Characterization of camera shake

Henry Dietz, William Davis, and Paul Eberhart; University of Kentucky; Lexington, Kentucky

Abstract

In the early days of photography, emulsions were not very sensitive to light and lenses had relatively small apertures, so long exposures were needed and cameras were generally mounted on solid, stationary, supports. However, in modern use, cameras are nearly always hand-held – and this introduces shake. Vibrations also are introduced by the complex moving systems within a camera and lens.

Although many cameras now incorporate mechanisms for minimizing the detrimental impact of shake, and there is a standard test procedure to measure effectiveness of such measures, there is surprisingly little published on the characterization of camera shake itself. The current work describes how inexpensive shake measurement hardware can be built, proposes a testing methodology for characterizing shake, and summarizes preliminary results obtained by measuring shake under a variety of conditions.

Introduction

Camera shake during exposures always has been a problem for photography.

For the first century and a half of photography, rigid tripods or large and heavy dollies were the primary ways to prevent shake. In the early 1970's, Garret Brown invented the *Brown Stabilizer*, which became the commercial product known as *Steadicam*[1]: a mobile, but bulky, device that uses a high inertial mass to dampen movements. More recently, various methods have been developed that allow active cancellation of shake. Canon and Nikon started selling lenses incorporating *optical image stabilization* (OIS) in the mid 1990s and Minolta introduced sensor-moving *in-body image stabilization* (IBIS) in the DiMAGE A1 in 2003; both technologies are now in common use. DJI has even implemented a *tripod mode* in drones that combines precision hovering with an intelligently-controlled camera gimbal. Of course, there are also computational methods for reducing motion blur by deconvolution, intelligent fusing of multiple images, etc.

To improve upon these methods, and to develop new methods, it is necessary to measure the shake a camera encounters in actual use. Clearly, every camera supporting OIS or IBIS implements some method for measuring shake, but the data from these mechanisms are generally not accessible to a user of the camera. CIPA, the Camera & Imaging Products Association, established a standard for testing the effectiveness of image stabilization systems in cameras[2], but the testing is about subjecting a camera to a reference vibration pattern (sampled at 500Hz), not measuring shake as the camera is actually used.

There are a wide variety of methods that could be used to measure camera shake, and several are described in the open literature.

It is possible to use analysis of captured images to determine camera movement. Photographing a static, in-focus, point target captures the PSF (point spread function) produced by camera motion during the exposure, but this "blur kernel" only describes the sum total effect of motion during the exposure. There are a variety of methods described for using a fixed camera to measure movement of an object[3][4][5], and these methods could be adapted to measure camera shake with a fixed subject. The key problem is that a static scene generally does not provide any insights about the time sequence of movements during a single exposure.

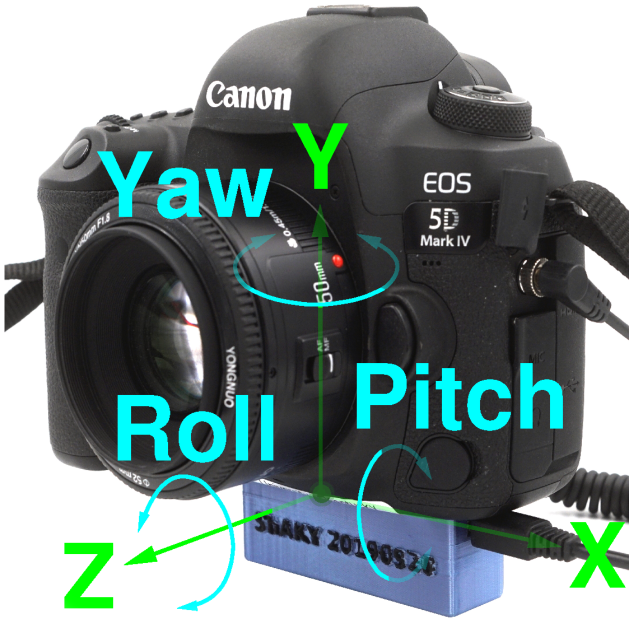


Figure 1. *Measurement orientations for ShAKY on a Canon 5DIV*

Using a dynamic target that incorporates multiple points (or scene objects) varying over time in a known pattern can provide motion sequence data. In fact, flickering dynamic targets have long been used for tasks like measuring consistency of shutter curtain speed in focal-plane shutters or determining precise time offsets between captures made by multiple cameras. By using a target array in which individual targets can flash in an arbitrary time-sequenced pattern, the relative positions of individual targets within a captured image can accurately encode the camera orientation at the moment of each flash. For example, a blinking LED array[6][7] can be used to sample motion during an exposure. Unfortunately, target complexity limits the total number of samples, focal-plane and electronic rolling shutter complicate pattern choice by having different regions of the sensors exposed during different intervals within an image capture, and the movement data recovered essentially has only two independent dimensions: sensor X and Y. Additionally, if OIS or IBIS is enabled while measurements are being taken, their performance is convolved with the measurements – although pairing this technique

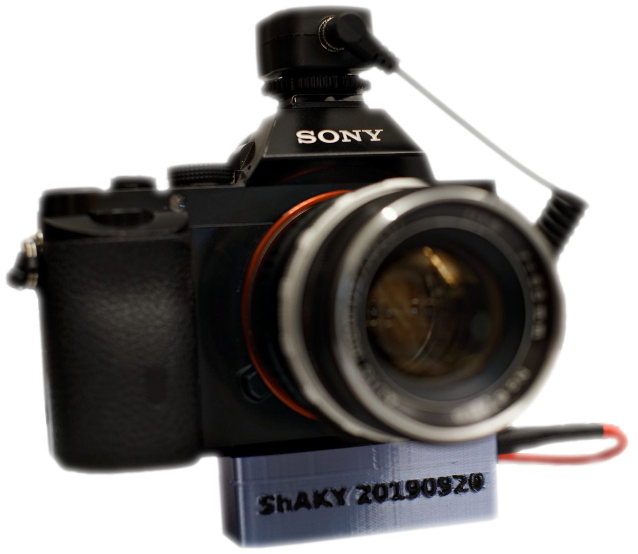


Figure 2. *ShAKY on a Sony A7.*

with another method could be used to isolate and study OIS or IBIS behavior.

A potentially better method for measuring camera shake is to directly measure movement using an IMU (inertial measurement unit). These devices commonly incorporate accelerometers, gyroscopes, and magnetometers, and can measure position, orientation, and possibly other properties. Safaee-Rad and Aleksic discuss how they measured shake for cell-phone cameras[8], which involved using the IMU built-into the cell phone. Unfortunately, except in some smartphones, camera-and-lens systems that contain IMUs generally do not provide an interface for accessing the position and orientation tracking data. Thus, an external IMU offers a more general-purpose solution and various research efforts[9][10] have used external IMUs for diverse purposes including computational image deblurring, user identification, camera control, and camera stabilization. In general, the biggest issues with IMUs concern calibration and filtering methods for minimizing sample noise while sustaining a fast sample rate. Typical filtered sample rates are below 60Hz, but the goal here is to at least match the 500Hz sample rate used by CIPA's reference waveforms.

After finding no low-cost commercial device suitable for directly attaching to a camera, we decided to create our own open-source hardware and software IMU-based system. It was designed to be a device that we and others could cheaply and easily replicate as a general-purpose tool to facilitate research toward better understanding and managing camera shake. That device, ShAKY, is shown in Figure 1 attached to a Canon 5DIV DSLR and in Figure 2 attached to a Sony A7 mirrorless camera. This IMU-based system can measure 6-axis shake for any camera at a rate up to approximately 1000Hz. In addition to describing the device and a measurement protocol, the current work presents preliminary results using this system to characterize camera shake.

ShAKY

ShAKY, SHift Angle Kentucky, is actually a 9-axis device with a complete build cost of less than $20, and it easily can be

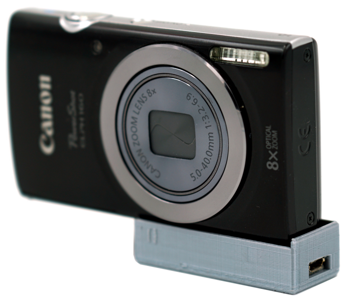


Figure 3. *Original ShAKY prototype on a Canon PowerShot ELPH 160*

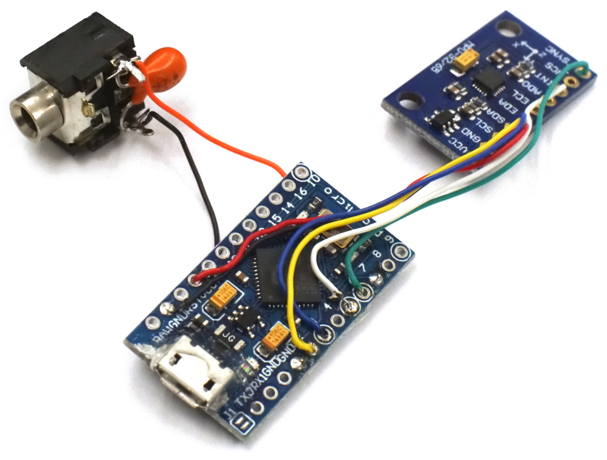
built in a single afternoon. Of course, it took us quite a bit longer than that because the unit described here is the result of refining the design through a series of prototypes. Issues that were discovered and addressed in earlier ShAKY prototypes included:

- There was a different housing design for each different camera shape (Canon PowerShot ELPH 160/180, Canon 5D IV, Sony A7, etc.). The goal was to be unobtrusive while placing the IMU as close as possible to the ideal position on the camera, as shown in Figure 3. However, the current design fits many different camera models without blocking the battery door, etc., on most and still positions the IMU comparably well.
- High-quality sampling rate was limited by processing overhead and formatting data for USB transmission to around 30Hz. By splitting processing between ShAKY and the USB host, the latest version can sustain high-quality sampling at more than 350Hz.
- There were difficulties in aligning shake data with timing of the shutter. We originally tried to deduce when the shutter fired from the reported waveform, but now can detect a synchronization signal and place a mark in the data stream. An appropriate synchronization signal can be generated by the X flash sync terminal of most cameras.

ShAKY Electronics

It used to be that devices like accelerometers were physically large and fairly expensive, but MEMS (Micro Electro-Mechanical Systems) technology has made them tiny and cheap. In ShAKY, the actual measuring is done by a MPU-9250[11]: a 3mm$\times$3mm multi-chip module, which contains two dies. One die implements both a 3-Axis gyroscope that measures roll, pitch, and yaw motion, and a 3-axis accelerometer that measures X, Y, and Z acceleration; both types of measurements are digitized using 16-bit ADCs. The other die is an AK8963 3-axis magnetometer which measures absolute orientation relative to the Earth's magnetic field – a digital compass with 14-bit resolution. The magnetometer can be used to correct for constant velocity movement and accumulated errors over a sequence of readings, but readout is significantly slower at just 8 samples per second, while the accelerometer is capable of 4000 and the gyroscope 8000.

Rather than buying the MPU-9250 as an isolated chip, it is available in the more-manageable form of a boardlet designed to easily be interfaced to an Arduino. We used an MPU-9250/GY-9250 9-axis inertial motion unit module, which cost about $8.50.



Pro Micro Vcc	MPU-9250 Vcc
Pro Micro Gnd	MPU-9250 Gnd
Pro Micro 2	MPU-9250 SDA
Pro Micro 3	MPU-9250 SCL
Pro Micro 5	MPU-9250 INT
Pro Micro 6	MPU-9250 FSYNC
Pro Micro 7	3.5mm Plug Outer
Pro Micro Gnd	3.5mm Plug Tip

Figure 4. *Wiring pattern for ShAKY.*

A second boardlet in ShAKY provides an Arduino processor and USB communications. We used an ATmega328 Arduino Pro Mini that cost about $5.60. The Arduino controls the MPU-9250, collects data from it, and streams that data, along with a timestamp and a synchronization marker, to a host computer via USB.

The third component in ShAKY is a jack for a 3.5mm two-conductor plug to provide a synchronization signal. This connector is commonly used for the flash end of a X flash sync cable. X flash sync in older systems tends to use a mechanical contact to connect the two wires while newer systems use electronic switching that imposes a polarity – ShAKY is designed to work with either. Some cameras need a hot-shoe adapter for an Xsync cable (as seen in Figure 2). Mechanical contacts can bounce, so a capacitor is placed across the jack outputs to debounce the signal.

The two boardlets and jack are easily wired together, as shown and described in Figure 4. The USB connector of the Arduino Pro Mini serves both to power the unit and to report shake measurements to a host computer.

ShAKY Software

Even with a chip as sophisticated as the MPU-9250, accurately tracking position and orientation requires fairly complex processing in software. The complexity comes primarily from the fact that both the accelerometers and gyroscopes give what is essentially incremental data that must be integrated in software, but there also are issues with noise, calibration (which is also sensitive to gravity), coupling of channels due to an offset mounting position, etc. The magnetometer can be used to help dynamically calibrate and correct for drift, but only occasionally – as its low 8Hz maximum sample rate permits.

Arduino software to extract sufficient accuracy from a MPU-9250 to aim a telescope to track planets is freely available[12], but the sample rate that software can sustain is two orders of magnitude below the raw readout rate of the accelerometers and gyroscopes. Tracking errors accumulate over time, but the critical tracking period is determined by the exposure time, and in such short intervals tracking error from even our simplest, fastest, processing is acceptable. However, we prefer to use software that gives very accurate results, yet runs fast by splitting the required processing between the embedded Arduino and the host computer connected via USB. In the worst case, the preferred version of our software samples at more than 350Hz.

Software inside ShAKY. The code running inside ShAKY simply controls the MPU-9250 and streams raw data samples over USB to the host computer. I2C communication with the MPU-9250 and USB transmission to the host are facilitated by libraries, and our original version was implemented in fewer than 100 lines of source code... but gave wildly inconsistent readings. The fast high-quality version we developed and now use employs calibration and filtering algorithms derived from those published by Madgwick[10], but splits the processing between the Arduino and USB host in order to sustain a higher data rate. Inside ShAKY's Arduino, it takes approximately 1100 lines of source code to perform the calibration and package and transmit data via USB.

When first connected, ShAKY awaits a single-character request code, then outputs a text string identifying itself, e.g., `ShAKY20200207\n`. It will then run a calibration sequence to compute bias and scale factors for the magnetometer and report those along with scale ranges for the accelerometer, gyroscope, and magnetometer. When calibration is done, all of that data is sent as human-readable ASCII text. This encoding is used because transmission speed of that data is not critical and correct functioning of the unit can thus be confirmed using any serial terminal to monitor ShAKY.

Actual samples from the IMU are encoded as raw binary data to minimize transmission time. The accelerometers and gyroscopes can all sustain a similar sample rate, but the magnetometer is much slower. Thus, the data is streamed over USB to the host in variable-size records that include magnetometer data only when an update is available. Each record transmitted over USB contains between 9 and 21 bytes, with the following structure:

Condition	**Bytes**	**Encodes**
	flags	Xsync=32, ...
	lo, hi	Microseconds since last record
(flags&1)	lo, hi	Accelerometer Register X
(flags&1)	lo, hi	Accelerometer Register Y
(flags&1)	lo, hi	Accelerometer Register Z
(flags&2)	lo, hi	Gyroscope Register X
(flags&2)	lo, hi	Gyroscope Register Y
(flags&2)	lo, hi	Gyroscope Register Z
(flags&4)	lo, hi	Magnetometer Register X
(flags&4)	lo, hi	Magnetometer Register Y
(flags&4)	lo, hi	Magnetometer Register Z

The accelerometer, gyroscope, and magnetometer read-out as data blocks with the same internal structure, so the Arduino is simply collecting the data into a contiguous chunk of memory and issuing a single `Serial.write()` to USB. Timing between samples

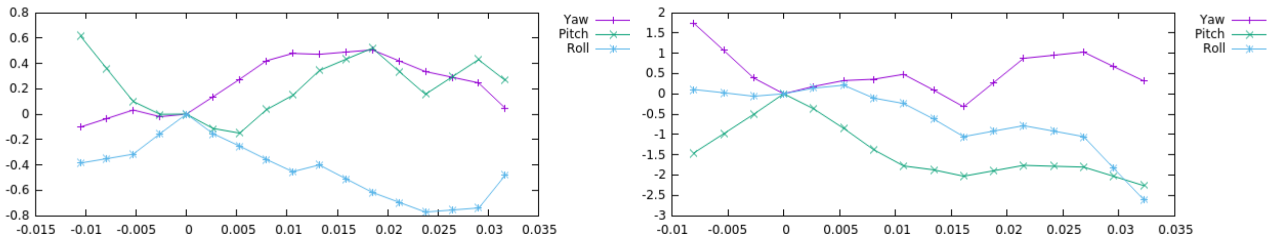


Figure 5. *Typical ShAKY data for a Sony A6500 (IBIS and EFC disabled) camera still on a tabletop (left) vs. hand-held (right).*

may vary due to the availability of new sensor data (especially the magnetometer), IMU readout delays, and USB communications with the host. The `InShAKY` software even can be told to send a separate USB packet as each type of sensor data is ready, which is how sample rates of 1000Hz or higher most easily can be achieved. In any case, the microseconds since last record field allows the precise sample timing to be recovered.

Accelerometer values are the measured acceleration in the named axis and gyroscope values are the incremental angular rotations about those axis. Of course, assignment of X, Y, and Z depends on the orientation of the sensor module as mounted on the camera, and actual acceleration is fundamentally indistinguishable from the force due to gravity. As ShAKY mounts the sensor, the IMU is sitting in what would be considered its upright position with Z pointing upward and X aligned with the axis looking out the lens. This is somewhat confusing because the usual convention for cameras is different, referring to X and Y as the sensor width and height dimensions, and Z as aligned with the lens axis. As shown in Figure 1, our software reports measurements in camera space, but does not currently automatically correct for the camera-dependent X, Y, and Z offsets between where the IMU is mounted and the center of the camera's sensor.

Figure 5 shows typical data captured when the sensor is stationary vs. hand-held. The Yaw, Pitch, and Roll are displayed in degrees/s.

Synchronization with shuttering. Although motion tracking is continuous, most applications of ShAKY need to extract the motion that occurs during image capture. Detecting the start of an exposure by merely examining the motion data proved unexpectedly infeasible; no method was found that worked consistently across different camera models. Thus, the latest ShAKY version uses the X flash synchronization signal from the camera – as it was used for the data shown in Figure 5.

When an X flash synchronization signal is detected, the current record is marked so that the host software can know that an exposure was triggered. The X sync signal does not happen at the start of an exposure, but is triggered when the shutter is fully open. For a focal plane shutter, that corresponds to the time at which the first curtain reaches the end of its travel. Empirically, that condition is commonly used to trigger the X sync signal even if the shutter speed selected is too fast for flash use. The result is that the X sync triggers with repeatable delay after the exposure interval has begun.

The fastest X sync speed quoted for the camera is a good approximation to how late the X sync signal will be. A camera using a focal-plane shutter with a 1/125s maximum X sync shutter speed will trigger X sync approximately 1/125s after exposure has begun. Shutter behavior in digital cameras is detailed in earlier work[13]; the most relevant points for the current work are:

- Nearly all focal-plane shutters use curtains that travel at a constant rate independent of shutter speed, and the same speed is simulated when a digital camera is set to use an electronic first curtain (EFC).
- Although older 135-format film cameras often used cloth horizontal-travel focal plane shutters with X sync speeds between 1/30s and 1/125s, DSLRs and mirrorless cameras generally have rigid-blade vertical-travel focal plane shutters with somewhat higher X sync speeds (e.g., high-end Canon DSLRs and Sony mirrorless cameras take less than 1/200s).
- Fully electronic (sometimes called "silent") shutter generally uses a much slower effective curtain speed, with many camera models taking 1/20s or longer to fully open.

Thus, focal-plane shutters essentially have two important speeds: the curtain-traversal time (X sync speed) and the exposure duration (shutter speed). The actual length of motion data recording for an exposure is the sum of these two numbers. For example, using a shutter with an X sync speed of 1/125s (0.008s) to capture a 1/60s (0.017s) exposure would mean that a motion trace of 0.025s duration would be needed. At ShAKY's approximately 1000Hz peak rate, the motion trace would begin 8 samples before the reading marked as X sync and continue for 17 samples.

Host software. Two main development branches of the host software exist. The first was written in MatLab, and operates on a file of ShAKY serial data captured using `cat` of the USB device. The second is written in C and has several versions, with the highest quality tracking coming from our 700-line program `hostshaky.c`, which employs Madgwick's[10] algorithms. It reads the data stream directly from the USB device into a circular buffer. If no shutter speed is specified on the command line, the C code will process and stream a continuous motion trace from ShAKY. If given the X sync and shutter speed values on the command line, it will automatically fork a parallel process to extract the samples corresponding to each image capture period and plot them using `gnuplot`.

USB packet processing and the timestamp fields both confirm a true sample rate of more than 350Hz, and even a bit above 1000Hz using incremental messages, can be maintained for extended periods. Empirically, the bottleneck is the IMU I2C communications, but USB serial drivers can be the limit if not configured for raw 230400 baud operation – which `hostshaky` does under Linux for the specified `/dev/ttyACM` port.

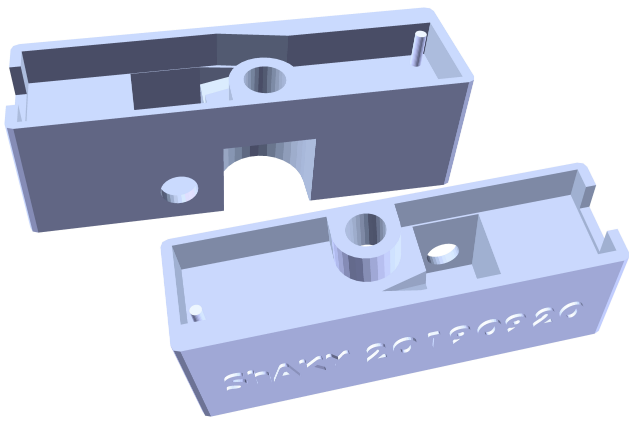


Figure 6. Front and back of 3D model of ShAKY housing.

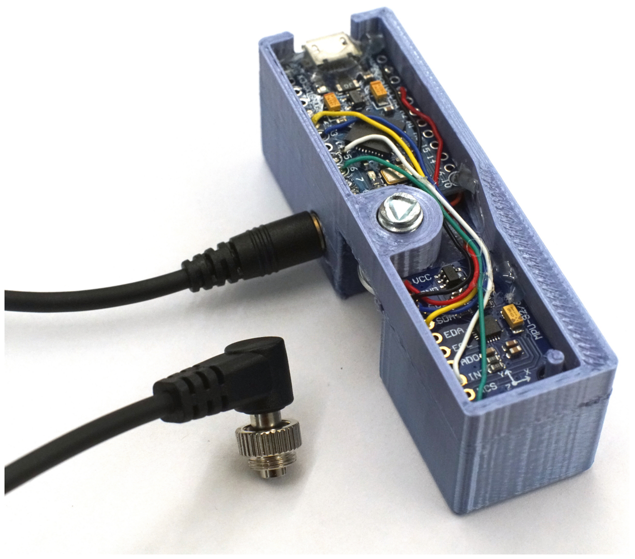
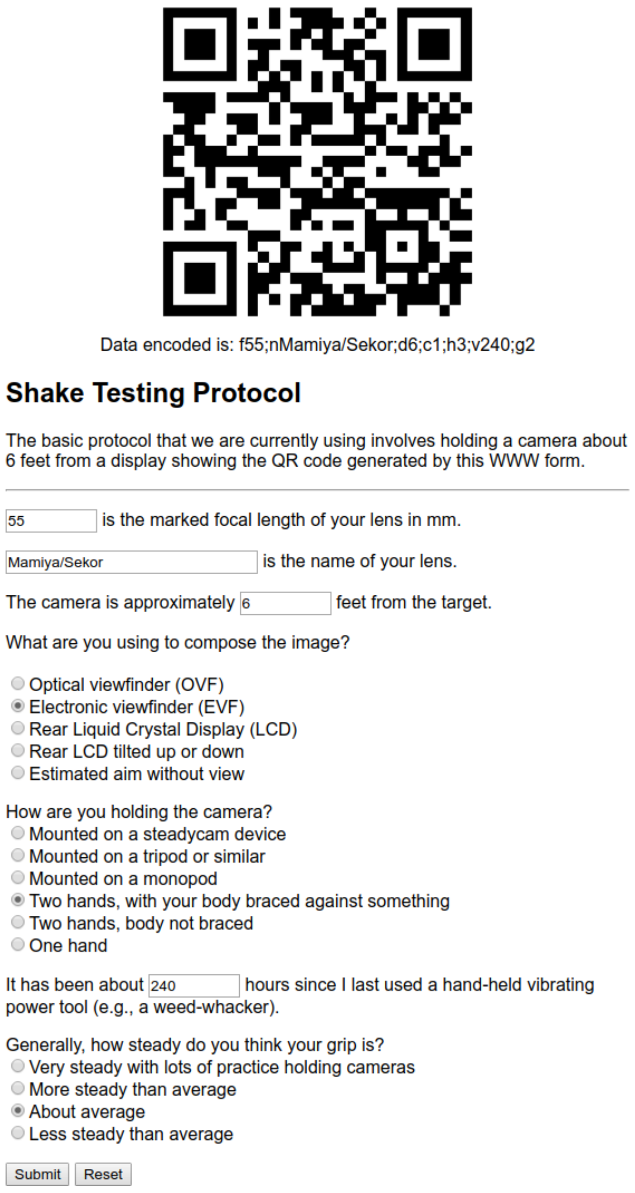


Figure 7. Circuitry inside the completed ShAKY device.

3D-Printed Housing

The 3D-printed housing for ShAKY was designed using `OpenSCAD`. Initially, versions were made as small as possible to be minimally obtrusive on the target camera, which meant there was a slightly different design for each camera model. Figure 6 shows the 3D model of current version (20190920) and Figure 7 shows the actual completed unit. This version is able to be used with most types of cameras. It attaches to the bottom of the camera using a standard 1/4-20 tripod screw.

On most cameras, the tripod socket is aligned with the X and Z center of the sensor plane, but the IMU obviously cannot be positioned in the center of the image sensor. The offsets from the ideal position depend on the camera model. Using the ShAKY20190920 housing on a Sony A7, the IMU is approximately -18.5mm X, -31mm Y, and +3mm Z from the ideal camera origin position. Although `hostshaky` does not currently correct for them, the measurements could be computationally corrected for these modest known offsets.



Data encoded is: f55;nMamiya/Sekor;d6;c1;h3;v240;g2

Shake Testing Protocol

The basic protocol that we are currently using involves holding a camera about 6 feet from a display showing the QR code generated by this WWW form.

55 is the marked focal length of your lens in mm.

Mamiya/Sekor is the name of your lens.

The camera is approximately 6 feet from the target.

What are you using to compose the image?

- Optical viewfinder (OVF)
- [✓ Electronic viewfinder (EVF)]
- Rear Liquid Crystal Display (LCD)
- Rear LCD tilted up or down
- Estimated aim without view

How are you holding the camera?

- Mounted on a steadycam device
- Mounted on a tripod or similar
- Mounted on a monopod
- [✓ Two hands, with your body braced against something]
- Two hands, body not braced
- One hand

It has been about 240 hours since I last used a hand-held vibrating power tool (e.g., a weed-whacker).

Generally, how steady do you think your grip is?

- Very steady with lots of practice holding cameras
- More steady than average
- [✓ About average]
- Less steady than average

Figure 8. ShAKY testing protocol form and QR code.

Measurement Procedure

Although there are many ways in which ShAKY could be used to obtain shake data for cameras (or other devices), it is useful to establish a consistent testing protocol. To ensure consistent recording of shake data, we created a CGI program accessed via a WWW form (Figure 8), that collects the relevant data and generates a machine-readable QR code encoding it. That QR code is then displayed on a monitor along with a target pattern that allows determination of sharpness of the captured image. Note that the QR code deliberately does not contain any information that would identify the person using the camera.

The make and model of camera, shutter speed used, shutter mode (e.g., use of EFC), and many other parameters need not be encoded in the QR code because they can be read directly from the EXIF data in the captured image.

Combining the captured image with the motion trace acquired using ShAKY provides all the data needed for a variety of research directions involving characterizing shake, improving methods for minimizing the detrimental effects on captured image quality, etc.

Results

The primary result is that we have created an open-source tool and test procedure, which we hope many others will use for measuring camera motion. Additionally, several preliminary observations from measurements taken are noteworthy:

- Testing with various cameras quickly revealed that there is a wide variation in shake profile even in consecutive shots taken by the same person in the same way.
- Using a Canon 5D IV, it was expected that holding the camera away from one's body to use the rear LCD would yield far more shake than using the OVF with the camera pressed against the user's face. Actual measurements show shake roughly doubles using the LCD. Using the EVF vs. rear LCD on a Sony A7 had a similar effect.
- As expected, when using a Sony A7, enabling EFC very significantly reduced shake as compared to using fully-mechanical shuttering.
- As a result of both OIS and IBIS becoming common in digital cameras[14], CIPA established a standard for testing the effectiveness of image stabilization systems in cameras[2]. On page 37, that document explains that yaw and pitch of camera shake were measured for "many people" and the "characteristic frequency and amplitude were extracted and synthesized to generate the vibration waveforms" that are used to drive a vibration platform for testing cameras. The other four dimensions of camera movement, roll, X, Y, and Z, are described as "practically negligible" when "the shooting distance is about 20 times the focal length." Our test results are mostly consistent with CIPA's claim, although waveform properties are not very consistent and roll is commonly significant (in off-axis areas of the image).
- Surprisingly, a two-handed grip on the camera often produced comparable or more shake compared to a one-handed grip. There also were differences in which axis had the most motion.

Conclusion

This paper has presented the design of an inexpensive, open source, motion tracking device suitable for conducting experiments involving camera movement: ShAKY. Although we may continue to refine ShAKY's 3D-printed housing, electronics, internal and host software, and recommended test procedure, everything is freely available from links at `Aggregate.Org/DIT/ShAKY`.

ShAKY was built to enable various research projects involving characterization and handling of camera shake, both for us and for the community as a whole. A test protocol, aided by an interactive WWW form, which greatly simplifies making many measurements also is presented. In addition, some very preliminary, yet interesting, shake characterization results were summarized.

References

[1] The History of Steadicam https://tiffen.com/pages/history-of-steadicam (accessed September 23, 2019)

[2] Standard of Camera & Imaging Products Association, CIPA DC-X011-Translation-2014, Measurement and Description Method for Image Stabilization Performance of Digital Cameras (Optical System), http://www.cipa.jp/image-stabilization/documents_e/DC-X011-2014_E.pdf (2014)

[3] David Mas, Julian Espinosa, Ana B. Roig, Belen Ferrer, Jorge Perez, and Carlos Illueca, "Measurement of wide frequency range structural microvibrations with a pocket digital camera and sub-pixel techniques," Appl. Opt. 51, Issue 14, pp. 2664-2671 (2012) https://doi.org/10.1364/AO.51.002664

[4] Belen Ferrer, Julian Espinosa, Ana B. Roig, J. Perez, and D. Mas, "Vibration frequency measurement using a local multithreshold technique," Opt. Express 21, Issue 22, 26198-26208 (2013) https://doi.org/10.1364/OE.21.026198

[5] Xiujun Lei, Yi Jin, Jie Guo, and Changan Zhu, "Vibration extraction based on fast NCC algorithm and high-speed camera," Appl. Opt. 54, Issue 27, pp. 8198-8206 (2015) https://doi.org/10.1364/AO.54.008198

[6] Kazuki Nishi and Tsubasa Onda, "Evaluation system for camera shake and image stabilizers," 2010 IEEE International Conference on Multimedia and Expo, Suntec City, 2010, pp. 926-931. DOI: 10.1109/ICME.2010.5583093

[7] Kazuki Nishi and Yuichi Matsuda, "Camera vibration measurement using blinking light-emitting diode array," Opt. Express 25, Issue 2, pp. 1084-1105 (2017) https://doi.org/10.1364/OE.25.001084

[8] Reza Safaee-Rad and Milivoje Aleksic "Handshake characterization and image stabilization for cell-phone cameras", Proc. SPIE 7241, Color Imaging XIV: Displaying, Processing, Hardcopy, and Applications, 72410V (19 January 2009); https://doi.org/10.1117/12.806118

[9] Roarke Horstmeyer, "Camera motion tracking for deblurring and identification," Proc. of MIT Media Lab MAS, 863, pp. 1-7. 2010.

[10] S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," 2011 IEEE International Conference on Rehabilitation Robotics, Zurich, 2011, pp. 1-7. DOI: 10.1109/ICORR.2011.5975346

[11] MPU-9250 Product Specification Revision 1.1, http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf (accessed September 24, 2019)

[12] Paul Shubham and Ganguly Samhita, Using the MPU9250 to get Real-time Motion Data, https://www.hackster.io/30503/using-the-mpu9250-to-get-real-time-motion-data-08f011 (accessed September 25, 2019)

[13] Henry Dietz and Paul Eberhart, "Shuttering methods and the artifacts they produce," IS&T Electronic Imaging, Photography, Mobile, and Immersive Imaging 2019, pp. 590-1-590-7(7)

[14] Norihiro Aoki, Hiroya Kusaka, Hiroyuki Otsuka, "Measurement and description method for image stabilization performance of digital cameras," Proc. SPIE 8659, Sensors, Cameras, and Systems for Industrial and Scientific Applications XIV, 86590O (19 February 2013);