In [1]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (5.16.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (fro
m plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\lenovo\appdata\roaming\python\python311\site-packages (fro
m plotly) (23.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install scikit-image
```

```
Requirement already satisfied: scikit-image in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (0.21.0
)
Requirement already satisfied: numpy>=1.21.1 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (from
scikit-image) (1.24.3)
Requirement already satisfied: scipy>=1.8 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (from sci
kit-image) (1.10.1)
Requirement already satisfied: networkx>=2.8 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (from
scikit-image) (3.1)
Requirement already satisfied: pillow>=9.0.1 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (from
scikit-image) (9.4.0)
Requirement already satisfied: imageio>=2.27 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (from
scikit-image) (2.31.1)
Requirement already satisfied: tifffile>=2022.8.12 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages
(from scikit-image) (2023.7.4)
Requirement already satisfied: PyWavelets>=1.1.1 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (f
rom scikit-image) (1.4.1)
Requirement already satisfied: packaging>=21 in c:\users\lenovo\appdata\roaming\python\python311\site-packages
(from scikit-image) (23.0)
Requirement already satisfied: lazy_loader>=0.2 in c:\users\lenovo\anaconda3\envs\python3\lib\site-packages (fr
om scikit-image) (0.3)
Note: you may need to restart the kernel to use updated packages.
```

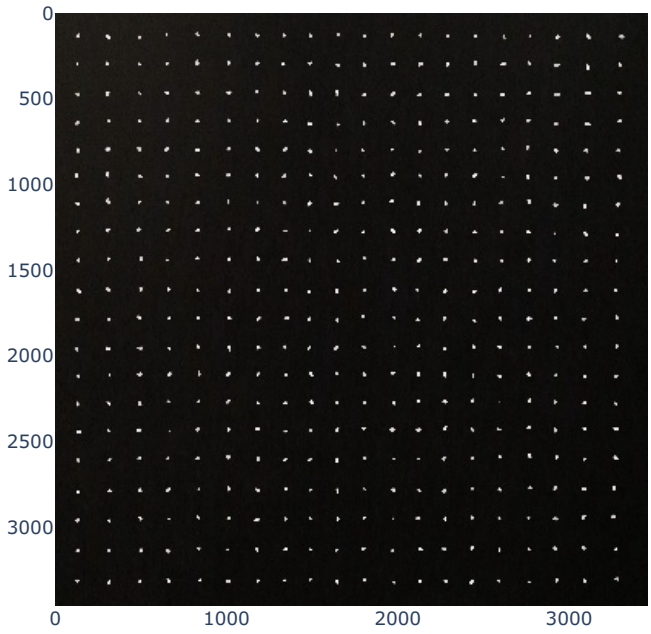
In [3]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from scipy import ndimage
from skimage import measure, color, io

import plotly
import plotly.express as px
import plotly.graph_objects as go
from skimage import data, filters, measure, morphology
from skimage import data
from skimage.color import rgb2gray
```

C1 Rice

In [4]:
```
Img_Small_C1 = cv2.imread("D:\\Test_AI\\Images_Datasets\\01_Training_and_Validate_images\\01_Small_C1.jpg")
```

In [5]:
```
Img_fig_Small_C1 = px.imshow(Img_Small_C1, binary_string=True)
Img_fig_Small_C1.update_traces(hoverinfo='skip')
```

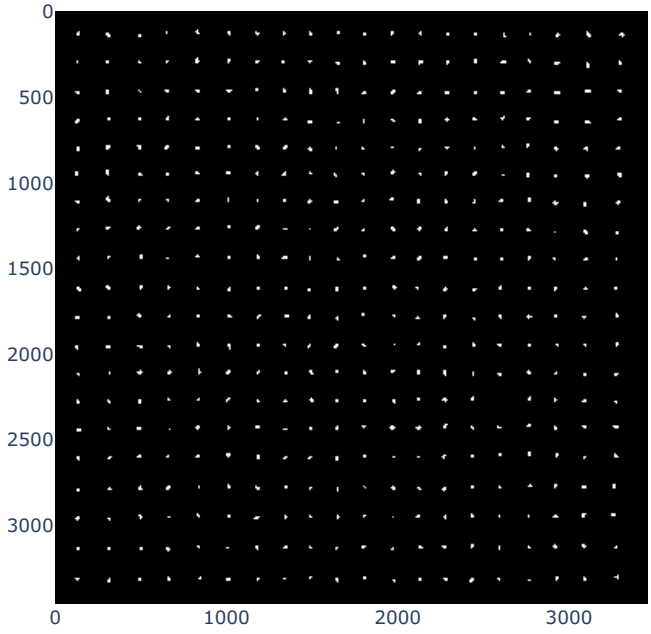


In [6]:
```
Img_Small_C1_Gray=rgb2gray(Img_Small_C1)
```

In [7]:
```
threshold_Small_C1 = filters.threshold_otsu(Img_Small_C1_Gray)
```

In [8]:
```
img_mask_Small_C1 = Img_Small_C1_Gray > threshold_Small_C1
img_mask_Small_C1 = morphology.remove_small_objects(img_mask_Small_C1, 15)
img_mask_Small_C1 = morphology.remove_small_holes(img_mask_Small_C1, 15)
```

In [9]:
```
Img_fig_Small_C1 = px.imshow(img_mask_Small_C1, binary_string=True)
Img_fig_Small_C1.update_traces(hoverinfo='skip')
```



In [10]:
```
labels_Small_C1 = measure.label(img_mask_Small_C1)
```

In [11]:
```
import pandas as pd
import numpy as np
from skimage.measure import regionprops, regionprops_table

# Small_C1
props_Small_C1 = regionprops_table(labels_Small_C1, properties=('area', 'major_axis_length', 'minor_axis_length
```

```
# Create empty lists to store property values
equiv_diameters = []
aspect_ratios = []
compactnesses = []
roundnesses = []
categories = []

# Iterate over regions for Small_C1
for idx, region in enumerate(regionprops(labels_Small_C1)):
    # Calculate properties for each region
    equiv_diameter = np.sqrt(4 * region.area / np.pi)
    aspect_ratio = region.major_axis_length / region.minor_axis_length
    compactness = (region.perimeter ** 2) / (4 * np.pi * region.area)
    roundness = (4 * region.area) / (np.pi * (region.major_axis_length ** 2))

    # Append the calculated values to the respective lists
    equiv_diameters.append(equiv_diameter)
    aspect_ratios.append(aspect_ratio)
    compactnesses.append(compactness)
    roundnesses.append(roundness)
    categories.append('Small Broke C1')  # Category 1 for Small_C1

# Create a dictionary with all the properties
props_Small_C1 = {
    'area': props_Small_C1['area'],
    'major_axis_length': props_Small_C1['major_axis_length'],
    'minor_axis_length': props_Small_C1['minor_axis_length'],
    'perimeter': props_Small_C1['perimeter'],
    'eccentricity': props_Small_C1['eccentricity'],
    'solidity': props_Small_C1['solidity'],
    'extent': props_Small_C1['extent'],
    'equiv_diameter': equiv_diameters,
    'aspect_ratio': aspect_ratios,
    'compactness': compactnesses,
    'roundness': roundnesses,
    'category': categories,
}

# Create a DataFrame from the dictionary
df_Small_C1 = pd.DataFrame(props_Small_C1)
```

In [12]:
```
df_Small_C1
```

Out[12]:

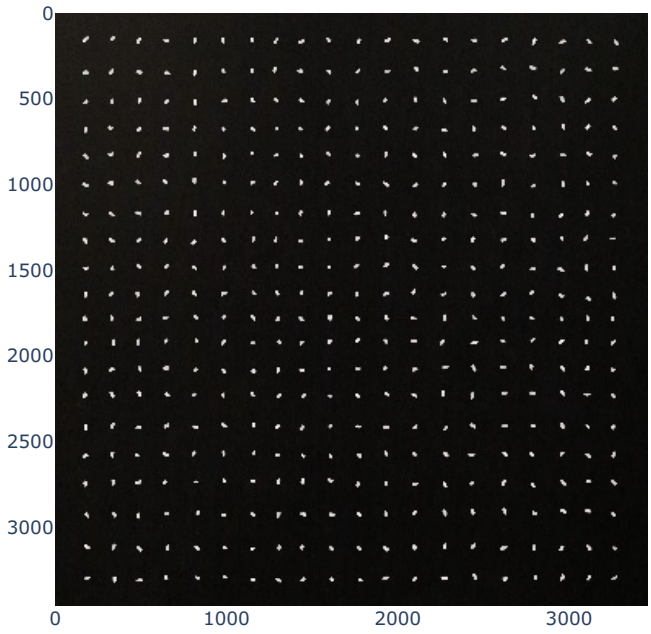
	area	major_axis_length	minor_axis_length	perimeter	eccentricity	solidity	extent	equiv_diameter	aspect_ratio	compactness	rou
0	413.0	27.069175	20.234124	78.911688	0.664265	0.956019	0.611852	22.931374	1.337798	1.199838	0
1	431.0	29.341232	18.783145	77.597980	0.768240	0.966368	0.720736	23.425760	1.562104	1.111767	0
2	372.0	24.753551	19.302683	71.455844	0.626036	0.958763	0.808696	21.763389	1.282389	1.092252	0
3	455.0	29.248325	20.394311	82.325902	0.716798	0.953878	0.627586	24.069150	1.434141	1.185364	0
4	341.0	24.252606	19.119481	72.769553	0.615230	0.929155	0.644612	20.836859	1.268476	1.235763	0
...	...	...	...	...	...	...	...	...	...	...	
395	437.0	29.425419	19.009271	78.669048	0.763324	0.966814	0.674383	23.588253	1.547951	1.126981	0
396	329.0	25.232797	16.786082	68.284271	0.746622	0.961988	0.746032	20.466944	1.503198	1.127809	0
397	207.0	18.638747	14.319941	52.384776	0.640103	0.962791	0.821429	16.234549	1.301594	1.054945	0
398	259.0	23.574455	14.259290	61.698485	0.796330	0.948718	0.685185	18.159544	1.653270	1.169605	0
399	179.0	16.510724	14.192555	48.627417	0.510974	0.962366	0.745833	15.096684	1.163337	1.051234	0

400 rows × 12 columns

Small Broken

In [13]:
```
Img_Small_Broke = cv2.imread("D:\\Test_AI\\Images_Datasets\\01_Training_and_Validate_images\\02_Smallbroke.jpg"
```

In [14]:
```
Img_fig_Small_Broke = px.imshow(Img_Small_Broke, binary_string=True)
Img_fig_Small_Broke.update_traces(hoverinfo='skip')
```

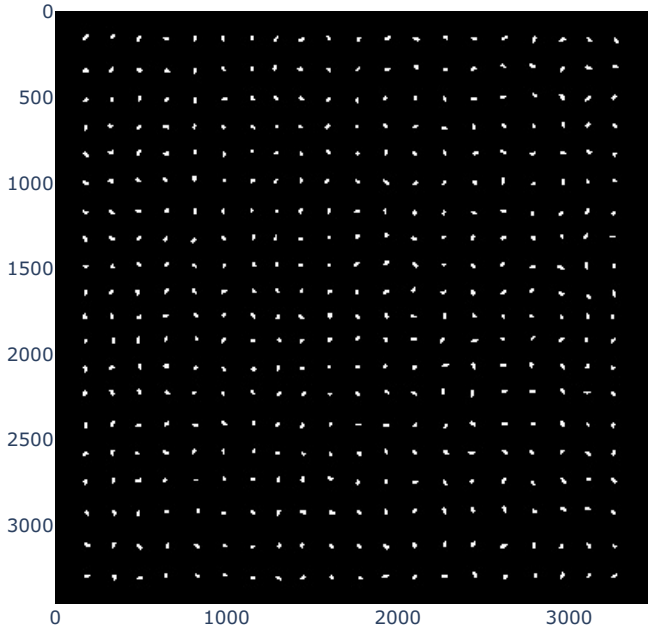


In [15]:
```
Img_Small_Broke_Gray=rgb2gray(Img_Small_Broke)
```

In [16]:
```
threshold_Small_Broke = filters.threshold_otsu(Img_Small_Broke_Gray)
```

In [17]:
```
img_mask_Small_Broke = Img_Small_Broke_Gray > threshold_Small_Broke
img_mask_Small_Broke = morphology.remove_small_objects(img_mask_Small_Broke, 15)
img_mask_Small_Broke = morphology.remove_small_holes(img_mask_Small_Broke, 15)
```

In [18]:
```
Img_fig_Small_Broke = px.imshow(img_mask_Small_Broke, binary_string=True)
Img_fig_Small_Broke.update_traces(hoverinfo='skip')
```



In [19]:
```
labels_Small_Broke = measure.label(img_mask_Small_Broke)
```

In [20]:
```
import pandas as pd
import numpy as np
from skimage.measure import regionprops, regionprops_table

# Small_Broke
props_Small_Broke = regionprops_table(labels_Small_Broke, properties=('area', 'major_axis_length', 'minor_axis_
```

```
# Create empty lists to store property values
equiv_diameters = []
aspect_ratios = []
compactnesses = []
roundnesses = []
categories = []

# Iterate over regions for Small_C1
for idx, region in enumerate(regionprops(labels_Small_Broke)):
    # Calculate properties for each region
    equiv_diameter = np.sqrt(4 * region.area / np.pi)
    aspect_ratio = region.major_axis_length / region.minor_axis_length
    compactness = (region.perimeter ** 2) / (4 * np.pi * region.area)
    roundness = (4 * region.area) / (np.pi * (region.major_axis_length ** 2))

    # Append the calculated values to the respective lists
    equiv_diameters.append(equiv_diameter)
    aspect_ratios.append(aspect_ratio)
    compactnesses.append(compactness)
    roundnesses.append(roundness)
    categories.append('Small Broke')  # Category 2 for Small_Broke

# Create a dictionary with all the properties
props_Small_Broke = {
    'area': props_Small_Broke['area'],
    'major_axis_length': props_Small_Broke['major_axis_length'],
    'minor_axis_length': props_Small_Broke['minor_axis_length'],
    'perimeter': props_Small_Broke['perimeter'],
    'eccentricity': props_Small_C1['eccentricity'],
    'solidity': props_Small_Broke['solidity'],
    'extent': props_Small_Broke['extent'],
    'equiv_diameter': equiv_diameters,
    'aspect_ratio': aspect_ratios,
    'compactness': compactnesses,
    'roundness': roundnesses,
    'category': categories,
}

# Create a DataFrame from the dictionary
df_Small_Broke = pd.DataFrame(props_Small_Broke)
```

In [21]:
```
df_Small_Broke
```

Out[21]:

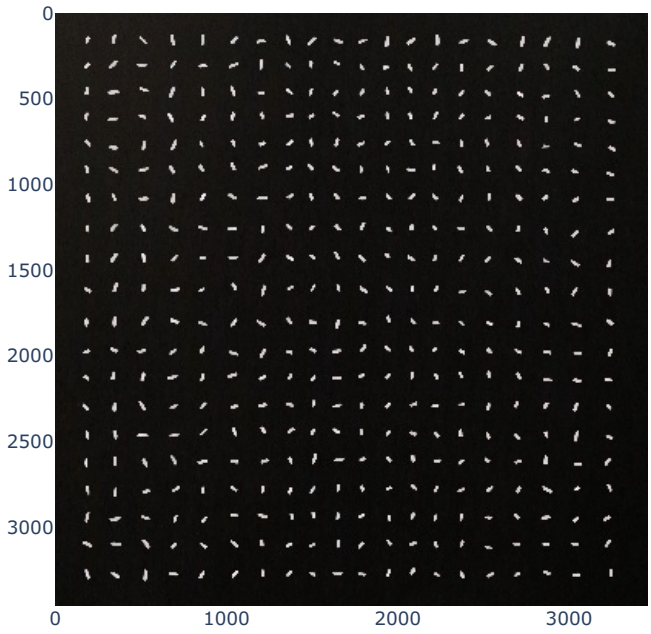
	area	major_axis_length	minor_axis_length	perimeter	eccentricity	solidity	extent	equiv_diameter	aspect_ratio	compactness	ro
0	593.0	34.549034	22.293055	94.225397	0.664265	0.968954	0.637634	27.477828	1.549767	1.191438	
1	771.0	44.108516	22.560561	111.882251	0.768240	0.964956	0.599068	31.331577	1.955116	1.291987	
2	632.0	38.994507	20.976194	100.811183	0.626036	0.959029	0.610039	28.367012	1.858989	1.279648	
3	584.0	38.825923	19.548198	97.112698	0.716798	0.968491	0.808864	27.268515	1.986164	1.285078	
4	570.0	34.697853	21.198817	92.811183	0.615230	0.962838	0.612903	26.939683	1.636783	1.202585	
...	...	...	...	...	...	...	...	...	...	...	
395	541.0	32.266642	21.712996	89.012193	0.763324	0.957522	0.735054	26.245430	1.486052	1.165445	
396	460.0	30.882097	19.103492	81.941125	0.746622	0.956341	0.755337	24.201037	1.616568	1.161545	
397	560.0	37.700454	19.242313	96.426407	0.640103	0.955631	0.658824	26.702325	1.959248	1.321278	
398	555.0	34.406192	21.009002	91.941125	0.796330	0.956897	0.754076	26.582851	1.637688	1.212040	
399	577.0	39.978054	18.833735	98.526912	0.510974	0.960067	0.742600	27.104598	2.122683	1.338826	

400 rows × 12 columns

Big Broke

In [22]:
```
Img_Big_Broke = cv2.imread("D:\\Test_AI\\Images_Datasets\\01_Training_and_Validate_images\\03_Bigbroke.jpg")
```

In [23]:
```
Img_fig_Big_Broke = px.imshow(Img_Big_Broke, binary_string=True)
Img_fig_Big_Broke.update_traces(hoverinfo='skip')
```

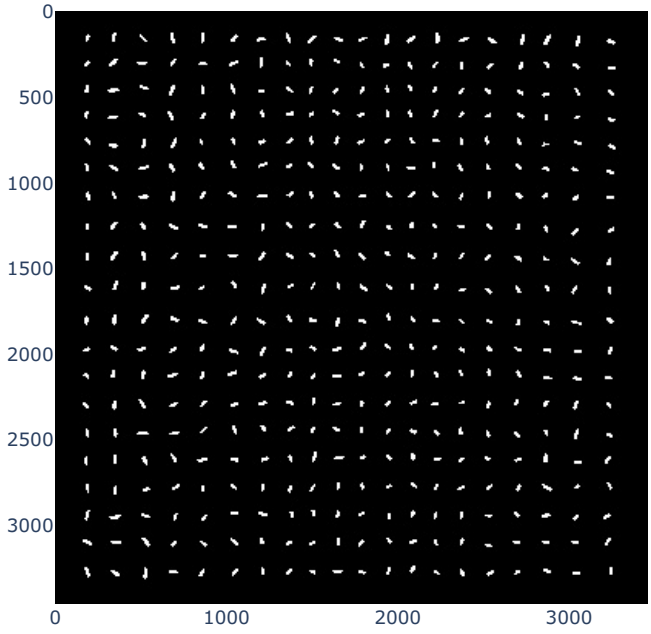


In [24]:
```
Img_Big_Broke_Gray=rgb2gray(Img_Big_Broke)
```

In [25]:
```
threshold_Big_Broke = filters.threshold_otsu(Img_Big_Broke_Gray)
```

In [26]:
```
img_mask_Big_Broke = Img_Big_Broke_Gray > threshold_Big_Broke
img_mask_Big_Broke = morphology.remove_small_objects(img_mask_Big_Broke, 15)
img_mask_Big_Broke = morphology.remove_small_holes(img_mask_Big_Broke, 15)
```

In [27]:
```
Img_fig_Big_Broke = px.imshow(img_mask_Big_Broke, binary_string=True)
Img_fig_Big_Broke.update_traces(hoverinfo='skip')
```



In [28]:
```
labels_Big_Broke = measure.label(img_mask_Big_Broke)
```

In [29]:
```
import pandas as pd
import numpy as np
from skimage.measure import regionprops, regionprops_table

# Big_Broke
props_Big_Broke = regionprops_table(labels_Big_Broke, properties=('area', 'major_axis_length', 'minor_axis_leng
```

```
# Create empty lists to store property values
equiv_diameters = []
aspect_ratios = []
compactnesses = []
roundnesses = []
categories = []

# Iterate over regions for Big Broke
for idx, region in enumerate(regionprops(labels_Big_Broke)):
    # Calculate properties for each region
    equiv_diameter = np.sqrt(4 * region.area / np.pi)
    aspect_ratio = region.major_axis_length / region.minor_axis_length
    compactness = (region.perimeter ** 2) / (4 * np.pi * region.area)
    roundness = (4 * region.area) / (np.pi * (region.major_axis_length ** 2))

    # Append the calculated values to the respective lists
    equiv_diameters.append(equiv_diameter)
    aspect_ratios.append(aspect_ratio)
    compactnesses.append(compactness)
    roundnesses.append(roundness)
    categories.append('Big Broke')  # Category 3 for Big_Broke

# Create a dictionary with all the properties
props_Big_Broke = {
    'area': props_Big_Broke['area'],
    'major_axis_length': props_Big_Broke['major_axis_length'],
    'minor_axis_length': props_Big_Broke['minor_axis_length'],
    'perimeter': props_Big_Broke['perimeter'],
    'eccentricity': props_Big_Broke['eccentricity'],
    'solidity': props_Big_Broke['solidity'],
    'extent': props_Big_Broke['extent'],
    'equiv_diameter': equiv_diameters,
    'aspect_ratio': aspect_ratios,
    'compactness': compactnesses,
    'roundness': roundnesses,
    'category': categories,
}

# Create a DataFrame from the dictionary
df_Big_Broke = pd.DataFrame(props_Big_Broke)
```

In [30]:
```
df_Big_Broke
```

Out[30]:

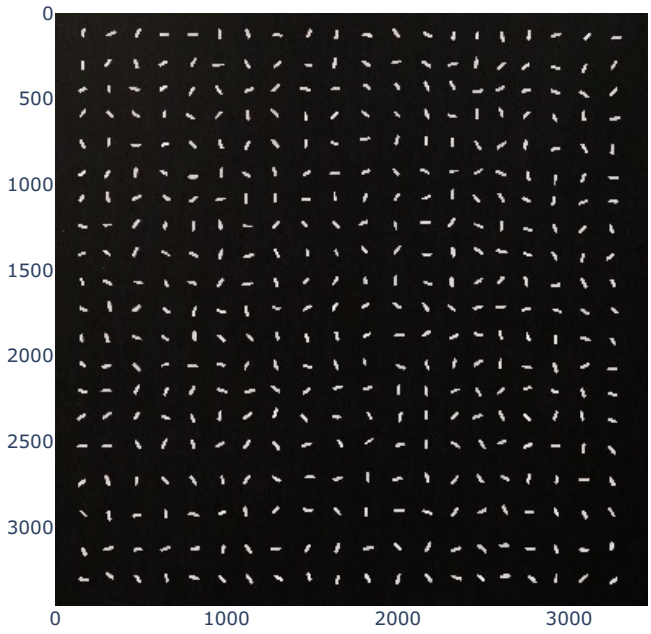
	area	major_axis_length	minor_axis_length	perimeter	eccentricity	solidity	extent	equiv_diameter	aspect_ratio	compactness	r
0	1147.0	68.219585	21.764530	150.426407	0.947742	0.972034	0.840293	38.215256	3.134439	1.569910	
1	1123.0	64.232447	22.818783	148.911688	0.934770	0.950085	0.679782	37.813331	2.814894	1.571332	
2	1167.0	67.691207	22.393077	153.982756	0.943697	0.949552	0.649055	38.546991	3.022863	1.616827	
3	1245.0	70.654341	22.904523	158.669048	0.945996	0.953292	0.820158	39.814360	3.084733	1.609182	
4	1046.0	62.702606	21.481508	145.882251	0.939484	0.961397	0.573465	36.493952	2.918911	1.619062	
...	...	...	...	...	...	...	...	...	...	...	
395	819.0	54.053746	19.591040	125.012193	0.932009	0.952326	0.712174	32.292154	2.759106	1.518487	
396	793.0	52.748518	19.517069	122.426407	0.929031	0.957729	0.688368	31.775446	2.702686	1.504067	
397	574.0	38.629126	19.371652	97.396970	0.865170	0.956667	0.590535	27.034043	1.994106	1.315131	
398	834.0	49.835363	21.912797	124.710678	0.898143	0.950969	0.549407	32.586528	2.274258	1.483991	
399	788.0	49.679429	20.371314	119.497475	0.912061	0.956311	0.730983	31.675113	2.438695	1.442054	

400 rows × 12 columns

Head Rice

In [31]:
```
Img_Head_Rice = cv2.imread("D:\\Test_AI\\Images_Datasets\\01_Training_and_Validate_images\\04_Head_Rice.jpg")
```

In [32]:
```
Img_fig_Head_Rice = px.imshow(Img_Head_Rice, binary_string=True)
Img_fig_Head_Rice.update_traces(hoverinfo='skip')
```

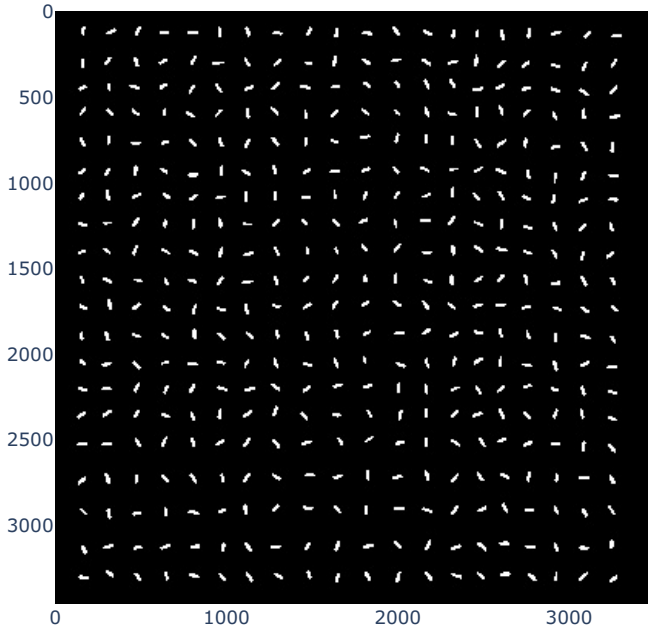


In [34]:
```
Img_Head_Rice_Gray=rgb2gray(Img_Head_Rice)
```

In [35]:
```
threshold_Head_Rice = filters.threshold_otsu(Img_Head_Rice_Gray)
```

In [36]:
```
img_mask_Head_Rice = Img_Head_Rice_Gray > threshold_Head_Rice
img_mask_Head_Rice = morphology.remove_small_objects(img_mask_Head_Rice, 15)
img_mask_Head_Rice = morphology.remove_small_holes(img_mask_Head_Rice, 15)
```

In [37]:
```
Img_fig_Head_Rice = px.imshow(img_mask_Head_Rice, binary_string=True)
Img_fig_Head_Rice.update_traces(hoverinfo='skip')
```



In [38]:
```
labels_Head_Rice = measure.label(img_mask_Head_Rice)
```

In [39]:
```
import pandas as pd
import numpy as np
from skimage.measure import regionprops, regionprops_table

# Head Rice
props_Head_Rice = regionprops_table(labels_Head_Rice, properties=('area', 'major_axis_length', 'minor_axis_leng
```

```
# Create empty lists to store property values
equiv_diameters = []
aspect_ratios = []
compactnesses = []
roundnesses = []
categories = []

# Iterate over regions for Small_Broke
for idx, region in enumerate(regionprops(labels_Head_Rice)):
    # Calculate properties for each region
    equiv_diameter = np.sqrt(4 * region.area / np.pi)
    aspect_ratio = region.major_axis_length / region.minor_axis_length
    compactness = (region.perimeter ** 2) / (4 * np.pi * region.area)
    roundness = (4 * region.area) / (np.pi * (region.major_axis_length ** 2))

    # Append the calculated values to the respective lists
    equiv_diameters.append(equiv_diameter)
    aspect_ratios.append(aspect_ratio)
    compactnesses.append(compactness)
    roundnesses.append(roundness)
    categories.append('Head rice')  # Category 4 for Head Rice

# Create a dictionary with all the properties
props_Head_Rice = {
    'area': props_Head_Rice['area'],
    'major_axis_length': props_Head_Rice['major_axis_length'],
    'minor_axis_length': props_Head_Rice['minor_axis_length'],
    'perimeter': props_Head_Rice['perimeter'],
    'eccentricity': props_Head_Rice['eccentricity'],
    'solidity': props_Head_Rice['solidity'],
    'extent': props_Head_Rice['extent'],
    'equiv_diameter': equiv_diameters,
    'aspect_ratio': aspect_ratios,
    'compactness': compactnesses,
    'roundness': roundnesses,
    'category': categories,
}

# Create a DataFrame from the dictionary
df_Head_Rice = pd.DataFrame(props_Head_Rice)
```

In [40]:
```
df_Head_Rice
```

Out[40]:

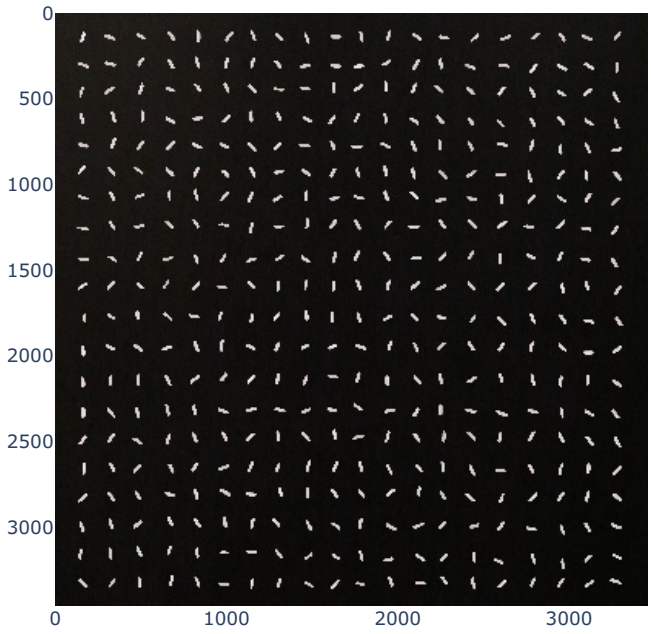
	area	major_axis_length	minor_axis_length	perimeter	eccentricity	solidity	extent	equiv_diameter	aspect_ratio	compactness	r
0	992.0	59.644196	21.689715	138.911688	0.931535	0.957529	0.712644	35.539466	2.749884	1.547947	
1	1122.0	64.586073	22.491459	150.852814	0.937405	0.957338	0.513736	37.796491	2.871582	1.614002	
2	1072.0	63.754415	21.809710	149.195959	0.939667	0.958855	0.528079	36.944726	2.923212	1.652378	
3	1097.0	62.030355	23.092146	143.698485	0.928124	0.978591	0.885391	37.373035	2.686210	1.497917	
4	1095.0	63.028184	22.473219	149.195959	0.934273	0.954664	0.539409	37.338952	2.804591	1.617671	
...	...	...	...	...	...	...	...	...	...	...	...
395	1155.0	70.539422	21.262428	160.124892	0.953490	0.956918	0.546875	38.348294	3.317562	1.766550	
396	1214.0	71.898082	21.804222	162.225397	0.952906	0.951411	0.624807	39.315554	3.297439	1.725080	
397	1101.0	63.417413	22.568901	145.923882	0.934532	0.963255	0.498641	37.441110	2.809947	1.539060	
398	1240.0	68.310124	23.498922	156.568542	0.938968	0.954580	0.691964	39.734331	2.906947	1.573177	
399	1031.0	60.887738	21.917654	139.095454	0.932965	0.978178	0.510143	36.231340	2.778023	1.493335	

400 rows × 12 columns

Whole Rice

In [41]:
```
Img_Whole_Rice = cv2.imread("D:\\Test_AI\\Images_Datasets\\01_Training_and_Validate_images\\05_Whole_Grains.jpg
```

In [42]:
```
Img_fig_Whole_Rice = px.imshow(Img_Whole_Rice, binary_string=True)
Img_fig_Whole_Rice.update_traces(hoverinfo='skip')
```

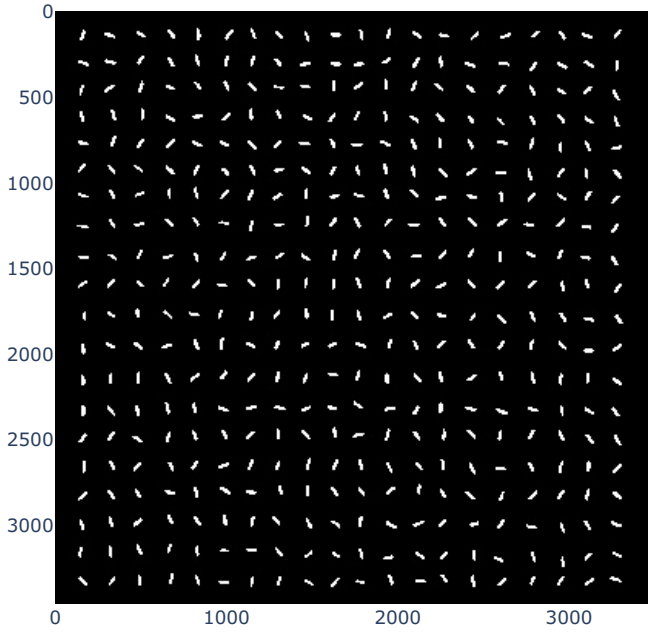


In [43]:
```
Img_Whole_Rice_Gray=rgb2gray(Img_Whole_Rice)
```

In [44]:
```
threshold_Whole_Rice = filters.threshold_otsu(Img_Whole_Rice_Gray)
```

In [45]:
```
img_mask_Whole_Rice = Img_Whole_Rice_Gray > threshold_Whole_Rice
img_mask_Whole_Rice = morphology.remove_small_objects(img_mask_Whole_Rice, 15)
img_mask_Whole_Rice = morphology.remove_small_holes(img_mask_Whole_Rice, 15)
```

In [46]:
```
Img_fig_Whole_Rice = px.imshow(img_mask_Whole_Rice, binary_string=True)
Img_fig_Whole_Rice.update_traces(hoverinfo='skip')
```



In [47]:
```
labels_Whole_Rice = measure.label(img_mask_Whole_Rice)
```

In [48]:
```
import pandas as pd
import numpy as np
from skimage.measure import regionprops, regionprops_table

# Small_Broke
props_Whole_Rice = regionprops_table(labels_Whole_Rice, properties=('area', 'major_axis_length', 'minor_axis_le
```

```
# Create empty lists to store property values
equiv_diameters = []
aspect_ratios = []
compactnesses = []
roundnesses = []
categories = []

# Iterate over regions for Small_Broke
for idx, region in enumerate(regionprops(labels_Whole_Rice)):
    # Calculate properties for each region
    equiv_diameter = np.sqrt(4 * region.area / np.pi)
    aspect_ratio = region.major_axis_length / region.minor_axis_length
    compactness = (region.perimeter ** 2) / (4 * np.pi * region.area)
    roundness = (4 * region.area) / (np.pi * (region.major_axis_length ** 2))

    # Append the calculated values to the respective lists
    equiv_diameters.append(equiv_diameter)
    aspect_ratios.append(aspect_ratio)
    compactnesses.append(compactness)
    roundnesses.append(roundness)
    categories.append('Whole Rice')  # Category 5 for Whole Rice

# Create a dictionary with all the properties
props_Whole_Rice = {
    'area': props_Whole_Rice['area'],
    'major_axis_length': props_Whole_Rice['major_axis_length'],
    'minor_axis_length': props_Whole_Rice['minor_axis_length'],
    'perimeter': props_Whole_Rice['perimeter'],
    'eccentricity': props_Whole_Rice['eccentricity'],
    'solidity': props_Whole_Rice['solidity'],
    'extent': props_Whole_Rice['extent'],
    'equiv_diameter': equiv_diameters,
    'aspect_ratio': aspect_ratios,
    'compactness': compactnesses,
    'roundness': roundnesses,
    'category': categories,
}

# Create a DataFrame from the dictionary
df_Whole_Rice = pd.DataFrame(props_Whole_Rice)
```

In [49]:
```
df_Whole_Rice
```

Out[49]:

	area	major_axis_length	minor_axis_length	perimeter	eccentricity	solidity	extent	equiv_diameter	aspect_ratio	compactness	r
0	1121.0	71.842431	20.099989	160.124892	0.960064	0.959760	0.538942	37.779644	3.574252	1.820129	
1	1267.0	71.773608	22.993371	163.154329	0.947296	0.948353	0.798362	40.164593	3.121491	1.671902	
2	1145.0	70.219144	21.094339	157.982756	0.953811	0.959765	0.629121	38.181923	3.328815	1.734619	
3	1125.0	66.577686	21.756793	150.325902	0.945098	0.958262	0.732422	37.846988	3.060087	1.598473	
4	1242.0	72.805019	22.353390	164.066017	0.951700	0.953185	0.450000	39.766362	3.257001	1.724669	
...	...	...	...	...	...	...	...	...	...	...	...
395	1069.0	67.617931	20.409624	151.823376	0.953359	0.953613	0.437398	36.892995	3.313042	1.715891	
396	946.0	66.236131	18.486550	146.509668	0.960262	0.953629	0.441026	34.705686	3.582936	1.805642	
397	1037.0	64.750593	20.753102	143.597980	0.947246	0.983871	0.866332	36.336612	3.120044	1.582370	
398	911.0	63.400140	18.713809	141.254834	0.955445	0.945021	0.690152	34.057616	3.387880	1.742924	
399	1032.0	67.552151	19.778571	147.840620	0.956177	0.971751	0.822967	36.248906	3.415421	1.685381	

400 rows × 12 columns

Combine to CSV file

In [50]:
```
csv_file_path = 'Small_C1.csv'
df_Small_C1.to_csv(csv_file_path, index=False)
print(f"Data has been exported to {csv_file_path}") # print a message to confirm the export

csv_file_path = 'Small_Broke.csv'
df_Small_Broke.to_csv(csv_file_path, index=False)
print(f"Data has been exported to {csv_file_path}") # print a message to confirm the export
```

```
csv_file_path = 'Big_Broke.csv'
df_Big_Broke.to_csv(csv_file_path, index=False)
print(f"Data has been exported to {csv_file_path}") # print a message to confirm the export

csv_file_path = 'Head_Rice.csv'
df_Head_Rice.to_csv(csv_file_path, index=False)
print(f"Data has been exported to {csv_file_path}") # print a message to confirm the export

csv_file_path = 'Whole_Rice.csv'
df_Whole_Rice.to_csv(csv_file_path, index=False)
print(f"Data has been exported to {csv_file_path}") # print a message to confirm the export
```

```
Data has been exported to Small_C1.csv
Data has been exported to Small_Broke.csv
Data has been exported to Big_Broke.csv
Data has been exported to Head_Rice.csv
Data has been exported to Whole_Rice.csv
```

In [51]:

```
import pandas as pd

# List of CSV file names
csv_files = [
    "Small_C1.csv",
    "Small_Broke.csv",
    "Big_Broke.csv",
    "Head_Rice.csv",
    "Whole_Rice.csv",
]

# Create an empty DataFrame to store the combined data
combined_prop_data = pd.DataFrame()

# Loop through the CSV files and append their data to the combined_data DataFrame
for file in csv_files:
    df = pd.read_csv(file)  # Read each CSV file
    combined_prop_data = combined_prop_data.append(df, ignore_index=True)  # Append data to the combined DataFr

# Save the combined data to a new CSV file
combined_prop_data.to_csv("Training_Data.csv", index=False)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_15164\1823850606.py:18: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat in
stead.

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_15164\1823850606.py:18: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat in
stead.

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_15164\1823850606.py:18: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat in
stead.

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_15164\1823850606.py:18: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat in
stead.

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_15164\1823850606.py:18: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat in
stead.
```

In [52]:

```
Training_Data = pd.read_csv("Training_Data.csv")
```

In [53]:

```
Training_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   area               2000 non-null   float64
 1   major_axis_length  2000 non-null   float64
 2   minor_axis_length  2000 non-null   float64
 3   perimeter          2000 non-null   float64
 4   eccentricity       2000 non-null   float64
 5   solidity           2000 non-null   float64
 6   extent             2000 non-null   float64
 7   equiv_diameter     2000 non-null   float64
 8   aspect_ratio       2000 non-null   float64
 9   compactness        2000 non-null   float64
 10  roundness          2000 non-null   float64
 11  category           2000 non-null   object 
dtypes: float64(11), object(1)
memory usage: 187.6+ KB
```

In [54]:
```
pd.DataFrame(Training_Data)
```

Out[54]:

	area	major_axis_length	minor_axis_length	perimeter	eccentricity	solidity	extent	equiv_diameter	aspect_ratio	compactness
0	413.0	27.069175	20.234124	78.911688	0.664265	0.956019	0.611852	22.931374	1.337798	1.199838
1	431.0	29.341232	18.783145	77.597980	0.768240	0.966368	0.720736	23.425760	1.562104	1.111767
2	372.0	24.753551	19.302683	71.455844	0.626036	0.958763	0.808696	21.763389	1.282389	1.092252
3	455.0	29.248325	20.394311	82.325902	0.716798	0.953878	0.627586	24.069150	1.434141	1.185364
4	341.0	24.252606	19.119481	72.769553	0.615230	0.929155	0.644612	20.836859	1.268476	1.235763
...	...	...	...	...	...	...	...	...	...	...
1995	1069.0	67.617931	20.409624	151.823376	0.953359	0.953613	0.437398	36.892995	3.313042	1.715891
1996	946.0	66.236131	18.486550	146.509668	0.960262	0.953629	0.441026	34.705686	3.582936	1.805642
1997	1037.0	64.750593	20.753102	143.597980	0.947246	0.983871	0.866332	36.336612	3.120044	1.582370
1998	911.0	63.400140	18.713809	141.254834	0.955445	0.945021	0.690152	34.057616	3.387880	1.742924
1999	1032.0	67.552151	19.778571	147.840620	0.956177	0.971751	0.822967	36.248906	3.415421	1.685381

2000 rows × 12 columns

In [55]:
```
unique_categories = Training_Data['category'].unique()
print(unique_categories)
```

```
['Small Broke C1' 'Small Broke' 'Big Broke' 'Head rice' 'Whole Rice']
```

In [ ]: