

Laboratory 1 Part 2: Introduction to Imaging

Objectives

- Make your imaging system more quantitative - Learn the focal length of your imaging system - Redesign your microscope for a specific magnification

Introduction

Scientific imaging typically requires quantitative measures. "Images are numbers" and those numbers are often significant to scientific questions. Often image values obtained by a camera are expressed in Arbitrary Digital Units (ADU), which are a function of the input light, detector sensitivity and conversion factors, and analog-to-digital conversion. You have already seen that you have a 10-bit camera whose values attain a maximum of 1023. However, scientifically, these values tend not to have a lot of absolute meaning without additional calibration. Similarly, the size and sampling of your image are dependent on how you have set up your optical system. In the following portions of the lab, you will calibrate ADU to have a scientific meaning (transmissivity) as well as establish the size and sampling of your image (so that you can label the actual sizes of features in your image).

Make your pixel values quantitative

Use the nominal microscope set up from Part 1 of the lab (including aperture wide open). Place a slide in the system and ensure that exposure and illumination settings are good. Use one of the "no filter" settings on the filter wheel (we won't use filters for the remainder of this lab). Similarly, focus the image and call this image \$Y\$.

```
In [1]: # Import and instantiate all classes
from lighting import Lighting
from pololu import Pololu
from camera import Camera
l = Lighting()
m = Pololu()
c = Camera()
l.set_intensity(10)
c.open()
f_wheel, stage_y, stage_x, ap, stage_z = 1, 2, 3, 4, 5
m.set_position(ap, 2496, blocking = True) # wide aperture
```

```
Light control initialized successfully.
Servo control initialized successfully.
Load uc480 library..
ThorCam opened successfully.
```

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
```

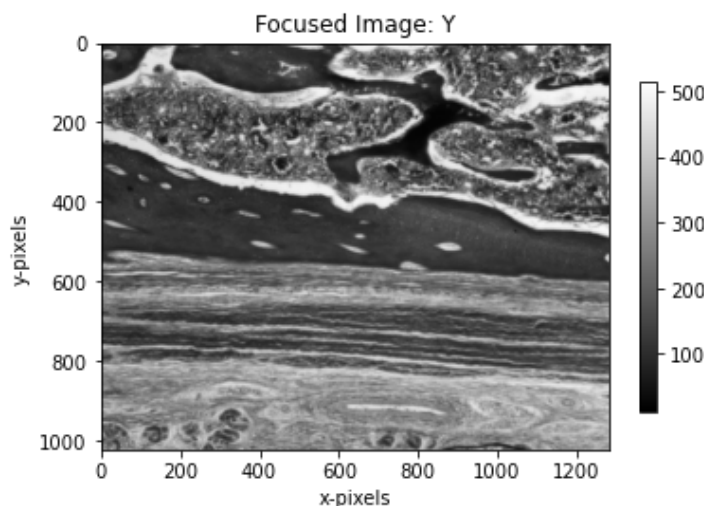
```
from IPython.display import display, clear_output
```

In [37]:

```
m.set_position(f_wheel, 1540, blocking = True) # no filter
m.set_position(stage_y, 500, blocking = True)
m.set_position(stage_x, 1000, blocking = True)
m.set_position(stage_z, 700, blocking = True)
c.set_exposure(10) # good exposure

Y = c.capture().copy()
x = plt.imshow(Y, cmap = "gray")
plt.title("Focused Image: Y")
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.colorbar(x, shrink=0.8)
```

Out[37]: <matplotlib.colorbar.Colorbar at 0x1b493eff910>



We would like to make the pixel values quantitative. For example, pixel values should denote the transmissivity of the sample (which can range from 0-100% transmission of the incident light). To obtain such quantitation, the system must be calibrated – e.g. what measurement values correspond to what transmission. To perform this calibration, we conduct two specialized image captures:

- “Dark” images – Turn off your LED illuminator and close the aperture all the way to prevent light from reaching the camera. Collect another set of 30 images. Average these and call this image \$D\$.
- “Air-only” images – Move away from the slide (as shown in the picture below) and collect a series of 30 images with nothing in the system. Find the average of these 30 images (to reduce noise). Call this image \$A\$.



Data collection: \$A\$, \$D\$, and \$Y\$

1. Mathematically (using \$A\$, \$D\$, and \$Y\$), what is the expression to obtain quantitative transmission data?

Write the expression and its derivation below.

In order to calculate the image transmission data we first need to calibrate the scale for what it

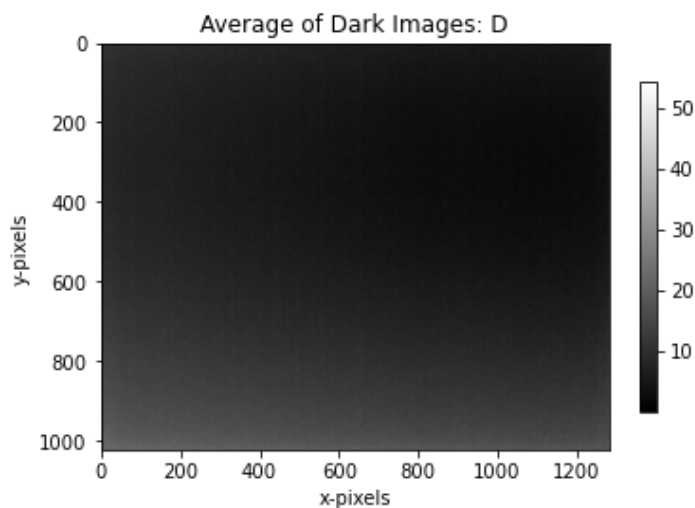
means to have 0% transmission and 100% transmission. We will assume that image D represents 0% and image A represents 100%. In order to obtain the percent transmission, we would divide image Y pixel values by the difference between the pixel values of image A and D so that we are dividing by the total possible range of pixel values for a given pixel. Therefore, we can get percent transmission by getting some image $T = (Y - D) / (A - D)$ for each pixel in T

In order to obtain quantitative transmission data

```
In [20]: l.set_intensity(0)
m.set_position(ap, 992, blocking = True)
dark_imgs = []
for i in range(30):
    dark_imgs.append(c.capture().copy())

D = np.stack(dark_imgs)
D = np.mean(D, axis = 0) # mean of dark images
x = plt.imshow(D, cmap = "gray")
plt.title("Average of Dark Images: D");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
```

Out[20]: <matplotlib.colorbar.Colorbar at 0x1b4915a0ac0>

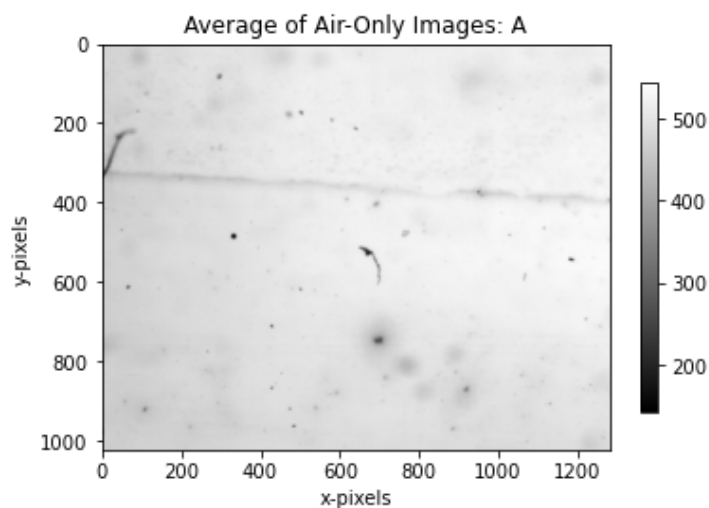


```
In [23]: l.set_intensity(10)
m.set_position(ap, 2496, blocking = True)
m.set_position(stage_y, 1000, blocking = True)
m.set_position(stage_x, 1000, blocking = True)

air_imgs = []
for i in range(30):
    air_imgs.append(c.capture().copy())

A = np.stack(air_imgs)
A = np.mean(A, axis = 0) # mean of dark images
x = plt.imshow(A, cmap = "gray")
plt.title("Average of Air-Only Images: A");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
```

Out[23]: <matplotlib.colorbar.Colorbar at 0x1b490f84640>

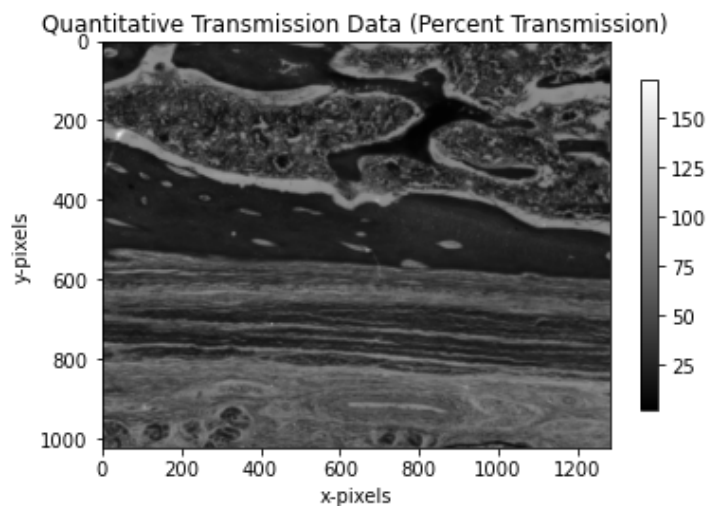


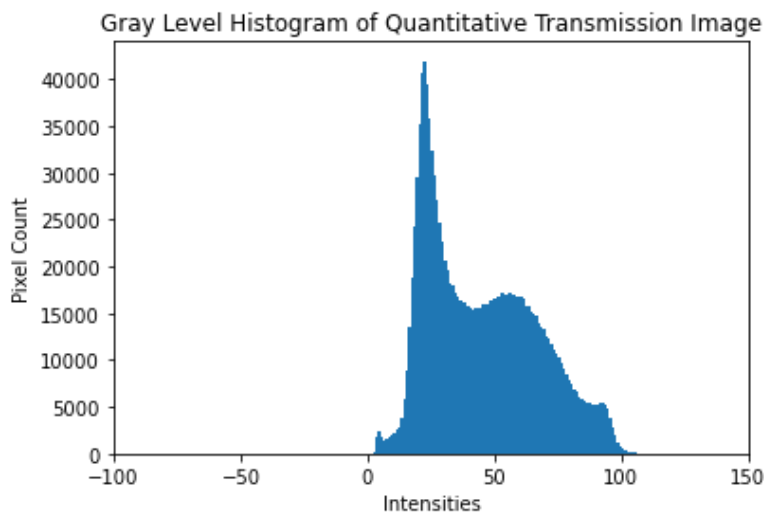
2. Create an image of your slide with quantitative transmission values and a colorbar, and show it below:

```
In [42]: qt = ((Y-D)/(A-D)) * 100
x = plt.imshow(qt, cmap = "gray")
plt.title("Quantitative Transmission Data (Percent Transmission)");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)

plt.figure()
plt.hist(qt.ravel(),1024,[0,1024]);
plt.ylabel("Pixel Count");
plt.xlabel("Intensities");
plt.title("Gray Level Histogram of Quantitative Transmission Image");
plt.xlim(-100,150)
```

Out[42]: (-100.0, 150.0)





In []:

Find the magnification of your microscope (Make your axes quantitative)

The actual pixel size of your DCC3240M camera is 5.3 μm . However, there is magnification between the object you are imaging and the image that falls on the sensor. To find the magnification, you need to measure the size of a known target in pixels.

We will be using the resolution target on your microscope. Move the target so that you can see the 100 μm grid (pink circle below, 100 μm is the distance per grid), and make sure it is in focus.



Use Jupyter interactive plotting function (refer to Lab 0), and estimate the size of a pixel in terms of the actual slide feature size in the object plane. You can read the coordinates of the pixel the cursor is hovering over at the bottom of the plot.

Counting pixels over several squares should improve your estimate.

In-class data collection: Image the 100 μm grid

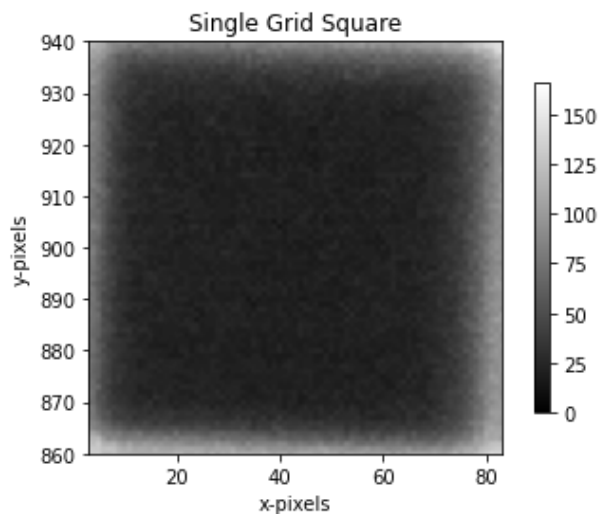
1. Show a zoomed image of the 100 μm grid and the distance you used to compute the pixel size:

In [27]:

```
# computer: imaging-8
c.set_exposure(10)
m.set_position(stage_y, 1500, blocking = True)
m.set_position(stage_x, 1030, blocking = True)
m.set_position(stage_z, 1700, blocking = True)
x = plt.imshow(c.capture().copy(), cmap = "gray")
plt.title("Single Grid Square");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
```

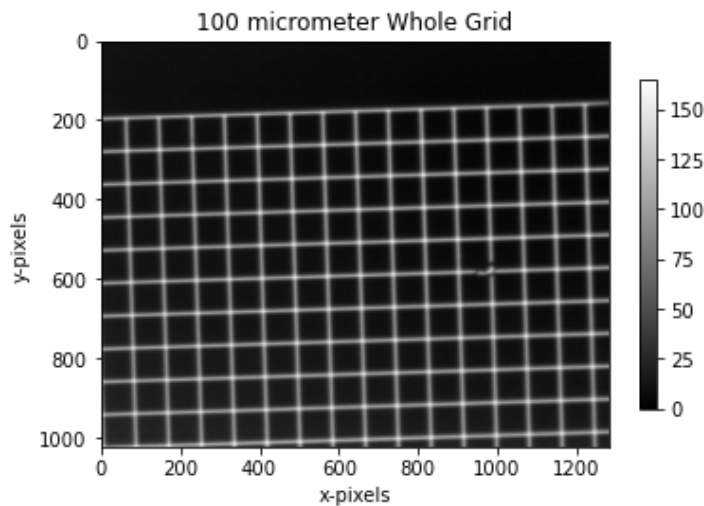
```
plt.colorbar(x, shrink=0.8)
plt.xlim(3,83) # x-values for one square
plt.ylim(860,940) # y-values for one square
```

Out[27]: (860.0, 940.0)



```
In [29]: x = plt.imshow(c.capture().copy(), cmap = "gray")
plt.title("100 micrometer Whole Grid");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
```

Out[29]: <matplotlib.colorbar.Colorbar at 0x271e73e64f0>



The grid shown above is the 100 micrometer grid, the first image is the zoomed-in image of a single box in the grid. The stage positions for this grid on imaging lab 8 are: stage_x = 1030, stage_y = 1500, and stage_z = 1700. The length of each side of the square is 100 micrometers so by using the graph and the zoomed in image we determined that 80 pixels is 100 micrometers.

2. What is the size of one pixel in the object plane?

1 pixel = 100 micrometers / 80 pixels = 1.25 micrometers.

3. Magnification is the ratio of the size of the detected image to the size of the real object. What is the magnification of your system?

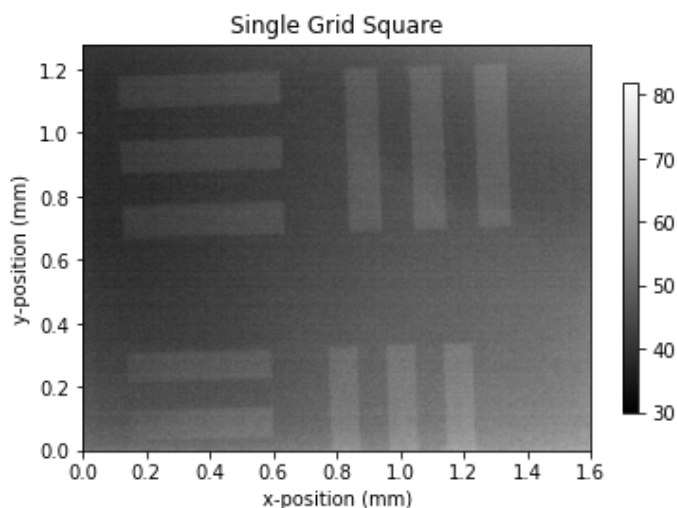
1 pixel = 5.3 micrometers (system without magnification) 1 pixel = 1.25 micrometers (system with magnification) magnification: $5.3 / 1.25 = 4.24$

4. Take an image of the line pairs region (green circle above) in the resolution target and paste it below with a color bar and proper axis labels (mm).

```
In [12]: # computer: imaging-8
c.set_exposure(10)
m.set_position(stage_y, 1640, blocking = True)
m.set_position(stage_x, 810, blocking = True)
m.set_position(stage_z, 1700, blocking = True)
m.set_position(f_wheel, 950, blocking = True) # no filter

img = c.capture().copy()
x = plt.imshow(img, cmap = "gray", extent = [0, 1.25/1000 * img.shape[1], 0, 1.25/1000 *
plt.title("Single Grid Square");
plt.xlabel("x-position (mm)");
plt.ylabel("y-position (mm)");
plt.colorbar(x, shrink=0.8)
```

Out[12]: <matplotlib.colorbar.Colorbar at 0x236014ad5b0>



5. What are the finest line pairs that you can resolve? Show a zoomed image including the line pairs above and below this limit. (The following table will help you find the lp/mm for each target. Values are in lp/mm. The element number is labeled on the left of each line pair target; the group number is on top.)

In []:



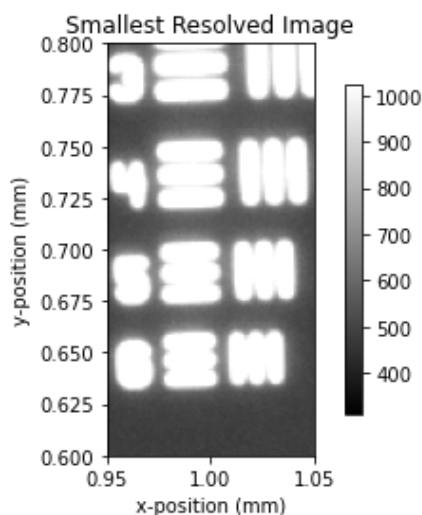
```
In [47]: # computer: imaging-8
c.set_exposure(100)
```

```

l.set_intensity(10)
m.set_position(ap, 2496, blocking = True)
m.set_position(stage_y, 1670, blocking = True)
m.set_position(stage_x, 730, blocking = True)
m.set_position(stage_z, 1500, blocking = True)
m.set_position(f_wheel, 1050, blocking = True)
img = c.capture().copy()
x = plt.imshow(img, cmap = "gray", extent = [0,1.25/1000 * img.shape[1], 0, 1.25/1000 *
plt.title("Smallest Resolved Image");
plt.xlabel("x-position (mm)");
plt.ylabel("y-position (mm)");
plt.colorbar(x, shrink=0.8)
plt.xlim(0.95, 1.05)
plt.ylim(0.6, .8)

```

Out[47]: (0.6, 0.8)

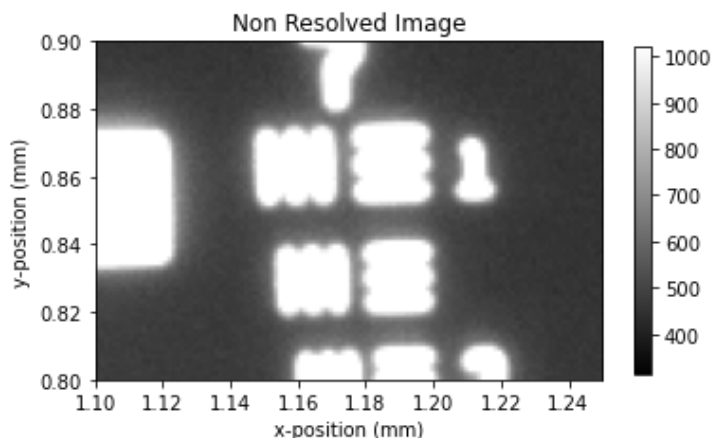


```

In [48]: # computer: imaging-8
c.set_exposure(100)
l.set_intensity(10)
m.set_position(ap, 2496, blocking = True)
m.set_position(stage_y, 1670, blocking = True)
m.set_position(stage_x, 730, blocking = True)
m.set_position(stage_z, 1500, blocking = True)
m.set_position(f_wheel, 1050, blocking = True)
img = c.capture().copy()
x = plt.imshow(img, cmap = "gray", extent = [0,1.25/1000 * img.shape[1], 0, 1.25/1000 *
plt.title("Non Resolved Image");
plt.xlabel("x-position (mm)");
plt.ylabel("y-position (mm)");
plt.colorbar(x, shrink=0.8)
plt.xlim(1.1, 1.25)
plt.ylim(0.8, 0.9)

```

Out[48]: (0.8, 0.9)



The best resolution we could get is 114 line pairs per millimeter. The pair of lines we are specifically referring to is the pair in the above image (Smallest Resolved Image) that occupies the space approximately at $x = [0.95, 1.05]$, $y = [0.625, 0.650]$. We are unable to resolve the 128 line pairs per millimeter (the next smallest pair of lines after 114 line pairs per millimeter) as shown in the above image (Non Resolved Image).

In []:

Devise a strategy to measure the focal length of your lens

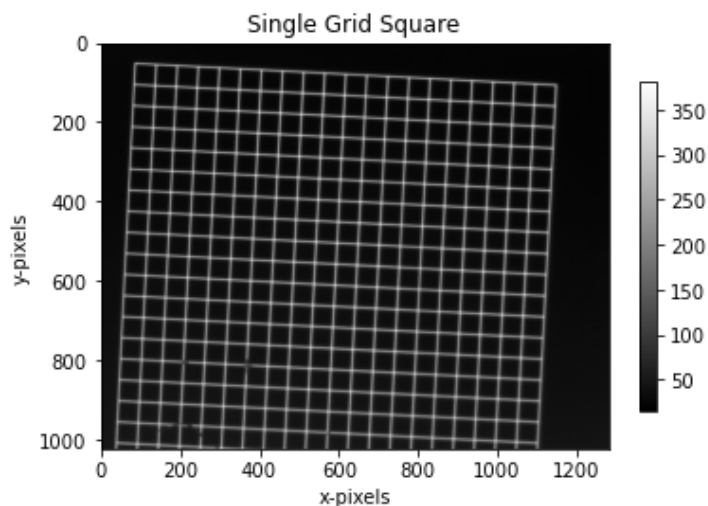
In lecture you learned about the lens equation, which permits you to compute the location of a focused image given the position of the object and the focal length of the lens. Thus, to design a system with a specific magnification, you need to know the focal length of the lens. This is straightforward if you can measure the distance between the lens and object and image; however, this is not necessarily easy when these planes are hidden within optical elements. Thus, we explore an alternate approach. Restating the lens equation, we know: $\frac{1}{a} + \frac{1}{b} = \frac{1}{f}$. Consider the case where you measure the magnification in two focused optical systems but change the distance to the camera by a known amount (e.g. adding an optical tube of known length, z). The lens equation holds for both systems, but a and b will both change. Thus, you have: $\frac{1}{a_1} + \frac{1}{b_1} = \frac{1}{f}$ and $\frac{1}{a_1+z} + \frac{1}{b_2} = \frac{1}{f}$. You already know how to measure magnification in each system and that magnification is related to object and image planes through the following equations: $M_1 = \frac{a_1}{b_1}$ and $M_2 = \frac{a_1+z}{b_2}$. This means you have four equations and four unknowns (a_1 , b_1 , b_2 , f) and two measured values (M_1 , M_2) and one known value (z).

In [12]:

```
# computer: imaging-5
c.set_exposure(10)
m.set_position(stage_x, 1700, blocking= True)
m.set_position(stage_y, 2000, blocking = True)
m.set_position(stage_z, 1000, blocking = True)
x = plt.imshow(c.capture().copy(), cmap = "gray")
```

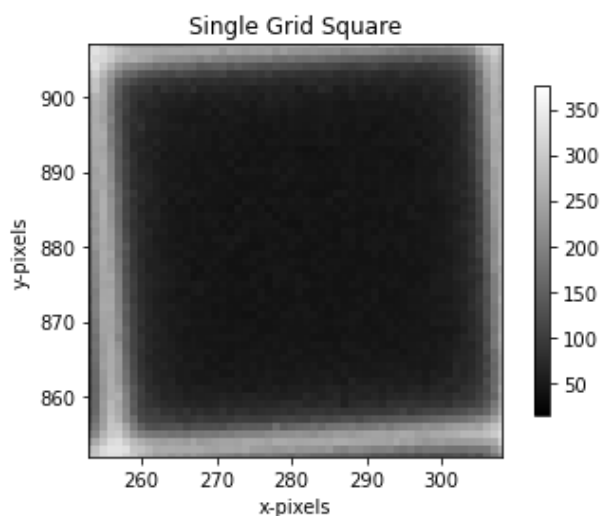
```
plt.title("Single Grid Square");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
```

Out[12]: <matplotlib.colorbar.Colorbar at 0x1bb79218cd0>



```
In [27]: # computer: imaging-5
x = plt.imshow(c.capture().copy(), cmap = "gray")
plt.title("Single Grid Square");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
plt.xlim(253, 308) ## 55 pixels x direction
plt.ylim(852, 907) ## 55 pixels y direction
```

Out[27]: (852.0, 907.0)



We get that 100 micrometers / 55 pixels = 1.83 micrometers per pixel. 1 pixel = 5.3 micrometers (system without magnification)

1 pixel = 1.83 micrometers (system with magnification) magnification: $5.3 / 1.83 = 2.92$

1.
D

$\begin{equation}$
 $\frac{1}{}$

2.
What

Build a

1. Derive and state the mathematical relationship between f and (M_1, M_2, z) . Show a few intermediate steps.

$$\begin{aligned}\frac{1}{a_1} + \frac{1}{b_1} &= \frac{1}{f} \\ \frac{a_1}{b_1} &= \frac{a_1}{f} - 1 \\ \frac{1}{a_1 + z} + \frac{1}{b_2} &= \frac{1}{f} \\ \frac{a_1 + z}{b_2} &= \frac{a_1 + z}{f} - 1 \\ M_1 &= \frac{a_1}{b_1} = \frac{a_1}{f} - 1 \\ M_2 &= \frac{a_1 + z}{b_2} = \frac{a_1 + z}{f} - 1 \\ f &= \frac{f(M_1 + 1) + z}{M_2 + 1} \\ f &= \frac{z}{M_2 - M_1}\end{aligned}$$

2. What is the focal length of your lens in mm? _____

Let $M_1 = 4.24$, $M_2 = 2.92$, and $z = 4.5 - 6.0 = -1.5'' = -38.1 \text{ mm}$

$$\begin{aligned}f &= \frac{-38.1}{2.92 - 4.24} \\ f &= 28.86 \text{ mm}\end{aligned}$$

4. Build a system to achieve a specific magnification goal

1. From the nominal microscope design that uses only the ½", 1", and 2" tubes, how much additional tube length is required to obtain a system with a magnification of 2.4x?

$$\begin{aligned}z &= f * (M_2 - M_1) \\z &= 28.86 * (2.4 - 2.92) \\z &= -15.01mm = -0.59in\end{aligned}$$

In order to get a magnification of 2.4X, we would need to decrease the total optical tube length by 0.59 in (about 0.6 in) from the original tube length of 4.5 in to 3.9 in.

2. How would you achieve this using available tubes? (The available tube lengths are: 2", 1", 0.5", 0.3").

Record the tube lengths you have designed, and we will build this system in class next time.

We are aiming to create an optical tube with a total length of approximately 3.9 inches. We can use one 2" tube, one 1" tube, and 3 0.3" tubes for a total of

$$length = 1(2) + 1(1) + 3(0.3) = 3.9in$$