Module_2:

Team Members: Kaiwen Huang, Christy Lee

Project Title: multi-depth lung biopsy image analysis pipeline for fibrosis disease prediction

Project Goal:

This project seeks to provide an image analysis pipeline that predicts and quantifies the extent of pulmonary fibrosis across different biopsy depths from the top of the lung. The pipeline will integrate line regression(interpolation) and scatter plot visualization to map fibrosis severity as a function of biopsy depth, supporting more precise and depth-resolved diagnostic evaluation of lung tissue.

Disease Background:

- Prevalence & incidence
    - estimated at 13.4 to 18.5 per 100,000 people have the disease in epidemiologic studies(Zhang and Newton)
    - Up to 20 % of interstitial lung disease may be familial(Liu et al.)
    - in the US, 132000-200000 people live with it
    - tends to affect more men than women
    - most in over 50 years of age
- Risk factors (genetic, lifestyle)
    - old age
    - being male
    - tobacco/cigarette smoking
    - genetic factors
    1. Mutations in TERT / TERC (telomerase genes)
    2. The MUC5B promoter variant(Michalski and Schwartz)
    - viral/metal&wood dust exposure
    - gastroesophageal reflux
- Symptoms(American Lung Association)
    - difficulty breathing
    - dry cough and fatigue
    - basic daily life tasks become difficult

    - there is a velcro like crackle in the lungs
    - can be associated w coronary artery disease and pulmonary hypertension
    - cyanosis (blue skin)
    - digital clubbing (roudning of nailbeds)
    - significant respiratory failure
- Standard of care treatment(s)(Jiang et al.)
    - cause not known
    - lung transplant without contraindications is the only cure
    - supplemental oxygen and pulmonary rehabilitation and FDA approved medicine can * * help (antifibrotic medications)
    - diagnosis through ct scan or lung biopsy
    - pulmonary function test (spirometry)
- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology) (Giacomelli et al.)
    - the tissue of the alveoli forms permanent scaring that impairs the lungs ability to absorb oxygen and function normally (interstitial tissue in lung)
    - fibroblasts develop into myofibroblasts which then sevrete retricular fibers (give structural strength) and elastic fibers (elasticity of lungs), then the myofibroblasts normally undergo apoptosis. In IFP, the type 2 pneumocytes overproliferate during the repair process
        - leads to too many myofibrilblasts and too much collagen
        - myofibrilblasts dont undergo apoptosis and continue to make too much collagen --> interstitial layer thickens --> problems with ventilation and oxygenation; lungs become too stiff --> restricted lung expansion (interstitial lung disease)
    - decrease total lung capacity
    - decreased forced vital capacioty
    - decreased forced expiratory volume in 1 sec
    - fluid filled alveoli covered by cysts ("honeycombing")

Citation:

1. Zhang, David, and Chad A. Newton. “Familial Pulmonary Fibrosis.” Chest, vol. 160, no. 5, Elsevier BV, Nov. 2021, pp. 1764–73, https://doi.org/10.1016/j.chest.2021.06.037. Accessed 8 Oct. 2025.
2. Liu, Qi, et al. “The Genetic Landscape of Familial Pulmonary Fibrosis.” American Journal of Respiratory and Critical Care Medicine, vol. 207, no. 10, American Thoracic Society, Jan. 2023, pp. 1345–57, https://doi.org/10.1164/rccm.202204-0781oc. Accessed 8 Oct. 2025.
3. Michalski, Jacob E., and David A. Schwartz. “Genetic Risk Factors for Idiopathic Pulmonary Fibrosis: Insights into Immunopathogenesis.” Journal of Inflammation

Research, vol. Volume 13, Informa UK Limited, Jan. 2021, pp. 1305–18, https://doi.org/10.2147/jir.s280958. Accessed 8 Oct. 2025.
4. American Lung Association. “Symptoms of Pulmonary Fibrosis.” Lung.org, 2025, www.lung.org/lung-health-diseases/lung-disease-lookup/pulmonary-fibrosis/introduction/symptoms?utm_source=chatgpt.com. Accessed 8 Oct. 2025.
5. Jiang, Mengna, et al. “Pulmonary Fibrosis: From Mechanisms to Therapies.” Journal of Translational Medicine, vol. 23, no. 1, Springer Science and Business Media LLC, May 2025, https://doi.org/10.1186/s12967-025-06514-2. Accessed 8 Oct. 2025.
6. Giacomelli, Chiara, et al. “Pulmonary Fibrosis from Molecular Mechanisms to Therapeutic Interventions: Lessons from Post-COVID-19 Patients.” Biochemical Pharmacology, vol. 193, Elsevier BV, Oct. 2021, pp. 114812–12, https://doi.org/10.1016/j.bcp.2021.114812. Accessed 8 Oct. 2025.

Data-Set:

- 78 black and white images paired with a recorded biopsy depth (µm) from the lung surface (ranging from 15 µm downward) were collected at different depths into a fibrotic mouse lung in Professor Pierce-Cottler's lab.
- A bleomycin-induced mouse model for pulmonary fibrosis is introduced in the experiment. In the the mouse was injected with bleomycin, which causes the mouse to develop a condition similar to IFP(Moeller A. et al. (2008))
- The mouse lung was fixed with paraformaldehyde, tissue were mounted in gel, and sliced with a cryotome/microtome and placed on a glass microscopy slide. (sliced in the transverse plane)
- the slices were immunostained with a fluorescent-labeled antibody, which binds to the protein of interest.
- Fluorescent signal intensity was captured in black-and-white (8-bit) images, where pixel brightness represents fluorescence magnitude—a quantitative marker of fibrosis severity.
- By analyzing the data-set, it supports the development of an image analysis pipeline to predict and quantify fibrosis severity as a function of biopsy depth, using scatter plots and linear regression (interpolation) to model fibrosis progression through the lung‘s vertical axis.

Data Analyis:

This will create a csv file of the percent of white pixels and black pixels in the images

In [ ]:

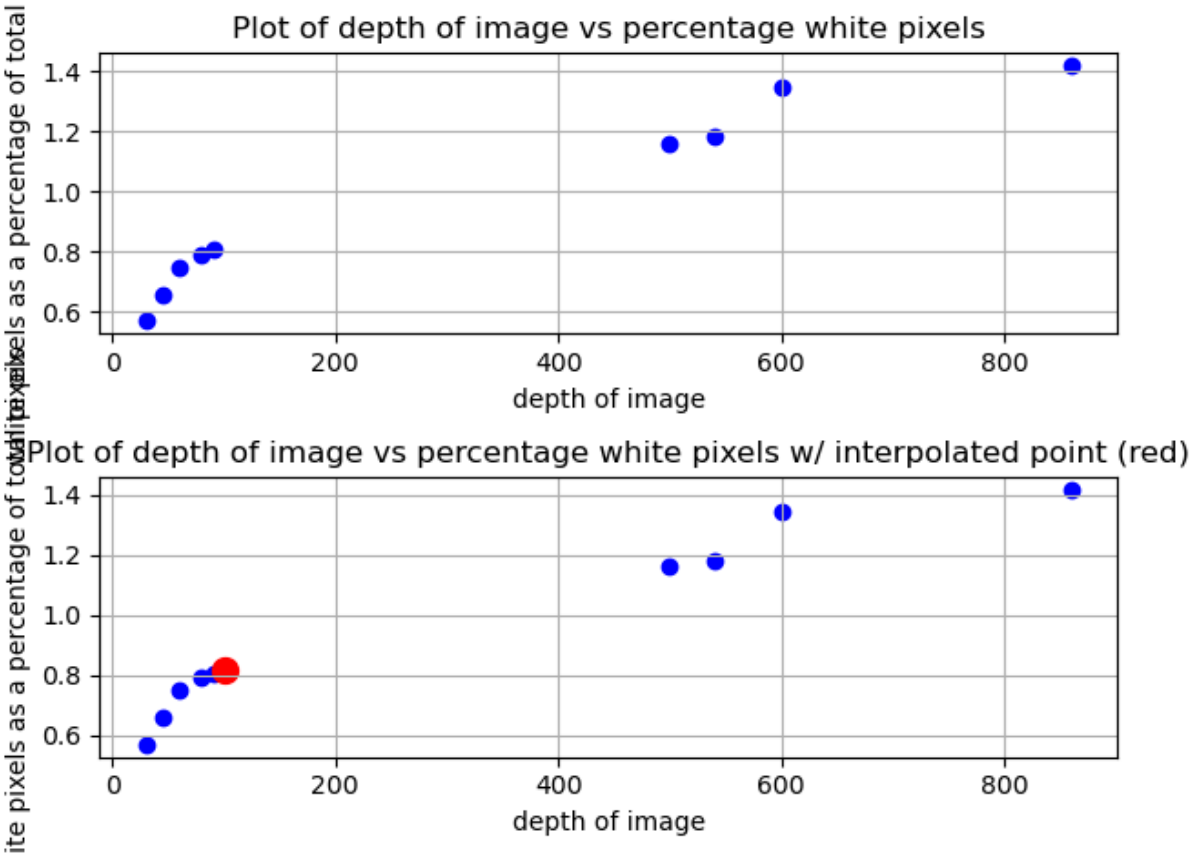
```
'''Module 2: count black and white pixels in a .jpg and extrapolate points''

from termcolor import colored
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
import pandas as pd

# Load the images you want to analyze

filenames = [
    r"MASK_Sk658 Llobe ch010017.jpg",
    r"MASK_Sk658 Llobe ch010018.jpg",
    r"MASK_Sk658 Llobe ch010019.jpg",
    r"MASK_Sk658 Llobe ch010021.jpg",
    r"MASK_Sk658 Llobe ch010022.jpg",
    r"MASK_SK658 Slobe ch010159.jpg",
    r"MASK_SK658 Slobe ch010048.jpg",
    r"MASK_Sk658 Llobe ch010032.jpg",
    r"MASK_Sk658 Llobe ch010024.jpg"
            ]

# Enter the depth of each image (in the same order that the images are liste

depths = [
    45,
    90,
    60,
    30,
    80,
    860,
    540,
    500,
    600
    ]

# Make the lists that will be used

images = []
white_counts = []
black_counts = []
white_percents = []

# Build the list of all the images you are analyzing

for filename in filenames:
    img = cv2.imread(filename, 0)
    images.append(img)

# For each image (until the end of the list of images), calculate the number

for x in range(len(filenames)):
    _, binary = cv2.threshold(images[x], 127, 255, cv2.THRESH_BINARY)
```

```
    white = np.sum(binary == 255)
    black = np.sum(binary == 0)

    white_counts.append(white)
    black_counts.append(black)

# Print the number of white and black pixels in each image.

print(colored("Counts of pixel by color in each image", "yellow"))
for x in range(len(filenames)):
    print(colored(f"White pixels in image {x}: {white_counts[x]}", "white"))
    print(colored(f"Black pixels in image {x}: {black_counts[x]}", "black"))
    print()

# Calculate the percentage of pixels in each image that are white and make a

for x in range(len(filenames)):
    white_percent = (100 * (white_counts[x] / (black_counts[x] + white_count
    white_percents.append(white_percent)

# Print the filename (on one line in red font), and below that line print th

print(colored("Percent white px:", "yellow"))
for x in range(len(filenames)):
    print(colored(f'{filenames[x]}:',"red"))
    print(f'{white_percents[x]}% White | Depth: {depths[x]} microns')
    print()

'''Write your data to a .csv file'''

# Create a DataFrame that includes the filenames, depths, and percentage of
df = pd.DataFrame({
    'Filenames': filenames,
    'Depths': depths,
    'White percents': white_percents
})

# Write that DataFrame to a .csv file

df.to_csv('Percent_White_Pixels.csv', index=False)

print("CSV file 'Percent_White_Pixels.csv' has been created.")

'''the .csv writing subroutine ends here'''

# Interpolate a point: given a depth, find the corresponding white pixel per

interpolate_depth = float(input(colored("Enter the depth at which you want t

x = depths
y = white_percents

i = interp1d(x, y, kind='quadratic')  # You can also use 'quadratic', 'cubic
interpolate_point = i(interpolate_depth)
print(colored(f'The interpolated point is at the x-coordinate {interpolate_c
```

```
depths_i = depths[:]
depths_i.append(interpolate_depth)
white_percents_i = white_percents[:]
white_percents_i.append(interpolate_point)

# make two plots: one that doesn't contain the interpolated point, just the
fig, axs = plt.subplots(2, 1)

axs[0].scatter(depths, white_percents, marker='o', linestyle='-', color='blu
axs[0].set_title('Plot of depth of image vs percentage white pixels')
axs[0].set_xlabel('depth of image')
axs[0].set_ylabel('white pixels as a percentage of total pixels')
axs[0].grid(True)


axs[1].scatter(depths_i, white_percents_i, marker='o', linestyle='-', color=
axs[1].set_title('Plot of depth of image vs percentage white pixels w/ inter
axs[1].set_xlabel('depth of image')
axs[1].set_ylabel('white pixels as a percentage of total pixels')
axs[1].grid(True)
axs[1].scatter(depths_i[len(depths_i)-1], white_percents_i[len(white_percent


# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```

```
Counts of pixel by color in each image
White pixels in image 0: 27561
Black pixels in image 0: 4166743

White pixels in image 1: 33746
Black pixels in image 1: 4160558

White pixels in image 2: 31331
Black pixels in image 2: 4162973

White pixels in image 3: 23900
Black pixels in image 3: 4170404

White pixels in image 4: 33151
Black pixels in image 4: 4161153

White pixels in image 5: 59426
Black pixels in image 5: 4134878

White pixels in image 6: 49491
Black pixels in image 6: 4144813

White pixels in image 7: 48667
Black pixels in image 7: 4145637

White pixels in image 8: 56360
Black pixels in image 8: 4137944

Percent white px:
MASK_Sk658 Llobe ch010017.jpg:
0.6571054458618164% White | Depth: 45 microns

MASK_Sk658 Llobe ch010018.jpg:
0.8045673370361328% White | Depth: 90 microns

MASK_Sk658 Llobe ch010019.jpg:
0.7469892501831055% White | Depth: 60 microns

MASK_Sk658 Llobe ch010021.jpg:
0.5698204040527344% White | Depth: 30 microns

MASK_Sk658 Llobe ch010022.jpg:
0.7903814315795898% White | Depth: 80 microns

MASK_SK658 Slobe ch010159.jpg:
1.4168262481689453% White | Depth: 860 microns

MASK_SK658 Slobe ch010048.jpg:
1.179957389831543% White | Depth: 540 microns

MASK_Sk658 Llobe ch010032.jpg:
1.1603116989135742% White | Depth: 500 microns

MASK_Sk658 Llobe ch010024.jpg:
1.3437271118164062% White | Depth: 600 microns
```

```
CSV file 'Percent_White_Pixels.csv' has been created.
The interpolated point is at the x-coordinate 100.0 and y-coordinate 0.81854
40859517384.
```



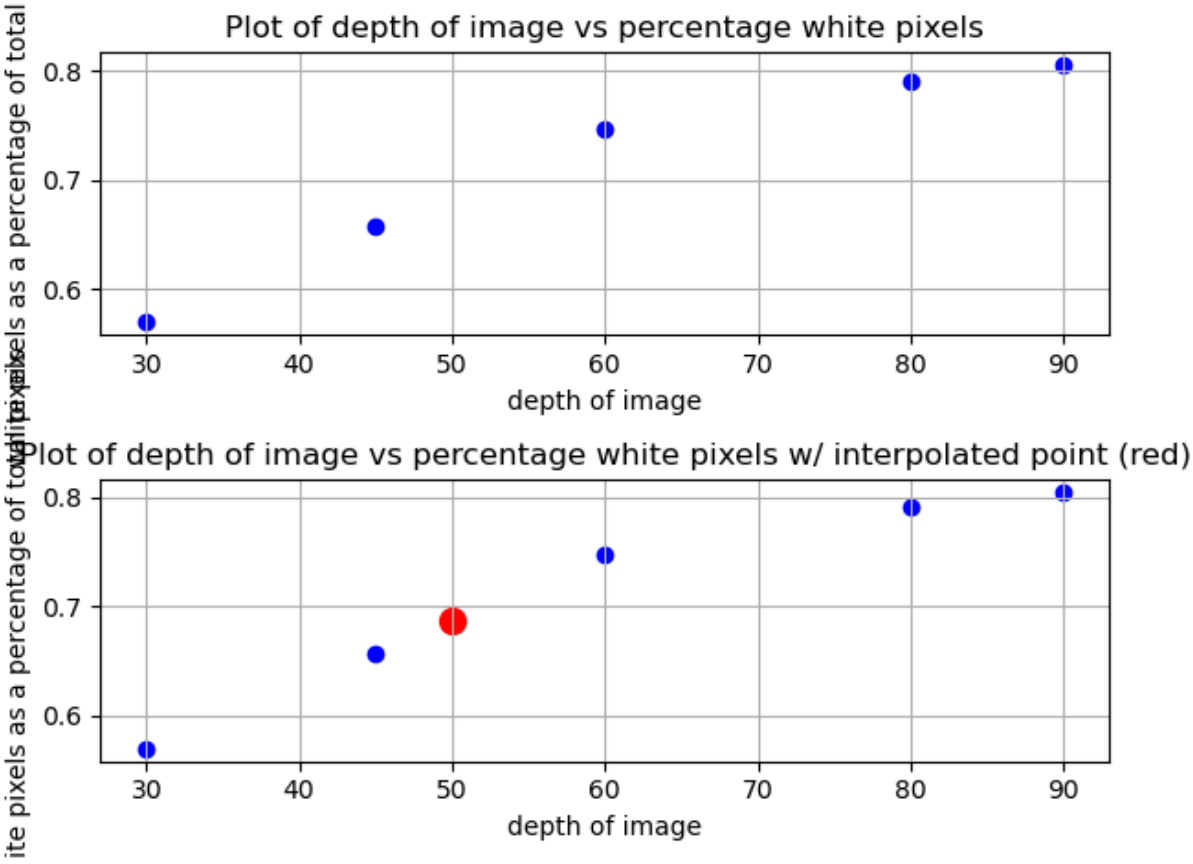
Here, the same procedure using slightly different images is shown:

In [ ]:
```
from termcolor import colored
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
import pandas as pd
# Load the images you want to analyze

filenames = [
    r"MASK_Sk658 Llobe ch010017.jpg",
    r"MASK_Sk658 Llobe ch010018.jpg",
    r"MASK_Sk658 Llobe ch010019.jpg",
    r"MASK_Sk658 Llobe ch010021.jpg",
    r"MASK_Sk658 Llobe ch010022.jpg"
            ]

# Enter the depth of each image (in the same order that the images are liste

depths = [
    45,
    90,
    60,
    30,
    80
```

```
    ]

# Make the lists that will be used

images = []
white_counts = []
black_counts = []
white_percents = []

# Build the list of all the images you are analyzing

for filename in filenames:
    img = cv2.imread(filename, 0)
    images.append(img)

# For each image (until the end of the list of images), calculate the number

for x in range(len(filenames)):
    _, binary = cv2.threshold(images[x], 127, 255, cv2.THRESH_BINARY)

    white = np.sum(binary == 255)
    black = np.sum(binary == 0)

    white_counts.append(white)
    black_counts.append(black)

# Print the number of white and black pixels in each image.

print(colored("Counts of pixel by color in each image", "yellow"))
for x in range(len(filenames)):
    print(colored(f"White pixels in image {x}: {white_counts[x]}", "white"))
    print(colored(f"Black pixels in image {x}: {black_counts[x]}", "black"))
    print()

# Calculate the percentage of pixels in each image that are white and make a

for x in range(len(filenames)):
    white_percent = (100 * (white_counts[x] / (black_counts[x] + white_count
    white_percents.append(white_percent)

# Print the filename (on one line in red font), and below that line print th

print(colored("Percent white px:", "yellow"))
for x in range(len(filenames)):
    print(colored(f'{filenames[x]}:',"red"))
    print(f'{white_percents[x]}% White | Depth: {depths[x]} microns')
    print()

'''Write your data to a .csv file'''

# Create a DataFrame that includes the filenames, depths, and percentage of
df = pd.DataFrame({
    'Filenames': filenames,
    'Depths': depths,
    'White percents': white_percents
})
```

```
# Write that DataFrame to a .csv file

df.to_csv('Percent_White_Pixels_2.csv', index=False)

print("CSV file 'Percent_White_Pixels_2.csv' has been created.")

'''the .csv writing subroutine ends here'''

# Interpolate a point: given a depth, find the corresponding white pixel per

interpolate_depth = float(input(colored("Enter the depth at which you want t

x = depths
y = white_percents

i = interp1d(x, y, kind='linear')  # You can also use 'quadratic', 'cubic',
interpolate_point = i(interpolate_depth)
print(colored(f'The interpolated point is at the x-coordinate {interpolate_d

depths_i = depths[:]
depths_i.append(interpolate_depth)
white_percents_i = white_percents[:]
white_percents_i.append(interpolate_point)

# make two plots: one that doesn't contain the interpolated point, just the
fig, axs = plt.subplots(2, 1)

axs[0].scatter(depths, white_percents, marker='o', linestyle='-', color='blu
axs[0].set_title('Plot of depth of image vs percentage white pixels')
axs[0].set_xlabel('depth of image')
axs[0].set_ylabel('white pixels as a percentage of total pixels')
axs[0].grid(True)


axs[1].scatter(depths_i, white_percents_i, marker='o', linestyle='-', color=
axs[1].set_title('Plot of depth of image vs percentage white pixels w/ inter
axs[1].set_xlabel('depth of image')
axs[1].set_ylabel('white pixels as a percentage of total pixels')
axs[1].grid(True)
axs[1].scatter(depths_i[len(depths_i)-1], white_percents_i[len(white_percent


# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```

```
Counts of pixel by color in each image
White pixels in image 0: 27561
Black pixels in image 0: 4166743

White pixels in image 1: 33746
Black pixels in image 1: 4160558

White pixels in image 2: 31331
Black pixels in image 2: 4162973

White pixels in image 3: 23900
Black pixels in image 3: 4170404

White pixels in image 4: 33151
Black pixels in image 4: 4161153

Percent white px:
MASK_Sk658 Llobe ch010017.jpg:
0.6571054458618164% White | Depth: 45 microns

MASK_Sk658 Llobe ch010018.jpg:
0.8045673370361328% White | Depth: 90 microns

MASK_Sk658 Llobe ch010019.jpg:
0.7469892501831055% White | Depth: 60 microns

MASK_Sk658 Llobe ch010021.jpg:
0.5698204040527344% White | Depth: 30 microns

MASK_Sk658 Llobe ch010022.jpg:
0.7903814315795898% White | Depth: 80 microns

CSV file 'Percent_White_Pixels_2.csv' has been created.
The interpolated point is at the x-coordinate 50.0 and y-coordinate 0.687066
7139689127.
```



Verify and validate your analysis:

- " The end-stage of this process—when the cysts increase their sizes—radiologically corresponds to honeycombing that initially appears in the subpleural areas of the lower lobes."
  - This article finds that there is more honeycombing in the lung in the lower lobes of the lung than the upper lobes
  - This validates our analysis where the graphs display the relationship that as the depth of the image increases, the percentage of white pixels as a percentage of all pixels increases.

https://pmc.ncbi.nlm.nih.gov/articles/PMC6164303/#sec2-medsci-06-00073

- lung fibrosis is the most prominent in peripheral and basal locations of the lung

https://pmc.ncbi.nlm.nih.gov/articles/PMC9489199/

The following code takes figure 2C in the following article: https://pmc.ncbi.nlm.nih.gov/articles/PMC5856260/#_ad93_ and analyzes the number of green pixels (fibrotic tissue) in comparison to all the other colored pixels (alveolar tissue (blue), bronchi (red), background (yellow)). The yellow pixels are not included in the calculation because it is background data and does not involve the lungs.

In [ ]:

```
'''Module: Count green pixels vs all other pixels in images'''

from termcolor import colored
import cv2
import numpy as np
import pandas as pd

# Load the images you want to analyze
filenames = [
    r"new_img.png"
]

# Make the lists that will be used
images = []
green_counts = []
other_counts = []
green_percents = []

# Build the list of all the images you are analyzing
for filename in filenames:
    img = cv2.imread(filename)  # Read as color image
    images.append(img)

# For each image, count green pixels vs other pixels
for x in range(len(filenames)):
    # Convert BGR to HSV color space (OpenCV uses BGR by default)
    hsv = cv2.cvtColor(images[x], cv2.COLOR_BGR2HSV)

    # Define range for green color in HSV
    # Hue for green is roughly 40-80, Saturation 40-255, Value 40-255
    lower_green = np.array([40, 40, 40])
    upper_green = np.array([80, 255, 255])

    lower_yellow = np.array([20, 100, 100])
    upper_yellow = np.array([30, 255, 255])

    # Create mask for green pixels
    green_mask = cv2.inRange(hsv, lower_green, upper_green)
    yellow_mask = cv2.inRange(hsv,lower_yellow, upper_yellow)

    # Count green and non-green pixels
    green = np.sum(green_mask == 255)
    yellow = np.sum(yellow_mask == 255)
    total_pixels = images[x].shape[0] * images[x].shape[1]
    other = total_pixels - green - yellow

    green_counts.append(green)
    other_counts.append(other)

# Print the number of green and other pixels in each image
print(colored("Counts of pixels by color in each image", "yellow"))
for x in range(len(filenames)):
    print(colored(f"Green pixels in image {x}: {green_counts[x]}", "green"))
    print(f"Other pixels in image not including yellow {x}: {other_counts[x]
    print()
```

```
# Calculate the percentage of pixels in each image that are green
for x in range(len(filenames)):
    green_percent = (100 * (green_counts[x] / (green_counts[x] + other_count
    green_percents.append(green_percent)

# Print the filename and percent green pixels
print(colored("Percent green pixels:", "yellow"))
for x in range(len(filenames)):
    print(colored(f'{filenames[x]}:', "red"))
    print(f'{green_percents[x]:.2f}% Green')
    print()

# Create a DataFrame that includes the filenames and percentage of green pix
df = pd.DataFrame({
    'Filenames': filenames,
    'Green percents': green_percents
})

# Write that DataFrame to a .csv file
df.to_csv('Percent_Green_Pixels.csv', index=False)

print(colored("CSV file 'Percent_Green_Pixels.csv' has been created.", "gree
```

```
Counts of pixels by color in each image
Green pixels in image 0: 74621
Other pixels in image not including yellow 0: 201560

Percent green pixels:
new_img.png:
27.02% Green

CSV file 'Percent_Green_Pixels.csv' has been created.
```

The following code will take the same graph and interpolation as above and calculate the actual percentage of white pixels at depth 100.

In [ ]:

```
'''Module 2: count black and white pixels in a .jpg and extrapolate points''

from termcolor import colored
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
import pandas as pd

# Load the images you want to analyze

filenames = [
    r"MASK_Sk658 Llobe ch010017.jpg",
    r"MASK_Sk658 Llobe ch010018.jpg",
    r"MASK_Sk658 Llobe ch010019.jpg",
    r"MASK_Sk658 Llobe ch010021.jpg",
    r"MASK_Sk658 Llobe ch010022.jpg",
    r"MASK_SK658 Slobe ch010159.jpg",
    r"MASK_SK658 Slobe ch010048.jpg",
```

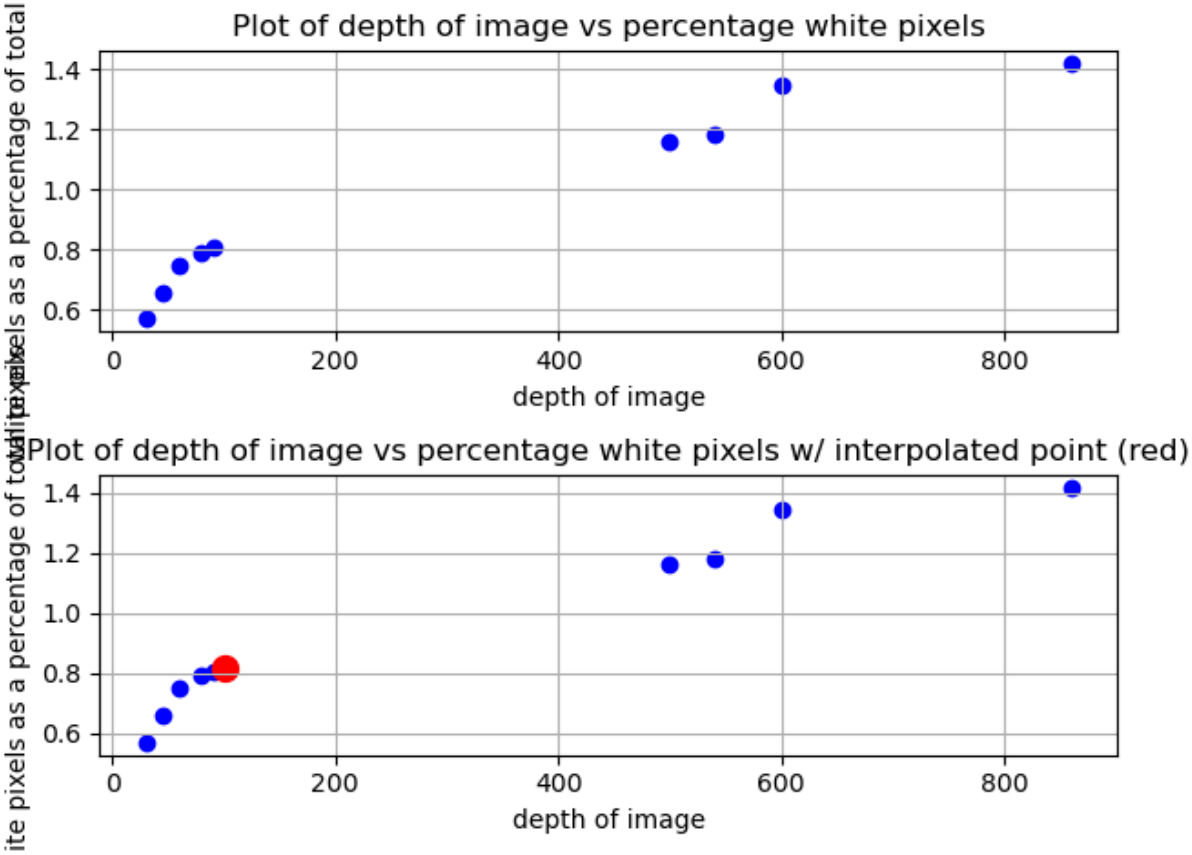
```
    r"MASK_Sk658 Llobe ch010032.jpg",
    r"MASK_Sk658 Llobe ch010024.jpg"
            ]

validation = r"MASK_Sk658 Llobe ch010023.jpg"

# Enter the depth of each image (in the same order that the images are liste

depths = [
    45,
    90,
    60,
    30,
    80,
    860,
    540,
    500,
    600
    ]

# Make the lists that will be used

images = []
white_counts = []
black_counts = []
white_percents = []

# Build the list of all the images you are analyzing

for filename in filenames:
    img = cv2.imread(filename, 0)
    images.append(img)

imgVal = cv2.imread(validation,0)

# For each image (until the end of the list of images), calculate the number

for x in range(len(filenames)):
    _, binary = cv2.threshold(images[x], 127, 255, cv2.THRESH_BINARY)

    white = np.sum(binary == 255)
    black = np.sum(binary == 0)

    white_counts.append(white)
    black_counts.append(black)


_, binary = cv2.threshold(imgVal, 127, 255, cv2.THRESH_BINARY)
whiteVal = np.sum(binary == 255)
blackVal = np.sum(binary == 0)

# Print the number of white and black pixels in each image.

print(colored("Counts of pixel by color in each image", "yellow"))
for x in range(len(filenames)):
    print(colored(f"White pixels in image {x}: {white_counts[x]}", "white"))
```

```
    print(colored(f"Black pixels in image {x}: {black_counts[x]}", "black"))
    print()

print(colored(f"White pixels in image {x}: {whiteVal}", "white"))

# Calculate the percentage of pixels in each image that are white and make a

for x in range(len(filenames)):
    white_percent = (100 * (white_counts[x] / (black_counts[x] + white_count
    white_percents.append(white_percent)

# Print the filename (on one line in red font), and below that line print th

print(colored("Percent white px:", "yellow"))
for x in range(len(filenames)):
    print(colored(f'{filenames[x]}:',"red"))
    print(f'{white_percents[x]}% White | Depth: {depths[x]} microns')
    print()

print("image at depth 100:")
print(f'{100 * (whiteVal / (blackVal + white_counts[x]))}% White | Depth: {d
'''Write your data to a .csv file'''

# Create a DataFrame that includes the filenames, depths, and percentage of
df = pd.DataFrame({
    'Filenames': filenames,
    'Depths': depths,
    'White percents': white_percents
})

# Write that DataFrame to a .csv file

df.to_csv('Percent_White_Pixels.csv', index=False)

print("CSV file 'Percent_White_Pixels.csv' has been created.")

'''the .csv writing subroutine ends here'''

# Interpolate a point: given a depth, find the corresponding white pixel per

interpolate_depth = float(input(colored("Enter the depth at which you want t

x = depths
y = white_percents

i = interp1d(x, y, kind='quadratic')  # You can also use 'quadratic', 'cubic
interpolate_point = i(interpolate_depth)
print(colored(f'The interpolated point is at the x-coordinate {interpolate_d

depths_i = depths[:]
depths_i.append(interpolate_depth)
white_percents_i = white_percents[:]
white_percents_i.append(interpolate_point)

# make two plots: one that doesn't contain the interpolated point, just the
fig, axs = plt.subplots(2, 1)
```

```
axs[0].scatter(depths, white_percents, marker='o', linestyle='-', color='blu
axs[0].set_title('Plot of depth of image vs percentage white pixels')
axs[0].set_xlabel('depth of image')
axs[0].set_ylabel('white pixels as a percentage of total pixels')
axs[0].grid(True)


axs[1].scatter(depths_i, white_percents_i, marker='o', linestyle='-', color=
axs[1].set_title('Plot of depth of image vs percentage white pixels w/ inter
axs[1].set_xlabel('depth of image')
axs[1].set_ylabel('white pixels as a percentage of total pixels')
axs[1].grid(True)
axs[1].scatter(depths_i[len(depths_i)-1], white_percents_i[len(white_percent


# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```

```
Counts of pixel by color in each image
White pixels in image 0: 27561
Black pixels in image 0: 4166743

White pixels in image 1: 33746
Black pixels in image 1: 4160558

White pixels in image 2: 31331
Black pixels in image 2: 4162973

White pixels in image 3: 23900
Black pixels in image 3: 4170404

White pixels in image 4: 33151
Black pixels in image 4: 4161153

White pixels in image 5: 59426
Black pixels in image 5: 4134878

White pixels in image 6: 49491
Black pixels in image 6: 4144813

White pixels in image 7: 48667
Black pixels in image 7: 4145637

White pixels in image 8: 56360
Black pixels in image 8: 4137944

White pixels in image 8: 37508
Percent white px:
MASK_Sk658 Llobe ch010017.jpg:
0.6571054458618164% White | Depth: 45 microns

MASK_Sk658 Llobe ch010018.jpg:
0.8045673370361328% White | Depth: 90 microns

MASK_Sk658 Llobe ch010019.jpg:
0.7469892501831055% White | Depth: 60 microns

MASK_Sk658 Llobe ch010021.jpg:
0.5698204040527344% White | Depth: 30 microns

MASK_Sk658 Llobe ch010022.jpg:
0.7903814315795898% White | Depth: 80 microns

MASK_SK658 Slobe ch010159.jpg:
1.4168262481689453% White | Depth: 860 microns

MASK_SK658 Slobe ch010048.jpg:
1.179957389831543% White | Depth: 540 microns

MASK_Sk658 Llobe ch010032.jpg:
1.1603116989135742% White | Depth: 500 microns

MASK_Sk658 Llobe ch010024.jpg:
1.3437271118164062% White | Depth: 600 microns
```

```
image at depth 100:
0.8902589887485771% White | Depth: 600 microns
CSV file 'Percent_White_Pixels.csv' has been created.
The interpolated point is at the x-coordinate 100.0 and y-coordinate 0.81854
40859517384.
```



As can be seen in the code above, at depth 100, the interpolated model predicts the percentage of white pixels to be 0.818%. From the image gathered by the lab at depth 100, the percetange of white pixels is calculated to be 0.890%. The two percentages of the theoretrical vs actual are very close in values and do not display a significant difference.

Conclusions and Ethical Implications:

- The deeper into the lungs, the more fibrotic tissue there is.
    - our data and sources show the pattern that the higher the depth
- This could indicate that when early signs of lung fibrotic tissue is found, biopsies should be taken from lower areas of the lung to accurately assess whether or not the lung is fibrotic.
    - a biopsy in lower lobes of the lung could be harmful to the patient, and as such there needs to be less invasive methods of taking biopsies (potentially image biopsies utilizing technologies like hyperspectral imaging)

- If a biopsy is taken from upper lobes of the lung and no fibrotic tissue is found, this does not eliminate the chance that the person has IPF that is developing in the lower lungs.
- In a best case scenario, a person with IPF will be diagnosed when the fibrotic tissues are concentrated in soley the lower lobes to allow for preventative measures to be taken.
- the development of computational methods is helpful to limiting the number of animal studies that must be conducted as computational methods allow a larger range of data to be extracted.
- The bleomycin-induced mouse model provided a controlled way to replicate progressive pulmonary fibrosis, allowing image collection at multiple depths for quantitative analysis.
- By analyzing images at different lung depths, we observed a consistent increase in white pixel density (fibrotic regions) as depth increased, confirming the spatial gradient of fibrosis severity.
- This depth-dependent analysis helps validate bleomycin as a reliable preclinical model for studying early- versus late-stage IPF development.
- The findings reinforce the ethical importance of computational image analysis to minimize invasive tissue sampling in live animals and reduce the total number of mice needed for fibrosis quantification.

Limitations and Future Work:

- as Dr. Pierce Cottler mentioned during class, veins are also showing up in the images in white. in the future, we hope to develop a method to differentiate between fibrosis and the veins/arteries in the images so they are not taken into account when determining number of white pixels in an image.
- The
- in the future, we hope to use images with a higher resolution with a smaller pixel size to make the relationship between white and black pixels more accurate (with lower resolution/lower number of pixels, there can be some pixels that are half white and half black, impacting the total count of white and black pixels)
- The bleomycin model used in this study induces acute fibrosis, which may differ in progression speed and distribution from human IPF; thus, translation to clinical conditions should be interpreted cautiously.
- Because the dataset included only a limited number of depth slices per sample, subtle transitions in fibrotic density between adjacent layers may not have been fully captured.

- Variability in staining intensity and lighting across depths may affect white pixel quantification; future work should normalize illumination conditions for consistent pixel classification.
- Additional analysis could compare fibrosis progression rates across time points after bleomycin exposure (e.g., day 7, 14, 21) to correlate depth-related fibrosis with disease timeline.
- Integration of 3D reconstruction methods from multiple 2D depth images could yield a volumetric fibrosis map, allowing more accurate modeling of bleomycin-induced IPF progression.

NOTES FROM YOUR TEAM:

This is where our team is taking notes and recording activity.

In [ ]:

QUESTIONS FOR YOUR TA:

We have no questions for the teaching team