Vedlegg 1: Python Kode.

Python versjon: 3.9.13
Pycharm versjon: 2023.3.3

```
from PIL import Image # Pillow 10.2.0
import numpy as np # numpy 1.26.4
import matplotlib.pyplot as plt # matplotlib 3.8.2
from skimage import filters, measure, segmentation, morphology # scikit-image 0.22.0
from skimage.draw import polygon_perimeter # scikit-image 0.22.0
from skimage.measure import find_contours, regionprops # scikit-image 0.22.0
import pandas as pd # Pandas 2.2.0
import tkinter as tk # Del av pycharm
from tkinter import filedialog # Del av pycharm
import os
import cv2 # OpenCV 4.9.0.80
Image.MAX_IMAGE_PIXELS = None # fjerner maks grensen for størrelsen på bilder som kan kjøres
# ved store bilder vil man potensielt få problemer med for lite RAM på maskinen.

# Set Start mappe
root = tk.Tk()
root.withdraw()
initial_dir = 'C:/Users /photofolder' # Erstatt med mappen med bildene som skal analyseres.


# Åpner mappen med bildene hvor du velger hvilket bilde du vil kjøre gjennom koden
# I dette tilfelle er det kun .tiff bilder som kan velges, men det kan endres hvis man vil.

file_path = filedialog.askopenfilename(initialdir=initial_dir, title='Select TIFF file',
                                       filetypes=[('TIFF files', '*.tiff')])
# Åpner bildet inn i Python miljøet.
img = Image.open(file_path)

# Gjør bildet om til «grayscale» altså svarthvit.
img_gray = img.convert('L')

# Gjør bildet om til en array for videre behandling.
img_array = np.array(img_gray)
```

```
# Gjør bildet uskarpt med «Gaussian blur» for å fylle inn rifter og hull i partikler, Sigma verdien er
# nivået av uskarphet.
img_blurred = filters.gaussian(img_array, sigma=1)

# Setter et minimums nivå for hvor lyst en pixel må være for å ikke bli fjernet.
thresh_value = 0.19  # Juster mellom 0 – 1 ut fra behov.
img_thresh = img_blurred > thresh_value # fjerner alt under minimums nivået.

# plt.imshow(img_thresh, cmap='gray') # Fjern « # » tegnet foran koden første gangen du kjører
# koden for å se hvordan bildet ser ut etter fjerning av pixler under satt grense. Hvis det ser riktig ut
# så kan du sette tilbake «#» da vil ikke den koden bli kjørt.

labels = measure.label(img_thresh) # Finner objekter på bildet

# Minimums størrelse på objekter på bildet satt i antall pixler, dette må Justers etter behov.
min_size = 150

# Fjerner objekter under satt minimums størrelse.
img_filtered = segmentation.clear_border(labels)  # Fjerner objekter som ligger inntil kanten av
# bildet de ikke burde være med, endre hvis du trenger å ha de med.

# plt.imshow(img_filtered) # For å sjekke hvordan bildet er så langt fjern «#» for å kjøre den biten
# med koden.
# plt.show()
img_filtered_1 = morphology.remove_small_objects(img_filtered, min_size=min_size)
# plt.imshow(img_filtered_1) # For å sjekke hvordan bildet er så langt fjern «#» for å kjøre den biten
# med koden.
# plt.show()

# Finner konturer til objekter i bildet.
contours = find_contours(img_filtered_1, level=0.9)

# Lager en farge kopi av orginal bildet for å kunne legge over konturene til opptelte objekter
img_rgb = img.convert('RGB')
img_rgb_array = np.array(img_rgb)

# Tegner konturer over kopien av bildet
for contour in contours:
    rr, cc = polygon_perimeter(contour[:, 0], contour[:, 1])
    img_rgb_array[rr, cc] = [0, 255, 0]  # Set contour color to red

# Setter navn på objektene
labels_filtered = measure.label(img_filtered_1)
```

```
# Regner ut forskjellige parameter for objekter som senter posisjonen, størrelse osv.
props_filtered = regionprops(labels_filtered)

# Lager en liste over hvor alle objektene er
label_centroid_map_filtered = {prop.label: prop.centroid for prop in props_filtered}

# Tegner nummeret til alle objektene over dem
for label, centroid in label_centroid_map_filtered.items():
    # Calculate the x and y positions for the text
    x, y = int(centroid[1]), int(centroid[0])
    # Draw the text at the centroid position
    plt.text(x, y, str(label), color='red', fontsize=10, ha='center', va='center')

# Viser bildet med nummererte opptelte objekter med konturer, Fjern «#» for å kjøre den delen av
# koden
#plt.imshow(img_rgb_array)
#plt.show()

# Hvis det er første gang du kjører koden før du har finjustert nivåene dine så fjerner du «#» foran
# koden under. Den vil da spørre deg om du vil fortsette koden hvis ting ser riktig ut eller stoppe hvis
# det er noe feil.
#continue_script = input("Do you want to continue running the script? (yes/no): ")

#if continue_script.lower() != 'yes':
#    print("Script terminated by user.")
#    exit()  # Terminate the script if the user does not want to continue

# Tegner nå alt over original bildet.
for label, centroid in label_centroid_map_filtered.items():
    # Calculate the x and y positions for the text
    x, y = int(centroid[1]), int(centroid[0])
    # Draw the text at the centroid position using OpenCV
    cv2.putText(img_rgb_array, str(label), (x, y), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0, 255), 1)

img_final = Image.fromarray(img_rgb_array) # gjør om til array

# Sett mappe for å lagre bildet med konturer og nummererte objekter.
save_folder = 'C:/Users/ '
# Erstatt med din mappe

base_name = os.path.splitext(os.path.basename(file_path))[0]
# Definerer fil navnet for bildet, i dette tilfelle så blir det orginale navnet til bilder pluss
# «_Python_Count.tiff»
file_name = base_name + "_python_count.tiff"
```

```
# Lager fil adressen
full_file_path = os.path.join(save_folder, file_name)

# Lager bildet i Tiff format for å ha mest mulig informasjon
img_final.save(full_file_path, format='TIFF')

print(f"Image saved as {full_file_path}")

# regner ut egenskaper for alle opptelte objekter
props_filtered = regionprops(labels_filtered)

# Etablerer en tabell
data = []

# Trekker ut ønsket informasjon
for prop in props_filtered:
    # posisjon
    location = prop.centroid
    # Areal for objektene
    area = prop.area
    # Nummeret dens
    number = prop.label
    # Setter informasjonen inn i tabellen
    data.append([number, area, location])

# Lager en dataframe fra informasjonen i tabellen
df = pd.DataFrame(data, columns=['number', 'area', 'location'])
df['threshold_value'] = thresh_value # legger til en kolonne med min pixel verdi
df['min_size'] = min_size # Legger til minimums størrelse på objektene
# Add the original TIFF file name to the DataFrame # Legger til orginale navnet til bildet
df['file_name'] = os.path.basename(file_path)

# Sett mappe for å lagre dataframe som .CSV gil
output_folder = 'C:/Users/'
output_file = os.path.join(output_folder, 'vv.csv')

# Sjekker at mappen eksisterer
os.makedirs(output_folder, exist_ok=True)

# Sjekker om filen eksisterer og om den er tom.
# if not os.path.exists(output_file) or os.stat(output_file).st_size == 0:
    # Hvis filen ikke eksisterer så lager den filen med overskrifter
  #  df.to_csv(output_file, mode='w', index=False)
```

```
# else:
   # Hvis den eksisterer legger den bare til dataen under den andre
 #  df.to_csv(output_file, mode='a', index=False, header=False)

# Printer til slutt en melding om at dataen er lagret og printer så en oversikt i slik at du kan se antall
print("DataFrame has been successfully exported to CSV.")
print(df)
```