In [58]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [59]:
```
# Load the os library
import os

# Load the request module
import urllib.request

# Import SSL which we need to setup for talking to the HTTPS server
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

# Create a directory
try:
    os.mkdir('medallion painting')

    #https://framemark.vam.ac.uk/collections/2006BH7789/full/1400,/0/defa

    for img_i in range(89, 96):

        # create a string using the current loop counter
        f = '%02d' % img_i

        # and get the url with that string appended the end
        url = 'https://framemark.vam.ac.uk/collections/2006BH77' + f + '/

        # We'll print this out to the console so we can see how far we've
        print(url, end='\r')

        # And now download the url to a location inside our new directory
        urllib.request.urlretrieve(url, os.path.join('medallion painting'
except:
    #os.rm('img_align_celeba')
    print("You may need to delete the existing 'medallion painting' folde
```

```
https://framemark.vam.ac.uk/collections/2006BH7795/full/1400,/0/default.
jpg
```

In [65]:
```
files = os.listdir('medallion painting')# img.<tab>
import matplotlib.pyplot as plt
import numpy as np

print(os.path.join('medallion painting', files[0]))
plt.imread(os.path.join('medallion painting', files[0]))

files = [os.path.join('medallion painting', file_i)
 for file_i in os.listdir('medallion painting')
 if '.jpg' in file_i]
```

```
medallion painting/89.jpg
```

In [66]:
```
imgs = [plt.imread(files[file_i])
        for file_i in range(7)]
data = np.array(imgs) # make 'data' = our numpy array
```

```
mean_img = np.percentile(data,50, axis=0) # This is the mean of the 'batc
plt.imshow(mean_img.astype(np.uint8))
```

```
/var/folders/yw/k75t7n390ts0pt4ymwsb9x_r0000gn/T/ipykernel_13158/9278590
34.py:3: VisibleDeprecationWarning: Creating an ndarray from ragged nest
ed sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays wi
th different lengths or shapes) is deprecated. If you meant to do this,
you must specify 'dtype=object' when creating the ndarray.
  data = np.array(imgs) # make 'data' = our numpy array
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[66], line 4
      1 imgs = [plt.imread(files[file_i])
      2         for file_i in range(7)]
      3 data = np.array(imgs) # make 'data' = our numpy array
----> 4 mean_img = np.percentile(data,50, axis=0) # This is the mean of the 'batch' channel
      5 plt.imshow(mean_img.astype(np.uint8))

File <__array_function__ internals>:180, in percentile(*args, **kwargs)

File ~/miniforge3/envs/coding2/lib/python3.10/site-packages/numpy/lib/function_base.py:4166, in percentile(a, q, axis, out, overwrite_input, method, keepdims, interpolation)
   4164 if not _quantile_is_valid(q):
   4165     raise ValueError("Percentiles must be in the range [0, 100]")
-> 4166 return _quantile_unchecked(
   4167     a, q, axis, out, overwrite_input, method, keepdims)

File ~/miniforge3/envs/coding2/lib/python3.10/site-packages/numpy/lib/function_base.py:4424, in _quantile_unchecked(a, q, axis, out, overwrite_input, method, keepdims)
   4416 def _quantile_unchecked(a,
   4417                         q,
   4418                         axis=None,
   (...)
   4421                         method="linear",
   4422                         keepdims=False):
   4423     """Assumes that q is in [0, 1], and is an ndarray"""
-> 4424     r, k = _ureduce(a,
   4425                     func=_quantile_ureduce_func,
   4426                     q=q,
   4427                     axis=axis,
   4428                     out=out,
   4429                     overwrite_input=overwrite_input,
   4430                     method=method)
   4431     if keepdims:
   4432         return r.reshape(q.shape + k)

File ~/miniforge3/envs/coding2/lib/python3.10/site-packages/numpy/lib/function_base.py:3725, in _ureduce(a, func, **kwargs)
   3722 else:
   3723     keepdim = (1,) * a.ndim
-> 3725 r = func(a, **kwargs)
   3726 return r, keepdim

File ~/miniforge3/envs/coding2/lib/python3.10/site-packages/numpy/lib/function_base.py:4593, in _quantile_ureduce_func(a, q, axis, out, overwrite_input, method)
   4591     else:
   4592         arr = a.copy()
-> 4593 result = _quantile(arr,
   4594                    quantiles=q,
   4595                    axis=axis,
   4596                    method=method,
```

```
   4597                     out=out)
   4598 return result

File ~/miniforge3/envs/coding2/lib/python3.10/site-packages/numpy/lib/fu
nction_base.py:4691, in _quantile(arr, quantiles, axis, method, out)
   4687 previous_indexes, next_indexes = _get_indexes(arr,
   4688                                                virtual_indexes,
   4689                                                values_count)
   4690 # --- Sorting
-> 4691 arr.partition(
   4692     np.unique(np.concatenate(([0, -1],
   4693                              previous_indexes.ravel(),
   4694                              next_indexes.ravel(),
   4695                              ))),
   4696     axis=DATA_AXIS)
   4697 if np.issubdtype(arr.dtype, np.inexact):
   4698     slices_having_nans = np.isnan(
   4699         take(arr, indices=-1, axis=DATA_AXIS)
   4700     )

ValueError: operands could not be broadcast together with shapes (1387,1
400,3) (1421,1400,3)
```

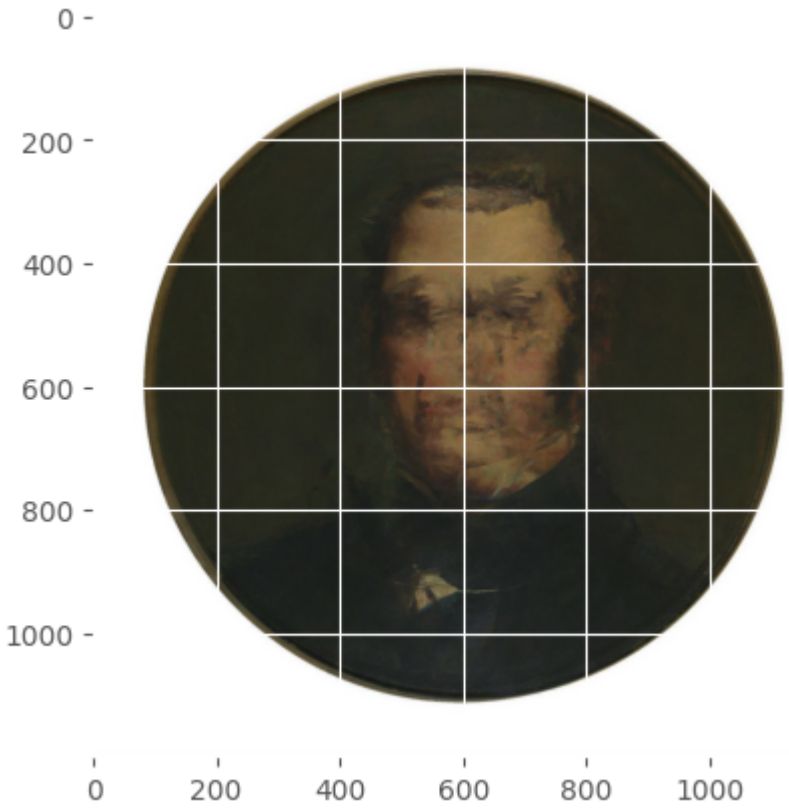
ValueError: operands could not be broadcast together with shapes (1387,1400,3) (1421,1400,3) Uniformity of image size using birme

In [67]:
```
files = os.listdir('medallion painting Uniformity')# img.<tab>
import matplotlib.pyplot as plt
import numpy as np

files = [os.path.join('medallion painting Uniformity', file_i)
 for file_i in os.listdir('medallion painting Uniformity')
 if '.jpg' in file_i]
```

In [83]:
```
imgs = [plt.imread(files[file_i])
        for file_i in range(7)]
data = np.array(imgs) # make 'data' = our numpy array
mean_img = np.median(data, axis=0) # This is the mean of the 'batch' chan
plt.imshow(mean_img.astype(np.uint8))
```

Out[83]:
```
<matplotlib.image.AxesImage at 0x123daa110>
```



In [ ]: