

# Revisiting Autofocus for Smartphone Cameras

Abdullah Abuolaim, Abhijith Punnappurath, and Michael S. Brown

Department of Electrical Engineering and Computer Science  
Lassonde School of Engineering, York University, Canada  
{abuolaim, pabhijith, mbrown}@eecs.yorku.ca

**Abstract.** Autofocus (AF) on smartphones is the process of determining how to move a camera’s lens such that certain scene content is in focus. The underlying algorithms used by AF systems, such as contrast detection and phase differencing, are well established. However, determining a high-level objective regarding how to best focus a particular scene is less clear. This is evident in part by the fact that different smartphone cameras employ different AF criteria; for example, some attempt to keep items in the center in focus, others give priority to faces while others maximize the sharpness of the entire scene. The fact that different objectives exist raises the research question of whether there is a preferred objective. This becomes more interesting when AF is applied to videos of dynamic scenes. The work in this paper aims to revisit AF for smartphones within the context of temporal image data. As part of this effort, we describe the capture of a new 4D dataset that provides access to a full focal stack at each time point in a temporal sequence. Based on this dataset, we have developed a platform and associated application programming interface (API) that mimic real AF systems, restricting lens motion within the constraints of a dynamic environment and frame capture. Using our platform we evaluated several high-level focusing objectives and found interesting insight into what users prefer. We believe our new temporal focal stack dataset, AF platform, and initial user-study findings will be useful in advancing AF research.

**Keywords:** autofocus, focal stack, AF platform, low-level computer vision

## 1 Introduction

One of the crucial steps in image capture is determining what part of the scene to focus on. In this paper, we examine this problem for smartphone cameras because smartphones now represent the dominant modality of video and image capture performed by consumers. While manual focus is possible on smartphones—either through direct manipulation of the lens position or by clicking on regions of interest in the scene—most users rely on the camera’s autofocus (AF) mechanism.

The goal of AF is straightforward. Given some high-level objective of what scene content or image region is desired to be in focus, AF systems attempt to

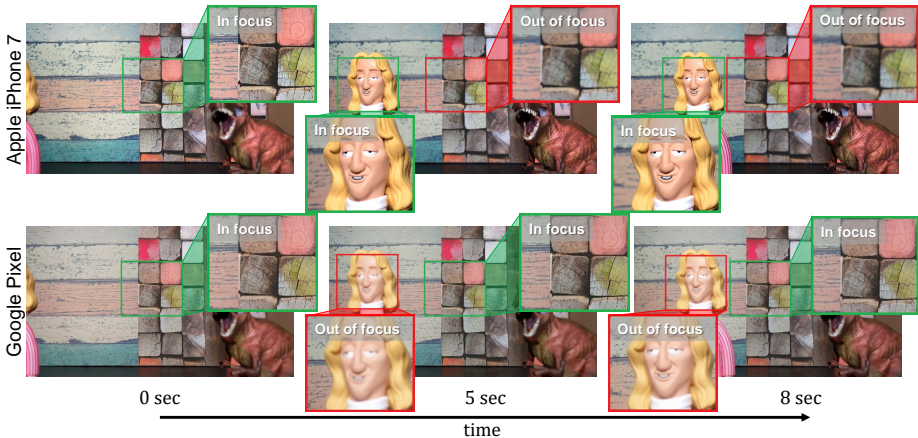


Fig. 1: An Apple iPhone 7 and Google Pixel are used to capture the same dynamic scene controlled via translating stages. At different time slots in the captured video, denoted as 0 sec, 5 sec, 8 sec, it is clear that each phone is using a different AF objective. It is unclear which is the preferred AF objective. This is a challenging question to answer as it is very difficult to access a full (and repeatable) solution space for a given scene.

move the lens such that these regions appear sharpest. From an optical point of view, the sharpness correlates to the desired image region lying within the lens’s depth of field. Smartphone cameras, as opposed to digital single-lens reflex (DSLR) and point-and-shoot cameras, are unique in this regard, since they have fixed apertures and depth of field is therefore restricted to lens position only.

The low-level algorithms used to determine image sharpness—for example, contrast detection and phase differencing—are well established. What is more challenging is using these low-level algorithms to realize high-level AF objectives for dynamic scene content in a temporal image sequence (i.e., video). This is evident from the variety of different AF criteria used by different smartphone cameras. Figure 1 shows an illustrative example. In this example, an Apple iPhone 7 and a Google Pixel have captured a scene with objects that move on a translating stage. The translating stage and controlled environment allow each camera to image the same dynamic scene content. We can see that each camera is focusing on different image regions at the same time slots in the video.

This begs the question of which of these two approaches is preferred by a user. From a research point of view, one of the major challenges when developing AF algorithms is the inability to examine the full solution space since only a fixed focal position can be captured at each time instance. While it is possible to capture a full focal stack for a static scene, it is currently not possible for a temporal image sequence in a dynamic environment. Moreover, there are additional constraints in an AF system beyond determining the right focal position given a full focal stack. For example, the lens cannot be instantaneously moved to

the correct focal position; it can only advance either forward or backward within some fixed amount of time, and within this time quantum the scene content may change and the current video frame may advance. This lack of access to (1) temporal focal stack data and (2) an AF platform that holistically incorporates lens motion, scene dynamics, and frame advancement is the impetus for our work.

**Contribution** The contribution of this work is a software platform for AF research and an associated 4D temporal focal stack dataset. Our AF platform allows the design, testing, and comparison of AF algorithms in a reproducible manner. Our focal stack dataset is composed of 33,000 full-frame images consisting of 10 temporal image sequences, each containing 50–90 full focal stacks. Our software platform provides an AF application programming interface (API) that mimics the real-time constraints, including lens motion timing with respect to scene motion and frame advancement. Additionally, we have performed analysis on several smartphone AF algorithms to come up with a set of representative high-level AF objectives. Using our platform and data we have implemented these algorithms to produce similar outputs found on real phones and used the results to perform a user study to see if there are any preferences. Our user study reveals that overall lens motion, and not necessarily the actual scene content in focus, is the predominant factor dictating preference. We believe our dataset and software platform will provide further opportunities for revisiting AF research.

## 2 Related work

Work relating to autofocus and focal stack datasets is discussed in this section.

**AF for cameras** AF technologies have been around for several decades and a full discussion regarding existing AF methods is outside the scope of this paper. Here, we provide background to methods used in smartphone devices and that are related to our platform. The vast majority of smartphone cameras have simple optical systems with a fixed aperture that limits focus to lens motion (and not aperture adjustments). There are active AF methods that use auxiliary hardware, such as laser depth sensors; however, this paper focuses only on passive AF methods that rely on data captured from the image sensor.

There are two predominant types of passive AF: phase difference autofocus (PDAF) and contrast detection autofocus (CDAF). PDAF operates at a hardware/optics level and aims to adjust the lens position such that the phase between two light rays coming from a scene point is matched. The PDAF hardware module can be designed in two ways: (1) half sub-mirror with line sensor as used in older DSLR cameras [1,2] and (2) on-sensor dual-pixel layouts used in modern DSLR and smartphone cameras [3,4]. Compared with CDAF, PDAF methods are able to approximate the optimal lens position in a single processing step; however, PDAF alone is generally not sufficient to give an accurate focusing lens position.

CDAF is the most common approach used in DSLR and smartphone cameras. CDAF operates by applying low-level image processing algorithms (i.e., gradient magnitude analysis) to determine the sharpness of a *single* image or re-










Scene	1	2	3	4	5	6	7	8	9	10
Example image										
Category	NF		FF		NF	FF	FB		NF	FF
Camera	stationary				moving	stationary		moving	stationary	
Textured background	✓	✗	✓							
Face	✗		✓		✗	✓		✗	✓	
Motion switches	1				3	0	2		2	
Video length	21.6 sec				27.5 sec	29 sec	30.8 sec		39.1 sec	
Discrete time points	51				61	71		91		

Table 1: The 10 scenes/image sequences in our AF dataset. See Sec. 3.3 for detail of the table and video/image sequence description. The final table row, discrete time points, denotes the number of full focal stacks per captured temporal image sequence.

gion of interest (ROI) in an image [5]. Because CDAF works on a single image, the camera lens needs to be moved back and forth until the image sharpness measure is maximized [6]. Many different sharpness measures have been proposed and several surveys exist that examine their performance under various conditions [7,8].

Most of the recent smartphone cameras use so-called *hybrid AF* that utilizes both PDAF and CDAF. In particular, the hybrid AF performs PDAF first to move the lens to a position close to the optimal focusing position and then performs CDAF to accurately fine-tune the lens position to reach the optimal focusing position [9].

**Focal stack datasets** Beyond various ad hoc focal stack data available online from class projects and photography enthusiasts, there are very few formal focal stack datasets available for academic research. Two notable datasets are by Mousnier et al. [10] and Li et al. [11]. The dataset in [10] provides 30 focal stacks of static scenes of images of size  $1088 \times 1088$  pixels. The dataset in [11] captured 100 focal stacks of image size  $1080 \times 1080$  pixels, again of static scenes. The number of images per focal stack ranges from 5 to 12. These datasets are not intended for the purpose of AF research, but instead target tangentially related topics, such as digital refocusing [12,13,14], depth from defocus [15,16], and depth from focal stacks [17].

In addition, the focal stacks in these datasets are synthetically generated based on the Lytro light field camera [18,19]. Unfortunately, the consumer-level Lytro devices do not support video capture. The new Lytro Cinema does offer video light field capture, but the cost of renting this device is prohibitively high (in the hundreds of thousands of dollars). Moreover, the Lytro Cinema is not representative of smartphones. Unlike the datasets in [10,11], our dataset provides

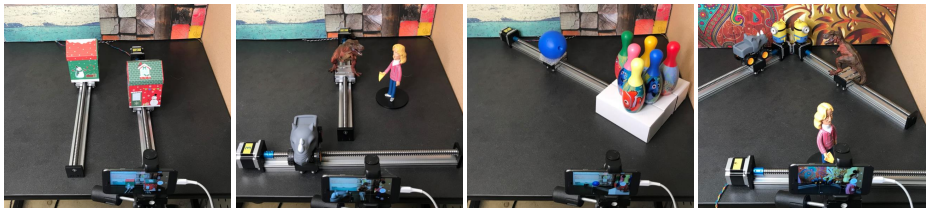


Fig. 2: A top view of our capture environment. Each shoot contains the scene components: linear stage actuators, smartphone camera, tripod, objects, and scene background.

a much larger focal stack of 50 images of size  $3264 \times 1836$  pixels, and consists of 10 temporal image sequences with up to 90 full focal stacks per sequence.

### 3 AF analysis and dataset capture

#### 3.1 Capture environment

To begin our effort, we constructed an environment that allowed scenes with different content and moving objects to be imaged in a repeatable manner. All videos and images were captured indoors using a direct current (DC) light source to avoid the flickering effect of alternating current lights [20]. To control scene motion, we used three DIY-CNC linear stage actuators that were controlled by a ST-4045-A1 motor driver and Arduino/Genuino Uno microcontroller. Each linear stage has a travel length of 410mm and uses a stepper motor of Nema 23 24V 3A 1N.M. The three linear stage actuators can be combined together to give more degrees of freedom. We calibrated our motors to allow 106 equal steps of 3.87mm each with a motion speed of 9.35mm/s.

#### 3.2 Analysis of smartphones AF

Within this environment, we analyzed the performance of three representative consumer smartphones (Apple iPhone 7, Google Pixel, Samsung Galaxy S6) to observe their behaviour under different scenarios. The cameras are positioned such that their fields of view are as similar as possible. The frame rate for video capture is fixed at 30 frames/sec. Given the different optical systems and image formats among the cameras, there are slight differences in the field of view, but these differences are negligible in terms of their effect on the AF outcomes.

We experimented with a wide range of scene configurations, such as an object with a figurine with a human face, textured backgrounds, and various moving objects. As previously illustrated in Figure 1, we observed that the AF behaviors differ between phones. For example, in one experiment we set up a textured

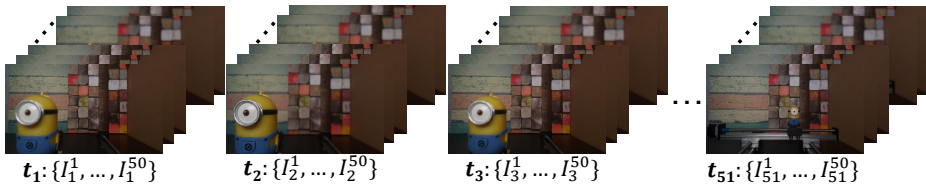


Fig. 3: Example of the temporal image sequence for scene 3. Focal stacks consist of  $I_i^1, \dots, I_i^{50}$  images for each time point  $t_i$ .

background and a textured object to move horizontally from left to right with respect to the camera. We observed that for the Google Pixel and Samsung Galaxy S6 Edge, the foreground object becomes in focus only when it is inside the center of the image; otherwise it is out of focus. For the same setup captured by an Apple iPhone 7, however, the foreground object is in focus most of the time regardless of its position from the center. In another experiment with a figurine with a human face, we observed that the three smartphones detected the face in a video, but only Apple iPhone 7 focused on the face region.

### 3.3 Scene and image sequence capture

Based on our observations, we settled on 10 representative scenes that are categorized into three types: (1) scenes containing no face (NF), (2) scenes with a face in the foreground (FF), and (3) scenes with faces in the background (FB). For each of these scenes, we allowed different arrangements in terms of textured backgrounds, whether the camera moves, and how many types of objects in the scene change their directions (referred to as *motion switches*). Table 1 summarizes this information. Figure 2 shows the physical setup of several of the scenes.

For each of these 10 scenes, we captured the following data. First, each scene was imaged with the three smartphone cameras. This video capture helps to establish high-level AF objectives used on phones and determines the approximate video length needed to capture the overall scene dynamics. The duration of these videos is provided in Table 1. Due to limits on the supplemental materials, representative down-sampled versions of the videos are provided.

Next, we captured temporal focal stacks for each of these scenes. We refer to these as *image sequences* to distinguish them from the actual videos. To capture each image sequence, we replicated the video capture in a stop-motion manner. Specifically, the objects in the scene are moved in motion increments of 3.87mm between consecutive *time points*. We used the Samsung Galaxy S6 Edge to perform the image capture using a custom Android app that fixed all camera settings (e.g., ISO, white balance, shutter speed). Our app also controlled the lens position, such that for each *time point*  $t_i$ , we captured a focal stack of 50 images where the camera lens is moved in linear steps from its minimum to maximum position. The last row in Table 1 shows also the number of *time points* for each captured temporal image sequence. In this paper we use the term *time*

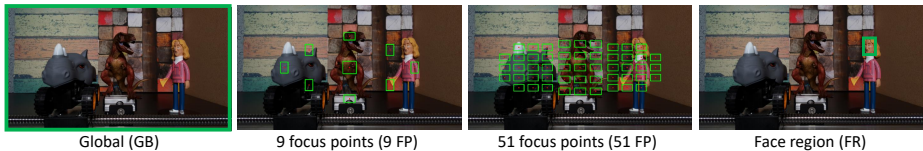


Fig. 4: Our four AF objectives. The region bounded in a green box is a candidate for ROI(s).

*point* to denote time slot in our stop-motion data. We also use the term *frame* to denote real-time video frame, either from a real video or an output produced by our AF platform.

Figure 3 shows an example of scene 2 with 50 *time points*. Each *time point*  $t_i$  in Figure 3 has a focal stack of 50 images that are denoted as  $I_i^j$ ,  $j = 1, \dots, 50$ , where  $i$  denotes time point and  $j$  indexes the focal stack image associated to a specific lens position.

## 4 AF platform and API

We begin with a short discussion on how our platform emulates PDAF and CDAF as these are the low-level algorithms of any AF system. This is followed by a discussion on the overall platform and associated API.

### 4.1 PDAF/CDAF emulation

The CDAF and PDAF process can be divided into three main steps: first, determine a desired region of interest (ROI) based on the high-level AF objective; second, measure the sharpness or phase of the ROI selected; third, adjust the lens position to maximize the focus.

Based on the observed behaviour of the captured video from our three smartphone cameras on the 10 scenes, we determine four high-level AF objectives in terms of ROI as follows: (1) global ROI targeting the whole image; (2) a layout of 9 focus points with 9 ROIs; (3) a layout of 51 focus points with 51 ROIs (similar to the global ROI); (4) and a *face region* ROI where the largest region of detected faces is set as the ROI. Figure 4 shows the ROI(s) for each objective bounded in a green box.

Our AF platform provides the flexibility to manually specify the ROI; however, based on the above four objectives, we provide these as presets that the user can select. To facilitate the face region objective for our dataset, we manually labeled the face regions to avoid any face detection algorithm mistakes. Our platform allows retrieval of the labeled face region via an API call; however, when the pre-defined face region is selected, this call is automatically performed and the ROI set to the face region. Regarding the sharpness measure for the CDAF, we provide two gradient based filters—namely, Sobel and Prewitt operators. Based on Loren’s findings in [7], the Sobel and Prewitt filters are the most

API call	Description	Return values	Clock cycles
<code>setScene(int sc)</code>	Select one of the 10 scenes, <code>sc=0, ..., 9</code>	<code>null</code>	0
<code>setRegion(int [] reg)</code>	Set the region either by selecting one of the predefined regions: Global ( <code>reg=0</code> ), 9 Focus Points ( <code>reg=1</code> ), 51 Focus Points ( <code>reg=2</code> ) or Face Region ( <code>reg=3</code> ), or by passing an array of size $r \times 4$ where $r$ is the number of regions. Each region has offset (x,y), width, and height.	<code>null</code>	0
<code>setSharpMeasure(int sh)</code>	Select one of the two predefined sharpness measures: Sobel ( <code>sh=0</code> ) or Prewitt ( <code>sh=1</code> ).	<code>null</code>	0
<code>setKernelSize(int ker)</code>	Select one of the three predefined kernel sizes: 3 ( <code>ker=0</code> ), 5 ( <code>ker=1</code> ) or 7 ( <code>ker=2</code> ).	<code>null</code>	0
<code>recordScript()</code>	Start recording the subsequent API calls in a script.	<code>null</code>	0
<code>endScript()</code>	Stop recording the subsequent API calls in a script.	<code>null</code>	0
<code>callPD(int <math>\rho</math>)</code>	Compute phase difference and return approximate optimal lens position $p \pm \rho$ .	$[C_{loc}, C_{glob}, I_{C_{glob}}^j, j, p]$	1
<code>callCD(function fun)</code>	Allow the user to pass custom contrast detection AF implementation as a function. Default Sobel/Prewitt with kernel size as set by user. <code>fun</code> is a function written in Python format.	$[C_{loc}, C_{glob}, I_{C_{glob}}^j, j, score]$	1 (if default) or defined by user
<code>moveLensForward()</code>	Move the lens a step forward.	$[C_{loc}, C_{glob}, I_{C_{glob}}^j, j]$	1
<code>moveLensBackward()</code>	Move the lens a step backward.	$[C_{loc}, C_{glob}, I_{C_{glob}}^j, j]$	1
<code>noOp()</code>	No operation. No lens movements. Used to increment $C_{loc}$ in order to move in global time $C_{glob}$ .	$[C_{loc}, C_{glob}, I_{C_{glob}}^j]$	1
<code>getFaceRegion()</code>	Detect face(s) and return face region(s) <code>int face[]</code> if exists. <code>face[]</code> is an array of size $m \times 4$ where $m$ is the number of face regions. Each face region has offset (x,y), width, and height.	$[C_{loc}, C_{glob}, I_{C_{glob}}^j, \text{face}[]]$	0

Table 2: API calls with their parameters and return values. Each API call incurs a cost related to the number of internal clock cycles.  $C_{loc}$  current clock cycle,  $C_{glob}$  current time point,  $I_{C_{glob}}^j$  current image at current  $C_{glob}$  and current lens position  $j$ ,  $p$  optimal lens position, and  $score$  is the score of gradient energy (default or defined by user). See supplemental materials for more API details.

accurate among other sharpness measure methods. The size of these filters can also be controlled.

## 4.2 AF platform and API calls

Our AF API is designed to emulate AF in smartphones. The platform and API impose constraints on lens motion timing with respect to scene motion and video frame rate. As such, our API and platform have a *local* and *global* virtual clock. The local clock, denoted as  $C_{loc}$ , emulates the real-time internal clock on the smartphone, whereas the global clock,  $C_{glob}$ , emulates the real-world timing (scene dynamics).

**Platform timing** Since the Samsung Galaxy S6 was used to capture our dataset, we measured its performance to establish the mapping between the local and global clocks. Specifically, we measured how long it took the camera to respond to a scene change at a different focal positioning by sweeping the lens to this position while capturing video. To do this, we set up two objects: a textured flat background and textured flat foreground; both are parallel to the camera plane at different depth layers (one close and one far). The background object appears at the beginning of video capturing and is in focus; then, after a short delay we immediately display the foreground object closer to the camera, which



causes the AF system to move the focus from background to foreground. Later, we decompose the captured video into frames and count how many frames it required to move from background to foreground. For the exact same scene scenario, we collected a full focal stack (50 images), previously discussed. To obtain how many steps the lens moved, we use the focal stack to compute at which lens positions the background and foreground objects are in focus.

Once we obtain the number of lens steps and number of frames required, we can compute from lens step to frame unit (33.33 msec). Therefore, we estimated the Samsung Galaxy S6 Edge requires 42 msec to move the lens one step (including image capturing and AF processing).

The time required for the translating stage motor to move one step (3.87mm) is 414 msec. Recall that a single translating stage motor step in real time is equivalent to a discrete time point in our stop-motion setup. Therefore, the number of steps  $s$  allowed for the lens to move in one time point is equal to  $414/42 \approx 9.86$  steps. Based on this approximate calculation, we fix  $s$  to 10 steps and we relate  $s$  to the local clock  $C_{loc}$  (one lens movement costs one clock cycle). Accordingly, the corresponding global clock  $C_{glob}$  increments every 10 clock cycles. Thus our relationship is: 10  $C_{loc}$  advances  $C_{glob}$  by 1.

**API** Our API is based on Python and provides 12 primitive calls as described in Table 2. See supplemental materials for more details.

The `recordScript()` and `endScript()` API calls are used to save the API calls and load them later for user algorithm playback purposes. These calls are also useful for capturing metadata about the performance of the algorithm—for example, lens position, API call made at each clock cycle, and ROI selected.

Our `callPD(int  $\rho$ )` API call is used to emulate the PDAF available on most high-end smartphone cameras. The *real* PDAF routine on a camera is able to find the approximate lens position for a desired ROI close to the optimal focal frame in the focal stack within a single processing pass of the low-level raw image. On real cameras, the PDAF result is obtained at a hardware level based on a proprietary layout of dual-pixel diodes placed on the sensor. We were not able to access this data and provided it as part of our focal stack dataset. As a result, we instead emulate the result of the phase difference by running CDAF targeted to the specified ROI on the whole focal stack at the current time-point  $t_i$  defined by the global clock  $C_{glob}$ . As mentioned previously, real camera PDAF is performed first to move the lens closer to the optimal focusing position; afterwards CDAF is typically performed to refine the lens position. To mimic this near optimality, we apply an inaccuracy tolerance  $\rho$  on the optimal focusing position obtained. This inaccuracy tolerance allows the estimated lens position to lie randomly around the optimal by  $\pm[0, \rho]$  and is a parameter that can be passed to the API.

### 4.3 Example implementation

Alg. 1 provides simple pseudo-code based on our API to demonstrate how an AF algorithm based on the *global objective* for Scene 4 can be implemented. Real Python examples and script recording and video outputs are provided in

---

**Algorithm 1** Example of a Global ROI Objective using Optimal PDAF
 

---

```

1: Start API
2: setScene(Scene4)
3: setRegion(Global)
4: recordScript()           //Create a script and start recording API calls
5: while not end of time points do
6:   if time point  $t_i$  incremented then
7:      $C_{loc}, C_{glob}, I_i^j, j, p \leftarrow \text{callPD}(0)$ 
8:   else if optimal lens position  $p >$  current lens position  $j$  then
9:      $C_{loc}, C_{glob}, I_i^j, j \leftarrow \text{moveLensForward}()$ 
10:  else if optimal lens position  $p <$  current lens position  $j$  then
11:     $C_{loc}, C_{glob}, I_i^j, j \leftarrow \text{moveLensBackward}()$ 
12:  else if optimal lens position  $p ==$  current lens position  $j$  then
13:     $C_{loc}, C_{glob}, I_i^j \leftarrow \text{noOp}()$ 
14:  end if
15:   $Video \leftarrow I_i^j$            //Write the acquired image into a video
16: end while
17: endScript()           //Close and summarize the script(e.g., # of lens movements)

```

---

the supplemental materials. In this simple example, we set the  $\rho$  to zero, which results in `callPD()` calls returning the optimal lens position.

Based on our implementation in Alg. 1, the time point  $t_i$  will be incremented by API every 10 clock cycles (as discussed before in Section 4.2). At each clock cycle API returns an image, which means we will get 10 images at each  $t_i$ . The total number of images returned by the API for a specific scene thus is equal to  $10 \times n$  where  $n$  is the scene size in *time points*. To generate an output video for a scene, we write each image at each clock cycle out to a video object. Running Alg. 1 will return metadata about the performance of the *global objective* for Scene 4. In Figure 5 we show the lens position over local time (clock cycles) for the *global objective* (GB) in the dark blue solid line. From Figure 5 we can analyze the lens movements over time, where the GB has fewer lens movements and less oscillation. Figure 5 also shows the lens position over time for other objectives for Scene 4.

## 5 User study on AF preference

We conducted a user study to determine if there was any particular preference for the different AF methods. As shown in Figure 5, the AF platform gave us the opportunity to track exact lens movement for each method. Lens motion was treated as a potential factor.

**Preparation** For this study we defined scene number, objective, and lens motion as our independent variables; the user preference is our dependent variable. We adopted a force-choice paired comparison approach that requires each participant in the study to choose a preferred video from a pair of videos. Both videos in a given pair are of the same scene but have different AF objectives. We used all

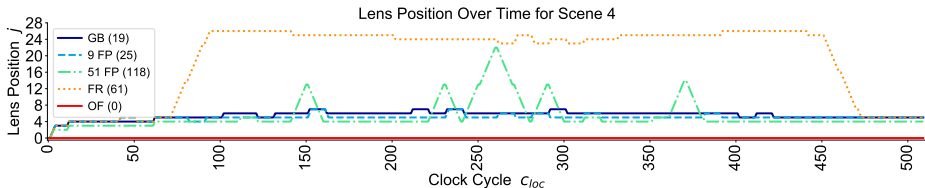


Fig. 5: This figure shows the lens position for each clock cycle for Scene 4 for each objective test. Total number of lens movements is shown in parentheses. An out-of-focus objective (OF) is included that does not move the lens over the entire sequence. For Scene 4, the 51 focus points (51 FP) objective oscillates the most. For the face region (FR) objective, the face did not enter the scene until clock cycle 70—the 9 focus points (9 FP) are applied by default when the face is not present. Global (GB) and 9 FP objectives tend to oscillate less than others with fewer lens movements.

10 scenes from our dataset for the study. There are six scenes with faces and four without. For the scenes with faces, there are four AF objectives—namely, global, 9 focus point, 51 focus point, and face region. The scenes without faces have only the first three AF objectives.

We generated output videos through our API and using our dataset and modifications of Alg. 1 for each AF objective on all scenes (example video frames from Scene 1 are shown in Figure 6). Due to limits on the supplemental materials, representative down-sampled versions of the user study videos are provided. Additionally, for each scene, we have generated an out-of-focus video, where all scene elements are out of focus. Those out-of-focus videos are generated through our API by fixing the lens to the maximum position and calling `noOp()` till the end-of-scene time points. However, for Scene 6, we omitted this objective because there is no lens position that makes all scene elements out-of-focus. Therefore, there are five scenes in total with five AF objectives (with the out-of-focus objective added), and another five scenes with only four AF objectives. The total number of paired comparisons is  $5 \times \binom{5}{2} + 5 \times \binom{4}{2} = 80$ .

**Procedure** We collected 10 opinions for each video pair from 80 participants (34 females and 46 males) ranging in age from 18 to 50. Each subject was shown 10 video pairs selected in random order. We designed a simple graphical user interface that allows the user to view video pairs, one pair after the other, and easily examine the difference in AF behavior. The interface allows the participants to watch the two videos in the current pair any number of times before they make a selection and proceed to the next pair. A snapshot of our interface is provided in the supplementary material. The survey takes on average three to five minutes to complete. The experiments were carried out indoors with calibrated monitors and controlled lighting.

**Outcomes** Recall our scenes are categorized as Cat. 1: scenes with no face (NF), Cat. 2: scenes with a prominent face in the foreground (FF), and Cat. 3: scenes in which the face is in the background (FB). For each category, we aggregated

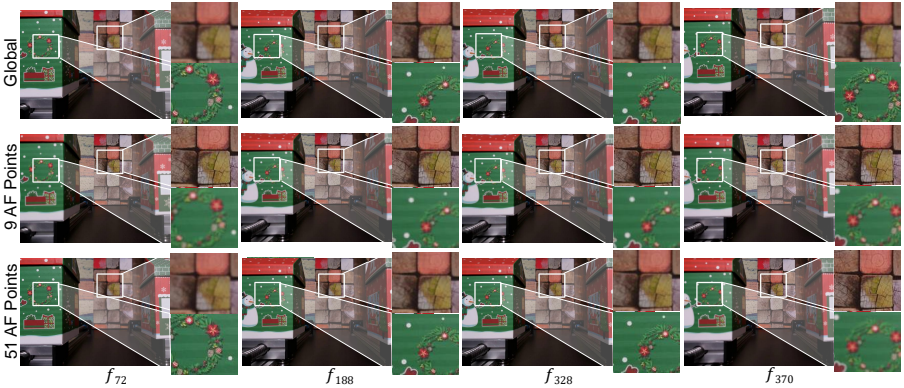


Fig. 6: Example output video frames generated by our AF platform using different objectives applied on Scene 1 over time. See supplemental materials for additional results for other scenes.

user votes into an overall score that represents user preference by counting the number of times each AF objective is preferred over any other objective. These results are presented in Figures 7 and 8. In Figure 7, in the first column, we show average user preference per AF objective for each category (i.e., aggregated over scenes). We can see that for NF videos, the global (GB) AF objective is the most preferred. For the FF videos, the face region (FR) AF objective is the most preferred. For the FB videos, there is no strong preference among the three objectives GB, 51 focus points (51 FP), and FR, but the most preferred is GB followed by FR. Additionally, we calculated the 95% confidence intervals for these results as represented by the error bars, which indicate the statistical significance of the results. Furthermore, the plots on the right of Figure 7 represent the user preference per objective for individual scenes (lower plots) with a corresponding number of lens movements (upper plots with grey bars) for each of the three categories. The individual scene plots also confirm the observations from the aggregate plots for all cases except Scene 9.

To examine the correlation between user preference and the number of lens movements for each category, we plotted the user preference vs. lens movements for each category, as shown in Figure 8. We see that there is a clear correlation between user preference and lens movements, suggesting that users tend to prefer the objectives with fewer lens movements. This is indicated by the negative correlation coefficients shown on the plots.

For the second category that contains a prominent face in the foreground, the results suggest that users prefer the face AF that locks onto the face even if more motion of the lens is required to achieve this objective. This voting pattern can be seen in the second row in Figure 7, where the FR AF objective receives a higher percentage of votes than the GB AF, which has the least amount of lens motion. Also note that the 51 focus points (51 FP) objective has the highest amount

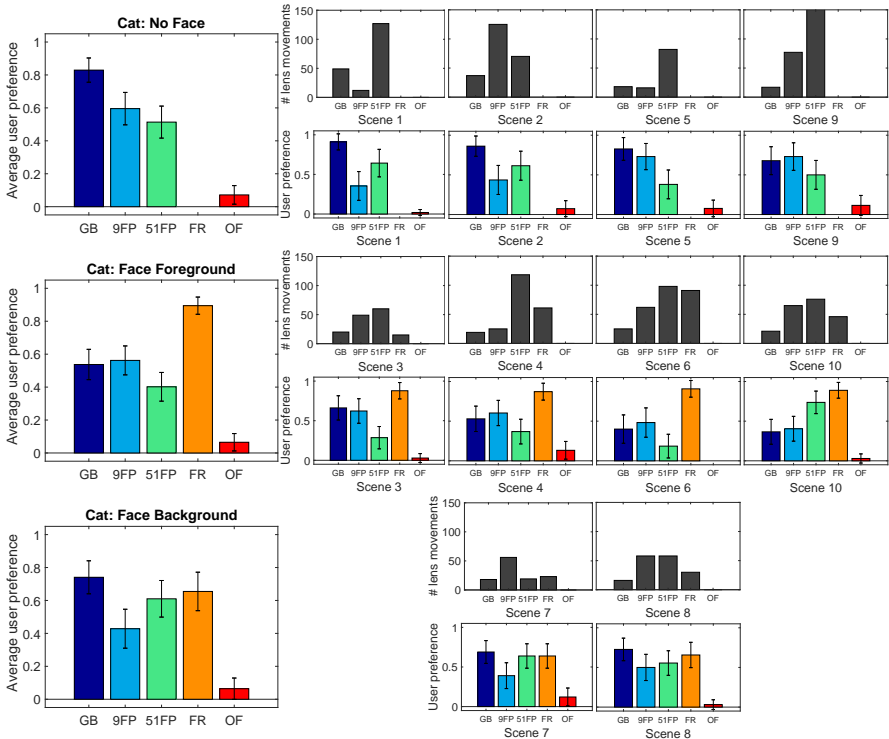


Fig. 7: User preference of AF objectives for three scene meta-categories: no face (NF), face in foreground (FF), and face in background (FB) for AF objectives: global (GB), 9 focus points (9 FP), 51 focus points (51 FP), face region (FR), and out-of-focus (OF). The left column shows the average user preference. The small plots on the right show user preference (lower plots) and lens movements (upper plots in grey) for individual scenes.

of lens motion and is the least preferred. In the third category that contains a face in the background, users do not seem to have any strong preference, as seen by the near-equal distribution of votes across 51 FP, GB, and FR, all of which interestingly have roughly the same amount of lens motion (third row in Figure 7). It is also important to note that in agreement with our findings for the first two categories, the objective with the highest amount of lens movement, which in this case is the 9 focus points (9 FP) objective, is the least preferred.

The out-of-focus (OF) objective is preferred the least across all three categories although it has the least amount of lens motion. This agrees with the common wisdom that at least a part of the scene has to be in focus, and simply minimizing the amount of lens motion does not induce a higher preference.

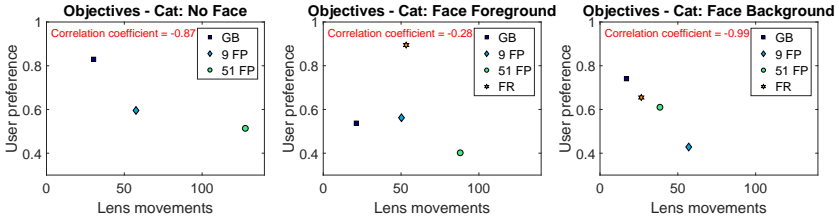


Fig. 8: The relationship between user preference and number of lens movements for AF objectives for the three scene meta-categories. Left: no face (NF). Middle: face in foreground (FF). Right: face in background (FB).

## 6 Discussion and summary

This paper has developed a new software platform and dataset focused on autofocus for video capture with smartphone cameras. To this end, we constructed a hardware setup that allows dynamic scenes to be accurately “replayed”. Using this environment, we analyzed representative smartphone cameras’ AF behaviour under 10 different scenes with various motions, backgrounds, and objects (including an object serving as a proxy for a human face). We also captured these scenes with discrete time points, producing a 4D temporal focal stack dataset for use in AF research. The overall dataset consists of 33,000 smartphone camera images and will be made publicly available. We also developed an AF platform that allows the development of AF algorithms within the content of a working camera system. API calls allow algorithms to simulate lens motion, image access, and low-level functionality, such as phase and contrast detection. This platform also restricts an AF algorithm to operate within a real camera environment, where lens motion that is directly tied to the systems clock cycle and scene motion is required to access different images in the focal stack.

From our analysis of the cameras’ AF behaviour we examined four high-level AF objectives—namely, global, 9 focus points, 51 focus points, and face region. Using our AF platform, we implemented these high-level AF objectives to produce several video outputs that were used in a user study. Because our AF platform allowed accurate analysis of the underlying AF algorithms, we were able to determine that user preference is correlated higher to the overall lens motion than the actual scene objective used. For scenes with faces, focusing on the face (when sufficiently large) took priority, followed by the amount of lens motion. While these findings are somewhat intuitive (e.g., no one likes a scene with too much lens wobble), as far as we are aware, this is the first study to confirm these preferences in a controlled manner. We believe having access to our temporal focal stack dataset and AF platform will be a welcomed resource for the research community.

**Acknowledgments** This study was funded in part by the Canada First Research Excellence Fund for the Vision: Science to Applications (VISTA) programme and an NSERC Discovery Grant.

## References

1. Ohsawa, K.: Focus detecting device and method of operation (1996) US Patent 5,530,513.
2. Inoue, D., Takahashi, H.: Focus detecting device and camera system using the same device (2009) US Patent 7,577,349.
3. Śliwiński, P., Wachel, P.: A simple model for on-sensor phase-detection autofocusing algorithm. *Journal of Computer and Communications* **1**(06) (2013) 11
4. Jang, J., Yoo, Y., Kim, J., Paik, J.: Sensor-based auto-focusing system using multi-scale feature extraction and phase correlation matching. *Sensors* **15**(3) (2015) 5747–5762
5. Jeon, J., Lee, J., Paik, J.: Robust focus measure for unsupervised auto-focusing based on optimum discrete cosine transform coefficients. *IEEE Trans. on Consumer Electronics* **57**(1) (2011)
6. Vuong, Q.K., Lee, J.w.: Initial direction and speed decision system for auto focus based on blur detection. In: *Consumer Electronics (ICCE), 2013 IEEE International Conference.* (2013)
7. Shih, L.: Autofocus survey: a comparison of algorithms. In: *Digital Photography III.* Volume 6502. (2007) 65020B
8. Mir, H., Xu, P., Van Beek, P.: An extensive empirical evaluation of focus measures for digital photography. In: *Digital Photography X.* Volume 9023. (2014) 90230I
9. Nakahara, N.: Passive autofocus system for a camera (2006) US Patent 7,058,294.
10. Mousnier, A., Vural, E., Guillemot, C.: Partial light field tomographic reconstruction from a fixed-camera focal stack. *arXiv preprint arXiv:1503.01903* (2015)
11. Li, N., Ye, J., Ji, Y., Ling, H., Yu, J.: Saliency detection on light field. In: *CVPR.* (2014) 2806–2813
12. Baxansky, A.: Apparatus, method, and manufacture for iterative auto-focus using depth-from-defocus (2012) US Patent 8,218,061.
13. Zhang, W., Cham, W.K.: Single-image refocusing and defocusing. *IEEE Trans. on Image Processing* **21**(2) (2012) 873–882
14. Cao, Y., Fang, S., Wang, Z.: Digital multi-focusing from a single photograph taken with an uncalibrated conventional camera. *IEEE Trans. on image processing* **22**(9) (2013) 3703–3714
15. Tang, H., Cohen, S., Price, B., Schiller, S., Kutulakos, K.N.: Depth from defocus in the wild. In: *CVPR.* (2017)
16. Alexander, E., Guo, Q., Koppal, S., Gortler, S., Zickler, T.: Focal flow: Measuring distance and velocity with defocus and differential motion. In: *EECV.* (2016) 667–682
17. Suwajanakorn, S., Hernandez, C., Seitz, S.M.: Depth from focus with your mobile phone. In: *CVPR.* (2015) 3497–3506
18. Levoy, M.: Light fields and computational imaging. *Computer* **39**(8) (2006) 46–55
19. Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., Hanrahan, P.: Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR* **2**(11) (2005) 1–11
20. Sheinin, M., Schechner, Y.Y., Kutulakos, K.N.: Computational imaging on the electric grid. In: *CVPR.* (2017)