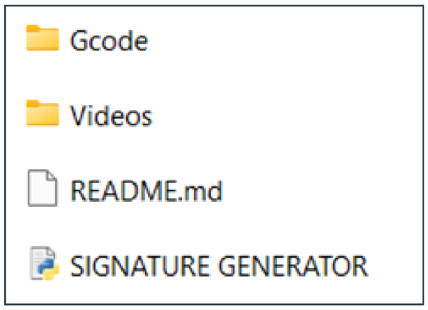
Forge-Bot Documentation

Team Members: Carlos Salazar (cds4672)
Jose Bendana (jb65669)

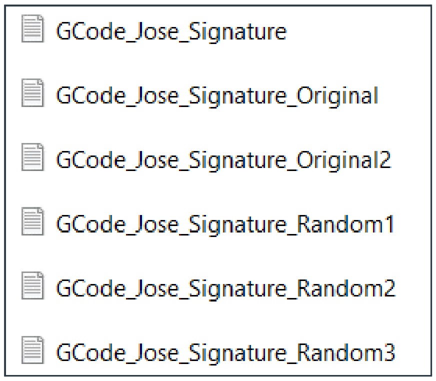
Zip File Description:

The file "Forge-Bot-Final-Project-main.zip" contains all the necessary files to be able to run the program. In it you can find two folders and a Python file.



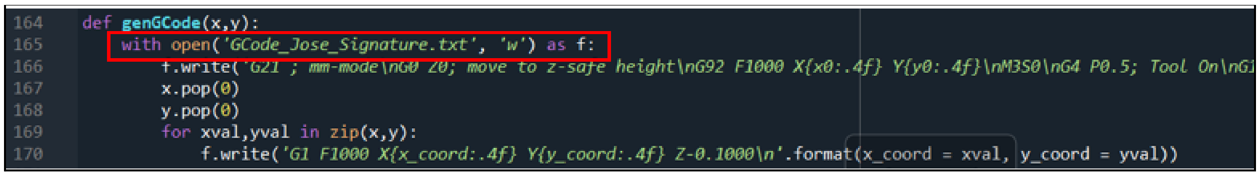
Gcode
Videos
README.md
SIGNATURE GENERATOR

The .txt files containing the G code generated by the file "SIGNATURE GENERATOR.py" will be saved in the "Gcode" folder. In the folder are several .txt files that are examples generated by the program. Every time the program is ran, a new .txt file will be generated with the name "GCode_Jose_Signature" so you must change the name of the file, otherwise it will overwrite the information.



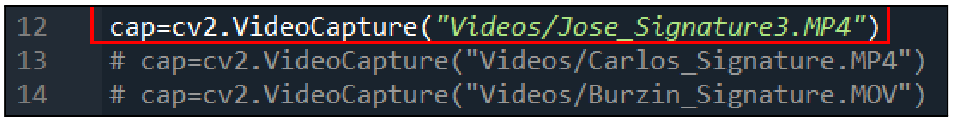
GCode_Jose_Signature
GCode_Jose_Signature_Original
GCode_Jose_Signature_Original2
GCode_Jose_Signature_Random1
GCode_Jose_Signature_Random2
GCode_Jose_Signature_Random3

The change of the file name can be done directly in the .txt file or the name can be modified in line 165 of the python file



```
164  def genGCode(x,y):
165      with open('GCode_Jose_Signature.txt', 'w') as f:
166          f.write('G21 ; mm-mode\nG0 Z0; move to z-safe height\nG92 F1000 X{x0:.4f} Y{y0:.4f}\nM3S0\nG4 P0.5; Tool On\nG1
167          x.pop(0)
168          y.pop(0)
169          for xval,yval in zip(x,y):
170              f.write('G1 F1000 X{x_coord:.4f} Y{y_coord:.4f} Z-0.1000\n'.format(x_coord = xval, y_coord = yval))
```

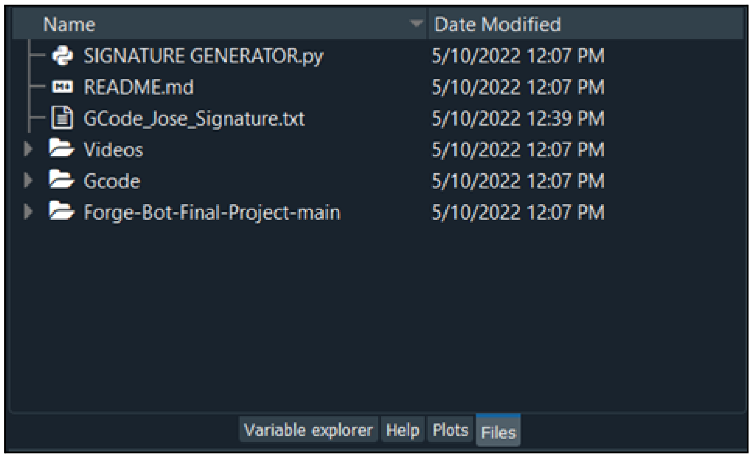
In the "Videos" folder are the videos used by the python program. Inside the folder there are 3 videos, to be able to open them in the program you must verify the address of the files and modify that address in line 12 of the code



```
12    cap=cv2.VideoCapture("Videos/Jose_Signature3.MP4")
13    # cap=cv2.VideoCapture("Videos/Carlos_Signature.MP4")
14    # cap=cv2.VideoCapture("Videos/Burzin_Signature.MOV")
```

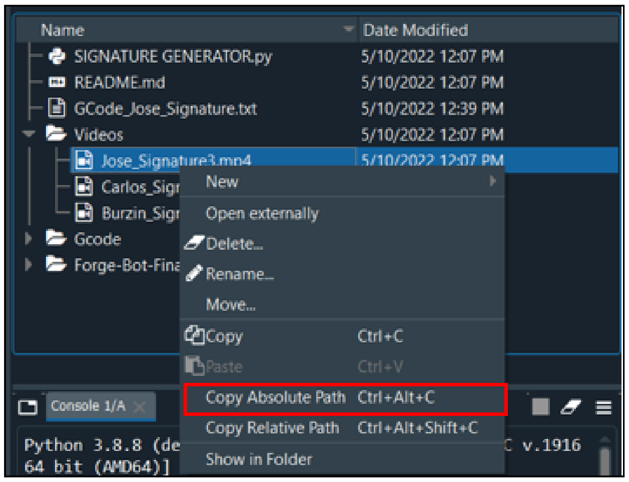
In order to verify the address of the video we can follow the following steps:

1. Go to the upper right corner of spyder and click on Files. There you find all the files and folders of the project.



Name	Date Modified
SIGNATURE GENERATOR.py	5/10/2022 12:07 PM
README.md	5/10/2022 12:07 PM
GCode_Jose_Signature.txt	5/10/2022 12:39 PM
Videos	5/10/2022 12:07 PM
Gcode	5/10/2022 12:07 PM
Forge-Bot-Final-Project-main	5/10/2022 12:07 PM

2. Expand the "Videos" folder and right click on the file whose location we want to know

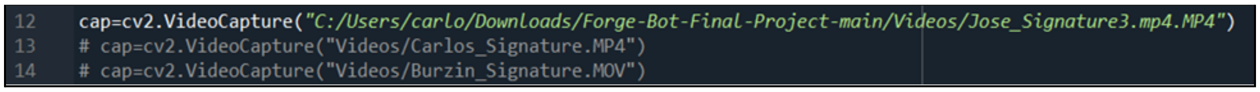


Name	Date Modified
SIGNATURE GENERATOR.py	5/10/2022 12:07 PM
README.md	5/10/2022 12:07 PM
GCode_Jose_Signature.txt	5/10/2022 12:39 PM
Videos	5/10/2022 12:07 PM
Jose_Signature3.mp4	5/10/2022 12:07 PM
Carlos_Sign	
Burzin_Sign	
Gcode	
Forge-Bot-Fina	

```
Python 3.8.8 (de                                    C v.1916
64 bit (AMD64)]
```

3. Select "Copy Absolute Path" and automatically copy us the address of the selected file.

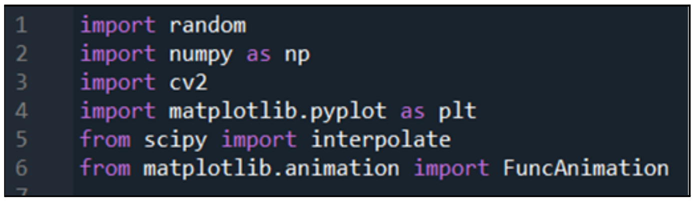
4. Finally, paste that address in line 12 of the python file.



```
12    cap=cv2.VideoCapture("C:/Users/carlo/Downloads/Forge-Bot-Final-Project-main/Videos/Jose_Signature3.mp4.MP4")
13    # cap=cv2.VideoCapture("Videos/Carlos_Signature.MP4")
14    # cap=cv2.VideoCapture("Videos/Burzin_Signature.MOV")
```

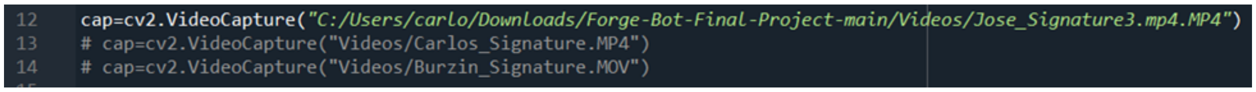
The last file that we have in the main project folder is the "SIGNATURE GENERATOR.py" file, this contains all the code that will generate a random signature based on the video signature and will also generate a G-code file to be able to control a xy cnc machine.

From line 1 to 6 the user can find the libraries that will be used in the program, the main one being "cv2" OpenCV



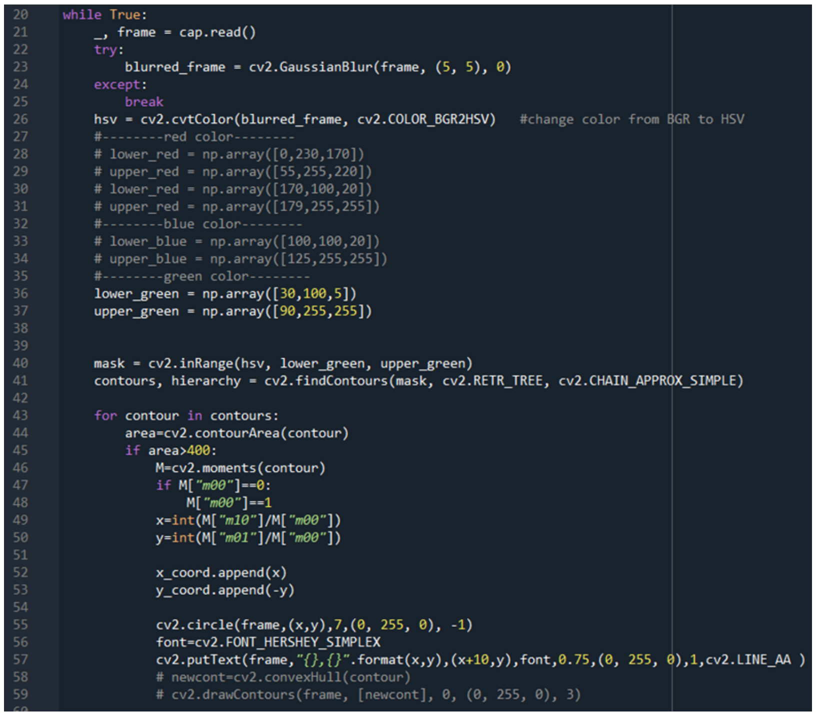
```
1   import random
2   import numpy as np
3   import cv2
4   import matplotlib.pyplot as plt
5   from scipy import interpolate
6   from matplotlib.animation import FuncAnimation
```

From lines 12 to 16 the user can specify the address of the video that he/she wishes to use in the program.



```
12  cap=cv2.VideoCapture("C:/Users/carlo/Downloads/Forge-Bot-Final-Project-main/Videos/Jose_Signature3.mp4.MP4")
13  # cap=cv2.VideoCapture("Videos/Carlos_Signature.MP4")
14  # cap=cv2.VideoCapture("Videos/Burzin_Signature.MOV")
```

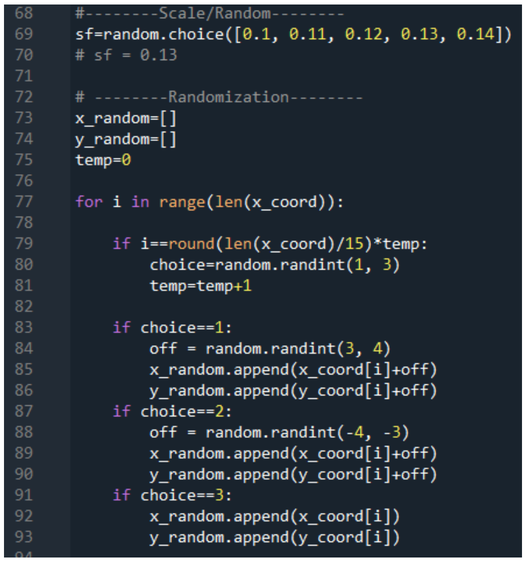
From lines 20 to 59 is a While Loop. Here several tasks are accomplished, a mask is created with a range of colors (green for this example), contours are detected, a centroid of the tracker ball object is calculated and the coordinates are stored in an array.



```
20  while True:
21      _, frame = cap.read()
22      try:
23          blurred_frame = cv2.GaussianBlur(frame, (5, 5), 0)
24      except:
25          break
26      hsv = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)   #change color from BGR to HSV
27      #--------red color--------
28      # lower_red = np.array([0,230,170])
29      # upper_red = np.array([55,255,220])
30      # lower_red = np.array([170,100,20])
31      # upper_red = np.array([179,255,255])
32      #--------blue color--------
33      # lower_blue = np.array([100,100,20])
34      # upper_blue = np.array([125,255,255])
35      #--------green color--------
36      lower_green = np.array([30,100,5])
37      upper_green = np.array([90,255,255])
38
39
40      mask = cv2.inRange(hsv, lower_green, upper_green)
41      contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
42
43      for contour in contours:
44          area=cv2.contourArea(contour)
45          if area>400:
46              M=cv2.moments(contour)
47              if M["m00"]==0:
48                  M["m00"]==1
49              x=int(M["m10"]/M["m00"])
50              y=int(M["m01"]/M["m00"])
51
52              x_coord.append(x)
53              y_coord.append(-y)
54
55              cv2.circle(frame,(x,y),7,(0, 255, 0), -1)
56              font=cv2.FONT_HERSHEY_SIMPLEX
57              cv2.putText(frame,"{},{}".format(x,y),(x+10,y),font,0.75,(0, 255, 0),1,cv2.LINE_AA )
58              # newcont=cv2.convexHull(contour)
59              # cv2.drawContours(frame, [newcont], 0, (0, 255, 0), 3)
```

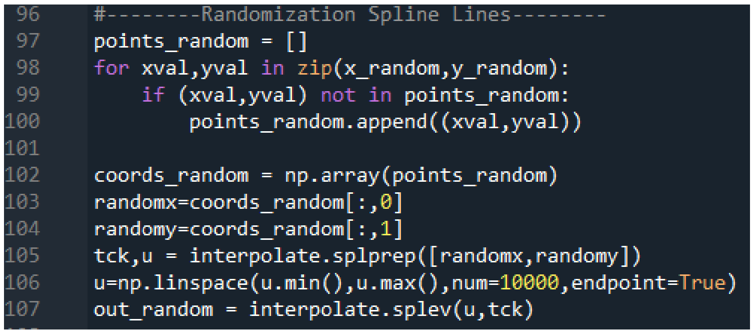
Lines 68-93 are the randomization process which includes dividing the coordinate point array into 15 segments and applying one of three randomization methods to each one. These methods are:

1. Move all points in positive x and y
2. Move all points in negative x and y
3. Move no points



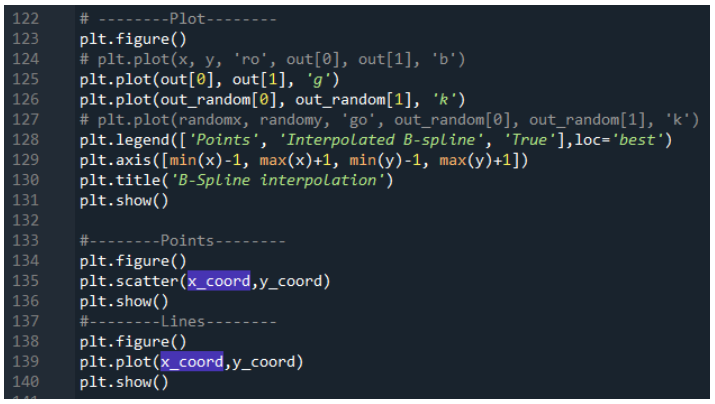
```
68  #--------Scale/Random--------
69  sf=random.choice([0.1, 0.11, 0.12, 0.13, 0.14])
70  # sf = 0.13
71
72  # --------Randomization--------
73  x_random=[]
74  y_random=[]
75  temp=0
76
77  for i in range(len(x_coord)):
78
79      if i==round(len(x_coord)/15)*temp:
80          choice=random.randint(1, 3)
81          temp=temp+1
82
83      if choice==1:
84          off = random.randint(3, 4)
85          x_random.append(x_coord[i]+off)
86          y_random.append(y_coord[i]+off)
87      if choice==2:
88          off = random.randint(-4, -3)
89          x_random.append(x_coord[i]+off)
90          y_random.append(y_coord[i]+off)
91      if choice==3:
92          x_random.append(x_coord[i])
93          y_random.append(y_coord[i])
```

Finally the overall scale is randomized.

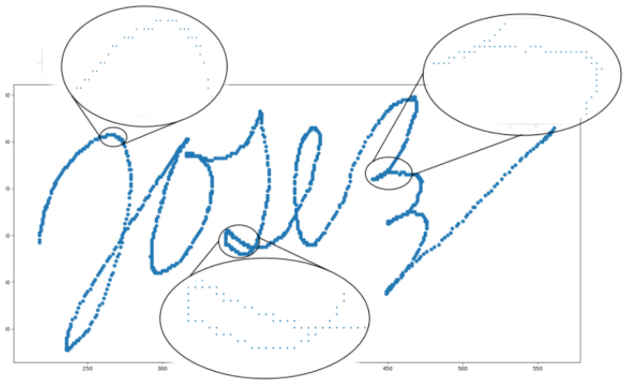


```
96   #--------Randomization Spline Lines--------
97   points_random = []
98   for xval,yval in zip(x_random,y_random):
99       if (xval,yval) not in points_random:
100          points_random.append((xval,yval))
101
102  coords_random = np.array(points_random)
103  randomx=coords_random[:,0]
104  randomy=coords_random[:,1]
105  tck,u = interpolate.splprep([randomx,randomy])
106  u=np.linspace(u.min(),u.max(),num=10000,endpoint=True)
107  out_random = interpolate.splev(u,tck)
```

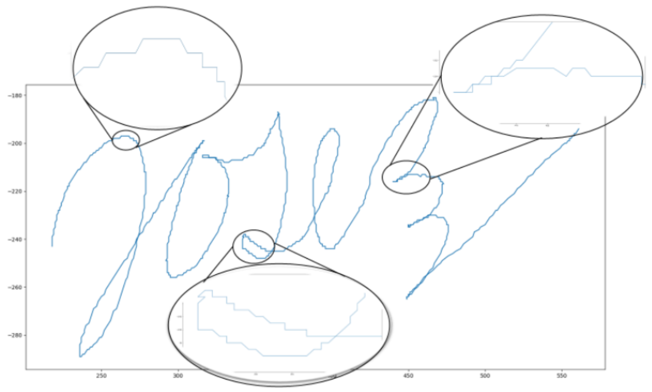
From line 122 to 140 we generate 3 figures. The first is a scatter plot of the original points obtained by openCV.



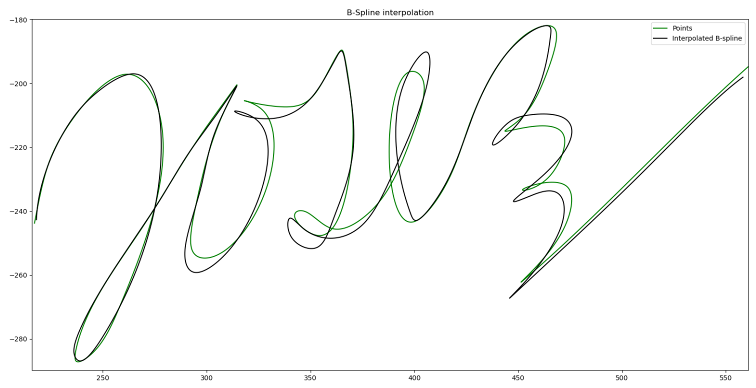
```
122  # --------Plot--------
123  plt.figure()
124  # plt.plot(x, y, 'ro', out[0], out[1], 'b')
125  plt.plot(out[0], out[1], 'g')
126  plt.plot(out_random[0], out_random[1], 'k')
127  # plt.plot(randomx, randomy, 'go', out_random[0], out_random[1], 'k')
128  plt.legend(['Points', 'Interpolated B-spline', 'True'],loc='best')
129  plt.axis([min(x)-1, max(x)+1, min(y)-1, max(y)+1])
130  plt.title('B-Spline interpolation')
131  plt.show()
132
133  #--------Points--------
134  plt.figure()
135  plt.scatter(x_coord,y_coord)
136  plt.show()
137  #--------Lines--------
138  plt.figure()
139  plt.plot(x_coord,y_coord)
140  plt.show()
```



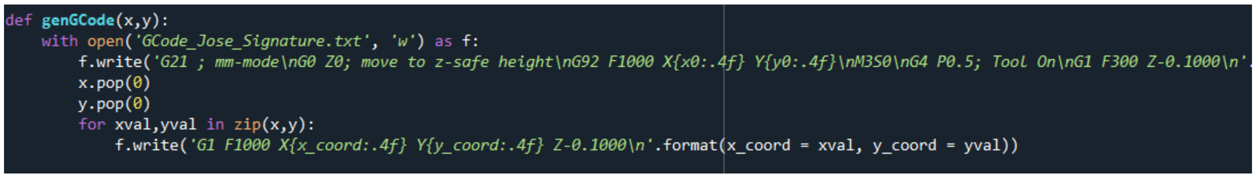
The second is a figure of the points joined by lines without smoothing the curves, it is done without using spline lines



The third image contains two signatures, the first in green color is the original signature corrected by using spline lines and in black we can find the signature randomized by the program, which as we can see in the image is very similar but not the same .



In lines 96-107 the randomized array is used to create spline lines for the G-code coordinates.



```
def genGCode(x,y):
    with open('GCode_Jose_Signature.txt', 'w') as f:
        f.write('G21 ; mm-mode\nG0 Z0; move to z-safe height\nG92 F1000 X{x0:.4f} Y{y0:.4f}\nM3S0\nG4 P0.5; Tool On\nG1 F300 Z-0.1000\n'
        x.pop(0)
        y.pop(0)
        for xval,yval in zip(x,y):
            f.write('G1 F1000 X{x_coord:.4f} Y{y_coord:.4f} Z-0.1000\n'.format(x_coord = xval, y_coord = yval))
```

This function generates a txt file that can be inputted as G-code to the Openbuilds plotter. The G-code control software can be found at this link.

Code Functionality Description Summary:

1. Import Libraries
2. Import Video
3. While Loop (Per Frame):
    a. Gaussian Blur
    b. Convert from BGR to HSV format
    c. Define green color range and create a black and white mask
    d. Get contours
    e. Get area created by contrours and center of mass
    f. Display circle with x,y coordinates in image
    g. Populate x_coord, y_coord array
4. Randomization
    a. Divide scatter points into 15 segments
    b. Apply one of three distortion options to each segment
        i. Move all points in positive x and y
        ii. Move all points in negative x and y
        iii. Move no points
    c. Randomize scale
5. Apply spline lines to connect each scatter point
6. Generate G code as txt file