Hacking A Consumer DSLR Camera for Computational Imaging

Meredith Burkle
Stanford University
mnburkle@stanford.edu

Ned Danyliw
Stanford University
edanyliw@stanford.edu

Sam Girvin
Stanford University
sgirvin@stanford.edu

Abstract

We built a hardware and software system for controlling a Canon EOS T5i and attached EF-S 18-55mm IS II lens. This system enables users to remotely program the camera to take a series of exposures with configurable settings. Via the addition of some custom external hardware, direct control of focus and the position of the lens stabilization element are also controllable. This is a highly flexible system with many possible uses, but in this paper we demonstrate depth-invariant image capture via single-exposure focal sweep and super-resolution via subpixel shifting sequential image capture.

1. Introduction

Computational imaging is a rapidly growing field that promises to boost image quality, reduce camera size, and even capture new types of information through the use of digital image processing techniques and innovative sensor design. These state-of-the-art improvements are primarily made possible by engineering the entire image capture pipeline, controlling both the camera hardware as well as underlying software algorithms. Thus, there is a dependency on custom hardware that makes it difficult and expensive to enter into this area and test potential ideas. The goal of our project was to lower some of these barriers to entry by allowing researchers to easily modify off-the-shelf DSLR hardware for computational imaging tasks.

Even without modification, consumer-level digital cameras come equipped with high quality digital image sensors and electronically actuated optics, making them an attractive development platform for computational imaging researchers. The difficulty is that no consumer camera allows full control over all of its available hardware, thus limiting its use for computational photography. However, by hacking a standard lens, the user can intercept the hardware within the image capture pipeline, allowing them to engineer custom point-spread functions (PSFs) or capture more information from the scene than previously possible.

This paper describes modifying a Canon EF-S 18-55mm IS II lens to be used for computational photography. This lens is the standard kit lens offered with several Canon cameras including the Rebel T5i, also used in this paper. The lens can be purchased for under $200 and offers electronic focus control as well as an internal image stabilization system, making it a great candidate for hacking. By making these actuators externally controllable, multiple computational imaging techniques become possible with a consumer DSLR including depth invariant capture using focal sweep techniques, super-resolution, and light field imagery which are all demonstrated in the following sections.

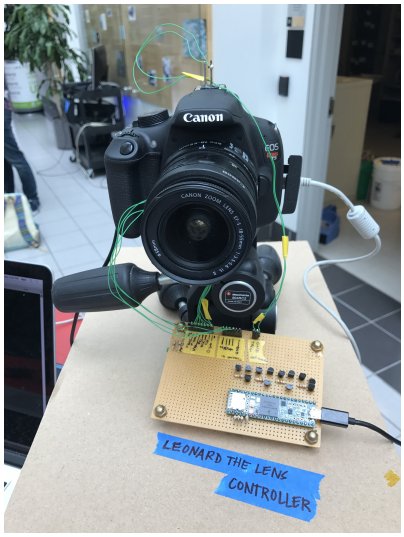

Figure 1: Final camera system

2. Related Work

External focus control for DSLR hardware has been previously developed by Bando, *et al*. [1][4] for computational imaging applications. In that work, they demonstrate the optimality and use of single-exposure focal sweep techniques to capture depth- and 2D motion-invariant images. They gain control of the focus by blocking signals coming from the camera and sending their own commands to the lens with an external microcontroller.

The use of a lens-based image stabilization systems for computational imaging was demonstrated by McClosckey, *et al*. in [2]. Lens-based image stabilization is accomplished by an electronically-controllable floating glass element in the lens. The glass element is attached to coils that, when actuated, move the lens in the plane perpendicular to the optical axis. They modified a Canon EF 70-200mm f/4 L IS USM lens to drive the image stabilization coils externally, using the lens' existing coil position sensors as inputs to a PID controller. They demonstrate dynamic control of the image stabilization element and use it to generate custom motion-based point spread functions for motion invariant image capture and simulation of a cross filter. Their lens modifications disabled aperture control and require a computer to trigger the shutter.

3. Hacking the Lens

3.1. Hardware component

The hardware system consists of a Teensy microcontroller, a level-shifter and H-bridge circuit to drive the image stabilization coils, and a modified camera lens. At a high level, the system uses the external microcontroller to bypass the camera and lens' native hardware to allow for remote control of focus, lateral movement of the image stabilization element, and camera settings.

3.1.1 Focus control

To control the focus, we used the method outlined by Bando in [3]. The goal is to stop commands from the camera from reaching the lens and replace those commands with those piped in from our microcontroller. In order to do this, we removed the lens mount and soldered lines to the the pins that normally get driven by the camera body. The pads that typically contact the body of the lens are electrically isolated with Kapton tape.

Figure 2: Focus modification

By soldering to the center of the pins, rather than directly to the pads, the lens can returned to normal operation by simply removing the Kapton tape and disconnecting the microcontroller. Lens commands are generated by a Teensy 3.5 microcontroller, using code from Bando *et al*. that has been ported from Arduino to the faster, more powerful Teensy. The lens communicates over a 100kHz SPI interface, MSB first.

3.1.2 Image stabilization element control

Control of the image stabilization element is slightly more involved. The floating element is held in the center of the optical axis with three springs, and two coils control the relative position of the lens element, one in the horizontal direction and one in the vertical direction. Image stabilization commands are sent to the coils via a drive IC on the main PCB of the lens. After desoldering the drive IC, the image stabilization coils can be controlled directly by soldering lines to test pads on the PCB. The coils are driven with 30kHz 5V PWM signals, and require about 315mA current to reach the full range of motion. The Teensy can't source this much current directly, so a level shift and H-bridge are necessary drive the coils. Applying a PWM signal to one end of a coil while holding the other end to ground moves the element in one direction, and switching which end of the coil is driven switches the direction of motion.

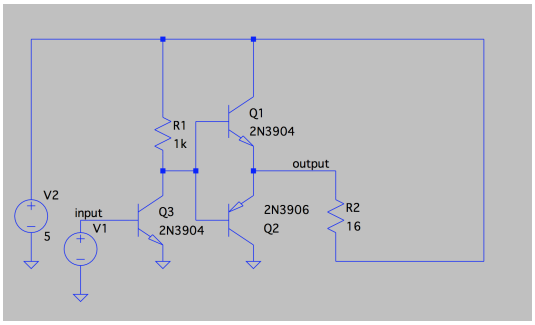

Figure 3: The coil drive circuitry. Each leg of the image stabilization coils is driven by one of these circuits

3.2. Software component

To interface with the customized lens, we implemented the following imaging pipeline shown in Figure 5. The Python framework allows queuing multiple capture events and camera settings and handles the communication between the camera, computer, and lens controller board allowing us to write and run tests quickly. The captured images are then saved to the computer's hard drive and processed in MATLAB.

The Python testing framework works by queuing up captures and then sending the commands through a Python hardware abstraction layer (HAL) which splits the appropriate actions between the lens controller and commands

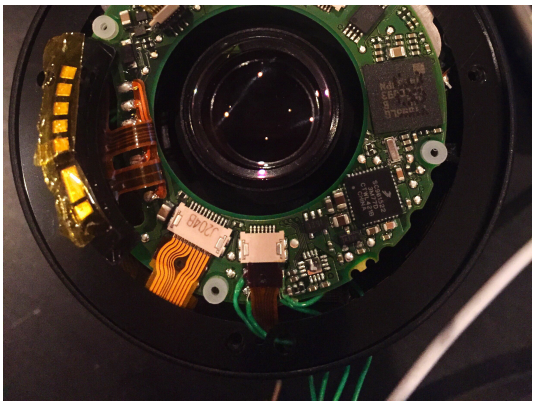

Figure 4: Detail of coil modification. Note the removed controller.

sent to the camera itself over USB. The camera interface is built on top of gphoto2 which allows for the control over most of the camera's options over USB including taking and downloading images from the camera. Through this interface we can control parameters such as exposure, ISO, and aperture. The HAL library communicates with the lens controller over a serial link with a custom designed command structure which allows us to command events to the camera for immediate or triggered execution. This structure allows events to occur before and during the capture event enabling PSF engineering for computational imaging algorithms. The command structure is shown below:

[*trigger cmd*] [*action cmd*] [*action arg*]

Each command can be variable-length depending on the expected arguments and are decoded through a state machine running on the lens controller described in more detail below. The trigger commands can range from "execute immediately" to "execute after exposure". Currently we have implemented only triggering immediately and right after image capture begins but have the infrastructure in place to allow for more complex triggering such as 5 milliseconds after exposure begins.

The firmware running on the lens controller was executed on a Teensy v3.5 microcontroller. The controller handled parsing the commands sent from the computer over a serial link and taking the appropriate actions. The firmware can handle queuing actions up for execution after detecting a new exposure beginning (sensed through the camera's flash hotshoe. The controller then handles driving the lens' internal image stabilization system by commanding the appropriate PWM duty cycles to position the lens element in the desired position. To adjust other lens settings like the auto-focus mechanism, the microcontroller acts as the camera and sends the appropriate SPI commands to the lens directly.

3.3. Hardware characterization

After developing the hardware and software to control it, we ran tests to determine the limits of the hardware.

3.3.1 Measuring the speed of the focal sweep

The maximum speed of the focal sweep is an important parameter to measure, because it determines the minimum exposure time for a focal sweep image.

We measured the time to move from infinity focus to close focus and vice versa by recording the sound of the focus motor. Moving the lens from infinity to close focus takes about 0.375 seconds (Figure 6), while changing focus from close to infinity takes about 0.4 seconds (Figure 7).

The camera can only be set to a set of discrete exposure lengths, so the minimum exposure time for a focal sweep image using the EF-S 18-55mm IS II lens is 0.4s.

3.3.2 Image stabilization control

First, we characterized the image stabilization control by measuring the stability of the system, taking a series of images at a specified (non-zero) image offset and measuring the difference images (Figure 8). The difference image was essentially black, implying subpixel position stability over time.

For control purposes, we also measured distance that the lens moves per discrete PWM drive level. Driving the lens with 0%, 50%, and 100% duty cycle PWM in each direction, we imaged a dot at each level and examined the difference image (Figure 9). The duty cycle to image movement relationship can be approximated as linear with 3.13% duty cycle (4 PWM levels) corresponding to a 1 pixel movement in the image.

4. Computational Imaging

In addition to the hardware system and the software to control and command that system, we also implemented a variety computational imaging algorithms in MATLAB that demonstrate the current capabilities of our lens control system and pave the way for engineering custom PSFs.

4.1. Focal Sweep

The focal sweep method sweeps a plane of focus by moving the lens along the optical axis throughout the exposure. This method results in an image that is blurry, but has been blurred by a PSF that is invariant to object depth, which then bypasses the need for PSF identification to perform defocus deblurring. The focal sweep method has also been shown to make motion blur invariant to object speed and in-plane motion direction as referenced in Bando *et al.* [1][4], so being able to perform focal sweep image capture

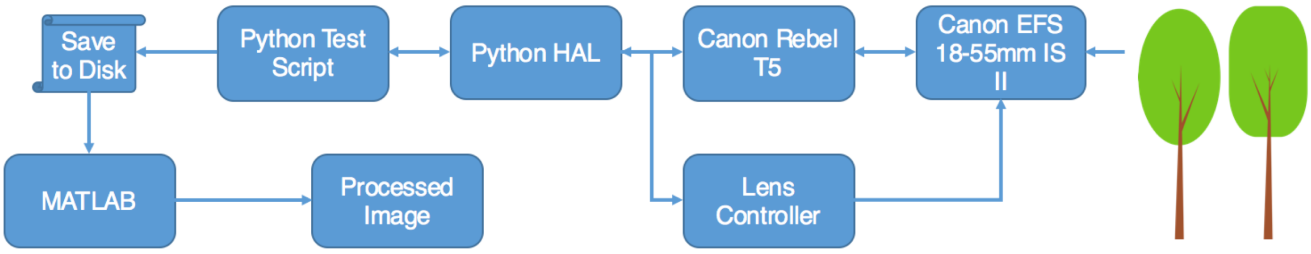

Figure 5: Image capture pipeline

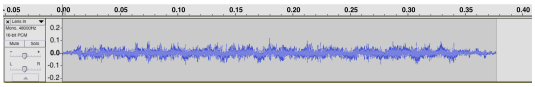

Figure 6: Audio recording of lens sweep from infinity to zero.

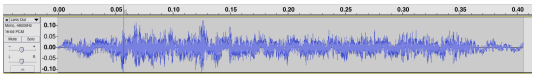

Figure 7: Audio recording of lens sweep from zero to infinity.

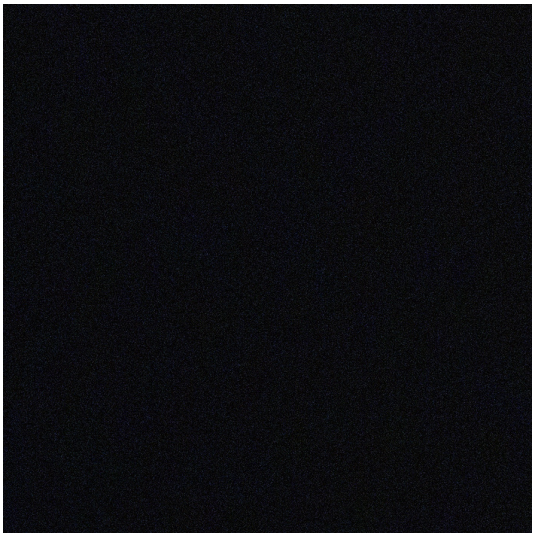

Figure 8: Difference image of 10 consecutive images measured at the same lateral lens displacement.

with a consumer lens can provide large potential improvements.

4.1.1 Image Capture

In order to properly deconvolve the focus swept image, we first need to characterize the focus sweep PSF. This was

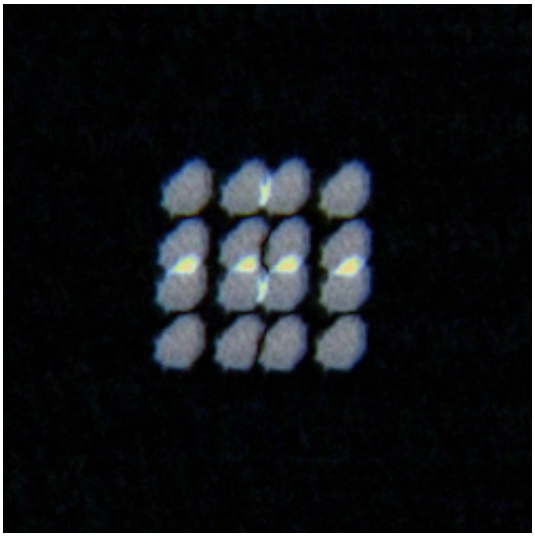

Figure 9: Grid demonstrating image shift due to lateral lens movement.

done using a point source generated with a pinhole lens (shown in Figure 10) and capturing the image while sweeping the focus during exposure. A custom PSF was fit to the image in Matlab to reduce the effect of measurement errors and noise. This was done by modeling the PSF as a series of added Gaussians and fitting the best response to the measured data.

To evaluate the depth-invariance of the PSF we performed another test where the point source was moved to different distances and PSF of a focal sweep was measured at each point.

After acquiring the captured PSF, we created a scene to showcase the depth invariance achieved by performing a focal sweep during exposure. This scene consisted of objects at various depths such that they were out of focus when captured during a normal exposure. Using deconvolution with ADMM we can recover an “all-in-focus” image of the scene.

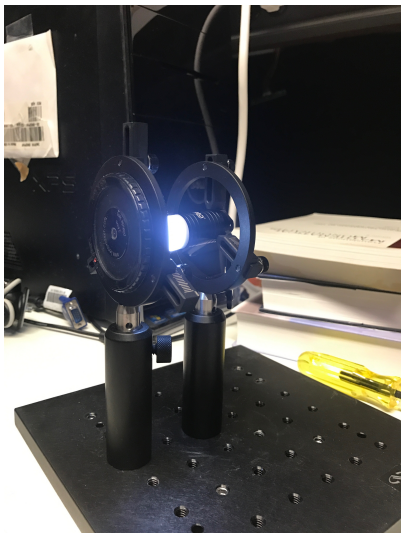

Figure 10: Point source using a pinhole lens.

4.1.2 Algorithm

In order to perform defocus deblurring, we used the additive Gaussian PSF acquired through measurement and modeling. Given a point spread function and a blurry image, there are many possible algoirthms we can use to reconstruct the deblurred image. We primarily use ADMM and a TV Prior to reconstructed a deblurred estimate for the image by 1) converting the RGB image to YCrCb, 2) performing ADMM + TV on each channel (Y, Cr, Cb) separately, and then 3) concatenating each channel together. This method was ideal for experimenting as we could choose different hyper-parameters for each channel. The code to test and run this algorithm is located in the `simulation` folder in `admm_script.m`. Within the code, you can also point the script to different file directories and adjust parameters controlling the ADMM + TV algorithm.

4.2. Super-resolution

Another computational imaging feature we've implemented is super-resolution, which uses images taken at subpixel shifts to reconstruct an image of a higher resolution. This allows capturing an image at a higher resolution than the sensor could capture.

4.2.1 Image Capture

To capture the images used for super-resolution, we captured a set of four images of the same static scene with subpixel shifts between each image. Each picture was captured with a half-pixel shift in a box pattern. To test the effectiveness of the technique we captured images of a test pattern with a range of spatial frequencies as well as a scene with a variety of high frequency components.

4.2.2 Algorithm

Using the lateral lens movement, we are able to construct an image double the resolution of the originally captured images. Through hacking the image stabilization element, we are able to take multiple images of the same scene with sub-pixel shifts. By capturing the scene at 4 points, we can think of these as 4 samples of a higher resolution image. We can then use gradient descent and interpolation to reconstruct the an estimate of a higher resolution image based on the 4 "samples." This code is in the `simulation` folder and can be tested and run using the `superresolution_script.m` file. Within the code, you can also test different image directories and choose particular 256×256 blocks of an image to run the super resolution on, .

5. Evaluation/Results

The results from our demonstrated computational imaging methods, focal sweep and super resolution, are shown below. We chose to implement these methods as they are indicative of the capabilities of our lens control system.

5.1. Focus Sweep

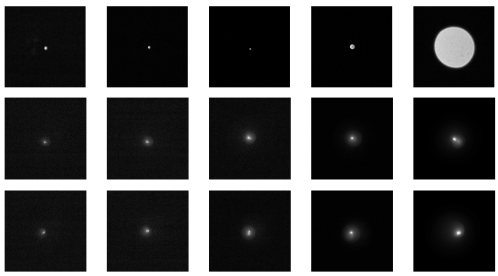

Figure 11: Depth invariant PSFs. Image was focused at center point at 1.8m. The distances are from left to right - 3.3m, 2.55m, 1.8m, 1.05m, and 0.3m. The top row is the lens PSF, second row is focus sweep from infinity to zero, and the bottom is focus sweep zero to infinity.

For the focal sweep image, we a PSNR of around 17 dB, depending on the λ and ρ parameters set for each channel (RGB or YCrCb). The final deblurred image along with the original focal sweep image are shown in Figure 12 and 13. In our final image, we noticed that there was still a slight haze around some of the text and lines, which could be due to the fact that our modelled PSF doesn't quite match that of the particular lens we are using. That said, we took focal sweep images sweeping from close to far as well as far to close, and both images result in a PSNR of 17 dB which further corroborates our lens control system's ability to capture images with depth-invariant PSFs.

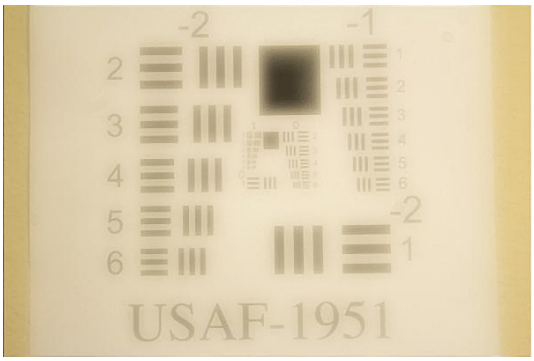

Figure 12: Deblurred focal sweep image

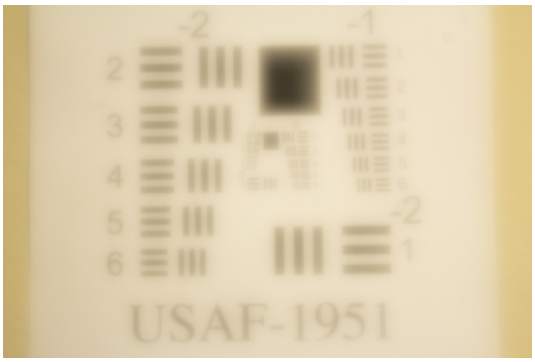

Figure 13: Original focal sweep image

5.2. Super-Resolution

For the super-resolution image, evaluation is slightly more involved. This is due to the fact that there is no "ground truth" image of a higher resolution with which to compare our super-resolution result. Because we did not have a reference super-resolved image to use in order to report the PSNR and mean-squared error, we instead looked at the spatial frequencies present in the test pattern that we used. The results are shown in Figures 14 and 15, where

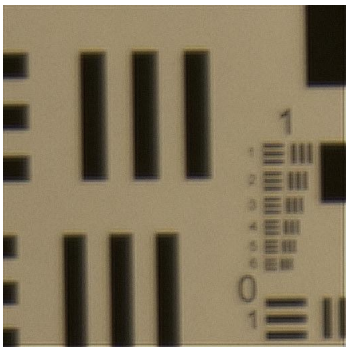

Figure 14: Super-resolved USAF test image

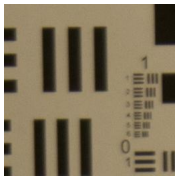

Figure 15: Original USAF test image

6. Future Work

6.1. Lightfield

Taking a set of images at a different lateral offsets, it is possible to generate a lightfield of a static scene. Each image corresponds to a slightly different perspective of the scene, which encodes both the intensity and angle of incoming light. Shifting and adding the different viewpoints can be used to generate different depths of field.

We captured all of the images necessary to generate a lightfield, but the parallax between the constituent images was not large enough to generate any significant lightfield effects.

6.2. Other

Other possibilities for future applications include the generation of custom filters with engineered point spread functions, motion blur deconvolution, and more sophisticated implementations of depth invariant capture and super resolution. Another interesting application could be coded focal stack photography as performed by Lin *et al.* [3] which opens up possibilities for photography with non-planar focal surfaces and multiplexed focal stack acquisition, as well as full-resolution depth and all-in-focus images using sparse coding techniques.

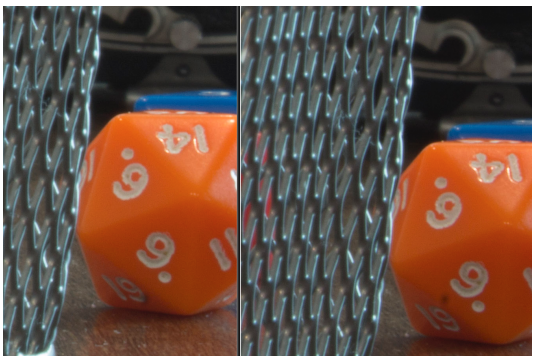

Figure 16: A detail of the kind of parallax possible between different views in the lightfield. The parallax effect is very slight, but you can see slight overlap between the metal grating and the die.

7. Conclusion

Overall, we were able to create a robust system with functioning hardware, firmware, and software that is able to control the lens along multiple axes, allowing for a variety of computational imaging tasks. Although we have only demonstrated super-resolution and focal sweep within this paper, there is a potential for engineering customized PSFs which would greatly increase the capabilities for testing and implementing endless computational tasks.

8. Acknowledgements

Thanks to Gordon Wetzstein for mentoring our project and providing equipment, Yosuke Bando for providing initial focus control code, and Keenan Molner for help with the optical setup for PSF measurement.

9. Appendix

Listing 1: Demo Python test script.

```
from camera import EE367Camera
from image_capture import ImageCaptureQueue
from image_capture import ImageCapture

ISO = '100'
EXPOSURE = '0.5'
APERTURE = '5.6'

# displacement per pixel
dx = 4
dy = 4

comport = '/dev/tty.usbmodem2519941'
cam = EE367Camera(comport)

# set camera parameters
cam.set_exposure(EXPOSURE)
cam.set_aperture(APERTURE)
cam.set_iso(ISO)

# Create ImageCaptureQueue
test_set = ImageCaptureQueue(savedir='superres',
                             dry_run=False)

# Capture image at (1/2, 1/2)
image0 = ImageCapture(cam)
image0.add_action(cam.set_initial_pos, int(dx/2),
                  int(dy/2))

# Add to test queue
test_set.add_capture(image0)

# run test set
test_set.run()
```

References

[1] Y. Bando. An analysis of focus sweep for improved 2d motion invariance, 2006. ECCV06 submission ID 324. Supplied as additional material `eccv06.pdf`.

[2] S. McCloskey, K. Muldoon, S. Venkatesha. Motion invariance and custom blur from lens motion. *ICCP*, 2011.

[3] X. Lin, J. Suo, G. Wetzstein, Q. Dai, R. Raskar. Coded focal stack photography. *ICCP*, 2013.

[4] Y. Bando, H. Holtzman, R. Raskar. Near-invariant blur for depth and 2d motion via time-varying light field analysis. *TOG*, 2013.