Laboratory 1 Part 1: Introduction to Imaging

Objectives

- Learn how to build a basic microscope - Familiarize with stage and camera control (Exposure, focusing, lighting control) - Acquire an image of a biological slide

Introduction / Basic Rules

The first part of this laboratory is a tutorial to get familiar with your optical kit. The following is a step-by-step process describing how to assemble a basic microscope.
A number of basic rules apply to the handling of this kit. These rules include: - Any glass element in the kit should be handled carefully by the edges. Never place fingers on a surface through which light for imaging passes. This includes imaging targets and slides. - Dust and oils are bad for most optical elements and the camera. Avoid having the interior of the objective lens and the uncovered camera exposed for any length of time. Never touch the camera sensor. - Nothing in your kit should require a large amount of force to assemble or disassemble. If something isn't going together easily, there's probably something wrong. Never force any elements that aren't moving easily. If there are problems, ask the instructor or TA for help. - Please be a good custodian of your optical hardware kit. The kit is your responsibility for the duration of the course. With proper care these kits should be able to be used for many years. Thank you for your help in making this possible. - If any element of your kit appears damaged or if something doesn't appear right, contact the instructor or TA immediately.

Microscope assembly

We will begin this lab by assembling a basic microscope. You are provided with all of the hardware elements that are required to construct this imaging device. The figure below shows a complete inventory of all of the hardware elements in your optical kit. Take a moment to ensure all of the parts are present in your kit and to learn some of the nomenclature for the various elements.

In addition to the parts in the picture below, you will have access to a computer, a power supply for the lighting module, and a number of imaging targets.

Step 1: Construct the linear stage assembly.
First, we begin assembly of the imaging platform. This platform will hold the microscope stage and can be moved vertically to focus the imaging plane. We begin by affixing a mounting plate adapter to the linear stage using four shorter, silver ¼-20 machine screws. These should be hand-tightened using an appropriate Allen wrench. Note the orientation of the adapter plate. Once affixed, the 1" mount plate should be attached to the silver adapter plate using four 8-32 screws. Again, note the orientation of the mount – the 1" diameter channel in the mount plate should be parallel to the direction of travel of the linear stage. Lastly, prepare a (8" length) 1" diameter post by placing a ¼-20 set screw in one end of the post. Note that the set screw should have the Allen head *facing outward*.

Step 2: Affix main support posts and continue stage assembly
Pick a location relatively close to the edge of your breadboard centered along the short side of the platform. Screw the 1" post into the breadboard using the previously placed set screw. Loosen the screw on the 1" mount plate and pace the linear stage assembly on the 1" post as shown below. Align the linear stage to the breadboard grid and tighten the mount clamp.

Prepare a second 16"-long post using two set screws – one to connect two posts together; and another to fix the combined 16" post to the breadboard. Affix the post 10" from the 8"-long post that will hold the microscope stage. (Note that the breadboard has tapped holes with 1" spacing.)

Using four longer ¼-20 screws, attach the two L-brackets to the linear stage. Note that these brackets have a righthandside and lefthandside, and a top and a bottom. A notch should appear toward the center, top of the bracket as shown below. The hole patterns are also asymmetric and can only be assembled with the top, up. Only tighten these screws enough to hold on the brackets. We will fully tighten them once the microscope stage is attached.

Step 3: Finish microscope stage assembly
Next we attach the microscope stage to the linear stage. Handle the microscope stage with care. There is a large glass window in the stage that is fragile. Moreover, scratching this stage or getting excess dust or debris on the stage will complicate future experiments. This stage is held on with four more ¼-20 screws. Once you've gotten these screws place and started, you may tighten all eight screws on the two L-brackets. You should have just enough room for the stage between the two upright 1" posts. If necessary, you may need to adjust the knob on the stage which controls XY position.

Step 4: Assemble the lens-camera system components
In the next step we begin the assembly of the optical path of the imaging system. The major elements here are the objective lens, an optical tube assembly and the camera. These elements are more fragile that items in the previous steps and require additional care.

The lens is threaded differently than the optical tubes and requires an adapter. Place that adapter on the lens and hand tighten. A manual aperture (iris) may then be affixed to the back of the lens followed by an optical tube. For this lab, start with an optical tube of 1" length.

The camera is also threaded differently than the optical tubes and requires its own adapter. Hand tighten that adapter and place a ½" optical tube on the camera assembly. To the optical tube, the filter wheel assembly may be attached, followed by a 2" optical tube.

Step 5: Mount lens-camera system
The lens-tube-camera assembly is held in place by two optical tube mounts. Each of these is composed of the tube mount, a ½" diameter post, and a right-angle mount that attaches to the taller 1" post. Assemble each tube mount and roughly position the posts to accept the lens-tube-camera assembly.

NOTE: Do not allow the tube mounts or the tube assembly to fall onto the glass platform of the microscope stage. You can break the glass in the platform if these drop from a significant height.

Place the lens-tube half of the assembly in the bottom mount and the tube-camera assembly in the upper mount and screw them together. Make fine adjustments to the lens-tube-camera assembly ensuring that:
- The tube assembly is aligned with the circular hole in the bottom of the microscope stage. - The optical tube assembly is vertical and parallel to the 1" support post.
Once positioned, hand tighten the screws that hold the right angle mounts to the 1" and ½" posts. And, tighten the optical tube mounts to secure the lens-tube-camera assembly.

Step 6: Add the illumination module
Next, we will be attaching the lighting module to the microscope. This module is composed of three elements. A support arm, a coupler, and the lighting module itself. The support arm is attached to the linear stage using two ¼-20 screws. It should fit securely between the L-brackets at the lowest available screw positions (see below). The coupler and the light module slide together as shown below.

Do not apply significant pressure to the support arm. It is sufficient strong to hold the lighting module but will break easily if used to lift the stage, used as a brace for fastening screws, etc.

Step 7: Align the lighting module and add a diffuser
Note that the lighting module is quite bright. Do not look directly into the LED light.

The light module may be activated by plugging in the provided power supply. A brightness adjustment is provided on the side of the module for some control. Note that there is a lower threshold where the LED will not light. The glass diffuser should be placed in the channel created by the two L-brackets. ***This diffuser should slide in easily.*** If it does not, contact the instructor. Also note that this diffuser can slide out easily. ***Move your microscope with care to avoid letting the diffuser fall out and break.***

The position of the light module should be adjusted so that the circular illumination covers the circular hole in the bottom of the microscope stage.

Adjust the illumination at a low setting where the LED light is on and stable.

Step 8: Start imaging – refine microscope set up and start experimenting
Connect the camera to your computer using the provided cable. Note that a USB 3.0 port is required for reliable communication. There should be a green light on the camera if everything is working properly.
You should now be ready to start imaging with your microscope. To start taking images, place an imaging target or microscope slide on the stage. (See below.)

IMPORTANT: While stage clips are provided to hold a slide in place, the clips should NEVER be placed on the part of the slide with a sample, and NEVER used on the resolution target at all.

Remotely controlling the microscope

Remote log in to your system. Open the Camera app on Windows to get three views of the system. On Blackboard, we have provided labeled images (Bench Setup on Blackboard) to help you orient yourselves with system setup.

There are three modules you need to control programmatically: lighting, motion, and camera.

Lighting

Lighting is controlled via a USB relay board. Copy lighting.py into your working directory, initialize a lighting instance, and use the set_intensity function to adjust the light intensity. An intensity of 0 turns off the light; the maximum allowable intensity is 63.

Initialization:
$\mathfrak{o}mlight \in gimp$ or $tLight \in g$
$l = Light \in g()$ Main functions: - $l.\ set_{\int}ensity\left(\int ensity\right)$ set illumination level, 0 (off) to 63 (max) - $l.\ close()$ turn off light and close the lighting instance ***IMPORTANT: Turn off the light after you are finished with an experiment. The LED will overheat if it stays on for a long time.***

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()
```

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

In [3]:
```
l.set_intensity(10)
c.open()
f_wheel,stage_y,stage_x,ap,stage_z = 1,2,3,4,5
```

```
ThorCam opened successfully.
```

In [3]:
```
#l.close()
```

Motion

There are five servo motors controlling five components of the microscope. We communicate to the servo motors via a dedicated controller (Pololu). Copy the maestro-master folder, and pololu.m into your working folder.
Each of the five components is connected to a "channel" on the controller. The channel numbers are fixed for all systems. Each motor has a limited range that may be system dependent. The channel numbers and limits are shown below:

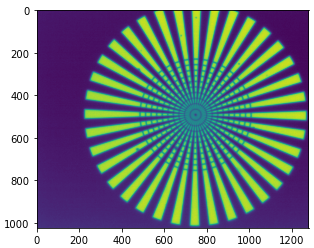
- channel 1 - filter wheel, limits: 496-2496
- channel 2 - stage y, limits: 496-2496
- channel 3 - stage_x, limits: 496-2496
- channel 4 - aperture, limits: variable on each station
- channel 5 - stage z, limits: 496-2496

Initialization:
$\mathfrak{o}mpololuimp$ or $tPololu$
$m = Pololu()$ Main functions: - $m.\ set_{p}osition(cha \cap el_{v}mber, position)$ set position for channel, limits are variable as shown above - $m.\ \geq t_{p}osition(cha \cap el_{v}mber)$ get position of channel - $m.\ s\ \top\ (cha \cap el_{v}mber)$ stop motion of channel - $m.\ close()$ close instance

In [5]:
```
m.set_position(stage_x, 1150, blocking= True)
m.set_position(stage_y, 2000, blocking = True)
```

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, clear_output

img = c.capture()
plt.imshow(img)
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x201eae5eca0>
```



Camera

We are using a Thorlabs DCC3240M camera for this class. Copy camera.m to your working folder.
Initialization:
$\mathfrak{o}mcameraimp$ or $tCamera$
$c = Camera()$
$c.\ open()$ Main functions: - $c.\ \cap ture()$ capture an image, return as numpy array - $c.\ \geq t_{\times}erties()$ get properties of camera - $c.\ set_{\exp o}sure(\exp osure)$ set exposure, in ms - $c.\ \geq t_{\exp o}sure()$ get exposure, in ms - $c.\ set_{a}merate(amerate)$ set framerate, in frame/sec - $c.\ \geq t_{a}merate()$ get framerate, in frame/sec - $c.\ close()$ close camera instance

Integrating lighting, motion, and camera capture

Move $sta \geq_{x}$ and $sta \geq_{y}$ to image the **STAR SECTOR 10D** (green cycle below).
If the target is not in the field-of-view of the camera, try moving the x and y stage using and $m.\ \geq t_{p}osition(cha \cap el_{v}mber)$ and $m.\ set_{p}osition(cha \cap el_{v}mber, position)$ to move the target.

Camera

We are using a Thorlabs DCC3240M camera for this class. Copy camera.m to your working folder.
Initialization:
$omcameraimp \text{ or } tCamera$
$c = Camera()$
$c.open()$ Main functions: - $c. \cap ture()$ capture an image, return as numpy array - $c. \geq t_{\propto}erties()$ get properties of camera - $c.set_{\exp o}sure(\exp osure)$ set exposure, in ms - $c. \geq t_{\exp o}sure()$ get exposure, in ms - $c.set_{a}merate(amerate)$ set framerate, in frame/sec - $c. \geq t_{a}merate()$ get framerate, in frame/sec - $c.close()$ close camera instance

When doing experiment, please monitor the stage motion at all time. The motors should be moving smoothly. If you observing jarring motions, or the camera is shaking, something is wrong. Please contact the instructors and TA immediately.
When finish using the equipments, close the modules, save your notebook, and shut down the kernel before logging out. Please make sure the light is turned off!

Focusing and image formatting

To change the focus, you rotate the screw adjustment on the linear stage (see below). In the remote setup, this is accomplished by moving the z-stage (variable name stage_z from above).

The following instructions are for the in-person version of the class, but are provided here for completeness.

IMPORTANT: This knob should rotate freely, if you experience resistance stop immediately. Note that you can drive the microscope stage into the lens if you are not careful (damaging the lens or microscope stage or target). Best practice is to focus "away from the lens." That is start with the stage near the lens and move the stage downward.

Practice finding the in-focus image for your target. Best focus is most easily found by looking at the sharpness of features (e.g. edges) in your image. If you are having difficulty, ask the instructor for help.

If you cannot find a good in-focus image and your image is mostly dark or mostly white, jump ahead to "adjusting signal levels".

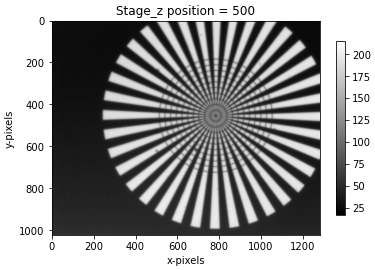
Your image is likely rotated and/or flipped relative to its position on the microscope stage. To fix this will require both hardware and software modifications. First, you should rotate the lens-camera assembly to be aligned with the XY-motion axes. This should be accomplished by loosening the tube clamps only (see below). ***WARNING: Do not let the tube assembly drop and fall on the microscope stage. Hand tighten clamps once the assembly is aligned.*** Your images are potentially rotated, flipped, or transposed. Modify your script by applying $np.rot90(img)$, $np.transpose(img)$, $np.fliplr(img)$, and/or $np.flipud(img)$ commands.

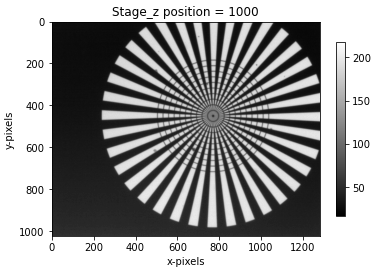
Data collection: Collect a series of images at different levels of focus.

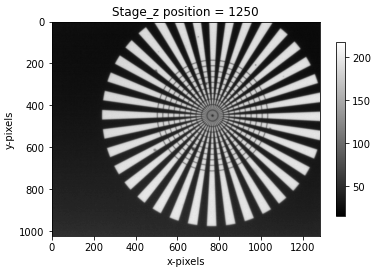
1. Which image would you consider is in focus? Show the in-focus below with x, y axis labels (in pixels) and a colorbar.

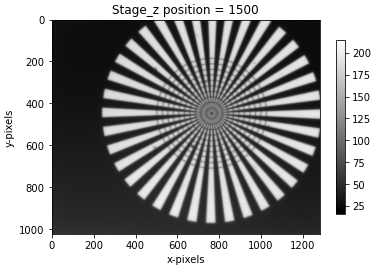
In [6]:
```
# Check a variety of z positions to see which image is the clearest
z_positions = [500, 1000, 1250, 1500, 2000, 2150, 2400]
images = []
for p in range(0, len(z_positions)):
    m.set_position(stage_z, z_positions[p], blocking = True)
    images.append(c.capture().copy())

l.set_intensity(0)
```

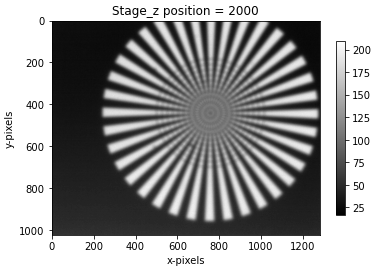
In [7]:
```
for i in range(len(images)):
    plt.figure()
    x = plt.imshow(images[i], cmap = "gray")
    plt.title("Stage_z position = " + str(z_positions[i]))
    plt.xlabel("x-pixels")
    plt.ylabel("y-pixels")
    plt.colorbar(x, shrink=0.8)
```

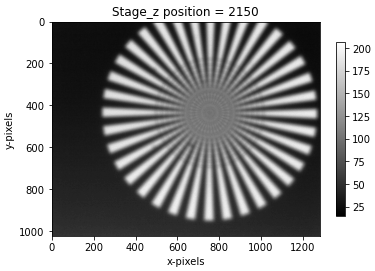


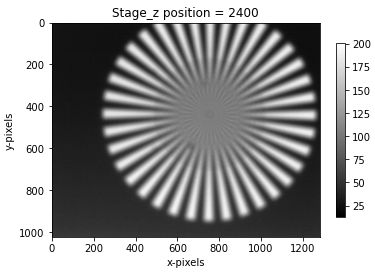






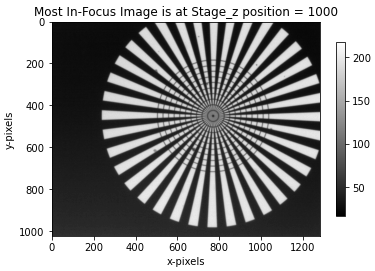






In [8]:
```
# Most In-Focus Image was at stage_z=1000
finFocus = plt.imshow(images[1], cmap = "gray")
plt.title("Most In-Focus Image is at Stage_z position = 1000")
plt.colorbar(finFocus, shrink=0.8)
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
```

Out[8]:
```
Text(0, 0.5, 'y-pixels')
```

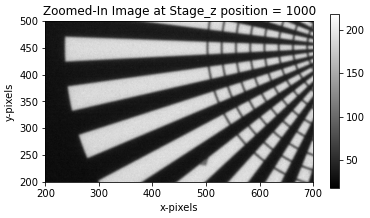


2. Why do you think this image is in focus? (Be specific and describe what features you are looking at. Provide a zoomed image to illustrate these features.)

We have classified this image as the most in focus image because the edges of the image are the sharpest. I.e. The edges at which the image transitions from black to white are most clear and defined. We believe that this image is in focus because the object is at an ideal position such that the incoming rays from the object through the lens converge and intersect close enough to where the camera is positioned. Since the camera is within the acceptable z position where the image will be in focus, we get a clear image when the object is at the distance from the camera associated with the stage_z position of 1000.
The zoomed view of the in-focus image exhibits clearer edges between the white and black portions of the image, whereas other images have blurrier edges. (Discussed in the next question). The triangular slices remain more defined (the black and white portions are clearly distinguishable with minimal blur) as the viewer looks from the edge of the circle towards the center of the circle. The pattern in the center of the circle is more distinguishable and clear as well.

In [9]:
```
# Changing window to get a closer look at slices
focZoom = plt.imshow(images[1], cmap = "gray")
plt.title("Zoomed-In Image at Stage_z position = 1000")
plt.colorbar(focZoom, shrink=0.8)
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.xlim(200,700)
plt.ylim(200,500)
```

Out[9]:
```
(200.0, 500.0)
```

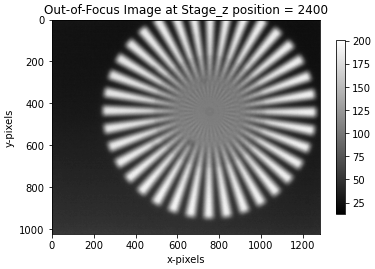


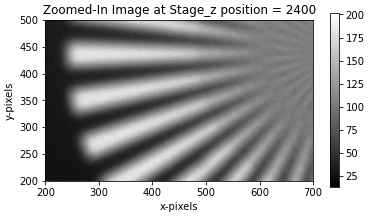
3. Pick an out-of-focus image and show it below. How is this image different to the in-focus image you showed above?

This image has less clear and less distinguishable edges than the above in-focus image. There is a blurrier transition from the black background to the white triangles. The pattern at the center of the circle is also less distinguishable.

In [10]:
```
# Out of focus image at stage_z=2400
finOFocus = plt.imshow(images[len(images) - 1], cmap = "gray")
plt.title("Out-of-Focus Image at Stage_z position = 2400")
plt.colorbar(finOFocus, shrink=0.8)
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")

# Changing window to get a closer look at slices
plt.figure()
focOZoom = plt.imshow(images[len(images)-1], cmap = "gray")
plt.title("Zoomed-In Image at Stage_z position = 2400")
plt.colorbar(focOZoom, shrink=0.8)
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.xlim(200,700)
plt.ylim(200,500)
```

Out[10]: (200.0, 500.0)





Adjusting image signal levels

The data that is acquired by your camera is 10-bit. This means that the camera is capable of reporting values between 0 and 1023. Generally, you will want to collect data that covers most of this range. The following terms will be important to know:
- **Underexposure** – A lack of integrated signal at the detector. Most of the values will be on the low end of the 0-1023 range. Python will automatically scale the colorbar to range between the minimum and maximum values, so the image won't necessarily appear dark; however, increased noise will be apparent.
- **Saturation** - The camera cannot represent signal levels above 1023. Thus, signal levels above that range will "saturate" (or be limited) at 1023. This results in a loss of contrast for any image regions above that level.
- **Overexposure** – Too much signal is also a problem for image quality. Generally overexposed images will exhibit saturation in some part of the image.
- **"Good" exposure** – This is somewhat qualitative; however, a "good" exposure should not be underexposed (too noisy) and should not have any saturated regions. e.g. All pixel values fall in the range 0-950.

There are three basic ways to change the signal level in your imaging system:

- **Changing the illumination level** - The lighting module has an adjustment that permits changing the brightness of the LED illumination.
- **Changing the size of the aperture** - An adjustable iris has been installed right above the lens. This aperture controls the diameter of the light beam that is exiting the rear side of the lens. Smaller apertures will reduce signal levels and larger apertures will increase signal levels.
- **Changing the exposure time** – Your camera has the ability to adjust the integration time over which signal is measured for individual images. The current setting can be found by looking at the Python camera object. (The variable `c` in your python script above.) You may set this value using the statement, for example, `c.set_exposure(10)` . This sets the exposure to 10 milliseconds. Longer integration times permit accrual of more signal, while shorter exposure times result in less signal. The maximum exposure time is limited by the camera framerate (also adjustable). For example, changing framerate with `c.set_framerate(1)` sets the rate to 1 frame/second permitting exposures up to 1000 ms.

Exposure and illumination experiments:
One of the following biological slides has been placed on your microscope. Set your LED illumination to the lowest stable setting. Open your aperture to the widest setting. Move the filter wheel to one of the open (no filter) settings.

Finding a good exposure setting for your image involves finding the maximum values in your image. While one could simply using Python's $np.\ \max\ ()$ function, this is sensitive to noise and single "bright" pixels. A better solution is to plot a histogram of pixel values and then select an exposure that yields a minimal number of saturated values. In the following cell, write code that will show an image from the microscope as well as its histogram of pixel values (with 1024 intensity bins).

Data collection: Collect a series of images at different levels of exposure.

In [11]:
```
# find the minimum and maximum aperture size
l.set_intensity(4) #low stable illumination intensity
m.set_position(ap, 4000, blocking = True)
m.get_position(ap) # get the max position (2496)
```

Out[11]: 2496

In [12]:
```
m.set_position(ap, 990, blocking = True)
m.get_position(ap) # get the min position (992)
```

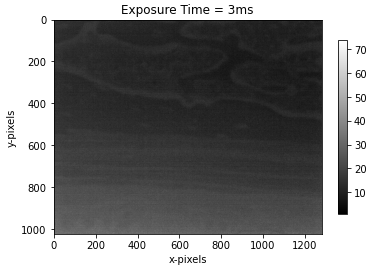
Out[12]: 992

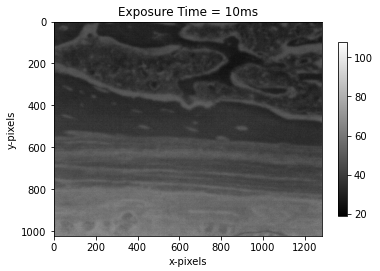
In [8]:
```
# Check a variety of ap positions to see which image is the best
l.set_intensity(4)
m.set_position(stage_y, 500, blocking = True)
m.set_position(stage_x, 1000, blocking = True)
m.set_position(stage_z, 1000, blocking = True)
m.set_position(f_wheel, 1540, blocking = True) # set filter to no filter
m.set_position(ap,2496, blocking = True) # highest aperture setting
exp_times = [3, 10, 100, 300, 500, 999] # ms
exp_images = []
for p in exp_times:
```

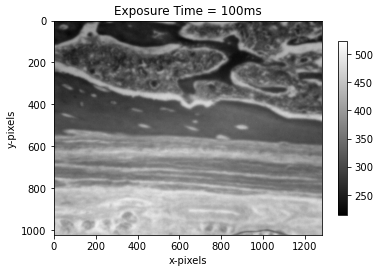
```
    c.set_exposure(p)
    c.capture() # make it wait before taking another exposure shot
    exp_images.append(c.capture().copy())
```

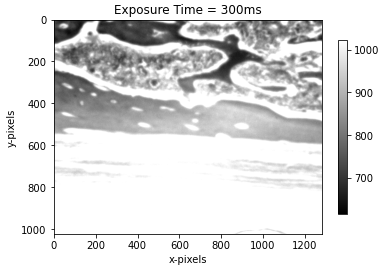
In [14]:

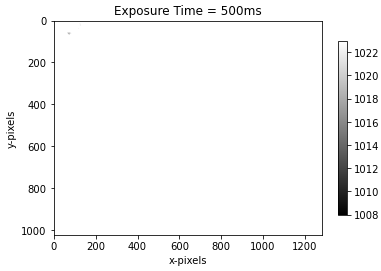
```
# print out images
for i in range(len(exp_images)):
    plt.figure()
    x = plt.imshow(exp_images[i], cmap = "gray")
    plt.title("Exposure Time = " + str(exp_times[i]) + "ms")
    plt.xlabel("x-pixels")
    plt.ylabel("y-pixels")
    plt.colorbar(x, shrink=0.8)
```

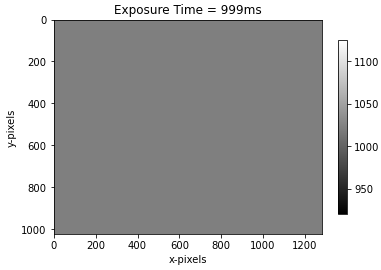












1. What is a good exposure setting for this scenario?
Show an image of the slide at these settings and paste it below:

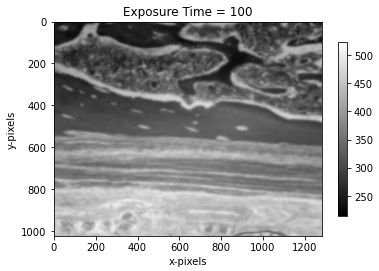
A good exposure setting found from the data collection above would be at 100ms. This one was chosen because 3ms and 10ms images were underexposed and the 500ms and 999ms were overexposed such that hardly any image detail could be seen. The 100ms gave better contrast compared to the 3ms and 10ms images and allowed us to better visualize the structures on the slide. We are able to tell that the image is the finger.

In [15]:

```
g = plt.imshow(exp_images[2], cmap = "gray")
```

```
plt.title("Exposure Time = " + str(exp_times[2]))
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.colorbar(g, shrink=0.8)
```

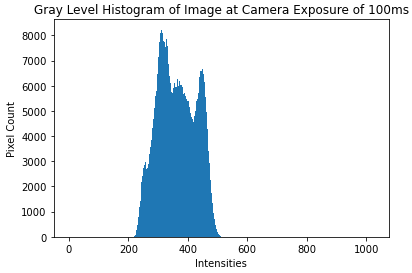
Out[15]: <matplotlib.colorbar.Colorbar at 0x267bacbe220>



2. Justify why this image represents a good exposure (include a gray level histogram as part of your discussion):

This image represents a good exposure because the pixel intensity values are not skewed to either extreme (minimum of 0 and maximum of 1023), as shown in the histogram below. If this were an over or under exposed image, we would have the pixel histogram with a large proportion of values at one end of the intensity spectrum or another. Therefore, since this histogram is not skewed very heavily to either extreme but still has variation, this is an appropriately exposed image. Additionally, this exposure is also in the "good exposure" range of intensities between 0 to 950.

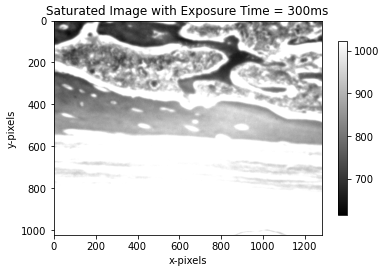
In [16]:
```
plt.hist(exp_images[2].ravel(),1024,[0,1024]);
plt.ylabel("Pixel Count");
plt.xlabel("Intensities");
plt.title("Gray Level Histogram of Image at Camera Exposure of 100ms");
```



3. Show an image of the slide where there is obvious saturation:

In [29]:
```
b = plt.imshow(exp_images[-3], cmap = "gray");
plt.title("Saturated Image with Exposure Time = 300ms");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(b, shrink=0.8)
```

Out[29]: <matplotlib.colorbar.Colorbar at 0x267be9b58e0>

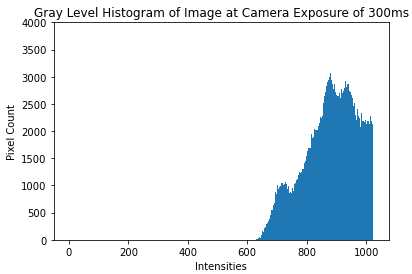


4. How can you tell that this image is saturated? Why is it saturated?

As you can see in the histogram below, many pixel intensities are close to the maximum value. As a result, the picture appears overexposed because the camera is letting in too much light, making it harder to distinguish features in the image. This means that the pixel intensities are also exceeding the "good exposure" range of (0,950) as there are many intensities above 950.

In [34]:
```
plt.hist(exp_images[-3].ravel(),1024,[0,1024]);
plt.ylabel("Pixel Count");
plt.xlabel("Intensities");
plt.title("Gray Level Histogram of Image at Camera Exposure of 300ms");
plt.ylim(0,4000) # graph's limits were being crazy so had to manually set limits
```

Out[34]: (0.0, 4000.0)

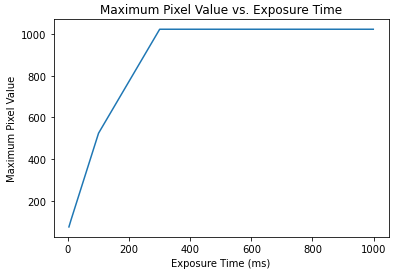


5. What is the relationship between exposure and maximum pixel value? Plot this for the data we collected and describe.

The general correlation between exposure and maximum pixel value is that as exposure increases, the maximum pixel

value (intensity in the graphs above) in the image increases as well. Additionally, there are more pixels closer to the maximum pixel values / higher pixel values as exposure time increases. Eventually as images become overexposed and hit a limit, all of the pixel values will be at the maximum (1023) for a 10-bit camera as the one we are using here.

In [19]:
```
max_values = []
for i in range(len(exp_images)):
    max_values.append(np.max(exp_images[i]))
plt.plot(exp_times,max_values);
plt.title("Maximum Pixel Value vs. Exposure Time");
plt.xlabel("Exposure Time (ms)");
plt.ylabel("Maximum Pixel Value");
```



Small aperture and image averaging

Data collection: Make multiple small aperture image acquisitions

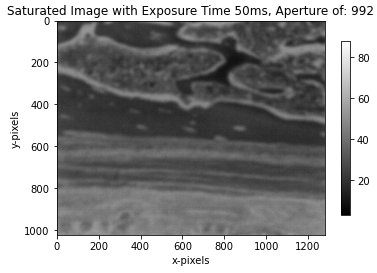
1. Close the aperture until your maximum image value is 80-100.
Show a single image of the slide at these settings and paste it below:

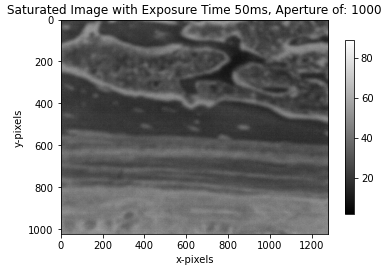
In [15]:
```
c.set_exposure(50)
l.set_intensity(10)
```

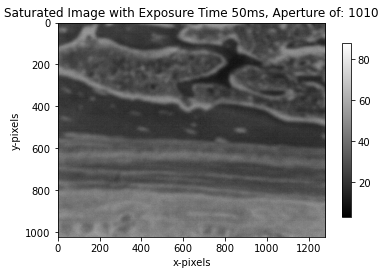
In [21]:
```
ap_arr = [992, 1000, 1010, 1020, 1030, 1040, 1050, 1060, 1070]
ap_imgs = []
print("Max Image Values")
for a in ap_arr:
    m.set_position(ap,a, blocking = True)
    imgs = c.capture().copy()
    print(np.max(imgs))
    plt.figure()
    x = plt.imshow(imgs, cmap = "gray");
    plt.title("Saturated Image with Exposure Time 50ms, Aperture of: " + str(a));
    plt.xlabel("x-pixels");
    plt.ylabel("y-pixels");
    plt.colorbar(x, shrink=0.8)
    ap_imgs.append(imgs)



plt.figure()
plt.hist(ap_imgs[4].ravel(),1024,[0,1024]);
plt.ylabel("Pixel Count");
plt.xlabel("Intensities");
plt.title("Gray Level Histogram of Image at Aperture of: " + str(ap_arr[4]));
plt.xlim(0,150)
```

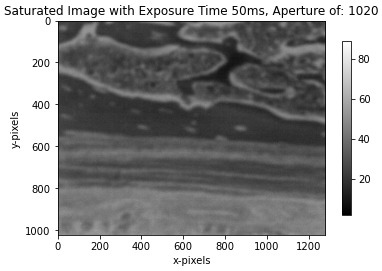
```
Max Image Values
88
89
88
89
84
89
95
94
98
```

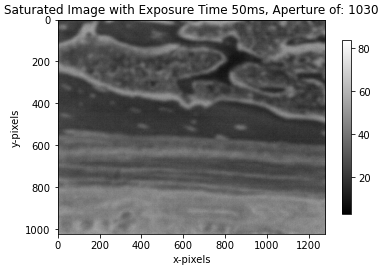
Out[21]: (0.0, 150.0)

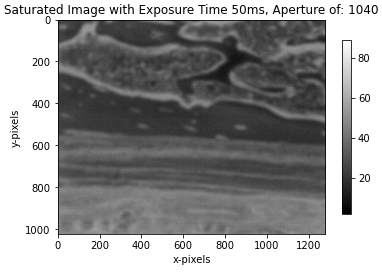


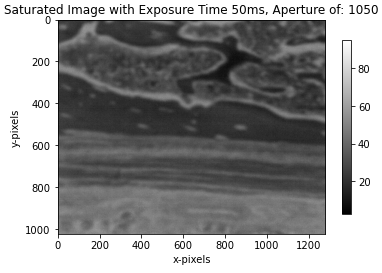


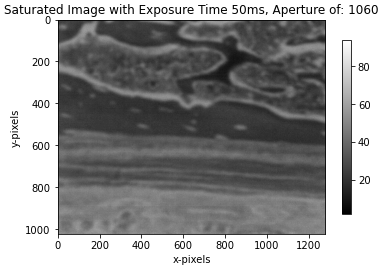

Saturated Image with Exposure Time 50ms, Aperture of: 1010
y-pixels
x-pixels

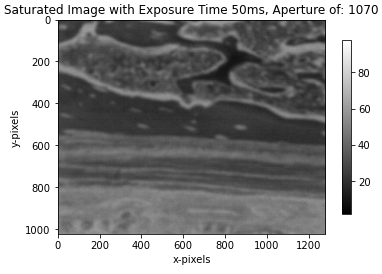

Saturated Image with Exposure Time 50ms, Aperture of: 1020
y-pixels
x-pixels

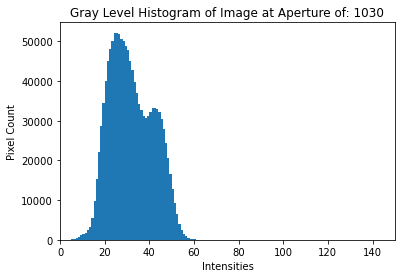

Saturated Image with Exposure Time 50ms, Aperture of: 1030
y-pixels
x-pixels


Saturated Image with Exposure Time 50ms, Aperture of: 1040
y-pixels
x-pixels


Saturated Image with Exposure Time 50ms, Aperture of: 1050
y-pixels
x-pixels

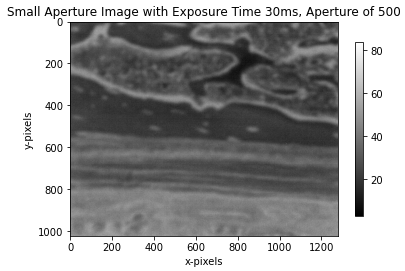

Saturated Image with Exposure Time 50ms, Aperture of: 1060
y-pixels
x-pixels


Saturated Image with Exposure Time 50ms, Aperture of: 1070
y-pixels
x-pixels



In [22]:
```
x = plt.imshow(ap_imgs[4], cmap = "gray")
plt.title("Small Aperture Image with Exposure Time 30ms, Aperture of 500");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
```

Out[22]:
```
<matplotlib.colorbar.Colorbar at 0x201ed15f250>
```



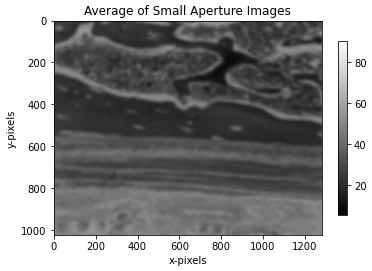
2. How is this image different than the good image you showed above? Why?

This image is unable to resolve as many of the smaller image features when compared to the images with better exposure. This image also has more noise which makes it blury. This is because these low light pixtures are highly sensitive to noise. Because all of the light values are in a relatively small range, low levels of noise have an exaggerated effect on image resolution that is not as present if the light values had a larger range.

3. Show an average of all the images that were acquired with the small aperture:

In [23]:
```
avg = np.stack(ap_imgs)
avg = np.mean(avg, axis = 0)
x = plt.imshow(avg, cmap = "gray")
plt.title("Average of Small Aperture Images");
plt.xlabel("x-pixels");
plt.ylabel("y-pixels");
plt.colorbar(x, shrink=0.8)
```

Out[23]:
```
<matplotlib.colorbar.Colorbar at 0x201eea3deb0>
```

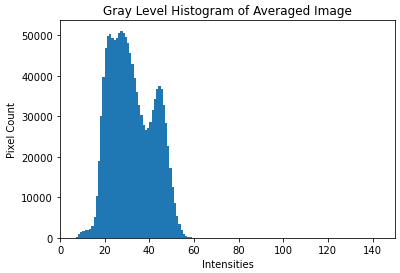


4. What happens when you average many of the images with small image values? Why?

It seems like the averaged image is slighltly less blury than the original images. This is likely due to the fact that the averaging process helps decrease some of the variability introduced by the noise. This helps improve The overall image resolution. Evidence of this decreased variability is how the gray level histogram of the average image has a sharper incline towards the mode image value than in the histogram before averaging.

In [24]:
```
plt.figure()
plt.hist(avg.ravel(),1024,[0,1024]);
plt.ylabel("Pixel Count");
plt.xlabel("Intensities");
plt.title("Gray Level Histogram of Averaged Image");
plt.xlim(0,150)
```

Out[24]:
```
(0.0, 150.0)
```



Moving the filter wheel

Move the filter wheel to filter 1, 2, and 3 respectively. There are three color filters in the filter wheel: 460 nm (blue), 540 nm (green), and 600 nm (orange). You will obtain one of the following four slides for imaging.

1. What do we need to do to get a good image?

In order to get a good image, we need to set an appropriate exposure time so that the image is not under or over exposed. Thus, we can acquire maximal detail and contrast for the structure on the slide. In terms of the filter wheel, we need to make sure that the wheel is properly positioned under the camera such that the view of the camera is fully covered by the color (or non color) portion of the wheel. This precision will minimize potential glare caused by the filter wheel and will also make sure that the view of the camera is not obstructed by the black plastic part of the color wheel. We should also choose the filter with the appropriate color to capture the image detail. This color could depend on the color of the dye that was used on the slide. For example, if blue dye was used, the detail on the slide will be distinguished by the blue stain, so using a blue filter will help us better visualize the details that have been stained blue.

Data collection: Image the biological slide with different color filters

2. Show the image data below. What effects do the filters have on the images of each slide? Explain why.

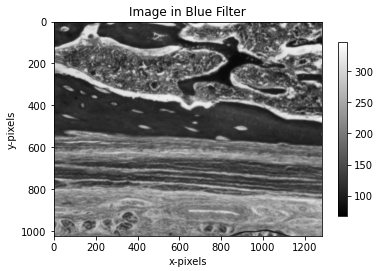
Essentially, the filter allows only light within a certain range of wavelengths to pass through, thus filtering out the other colors. For example, the filter acts as a bandpass filter, meaning that if a filter is a blue filter (associated with low wavelengths on the visible spectrum) then, this filter will only allow for those wavelengths to reach the camera. Therefore, an image taken with the blue filter will most heavily feature the blue sections of the slide. The blue filter highlights the details on the slide stained in blue. Similarly, the orange filter allows more of the part of the image stained pink to come through.

In [25]:
```
m.get_position(f_wheel)
```

Out[25]:
```
1540
```

In [35]:
```
l.set_intensity(20)
m.set_position(f_wheel,1965, blocking = True) #blue filter
c.set_exposure(100)
m.set_position(ap,2496,blocking = True)
blue = plt.imshow(c.capture().copy(), cmap = "gray")
plt.title("Image in Blue Filter")
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.colorbar(blue, shrink=0.8)
```

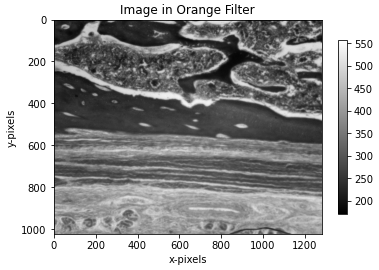
Out[35]:
```
<matplotlib.colorbar.Colorbar at 0x267b6815580>
```



This is the blue filter image, this image shows the most contrast within the boundary of the finger (we are assuming this slide is the finger).

In [36]:
```
l.set_intensity(20)
m.set_position(f_wheel,1805, blocking = True) #orange filter
c.set_exposure(100)
m.set_position(ap,2496,blocking = True)
red = plt.imshow(c.capture().copy(), cmap = "gray")
plt.title("Image in Orange Filter")
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.colorbar(red, shrink=0.8)
```

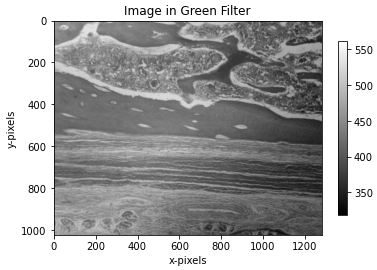
Out[36]:
```
<matplotlib.colorbar.Colorbar at 0x267b6ae06d0>
```



This is the orange filter image. There is more exposure outside the boundary of the finger, but there is some more detail visbile in this section compared to the blue filter. For example, inside of the finger, the details are more clearly seen and there is higher contrast in the bottom fifth of the image.

In [37]:
```
l.set_intensity(20)
m.set_position(f_wheel,2085, blocking = True) #blue bitty
c.set_exposure(100)
m.set_position(ap,2496,blocking = True)
green = plt.imshow(c.capture().copy(), cmap = "gray")
plt.title("Image in Green Filter")
plt.xlabel("x-pixels")
plt.ylabel("y-pixels")
plt.colorbar(green, shrink=0.8)
```

Out[37]:
```
<matplotlib.colorbar.Colorbar at 0x267b6a42760>
```



This is the green filter image. There are no obviously overexposed sections here and the image shows more detail in the bottom half of the image. However, it does not highlight the stained structures and details as well as the blue and orange filters did.

In [38]:
```
l.close()
```