Cambits:

A Reconfigurable Camera System

Makoto Odamaki and Shree K. Nayar

Computer Science Department
Columbia University
New York, NY 10027
mo2615@columbia.edu
nayar@cs.columbia.edu

Technical Report CUCS-002-16
February 12, 2016

Abstract

Cambits is a set of physical blocks that can be used to build a wide variety of cameras with different functionalities. A unique feature of Cambits is that it is easy and quick to reconfigure. The blocks are assembled using magnets, without any screws or cables. When two blocks are attached, they are electrically connected by spring-loaded pins that carry power, data and control signals. With this novel architecture we can use Cambits to configure various imaging systems. The host computer always knows the current configuration and presents the user with a menu of functionalities that the configuration can perform. We demonstrate a wide range of computational photography methods including HDR, wide angle, panoramic, collage, kaleidoscopic, post-focus, light field and stereo imaging. Cambits can even be used to configure a microscope. Cambits is a scalable system, allowing new blocks and accompanying software to be added to the existing set.

1. Introduction

The cameras in our phones and tablets have turned all of us into avid photographers. Today, we regularly use these devices to capture special moments and to document our lives. One attractive feature of camera phones is that they are compact and fully automatic, enabling the user to simply point and shoot without having to adjust various settings. When we need to capture photos of high aesthetic quality, however, we resort to a more sophisticated DSLR camera where a variety of lenses and flashes can be used in an interchangeable fashion. This flexibility to reconfigure the camera is important to span the entire range of real-world imaging scenarios as well as to enable the user to be more creative.

A few attempts have been made to make the camera a more flexible device, both in terms of hardware and software. For example, Ricoh's GXR camera has interchangeable lens units, each with a different type of sensor [1]. Some manufacturers make their cameras more flexible by releasing application program interfaces (APIs). Developers can use the APIs to control various camera parameters and to create new image processing tools. For example, in 2015, Olympus released the Open Platform Camera, which can be controlled via Wi-Fi and Bluetooth [2]. At the high end of the market, RED released a modular camera that has interchangeable parts - lenses, battery packs and broadcast modules [3]. Although they provide some level of flexibility, the above products are limited in the types of images they can produce.

In the realm of research, Levoy et al. [4] proposed a computational photography platform, Frankencamera, which has an API, a sensor interface and an image processing unit. This system can be used to implement various computational imaging methods. However, its hardware is relatively rigid which limits the extent to which it can be reconfigured. Manakov et al. [5] proposed a camera system that can accommodate different optical add-ons. The system uses kaleidoscope-like imaging to make optical copies of the captured image. Different filters are then used to pro-

duce HDR, multispectral, polarization and light field images. This system provides some reconfigurability, but is bulky and hard to scale in terms of functionality. Finally, reconfigurability is a well-explored topic in the field of robotics [6]. For instance, Eric and Gross [7] have developed a robot kit for science education, which include blocks with different functionalities called Cublets.

In this paper, we present Cambits, a set of physical blocks that can be used to build a wide variety of cameras with different functionalities. The blocks include sensors, actuators, lenses, optical attachments and light sources. The blocks are assembled using magnets, without any screws or cables. When two blocks are attached, they are electrically connected by spring-loaded pins that carry power, data and control signals. The host computer always knows the current configuration and automatically presents the user with a menu of imaging functionalities to choose from. We use Cambits to demonstrate a wide range of computational photography methods including HDR, wide angle, panoramic, collage, kaleidoscopic, post-focus, light field and stereo imaging. It is also used to configure a microscope. Cambits is a scalable system, allowing new blocks and computational photography algorithms to be added to the existing set.

2. Concept

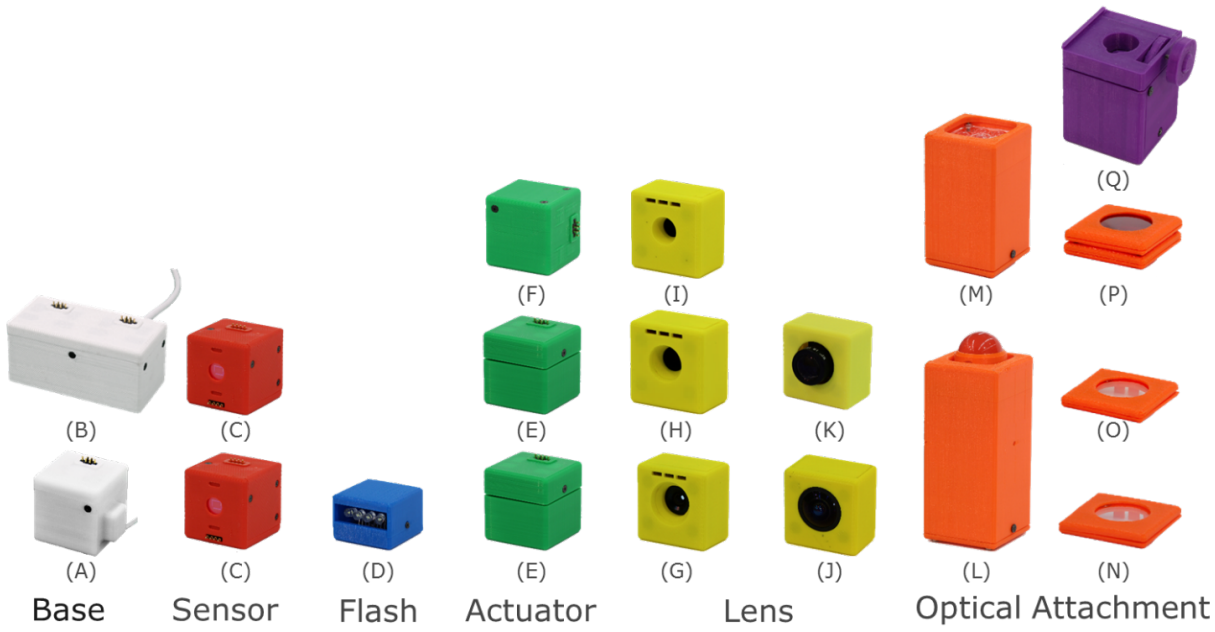
Figure 1(a) shows the set of blocks that constitute Cambits. The blocks come in different colors, each color indicating a specific function. We have white for base, red for image sensor, blue for flash, green for actuators and spacers, yellow for lenses, and orange and purple for optical attachments. As shown in Figure 1(b), the host computer always knows the current configuration of Cambits. It uses a suite of computational photography algorithms to produce a variety of images. The system has been designed to have the following attributes:

1. **Ease of assembly:** The blocks are attached using magnets, without any screws or cables. The configuration of blocks can be changed without requiring a reboot of the hardware or the software.

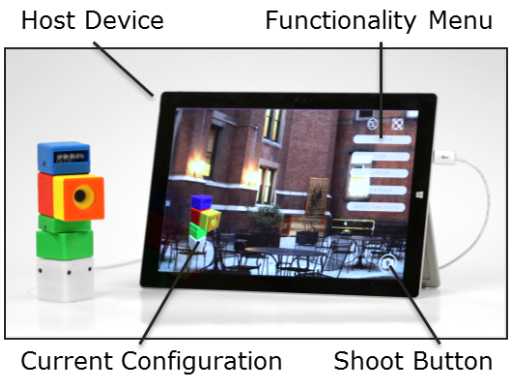
2. **Self-identification:** The host computer can detect the current configuration of the system. This information is conveyed to the user via a 3D visualization and a menu of functionalities that it can perform.

3. **Diverse functionality:** Since there are many different types of blocks and many of them can be controlled, a diverse set of camera systems can be configured where each one produces a different type of image.

4. **Scalability:** The design of the architecture of the hardware and the software of the system makes it inherently scalable. New blocks and computational photography algorithms can be easily added to the existing set.



(a)



(b)

Block Type	Specification
Base	(A) Single
	(B) Dual
Sensor	(C) 1.3MP, 1288 x 964, 1/3'' CCD, 15 fps
Flash	(D) 4 LEDs with Controller, Max. Current 25mA/LED
Actuator	(E) Single Axis Rotary Actuator, 180 ° Range
Spacer	(F) Right Angle
Lens	(G) 12mm, F2.0, Horizontal FOV 22.2 °
	(H) 8mm, F2.0, Horizontal FOV 33.8 °
	(I) 4.3mm, F2.0, Horizontal FOV 87.7 °
	(J) 1.3mm, Fisheye, F2.8, Horizontal FOV 180 °
	(K) 16mm, F1.4, with Piezoelectric Linear Actuator
Optical Attachment	(L) Soft focus
	(M) Warm
	(N) Polarization
	(O) Lens Array, 7 Lenslets
	(P) Teleidoscope
Microscope	(Q) Objective Lens, x1.45, with LED Illumination

(c)

Figure 1: *(a) The components of Cambits. (b) One configuration of Cambits. The host computer displays a 3D visualization of the current configuration and a menu of functionalities it can perform. (c) The list of Cambits blocks and their specifications.*

3. System Architecture

The key attributes of Cambits described in the previous section are made possible by the design of its hardware and software architecture. We now describe the details of the architecture.

3.1 Mechanical and Electrical Connections

Each Cambits block is 40mm along at least two of its three dimensions. The chassis of each block is 3D printed. The chassis includes sockets close to its corners that hold magnets. The magnets are used to not only attach blocks to each other but also mechanically align them, as shown in Figure 2(a). The alignment is aided by convex and concave bumps on the surface of the chassis. The polarities of the magnets in each block are also chosen such that it is not possible for the user to attach two blocks that are incompatible. For instance, a lens block cannot be directly attached to an actuator block. When two blocks are attached, a set of either 4 or 6 spring-loaded pins on one block (see Figure 2(b)) are aligned and electrically connected to contact pads on the other block. The system uses USB 2.0 for the data signal and I2C for the control signal.

3.2 Tree Structure with Bucket Brigade

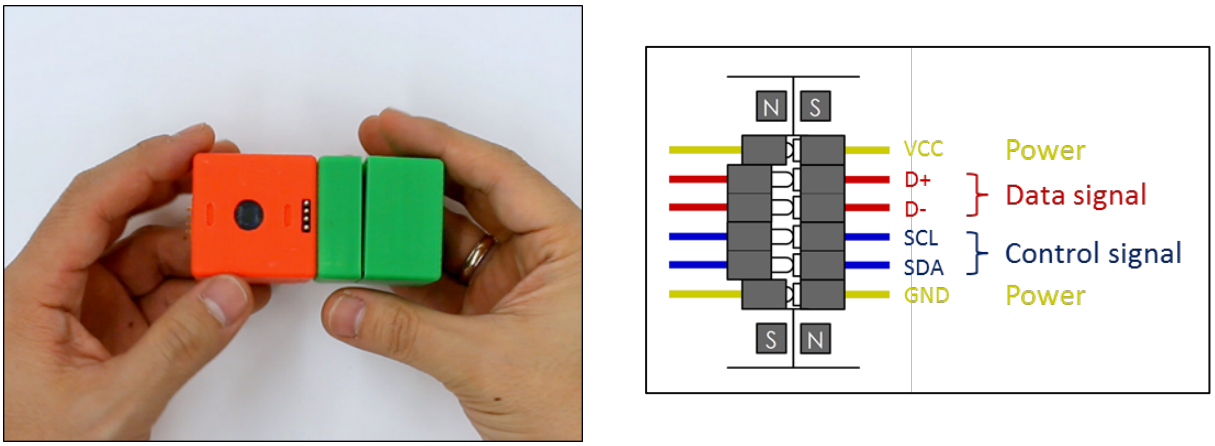
Each Cambits block has three types of pins to convey power, data signals, and control signals. The data signal conveys image data from the sensor block. The control signal conveys the configuration data upstream and various commands, such as the actuator block's rotation parameters or the flash's strobing parameters, downstream.

These signals have a tree structure. The host device, the root of the structure, provides electrical power to all the blocks in the configuration, detects the current configuration of the entire tree structure, and control all the blocks within the tree. Each block is allowed to connect with multiple blocks. Upstream is defined as the direction toward the host device, and downstream is defined as the direction in which components proceed forward from the host device (see Figure 3).

The data signal flows upstream directly from each sensor block to the host device. However, the control signals are passed in a bucket brigade manner from one component to the next. Each block can communicate via control signals only with blocks that are connected to it. When a block is attached to the system, it scans the components downstream. If it detects any blocks, it reads the configuration data of the blocks that are downstream, adds its own identity and address to the data, and sends this information upstream. As a result, the host device can detect the complete order of the configuration. Instead of this architecture, if we had used a conventional electrical bus for the control signals, the system would not have been able to detect the order of the blocks. In addition, a conventional bus would not be able to detect configurations in which multiple blocks have the same address, which is a possibility when using the I2C interface.

When the host device seeks to control a specific block in the tree structure, it will send its command and the address of the block to the base block. The base block

and subsequent blocks pass the command downstream in a bucket brigade fashion. The addressed block receives the command and executes it.



(a) (b)

Figure 2: *(a) Mechanical assembly and alignment of blocks using magnets. (b) Electrical connection between blocks using spring-loaded pins that carry power, data and control signals.*

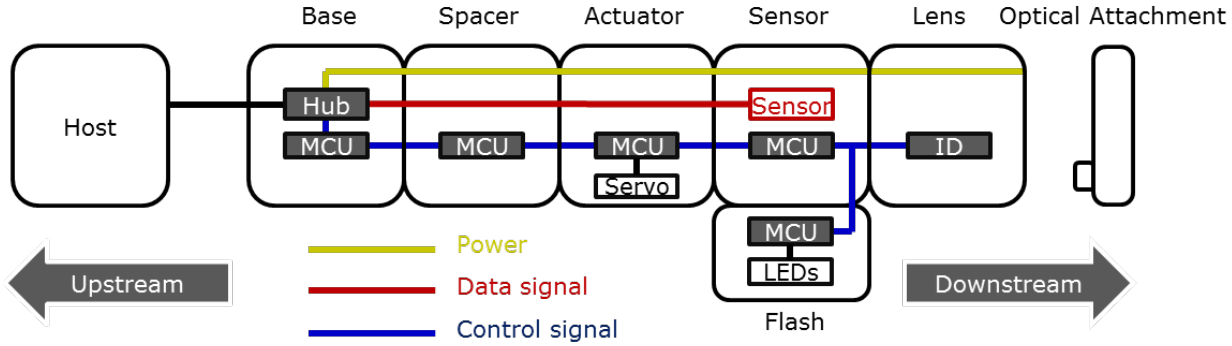


Figure 3: *Tree architecture used to implement Cambits. Power flows downstream, data flows upstream, and control signals are communicated in a bucket brigade fashion.*

3.3 The Controller Board

A key aspect of our design is the controller board which sits insides the base, actuator, spacer and sensor blocks. It includes a microcontroller unit (MCU) (Texas Instruments MSP430F5510), which has two serial ports for the bucket brigade. The controller board has an upstream interface and a downstream interface (see Figure 4).

When a block with the controller is attached to the system, it automatically turns on and then the firmware on its MCU starts to scan downstream repeatedly for approximately 100 msec to communicate with its adjacent blocks. When the block is removed from the system, it loses power and the firmware stops.

Each block has a power circuit to prevent inrush current and voltage drop when it is attached. This circuit can maintain a steady input voltage. Owing to this circuit, the system can be reconfigured without requiring a reboot of the hardware (blocks) or the software running on the host computer.

Additionally, the controller board can control other devices such as the servo motor in the actuator block and the LED controller in the flash block by using the I2C bus, pulse width modulation (PWM), and general purpose input outputs (GPIO), based on commands it receives from the host device. For example, when an actuator block receives the command for rotation, the MCU generates the pulse signal needed to drive its servo motor.

In term of scalability, the I2C interface is useful, because it is widely used in the field of embedded systems. This allows us to add various devices such as a light sensor, acoustic sensor, IR sensor, GPS, IMU and multispectral light source to the current Cambits set.

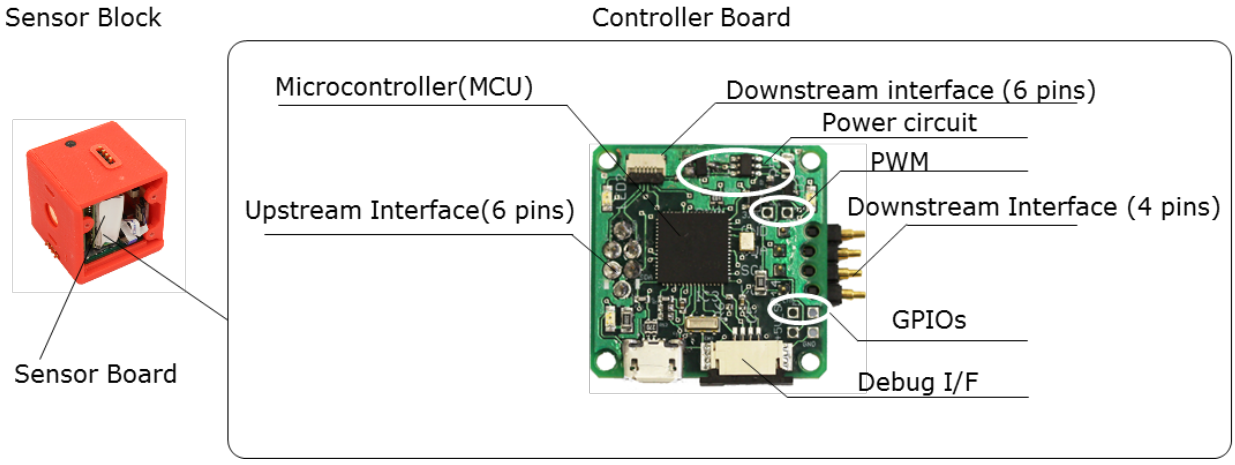


Figure 4: *The base, actuator, spacer and sensor blocks include a controller board which allows a block to communicate with its adjacent blocks.*

3.4 Lens Blocks and Optical Attachments

The lens block has an identification board that includes an I2C expander device. It can detect the identification number of the lens type itself and an additional optical attachment connected to the lens such as a soft focus filter, lens array or teleidoscope. The optical attachment does not have any electrical parts. Instead, it has up to three bumps that push against mechanical switches on the lens block to generate a 3-bit code that the lens block can use to identify the attachment. This information is sent by the lens block upstream.

3.5 Sensor Block

The sensor block includes a Point Grey camera board (BFLY-U3-13S2C-CS). The board can produce 1.3 megapixel video in various formats, including YUV411 and RGB8, and send the video upstream as a USB 2.0 data signal. The length of the da-

ta signal line and the number of connectors are minimized so as to enable high frequency (480Mbps) transmission, which is needed to preserve the integrity of the video [8]. Various imaging parameters of the sensor board, such as exposure time and gain, are controllable from the host device.

3.6 Software

The software system that runs on the host device captures images from the Cambits system, provides the user a 3D visualization of the current configuration, and allows the user to apply various computational photography methods to the captured images. Our current implementation runs on Windows and is based on open source libraries such as Open CV-v2.4.10 and Cinder v1.20 (see Figure 5). It also uses the Point Grey SDK Flycapture2 to control the image sensors.

The Cambits API can receive images from the Point Grey SDK and control camera parameters such as exposure time and gain. Also, the API can control various devices on the tree architecture, such as servo motors, linear actuators, and LEDs, via serial ports. The API and the open libraries allow developers to easily add new blocks and image processing algorithms to the current system.

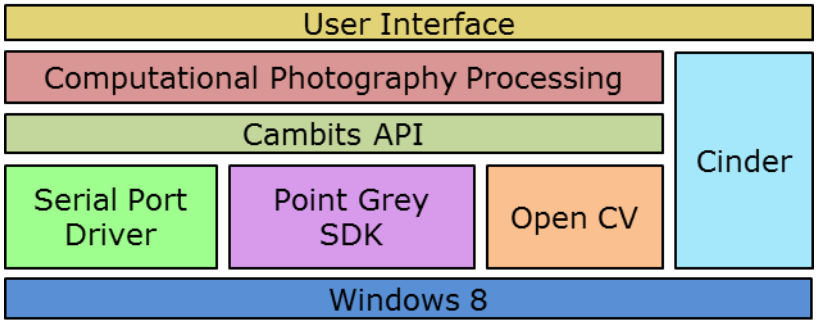


Figure 5: *Software architecture of Cambits.*

4. Functionalities

We have used Cambits to assemble a wide range of computational cameras. Figures 6 show a few examples of the systems we have configured. To construct a basic camera we use a base, a sensor block and an 8mm lens block. In high dynamic range mode, the camera captures a set of images with different exposures, computes an HDR image and then tone maps it [9][10]. This basic camera can also be moved around to capture a set of images that are fused to obtain a scene collage. We detect the features in each image using SIFT, reduce the outliers by using RANSAC, and find corresponding features between all pairs of images. The image that has the most corresponding features with all the remaining images is used as the center image of collage. The remaining images are transformed to align with the center image and overlaid to obtain the collage [11]. Cambits includes a variety of lenses. A fisheye lens can be used to capture a wide-angle image. Since the host knows the type of lens that is being used, the software automatically maps the captured image to a perspective one without barrel distortion [12].

As with the lenses, a wide variety of optical filters can be attached to the lens of the imaging system. These include simple optical filters like diffusion and polarization, as well as more complex ones like a lens array and a teleidoscope. The lens array attachment has 7 ball lenses and produces a 4D light field image of the scene [13]. The teleidoscope attachment produces a kaleidoscope image. A ball lens in the front of the attachment captures the scene image and a set of planar mirrors between the ball lens and the lens block creates multiple rotated copies of the image.

Our focal stack lens block includes a linear actuator that physically sweeps the lens to capture a set of images that correspond to different focus settings. This stack of images is then processed using our refocusing algorithm to generate an interactive image where the user can simply click on any part of the image to bring it into focus [14][15].

We can insert a rotary actuator between the base and the sensor to scan a panorama of the scene. Left and right strips can be taken from the same sequence of images to generate a stereo panorama for virtual reality [16]. A second rotary actuator can be added to the system to configure a pan/tilt camera system.

Cambits is not restricted to a single image sensor. Our second base can be used with two sensor blocks and lenses to create a stereo camera system. The left and right video streams from this system are processed in real time to produce a gray coded depth video of the scene.

Finally, Cambits can be used to assemble a microscope. Our microscope attachment includes an objective lens, a mechanism to adjust the height of the sample slide to bring the sample into focus, and an LED light to bright-field illuminate the sample. Alternatively, the ambient illumination in the environment can be used to back-light the sample.

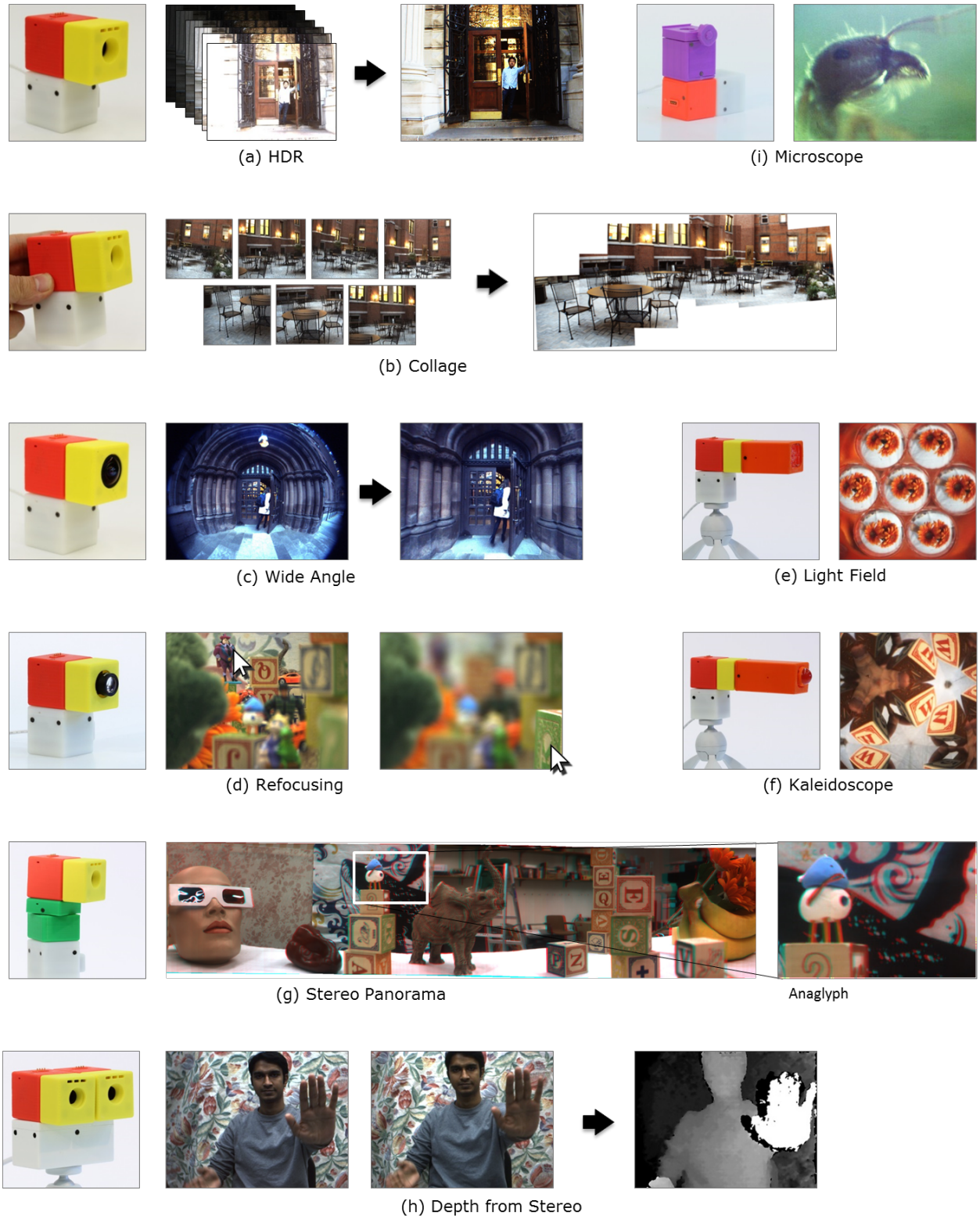


Figure 6: *Examples of results produced using Cambits*

5. Conclusion

Cambits is a versatile modular imaging system that enables the user to create a wide range of computational cameras. Our current prototype is a proof-of-concept that we have used to demonstrate the key attributes of Cambits – ease of assembly, self-identification and diverse functionality. Using our current implementation, we have shown that Cambits can be a powerful platform for computational photography, enabling the user to express their creativity along several dimensions. An important aspect of Cambits is that it is design to be an open platform that is scalable. One can add several other hardware blocks such as structured light sources, multispectral sources, telescopic optical attachments and even non-imaging sensors for measuring acceleration, orientation, sound, temperature, pressure, etc. One can imagine developing algorithms that use such a diverse set of sensors to trigger/control various image capture and processing strategies.

Acknowledgements

This research was done at the Computer Vision Laboratory at Columbia University, while Makoto Odamaki was a Visiting Scientist from Ricoh Company, Ltd., Japan. The authors thank William Miller for designing and 3D printing the chassis of the Cambits blocks, Wentao Jiang for his contribution to the user interface, and Daniel Sims for editing the demonstration video. Divyansh Agarwal, Ethan Benjamin, Jihan Li, Shengyi Lin and Avinash Nair implemented several of the computational photography algorithms. The authors thank Anne Fleming for proofreading this paper.

Bibliography

[1] https://www.ricoh.com/r_dc/gxr/
[2] https://opc.olympus-imaging.com/en_sdkdocs/index.html
[3] http://www.red.com/products
[4] Adams, Andrew, et al. "The Frankencamera: an experimental platform for computational photography." ACM Transactions on Graphics (TOG). Vol. 29. No. 4. ACM, 2010.
[5] Manakov, Alkhazur, et al. "A reconfigurable camera add-on for high dynamic range, multispectral, polarization, and light-field imaging." ACM Transactions on Graphics 32.4 (2013): 47-1.
[6] Yim, Mark, et al. "Modular self-reconfigurable robot systems [grand challenges of robotics]." Robotics & Automation Magazine, IEEE 14.1 (2007): 43-52.
[7] Schweikardt, Eric, and Mark D. Gross. "roBlocks: a robotic construction kit for mathematics and science education." Proceedings of the 8th international conference on Multimodal interfaces. ACM, 2006.
[8] High Speed USB Platform Design Guidelines Rev. 1.0
http://www.usb.org/developers/docs/hs_usb_pdg_r1_0.pdf
[9] Debevec, Paul E., and Jitendra Malik. "Recovering high dynamic range radiance maps from photographs." ACM SIGGRAPH 2008 classes. ACM, 2008.
[10] Reinhard, Erik. "Parameter estimation for photographic tone reproduction."Journal of graphics tools 7.1 (2002): 45-51.

[11] Nomura, Yoshikuni, Li Zhang, and Shree K. Nayar. "Scene collages and flexible camera arrays." Proceedings of the 18th Eurographics conference on Rendering Techniques. Eurographics Association, 2007.
[12] Schneider, D., E. Schwalbe, and H-G. Maas. "Validation of geometric models for fisheye lenses." ISPRS Journal of Photogrammetry and Remote Sensing 64.3 (2009): 259-266.
[13] Georgiev, Todor, et al. "Spatio-Angular Resolution Tradeoffs in Integral Photography." Rendering Techniques 2006 (2006): 263-272.
[14] Ng, Ren, et al. "Light field photography with a hand-held plenoptic camera." Computer Science Technical Report CSTR 2.11 (2005): 1-11.
[15] Zhou, Changyin, Daniel Miau, and Shree K. Nayar. "Focal sweep camera for space-time refocusing." (2012).
[16] Peleg, Shmuel, and Moshe Ben-Ezra. "Stereo panorama with a single camera." Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.. Vol. 1. IEEE, 1999.