Stony Brook University

Surface Roughness Analysis for Atomic Force Microscope using Python

CME 502 Final Project Presentation

5/4/2023

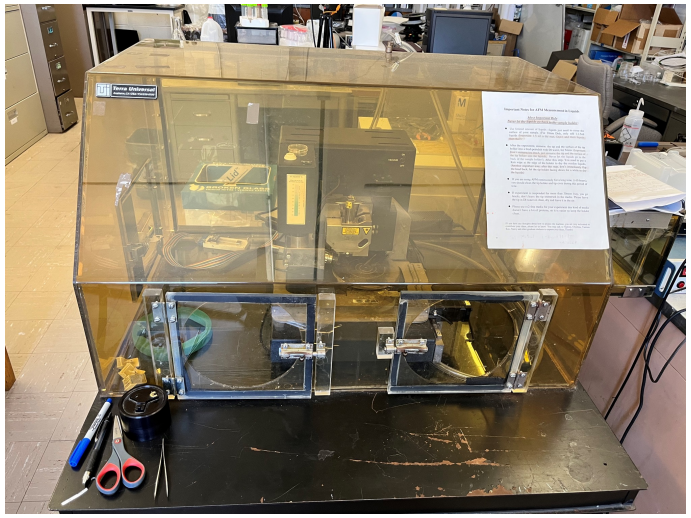
Huiting Luo

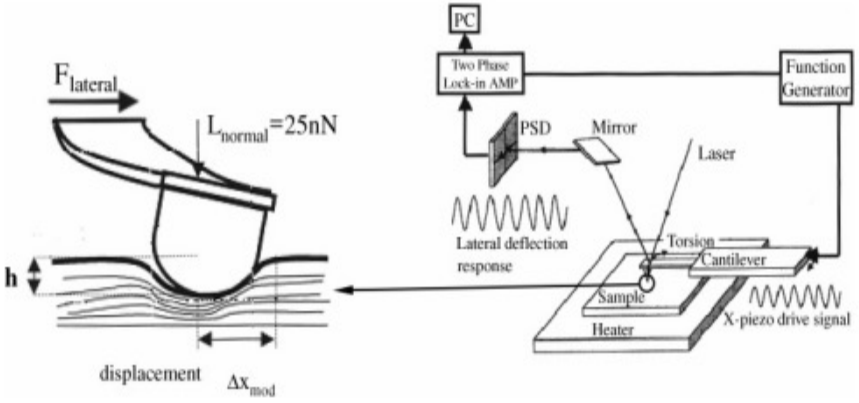
FAR BEYOND

Stony Brook University

Introduction

- Atomic force microscopy (AFM)
  - AFM is a widely used surface analysis technique for investigating nanoscale structure of materials.
  - Our lab has utilized a Bruker AFM for over 40 years and has been used in numerous research projects.
    - Explore the growth of cells on polymers scaffold
    - Examine the effects of polymer additives on perovskite grain boundary theory
    - Investigate the interaction between polymer thin films at the interface
- Surface roughness is an important parameter in characterizing the texture and visual qualities of a material's surface





FAR BEYOND

Stony Brook University

Problem

- Bruker's NanoScope Analysis is a powerful software for processing AFM data
  - But unfortunately, it was only installed on certain outdated computers in out lab
  - The software has been discontinued and is no longer available for download
  - Does not support MAC OS

System Requirements

Required
2GHz CPU, 1GB RAM
50GB Hard Drive
Windows XP, Vista, Windows 7
Minimum 800x600 monitor

Recommended
3GHz CPU, 2GB RAM
50GB Hard Drive
Windows XP, Vista, Windows 7
Minimum 1280x1024 monitor

You can download a copy of NanoScope Analysis from ftp://Release:Z3Analysis91@sboftp.bruker-nano.com. Once the software is installed, hit the F1 key or click on the blue button with the question mark on the toolbar for the online manual. A list of major new features is available under the "Help" menu – click on "What's New…"

No longer available for download

FAR BEYOND

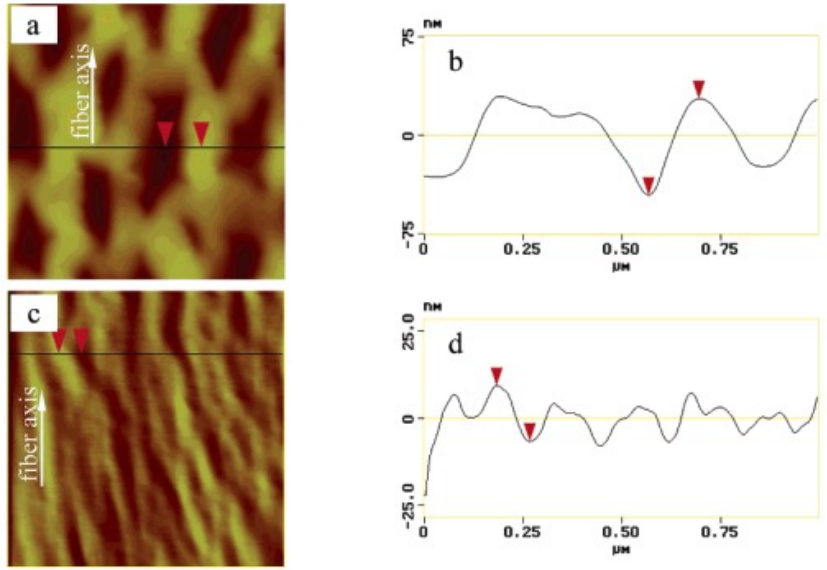
Stony Brook University

Propose solution

- The goal of this project is to develop an AFM image analysis program that can calculate the surface roughness average, Ra, using python
  - Ra is the arithmetic average of the absolute values of the profile heights over the evaluation length
  - Two modes: line Ra & area Ra
- AFM output data is a topography picture → extract the depth values from the picture
  - Calculate roughness average, Ra using eqn 1.

$$R_a = \frac{1}{L}\int_0^L |Z(x)|dx \text{ (eqn 1.)}$$

Where Z(x) is the function that describes the surface profile analyzed in terms of height (Z) and position (x) of the sample over the evaluating length "L"

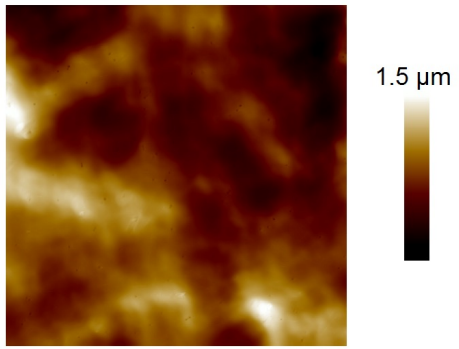


FAR BEYOND

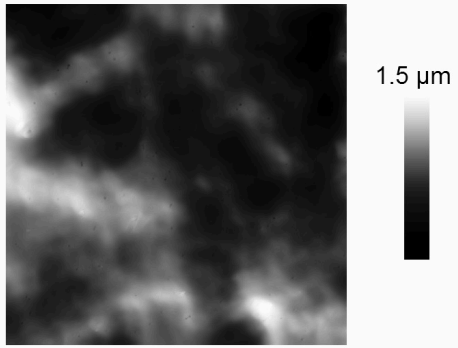
Overall workflow

Stony Brook University

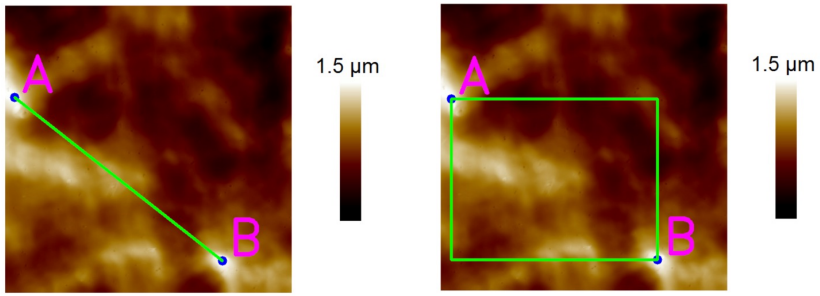
User input



Read image & Convert to grayscale

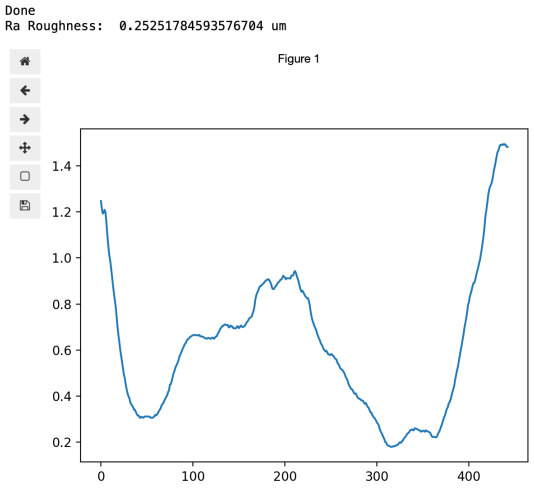


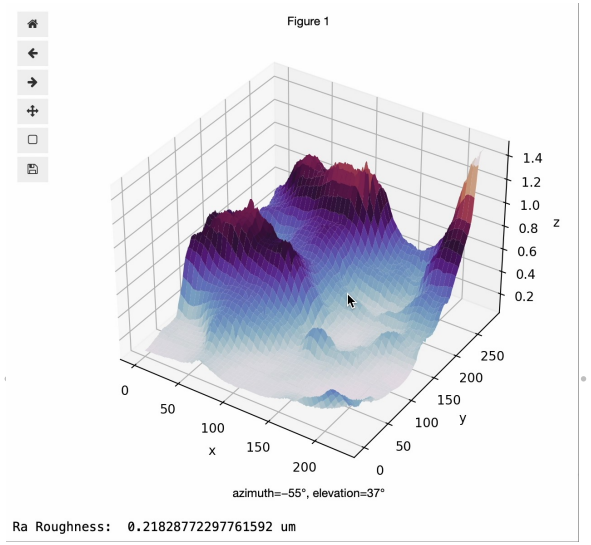
Select region of interested



Press the "return" key on keyboard

Output the Ra value and a height profile





Stony Brook University

Method – Python modules

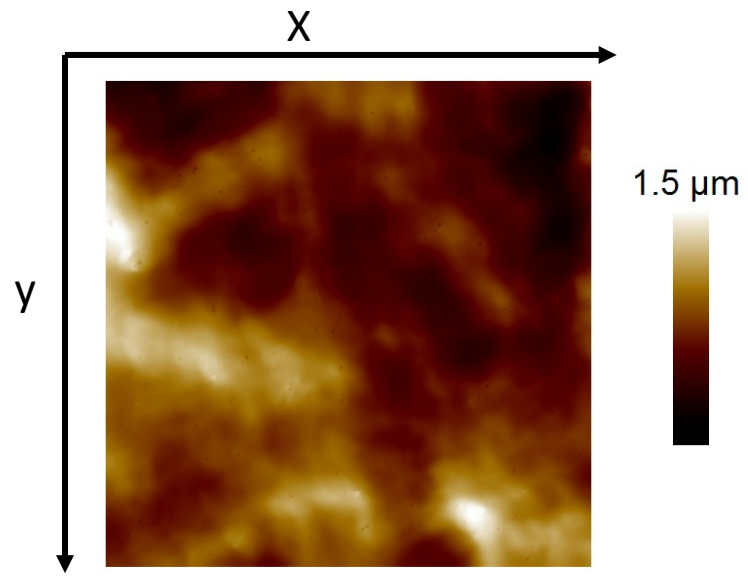
```
import cv2 # module import name for opencv
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
# generates an ipywidget that renders plots in a control
%matplotlib widget
from mpl_toolkits.mplot3d import Axes3D # 3D graphing
```

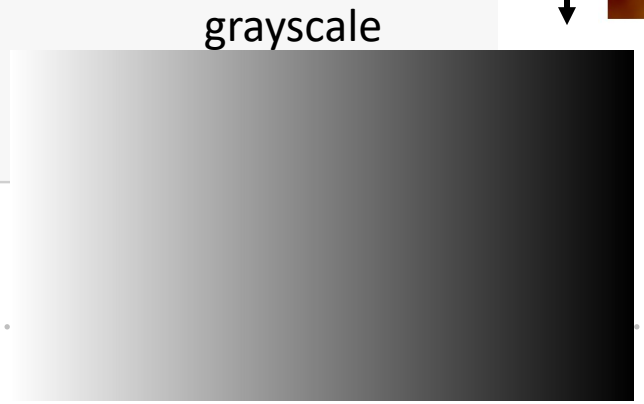
- cv2 → opencv
  - cv2.rmread → read an image
- Numpy → mathematical operations on arrays
- Interpolate → linear interpolation
- Matplotlib → 2d plotting for heights profile
- Axes3D → 3d graphing for heights profile

Stony Brook University

Method – User inputs & image read

```
########## INPUT ##########

imageName = 'pla3d0325.002_1'+'.png'

high = 1.5 # unit micron
low = 0  # unit micron

mode = 'line' #line or area


################################################
# image reading
img = cv2.imread('/Users/huitingluo/CME502_Final_Project/AFM images/'+imageName)

# convert to GrayScale image
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# ratio of scale bar and the range of grayscale values
ratio = (high-low)/(255-0)
```

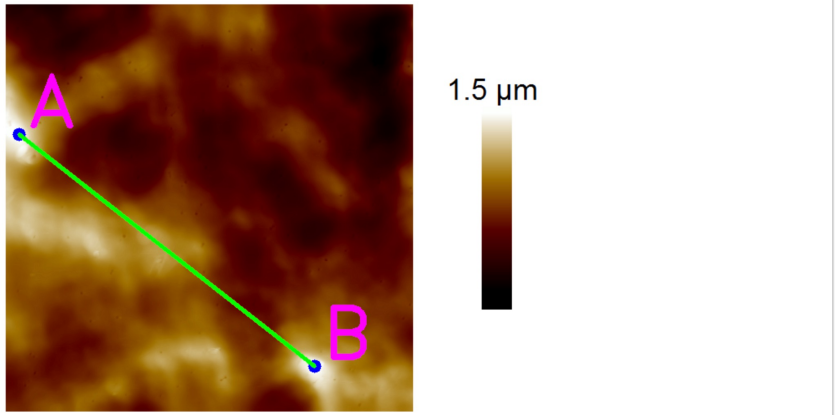


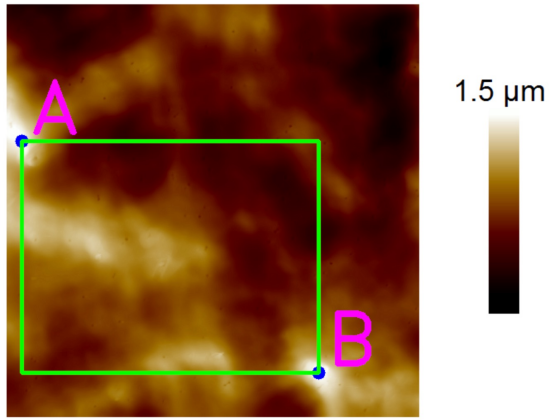


FAR BEYOND

Stony Brook University

Method – points selection



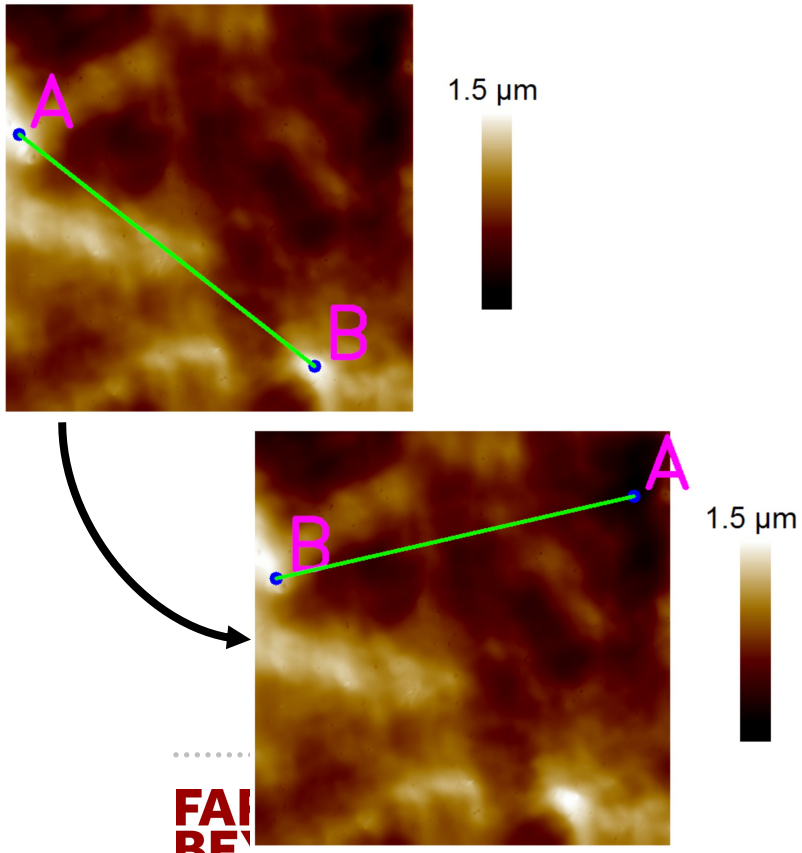



```python
# Function: Pick Starting & ending points
def Pick_two_points(image, *args):
    global counter
    counter = 0
    # Create point matrix get coordinates of mouse click on image
    point_matrix = np.zeros((2,2),int)
    clone = image.copy()
    img = image.copy()
    def mousePoints(event,x,y,flags,params):
        global counter
        # Left button click
        if event == cv2.EVENT_LBUTTONDOWN: # indicates that the left mouse button is down
            if counter <2:
                point_matrix[counter] = x,y
                counter = counter + 1

        for x in range (0,2):
            #Draw circle on image with mouse click
            cv2.circle(img,(point_matrix[x][0],point_matrix[x][1]),8,(255,0,0),cv2.FILLED)
            cv2.putText(img,
                        'A' if x==0 else 'B',
                        (point_matrix[x][0]+10,point_matrix[x][1]-10),
                        cv2.FONT_HERSHEY_DUPLEX,
                        3.0,
                        (255,0,255),
                        3
                       )
            if counter == 2:
                # assign starting & ending points
                starting_x = point_matrix[0][0]
                starting_y = point_matrix[0][1]

                ending_x = point_matrix[1][0]
                ending_y = point_matrix[1][1]


                if 'line' in args:
                    # Draw line between starting & ending points
                    ImgWithMarker=cv2.line(img, (starting_x, starting_y), (ending_x, ending_y), (0, 255, 0), 3)
                if 'area' in args:
                # Draw rectangle for area of interest
                    ImgWithMarker=cv2.rectangle(img, (starting_x, starting_y), (ending_x, ending_y), (0, 255, 0), 3)
```

FAR BEYOND

Stony Brook University

Method – points selection



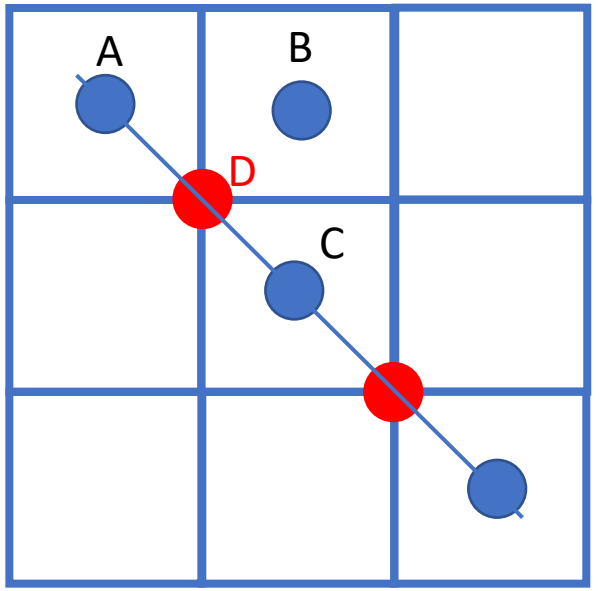

```
  # keep the image window active
    while True:
        cv2.namedWindow('AFM', cv2.WINDOW_NORMAL)
        # Mouse click event on original image
#        cv2.resizeWindow('AFM', image.shape[1], image.shape[0])
        cv2.setMouseCallback('AFM', mousePoints)
        cv2.imshow('AFM', img)
        key = cv2.waitKey(1) & 0xFF # maintain output window until user presses a key

         # if the 'r' key is pressed, <<<reset>>>> the region
        if key == ord("r") and counter ==2:
            counter = 0
            point_matrix = np.zeros((2,2),int)
            img = clone.copy()
        # if the 'enter' key is pressed, break from the loop
        elif key == ord("\r"):
            break

    print('Done')
#     cv2.destroyAllWindows()
#     cv2.waitKey(1) #???? should distroy  or not
                                          #confirm selection
    return point_matrix

#################################################################
```

Stony Brook University

Method – calculate line Ra

```
StartEndPoints  = Pick_two_points(img, mode)

start_x = StartEndPoints[0][0]
start_y = StartEndPoints[0][1]
end_x = StartEndPoints[1][0]
end_y = StartEndPoints[1][1]

d_x = abs(end_x - start_x) # d_x : number of pixels at x axis
d_y = abs(end_y - start_y) # d_y : number of pixels at y axis

L = int(np.sqrt(d_x**2+d_y**2)+1) # L: Number of pixels in the line

z_p = imgGray[min(start_y,end_y):max(start_y,end_y), min(start_x,end_x):max(start_x,end_x)]
z = high-ratio*(255-z_p)
# z = imgGray[min(start_y,end_y):max(start_y,end_y), min(start_x,end_x):max(start_x,end_x)] # start & end
#z = zg*100/255+50

if mode == 'line': # line interpolation

    f = interpolate.interp2d(np.arange(z.shape[1]), np.arange(z.shape[0]),  z, kind='linear')
    #f = interpolate.interp2d(x, y, z, kind='linear')

    Xnew = np.linspace(0,z.shape[1],L)
    Ynew = np.linspace(0,z.shape[0],L)
    Znew = f(Xnew,Ynew)
    if (start_x-end_x)*(start_y-end_y)<0:
        Z_dial = np.fliplr(Znew).diagonal()
        if start_y>end_y:
            Z_dial = Z_dial[::-1]
    else:
        Z_dial = np.diagonal(Znew)
    if start_x > end_x and start_y>end_y:
        Z_dial = Z_dial[::-1]
    plt.plot(Z_dial)

    Ra = np.sum(np.abs(Z_dial-np.mean(Z_dial)))/Z_dial.size
```

Linear interpolation



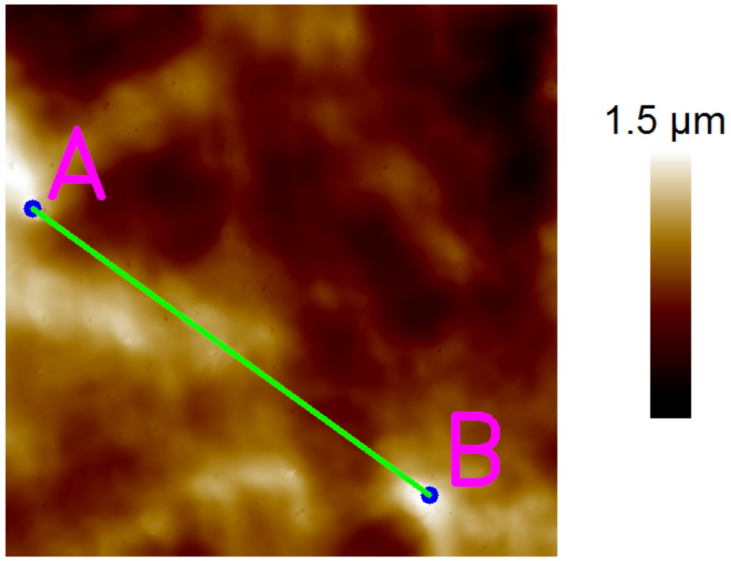
Stony Brook University

Method – calculate area Ra

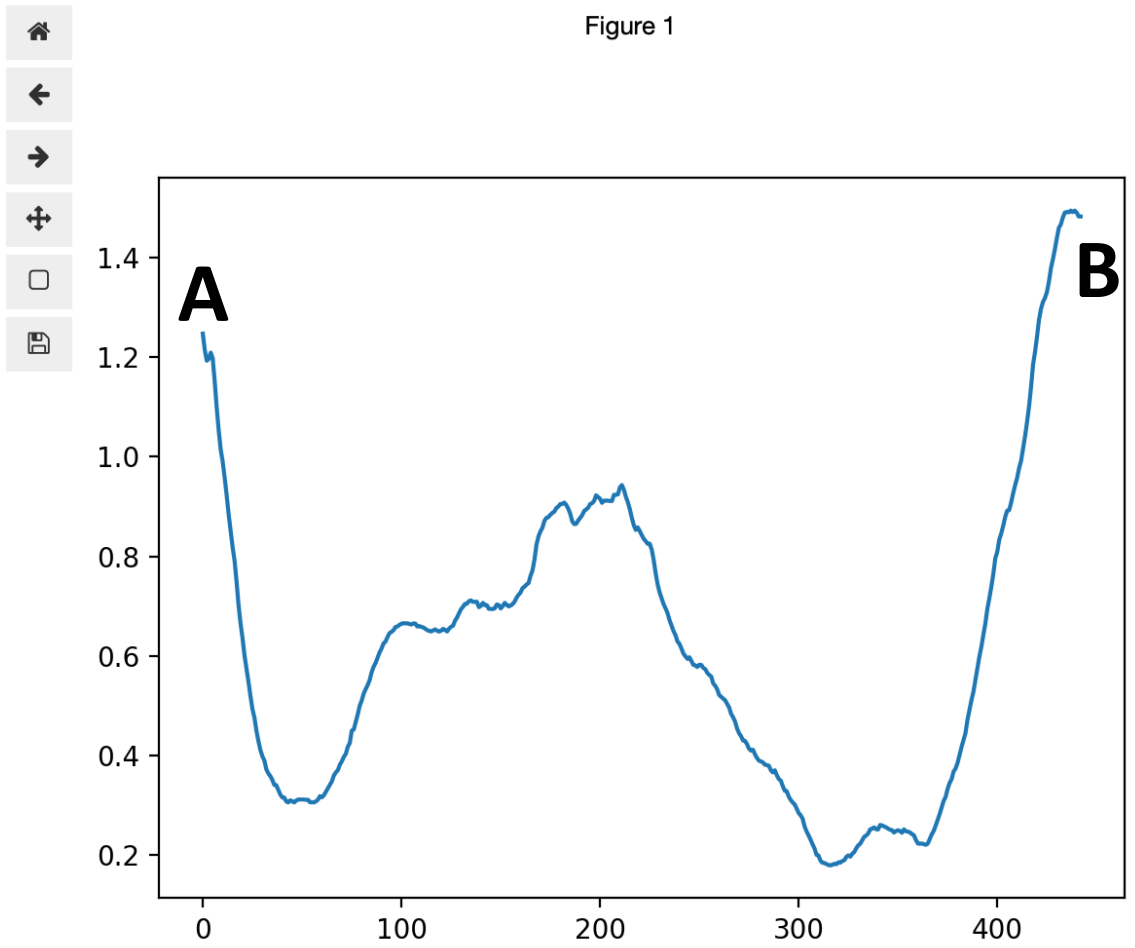
```
elif mode == 'area':
    x = np.arange(z.shape[1])
    y = np.arange(z.shape[0])
    X,Y = np.meshgrid(x,y)
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.plot_surface(X, Y, z, cmap=plt.get_cmap('twilight'))
    ax.set_title("3d")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.set_zlabel("z")
    #plt.savefig('3D.png') # save fig in the Main Folder
    plt.show()


    Ra = np.sum(np.abs(z-np.mean(z)))/z.size

print('Ra Roughness: ', Ra, 'um')
```

Stony Brook University

Result – line Ra



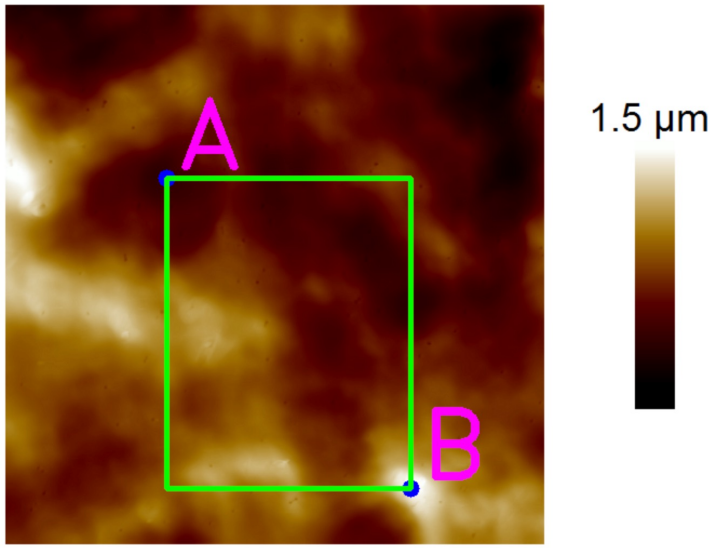

```
Done
Ra Roughness:  0.25251784593576704 um
```

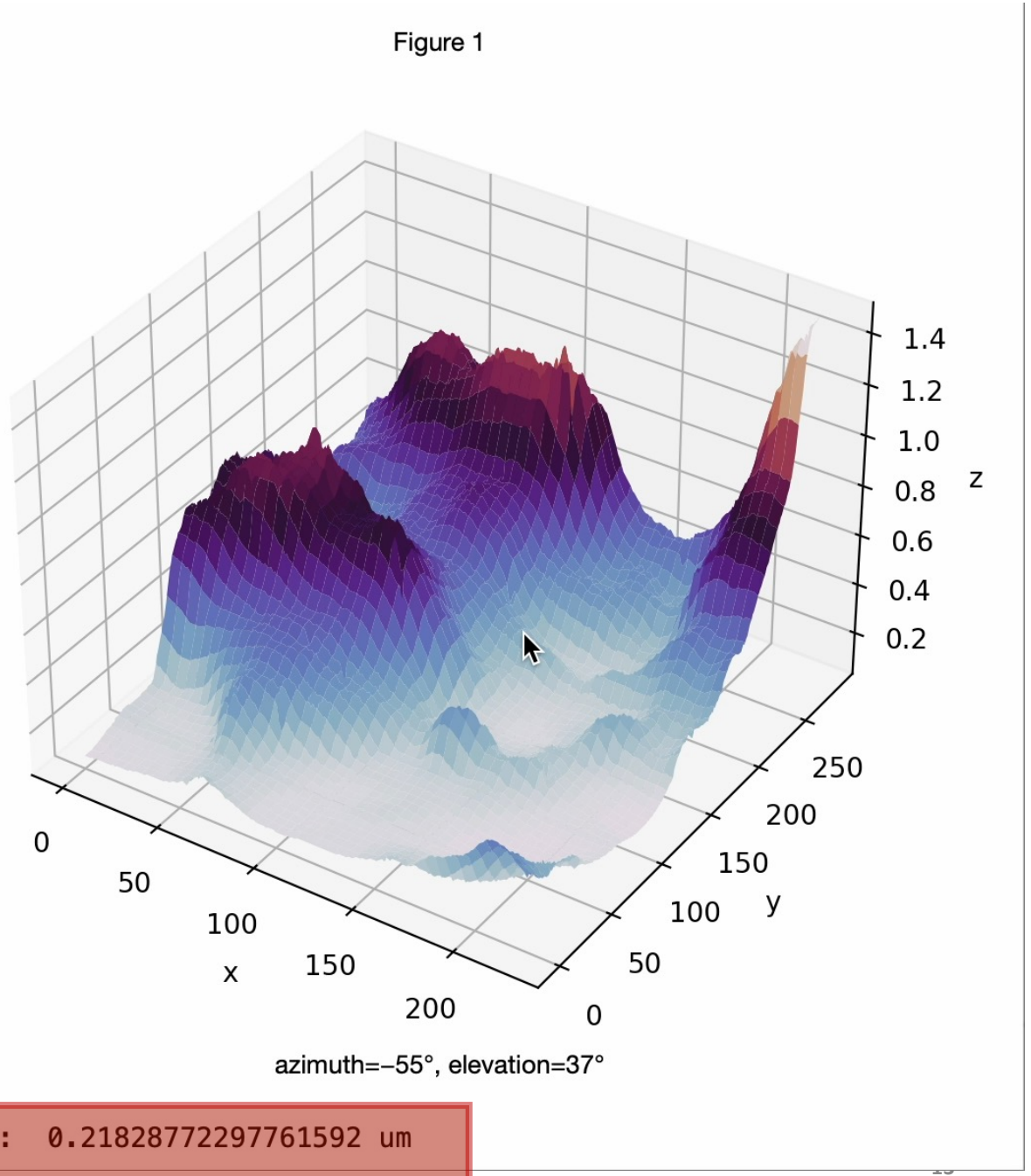


FAR BEYOND

Stony Brook University

Result – area Ra





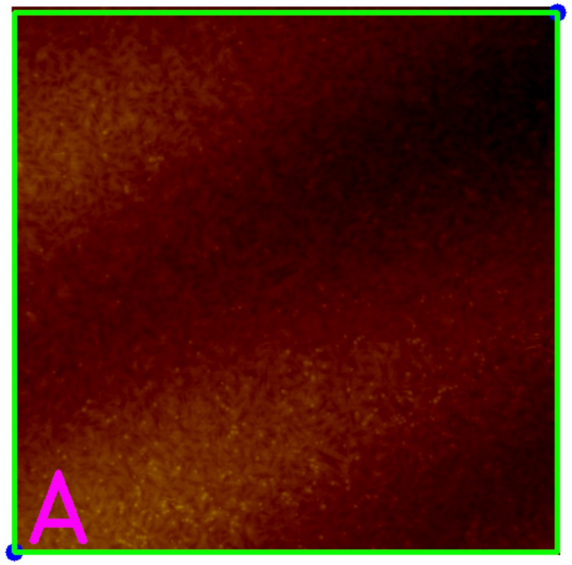
Ra Roughness: 0.21828772297761592 um

FAR BEYOND

Stony Brook University

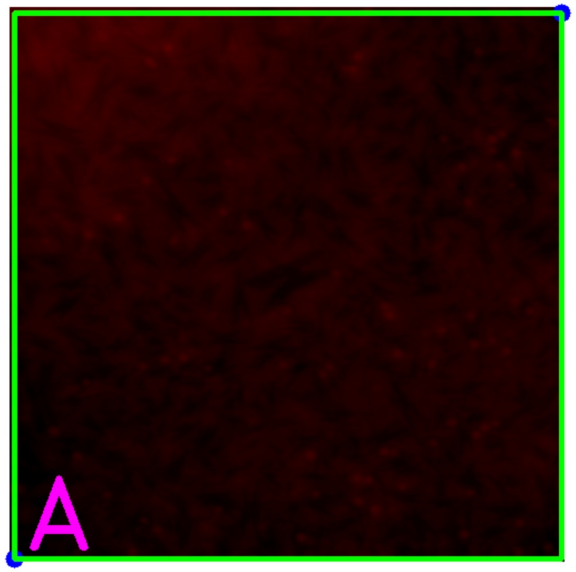
Research: effect of coated substrate for cell growth – spincast coated PLA & 3d printed PLA

Spincast sample 1



Ra = 0.0496 um

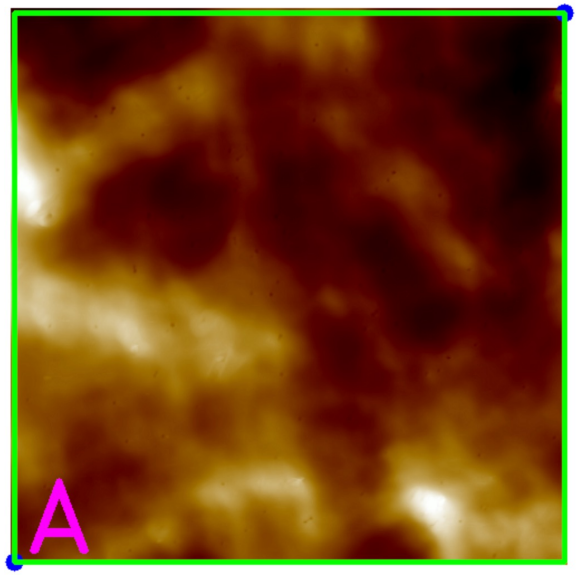
Spincast sample 2



Ra = 0.0106 um

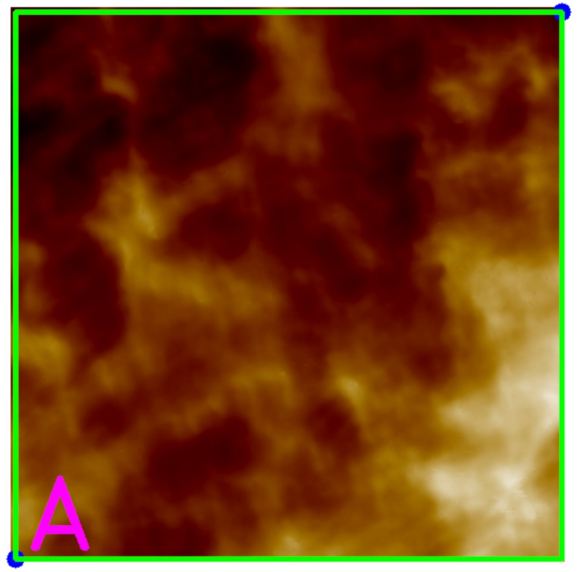
Flat

3d printed sample 1



Ra = 0.2276 um

3d printed sample 2



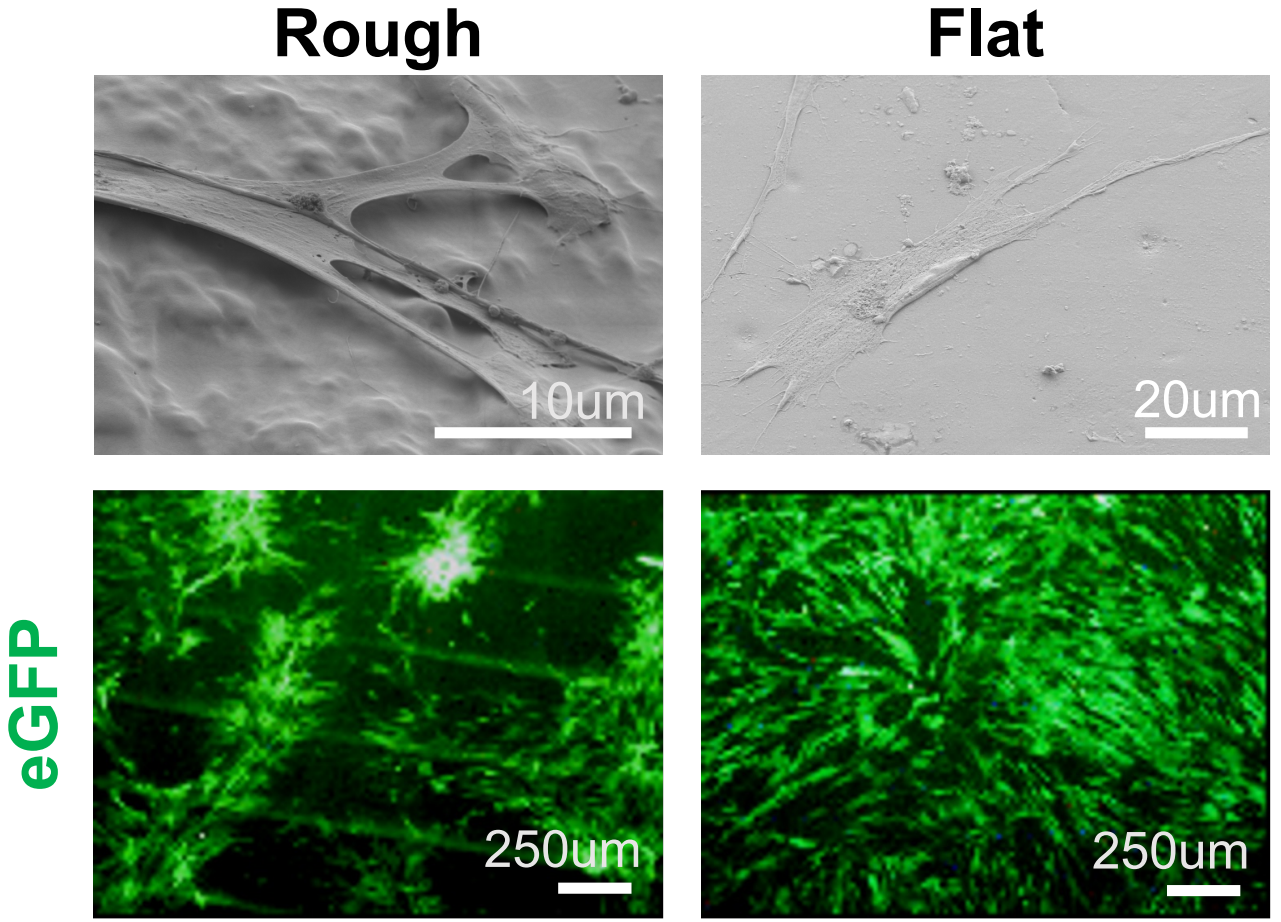
Ra = 0.4109 um

Rough

FAR BEYOND

Stony Brook University

Research: effect of coated substrate for cell growth – spincast coated PLA & 3d printed PLA



Cells grow better in a flat surface

FAR BEYOND

Stony Brook University

Conclusion

- The program of surface roughness analysis for AFM is developed
- Some advantage of the program
  - Beneficial for all group members in our lab
  - Both window and Mac users friendly
  - Free for everyone
  - Analyze the surface roughness of any region of interested
    - In nanoScope, image cropping is needed to get the area Ra of an interested area, and the cropping can only be square. Only horizontal line Ra is avaliable