https://doi.org/10.2352/ISSN.2470-1173.2017.12.IQSP-247

Autofocus Analysis: Latency and Sharpness

Katrina Passarella (kpassarella@google.com)
Brett Frymire (bfrymire@google.com)
Ed Chang (changed@google.com)

Abstract

In this study a new camera testing method is introduced to determine and analyze the autofocus latency of cameras. This analysis allows for objective comparison and tuning of autofocus algorithms in order to deliver both sharp images and the optimal user experience with a camera. Given images taken in variable illuminance conditions with different methods of focus reset, along with high-speed recordings of the camera viewfinder throughout the reset and capture, machine vision is used to extract three different types of latencies:

- *The first latency is the autofocus time, which is measured from the end of the focus reset to full stability, as measured by slanted-edge sharpness in the camera viewfinder.*
- *The next latency is the user interface latency, which also comes from the viewfinder and is the time between the camera trigger and when the user interface of the camera indicates that a capture took place.*
- *The final latency is the captured image latency, which is taken from the captured image itself and is the time between the camera trigger and when the image is actually captured.*

In addition, we measure the sharpness of the final captured image in each test. Commercially available smartphone devices were tested using this method, showing significantly different results in both latency and sharpness measurements and uncovering trends in sharpness-latency trade-offs.

Introduction

In this study, we have explored a new camera evaluation method centered around the testing and analysis of autofocus algorithms. Exploring how autofocus algorithms on varied camera modules perform in comparison to each other and how they affect the optimal user experience with a smartphone camera can give insight to use in parameter tuning and benchmarking between cameras. This kind of analysis is best done in an automated and repeatable process, controlling as many test parameters as possible, in order to obtain the most objective and comparable results possible. Such a method, which has been developed and already used on commercially-available smartphones, is presented.

Hardware and Control Software

The first component to this fully automated autofocus testing system is customized testing hardware and corresponding control software. Our self-containing test station consists of a chart holder, chart magazine, adjustable light sources, LED tracer box, camera platform, autofocus reset devices, and a high-speed camera.

The chart holder has a magazine containing a chart used as the target in the image capture and also a chart with illuminance meters to measure light levels in the center and edges of the chart during test setup and calibration. The LED tracer box, overlaying a portion of the chart, has a grid of LED lights that sequentially light up when triggered at the start of the test. The tracer is included in the captured image to timestamp the capture and exposure time based on the positions of the lit LED dots. Further, the autofocus reset devices include an occluder and rotation stage. The occluder moves a near target into the test camera field of view, blocking the view of the far target. The rotation stage rotates the test camera to different angles so that it can point towards different targets located at various distances.

All of the hardware components are controlled by custom software, leading to a fully automated data collection system. This not only reduces the risk of human error during testing, as there are many components in play, but it also adds to the consistency and repeatability available in such a test.

Testing Parameters

There are certain parameters that we control that affect the latency and sharpness results. These include the camera-to-target distance, focus reset mode, light level, target type, reset delay, HDR+ mode, capture delay, and number of iterations. We have two general testing modes:

1. The first tests detection of objects and involves occluding the camera field of view with a near target consisting of a high-contrast image at 0.2 meters and keeping it there for a reset delay period of 3 seconds, then removing it to test the camera's ability to switch focus to a target of slanted edges placed 2 meters away.
2. Our second test looks at how the smartphone's gyro detection influences autofocus. The smartphone camera platform is rotated such that the camera has 3 seconds of reset delay to focus on a busy image 0.2 meters to the side, and then it is rotated back to be evaluated on its ability to refocus on a target of slanted edges placed 2 meters away.

On top of the reset modes, we test different capture delays (0 milliseconds through 1000 milliseconds), which is the time between when the focus reset finishes and the camera is triggered to take a picture. Each of these capture delays are repeated 10 times to ensure the gathering of statistically significant data sets. Further, we run each of the above tests with all of the capture delays and iterations under different testing parameters, such as with HDR on or off, in addition to low and high illuminance levels.

In addition, the chart with illuminance meters helps maintain and measure lighting across the tests, and the adjustable light sources throughout the isolated testing room allow for consistency between the near and far targets to eliminate an autoexposure (AE) latency variable. Further, the near and far targets have consistent color schemes such that auto white balance (AWB) variables do not add to the latency, either. This allows for a focus on the latency from the autofocus (AF) algorithm in the 3A (AE, AWB, AF) algorithm set.

Analysis Software

Following the capture of the images and video recordings of the smartphone's screen, the software uses machine vision algorithms to extract latency and sharpness measurements from the data set. These are plotted together to report the objective performance of the camera's autofocus algorithm. These measurements include:

- the autofocus time, which is seen as the time from the end of the focus reset to full stability;

- the user interface latency, which is the time between the camera trigger and when the user interface of the camera indicates that a capture took place;
- the captured image latency, which is the time between the camera trigger and when the image is actually captured;
- and the sharpness of the captured image.

Each of these measurements is examined in detail in the following sections.

Autofocus Time

The first measurement in the latency analysis is the autofocus (AF) time, which is the time between the focus reset and when the focus has stopped moving and thus stabilized. This focus time is directly caused by the speed of the autofocus algorithm to find the optimal focus position and move the lens to it, and in some cases it can be influenced to start or speed up by the capture trigger, hence the importance of varied capture delays in the testing parameters to look for this added factor.

This analysis method uses a high speed video recording of the smartphone's screen, computer vision, and a very thorough understanding of the control software's timing of the trigger to extract this AF time. Using computer vision, the positions of fiducials in the camera field of view can be searched for in the image and then used to find a slanted edge in the target off of which to measure the sharpness per frame in the video.

This is a complex task, as the varied spatial frequencies of the video camera and of the phone's screen cause heavy aliasing and flickering in the recording. For this reason, conventional sharpness measures (for instance, the Modular Transform Function (MTF)) cannot be used, as they would be unreliable and noisy with such aliasing present around and on the slanted edge. Instead, the sharpness calculation must be done more rudimentarily, creating a shifted edge profile for the edge after first oversampling the pixels in the box surrounding the edge as follows. When the edge is vertical, this involves taking the individual rows of the region of interest surrounding the slanted-edge and re-sampling them to increase the size of the row. Then, horizontally shift each row such that the crossover from the dark pixels to the light pixels occurs at the same index in each row array. When the edge is horizontal, this is done column-wise. Then, the number of indices over which this crossover exists is used as a measure of sharpness. As this number decreases, the sharpness of the edge increases, as this means the crossover width from one side of the edge to the other took less pixels, indicating a sharper edge. Finally, these values are normalized across the frames to maintain a relative change uninfluenced by the raw number found, as in this case we are not interested in the raw sharpness but rather when the sharpness converges. Once this number converges, the autofocus has stabilized and the autofocus algorithm appears to have finished. Using the timestamp from the video frame at which this occurs and our knowledge of the timing of the hardware control software's timing of the focus reset, we extract the autofocus time.

User Interface Latency

Next, we measure the latency that the user experiences with the camera user interface (UI) and viewfinder, which we will refer to as the UI latency. As an example, on many smartphones, image capture is indicated by the screen going dark. This is the latency that the user directly experiences, and the ideal situation involves the screen going dark immediately following the trigger to take a picture. Thus, the UI latency is measured from the moment the camera is triggered to take a picture to when the UI indicates capture, and the ideal latency is zero.

Similar to the autofocus time above, we use a high speed video recording of the smartphone's screen to extract this UI latency. Computer vision is used to find the location of the same slanted edge used for the autofocus latency in the middle of the target based on the relative locations of fiducials. A slanted edge is used because we know that roughly half of the pixels are dark whereas the remaining are brighter. Based on this assumption, the software can tell when the UI goes dark because the average pixel value will be noticeably different when this occurs than at any frame leading up to it. Further, the average pixel values per video frame are normalized to increase the consistency of finding this point no matter the contrast on the slanted edge, as, again, we are interested in measuring a change rather than a raw pixel value.

Also to be noted, this indication of the screen going dark is not affected by the heavy aliasing from the video capture of the smartphone's screen, as the amount of aliasing is consistent throughout the frames. The frequencies and positions of the mounted smartphone and video camera do not change throughout the capture sequence, allowing for a consistent period between the aliased dots, meaning that, for example, if they all shift one pixel the mean value is conveniently still the same in the boxed region of interest.

Captured Image Latency

Additionally, although it is not generally noticed, the user interface does not always indicate capture (for instance, with the screen going dark) at the actual capture time of the saved image, for a variety of algorithmic reasons. Thus, an additional latency of the actual captured image is measured. This capture time is essentially a timestamp in the output image that is recorded with the use of an LED light tracer with grid dots that light up sequentially throughout the capture sequence such that the positions of the lit LED lights is captured along with the image target.

We implement a computer vision algorithm to find fiducials in the captured image and then the relative position of the LED tracer box's grid. Once we identify the lit LED lights, their positions are used to extract the timestamp of the capture. This is not a simple task, as there are factors that can cause the grid of LED lights to be unclear to read. However, the algorithm reading needs to be robust enough to handle harsh testing conditions, as the autofocus algorithm must be tested in all types of conditions in order to get a true understanding of its performance. We explore two such complex situations:

First, testing in dark conditions can cause the images to be noisy or blow out the lit LED lights such that they bleed into other areas of the grid. A way of counteracting this involves normalizing and gray-scaling pixel values in the grid to locate the center of the dots in question. Normalization takes away the exposure factor, as we care that there is a difference between the pixel values of the lit lights and non-lit lights, not the exact pixel values.

In addition, when you're searching for a, say red, LED light, it seems more intuitive to only check the red color channel for when it peaks. However, gray-scaling the image by weighing all three of the RGB color channels can help find the center of the dot. The surrounding grid is black, so when the lights are not blown out (in "good" lighting conditions), the center of the red dots still have the largest pixel values when gray-scaled, as they are full red while the blue and green values are at a minimum, while everywhere else all color channels are at a minimum. Additionally, when the lighting conditions are not as ideal and the LED lights bleed outside their respective grid locations, both the dot centers and bleeding area have full red pixel values, but the centers are stronger and actually have significant blue and green values, allowing for a distinction between the centers and edges of the dots that can only be seen when looking at all three color channels instead of just the red one. In this case, the gray-scaling helps because it weights in all three of the color channels, as opposed to only checking the red channel.

Second, when the image is out-of-focus (which occurs frequently when testing autofocus latencies), the lit LED lights are still noticeable, but the artifacts and fiducials used to locate them in the image may be out-of-focus enough to make their search unreliable. These types of conditions need to be understood and flagged with low confidences, as they could lead to looking in incorrect locations for the grid and causing inaccurate results.

Based on knowledge of the timing of the hardware control script that triggered the image capture and reaction times of the hardware, we calculate the latency between the trigger and the actual image capture.

Sharpness

In addition to measuring the latencies involved in the autofocus algorithm and image capture, the sharpness of the captured JPEG image is measured and recorded. This allows for an analysis of the trade-off between taking a quick but potentially out-of-focus image and waiting to ensure a fully focused image before capturing, as solely measuring image capture latency tells only half the story.

This is achieved by using computer vision to locate fiducials and then the corresponding location of a slanted edge in the middle of the captured image and using the Modular Transform Function (MTF) to get 50% of the peak value, a measurement known as the MTF50 measure. Further, to take out the factor of varied post-processing sharpening done by different camera modules, this pixels/cycle value is converted into a maximum sharpness percentage, normalizing the value to the maximum MTF50 value observed with the camera. This normalization moves the analysis focus from raw output sharpness to whether or not the autofocus algorithm had time to finish before capture or not, which is what this testing system is intended to analyze.

Findings

This testing system has debuted on various smartphones cameras, and a few findings have come about. It has shed light on strengths and weaknesses in the autofocus modules of different cameras, highlighting algorithmic decisions on trade-offs between quick but blurry performance and slow but sharp images. Worse autofocus algorithms had fairly binary performance, either waiting to capture a fully focused image or quickly capturing a completely blurry image, not making the user wait. Of course, the optimal camera experience results in a fast, sharp image, but next-best options uncover a subjective choice between immediate shutter time, slow but fully sharp images, and something in between. A testing system as described here can help with tuning the camera while maintaining awareness of this trade-off, converging towards an optimized solution that balances the two options.

A sample results table is given here:

Measure-ment	**Light level**	**Focus Reset**	**Camera A**	**Camera B**	**Improvement**
Autofocus Latency	300 lux	rotation	146 ms	62 ms	2.4x
		occluder	979	504	1.9x
	5	rotation	460	437	1.1x
		occluder	1094	849	1.3x
User Interface Latency	300	rotation	605	255	3.0x
		occluder	585	257	2.3x
	5	rotation	1204	270	4.5x
		occluder	1107	453	2.4x
Captured Image Latency	300	rotation	590	195	3.0x
		occluder	548	191	2.9x
	5	rotation	1113	215	5.2x
		occluder	1004	380	2.6x
Sharpness	300	rotation	83	95	1.1x
		occluder	67	94	1.4x
	5	rotation	55	70	1.3x
		occluder	35	65	1.9x

In addition, the development of this testing system has brought findings aside from the actual latency and sharpness measurements. Most straightforward, the number of controlled parameters goes to show how many factors can influence the performance and measurement of an autofocus algorithm or any algorithm in a camera module. These types of factors must always be watched in an objective measurement system, and are more easily controlled when the system is fully automated as described here, as it's simple to add in additional parameters when needed if the system is built well. Further, the complex solutions in the analysis software to counteract the noise from poor testing conditions and the use of a video recording of a phone screen can be used as examples for related analysis systems and extended for further use. These situations are not unique to just this system, and could appear in any testing automation and analysis.

Use and Impact

The purpose of this testing system is to be able to objectively measure the latencies and sharpness of a camera for algorithmic tuning and also for the sake of comparisons with other cameras. The custom-built hardware components and corresponding control script allow for consistency in measurements, as confirmed through repetitive testing runs. This is an data consistency that could not be achieved with conventional subjective hand-held autofocus testing. Further, the fully-automated objective and normalized software analysis of the raw test data allow for consistent measurements to tune against and benchmark camera phones.

Commercially available smartphones have already been tested, compared, and tuned using this testing system, resulting in consistent results per phone, varying results across smartphones and benchmarks, and noticeable improvements or worsening following algorithm tweaks during tuning cycles. A testing system as described here has been able to help with tuning a camera while maintaining awareness of the sharpness-speed trade-off, working towards a convergence to an optimized solution that balances the two options.

Conclusion

This automated autofocus testing system presents a method for gathering and analyzing the latencies related to autofocus along with corresponding sharpness scores. Using specialized hardware and keeping sensitive parameters in control, control software gathers data sets. Then, analysis software extracts three latencies in particular, namely the time from the end of the focus reset to focus stabilization (autofocus time), the time from the camera trigger to the capture indication in the camera viewfinder (user interface latency), and the time from the camera trigger to the time stamped with the actual saved image (captured image latency). Further, the sharpness of each saved image is measured and correlated to the latency measurements for a comprehensive report on autofocus performance.

This objective autofocus analysis allows for direct comparisons between camera modules and autofocus algorithms, along with assistance and feedback in tuning to create the optimal user experience with a smartphone camera while exploring the trade-off between autofocus speed and sharpness of the captured image.