphones.

In the physical sense, AVA really wasn't a multicamera at all, but a collection of independent cameras scattered among rooms and hallways within one floor of a building. You can see some of the cameras in the ceiling tiles in Figure 5. The thing that made AVA logically a multicamera was the common software interface that allowed higher-level processing to implement multi-camera control and acquisition of data streams in real time.

Each of the firewire cameras had a complete compact PC associated with it. Using standard network time mechanisms, the PCs maintained an approximate notion of global time that was used to trigger and tag events. Using NTP or PTP to establish global time was only accurate to perhaps 0.01s, but that was usually sufficiently accurate for video framerates under 30FPS. Much of the complexity of the system was invested in translating requests into a generic control language and managing sharing of devices. For example, a camera could be streaming video to a workstation when a request is issued to collect temporally-synchronized streams from a set of cameras including the one

Figure 5. *Some of the ceiling-mounted cameras in AVA*

Figure 6. *Stereo Capture*

already streaming. Put simply, the entire collection of cameras was expected to provide whatever real-time data was requested by any set of higher-level computations attempting to serve as a virtual assistant: basic security monitoring, tracking individuals, anomalous behavior analysis, etc.

Stereo capture, 2014

In 1999, our first stereo multicamera simply mounted two D-340R cameras side-by-side on a straight flash bracket and required either that the user press both shutter buttons simultaneously or that both cameras be tethered via RS232C to a laptop computer. The self-contained portable stereo capture rig shown in Figure 6 was built primarily to test new mounting and camera control technologies.

One of the biggest issues in stereo photography is being able to make the left and right views match as well as possible. One aspect of matching is timing: the shutters should be fired at the same time. However, the best results also require matching the shutter speeds, focus distance, depth of field, focal length (of zoom lenses), color balance, etc. There is also the issue of making the stereo baseline be appropriate; in this case, flipping one of the cameras upside-down enabled the cameras to be positioned with the standard human eye separation rather than an excessively wide separation that would exaggerate the stereo effect. This flipped mount, and the contoured grip, are complex parts we made by 3D printing using a consumer-level printer at a cost of about $1.

Earlier stereo capture rigs tended to use mechanical means for synchronization of the shots. However, by using Canon PowerShot A4000 cameras under CHDK (Canon Hack Development Kit)[6], we were able to electronically trigger the cameras. The electronic triggering is accomplished by simply applying 5V to the USB connector in each camera, which in this 3D-printed rig is accomplished by pressing a button on the grip that closes a circuit with a 9V battery and 5V regulator.

The nice thing about this USB triggering is that it can be detected by a user-written script, so the cameras can be ready to fire with minimal delay and the action associated with the trigger signal can be arbitrarily complex. However, the USB cable is used exclusively for this synchronization; there is no mechanism for communicating between the cameras. This means that all cooperative behaviors must be pre-programmed. For example, if lighting conditions require a flash exposure, it would make the most sense for only one of the two cameras to fire its flash, not both, but this could only be implemented by a priori selecting which one fires its flash. Even a little communication would be helpful; for example, if there are other images already taken, the shots captured as a stereo pair might not end-up with similar filenames in the two cameras, whereas it would take very little communication to agree upon naming for each pair of captures. In this sense, it would be much better to use USB to tether the cameras to a single-board computer.

FourSee, 2015

TDCI (Time Domain Continuous Imaging)[7] is a new approach to imaging in which the image data is fundamentally continuous in the time domain: frameless. Image data is represented as a continuous waveform per pixel, and virtual exposures can be rendered for any time period by integrating the area under the pixel curves. The catch is that no sensors currently directly capture data in that form, so the continuous waveforms are derived from analysis of conventionally-acquired still image sequences or video frames[8]. The key is to obtain images covering the complete time interval without gaps and with both high temporal precision and high dynamic range. FourSee is a multicamera designed to do just that.

The total cost of all components of FourSee was shockingly low – about $800. Figure 7 shows the key components of the FourSee multicamera and Figure 8 shows the fully assembled unit. The central lens (a 135mm f/4.5 large-format enlarger lens) is mounted in a 3D-printed threaded focusing tube and projects an image onto a screen inside the rather complex 3D-printed housing. Four Canon PowerShot N cameras (purchased as factory refurbs) are focused on the projected image, thus producing four easily aligned capture streams all with the point of view defined by the central lens. The component cameras are coordinated, using CHDK programming and USB synchronization (with a 5V rechargeable battery integrated in the 3D-printed housing), to capture deliberately skewed exposures.

In operation, two cameras shoot high-speed video at 240FPS, but skewed by approximately half a frame time (two 1ms jiffies) so that an approximately 480FPS sequence can be de-

Figure 7. *Basic components of FourSee*

Figure 8. *FourSee*

rived. A third camera shoots conventional 1080 video at 24FPS, slightly overexposing to expand the dynamic range well beyond what the 240FPS video can capture. The fourth camera slowly shoots a sequence of stills overexposing even more, so that dynamic range is taken even further into the shadows. Combining the image data from all four cameras, it is theoretically possible to synthesize TDCI stream data with temporal accuracy of better than 1/480s (potentially much better, using rolling shutter timing) and as much as 16-20EV of dynamic range.

Although FourSee is a fully self-contained multicamera and theoretically could be very easy to use, in practice it is quite awkward. One problem comes from the fact that manual focus of the central lens is very difficult to judge by looking at the live view of the component cameras. Another problem lies in the mechanics of turning the cameras on/off, accessing batteries for charging, and reading the captures from the microSD cards. Finally, the fact that the captures not only have different framerates, but also different resolutions and orientations complicates the conversion to a single TDCI stream. In fact, the only things not awkward are the physical alignment of component cameras and implementing the complex synchronized exposure pattern on press of either the momentary or latching multicamera shutter button.

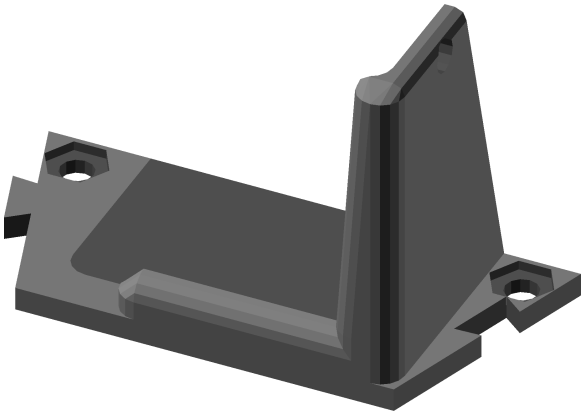

Figure 9. *KREight 3D-printed mounting part*

Figure 10. *KREight (Kentucky Radial 8)*

KREight (Kentucky Radial 8), 2017

KREight (Kentucky Radial 8) is a stand-alone 360° synchronized capture system using eight Canon PowerShot SX530 HS cameras ($130 each as refurbs) to capture 128MP images.

At some level, KREight is a testament to the wonders of 3D printing. Although the base is wood painted gray, the camera mounts, shown in Figure 9, are entirely 3D printed and (as in FourSee) require no shimming to align. The mounts are even color-coded such that camera 0's position is white and the rest are blue. There is not sufficient space to unscrew a camera once it has been installed in the final configuration, so the assembly procedure begins by mounting each camera in its 3D-printed holder using a standard 1/4-20 screw in the tripod socket. There is a rectangle of craft foam glued to the 3D-printed holder to help steady the camera position as it is screwed into place. The holders are then bolted into place on the base board.

Although the battery/SD card door is not blocked by the holder, the holder does need to be removed from the base in order to fully open the door. For this reason, we also produced 3D-printed dummy battery inserts so that the cameras could be externally powered. In practice, we found that battery life was long enough to make the awkwardness of changing batteries preferable to the wiring of centralized power to all dummy batteries.

Although it isn't shown in Figure 10, in operation the mounted cameras are each tethered to a central powered USB hub. These Canon cameras are again programmed using CHDK

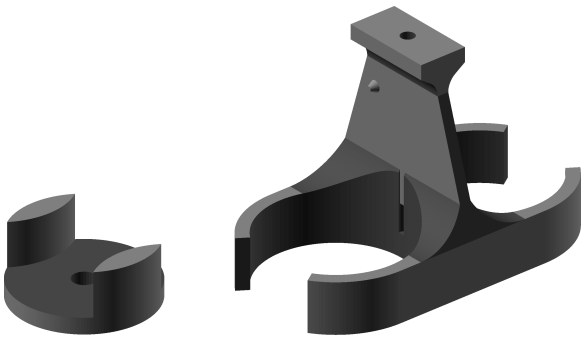

Figure 11. *KREighteen 3D-printed mounting parts*

Figure 12. *KREighteen camera-pair mount*

so that they sit idle until they see 5V on their USB connector, using the 5V signal to mark the start of a fully pre-programmed capture sequence.

The SX530 cameras support Canon's wifi control interface via a smartphone app. We had hoped to be able to use the wifi interface for out-of-band communication between the cameras, keeping the USB interface for synchronization. However, the wifi interface has not yet been hacked in a useful way.

KREighteen (Kentucky Rectangular 18), 2017

KREighteen (Kentucky Rectangular 18) is an array of 18 Canon PowerShot SX530 HS cameras running CHDK, but it is intended to behave more like a pair of nine-camera arrays sharing the same point of view. The idea is for each of these two 144MP nine-camera systems to capture the same scene from the same point of view, producing a stitched final image with about 90MP resolution. For TDCI, the two sub-arrays are fired at temporally interleaved times. The system can also be used for HDR imaging by simultaneously firing the two sub-arrays with different exposure parameters.

Cameras within the array are mounted in pairs, one camera from each sub-array, using the 3D-printed mounting parts shown in Figure 11. The part on the right holds a pair of SX530 HS PowerShots in fairly precise alignment by clamping the throats of their lenses; the cameras literally slide in and stop with their tripod socket locked in place over the conical bump. The cameras within a pair are upside-down to allow the lenses to be as close to each other as the shapes of the cameras can allow. Although

Figure 13. *KREighteen (Kentucky Rectangular 18) in operation*

there is significant parallax between cameras within a pair when the zooms are set to a very long focal length or the subject is very close, most of the 50X zoom range is usable with subjects at distances greater than 10 feet. The hole at the top of the right part is threaded for a standard 1/4-20 mounting screw, which attaches the pair to a standard adjustable tripod head. That head is in turn attached to a wooden mounting bar using a bolt through the part on the left, which essentially serves as an oversize washer to orient and lock the adjustable head on the wooden bar. Figure 12 shows a complete pair mount. KREighteen is thus a pair of physically-interleaved nine-camera arrays in which each pair of component cameras can be aimed independently while keeping the pair aligned, thus allowing precise positioning of cameras within each nine-camera array to have an appropriate field overlap to facilitate stitching with the selected focal length.

The fully assembled KREighteen is shown in operation in Figure 13. Again, coordination of the cameras is accomplished by CHDK software detecting the application of 5V to the camera USB connector. The 5V signal is supplied by one or two USB batteries routed through switchable USB hubs mounted on the rear of the wooden frame for the array. Thus, for TDCI capture, the procedure is simply to manually turn each camera on so it boots into our software running under CHDK, and then to turn-on the single USB hub switch that sends the 5V signal to all the cameras. We had hoped to use the wifi support built-into the SX530 HS cameras for collecting the images captured, but could not make that function reliably in the array, so the images were collected either using the USB cables or physically removing the SD cards.

Although KREighteen worked fairly well, we found that physical alignment was a huge problem. By design, there was barely enough space between the pairs of cameras to reach in and adjust the tripod heads. This problem was further complicated by the fact that it was difficult to see the rear LCDs to be able to adjust the aim of each camera pair. In sum, the array required too much manual adjustment. The adjustment also had to be done largely sequentially; it was possible for two people to be adjusting cameras on opposite sides of the array simultaneously, but time to adjust still approached half an hour. Not only was this slow set-up time annoying, but the cameras that had been turned on first slowly heated up; one would expect this heating to

Figure 14. *Kodama uses three cameras, six lenses, for 360° TDCI*

only negatively impact battery charge life, but the heating apparently altered the internal clock frequencies in the cameras. The change was subtle, but resulted in accumulation of timing error as the cameras ran synchronized capture sequences. The result was enough clock drift to lose synchronization long before batteries ran out of charge. Of course, occasionally sending a USB sync signal could be used to implement a clock update protocol to prevent the skewing.

Kodama, 2017

Kodama is an oddly-shaped USB-tethered array of three Insta360 Air cameras to capture temporally-interleaved HDR (High Dynamic Range) 3008x1504 video for TDCI. Technically, this tiny multicamera is an array of arrays, as each of the three component cameras has two lenses and sensors. It took several design iterations to optimize the 3D-printed body design, but the version shown holds the component cameras in alignment such that all portions of the environment are seen by at least two of the three cameras. Despite that, total parts cost was under $400.

Using Kodama is simply a matter of plugging all three USB cables into a computer that sees them as webcams. However, simultaneously high data rates on three separate USB devices are not handled well by most computers. In addition, although the goal is to be able to increase temporal resolution (and perhaps dynamic range) for TDCI by skewing exposures, the control of exposure details is relatively poor. To obtain the best quality would require empirically measuring differences between camera exposure details by comparing the images streamed from them[9].

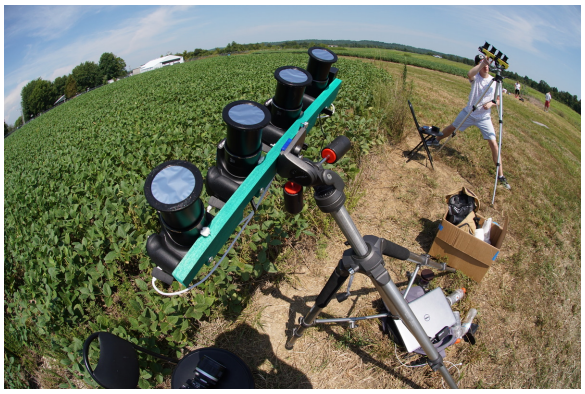

Figure 15. *Green and yellow MASKs at Princeton, KY*

MASK, 2017

The MASK (Multicamera Array Solar from Kentucky) multicameras are simple four-camera linear arrays. Each MASK was literally just four Canon PowerShot SX530 HS cameras mounted on a wooden rail with USB triggering via a switched hub mounted on one end of the rail. We made five MASK multicameras to record the solar eclipse on August 21, 2017 from two locations: a partial visible from outside our lab on the University of Kentucky's Lexington campus and the total visible from the Princeton, Kentucky, airport.

Each MASK was dedicated to a particular type of capture. Some had filters to capture multi-spectral data, others were intended to capture high dynamic range (HDR) using skewed exposure settings. All were programmed to capture sequences for later conversion into TDCI streams and all had specially-made solar filters in custom 3D-printed holders that fit the SX530 HS lens bayonet.

Unfortunately, we grossly underestimated the difficulty in manually having the MASKs track the sun at the equivalent of 1200mm focal length. We knew framing would be very touchy because we had to shim the camera mounts with thin paper to align them at that focal length. However, we had expected to use the live view on the back of a camera to check framing, but the SX530 HS cameras have a fixed rear LCD, so it can be very difficult to position yourself to be able to see that display. This proved to be a fatal problem using the MASK multicameras to photograph the solar eclipse; only about 10% of our captures are of good quality. The cameras needed to be pointed up at a high angle, which made it very difficult to get behind the LCD and, except during totality, the bright sun lighting the light-colored ground at the airport made such a strong reflection on the LCD that it was nearly impossible to see the live view. This problem was compounded by the fact that the tripods we used were not really rigid enough to support the MASK arrays at the necessary angle and removing/replacing the solar filters around totality disturbed the shimmed alignment.

Adding insult to injury, it was hot enough that some of the thin PLA plastic in our 3D-printed filter holders deformed slightly, making it harder to remove and replace the filters around totality. Ordinary PLA becomes soft and can be deformed by a temperature of approximately 140°F. Fortunately, there are other

materials, including heat-treated PLA, which can withstand much higher temperatures. We had not anticipated that the filters would be heated to that level by simply sitting in direct sunlight.

Lessons learned

The multicameras described above span a wide range of purposes and configurations, yet a number of common issues are evident. The following briefly summarizes lessons learned.

Programmable camera modules

One of the repeating themes in nearly all our multicameras is that it is preferable to build multicameras using programmable cameras as modules rather than "dumb" camera modules. Surprisingly, this is not how most groups build their multicameras. Although it has long been standard practice in modern computing systems to offload as much computation to Co-processors as possible, people still often think of cameras as they were when they recorded images on film.

There have been a number of attempts to build programmable cameras[10, 11], but none are price/performance competitive with self-contained consumer cameras. Yet, consumer camera manufacturers have been slow to respond to the market for programmability. Fortunately, user communities have developed ways to program various consumer cameras. Most of Sony's cameras support an Android-based camera app mechanism that Sony oddly has not opened to outside developers, but has been made somewhat accessible by the OpenMemories project[12]. Magic Lantern[13] is community-developed software that provides programmability and enhanced features for a variety of Canon EOS models. The CHDK (Canon Hack Development Kit)[6] supports programming of many Canon PowerShot models using compiled native code, Lua or BASIC scripts, and even a remote execution facility called chdkptp.

Although we have experimented with OpenMemories and Magic Lantern, CHDK cameras tend to offer the best price/performance for multicameras and self-contained single-camera programmable systems. The average cost of the Canon PowerShots we use is around $100 per camera, which not only buys a 12-20MP image sensor with a power zoom and focus lens, but also a programmable dual-core 32-bit ARM-based computing system. Even a relatively small degree of programmability can help in major ways. For example, an external trigger would only enable a "dumb" camera to perform one function, whereas we commonly use external signals to trigger complex operations that may involve many steps and might even dynamically modify those operations based on the state of the camera as determined by non-trivial processing of local sensor data in real time.

Synchronization of local clocks

It is natural to assume that synchronization of component cameras within a multicamera will be the primary issue. The standard model is that a separate controller (computer) broadcasts a signal to all cameras when they should perform an image capture. What makes that type of synchronization difficult is not that synchronization is hard, but that broadcasting such a signal is inherently open-loop control and the component cameras might not be ready to do what is requested in a timely manner.

The only way to have a camera ready to perform an operation is for it to have some sense of time so that it can schedule its internal operations to meet the performance goals of the multicamera. That sense of time comes from having fast local access to a clock. Thus, the broadcast signals are not used to trigger captures per se, but to establish synchronization of local clocks.

Local storage and processing

A multicamera is a parallel computer in which each node is creating a large volume of new data in real time. Sending all data to a central unit for storage and/or processing creates a serial bottleneck that does not allow scaling to more than a few nodes.

For example, consider Kodama. Each of the three component Insta360 Air cameras is capable of capturing a video stream of 3008x1504 images, but there is no internal storage. The data rate from a single full-resolution video feed is sufficient to saturate a typical USB interface; some smartphones and even some laptops cannot handle a continuous feed at full resolution. Feeding full-resolution video simultaneously from Kodama's three Insta360 Air cameras into a single USB controller causes corruption of the videos on all but one of the laptop computers we have tested. Collecting live video streams from many such cameras into a single machine would be challenging.

Instead, suppose that each of the component cameras has significant local storage. Writing image data in parallel across these local storage systems can more more easily accommodate very high aggregate data capture rates. However, combining local storage with local processing can enable the communicated data rate to be significantly reduced while still providing intelligent real-time behavior. For example, many industrial cameras allow specification of a "region of interest" (ROI) such that the camera only transmits image data for the selected crop region rather than data for the full sensor; it is not uncommon that small ROIs can significantly multiply the transmitted frame rate. The catch is that the controller must specify the ROI without seeing the full image. Moving the selection of the ROI inside the component camera can allow a computation to evaluate more of the image to intelligently select the ROI. For example, $100 Canon PowerShots automatically identify faces in captured images and tag the JPEG files with a list of face bounding boxes. Thus, it is straightforward for a Canon PowerShot to be programmed using CHDK to only stream the ROIs containing the faces rather than the full captured image. All sorts of intelligent filtering of the image stream can be implemented locally, limited primarily by the level of programmability permitted by the camera and the computing resources available.

Of course, the best way to use local processing is with actions across the multicamera coordinated as a parallel computation. In other words, local processing should include the ability to communicate with the control interface and with other component cameras. For example, in a multi-camera surveillance system, seeing a face in one camera's view could be used to trigger capturing an ROI in another camera that corresponds to the same general area the face was seen in. This type of real-time messaging and coordination can improve system performance in many ways.

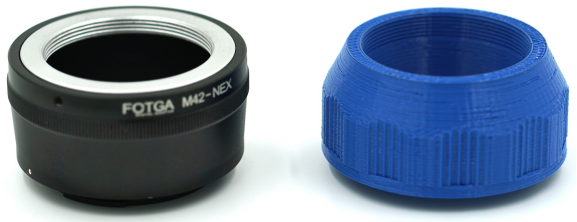

Figure 16. *M42 to Sony E-mount lens adapters: commercial and printed*

Physical mounting and alignment

Although it seems conceptually easy to mount a group of cameras together, the truth is that positioning and alignment of optical systems always requires high precision. Alignment issues are often the primary reason a multicamera fails to be as useful as intended. It is possible for the physical arrangement to be somewhat imprecise using computational processing to correct, but computational corrections can be expensive, generally require a calibration process, and still produce at least slightly inferior image quality.

The ability to custom design and cheaply fabricate jigs and adapters using various rapid prototyping technologies is a game changer in the quest for better optical mounting and alignment. Our earliest multicameras used mounting systems that were essentially adventures in precision woodworking – and machining wood, plastic, or metal is still used for large parts (primarily base frames). However, the majority of custom mounting components in our multicameras are now 3D printed. 3D printing is significant primarily because it allows production of parts with complex shapes and accurate dimensions. Although industrial cameras often come in easy-to-mount rectangular metal boxes, they are relatively expensive, dumb (non-programmable and missing features like image stabilization or even autofocus), and tend to employ poorer sensors than more consumer-oriented cameras; the shapes of consumer cameras are usually designed to be handled by humans, using complex curves and a variety of materials. In many ways, 3D printing has become the glue that allows consumer camera equipment to be treated as modular components with physically-compatible interfaces.

How accurate can 3D-printed objects be? Using a sub-$400 printer extruding PLA plastic, it is easy to hold tolerances to within 0.1mm. For example, some M42 mount lenses have the aperture ring overhang the 42mm diameter 1mm pitch mount flange, preventing them from being used on commercial M42 adapters. Figure 16 shows our printed solution, which contains about $0.25 worth of opaque blue PLA.

It has been our general experience that the more rigidly set the positioning and alignment of the component cameras is, the more usable the multicamera. Thus, locking component cameras into fixed positions has been a winning strategy; for example, KREight is orders of magnitude easier to use than KREighteen because of KREighteen's multitude of alignment adjustments. This can also be considered an error in defining the goal for KREighteen: because the component cameras have 50X zoom lenses, we set a goal of being able to use the array at different focal lengths for different photographic tasks, and that required the camera mounts to be adjustable in order to have appropriate overlap between camera views for stitching. In contrast, although KREight uses the same component cameras, they were assumed to always be set at the same focal length (determined by appropriate overlap in their one camera per 360°/8 layout).

An alternative we are currently investigating is making each of the component camera mounts adjustable under computer control. The complexity of controlling so many separate axes would have been prohibitive just a few years ago, but servos, steppers, and drive electronics have become commodity technologies. Professor Dietz is currently supervising an electrical and computer engineering senior project team developing hardware and software that will be able to automatically adjust camera alignment in multicameras like KReighteen using analysis of camera images to guide the alignment process.

Live view

In consumer cameras, it is now expected that a digital camera will have some provision for an electronically-generated live view. Even DSLRs (digital single-lens reflex cameras) using an optical viewfinder are expected to also provide the option of live view on a rear LCD panel. However, multicameras do not always have a structure that would make it easy to provide a live view.

One issue is that the only meaningful live view for a multicamera might require extensive processing that isn't feasible in real time. However, a method for checking framing, focus, and exposure is still needed.

Using the live view on the back of a camera to check framing (or to adjust alignment) can be very awkward. For example, the SX530 HS cameras have a fixed rear LCD, so it can be very difficult to position yourself to be able to see that display. This proved to be a fatal problem using the MASK multicameras to photograph a solar eclipse. The cameras were pointed up at a high angle making it difficult to get behind the LCD and, except during totality, the bright sun lighting the ground made such a strong reflection on the LCD that it was nearly impossible to see the live view. Unfortunately, our earlier tests had not revealed this problem because differences in the camera angle and ground appearance during our tests did not cause problematic reflections on the LCD.

Fault tolerance

Any system that consists of a large number of subsystems essentially multiplies the subsystem failure rate: a multicamera is very likely to have one or more component cameras fail. Failure does not necessarily mean the camera itself has failed, but could mean something as simple as the battery's charge being exhausted or that a USB cable came loose. The most likely true failure would be an SD card wearing out – each memory cell on an SD card has a lifespan of only thousands of writes. The key is to design the system so that such failures can be easily dealt with in the field. The general solution is to always bring spares and whatever tools would be needed to install the replacement.

The most insidious type of failure we have encountered involves having the wrong software installed in a component camera. In KREighteen, the software executed for each group of nine component cameras is slightly different. The difference is para-

metric, one group delays its shots relative to the other, but unfortunately there is no easy way to make that parameter be delivered to the camera while the USB port is being used for 5V-signal synchronization. Thus, there are two different SD card images for the two different sets of nine cameras each. In one shoot, we accidentally swapped the SD cards of two component cameras, resulting in two cameras firing with the wrong groups. It is also easy to lose track of which SD cards came from which component cameras.

The correct solution to these SD-card mix-ups is to have an out-of-band communication channel that can be used to provide the software in each camera with appropriate parameters. For Canon PowerShots under CHDK, this can be done using chdkptp to directly control the in-camera scripts via USB tethering. The cost is that synchronization timing accuracy is poorer using the USB protocol than it is using the 5V detection method. The ideal out-of-band interface would be 802.11 wifi, which some of the cameras implement, but has not been hacked to the point of being a reliable mechanism for this purpose. We have had significant success using Toshiba FlashAir[14] wifi SD cards to implement bidirectional out-of-band communications with the camera. In fact, the FlashAir cards are themselves programmable using Lua, allowing offloading of some tasks from the camera's processors. Although they somewhat cost more than ordinary SD cards, the real problem is that actively writing messages into the wifi SD card implies a lot of write cycles, which quickly can lead to SD card failure.

Conclusion

The various multicameras discussed in this paper span a period of 18 years. During that time, digital camera technology has changed very significantly. However, one aspect has not changed: most people still think of cameras in terms of how film cameras worked. A film camera was a non-programmable device that could be commanded to capture an image with various capture parameters, so that's what people expect of them and what most camera manufacturers tune their cameras to do. The problem is that multicameras are fundamentally not behaving very much like film cameras.

A component camera in a multicamera should be a relatively general-purpose computer with the ability to intelligently control image capture. These component computing elements should be able to coordinate their actions as a parallel computer – in other words, they should be able to communicate with each other and with a front-end system that presents a coherent view of the entire multicamera to the user. In a very real sense, a multicamera should be a cluster supercomputer.

Acknowledgments

This work is supported in part under NSF Award #1422811, *CSR: Small: Computational Support for Time Domain Continuous Imaging.*

References

[1] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy, High performance imaging using large camera arrays, *ACM Trans. Graph. 24, vol. 3, pp. 765-776; doi:10.1145/1073204.1073259* (2005)

[2] Kensuke Ikeya, Kensuke Hisatomi, Miwa Katayama, Tomoyuki Mishina, and Yuichi Iwadate, Bullet Time Using Multi-viewpoint Robotic Camera System, *Proc. 11th European Conference on Visual Media Production, pp. 1:1–1:10, doi:10.1145/2668904.2668932* (2014)

[3] Brian Leininger, Jonathan Edwards, John Antoniades, David Chester, Dan Haas, Eric Liu, Mark Stevens, Charlie Gershfield, Mike Braun, James D. Targove, Steve Wein, Paul Brewer, Donald G. Madden, Khurram Hassan Shafique, Autonomous real-time ground ubiquitous surveillance-imaging system (ARGUS-IS), *Proc.SPIE 6981, Defense Transformation and Net-Centric Systems 2008, 69810H; doi:10.1117/12.784724* (2008)

[4] Helmut Dersch, Panorama tools: open source software for immersive imaging, *The international VR photography conference proceedings, vol. 1* (2007)

[5] J. Hannemann, K. Donohoue and H. Dietz, A cluster-based computing infrastructure for wide-area multi-modal surveillance networks *Proc. IEEE International Symposium on Circuits and Systems, pp. 2070-2073, doi:10.1109/ISCAS.2008.4541856* (2008)

[6] Canon Hack Development Kit (CHDK), *http://chdk.wikia.com/wiki/CHDK* (2018)

[7] Henry Gordon Dietz, Frameless, time domain continuous image capture, *Proc. SPIE 9022, Image Sensors and Imaging Systems 2014, 902207 (March 4, 2014); doi:10.1117/12.2040016.* (2014)

[8] Henry Dietz, Paul Eberhart, John Fike, Katie Long, Clark Demaree, and Jong Wu, TIK: a time domain continuous imaging testbed using conventional still images and video, *Electronic Imaging, Digital Photography and Mobile Imaging XIII, pp. 58-65(8)* (2017)

[9] Helmut Dersch, Subframe Video Post-Synchronization, *http://webuser.fh-furtwangen.de/d̃ersch/sync/sync_1.pdf* (November 24, 2016).

[10] Andrew Adams, Eino-Ville Talvala, Sung Hee Park, David E. Jacobs, Boris Ajdin, Natasha Gelfand, Jennifer Dolson, Daniel Vaquero, Jongmin Baek, Marius Tico, Hendrik P. A. Lensch, Wojciech Matusik, Kari Pulli, Mark Horowitz, and Marc Levoy, The Frankencamera: An Experimental Platform for Computational Photography, *ACM SIGGRAPH 2010 Papers, pp. 29:1–29:12, doi:10.1145/1833349.1778766* (2010)

[11] Eben Upton and Gareth Halfacree, Raspberry Pi user guide *John Wiley & Sons* (2014)

[12] OpenMemories, *https://github.com/ma1co/OpenMemories-Tweak* (2018)

[13] Magic Lantern, *http://www.magiclantern.fm/* (2018)

[14] Toshiba FlashAir, *http://www.toshiba-memory.com/cms/en/products/wireless-sd-cards/FlashAir* (2018)

Author Biography

Henry (Hank) Dietz is a Professor in the Electrical and Computer Engineering Department of the University of Kentucky. He and the student co-authors of this paper, Clark Demaree, Paul Eberhart, Chelsea Kubal, and Jong Yeu Wu, have been working on a wide variety of computer engineering topics from parallel supercomputing through computational photography, with a focus on making Time Domain Continuous Image capture and processing practical. See Aggregate.Org for more information about their research.