CZ BioHub_Take Home Test

Bo-Yen Chang

For Associated R&D Engineer

For Each Exercise:

- **Work**
- **Choice Made**

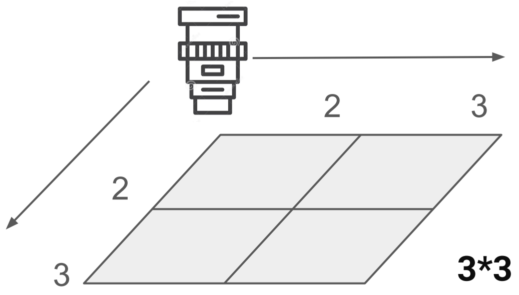
- **Challenges**
- **Solution**

Exercise 1

Exercise 1 — Explain the Work, Choice Made

Acquiring a multi-dimensional dataset using microscopy simulation

Work Explanation



```
z_step = 0.1
num_z_slices = 12
spacing = 500
channels = ['DAPI', 'FITC']
```

Choice made

Pycro Manager

- `pycromanager` —— complex multi-dimensional acquisition and real-time image processing
- `pymmcore-plus` —— lightweight, direct hardware control, without UI or advanced features.

Multi_dimensional_Acq_Events

```
# Step 6: Generate acquisition events, Using multi_d_acquisition_ev
events = multi_d_acquisition_events(
    num_time_points=1,     # Set as 1, acquisit once at each event
    z_start=0, z_end=z_step*(num_z_slices-1.5), z_step=z_step, #Bas
    channels=channels,
    xy_positions=positions,
    order='tpcz'           # Acqusition order: time, position, chan
)
```

Exercise 1 — Challenges and solutions

Challenges

- Connect With the Micro-Manager

Solutions

→ Look into the Official Document

- Connect to port 4827
- `pip install pycromanager`
- `import pycromanager`

Challenges

- Data Acquisition on Micro-Manager

Solutions

→ Official Document

```
with Acquisition(directory='/path/to/saving/dir', name='acquisition_name') as acq:
    events = multi_d_acquisition_events(
                                        num_time_points=4, time_interval_s=0,
                                        channel_group='Channel', channels=['DAPI', 'FITC'],
                                        z_start=0, z_end=6, z_step=0.4,
                                        order='tcz')
    acq.acquire(events)
```

Challenges

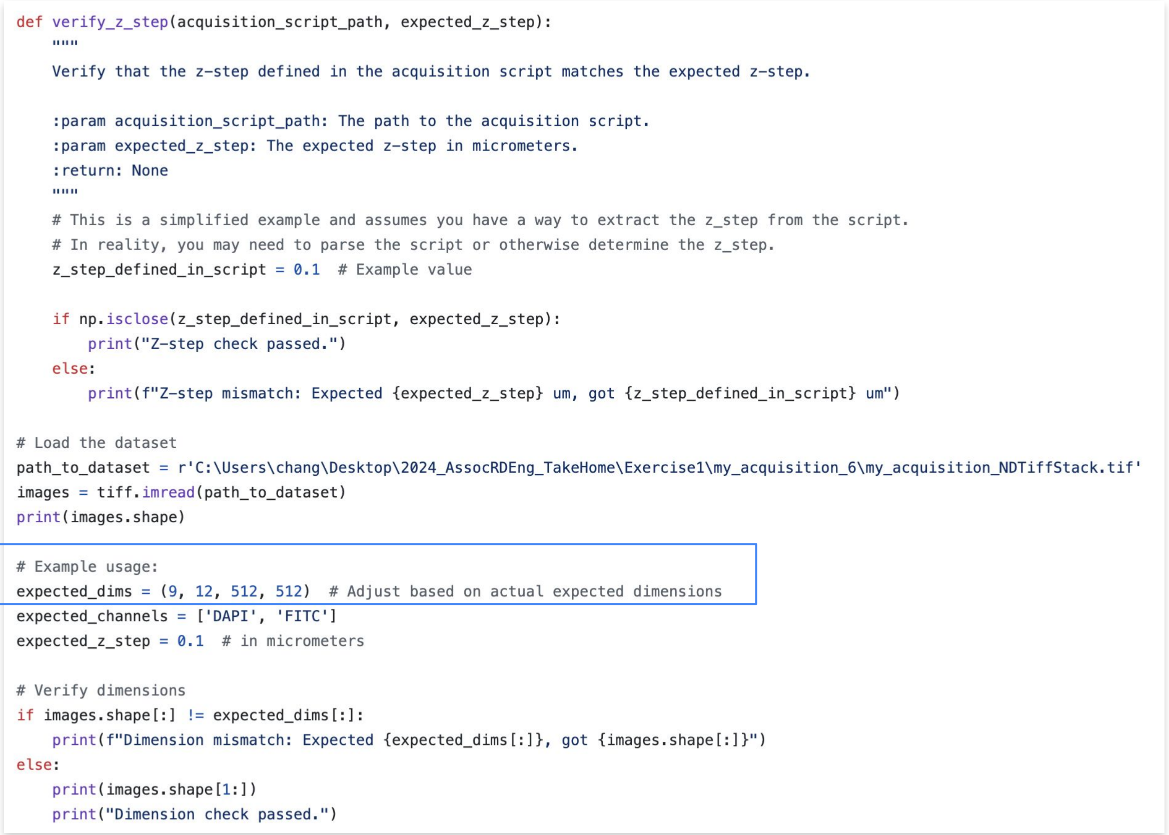
- Dealing multi-dimensional data

Solutions

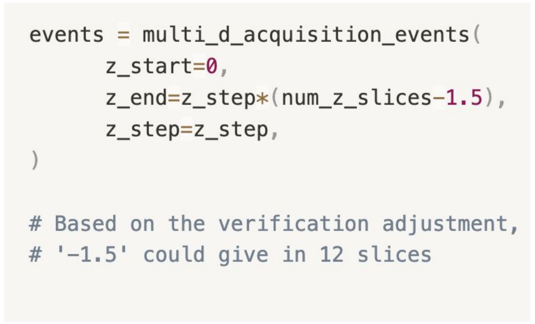
→ Two Method:

1. `events = multi_d_acquisition_events()`
2. Loop in the Acquisition Function

Exercise 1 — Verification Code to Check the Shape



```python
def verify_z_step(acquisition_script_path, expected_z_step):
    """
    Verify that the z-step defined in the acquisition script matches the expected z-step.

    :param acquisition_script_path: The path to the acquisition script.
    :param expected_z_step: The expected z-step in micrometers.
    :return: None
    """
    # This is a simplified example and assumes you have a way to extract the z_step from the script.
    # In reality, you may need to parse the script or otherwise determine the z_step.
    z_step_defined_in_script = 0.1  # Example value

    if np.isclose(z_step_defined_in_script, expected_z_step):
        print("Z-step check passed.")
    else:
        print(f"Z-step mismatch: Expected {expected_z_step} um, got {z_step_defined_in_script} um")

# Load the dataset
path_to_dataset = r'C:\Users\chang\Desktop\2024_AssocRDEng_TakeHome\Exercise1\my_acquisition_6\my_acquisition_NDTiffStack.tif'
images = tiff.imread(path_to_dataset)
print(images.shape)

# Example usage:
expected_dims = (9, 12, 512, 512)  # Adjust based on actual expected dimensions
expected_channels = ['DAPI', 'FITC']
expected_z_step = 0.1  # in micrometers

# Verify dimensions
if images.shape[:] != expected_dims[:]:
    print(f"Dimension mismatch: Expected {expected_dims[:]}, got {images.shape[:]}")
else:
    print(images.shape[1:])
    print("Dimension check passed.")
```



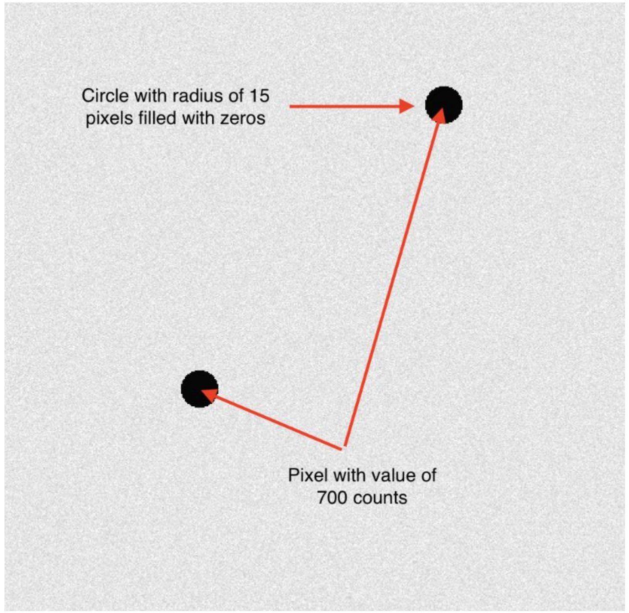
```python
events = multi_d_acquisition_events(
        z_start=0,
        z_end=z_step*(num_z_slices-1.5),
        z_step=z_step,
)

# Based on the verification adjustment,
# '-1.5' could give in 12 slices
```

Exercise 2

Exercise 2 — Explain the Work, Choice Made

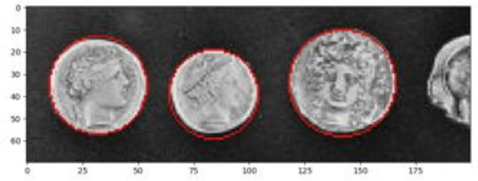
Modifying images in real-time during acquisition based on pixel value

Work Explanation


Circle with radius of 15 pixels filled with zeros
Pixel with value of 700 counts

Choice made

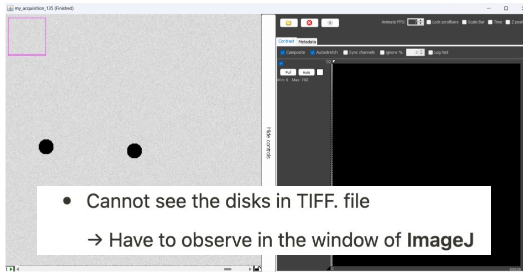
Skimage.draw.disk



Process Image After Acquisition → `image_process_fn`

```
def img_process_fn(image, metadata):

    ### send image and metadata somewhere ###

# this acquisition won't show a viewer or save data
with Acquisition(image_process_fn=img_process_fn) as acq:
    ### acquire some stuff ###
```

Exercise 2 — Challenges and solutions



- Cannot see the disks in TIFF. file

  → Have to observe in the window of **ImageJ**

Challenges

- Cannot do disk drawing in Acquisition Function()
  - Acquisition Func() can only do event acquiring

- Data Type — **`event`** , **`image`** wrong
  - `image` should be 'numpy.array' but show 'dict'
  - `event` should be dict but show 'numpy.array' or 'queue'

Solutions

→ Have to use **`image_process_fn`** to call out def()

- Define two function
  - **`process_image(image)`**
    - Draw disk when image =700
  - **`image_process_fn(image, metadata)`**
    - Processing modified image (Save pics)
    - `metadata` is the event in this function

→ 1. Print in each process to see its 'type', 'shape'
  2. Look into official document for function usage
  3. Ask for micro-manager community

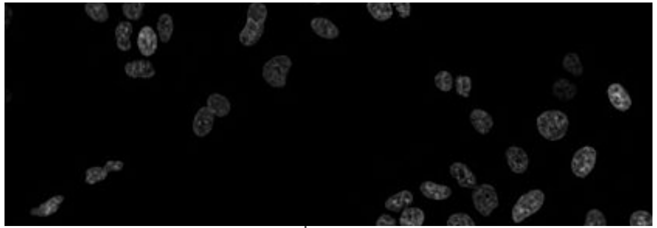
- **`image_process_fn(image, metadata)`**
  - has strict order for sending back value
- The `event` in **`image_process_fn()`** called `metadata`

Exercise 3

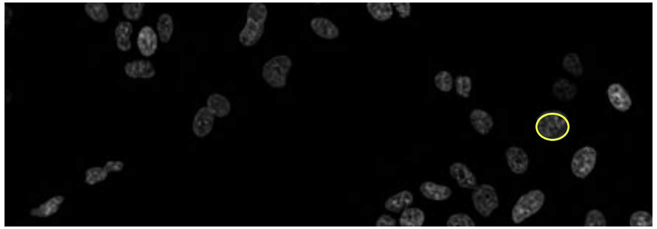
Exercise 3 — Explain the Work, Choice Made

Segmenting cell nuclei and measuring their shape deviations

Work Explanation



Segmentation

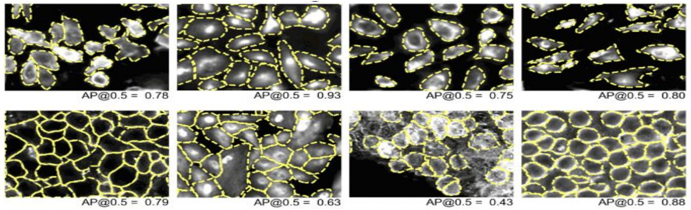


Eccentricities →

Choice made

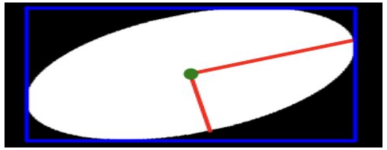
Segmentation - Cellpose:

- easy to access and specific to cell
- model()



Eccentricities - Skimage:

```
measure.regionprops(mask).eccentricity
```



Exercise 3 — Challenges and solutions

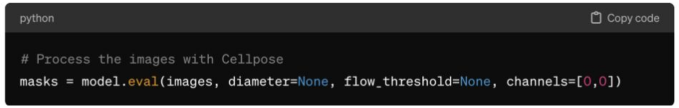
Challenges

- The images cannot be processed directly

- The data including 17 images, all of the images have multiple dimension

- masks = model.eval()

  `'list' object has no attribute 'ndim'`



python Copy code

```
# Process the images with Cellpose
masks = model.eval(images, diameter=None, flow_threshold=None, channels=[0,0])
```

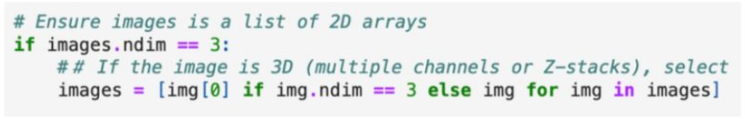
Solutions

→ To ensure ‘image’ was type: ‘numpy array’

- using `io` in `scikit-image` to read the TIFF.

→ 1. Using ‘for’ loop to go over each images
2. Ensure images is the list of 2D arrays

- Only take first channel for processing



```
# Ensure images is a list of 2D arrays
if images.ndim == 3:
    ## If the image is 3D (multiple channels or Z-stacks), select
    images = [img[0] if img.ndim == 3 else img for img in images]
```

→ model eval()

- model eval() must send back 4 variables
- `masks, flows, styles, diams = model.eval()`

What I have learned

- Learning the automation of microscopy from scratch
- Ask ‘Official Document’ and ‘Community’ for help
- Seeking for multiple tools to try more in different angles

Thanks!