

# Depth from Defocus

Shree K. Nayar

Monograph: FPCV-3-4

Module: Reconstruction I

Series: First Principles of Computer Vision

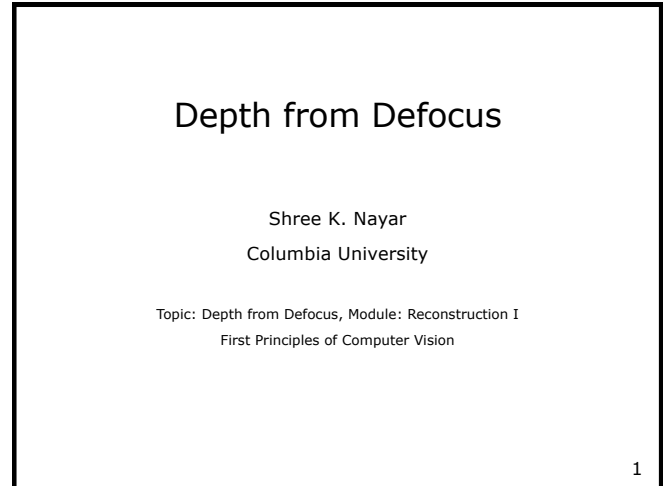
Computer Science, Columbia University

March, 2025

[FPCV Channel](#)

[FPCV Website](#)

In this lecture, we will develop techniques for recovering the three-dimensional structure of a scene by estimating the degree of image blur, or defocus, for each scene point.



Here is an image taken using a camera with a shallow depth of field. The bird 1 is within the depth of the field of the camera and hence is in focus, while the birds that lie outside of the depth of field are out of focus. Note that the degree of defocus increases with the distance of the object from the depth of field. This suggests that if we are able to estimate the amount of blur around a point in the image, then we can estimate the depth of the corresponding point in the scene. By doing this for every point in the image, we can recover the 3D structure of the entire scene.



Unfortunately, the problem of estimating the blur of a point from a single image is not well constrained. Consider a small patch around an image point. To estimate the blur of this patch, we need to have prior knowledge of what the patch would have looked like if it were in focus. If we simply print out the above image on a sheet of paper, attach the sheet to a wall, and take a focused image of the wall, the blurred birds in the printed sheet will appear blurred in the captured image, even though the sheet is in focus. In short, recovering depth from defocus using a single image is an under-constrained problem. This suggests that we need more than one image. We will present methods that use multiple images to compute depth by estimating the differences in defocus of each patch in the multiple images.

We begin by developing a model for defocus. We know from the lecture on image formation that if a point is out of focus, it is projected onto a blur circle in the image. We need a mathematical model for that distribution of light within the blur circle, which is called the point spread function.

Next, we will discuss our first depth estimation method, called depth from focus. In this method, we have a camera with a plane of focus. By controlling the focus setting of the camera, we sweep the plane of focus through the 3D scene.

The focus setting can be varied by either changing the position of the image sensor or moving the lens. While changing the focus setting, we capture a sequence of images, and this set of images is referred to as a focal stack. If we consider how a small image patch varies through the focus stack, we can expect it to come into focus and then go out of focus. If we can detect where in the stack the patch comes into focus, we can use the corresponding focus setting to estimate depth of the scene patch. This process can then be applied to all the image patches to obtain the 3D structure of the scene. We will determine the smallest number of images needed to perform depth from defocus.

Finally, we will look at the more challenging problem of recovering depth from defocus. Instead of acquiring a focal stack, we capture just two images under different focus settings, measure the relative blur of each patch in the two images, and estimate the focus setting for which it would have been in focus. This information is once again used to compute the depth of the scene patch.

Thanks to the Gaussian lens law we discussed in the lecture on image formation, very small changes in distance between the lens and the image sensor can result in significant changes in the position of the plane of focus. This makes it possible to use depth from focus/defocus in a wide variety of settings, ranging from microscopy to photography using mobile phone cameras.

### Depth from Defocus

---

Methods to compute depth by analyzing the degree of focus or defocus in images.

Topics:

- (1) Point Spread Function
- (2) Depth from Focus
- (3) Depth from Defocus

3

In the lecture on image formation, we discussed the relationship between the location of a point in the scene and the blur circle it produces in the image. We will now discuss the distribution of light within the blur circle, called the point spread function.

## Point Spread Function

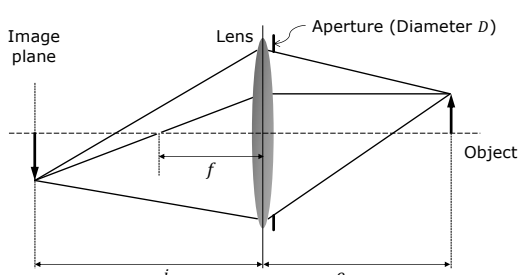
Shree K. Nayar  
Columbia University

Topic: Depth from Defocus, Module: Reconstruction I  
First Principles of Computer Vision

4

First, let us review some of the image formation concepts we have covered, starting with the Gaussian lens law. Here, we have a lens with an aperture of diameter  $D$ , and an object (a single scene point) at a distance  $o$  from the lens. Light from this scene point is refracted by the lens and focused at a point which is at distance  $i$  behind the lens. The plane that is parallel to the lens on which the image is formed is called the image plane. We know that the Gaussian lens law relates  $o$  and  $i$  through the focal length  $f$  of the lens.

### Gaussian Lens Law

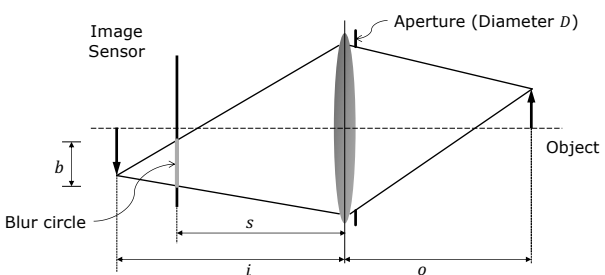


Gaussian lens law:  $\frac{1}{f} = \frac{1}{i} + \frac{1}{o}$       $f$ : Focal length

REVIEW 5

What happens when the image sensor does not lie on the image plane? We know that the image of the scene point is now a circular patch called the blur circle. Assume that the distance of the image sensor from the lens is  $s$ . Using similar triangles, we get the expression 1 for the diameter  $b$  of the blur circle. We can see that if we move the image sensor away from the image plane and towards the lens,  $s$  gets smaller, therefore  $s/i$  gets smaller, and hence the blur diameter  $b$  gets larger. Note that  $b$  only depends on the distance  $o$  of the scene point from the lens, and not the distance of the scene point from the optical axis of the lens. That is, all scene points that lie on a plane that is parallel to the lens will be equally blurred.

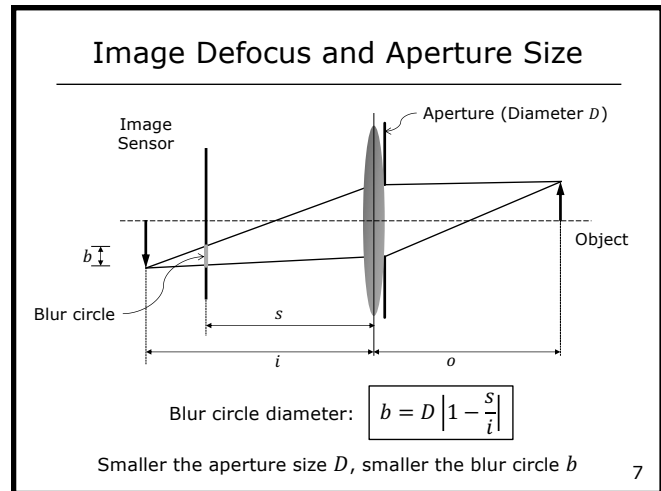
### Image Defocus and Sensor Location



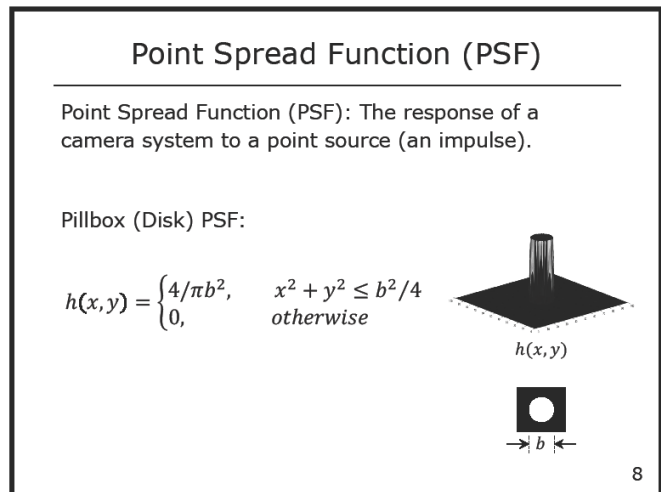
Blur circle diameter:  $b = D \left| 1 - \frac{s}{i} \right|$  1

Farther the sensor from image plane, larger the blur circle  $b$  6

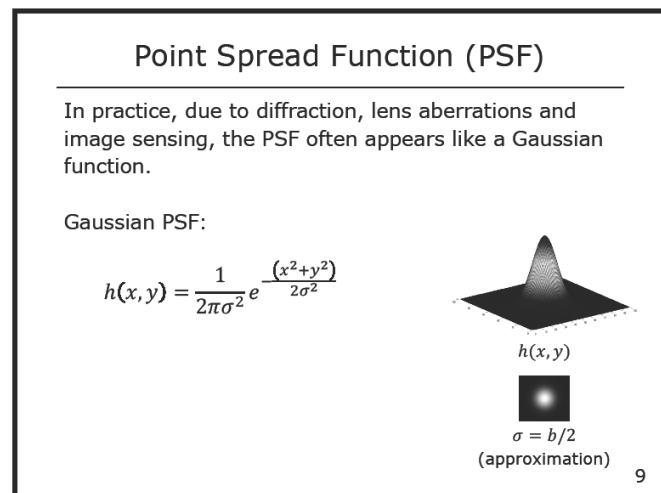
The blur diameter  $b$  is proportional to the diameter  $D$  of the lens aperture. Therefore, if we reduce the diameter of the aperture, we will get a smaller blur circle, as shown here. There are therefore two ways to control the blur of a scene point—moving the image sensor with respect to the lens and changing the aperture of the lens.



The distribution of light within the blur circle is called the point spread function (PSF). As mentioned in the image formation lecture, this function is also called the impulse response of the lens. In the case of an ideal lens, we can assume the light distribution to be uniform within the blur circle. The PSF in this case is called the pillbox function. It has a circular boundary and a constant value inside, which is 4 divided by  $\pi b^2$ , where  $b$  is the diameter of the blur circle. This constant reduces when the size of the blur circle increases, so that the total light energy that falls within the blur circle is always equal to the light energy received by the lens from the scene point.



In practice, however, we rarely get a perfect pillbox function because of many phenomena. One of these is diffraction due to the bending of light at the perimeter of the aperture. Additionally, since the lens is not an ideal one, the PSF will include other effects such as chromatic aberrations, geometric aberrations, and lens imperfections. Finally, while the optical image formed on the image sensor is a continuous one, it is sampled by pixels that have the shape of a box. One way to model the captured discrete image is as the result of the continuous optical image



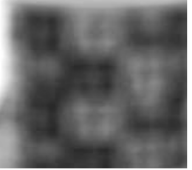


falling on the sensor plane, convolved with a box filter that matches the size of a pixel, and finally sampled at the locations of the pixels. The convolution with the box filter has the effect of further blurring the optical image. Given all the above effects, it is safe to assume that the PSF is not the ideal pillbox function but rather a blurred version of it. For this reason, the PSF is often assumed to be a Gaussian function. The  $\sigma$  of the Gaussian function is then related to the diameter  $b$  of the pillbox function. As a rule of thumb, it is often assumed that  $\sigma$  is equal to half of  $b$ .

Let us look at how the PSF manifests in the case of a defocused image. Consider the focused image  $f_o$  of the mug shown on the left. For the purpose of our discussion here, let us assume that the depth within the image is constant. Then, the defocused image  $f$  on the right can be modeled as the result of convolving the focused image  $f_o$  with the point spread function  $h$ . Note that this model holds true only when the depth within the image is constant. If the depth varies, the PSF would vary from point to point and hence the defocus cannot be assumed to be shift-invariant. Remember that convolution is valid only when the process is linear and shift-invariant. In practice, however, we are interested in estimating the blur at each scene point, independent of other points. So, if the depth of the scene varies gradually, we can assume that the above relation between the focused and defocused images is valid for each small image patch.

### Defocus as Convolution

Within a region where scene depth is constant...


\*

=


$f_o(x,y)$ 
 $h(x,y)$ 
 $f(x,y)$

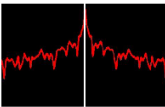
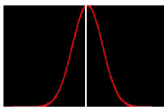
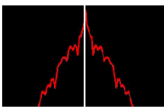
...Defocus is linear and shift invariant, and therefore can be expressed as a convolution.

10

Now, let's take a look at what defocus means in the frequency, or Fourier, domain. Let's assume that the Fourier transform of the original focused image is  $F_0(u, v)$ . For simplicity, we are only showing 1D slices of the magnitude component of 2D Fourier transforms here. Since defocus is a convolution in the spatial domain, we know it is a multiplication in the Fourier domain. So, the Fourier transform  $F(u, v)$  of the defocused image is the  $F_0(u, v)$  multiplied with the Fourier transform  $H(u, v)$  of the PSF. Since the PSF is Gaussian in spatial domain, we know its Fourier transform is also Gaussian, which serves to attenuate the higher frequencies in  $F_0(u, v)$ . Therefore, defocus acts as a low-pass filter. It allows the low frequencies in the original image to pass through, but

### Defocus in Fourier Domain

Defocus can be represented as product of Fourier transforms


×

=


$F_0(u, v)$ 
 $H(u, v)$ 
 $F(u, v)$

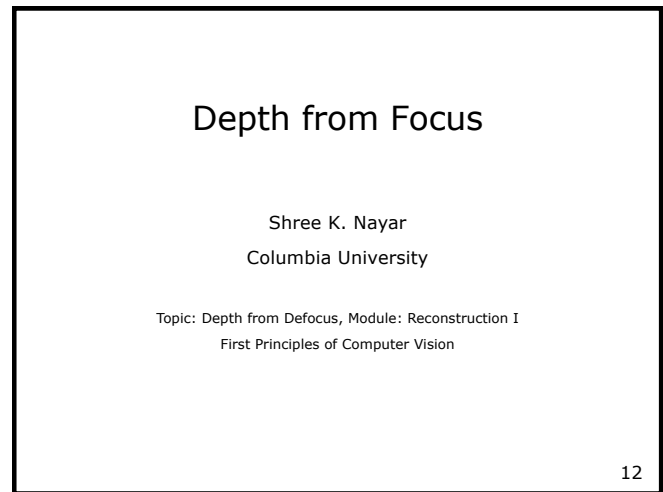
(1D slice of Fourier transform)

Defocus is a Low-Pass Filter

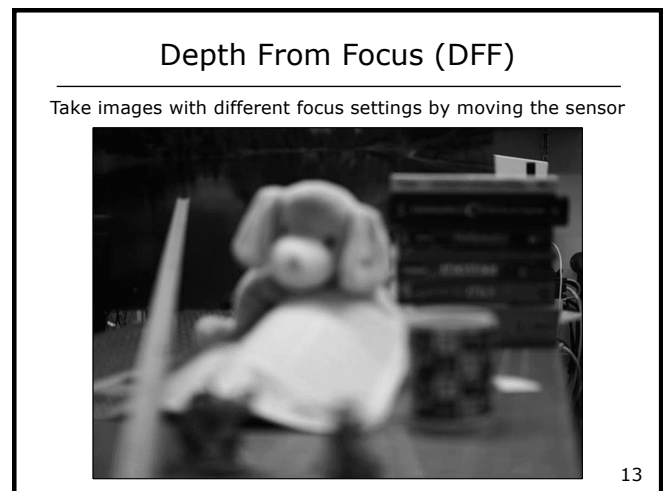
11

it attenuates the high frequencies. Thus, if we wanted to recover depth from defocus, we should be paying attention to the high frequencies in the image.

We now present our first method for recovering depth, called depth from focus.




Here is a scene that has been captured with a camera with a shallow depth of field. The idea behind depth from defocus is to take a series of images with different focus settings, called a focal stack. For each image patch, we find the image in the stack for which the patch is best focused. Since we know the focus setting for that image, we know where the plane of focus is, and hence we can estimate the depth of the corresponding scene patch. There might be some slight magnification changes in the focal stack. Since we know the focus setting for each image, we can correct for such magnification changes. So, our problem is simple—for each image pixel, we need to find where in the focal stack the patch around the pixel is best focused.



Here we see the focal stack for a single image patch. The distance  $s$  between the image sensor and the lens (the focus setting) for each image patch is indicated below it. We see that this patch is best focused for  $s = 51.25$  mm. So, if we have a method for finding the best focused patch, the corresponding sensor distance  $s$  can be plugged into the Gaussian lens law to estimate the distance  $o$  of the corresponding scene patch. For instance, if the focal length  $f$  of the lens is 50 mm, for  $s = 51.25$  mm, we get an object distance  $o = 2.05$  m. One thing worth highlighting here is that a small change in the location of the sensor behind the lens, corresponds to a significant changes in the location of the plane of focus in front of the lens, i.e., in the scene. Therefore, fine control of the optics of an imaging system is sufficient to recover the 3D structure of a large scene.

### Depth From Focus

For each small image patch, find when it is best focused.



$s = 50.95$ 
 $s = 51.10$ 
 $s = 51.25$ 
 $s = 51.40$ 
 $s = 51.55$ 
 $s = 51.70$ 
 $s = 51.85$   
(mm)

Obtain scene depth using gaussian lens law.

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{o} \Rightarrow \boxed{o = \frac{sf}{s-f}}$$

Ex:  $s = 51.25$  mm  
 $f = 50$  mm  
 $o = 2.05$  m

How to find the best focused image? 14

How do we find the best focused image for a patch? We will draw inspiration from image processing. Since defocus is low-pass filtering, it attenuates the high frequencies severely and does not do much to the low frequencies. Hence, we want to measure the amount of high-frequency content within each patch to estimate its degree of focus. One way to do this is to measure how rapidly intensity changes within the patch. We know that intensity changes can be measured using derivatives of the image. In particular, we can use the second derivative. We could use the

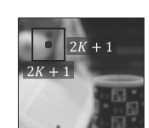
### Focus Measure

Since defocus attenuates high frequencies, use a high-pass filter to measure amount of high frequency content within each small patch.

For example, use:  $\nabla_M^2 f = \left| \frac{\partial^2 f}{\partial x^2} \right| + \left| \frac{\partial^2 f}{\partial y^2} \right|$  (Similar to Laplacian)

Focus Measure: Sum of the square of (modified) Laplacian responses within a small window.

$$M(x, y) = \sum_{i=x-K}^{x+K} \sum_{j=y-K}^{y+K} \nabla_M^2 f(i, j)$$

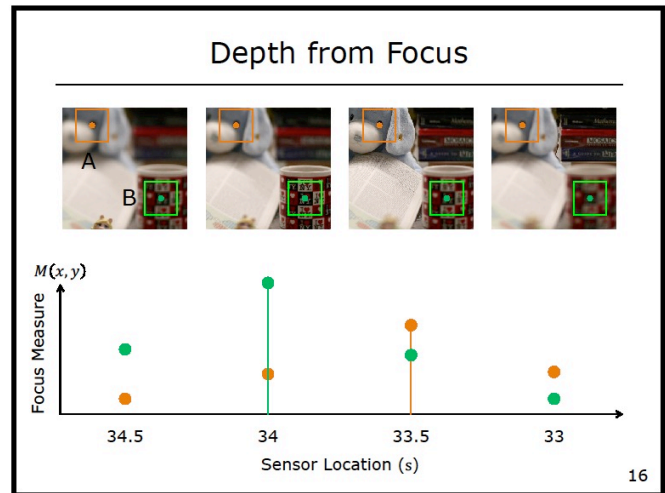


15

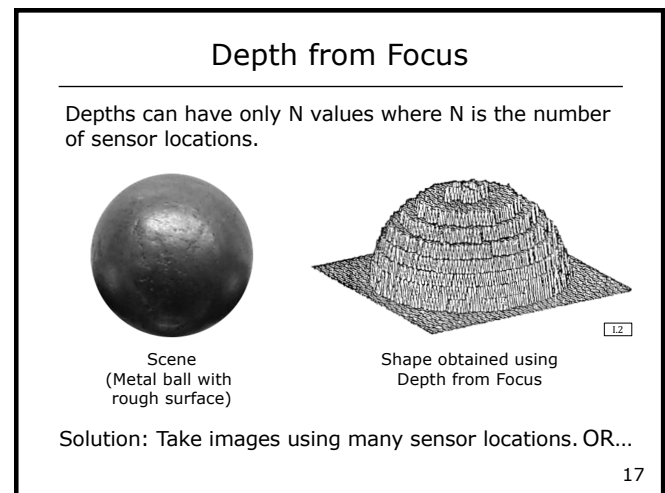
Laplacian, which is the sum of the second derivatives of an image with respect to the  $x$  and  $y$  coordinates. The issue with the Laplacian is that the second derivative in the  $x$  direction might end up canceling the second derivative in the  $y$  direction. To fix this problem, we use the sum of the absolute values of the two derivatives. We call this the modified Laplacian operator,  $\nabla_M^2$ . We apply this operator to each image in the focal stack. The focus measure  $M(x, y)$  at each pixel in an image is computed by summing the modified Laplacian values within a window of size  $(2K + 1, 2K + 1)$  around the pixel. To ensure that our computed scene depth has good spatial resolution, we need to keep the size of this window small, say, 3x3. Note that  $M(x, y)$  will be high if the intensity fluctuations within the window are high.



Let us see how well the modified Laplacian works as a measure of focus. Here is a stack of images, corresponding to different sensor locations,  $s$ . We see that window A produces a maximum focus measure when  $s$  equals 33.5 mm, while window B produces a maximum measure when  $s$  equals 34 mm. These values can be plugged into the Gaussian lens law to compute the depths of the corresponding scene patches.



Let us see how this works for a simple object such as the metal sphere seen here. The sphere is rough and hence has a slight texture. It is this texture that produces high frequencies in the image and allows us to recover depth from focus. On the other hand, if we had a object devoid of texture, then its images would not vary as we change the focus setting. In other words, for each patch, the focus measure will be the same through the entire focal stack. Therefore, for depth from focus to work, the scene must be textured.



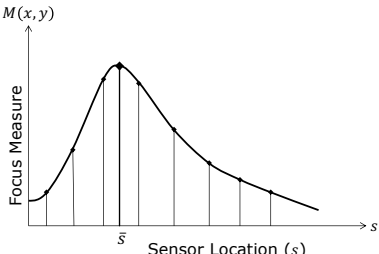
On the right is shown the computed depth map of the sphere. Since we have used a small number of focus settings (a small number of images in our focal stack), the depth map ends up with a small number of discrete depth values. In fact, in this example, the number of steps in the depth map equals the number of images in the focal stack.

The obvious way to get a smoother depth map is to increase the number of captured images. Note that the time needed to capture the images is linear in the number of captured images. In many applications, it is desirable to keep the capture time as small as possible. For instance, if we want to produce depth maps at video rate, we simply cannot afford to capture a large set of images. To address this problem, we use interpolation.

It turns out that, for any textured patch, if we plot the focus measure as a function of the sensor location, we will get a bell-curve shape like the one shown here. Irrespective of the texture, this function has a peak that corresponds to the sensor location  $\bar{s}$  for which the patch is best focused. The value of the peak and the width of the curve would depend on the nature of the texture itself. We therefore model the focus measure function as a Gaussian. Our goal is to fit a Gaussian to the discrete focus measures  $M(s_i)$  and then find the location  $\bar{s}$  of its peak.

### Gaussian Interpolation

Peak of focus measure curve is a Gaussian-like function.

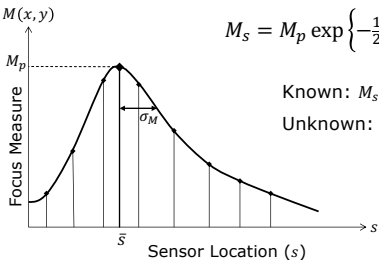


Mean of the Gaussian may be used as the sensor location corresponding to the "best focus."  
[Nayar 1994] 18

Here is the Gaussian model used to model the focus measure function, where  $M_p$  is the peak value,  $\bar{s}$  is the mean value, and  $\sigma_m$  is the standard deviation. Given the discrete measurements  $M_{s_i}$ , we are interested in finding  $M_p$ ,  $\sigma_m$  and  $\bar{s}$ .  $M_p$  and  $\sigma_m$  are related to the appearance of the patch—the amount of high-frequency content in it. More important to us is  $\bar{s}$ , which is the focus setting for which the image patch would be best focused. Note that  $\bar{s}$  is likely to lie in between two of the sensor locations  $s_i$  that were used to capture the focal stack.

### Gaussian Interpolation

Peak of focus measure curve is a Gaussian-like function.



$$M_s = M_p \exp\left\{-\frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_m}\right)^2\right\}$$

Known:  $M_{s_i}, s_i$   
Unknown:  $M_p, \bar{s}, \sigma_m$

How many ( $M_{s_i}, s_i$ ) samples do we need to estimate  $\bar{s}$ ? 19

To fit the Gaussian model to the focus measures  $M_{s_i}$ , we linearize the model by taking the natural log on both sides to get **1**. This equation can be used to solve for  $M_p$ ,  $\sigma_m$  and  $\bar{s}$  using just three discrete focus measures. Since our Gaussian model is likely to most closely approximate the focus measure function near its peak, we use the largest three discrete focus measures ( $M_{s_1}$ ,  $M_{s_2}$ , and  $M_{s_3}$ ) and their corresponding sensor locations ( $s_1$ ,  $s_2$ , and  $s_3$ ), to solve for  $M_p$ ,  $\sigma_m$  and  $\bar{s}$ .

### Gaussian Interpolation

Linearize problem to solve for  $(\bar{s}, M_p, \sigma_m)$ .

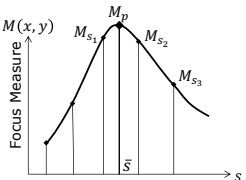
Gaussian:  $M_s = M_p \exp\left\{-\frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_m}\right)^2\right\}$

Taking the natural logarithm:  $\ln M_s = \ln M_p - \frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_m}\right)^2$  **1**

Three sensor positions:

$$\ln M_{s_1} = \ln M_p - \frac{1}{2}\left(\frac{s_1-\bar{s}}{\sigma_m}\right)^2$$

$$\ln M_{s_2} = \ln M_p - \frac{1}{2}\left(\frac{s_2-\bar{s}}{\sigma_m}\right)^2$$

$$\ln M_{s_3} = \ln M_p - \frac{1}{2}\left(\frac{s_3-\bar{s}}{\sigma_m}\right)^2$$


Where,  $M_{s_1}, M_{s_2}, M_{s_3}$  are the three largest values. 20

Shown here is the closed-form solution for  $\bar{s}$ . We can now plug  $\bar{s}$  into the Gaussian lens law to get a more accurate depth estimate.

To summarize our depth from focus method, we take a stack of images corresponding to a discrete set of sensor locations, and apply the focus measure to this stack to get discrete focus measure values. For each image point, we pick the largest three focus measure values, and fit the Gaussian model to them to find the mean value  $\bar{s}$ , which is then used to compute depth.

### Depth Estimation

---

Solving for  $\bar{s}$ :

$$\bar{s} = + \frac{(\ln M_{s_2} - \ln M_{s_3})(s_2^2 - s_1^2)}{2(s_3 - s_2)\{(\ln M_{s_2} - \ln M_{s_1}) + (\ln M_{s_2} - \ln M_{s_3})\}} - \frac{(\ln M_{s_2} - \ln M_{s_1})(s_2^2 - s_3^2)}{2(s_3 - s_2)\{(\ln M_{s_2} - \ln M_{s_1}) + (\ln M_{s_2} - \ln M_{s_3})\}}$$

Obtain scene depth using Gaussian Lens Law:


$$o = \frac{\bar{s}f}{\bar{s} - f}$$

21

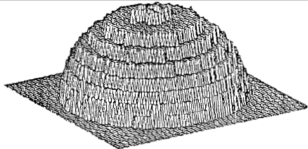
Here is the rough sphere we looked at before. The top-right image is the depth map we previously obtained by simply using the maximum focus measure (without interpolation) to compute depth. The bottom-right image shows the depth map we get using the Gaussian interpolation method described above.

### Depth from Focus: Result


---



Scene  
(Metal ball with rough surface)



Depth without Gaussian Interpolation



Depth using Gaussian Interpolation 22

Here we see a depth from focus system developed for microscopy. The stage of the microscope that holds the sample (object of interest) is motorized 1. The focus setting is varied by moving the sample with respect to the objective lens of the microscope, instead of moving the image sensor. Since the microscope has a very narrow depth of field, even a tiny motion of the stage (a few microns) can change the focus of the sample dramatically. A focal stack of the sample is obtained by capturing images while the stage moves with respect to lens.

### A Depth from Focus System

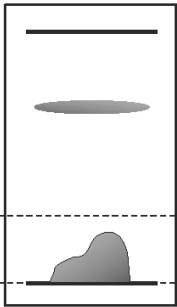
---

Image Sensor

Optics

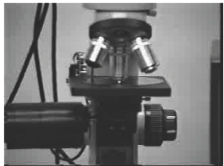
Plane of Focus

Reference Depth  $z = 0$



DFF in a Microscope

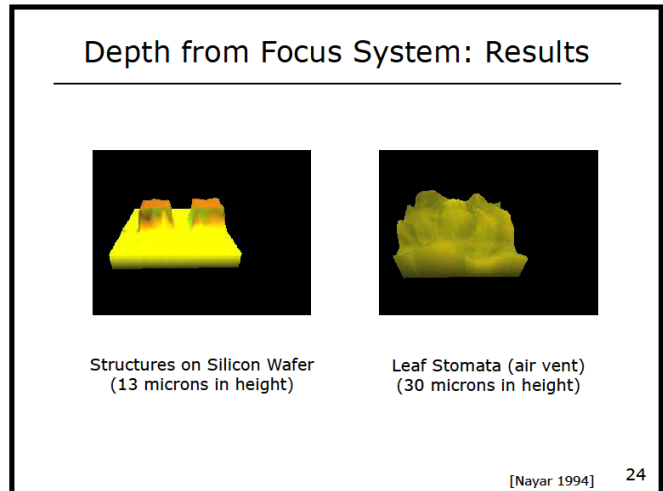
1



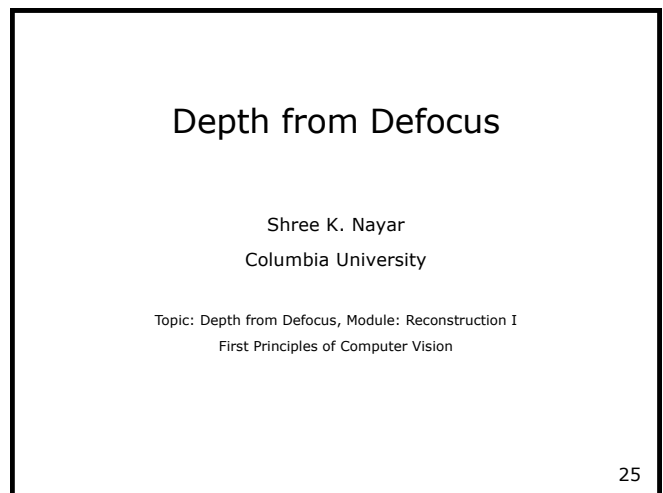
Object on Stage

[Nayar 1994] 23

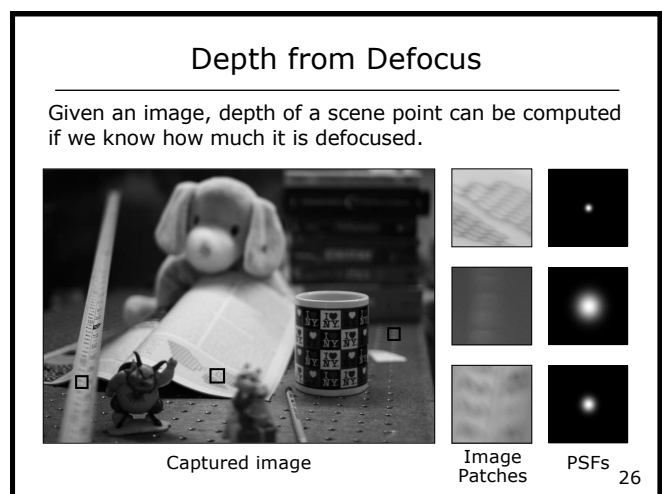
Here are a couple of 3D structures reconstructed using the above microscopic system. On the left is a silicon wafer with two structures that are approximately 13 microns in height, and on the right is a stomata (air vent on a leaf) which is about 30 microns in height. Today, microscopic depth from focus is used for a variety of visual inspection tasks in factory automation.



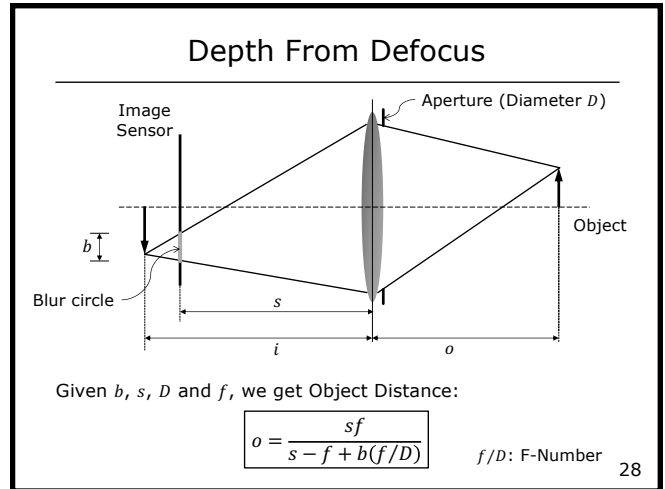
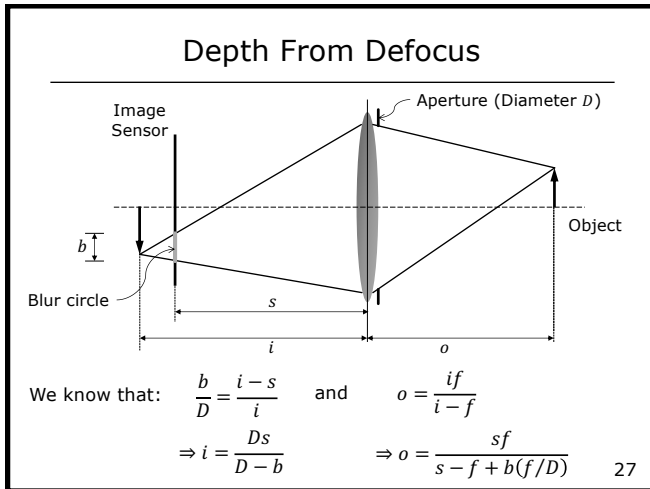
The one disadvantage of depth of focus is that, in order to get high depth accuracy, a large number of images need to be captured. In a typical application, between 10 and 20 images are used, which is too many for some real-time applications. Therefore, we would like to accomplish what depth from focus does, but with a very small number of images. This is the goal of depth from defocus.



Depth from defocus is based on the observation that, instead of determining when a scene patch is in focus, if we can determine how much the patch is blurred in an image, then that blur (PSF) is all we need to estimate depth. Here is an image taken with a camera with a limited depth of field. Shown on the right are three patches and their corresponding PSFs, which can be seen to differ in size. If we can estimate the size of the PSF corresponding to each scene patch, we can find its depth. Unfortunately, there is no way to uniquely determine a patch's blur from a single image.

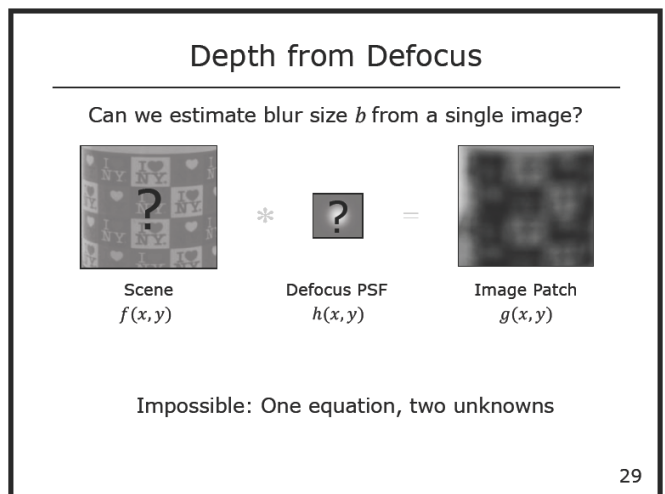


Depth from defocus uses two images of a scene, taken with different focus settings, to estimate depth.



We already know how defocus works. Here is a point and its blur circle in the image. We know the relation between the diameter  $b$  of the blur circle, the diameter  $D$  of aperture, the distance  $i$  between the lens and image plane, and distance  $s$  of the lens from the sensor. If we know the blur circle diameter  $b$ , we can find  $i$ . Then, the Gaussian lens law can be used to find the object distance  $o$ .


Our goal, therefore, is to find the diameter of the blur circle. We know that the focused image patch  $f$  on the left, convolved with an unknown PSF  $h$ , gives us the captured blurred image patch  $g$  on the right. So, we have one equation but two unknowns—the focused image patch  $f$  and the width  $\sigma$  of the PSF.



What if we capture two images with different focus settings? The focused image  $f$  convolved with the first PSF of width  $\sigma_1$  gives us  $g_1$ . The same focused image, convolved with a second PSF of width  $\sigma_2$ , gives us  $g_2$ . Now we have two equations, but three unknowns— $f$ ,  $\sigma_1$ , and  $\sigma_2$ . Irrespective of how many images we capture under different focus settings, the number of equations always falls short of the number unknowns by one. However, we do have additional information regarding our imaging system that we can use to overcome this shortfall.


### Depth from Defocus

What if we have two images with different defocus?




$f(x,y)$

\*




$h_{\sigma_1}(x,y)$

=




$g_1(x,y)$




$f(x,y)$

\*



$h_{\sigma_2}(x,y)$

=



$g_2(x,y)$


Two equations, three unknowns 30

Assume that, instead of changing the sensor position, we change the aperture diameter to capture two images of the scene. Assume the aperture diameter in the first case is  $D_1$ , which produces a PSF of width  $\sigma_1$ , and in the second case is  $D_2$ , which produces a PSF of width  $\sigma_2$ . We do not know  $\sigma_1$  or  $\sigma_2$  because they depend on the depth of the scene patch. However, from the two equations here for the blur circle diameters  $b_1$  and  $b_2$ , we see that the ratio of  $\sigma_1$  and  $\sigma_2$  is equal to the ratio of  $D_1$  and  $D_2$ . Since we have full control over the imaging system, we know  $D_1$  and  $D_2$ .

### The Third Equation


If the two images were taken with two known apertures, then their blur sizes are related.

$d_1$



PSF ( $\sigma_1$ )

$d_2$



PSF ( $\sigma_2$ )

Blur circles:

$$b_1 = D_1 \left| 1 - \frac{s}{i} \right|$$

$$\sigma_1 = b_1/2$$

$$b_2 = D_2 \left| 1 - \frac{s}{i} \right|$$

$$\sigma_2 = b_2/2$$

$$\frac{\sigma_1}{\sigma_2} = \frac{D_1}{D_2}$$

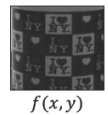
31

Now we see that, when we take two images using different (known) aperture settings, we have three equations and three unknowns— $f$ ,  $\sigma_1$ , and  $\sigma_2$ . It is convenient to look at these three equations in frequency domain. Since we know that convolution in the spatial domain is equivalent to multiplication in the Fourier domain, we get the two equations at the bottom for  $G_1$  and  $G_2$ . Our third equation related to the ratio of  $\sigma_1$  and  $\sigma_2$  remains the same.


### Depth from Defocus

For each image patch we have:


Three unknowns



$f(x,y)$



$\sigma_1$



$\sigma_2$

Three equations

$$g_1(x,y) = f(x,y) * h_{\sigma_1}(x,y)$$

$$g_2(x,y) = f(x,y) * h_{\sigma_2}(x,y)$$

$$\sigma_1/\sigma_2 = D_1/D_2$$

In Fourier Domain

$$G_1(u,v) = F(u,v) \times H_{\sigma_1}(u,v)$$

$$G_2(u,v) = F(u,v) \times H_{\sigma_2}(u,v)$$

$$\sigma_1/\sigma_2 = D_1/D_2$$

32

We can simply take the ratio of  $G_1$  to  $G_2$  to get expression [1]. Interestingly,  $F(u, v)$ , which is the Fourier transform of the focused image  $F(x, y)$ , gets cancelled out. Therefore, we get the ratio of the Fourier transforms of the first PSF and the second PSF. We will assume the PSF to be modeled as a Gaussian. We know that the Fourier transform of a Gaussian is a Gaussian. So,  $G_1$  by  $G_2$  is a ratio of two Gaussians with standard deviations  $\sigma_1$  and  $\sigma_2$ .

Now, we can take the natural log on both sides and get equation [2], which has two unknowns,  $\sigma_1$

and  $\sigma_2$ . Remember that we also have our third constraint, which relates the  $\sigma_1$  and  $\sigma_2$  to the ratio of the aperture diameters  $D_1$  and  $D_2$ . We can therefore solve for  $\sigma_1$  and  $\sigma_2$  using any one frequency  $(u, v)$  as long as the values of  $G_1$  and  $G_2$  are non-zero for that frequency. Then, we can use either  $\sigma_1$  or  $\sigma_2$  to find the corresponding diameter of the blur circle, and use it to find the depth of the scene point.

The problem with the above approach is that we need to use a high frequency (large values for  $u$  and  $v$ ) because it is the high frequencies that are most sensitive to defocus. However, high frequencies are also most impacted by image noise. As a result, the above approach to depth from defocus tends to be sensitive to noise.

### A Naïve DFD Algorithm

---

Cancel out  $F(u, v)$ :

$$\boxed{1} \quad \frac{G_1(u, v)}{G_2(u, v)} = \frac{F(u, v) \times H_{\sigma_1}(u, v)}{F(u, v) \times H_{\sigma_2}(u, v)} = \frac{H_{\sigma_1}(u, v)}{H_{\sigma_2}(u, v)}$$

Substitute for  $H_{\sigma_1}(u, v)$  and  $H_{\sigma_2}(u, v)$ :

$$\frac{G_1(u, v)}{G_2(u, v)} = \frac{\exp(-2\pi^2(u^2 + v^2)\sigma_1^2)}{\exp(-2\pi^2(u^2 + v^2)\sigma_2^2)}$$

Taking the natural logarithm on both sides:

$$\boxed{2} \quad \sigma_1^2 - \sigma_2^2 = \frac{\ln G_2(u, v) - \ln G_1(u, v)}{2\pi^2(u^2 + v^2)} \rightarrow \begin{matrix} \text{Sensitive to noise} \\ \text{making solution} \\ \text{unstable} \end{matrix}$$

$$\sigma_1/\sigma_2 = D_1/D_2$$

Solve the above to get  $\sigma_1$  and  $\sigma_2$ . Use either one to obtain object distance (depth).

[Pentland 1987] 33

Let us look at a more direct approach to estimating depth, called reconstruction-based depth from defocus. Let us assume for a minute that that our unknowns —  $\sigma_1, \sigma_2$ , and the focused image  $f$  — are all known. Then, we know that  $g_1$  and  $g_2$  can be reconstructed by convolving  $f$  with the PSFs corresponding to  $\sigma_1$  and  $\sigma_2$ , respectively. So, we formulate the reconstruction error  $E$  in [1] as the sum of the square of the error in the reconstruction of  $g_1$  and the square of the error in the reconstruction of  $g_2$ . We also have our additional constraint [2] from before, which we use to express  $\sigma_2$  in terms of  $\sigma_1$  in the reconstruction error [3]. Now, we want to find the  $f$  and  $\sigma_1$  that minimize the reconstruction error  $E$ .

### Reconstruction-Based Depth from Defocus

---

Find  $(\sigma_1, \sigma_2, f(x, y))$  that minimizes Reconstruction Error:

$$E(\sigma_1, \sigma_2, f) = \|g_1 - (h_{\sigma_1} * f)\|^2 + \|g_2 - (h_{\sigma_2} * f)\|^2 \quad \boxed{1}$$

We know that  $\sigma_2 = \sigma_1 D_2/D_1$  [2]

Rewrite  $E$  as a 2-variable function:

$$E(\sigma_1, f) = \|g_1 - (h_{\sigma_1} * f)\|^2 + \|g_2 - (h_{(\sigma_1 D_2/D_1)} * f)\|^2 \quad \boxed{3}$$

[Favaro 2003] 34

To find the  $f$  and  $\sigma_1$  that minimize  $E$ , we take the derivative of  $E$  with respect to  $\sigma_1$  and with respect to  $f$ , and set each of them to zero. Optimization is then used to solve for  $f$  and  $\sigma_1$ . As before, we can find  $b_1$  from  $\sigma_1$ , and use it to find the depth  $o$  of the scene patch. This is repeated for all scene patches to get the 3D structure of the scene.

To summarize, depth from focus enables us to take just two differently focused images of a scene, using two known apertures, and compute the 3D structure of the scene.

### Computing Depth From Defocus

---

Find  $(\sigma_1, f)$  using:  $\frac{\partial E}{\partial \sigma_1} = 0$  and  $\frac{\partial E}{\partial f} = 0$

Using  $\sigma_1$  compute size of blur circle:  $b_1 = 2\sigma_1$

Object distance (depth):

$$o = \frac{s_1 f}{s_1 - f + b_1(f/D)}$$

35

For depth from defocus, we changed the focus setting by changing the diameter of the aperture. Alternatively, we can use the method we used for depth from focus, which is change the location of the image sensor. Both approaches are shown here. In the case of changing the position of image sensor, the equations used to compute depth from defocus will be need to be modified, but the underlying principle remains the same—using two differently blurred images to compute depth.

### Capturing Defocused Images

---

Method 1: Change Aperture

Method 2: Move Sensor

36

In the example shown here, we have two images captured by moving the sensor location. In the far-focused image, objects closer to the camera are more blurred, while in the near-focused image the farther objects are more blurred. On the right is the 3D structure of the scene computed from these two images using the reconstruction-based method we described in slide 34.

### Depth from Defocus: Result

---

Far Focused

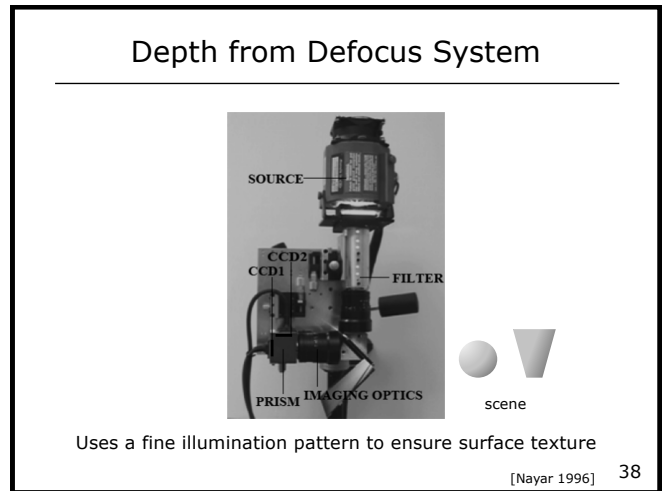
Near Focused

Estimated 3D shape

37

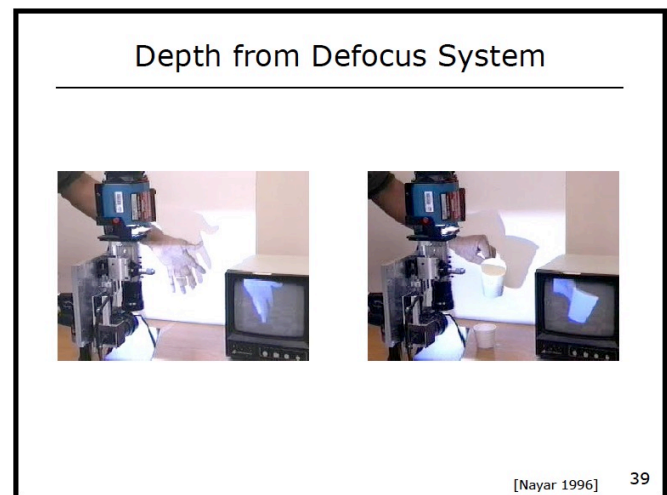


Here is an example of a real-time (video-rate) depth from defocus system. The lens (labeled as imaging optics) captures an image of the scene that is split into two identical images using a beam splitter. The two images are simultaneously captured using two image sensors (labeled CCD1 and CCD2), where the effective distances of the two sensors from the lens are offset with respect to each other by a small distance. Due to the offset, the two sensors simultaneously capture two differently focused images of the scene, which are used to perform depth from defocus. Due to the simultaneous image capture, the system is able to produce a video (30 frames per second) of the scene.



Note that if a surface is texture-less, its defocused image is identical to its focus image. Therefore, depth from focus, or defocus, only works for scene patches that are textured. To overcome this limitation, the above system projects a very fine illumination pattern on the scene to ensure that it is textured everywhere. The pattern is generated by placing a textured optical mask (filter) between a light source and the scene.

Here we see the above real-time depth from defocus system being used to recover the structures of dynamic scenes—a moving hand on the left, and milk being poured out of a cup on the right. The computed depth maps are displayed on the small monitor on the right, where the brightness of each scene point is inversely proportional to its depth.



## References and Credits

Shree K. Nayar  
Columbia University

Topic: Depth from Defocus, Module: Reconstruction I  
First Principles of Computer Vision

40

## References: Papers

[Favaro 2003] P. Favaro, A. Mennucci and S. Soatto, "Observing shape from defocused images". IJCV, 2003.

[Nayar 1994] S. K. Nayar and Y. Nakagawa, "Shape from Focus," PAMI, 1994.

[Nayar 1996] S. K. Nayar, M. Watanabe, and M. Noguchi, "Real-time focus range sensor". PAMI, 1996.

[Pentland 1987] A. Pentland, "A New Sense for Depth of Field". PAMI, 1987.

[Subbarao 1994] M. Subbarao and G. Surya, "Depth from defocus: A spatial domain approach". IJCV, 1994

41

## Image Credits

- I.1 [https://flickr.com/photos/tim\\_ellis/75690428/](https://flickr.com/photos/tim_ellis/75690428/) Tim Ellis. Licensed under CC BY-NC 2.0.
- I.2 Paolo Favaro. Used with permission.

42

---

**Acknowledgements:** Thanks to Roshan Kenia, Kevin Chen, Nikhil Nanda and Ayush Sharma for their help with transcription, editing and proofreading.

## References

[Favaro 2003] P. Favaro, A. Mennucci and S. Soatto, Observing shape from defocused images, IJCV, 2003.

[Nayar 1994] S. K. Nayar and Y. Nakagawa, Shape from Focus, IEEE PAMI, 1994.

[Nayar 1996] S. K. Nayar, M. Watanabe and M. Noguchi, Real-time focus range sensor, IEEE PAMI, 1996.

[Pentland 1987] A. Pentland, A New Sense for Depth of Field, IEEE PAMI, 1987.

[Subbarao 1994] M. Subbarao and G. Surya, Depth from defocus: A spatial domain approach, IJCV, 1994.