Ring Pull Strain Analysis (RPSA) Tutorial

Introduction

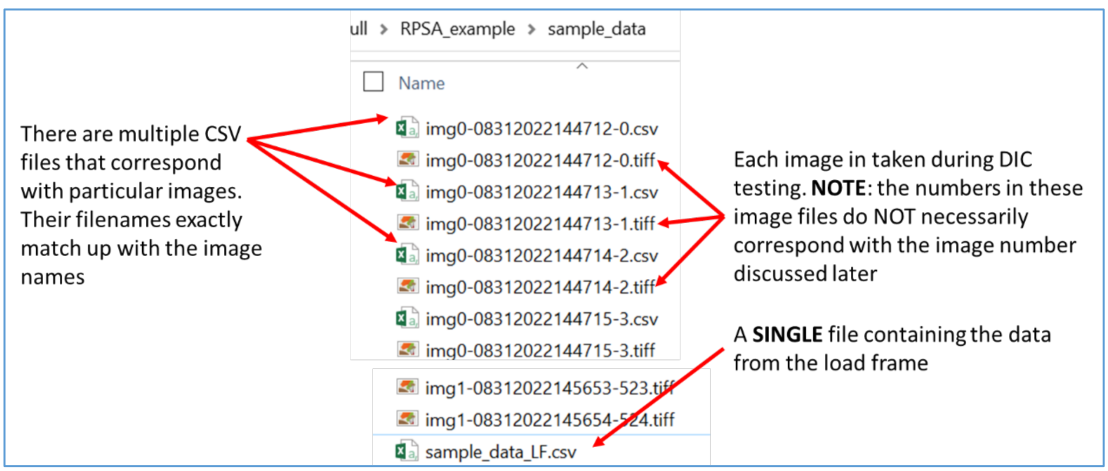
This tutorial is intended for a novice user to be able to run the Ring Pull Strain Analysis program.

Getting Started

Please download Python from anaconda.org. There are multiple ways to download Python, however, this one contains all the necessary modules and the features of Anaconda will be displayed heavily in this tutorial.

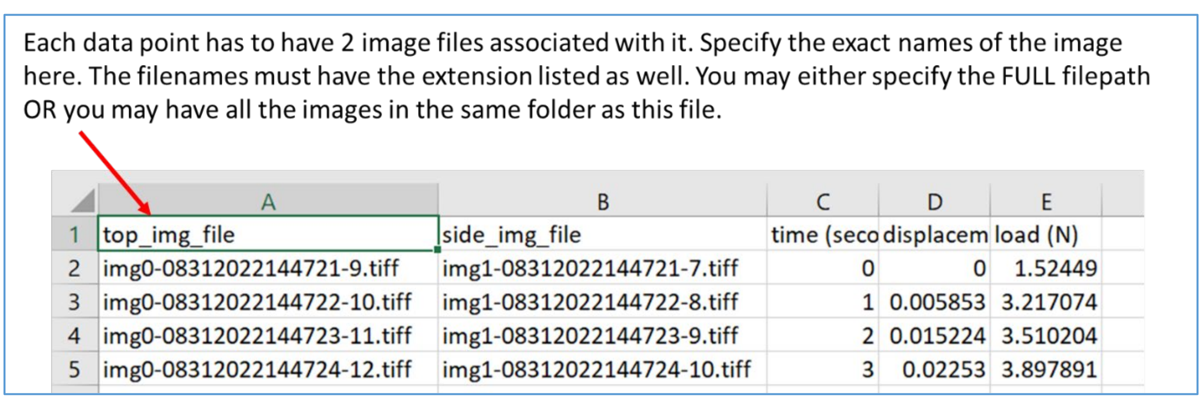
File Structure

The files that are needed to run this program are shown below



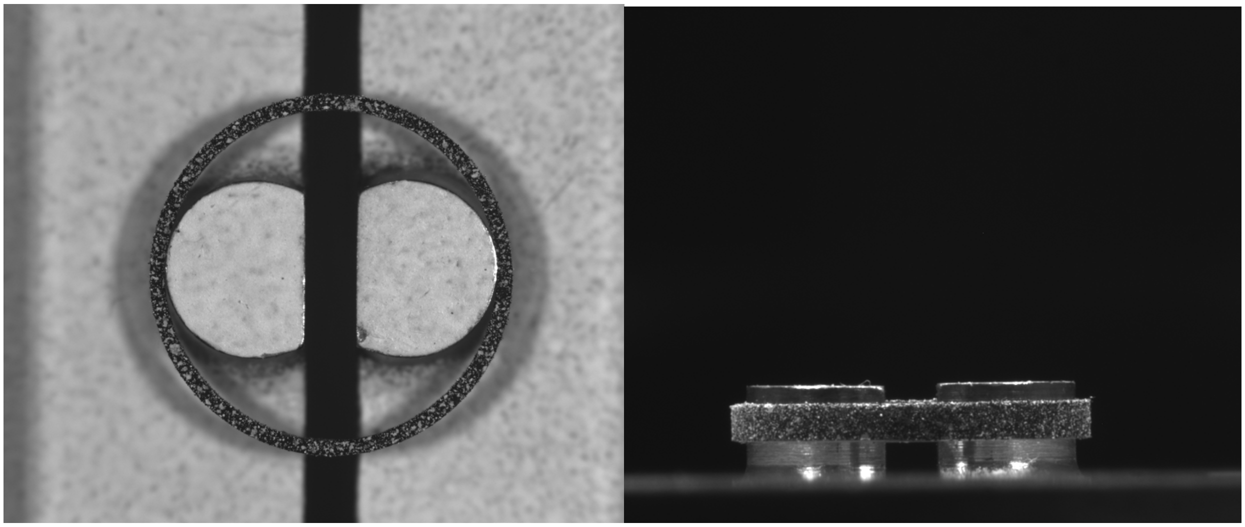
Here are some images of each file. Note that these file must have specific headers. This information is found in the RPSA Documentation:

Load Frame File:



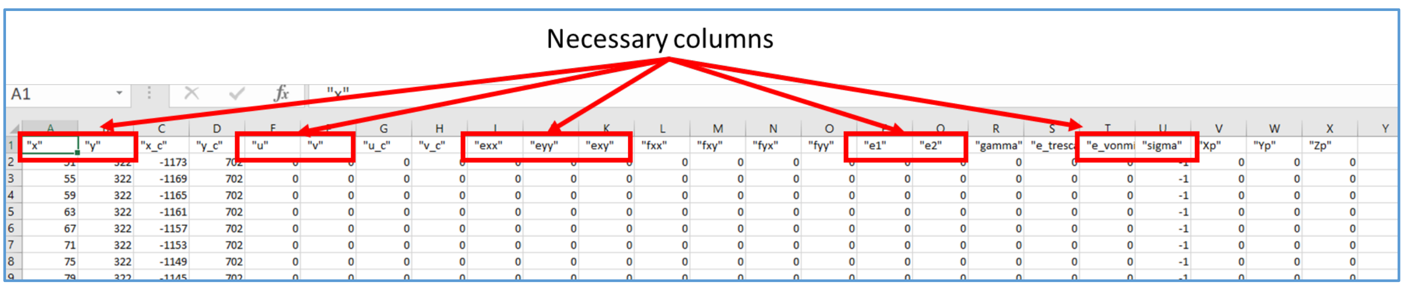
	A	B	C	D	E
1	top_img_file	side_img_file	time (seco	displacem	load (N)
2	img0-08312022144721-9.tiff	img1-08312022144721-7.tiff	0	0	1.52449
3	img0-08312022144722-10.tiff	img1-08312022144722-8.tiff	1	0.005853	3.217074
4	img0-08312022144723-11.tiff	img1-08312022144723-9.tiff	2	0.015224	3.510204
5	img0-08312022144724-12.tiff	img1-08312022144724-10.tiff	3	0.02253	3.897891

Image Files:



csv files:

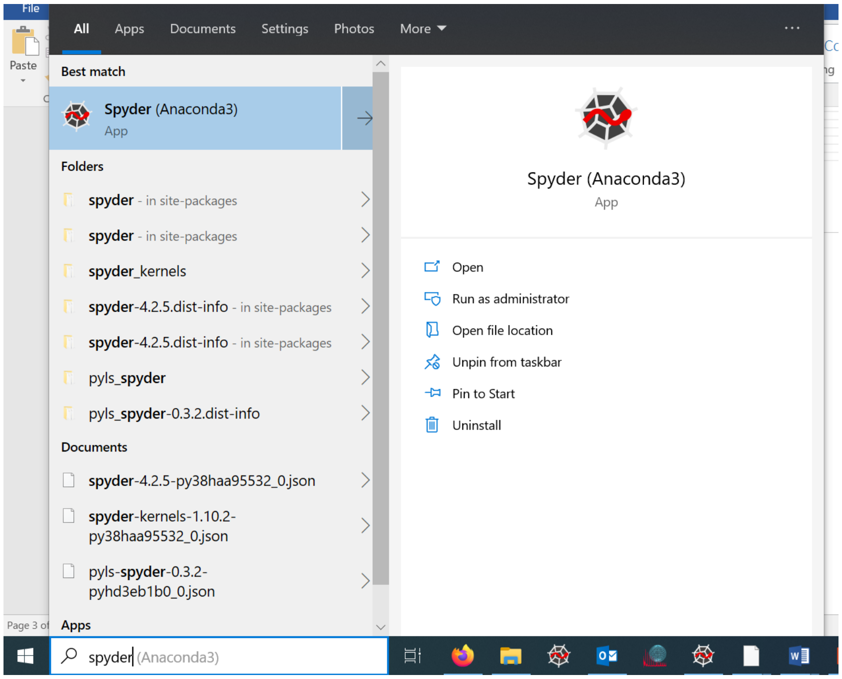
- These files are most often generated from a third party digital image correlation software. Supported file modes are VIC-2D 6, VIC-2D 7, and DICengine.



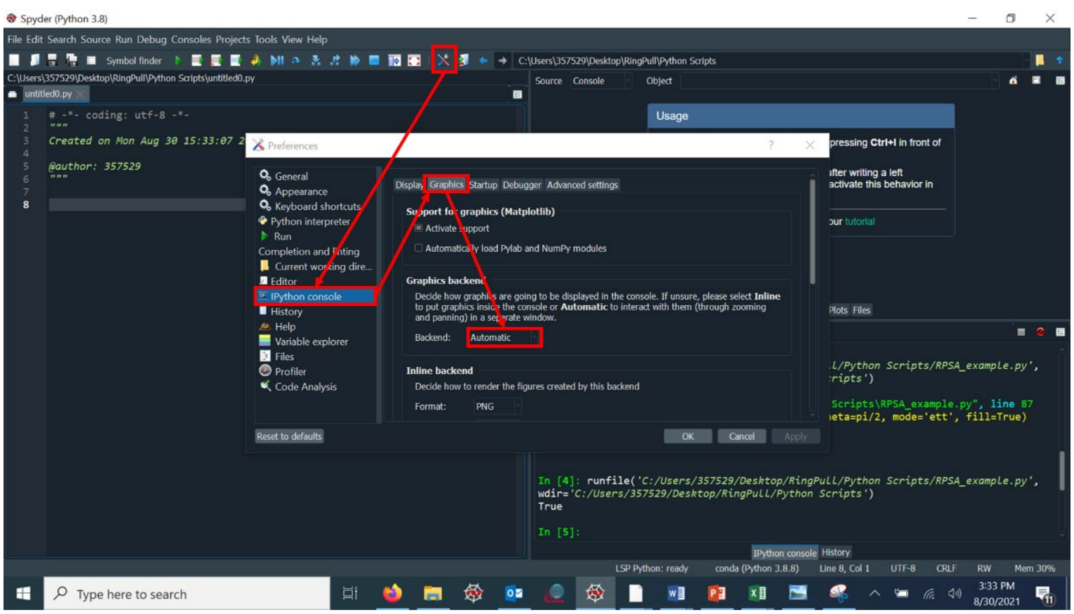
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	"x"	"y"	"x_c"	"y_c"	"u"	"v"	"u_c"	"v_c"	"exx"	"eyy"	"exy"	"fxx"	"fxy"	"fyx"	"fyy"	"e1"	"e2"	"gamma"	"e_tresc	"e_vonmi	"sigma"	"Xp"	"Yp"	"Zp"	
2	51	322	-1173	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
3	55	322	-1169	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
4	59	322	-1165	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
5	63	322	-1161	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
6	67	322	-1157	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
7	71	322	-1153	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
8	75	322	-1149	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	
9	79	322	-1145	702	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	

Opening and running the script

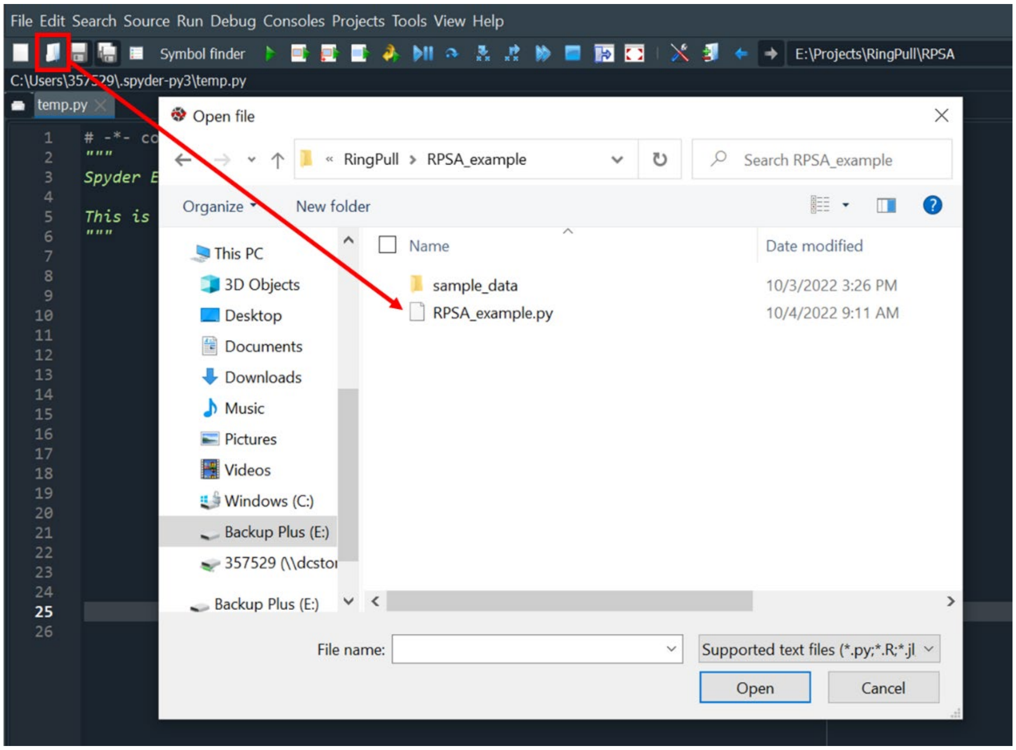
Start by opening Spyder:



It is required to change your plot settings to anything except inline. This will allow Spyder to create additional windows for the figures you plot.



Now, go ahead and open the RPSA example script:



Now, walking through the script. Here are some imported packages. If you downloaded Anaconda, these should all be available with minimal hassle. We also define pi as we will often use it. Note the red box – this is where we import the module RingPullStrainAnalysis.py from a different folder.

```python
## import the module from a different folder
import sys
sys.path.insert(0, 'E:\\Projects\\RingPull\\RPSA')
from RingPullStrainAnalysis import *

import numpy as np
pi = np.pi
import matplotlib.pyplot as plt

```

Here are the variables you can change in the code. The most important are the directory and the filename for the load frame data. Here is also where you specify geometric dimensions of the ring.

This variable must have the full filepath

Requires either double backslashes (shown) or single frontslash

```
## set all the variables you will need to run this RingPullStrainAnalysis code:

## the file with the load frame data and their corresponding images
LF_file = 'E:\\Projects\\RingPull\\RPSA_example\\sample_data\\sample_data_LF.csv'
#The DIC analysis software that was used
DIC_software = 'VIC-2D'
## geometric dimensions of the ring pull test
d_mandrel = 3.0
OD = 10.3
ID = 9.46
W  = 1
```

All filenames must have their extensions on them

This line of code creates an instance of the RingPull class which is intended to encompass an entire test. Practically, it loads in the load frame data and stores it in memory.

```

## create RingPull object
test = RingPull(LF_file=LF_file,
                DIC_software=DIC_software,
                ID=ID, OD=OD, d_mandrel=d_mandrel, W=W,
                get_geometry=False)

```

You can do further analysis with each image in this test with these lines of code. **NOTE**: this may take awhile to run, so they are normally run once. They will output a csv file of all data that gets analyzed, so you can close the entire program and reload it and not need to rerun this function.

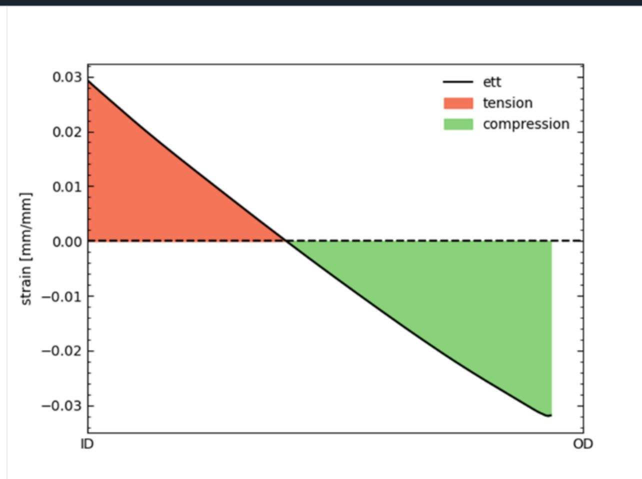
```

## Analyze the DIC images and pull out usefull parameters
## this is currently commented out because it has a long computation time
test.analyze_DIC()
## running this ^^ function, it saved the data to an output file called o
## you can also save the data without analyzing it (though you may miss o
test.save_data()
## you can also access this data in script with:
df = test.df

```

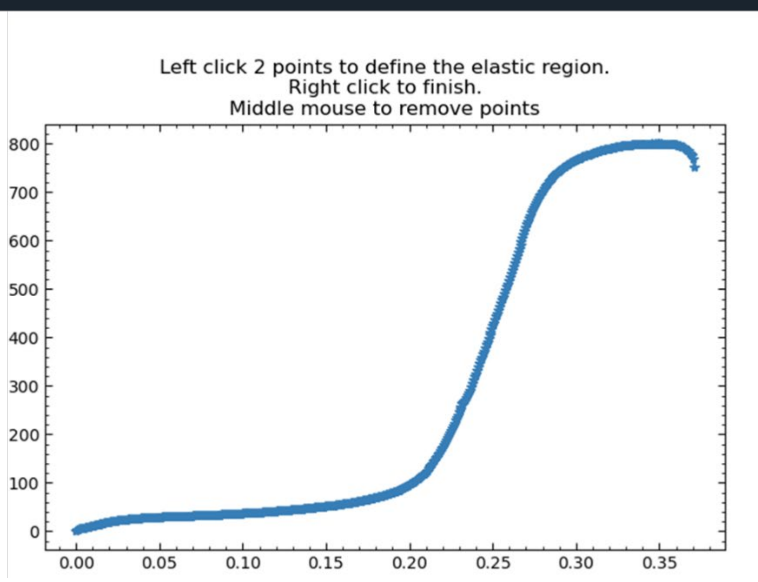
Now, the next method can create a plot of the strain across the thickness of the ring

```
## plot the strain distribution from one of the DIC images
ax = test.plot_strain_distribution(n=245, theta=pi/2, mode='ett', extrap=False, fill=True)
ax.legend(frameon=False)
```



You can also analyze this curve as if it were a tensile curve. When this command is run, user graphical input is requested.

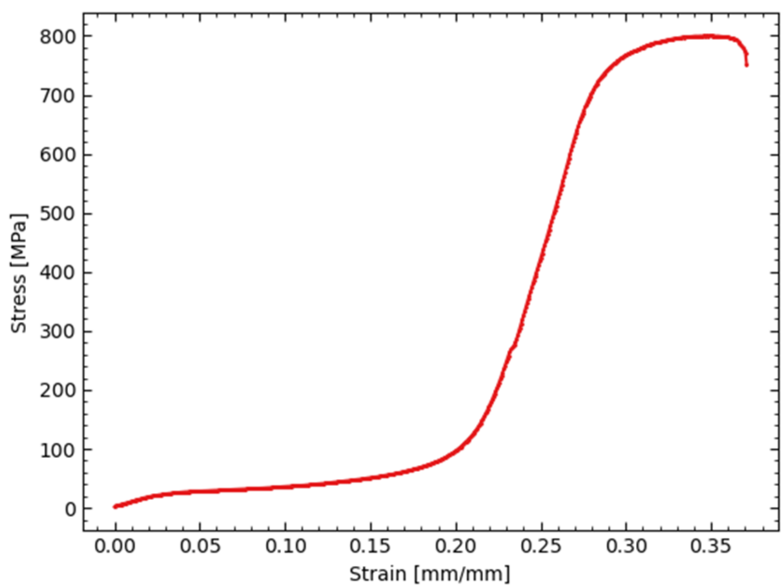
```
## analyze the curve as if it were a stress strain curve from a tensile test
## and output important material parameters
E,YS,UTS,eps_u,eps_nu,eps_total,toughness = test.process_stress_strain_curve()
```



```

## Some more plotting methods
ax = test.plot_stress_strain()
ax.set_xlabel('Strain [mm/mm]')
ax.set_ylabel('Stress [MPa]')

```



Now, we can open a single image file to look at a specific point in the test. You can also access the data in spyder’s variable editor.

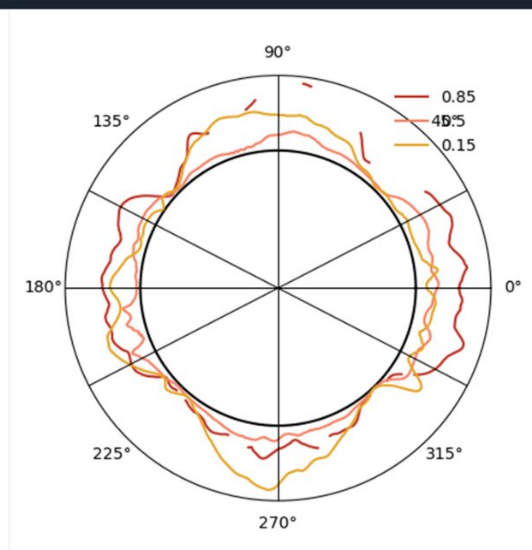
We opened a random image (number 354). Note that the image numbers correspond with their place in the load frame file, and NOT with any numbers in their filenames

```python
## open one of the DIC_image classes from the RingPull object
img = test.open_DIC(354)
print(type(img)==DIC_image)

## again, this data is pulled in from the csv file. You can see the data here:
df2 = img.df
```

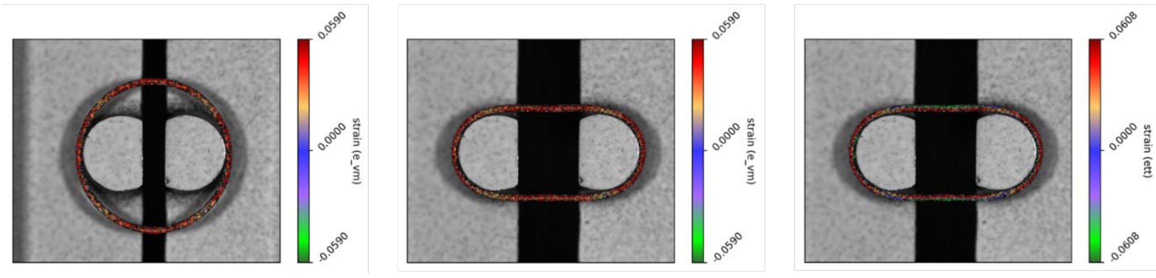
Now we can also plot a few more things, shown below.

```python
## plots the strain map on a polar plot. Plotting angles makes a new grid
## based on the test parameters
ax = img.plot_polar()
```

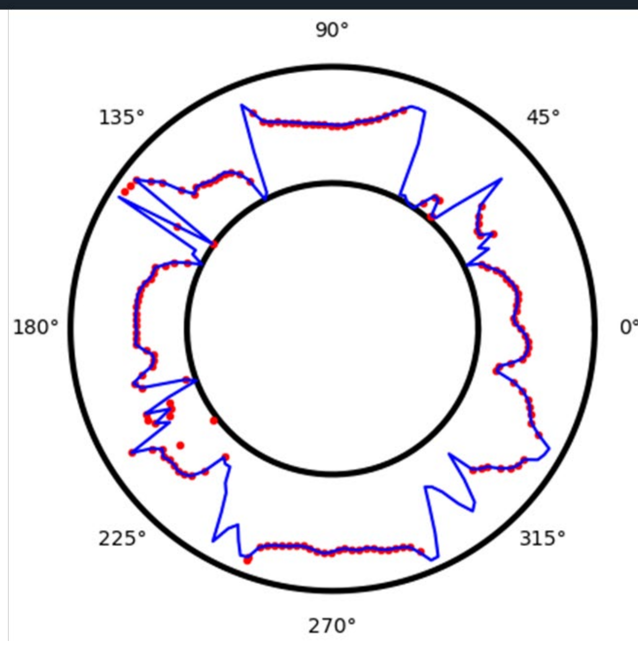


```python
## Plots the DIC results overlayed on the image
img.plot_DIC(state='reference',pixel_size=5)
img.plot_DIC(state='deformed',pixel_size=5)

## You can also plot strains in polar coordinates.
img.plot_DIC(state='deformed',mode='ett',pixel_size=5)
```




```python
## Plot the neutral axis of the ring
img.plot_neutral_axis()
```



```python

## if you want to put multiple figures on one plot
f = plt.figure()

ax = f.add_subplot(2,2,1)
test.plot_strain_distribution(n=354, ax=ax, theta=pi/2, mode='ett', fill=True,extrap=True)

ax = f.add_subplot(2,2,2)
img.plot_DIC(state='reference',mode='e_vm',ax=ax)

ax = f.add_subplot(2,2,3,projection='polar')
img.plot_neutral_axis(ax=ax)

ax = f.add_subplot(2,2,4)
test.plot_stress_strain(ax=ax)

```