Generalized Autofocus

Daniel Vaquero[1,2], Natasha Gelfand[1], Marius Tico[1], Kari Pulli[1], Matthew Turk[2]

[1]Nokia Research Center Palo Alto
{natasha.gelfand,marius.tico,kari.pulli}@nokia.com

[2]University of California, Santa Barbara
{daniel,mturk}@cs.ucsb.edu

Abstract

All-in-focus imaging is a computational photography technique that produces images free of defocus blur by capturing a stack of images focused at different distances and merging them into a single sharp result. Current approaches assume that images have been captured offline, and that a reasonably powerful computer is available to process them. In contrast, we focus on the problem of how to capture such input stacks in an efficient and scene-adaptive fashion. Inspired by passive autofocus techniques, which select a single best plane of focus in the scene, we propose a method to automatically select a minimal set of images, focused at different depths, such that all objects in a given scene are in focus in at least one image. We aim to minimize both the amount of time spent metering the scene and capturing the images, and the total amount of high-resolution data that is captured. The algorithm first analyzes a set of low-resolution sharpness measurements of the scene while continuously varying the focus distance of the lens. From these measurements, we estimate the final lens positions required to capture all objects in the scene in acceptable focus. We demonstrate the use of our technique in a mobile computational photography scenario, where it is essential to minimize image capture time (as the camera is typically handheld) and processing time (as the computation and energy resources are limited).

1. Introduction

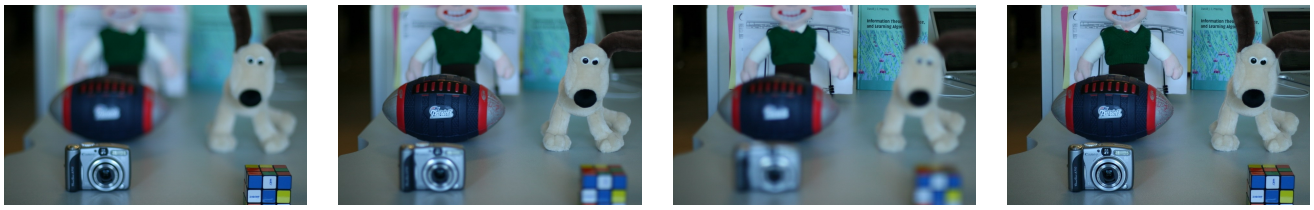
The size of a camera's aperture provides a trade-off between the depth of field and the amount of light that is captured by an image with a given exposure time. For an image to be sharp across a large range of depths in the scene, a small aperture is required. However, decreasing the aperture size is not always feasible. Most low-end cameras, such as those found in cellphones, have a fixed aperture size. While the aperture is usually small enough that most of the scene is in focus, photos that require close focusing, such as macro shots, exhibit a shallow depth of field. For cameras that do have control over the size of the aperture, decreasing the aperture size until the entire scene is in focus may not be feasible due to the lack of available light. Small apertures require slower shutter speeds, which can result in image blur due to handshake and motion of objects in the scene.

An alternative method for acquiring a single image with a large depth of field is to capture a set of photos focused at different depths, also known as a focal stack (Figure 1). This allows for using larger apertures and shorter exposure times, but each photo will have some areas that are sharp and some that are blurry. The images are then combined to produce a composite that is in focus everywhere using an all-in-focus image fusion algorithm [2, 10].

A large body of prior work in computer graphics, vision, and image processing has addressed the problem of combining focal stacks into all-in-focus images (see Section 2). These algorithms assume that a stack has already been captured and aim at producing the sharpest possible result, typically by first computing a sharpness mask for each image using a contrast measuring operator and then combining the images by mosaicking the sharpest regions.

The problem of how to capture focus stacks efficiently has received less attention. Hasinoff et al. [8] cover the entire range of depths from the closest focus distance to infinity efficiently with the minimum number of shots, given the characteristics of the particular camera and lens. However, the number of images required also depends on the scene itself: scenes with large depth variations will require more images, while scenes where all objects are close to each other require fewer images. We address the efficient capture of a focal stack by analyzing both the properties of the camera and the geometry of the given scene.

Why is capturing the smallest number of images important? After all, if the entire depth range is captured, the final result will still have all objects in focus, even if some images are focused on depths that do not contain anything in the scene and are therefore redundant. We are interested in both capturing and processing all the images online on



(a) (b) (c) (d)

Figure 1. All-in-focus imaging. A focal stack (a–c) is captured by focusing at different distances, and the images are fused to obtain an all-in-focus result (d). This example was captured by manually focusing a Canon 40D camera and fused offline using the algorithm in Agarwala et al. [2].

programmable cameras that typically have less processing power and memory than desktop computers. Capturing only as many images as needed for a given scene has several advantages:

- Shorter total capture time, which improves the user experience and has less risk of object motion in the scene. In practical handheld photography scenarios, motion due to handshake is significant and difficult to compensate for through image registration, especially when close objects are present (due to parallax);
- Less total data that needs to be captured. A dense focal stack may be too large to fit into the main memory of a programmable camera or a camera phone;
- Shorter processing time. All-in-focus algorithms are computationally demanding, some requiring expensive optimizations [2] or multiresolution operations [10]. Eliminating useless images from processing can dramatically reduce computation time;
- Processing more images than needed increases the likelihood of stitching artifacts, potentially lowering the quality of the result.

With the above requirements in mind, we set out to develop an all-in-focus capture and processing system that adapts to scene geometry. Our approach is inspired by passive autofocus techniques in digital photography, which, assuming a fixed aperture size, a static camera, and a static scene, analyze a stream of preview images while the lens sweeps to select a single best plane of focus, without relying on active light emission. We propose the *generalized autofocus* algorithm that automatically selects a minimal set of images, focused at different depths, such that each part of the scene is in focus in at least one of the images. The algorithm runs in real-time and is based on sweeping the lens while analyzing a metering stack of low-resolution sharpness measurements of preview images. We estimate the number of images in the minimal set and where these images need to be focused. We then capture the focal stack given by such images in high resolution, and merge it into an all-in-focus result directly on the camera.

We apply our technique in the mobile computational photography scenario on a Nokia N900 smartphone. However, the technique is platform-independent and could be implemented on any programmable camera that allows control over the focus mechanism. We demonstrate how our scene-adaptive capture algorithm results in faster capture and processing times than the techniques that capture and merge images that cover the entire depth of field independently of the scene. In summary, the main contributions of our work are:

- A novel scene-adaptive approach for selecting the minimal set of images to be captured for synthesizing an all-in-focus image. We analyze different strategies for metering the scene for sharpness, and propose a strategy that optimizes for speed while guaranteeing coverage of the entire depth range. We also introduce a plane-sweep algorithm for determining the minimal set of required images given the metering stack.
- The implementation of the first scene-adaptive, all-in-focus imaging system that runs entirely on a camera.

2. Related Work

Several techniques have been proposed to generate all-in-focus images from focal stacks based on multi-resolution decompositions [4, 5, 10], energy minimization [2] or focal connectivity analysis [7]. These techniques assume that a full focal stack has been captured offline and merge the sharpest regions of each image into an all-in-focus composite.

The problem of how to efficiently capture a focal stack has received less attention. The work of Hasinoff et al. [8] aims to minimize the total time spent capturing the images given a desired depth of field requested by the user. However, the method is not adaptive to the scene. It assumes that the entire requested set of depths needs to be in focus, even if some depths contain no objects. In contrast, our approach captures only those high-resolution images that actually contain sharp areas.

The ability to automatically select a distance where to focus is present in most modern digital cameras. Passive contrast-based autofocus techniques such as [9, 13] move the lens through a range of depths in the scene while looking for the maxima of a variety of sharpness measures [3]. The

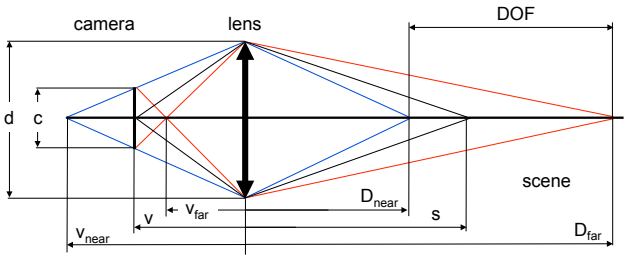


Figure 2. Depth-of-field boundaries.

maximum of the sharpness measure identifies the depth in the scene where most objects are in focus. Our technique extends this idea to the selection of several focal distances, such that the combined depths of field capture all objects in a given scene in acceptable sharpness.

To determine where the objects are located in a given scene, one can apply a variety of depth-from-focus algorithms [9, 12]. However, since these algorithms reconstruct the entire scene geometry, they are computationally expensive. We would like the focus computation to take not much more time than is needed to sweep the lens through its entire focus range, therefore we make a tradeoff between speed and accuracy. We meter based on a stack of low-resolution sharpness maps of the scene, which is fast but can miss some small isolated objects.

3. Generalized Autofocus

We now introduce a technique to efficiently meter the scene for sharpness and determine the minimal set of images that will compose the focal stack. The basic idea is to first capture a metering stack of low-resolution preview images and select the minimal subset of needed images, which are then recaptured as a full-resolution focal stack.

3.1. Depth-of-Field Limits

The relationship between the depth of field boundaries and the subject distance to the camera is important in the description of our method. In Figure 2, we show the geometry when a subject at distance s from a symmetrical lens with an aperture of diameter d is in focus at the focal distance v. Objects at distances D_{far} and D_{near} are imaged as blur spots, and would be in focus at focal distances v_{far} and v_{near}, respectively. When the diameter of the acceptable circle of confusion is c, D_{near} and D_{far} correspond to the boundaries of the depth of field. Let $N = \frac{f}{d}$ be the lens f-number, where f is the focal length and d is the aperture diameter. The depth of field limits are

$$D_{\text{near}} = \frac{sf^2}{f^2 + Nc(s-f)}; D_{\text{far}} = \frac{sf^2}{f^2 - Nc(s-f)}. \quad (1)$$

In practice, the minimum acceptable size for the circle of confusion is usually about twice the pixel size, due to blur

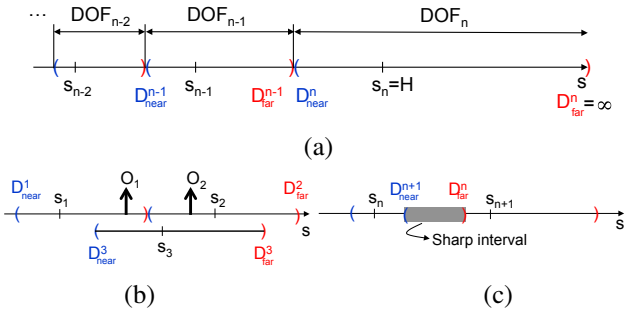


Figure 3. Capture strategies. (a) Maximal and disjoint depth of field intervals. (b) Capturing according to (a) may be suboptimal; focusing at s_1 and s_2 requires two images to cover objects O_1 and O_2, but focusing at s_3 uses a single image. (c) Capture with a moving lens; objects in the intersection of the depths of field at the beginning and at the end of the exposure are in focus.

introduced by, e.g., sensor demosaicking and light diffraction.

3.2. Scene Measurement

The first step of our method consists of capturing a metering stack of low-resolution preview images of the scene under different focus settings. Like traditional passive autofocus techniques, we assume that the aperture size is fixed. In this section, we discuss strategies for varying the focus distance while the images are captured, with three practical objectives:

1. Every object in the scene that lies between the closest and the farthest focus distances (typically infinity) must be sharp in at least one of the images. This is to guarantee that the final result will indeed be an all-in-focus image.
2. The total capture time should be as short as possible to minimize complications due to handshake and moving objects in the scene.
3. The number of images in the high-resolution focal stack (to be captured in a subsequent step) should be minimized.

We finally propose a capture strategy that achieves (1) and (2) while providing a good balance with (3).

Maximal and disjoint depth-of-field intervals. The optimal scene-independent way to achieve objective (1) is to capture the images that have maximal and disjoint depth of field intervals for every image, as in Figure 3(a). The maximal depth of field interval $DOF_n = (D^n_{\text{near}}, D^n_{\text{far}})$ is the one obtained when focusing at the *hyperfocal distance* H where $D_{\text{far}} = \infty$. The distance H can be calculated using Equation 1, and at this setting we get $(D^n_{\text{near}}, D^n_{\text{far}}) = (\frac{H}{2}, \infty)$. The next maximal interval $DOF_{n-1} = (D^{n-1}_{\text{near}}, D^{n-1}_{\text{far}}) = (D^{n-1}_{\text{near}}, D^n_{\text{near}})$ can be computed by setting D_{far} to D^n_{near} and using Equation 1. This strategy minimizes the number of images required to cover the entire depth range, as the depth

of field intervals are maximal, and achieves objective (1).

However, moving the lens from near to far focus while stopping at the optimal positions to capture an image is suboptimal with respect to objective (2), as the total capture time in this case is the sum of the time t_{NF} to move the lens, the total exposure time for the pictures taken, and the latencies involved with starting lens motion and bringing it to a complete stop between pictures. This overhead can be significant. On a Nokia N900 camera taking pictures indoors, each exposure takes 40 ms, and starting the lens movement takes time on the order of microseconds. On heavier SLR lenses, this delay may be considerably larger. Hence, we ideally would like to move the lens from near to far focus once while capturing the metering stack.

Capturing images with maximal and disjoint depths of field may also be suboptimal with respect to objective (3). To illustrate, consider the example in Figure 3(b). Selecting two images, focused at s_1 and s_2, so that objects O_1 and O_2 are in focus is not optimal, as capturing a single image focused at s_3 would contain both objects O_1 and O_2 in focus.

Binary search. One could do a binary search in the depth range to search for objects in focus, by capturing more images to optimize for objective (3). However, this is equivalent to performing a full depth estimation of the scene, and it significantly increases the capture time (contradicting objective (2)) due to the need of moving the lens between multiple spots to capture additional pictures.

Single lens sweep. We achieve (1) and (2) while providing a good balance with (3), by sweeping the lens once from near to far focus, and capturing the metering stack while the lens is moving. The total capture time is t_{NF} plus the latency of starting and stopping the lens only once, achieving (2); and objective (1) is reached when the lens is moved slowly enough. We now derive an expression for the maximum speed with which the lens can move, given a desired exposure time for each image in the metering stack.

The limits of the depth of field change as the lens moves. For a given exposure, the effective depth range that will be acceptably sharp is the intersection of the depth of field ranges at the beginning and at the end of the exposure; see Figure 3(c). For an exposure starting when the lens is focused at s_n, it is desirable to move the lens slowly enough so that s_n is still contained in the depth of field at the end of the exposure, keeping objects at s_n in focus. Let s_{n+1} be the subject distance at which the camera is focused at the end of the exposure, and D_{near}^{n+1} be the near limit of the corresponding depth of field; the above condition can be written as $D_{\text{near}}^{n+1} \leq s_n$. Therefore, the maximum speed for the lens is

$$S = \frac{s_{n+1} - s_n}{e}, \tag{2}$$

where $D_{\text{near}}^{n+1} = s_n$ and e is the exposure time. Setting D_{near} to s_n in Equation 1 and solving for s, and then setting s_{n+1} to s in Equation 2 gives

$$S = \frac{s_n N c (s_n - f)}{e(f^2 - s_n N c)}, \tag{3}$$

a function of known values. On a Nokia N900, $S = 2.637$ cm/s for an exposure time of 39.6 ms (used in our experiments indoors), and $S = 8.096$ cm/s for an exposure time of 12.9 ms (used in our experiments outdoors). This analysis ignores magnification effects resulting from changing the focal distance. However, these effects are negligible in our target application, as we use very coarse sharpness maps.

This strategy also has advantages with respect to objective (3). Usually, $s > f$, and $D_{\text{far}} - s > s - D_{\text{near}}$ for a given subject distance s. It follows that $s_{n+1} < D_{\text{far}}^{n}$, i.e., the depths of field for subsequently captured images overlap. These overlaps increase the chances of minimizing the number of images in the focal stack in situations such as the one depicted in Figure 3(b).

3.3. Sharpness Analysis

Once the metering stack has been captured while sweeping the lens, the goal is to localize the sharpest regions and determine the planes where to focus when capturing the high-resolution focal stack. We use low-resolution preview images to obtain coarse sharpness maps, which can be computed in real-time. A sharpness map is an image of the scene such that each pixel value is in the $[0, 1]$ range, indicating the level of sharpness of the objects at the region represented by the pixel. In our implementation (Section 4), we used 16×12 sharpness maps computed by the camera hardware. The sharpness analysis procedure consists of two classification steps:

1. Foreground/background classification. This step segments the 16×12 space into foreground (areas that appear sharp on at least one of the images in the metering stack but are blurry on the others) and background (areas that have similar sharpness in every image of the metering stack, such as a white wall).
2. Sharp/blurry classification. For a given foreground pixel, this procedure attempts to determine the images on the metering stack in which the pixel appears sharp, and the images in which the pixel is blurred.

Let T be the number of images in the metering stack. For a pixel (j, k) in the map, $Sp(i)$ denotes its sharpness at image i in the stack. In order to perform foreground/background classification, the standard deviation σ of the sharpness values $\{Sp(i)\}$, $i \in \{1, \ldots, T\}$ is computed. The pixel is classified as foreground if $\sigma > t_1$, and as background otherwise. In our implementation, the threshold $t_1 = 0.05$ was experimentally determined to yield good results.

The next step consists of performing sharp/blurry classification for the pixels determined to be part of the fore-

ground. For a pixel (j, k) in the foreground, let M be an image that maximizes its sharpness, i.e., $Sp(M) \geq Sp(i)$, $\forall i \in \{1, \ldots, T\}$. We classify a pixel at level (i) as sharp/blurry by the following criteria:

- (i) is sharp if $i = M$;
- (i) is sharp if $i < M$, $Sp(M) - Sp(i) < t_2$, and $(i+1)$ is sharp;
- (i) is sharp if $i > M$, $Sp(M) - Sp(i) < t_2$, and $(i-1)$ is sharp;
- (i) is blurry otherwise.

The threshold t_2 allows for some tolerance on the level of blurriness when deciding whether or not a pixel is sharp, by also labeling pixels with sharpness values close to the maximum sharpness in the stack as sharp. In our implementation, the threshold t_2 was experimentally set to 0.2.

3.4. Choice of the Minimal Set of Images

The sharpness analysis step from Section 3.3 outputs a *binary sharpness map* for each image in the metering stack, such that sharp pixels have value 1, and blurry pixels have value 0. It also returns a *binary foreground mask* that represents the foreground and background regions in the scene. Background pixels are not relevant in the computation of the final result; they could be picked from any image, since they all look similar. Hence, only foreground pixels are considered. The goal is to select a collection of images in the stack such that the union of their sharp foreground pixels is equal to all foreground pixels, and the size of this collection is minimized. This problem is known as the *set cover problem* [6], which in an abstract form is NP-complete.

However, we observe that two key properties cannot be assumed in general set cover problems, but are present in our problem:

- There is an ordering between sets, given by the order on how the images have been captured;
- For a given pixel in the sharpness map, its sharpness varies as a function of the images in the stack. This function is unimodal [12]; therefore, it becomes a box function once it is thresholded (Section 3.3).

We exploit this additional knowledge about the problem domain, and propose a *plane sweep* algorithm for selecting the minimal subset of required images. The idea is to simulate the lens sweeping process through the different depths of the scene, by analyzing the images in the sequence they were captured. A set A of active pixels and a set P of processed pixels are kept in memory. The set A contains pixels that appear sharp in the current image; P contains the pixels for which an image has already been included in the final result and do not need to be examined again. For every new image, the following sets of pixels are computed from P^C (the complement of P):

- F_0: pixels that were sharp in the previous image, but became blurry in the current image;
- F_1: pixels that were blurry in the previous image, but became sharp in the current image.

If $F_0 \neq \emptyset$, an image must be added to the result, as the pixels in F_0 will not be sharp in any further images on the stack. The previous image is added to the result, the pixels in A are added to P (as they have now been covered), and A is emptied. If $F_1 \neq \emptyset$, then the new sharp pixels are added to the active set.

A pseudocode version of the algorithm is shown below. It is linear in the number of captured images, as each image in the stack is analyzed only once. Each binary sharpness map in the stack S can be viewed as a set, such that a pixel at a given sharpness map is in the set if and only if it is sharp.

```
function minImageSet(stack of binary sharpness maps S,
                     foreground mask M)

// consider only foreground pixels
for each set B in S; do
  B = intersection(B, M);
end

S = S.add(empty);       // sentinel
set Active = empty;     // current list of sharp pixels
set Processed = empty; // pixels for which an image
                       // has been chosen
list selectedImages = empty; // result
i = 1;
for each set B in S; do
  // sharp pixels for which an image
  // has not yet been chosen
  set Sharp = B - Processed;
  set F0 = Active - Sharp;
  set F1 = Sharp - Active;

  // active sharp pixels that became blurry
  if F0 != empty; then
    // add the previous image to the result
    selectedImages.add(i-1);
    // mark the active pixels as processed
    Processed = union(Processed, Active);
    Active = empty;
  end

  // new sharp pixels
  if F1 != empty; then
    // add the new sharp pixels to the active set
    Active = union(Active, F1);
  end

  i = i + 1;
end
return selectedImages;
```

4. Implementation

We have implemented and tested a complete system for generalized autofocus on a Nokia N900 smartphone modified with the open-source FCam programmable-camera software stack [1] (`http://fcam.garage.maemo.org`). For a demonstration, please refer to the supplementary video. The N900 runs the Linux-based Maemo operating system, and has a 5 megapixel camera, 600 MHz OMAP 3 processor and 256 MB of RAM. Similar to most cameraphones and low-end consumer cameras, the lens has a short focal length of 5.2 mm, and a fixed and small aperture (f-number 2.8). The FCam API allows us to control

the lens focus by setting it to a specific distance, as well as specifying the lens motion speed. Streaming low-resolution (640×480) frames while the lens is moving is also possible.

Our focal stack capture application provides a viewfinder that auto-adjusts the exposure time by metering the light in the scene, and triggers the generalized autofocus routine when the user presses the shutter button halfway. The calculated exposure time is used to determine the speed with which the lens should be moved (Equation 3), and the lens is then swept from near focus (5 cm) to far focus (infinity) while the metering stack is captured. Moving the N900 lens at the slowest possible speed leads to a total sweep time of about 2 seconds.

The N900 hardware provides sharpness maps computed by dividing the image into a 16×12 grid, running the images through a $[-1\ 2\ -1]$ filter, and summing the absolute value of the responses for all pixels in each block, resulting in a 16×12 sharpness map. These maps are fed to the sharpness analysis procedure from Section 3.3. The plane sweep algorithm is then run to choose the minimal subset of images to compose the high-resolution focal stack. The algorithm runs in real-time; for a metering stack of 57 frames, the total time for analyzing the sharpness maps and computing the minimal subset is of about 10 ms.

Once the minimal set of images is determined, the user can trigger the capture of the high-resolution focal stack by fully pressing the shutter button. The lens is moved back to the desired positions and full-resolution (5 MP) images are captured whenever the lens comes to a complete stop. The images in the focal stack are corrected for differences in magnification. We found the scale factor to be 1.0055 per diopter of lens movement, by analyzing a focal stack of a planar checkerboard image. The images are then registered to each other (using [14]) to account for translation and rotation due to handshake. Finally, we use an open-source off-the-shelf implementation [11] of the Exposure Fusion algorithm [10] configured to maximize sharpness to generate the final all-in-focus result.

5. Experimental Results

We used our system to test the algorithm in a few scenes with different variations in depth, by capturing images using a handheld N900. In our first example, the goal was to create an all-in-focus image from a scene with three depth layers. We compared three different capture strategies. The first strategy was our generalized autofocus algorithm, which correctly determined that three images were needed, one for each layer. Figure 4(a-c) shows the three images in the focal stack, after magnification correction and registration. The cropped areas shown below each image illustrate the different levels of blur in different regions. Figure 4(e) shows the fused result. The second strategy consisted of capturing a full focal stack of 24 images comprising the maximal and disjoint depth of field intervals (Section 3.2) computed for a circle of confusion of 4.4 microns (twice the N900's pixel size), and fusing them. Figure 4(d) shows the all-in-focus result. Finally, we obtained a third result by simply running a standard autofocus algorithm and capturing a single image. Figure 4(f) displays the image obtained in this way.

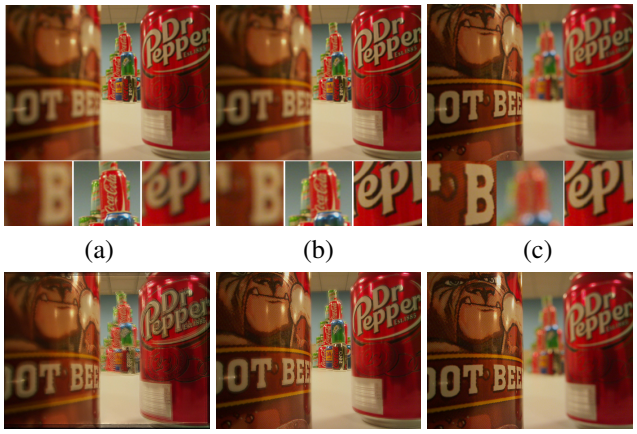


Figure 4. Scene with three depth layers: (a-c) the input images focused at different distances, (d) full focal stack, (e) generalized autofocus, and (f) standard autofocus. The camera was handheld during image capture, leading to camera motion and accentuated parallax for close objects, which are very difficult to compensate for using image registration for a full stack of 24 images – see Figure 5. On the other hand, registration works much better on the reduced set of 3 images chosen by our algorithm.

Figure 5. Registration issues due to parallax. (a-b) two frames from the full focal stack after registration. (c) the yellow square regions from (a) and (b). Notice the different distances between the brown and green cans due to parallax, caused by handshake.

As expected, both fusion approaches produce all-in-focus images, while only one of the layers appears focused in the single image result. However, using the full focal stack (Figure 4(d)) resulted in several artifacts due to the camera being handheld during image capture, which led to camera motion and accentuated parallax for close objects. This is very difficult to compensate for using the registration algorithm [14]. Figure 5 illustrates this problem. On the other hand, registration works much better on the minimal set of images given by our technique, resulting in fewer

artifacts (Figure 4(e)). This suggests that reducing the number of captured images to a minimum may be beneficial in terms of quality, as the fusion process will be less sensitive to motion.

We also compared the capture times for the three approaches. Table 1 lists the times spent on each step of the imaging process. The first row shows the time for the autofocus step, i.e., sweeping the lens and analyzing the metering stack in order to determine the best focus position(s). This time is dominated by the lens sweep time. Standard autofocus stops once a peak in sharpness is found, as it follows a Fibonacci search strategy [9]. On the other hand, generalized autofocus needs to sweep through the entire depth range to make sure all depths are covered. The capture time for high-resolution images is given by the exposure times of the photos and the times required for moving the lens between different focal distances, as well as latencies introduced by the memory management system. The generalized autofocus approach is much more efficient than a full focal stack capture in this case, as it avoids redundant images; standard autofocus is faster as it only requires the capture of a single image, but it does not produce an all-in-focus result.

Finally, we compared the times of the fusion step (both full focal stack and the smaller stack of the generalized autofocus approach) on a Linux PC (Intel Core 2 Duo, 2 GHz, with 2 GB of RAM) and on the Nokia N900. As shown in the third and fourth rows of Table 1, the fusion time is shorter for the generalized autofocus approach due to the reduced number of images. On the N900, processing power and memory are limited, and the full focal stack approach was not feasible; after 30 minutes of processing, the phone was forced to reboot due to memory constraints.

In summary, when compared to a full focal stack capture, the generalized autofocus approach provides a faster way of capturing a focal stack, allows for decreased processing time, minimizes problems with artifacts due to motion, and opens up the possibility of processing focal stacks directly on mobile devices. In addition, generalized autofocus is able to produce an all-in-focus image, while standard autofocus may contain blurry regions. This gain in depth of field comes at the expense of time: the generalized autofocus method needs to evaluate images along the entire depth range and possibly captures more than one image, while standard autofocus usually stops the lens sweep once a peak in sharpness is found and captures a single image.

We have also performed the comparison for two other scenes: a close-up scene (macro photography) and a scene consisting of distant trees, such that focusing at a single plane is sufficient to have the entire scene in focus. Figure 6 shows the results for the macro photography scene. In this case, four images were selected, and an analysis similar

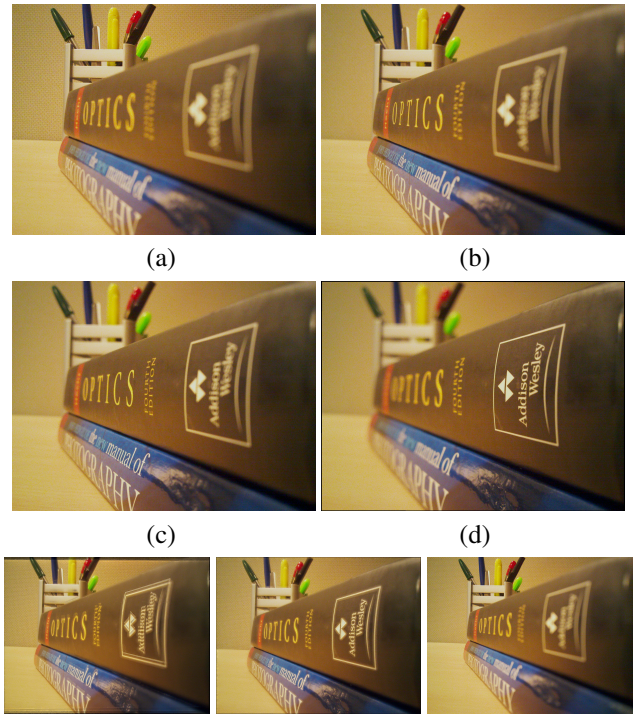


Figure 6. Macro photography scene: (a-d) the input images focused at different distances, (e) full focal stack, (f) generalized autofocus, and (g) standard autofocus.

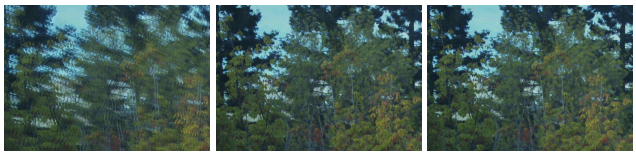


Figure 7. Scene with a single depth layer (far away objects): results for (a) full focal stack, (b) generalized autofocus, and (c) standard autofocus. Blindly using a full focal stack leads to artifacts due to movement of the trees.

to the first scene applies. For the scene with a single layer (Figure 7), the generalized autofocus algorithm correctly selected a single image, and the technique reduces to standard autofocus, while blindly fusing a full focal stack leads to artifacts due to movement of the trees. Table 1 shows the running times for all experiments.

Limitations. Our approach generalizes traditional autofocus approaches to enable the capture of focal stacks, and inherits some intrinsic limitations of passive autofocus techniques:

- The sharpness analysis heavily depends on contrast. Therefore, noisy images and dark environments may lead to erroneous results;
- A static camera and a static scene are assumed. The

Table 1. Capture and processing times for all experiments. GA = generalized autofocus, FS = full focal stack, SA = standard autofocus. (-) denotes a step not applicable to a given experiment. (X) denotes that a full focal stack could not be processed on the N900 due to its limited memory. "e" denotes the exposure time for each captured image.

	Cans (e = 39.6 ms)			Macro (e = 39.6 ms)			Trees (e = 12.9 ms)		
	GA	FS	SA	GA	FS	SA	GA	FS	SA
focusing	470 ms	-	210 ms	520 ms	-	180 ms	100 ms	-	40 ms
capture	1.67 s	8.92 s	0.56 s	1.59 s	7.31 s	0.41 s	0.59 s	7.53 s	0.57 s
fusion (PC)	40.61 s	290 s	-	50 s	281 s	-	-	277 s	-
fusion (N900)	674 s	X	-	588 s	X	-	-	X	-
total (PC)	42.75 s	298.92 s	0.77 s	52.11 s	288.31 s	0.59 s	0.69 s	284.53 s	0.61 s
total (N900)	676.14 s	X	0.77 s	590 s	X	0.59 s	0.69 s	X	0.61 s

technique is thus sensitive to camera movements during the capture stage and dynamic changes in the scene. However, focusing is performed at very coarse resolution, so it is robust to small amounts of motion. Also, our technique makes the capture of focal stacks less sensitive to motion-related issues, as the number of acquired high-resolution images is minimized.

6. Concluding Remarks

In this paper, we proposed the generalized autofocus algorithm for automatic capture of focal stacks in an efficient and scene-adaptive fashion. The algorithm meters the scene in the time that is required to sweep the lens from minimum to maximum focus distance and captures only the high resolution images that are necessary to produce an all-in-focus composite of the given scene. This allows the user to treat all-in-focus image capture just like regular photography, since no user adjustment of parameters is required. We demonstrate the feasibility of our algorithm by implementing a complete all-in-focus imaging system on a programmable camera.

Several directions are possible for future work. While our current hardware platform does not allow for changing the camera's aperture, the algorithm can be naturally extended to adjust both the focus distance and the aperture based on available light. A further extension would be to vary even more capture parameters, such as exposure time and sensor gain, to optimize not just depth of field but also exposure in different parts of the scene. Finally, while we focused on a fully automatic metering and fusion solution in this work, the problem of user-guided focusing is also interesting to explore. Given that modern cameras and cell phones are often equipped with a touchscreen, one can imagine "touch-to-focus" applications where the user indicates objects of interest in the scene. The camera then captures and merges images where such objects are in focus, while providing pleasing background blur in other areas.

References

[1] A. Adams, E.-V. Talvala, S. H. Park, D. E. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico, H. P. A. Lensch, W. Matusik, K. Pulli, M. Horowitz, and M. Levoy. The Frankencamera: an experimental platform for computational photography. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, 2010.

[2] A. Agarwala, M. Dontcheva, M. Agrawala, S. Druker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, 2004.

[3] T. Aydin and Y. Akgul. A new adaptive focus measure for shape from focus. In *British Machine Vision Conf.*, 2008.

[4] R. C. Bilcu, S. Alenius, and M. Vehviläinen. A novel method for multi-focus image fusion. In *Intl. Conf. on Image Processing*, 2009.

[5] A. Castorina, A. Capra, S. Curti, E. Ardizzone, and V. L. Verde. Improved multi-resolution image fusion. In *Intl. Conf. on Consumer Electronics*, 2005.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[7] H. Hariharan, A. Koschan, and M. Abidi. An adaptive focal connectivity algorithm for multifocus fusion. In *CVPR Workshop on Image Registration and Fusion*, 2007.

[8] S. W. Hasinoff, K. N. Kutulakos, F. Durand, and W. T. Freeman. Time-constrained photography. In *Intl. Conf. on Computer Vision*, 2009.

[9] E. Krotkov. Focusing. *Intl. Journal of Comp. Vision*, 1987.

[10] T. Mertens, J. Kautz, and F. V. Reeth. Exposure fusion. In *Pacific Graphics*, 2007.

[11] A. Mihal et al. Enblend 4.0. `http://enblend.sourceforge.net`.

[12] S. K. Nayar. Shape from focus. *Carnegie Mellon University, Department of Computer Science, Technical Report*, 1989.

[13] L. Shih. Autofocus survey: A comparison of algorithms. In *SPIE Electronic Imaging*, 2007.

[14] M. Tico and K. Pulli. Low-light imaging solutions for mobile devices. In *Asilomar Conference on Signals, Systems, and Computers*, 2009.