Full Pipeline (on Asbestos) - Oct 2018

Created: 31 Oct 2018
Last update: 1 Nov 2018

Goal: Run the full pipeline on the Asbestos set

---

1. Imports

In [1]:
```
# this will remove warnings messages
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

# import
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Normalizer
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.pipeline import Pipeline
from sklearn.metrics import silhouette_score

import imgutils
```

In [2]:
```
# Re-run this cell if you altered imgutils
import importlib
importlib.reload(imgutils)
```

Out[2]:
```
<module 'imgutils' from 'C:\\JADS\\SW\\Grad Proj\\sources\\imgutils.py'>
```

---

2. Data Definitions & Feature Specification

In [424]:
```
# Data:
datafolder = '../data/Asbestos_Aug30/SA_TileSet_Subset2_1K'
n_tiles_x = 2  # mostly for visualization
n_tiles_y = 2


# Features to use:
#feature_funcs = [imgutils.img_mean, imgutils.img_std, imgutils.img_median,
#                 imgutils.img_mode,
#                 imgutils.img_kurtosis, imgutils.img_skewness]
feature_funcs = [imgutils.img_std, imgutils.img_relstd, imgutils.img_mean,
                 imgutils.img_skewness,  imgutils.img_kurtosis, imgutils.img_mode, imgutils.img_range]
feature_names = imgutils.stat_names(feature_funcs)

# Size of the grid, specified as number of slices per image in x and y direction:
default_grid_x = 8
default_grid_y = default_grid_x
```

---

3. Import Data & Extract Features

In [425]:
```
# image import:
print("Scanning for images in '{}'...".format(datafolder))
df_imgfiles = imgutils.scanimgdir(datafolder, '.tif')
imgfiles = list(df_imgfiles['filename'])
print("# of images: {} \n".format(len(imgfiles)))

# feature extraction:
print("Feature extraction...")
print("- Slicing up images in {} x {} patches. ".format(default_grid_y, default_grid_x))
print("- Extract statistics from each slice: {} ".format(', '.join(feature_names)))
print("...working...", end='\r')
df = imgutils.slicestats(imgfiles, default_grid_y, default_grid_x, feature_funcs)
print("# slices extracted: ", len(df))
```

```
Scanning for images in '../data/Asbestos_Aug30/SA_TileSet_Subset2_1K'...
# of images: 4

Feature extraction...
- Slicing up images in 8 x 8 patches.
- Extract statistics from each slice: img_std, img_relstd, img_mean, img_skewness, img_kurtosis, img_mode, img_range
# slices extracted:  256
```

---

4. Machine Learning Pipeline

Hyper parameters

In [419]:
```
# data hyper-parameters
default_n_clusters = 3

# algorithm hyper-parameters:
kmeans_n_init = 10
```

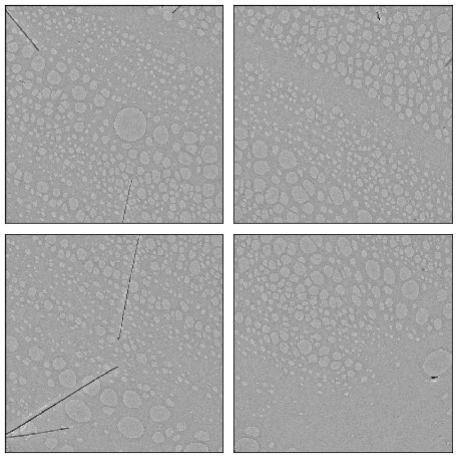
In [420]:
```
def run_ml_pipeline2(X, ml_name, ml_algorithm, standardize=True, use_pca=True, n_pca=None):

    # Setup algorithmic pipeline, including standardization
    pipeline = Pipeline([(ml_name, ml_algorithm)])

    # watch the order, pca should happen after scaling, but we insert at 0
    if (use_pca):
        pipeline.steps.insert(0,('pca', PCA(n_components=n_pca)))
    if (standardize):
        pipeline.steps.insert(0, ('scaling_{0}'.format(ml_name), StandardScaler()))

    # run the pipelines
    y = pipeline.fit_predict(X) # calls predict on last step to get the labels

    # report score:
    score = silhouette_score(X, y)

    return score, y
```

In [421]:
```
def run_ml_pipelines2(df_data, feature_cols, n_clusters, standardize=True, use_pca=True, n_pca=None):
    global kmeans_n_init

    X = df_data.loc[:,feature_cols]

    # Setup ML clustering algorithms:
    kmeans = KMeans(algorithm='auto', n_clusters=n_clusters, n_init=kmeans_n_init, init='k-means++')
    agglomerative =  AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean', linkage='complete')

    # run the pipelines
    print("Executing clustering pipelines...")
    score_kmeans, y_kmeans = run_ml_pipeline2(X, 'kmeans', kmeans, standardize = standardize, use_pca = use_pca, n_pca=n_pca)
    score_hier, y_hier = run_ml_pipeline2(X, 'hierarchical', agglomerative, standardize = standardize, use_pca = use_pca, n_pca=n_pca)
    print("Done\n")

    # collect data
    df_data['kmeans']=y_kmeans
    df_data['hierarchical']=y_hier

    # report results:
    print("\nClustering Scores:")
    print("K-means: ", score_kmeans)
    print("Hierarchical: ", score_hier)
```

In [422]:
```
run_ml_pipelines2(df, feature_names, default_n_clusters, standardize=True, use_pca=True)
```

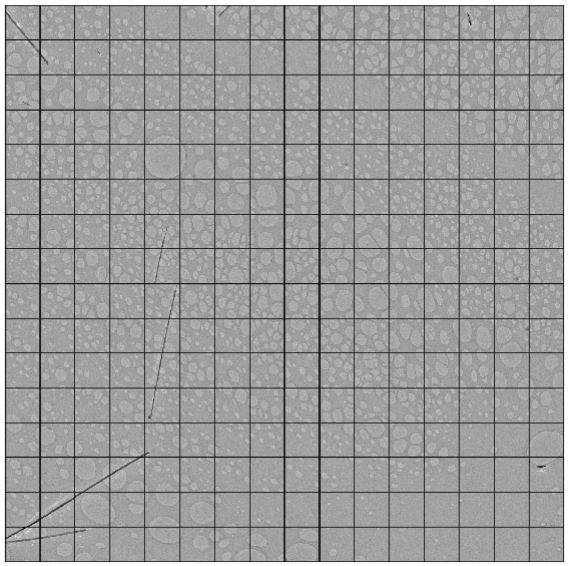
```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  0.8124487756829577
Hierarchical:  0.6750715209767348
```

In [427]:
```
imgutils.showimgset(imgfiles, 2,2, fig_size=(12, 8), relspacing=(0.05,0.05))
```



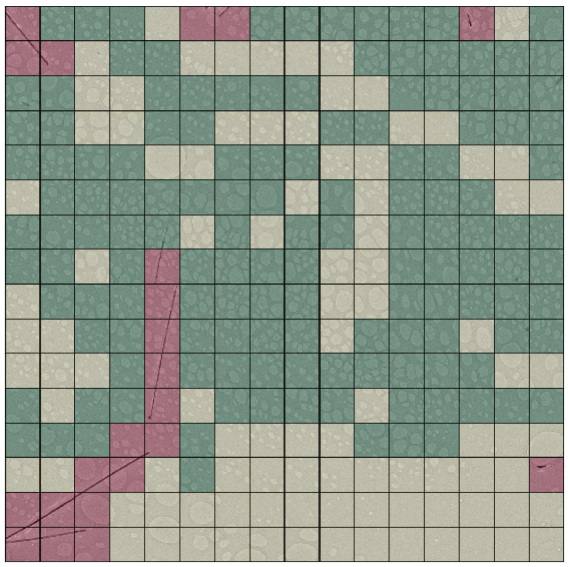
In [426]:
```
df['dummy'] = 0
imgutils.show_large_heatmap(df, 'dummy', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,10))
```

Heats from: dummy

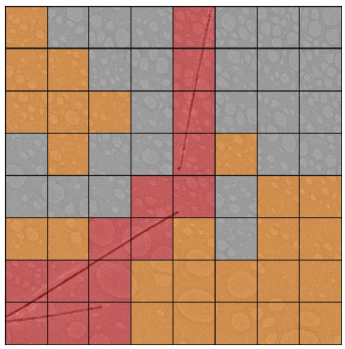


In [10]:
```
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,10))
```

Heats from: kmeans



In [11]:
```
imgfile1 = imgfiles[2]
#df_heats1 = df[df['filename']==imgfile1]['kmeans']
#heats = np.reshape(df_heats1.values, (default_grid_y, default_grid_x))

subimgs, heats = imgutils.getimgslices_fromdf(df, imgfile1, 'kmeans')
heats = heats / np.max(heats)
imgutils.showheatmap(subimgs, heats, cmapname='Set1', heatdepend_opacity = False)
```



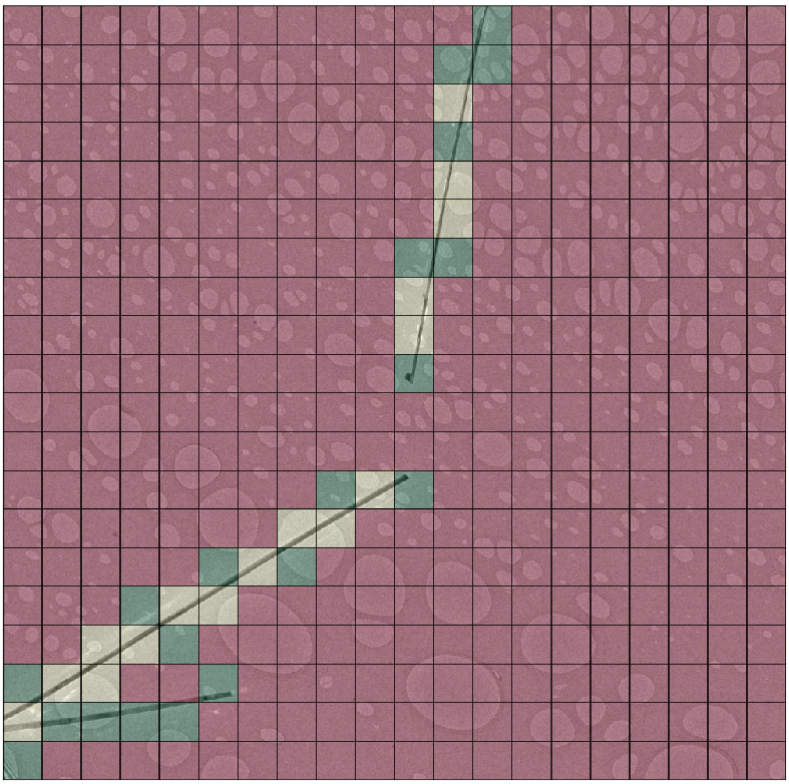
In [129]:
```
n_patches = 20
df2 = imgutils.slicestats([imgfile1], n_patches, n_patches, feature_funcs)
```

In [130]:
```
run_ml_pipelines2(df2, ['img_std', 'img_range'], 3, standardize=True, use_pca=True)
```

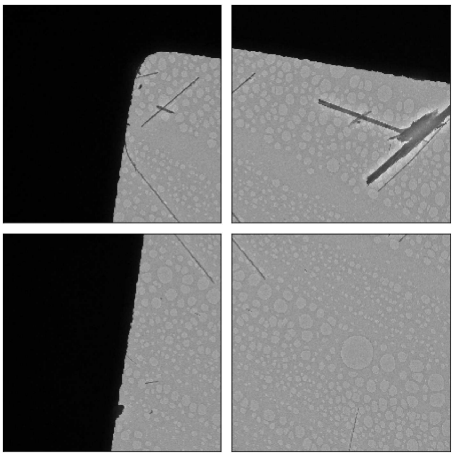
```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  0.843013588278119
Hierarchical:  0.8396362484112501
```

In [131]:
```
imgutils.show_large_heatmap(df2, 'kmeans', [imgfile1], n_rows=1, n_cols=1, fig_size=(14,14))
```

Heats from: kmeans



In [415]:
```
imgutils.showimgset(imgfiles, 2,2, fig_size=(12, 8), relspacing=(0.05,0.05))
```



For nice graphs, run it once more with two clusters and two levels of granularity

In [138]:
```
def change_clusternums(df, columnname, oldnew_dict):
    df[columnname].replace(oldnew_dict, inplace=True)

def swap_clusters(df, columnname, clust1, clust2):
    oldnew_dict = { clust1: clust2, clust2: clust1}
    df[columnname].replace(oldnew_dict, inplace=True)
```

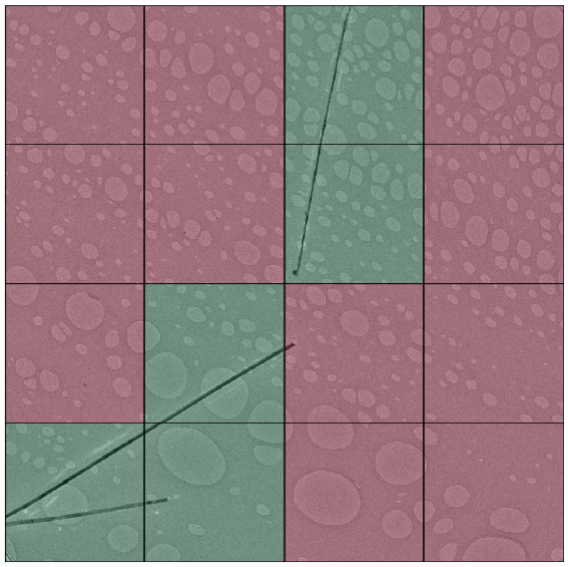
In [136]:
```
n_patches = 4
df_coarse = imgutils.slicestats([imgfile1], n_patches, n_patches, feature_funcs)
run_ml_pipelines2(df_coarse, feature_names, 2, standardize=True, use_pca=True)
```

```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  0.734126752570932
Hierarchical:  0.8292237708081216
```

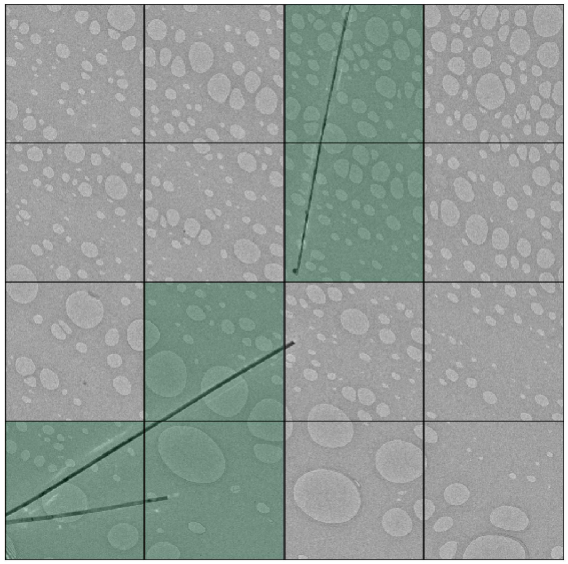
In [143]:
```
swap_clusters(df_coarse, 'kmeans', 0, 1)
```

In [144]:
```
imgutils.show_large_heatmap(df_coarse, 'kmeans', [imgfile1], n_rows=1, n_cols=1, fig_size=(10,10))
imgutils.show_large_heatmap(df_coarse, 'kmeans', [imgfile1], n_rows=1, n_cols=1, fig_size=(10,10), heatdependent_opacity=True)
```

Heats from: kmeans



Heats from: kmeans



Fine grained

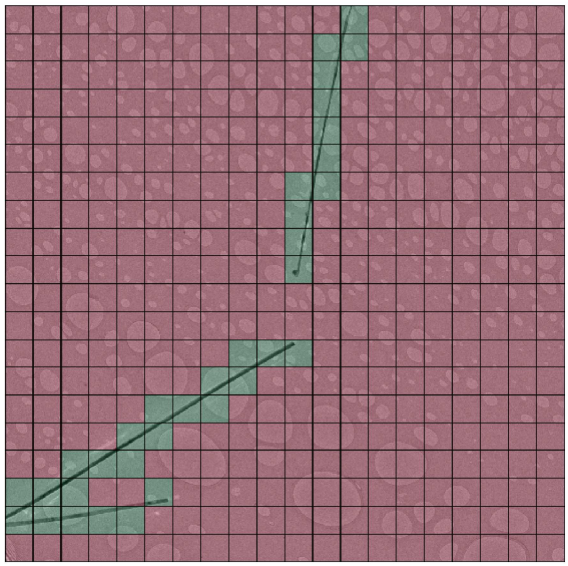
In [145]:
```
n_patches = 20
df_fine = imgutils.slicestats([imgfile1], n_patches, n_patches, feature_funcs)
run_ml_pipelines2(df_fine, feature_names, 2, standardize=True, use_pca=True)
```

```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  0.8634244460895917
Hierarchical:  0.7023326663792772
```

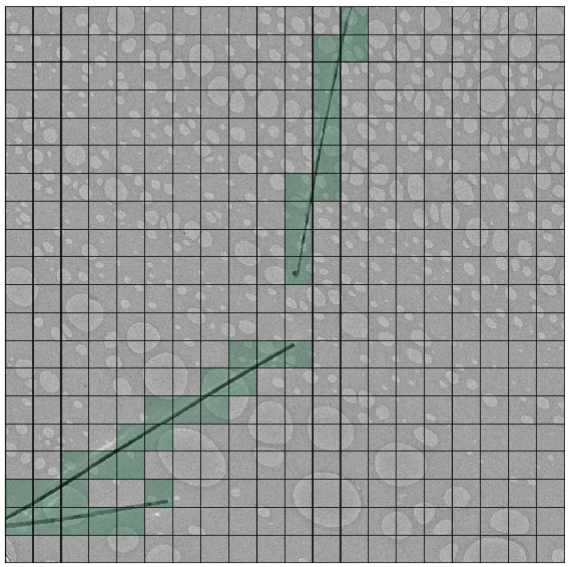
In [ ]:
```
#swap_clusters(df_fine, 'kmeans', 0, 1)
```

In [146]:
```
imgutils.show_large_heatmap(df_fine, 'kmeans', [imgfile1], n_rows=1, n_cols=1, fig_size=(10,10))
imgutils.show_large_heatmap(df_fine, 'kmeans', [imgfile1], n_rows=1, n_cols=1, fig_size=(10,10), heatdependent_opacity=True)
```

Heats from: kmeans



Heats from: kmeans



Use other scoring

Adjusting the ml_pipeline to use silhouette scoring based on it's last transformation: (later renamed the other ones to run_xxx2 to preserve them)

In [16]:
```
from sklearn.metrics import calinski_harabaz_score

def run_ml_pipeline(X, ml_name, ml_algorithm, standardize=True, use_pca=True, n_pca=None):

    # Setup 'manual' pipeline (not using sklearn pipeline as intermediates are needed)
    feat_data = X
    if (standardize):
        standardizer = StandardScaler()
        X_norm = standardizer.fit_transform(X)
        feat_data = X_norm
    if (use_pca):
        pca = PCA(n_components=n_pca)
        X_pca = pca.fit_transform(feat_data)
        feat_data = X_pca

    # run the pipelines
    y = ml_algorithm.fit_predict(feat_data) # calls predict oto get the labels

    # report score:
    #score = silhouette_score(feat_data, y)
    score = calinski_harabaz_score(feat_data,y)

    return score, y
```

In [17]:
```
def run_ml_pipelines(df_data, feature_cols, n_clusters, standardize=True, use_pca=True, n_pca=None):
    global kmeans_n_init

    X = df_data.loc[:,feature_cols]

    # Setup ML clustering algorithms:
    kmeans = KMeans(algorithm='auto', n_clusters=n_clusters, n_init=kmeans_n_init, init='k-means++')
    agglomerative =  AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean', linkage='complete')

    # run the pipelines
    print("Executing clustering pipelines...")
    score_kmeans, y_kmeans = run_ml_pipeline(X, 'kmeans', kmeans, standardize = standardize, use_pca = use_pca, n_pca=n_pca)
    score_hier, y_hier = run_ml_pipeline(X, 'hierarchical', agglomerative, standardize = standardize, use_pca = use_pca, n_pca=n_pca)
    print("Done\n")

    # collect data
    df_data['kmeans']=y_kmeans
    df_data['hierarchical']=y_hier

    # report results:
    print("\nClustering Scores:")
    print("K-means: ", score_kmeans)
    print("Hierarchical: ", score_hier)
```

In [18]:
```
run_ml_pipelines(df, feature_names, default_n_clusters, standardize=True, use_pca=True)
```

```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  194.26577788719692
Hierarchical:  147.09838838769846
```

More consistent with previous results and imo it is indeed better to assess the algorithm on how good it could cluster after all pre-processing

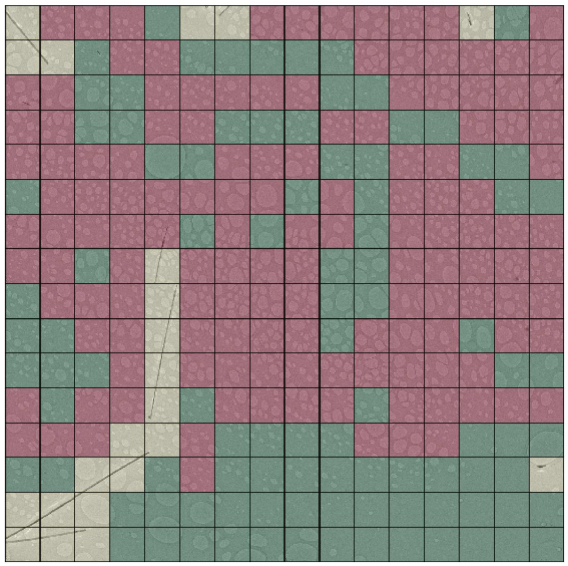
5. Visualize results

In [ ]:
```
imgutils.showimgset(imgfiles, 2,2, fig_size=(12, 8), relspacing=(0.05,0.05))
```

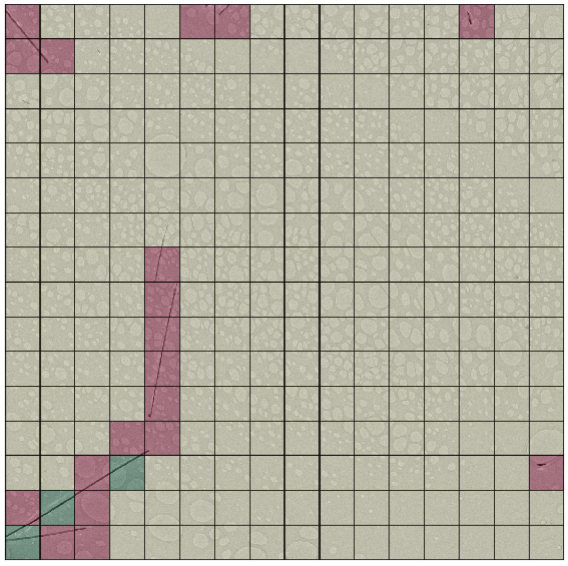
In [19]:
```
run_ml_pipelines(df, feature_names, default_n_clusters, standardize=True, use_pca=True)
s = (12,10)
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=s)
imgutils.show_large_heatmap(df, 'hierarchical', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=s)
```

```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  194.26577788719692
Hierarchical:  147.09838838769846
```

Heats from: kmeans



Heats from: hierarchical



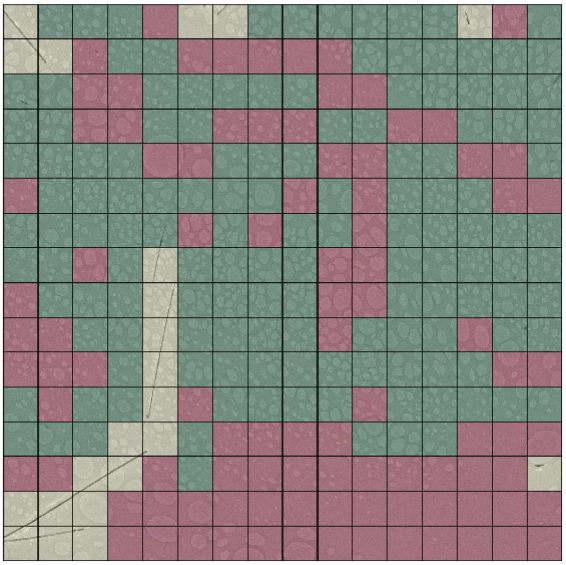
Here Hierarchical outperforms kmeans

Run it again without PCA and/pr normalization compare results

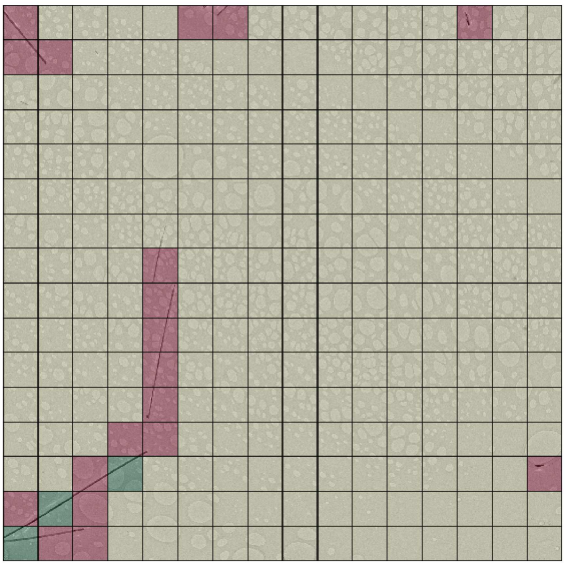
In [20]:
```
run_ml_pipelines(df, feature_names, default_n_clusters, standardize=True, use_pca=False)
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=s)
imgutils.show_large_heatmap(df, 'hierarchical', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=s)
```

```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  194.2657778871972
Hierarchical:  147.09838838769858
```

Heats from: kmeans



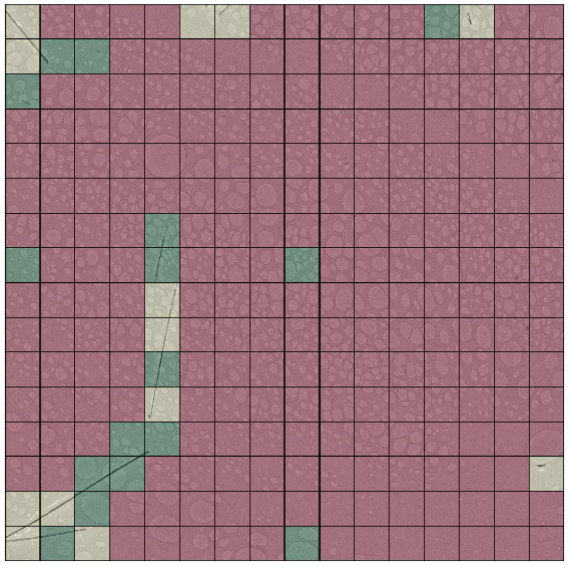
Heats from: hierarchical



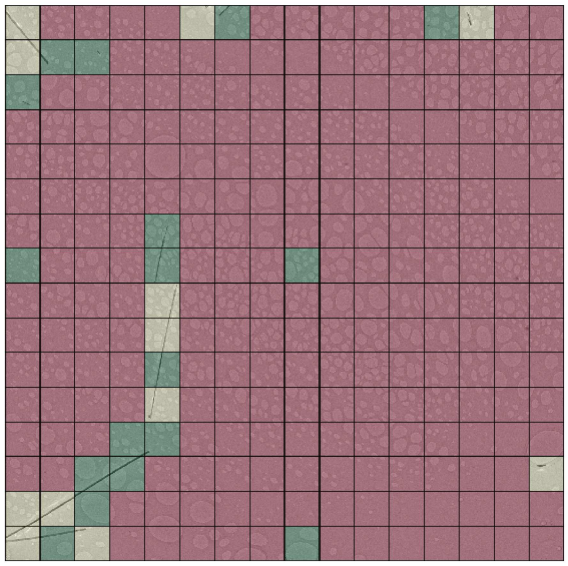
In [21]:
```
run_ml_pipelines(df, feature_names, default_n_clusters, standardize=False, use_pca=False)
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=s)
imgutils.show_large_heatmap(df, 'hierarchical', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=s)
```

```
Executing clustering pipelines...
Done


Clustering Scores:
K-means:  1400.6796383181056
Hierarchical:  1395.1250460779431
```

Heats from: kmeans



Heats from: hierarchical



On this dataset, not much difference between hierarchical and pca, with or without normalization

6. Combine import and pipeline:

In [22]:
```
def import_data(imagefolder):
    df_imgfiles = imgutils.scanimgdir(imagefolder, '.tif')
    return list(df_imgfiles['filename'])

def extract_features(imgfiles, feature_funcs, n_grid_rows, n_grid_cols):
    df = imgutils.slicestats(imgfiles, n_grid_rows, n_grid_cols, feature_funcs)
    feature_names = imgutils.stat_names(feature_funcs)
    return df, feature_names
```

In [23]:
```
def run_kmeans_pipeline(df_data, feature_cols, n_clusters, standardize=True, use_pca=True, n_pca= None):
    global kmeans_n_init

    ml_name="kmeans"
    ml_algorithm = KMeans(algorithm='auto', n_clusters=n_clusters, n_init=kmeans_n_init, init='k-means++')

    X = df_data.loc[:,feature_cols]
    score, y = run_ml_pipeline(X, ml_name, ml_algorithm, standardize = standardize, use_pca = use_pca, n_pca=n_pca)
    df_data[ml_name]= y

    return score

def run_hierarchical_pipeline(df_data, feature_cols, n_clusters, standardize=True, use_pca=True, n_pca=None):

    ml_name="hierarchical"
    ml_algorithm =  AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean', linkage='complete')

    X = df_data.loc[:,feature_cols]
    score, y = run_ml_pipeline(X, ml_name, ml_algorithm, standardize = standardize, use_pca = use_pca, n_pca=n_pca)
    df_data[ml_name]= y

    return score
```

In [24]:
```
def run_fullpipeline(imagefolder, n_image_rows, n_image_cols,
                     n_grid_rows, n_grid_cols, feature_funcs, n_clusters, fig_size=(8,6), return_df = False):
    """
    Run the full pipeline from import to visualization.
    """
    print("Working...\r")
    imgfiles = import_data(imagefolder)
    df, feature_names = extract_features(imgfiles, feature_funcs, n_grid_rows, n_grid_cols)
    print(feature_names)
    score_kmeans = run_kmeans_pipeline(df, feature_names, n_clusters, standardize=True, use_pca=True )
    score_hier = run_hierarchical_pipeline(df, feature_names, n_clusters, standardize=False, use_pca=False)

    print('Results:')
    print('Score k-means:', score_kmeans)
    print('Score hierarchical:', score_hier)

    print('Visualizing...')
    imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_image_rows, n_cols=n_image_cols, fig_size=fig_size)
    imgutils.show_large_heatmap(df, 'hierarchical', imgfiles, n_rows=n_image_rows, n_cols=n_image_cols, fig_size=fig_size)

    if return_df: return df
```

In [25]:
```
def run_fullpipeline_kmeans(imagefolder, n_image_rows, n_image_cols,
                     n_grid_rows, n_grid_cols, feature_funcs, n_clusters, fig_size=(8,6), return_df = False):
    """
    Run the full pipeline from import to visualization.
    """
    print("Working...\r")
    imgfiles = import_data(imagefolder)
    df, feature_names = extract_features(imgfiles, feature_funcs, n_grid_rows, n_grid_cols)
    print(feature_names)
    score_kmeans = run_kmeans_pipeline(df, feature_names, n_clusters, standardize=True, use_pca=True )

    print('Results:')
    print('Score k-means:', score_kmeans)

    print('Visualizing...')
    imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_image_rows, n_cols=n_image_cols, fig_size=fig_size)

    if return_df: return df
```

In [26]:
```
def run_fullpipeline_hierarchical(imagefolder, n_image_rows, n_image_cols,
                     n_grid_rows, n_grid_cols, feature_funcs, n_clusters, fig_size=(8,6), return_df = False):
    """
    Run the full pipeline from import to visualization.
    """
    print("Working...\r")
    imgfiles = import_data(imagefolder)
    df, feature_names = extract_features(imgfiles, feature_funcs, n_grid_rows, n_grid_cols)
    print(feature_names)
    score_hier = run_hierarchical_pipeline(df, feature_names, n_clusters, standardize=False, use_pca=False)

    print('Results:')
    print('Score hierarchical:', score_hier)

    print('Visualizing...')
    imgutils.show_large_heatmap(df, 'hierarchical', imgfiles, n_rows=n_image_rows, n_cols=n_image_cols, fig_size=fig_size)

    if return_df: return df
```

7. Try it out with different combinations of slices on a harder variant

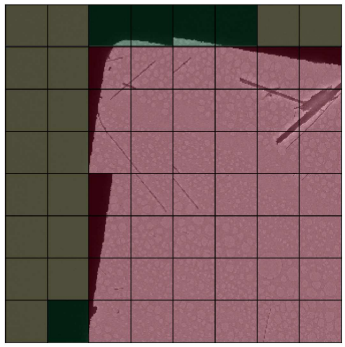
In [411]:
```
datafolder = '../data/Asbestos_Aug30/SA_TileSet_Subset1_1K'
```

4x4 - 3 clusters

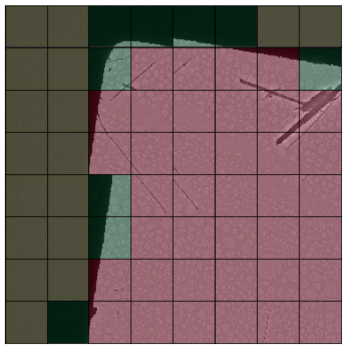
In [28]:
```
run_fullpipeline(datafolder, n_tiles_y, n_tiles_x, 4, 4, feature_funcs, 3)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 46.95791351238035
Score hierarchical: 178.32114103133864
Visualizing...
```

Heats from: kmeans



Heats from: hierarchical

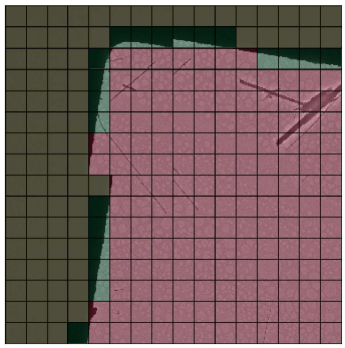


8x8, 3 clusters

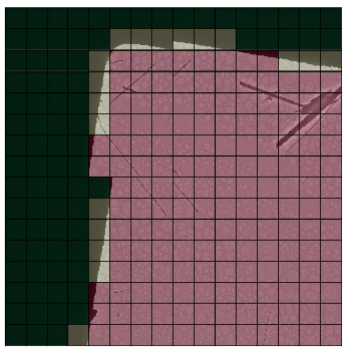
In [29]:
```
run_fullpipeline(datafolder, n_tiles_y, n_tiles_x, 8, 8, feature_funcs, 3)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 209.95444488932483
Score hierarchical: 1064.6665037184332
Visualizing...
```

Heats from: kmeans



Heats from: hierarchical

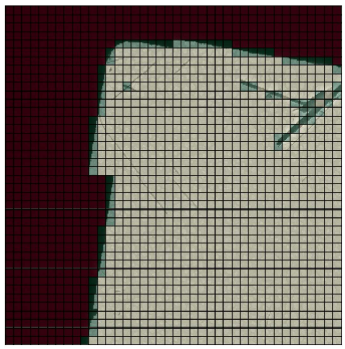


20x20, 3 clusters

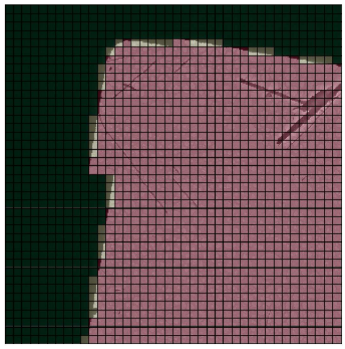
In [30]:
```
run_fullpipeline(datafolder, n_tiles_y, n_tiles_x, 20, 20, feature_funcs, 3)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 1279.6646382135298
Score hierarchical: 10242.284159797608
Visualizing...
```

Heats from: kmeans



Heats from: hierarchical



---

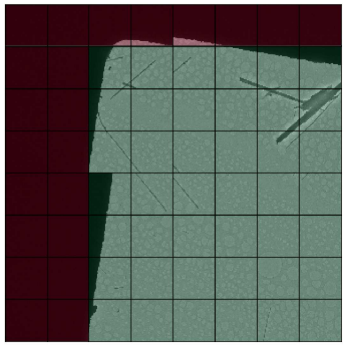
8. Try it out with different number of clusters

2 clusters (4x4 , 10x10, 20x20)

In [31]:
```
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 4, 4, feature_funcs, 2)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 35.316084968096696
Visualizing...
```

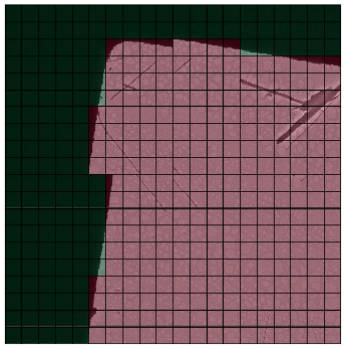
Heats from: kmeans



In [32]:
```
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 10, 10, feature_funcs, 2)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 240.14764120891513
Visualizing...
```

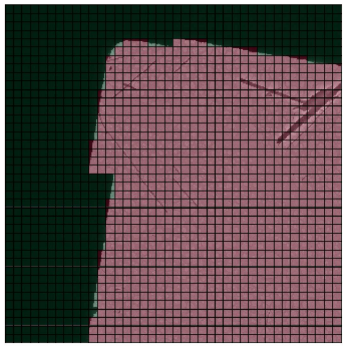
Heats from: kmeans



In [33]:
```
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 20, 20, feature_funcs, 2)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 1045.7229744130732
Visualizing...
```

Heats from: kmeans



Look again at hierarchical with scaling and pca on:

In [34]:
```
print(feature_funcs)
```

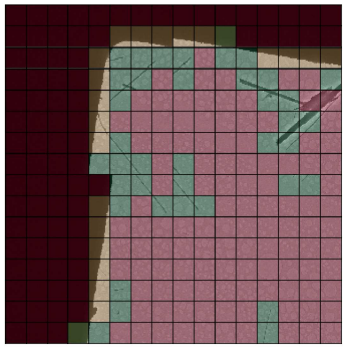
```
[<function img_std at 0x000000000B8C68C8>, <function img_relstd at 0x000000000B8C6840>, <function img_mean at 0x000000000B8C6268>, <function img_skewness at 0x000000000
B8C71E0>, <function img_kurtosis at 0x000000000B8C7158>, <function img_mode at 0x000000000B8C6F28>, <function img_range at 0x000000000B8C6400>]
```

no difference, all not good. Let's try if we drop the mean and mode (see if ignores the black)

In [35]:
```
try_funcs =  [imgutils.img_std, imgutils.img_relstd, imgutils.img_skewness,imgutils.img_kurtosis]
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 8, 8, try_funcs , 4)
```

```
Working...
['img_std', 'img_relstd', 'img_skewness', 'img_kurtosis']
Results:
Score k-means: 260.0707747939692
Visualizing...
```

Heats from: kmeans

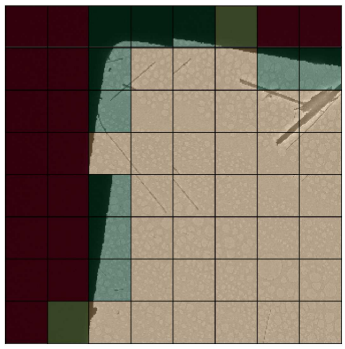


4 clusters (4x4, 10x10, 20x20)

In [36]:
```
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 4, 4, feature_funcs, 4)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 81.19023950894271
Visualizing...
```

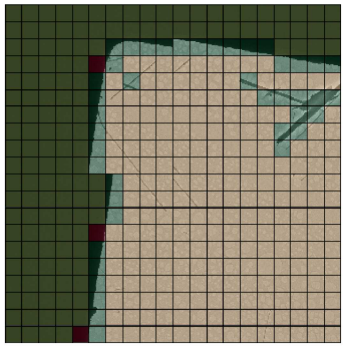
Heats from: kmeans



In [37]:
```
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 10, 10, feature_funcs, 4)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 392.4940373626443
Visualizing...
```

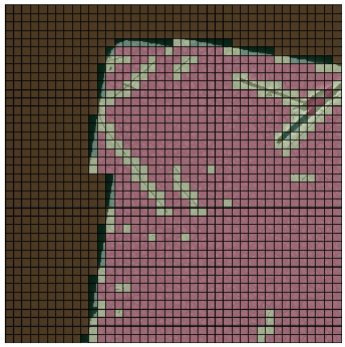
Heats from: kmeans



In [38]:
```
run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 20, 20, feature_funcs, 4)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 1512.787462954149
Visualizing...
```

Heats from: kmeans

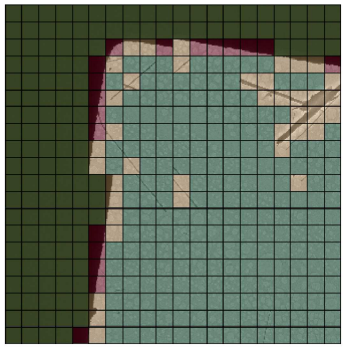


with smaller grid and 4 clusters, it starts to make sense

In [39]:
```
run_fullpipeline_hierarchical(datafolder, n_tiles_y, n_tiles_x, 10, 10, feature_funcs, 4)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score hierarchical: 3599.586258560254
Visualizing...
```

Heats from: hierarchical



AGAIN, THE BLACK TILES NEGATIVELY IMPACT RESULTS

Try two-step pipeline

step 1: filter out black tiles

step 2: cluster remaining tiles

Parametrize:

In [273]:
```
n_clusters_step1 = 3
n_clusters_step2_kmeans = 2
n_clusters_step2_hierarchical = 2

n_patches_x = 10
n_patches_y = 10
```

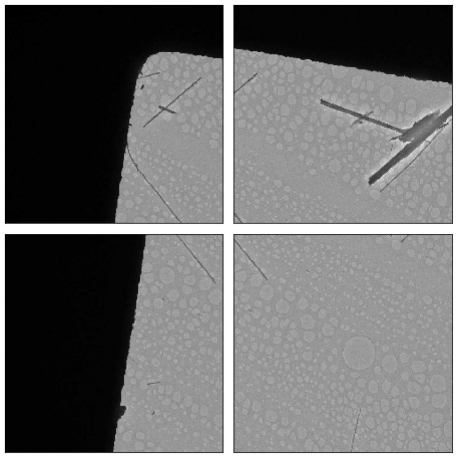
Feature extract

In [275]:
```
def change_clusternums(df, columnname, oldnew_dict):
    df[columnname].replace(oldnew_dict, inplace=True)

def swap_clusters(df, columnname, clust1, clust2):
    oldnew_dict = { clust1: clust2, clust2: clust1}
    df[columnname].replace(oldnew_dict, inplace=True)
```

In [234]:
```
# reset
df = df.drop(columns=['kmeans'])
df2 = None
df3 = None
```

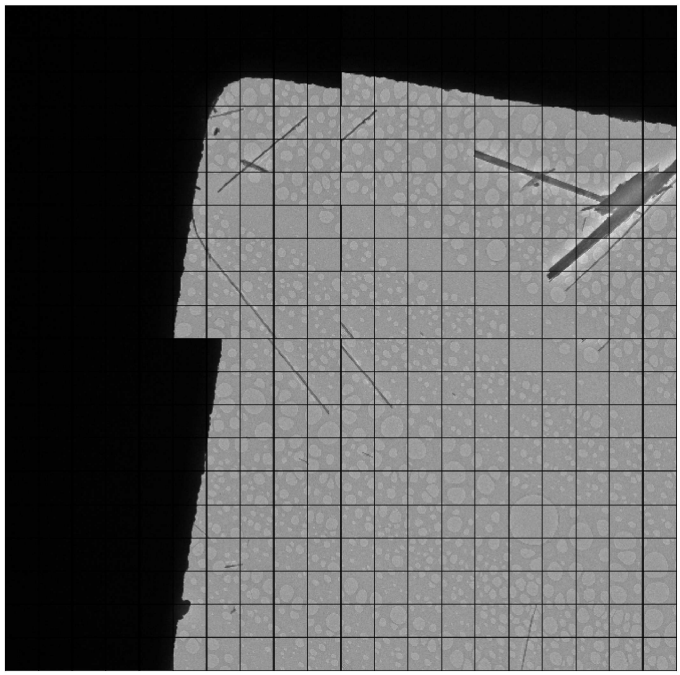
In [413]:
```
imgfiles = import_data(datafolder)
df, feature_names = extract_features(imgfiles, feature_funcs, n_patches_y, n_patches_x)
```

In [414]:
```
imgutils.showimgset(imgfiles, 2,2, fig_size=(12, 8), relspacing=(0.05,0.05))
```

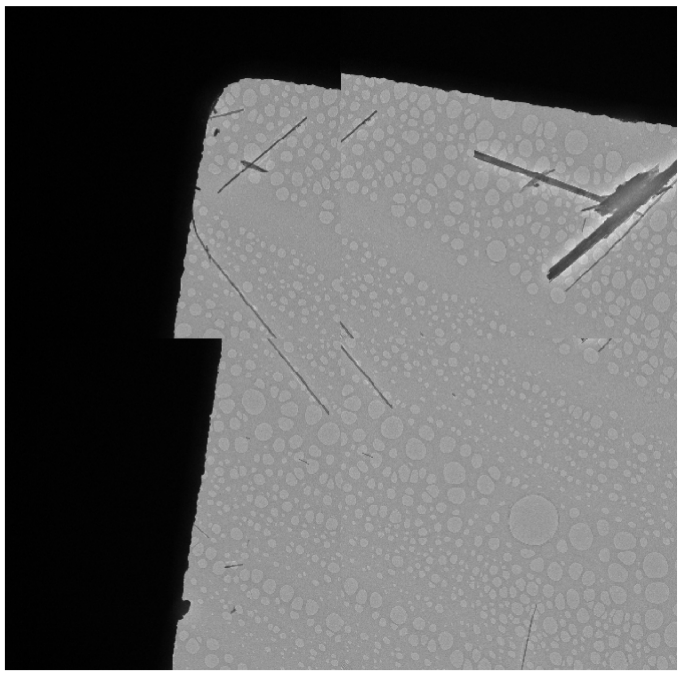


In [272]:
```
df['dummy']=0
imgutils.show_large_heatmap(df, 'dummy', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,12))
imgutils.show_large_heatmap(df, 'dummy', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,12), no_borders=True)
```

Heats from: dummy



Heats from: dummy



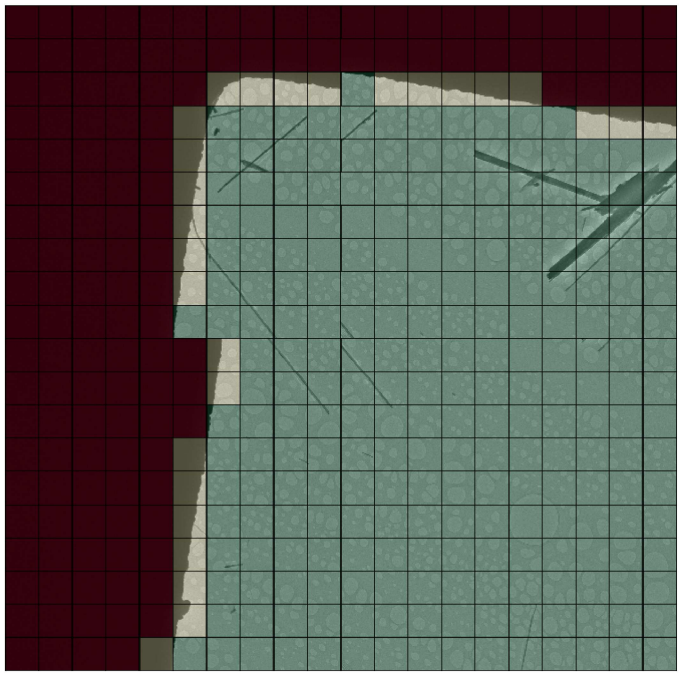
Step 1: filter-out black tiles

In [263]:
```
_ = run_kmeans_pipeline(df, feature_names, n_clusters_step1, standardize=True, use_pca=True )
```

In [267]:
```
#swap_clusters(df,'kmeans', 2, 1)
```

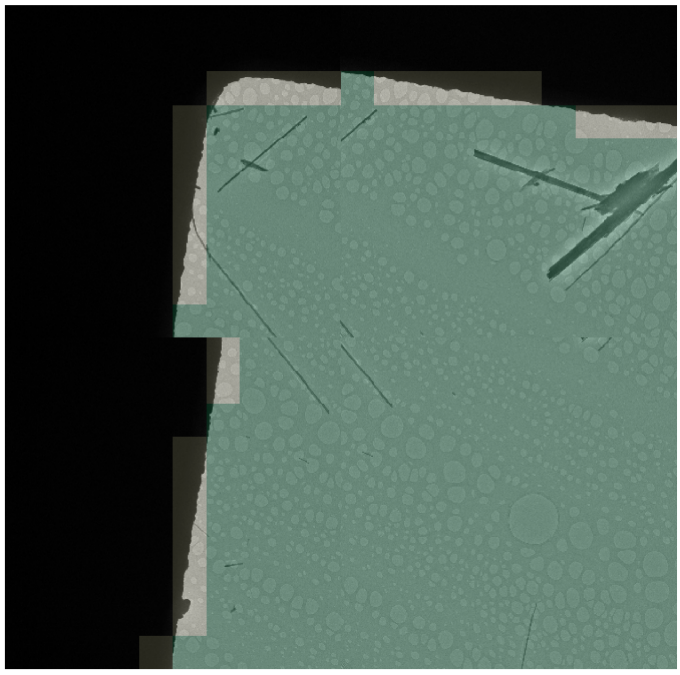
In [268]:
```
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,12))
```

Heats from: kmeans



In [270]:
```
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,12), heatdependent_opacity=True, no_borders=True)
```

Heats from: kmeans



In [240]:
```
df['kmeans'].value_counts()
```

Out[240]:
```
1    236
2    137
0     27
Name: kmeans, dtype: int64
```

In [241]:
```
# cat_select = 1
# we know for this it's the biggest set
i_max_count = df['kmeans'].value_counts()
cat_select = i_max_count.index[0]
print(cat_select)
```

```
1
```

In [242]:
```
df2 = df[df['kmeans']==cat_select]
df2.head(3)
```

Out[242]:

	filename	s_y	s_x	n_y	n_x	alias	img_std	img_relstd	img_mean	img_skewness	img_kurtosis	img_mode	img_range	kmeans
36	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	3	6	10	10	img0_3-6	748.522702	0.255418	2930.5758	-2.073972	3.524249	3073.785156	3656.0	1
37	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	3	7	10	10	img0_3-7	196.399301	0.062737	3130.4967	-0.710961	5.250237	3075.895508	2042.0	1
38	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	3	8	10	10	img0_3-8	403.815677	0.131236	3077.0134	-2.400551	7.167645	3069.577148	2722.0	1

In [243]:
```
score_kmeans = run_kmeans_pipeline(df2, feature_names, n_clusters_step2_kmeans, standardize=True, use_pca=True )
score_hierarch = run_hierarchical_pipeline(df2, feature_names, n_clusters_step2_hierarchical, standardize=False, use_pca=False )
```

In [244]:
```
df2['kmeans'].value_counts()
```

Out[244]:
```
0    190
1     46
Name: kmeans, dtype: int64
```

In [245]:
```
df2['hierarchical'].value_counts()
```

Out[245]:
```
0    206
1     30
Name: hierarchical, dtype: int64
```

In [246]:
```
df2=df2.rename(columns = {'kmeans':'kmeans2', 'hierarchical':'hierarchical2'})
```

In [247]:
```
df2.head(3)
```

Out[247]:

	filename	s_y	s_x	n_y	n_x	alias	img_std	img_relstd	img_mean	img_skewness	img_kurtosis	img_mode	img_range	kmeans2	hierarcl
36	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	3	6	10	10	img0_3-6	748.522702	0.255418	2930.5758	-2.073972	3.524249	3073.785156	3656.0	1	1
37	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	3	7	10	10	img0_3-7	196.399301	0.062737	3130.4967	-0.710961	5.250237	3075.895508	2042.0	0	0
38	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	3	8	10	10	img0_3-8	403.815677	0.131236	3077.0134	-2.400551	7.167645	3069.577148	2722.0	1	0

In [248]:
```
df3 = df.merge(df2, 'left')
```

In [249]:
```
df3.head(3)
```

Out[249]:

	filename	s_y	s_x	n_y	n_x	alias	img_std	img_relstd	img_mean	img_skewness	img_kurtosis	img_mode	img_range	kmeans	kmeans2	hi
0	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	0	0	10	10	img0_0-0	14.280093	0.418480	34.1227	0.475729	0.267368	30.925781	104.0	2	NaN	Na
1	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	0	1	10	10	img0_0-1	15.121837	0.396860	38.1027	0.371037	0.040804	30.939941	115.0	2	NaN	Na
2	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	0	2	10	10	img0_0-2	15.516127	0.384034	40.4020	0.387301	0.073373	39.863281	104.0	2	NaN	Na

In [250]:
```
df3['kmeans2'].fillna(value=-1, inplace=True)
```

In [251]:
```
df3['hierarchical2'].fillna(value=-1, inplace=True)
```

In [252]:
```
df3.head(3)
```

Out[252]:

	filename	s_y	s_x	n_y	n_x	alias	img_std	img_relstd	img_mean	img_skewness	img_kurtosis	img_mode	img_range	kmeans	kmeans2	hi
0	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	0	0	10	10	img0_0-0	14.280093	0.418480	34.1227	0.475729	0.267368	30.925781	104.0	2	-1.0	-1
1	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	0	1	10	10	img0_0-1	15.121837	0.396860	38.1027	0.371037	0.040804	30.939941	115.0	2	-1.0	-1
2	..\data\Asbestos_Aug30\SA_TileSet_Subset1_1K\T...	0	2	10	10	img0_0-2	15.516127	0.384034	40.4020	0.387301	0.073373	39.863281	104.0	2	-1.0	-1

In [253]:
```
df3['heats']=df3['kmeans2']+1
```

In [254]:
```
df3['heats'].value_counts()
```

Out[254]:
```
1.0    190
0.0    164
2.0     46
Name: heats, dtype: int64
```

In [255]:
```
# make the whole 2 clusters only
df3['heats'].replace({1:0}, inplace=True)
df3['heats'].value_counts()
```

Out[255]:
```
0.0    354
2.0     46
Name: heats, dtype: int64
```

In [256]:
```
df3['heats2']=df3['hierarchical2']+1
```

In [257]:
```
df3['heats2'].value_counts()
```

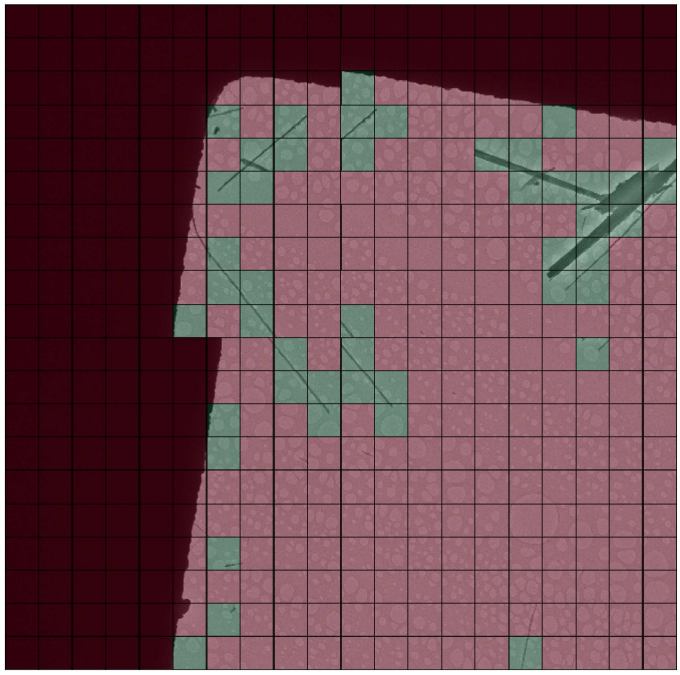
Out[257]:
```
1.0    206
0.0    164
2.0     30
Name: heats2, dtype: int64
```

In [258]:
```
# make the whole 2 clusters only
df3['heats2'].replace({1:0}, inplace=True)
df3['heats2'].value_counts()
```

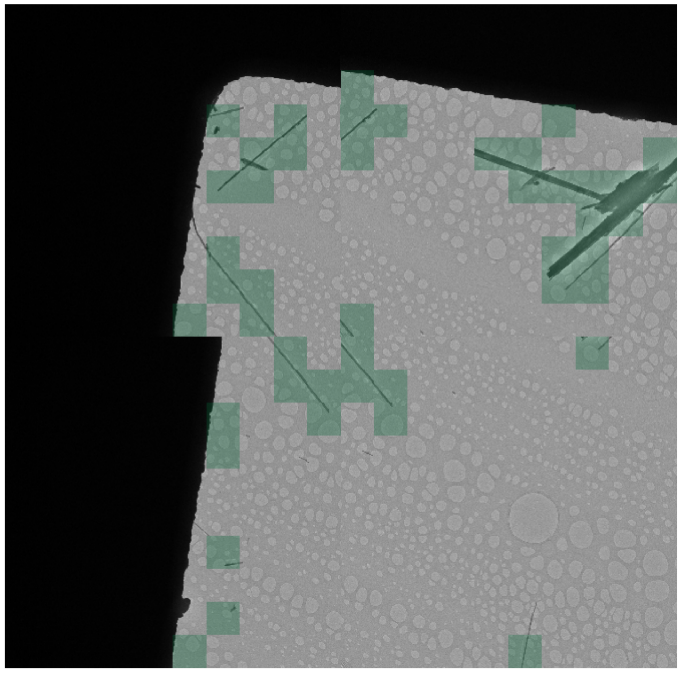
Out[258]:
```
0.0    370
2.0     30
Name: heats2, dtype: int64
```

In [259]:
```
imgutils.show_large_heatmap(df3, 'heats', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(16,12))
imgutils.show_large_heatmap(df3, 'heats', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(16,12), heatdependent_opacity=True, no_borders=True)
```

Heats from: heats

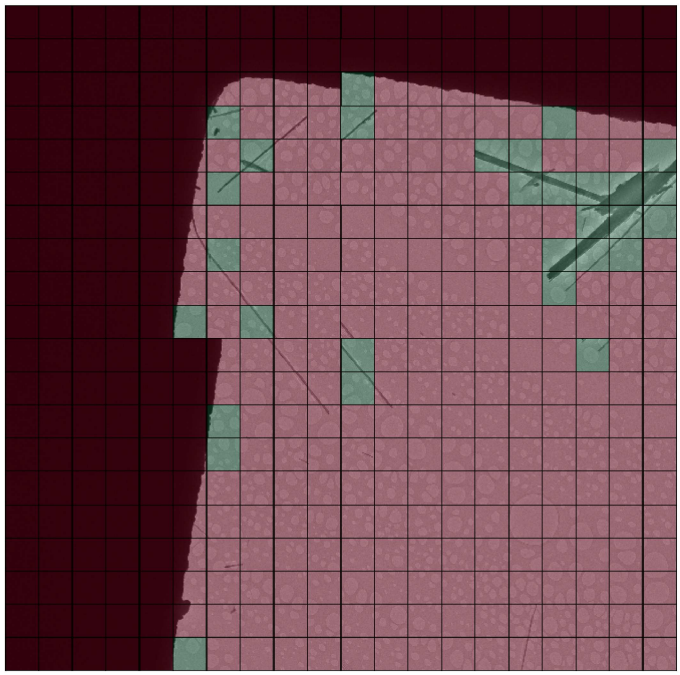


Heats from: heats



In [99]:
```
imgutils.show_large_heatmap(df3, 'heats2', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(16,12))
```

Heats from: heats2

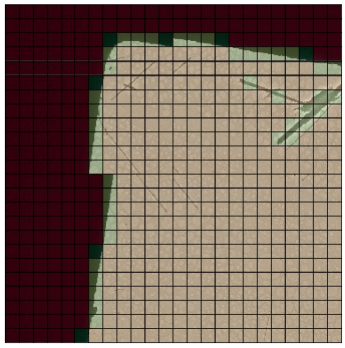


Alternative: run one with 4 clusters and small patch-size

In [285]:
```
df_direct4 = run_fullpipeline_kmeans(datafolder, n_tiles_y, n_tiles_x, 12, 12, feature_funcs, 4, return_df = True)
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 581.8925933285442
Visualizing...
```

Heats from: kmeans

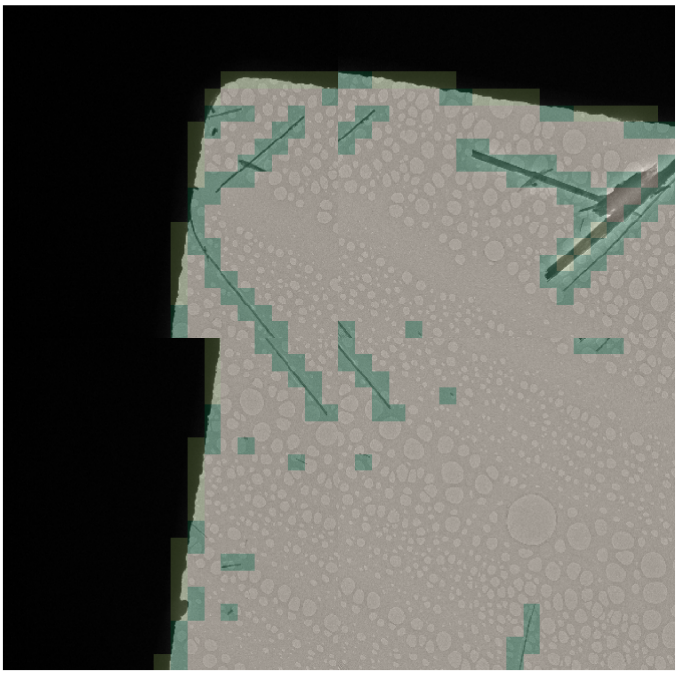


In [278]:
```
#swap_clusters(df_direct4, 'kmeans', 1,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-278-89b1c1e2e43a> in <module>()
----> 1 swap_clusters(df_direct4, 'kmeans', 1,2)

<ipython-input-275-6af1cbf3e9c4> in swap_clusters(df, columnname, clust1, clust2)
      4 def swap_clusters(df, columnname, clust1, clust2):
      5     oldnew_dict = { clust1: clust2, clust2: clust1}
----> 6     df[columnname].replace(oldnew_dict, inplace=True)

TypeError: 'NoneType' object is not subscriptable
```

In [283]:
```
imgutils.show_large_heatmap(df_direct4, 'kmeans', imgfiles, n_rows=n_tiles_y, n_cols=n_tiles_x, fig_size=(12,12), heatdependent_opacity=True, no_borders=True)
```

Heats from: kmeans



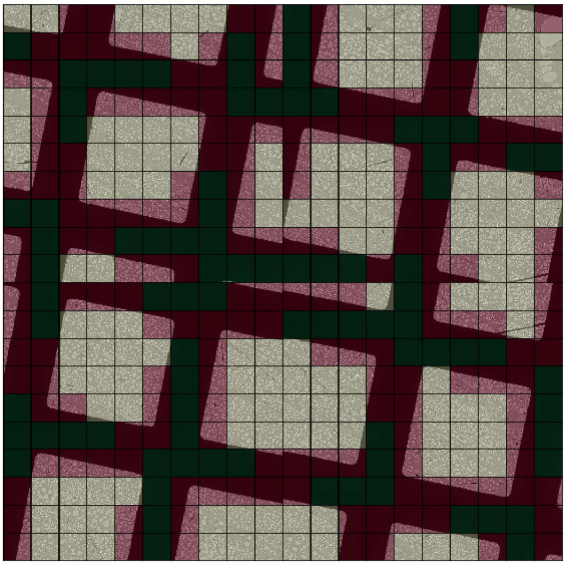
Yet another set

In [64]:
```
datafolder_lm = '../data/Asbestos_Aug30/LM_TileSet_Subset_1K'
imgfiles_lm = import_data(datafolder_lm)
```

In [65]:
```
run_fullpipeline_kmeans(datafolder_lm, n_tiles_y, n_tiles_x, 10, 10, feature_funcs, 3, fig_size=(12,10))
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 316.5090647082426
Visualizing...
```

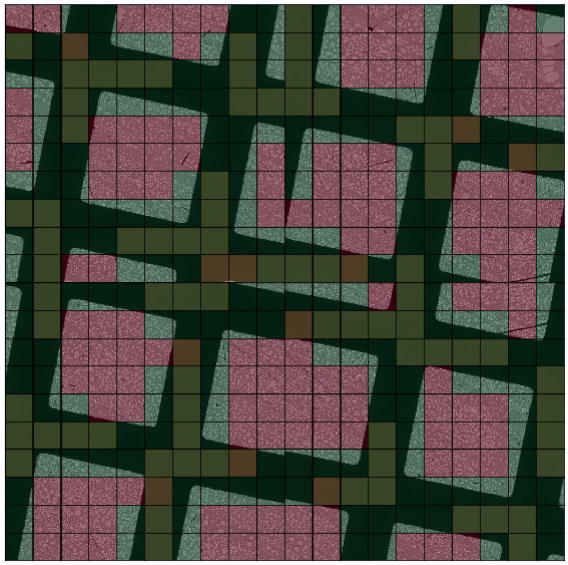
Heats from: kmeans



In [66]:
```
run_fullpipeline_kmeans(datafolder_lm, n_tiles_y, n_tiles_x, 10, 10, feature_funcs, 4,fig_size=(12,10))
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 509.5617385679495
Visualizing...
```

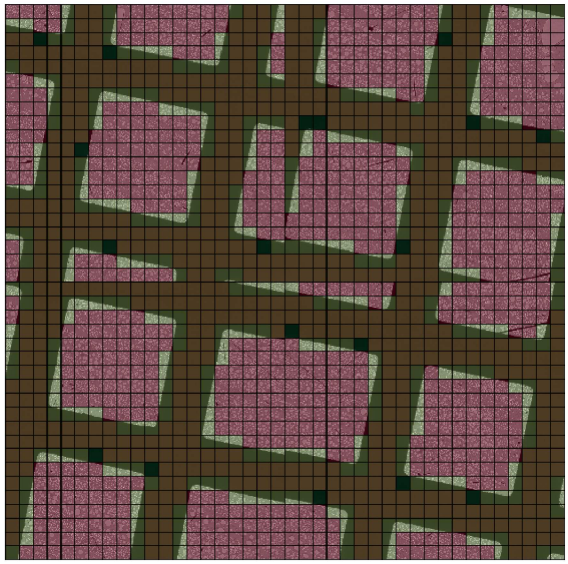
Heats from: kmeans



In [67]:
```
run_fullpipeline_kmeans(datafolder_lm, n_tiles_y, n_tiles_x, 20, 20, feature_funcs, 4, fig_size=(12,10))
```

```
Working...
['img_std', 'img_relstd', 'img_mean', 'img_skewness', 'img_kurtosis', 'img_mode', 'img_range']
Results:
Score k-means: 2809.532637641332
Visualizing...
```

Heats from: kmeans



2 step approach on this set

Parametrize

In [393]:
```
n_clusters_step1 = 3
n_clusters_step2_kmeans = 2
n_clusters_step2_hierarchical = 2

n_patches_x = 20
n_patches_y = 20

n_tiles_x3 = 2
n_tiles_y3 = 2

datafolder3= '../data/Asbestos_Aug30/LM_TileSet_Subset_1K'
```

In [394]:
```
# reset
df = df.drop(columns=['kmeans'])
df2 = None
df3 = None
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-394-8b9f7fc70ca1> in <module>()
      1 # reset
----> 2 df = df.drop(columns=['kmeans'])
      3 df2 = None
      4 df3 = None

C:\ProgramData\Anaconda3\envs\JADS\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   3695                                           index=index, columns=columns,
   3696                                           level=level, inplace=inplace,
-> 3697                                           errors=errors)
   3698
   3699     @rewrite_axis_style_signature('mapper', [('copy', True),

C:\ProgramData\Anaconda3\envs\JADS\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   3109         for axis, labels in axes.items():
   3110             if labels is not None:
-> 3111                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3112
   3113         if inplace:

C:\ProgramData\Anaconda3\envs\JADS\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors)
   3141                 new_axis = axis.drop(labels, level=level, errors=errors)
   3142             else:
-> 3143                 new_axis = axis.drop(labels, errors=errors)
   3144             result = self.reindex(**{axis_name: new_axis})
   3145

C:\ProgramData\Anaconda3\envs\JADS\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   4402             if errors != 'ignore':
   4403                 raise KeyError(
-> 4404                     '{} not found in axis'.format(labels[mask]))
   4405             indexer = indexer[~mask]
   4406         return self.delete(indexer)

KeyError: "['kmeans'] not found in axis"
```

1. Filter black tiles

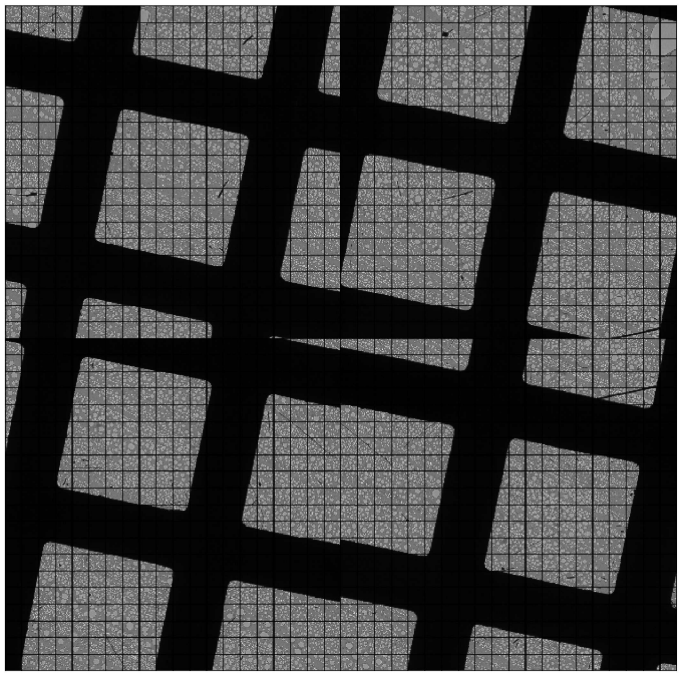
In [395]:
```
imgfiles3 = import_data(datafolder3)
```

In [396]:
```
df, feature_names = extract_features(imgfiles3, feature_funcs, n_patches_y, n_patches_x)
print(len(imgfiles))
```

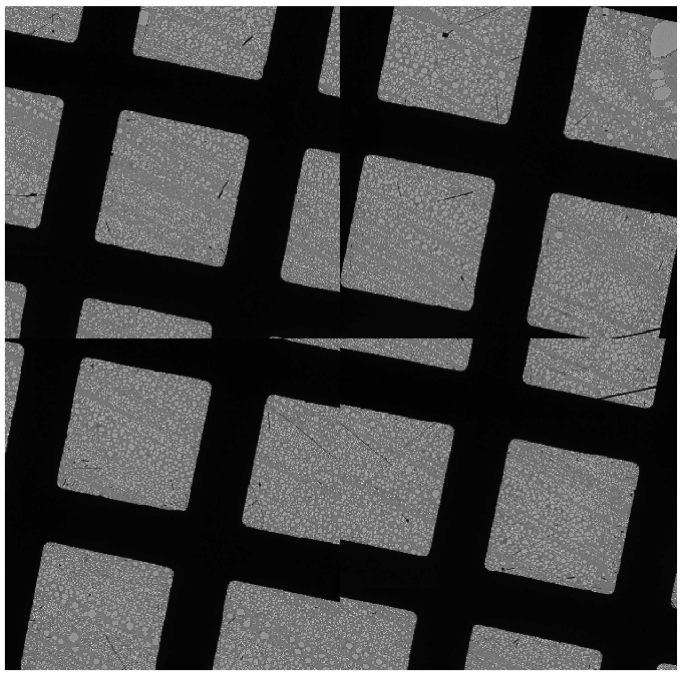
```
4
```

In [408]:
```
df['dummy']=0
imgutils.show_large_heatmap(df, 'dummy', imgfiles3, n_rows=n_tiles_y3, n_cols=n_tiles_x3, fig_size=(12,12))
imgutils.show_large_heatmap(df, 'dummy', imgfiles3, n_rows=n_tiles_y3, n_cols=n_tiles_x3, fig_size=(12,12), no_borders=True)
```

Heats from: dummy



Heats from: dummy

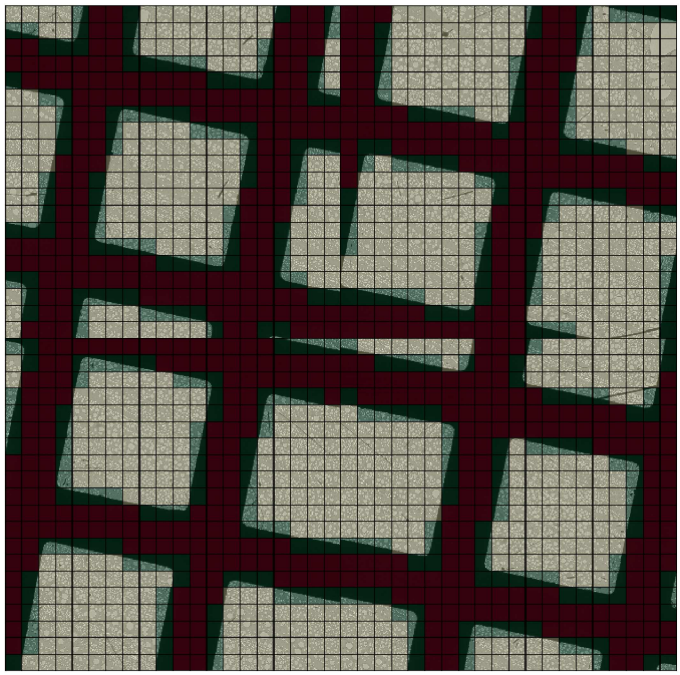


In [398]:
```
_ = run_kmeans_pipeline(df, feature_names, n_clusters_step1, standardize=True, use_pca=True )
```

In [399]:
```
#swap_clusters(df,'kmeans', 2, 1)
```

In [400]:
```
imgutils.show_large_heatmap(df, 'kmeans', imgfiles, n_rows=n_tiles_y3, n_cols=n_tiles_x3, fig_size=(12,12))
```

Heats from: kmeans



In [401]:
```
df['kmeans'].value_counts()
```

Out[401]:
```
1    673
0    570
2    357
Name: kmeans, dtype: int64
```

In [402]:
```
# cat_select = 1
# we know for this it's the biggest set
i_max_count = df['kmeans'].value_counts()
cat_select = i_max_count.index[0]
print(cat_select)
df2 = df[df['kmeans']==cat_select]
df2.head(3)
```

```
1
```

Out[402]:

	filename	s_y	s_x	n_y	n_x	alias	img_std	img_relstd	img_mean	img_skewness	img_kurtosis	img_mode	img_range	dummy	kmeans
0	..\data\Asbestos_Aug30\LM_TileSet_Subset_1K\Ti...	0	0	20	20	img0_0-0	1938.244745	0.205580	9428.1652	1.179754	0.444208	8452.095703	11244.0	0	1
1	..\data\Asbestos_Aug30\LM_TileSet_Subset_1K\Ti...	0	1	20	20	img0_0-1	2038.710986	0.219186	9301.2916	1.571238	1.730651	8555.300293	11957.0	0	1
2	..\data\Asbestos_Aug30\LM_TileSet_Subset_1K\Ti...	0	2	20	20	img0_0-2	2003.561952	0.219091	9144.8640	0.973276	2.168884	8503.500488	15523.0	0	1

In [403]:
```
score_kmeans = run_kmeans_pipeline(df2, feature_names, n_clusters_step2_kmeans, standardize=True, use_pca=True )
score_hierarch = run_hierarchical_pipeline(df2, feature_names, n_clusters_step2_hierarchical, standardize=False, use_pca=False )
```

In [404]:
```
df2=df2.rename(columns = {'kmeans':'kmeans2', 'hierarchical':'hierarchical2'})
df3 = df.merge(df2, 'left')
df3['kmeans2'].fillna(value=-1, inplace=True)
df3['hierarchical2'].fillna(value=-1, inplace=True)
df3['heats']=df3['kmeans2']+1
df3['heats2']=df3['hierarchical2']+1
```

In [405]:
```
df3['heats'].value_counts()
```

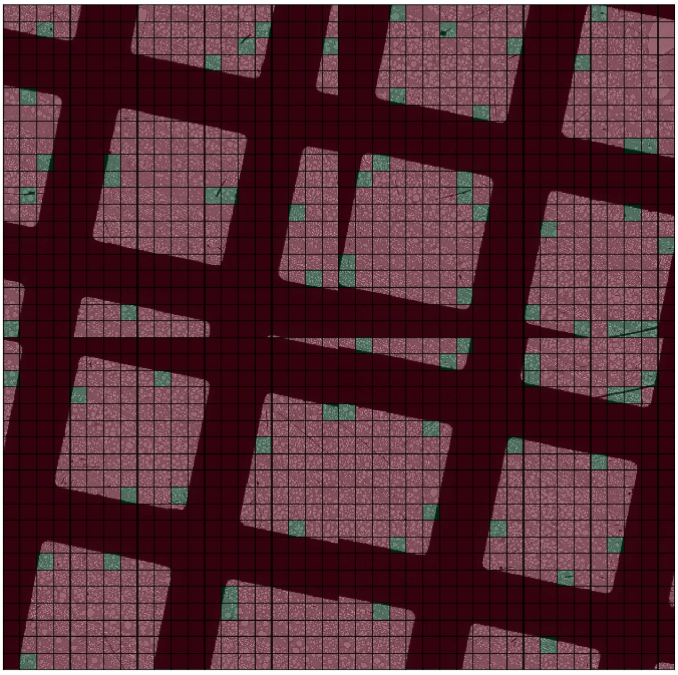
Out[405]:
```
0.0    927
1.0    600
2.0     73
Name: heats, dtype: int64
```

In [406]:
```
# make the whole 2 clusters only
df3['heats'].replace({1:0}, inplace=True)
df3['heats'].value_counts()

# make the whole 2 clusters only
df3['heats2'].replace({1:0}, inplace=True)
df3['heats2'].value_counts()
```

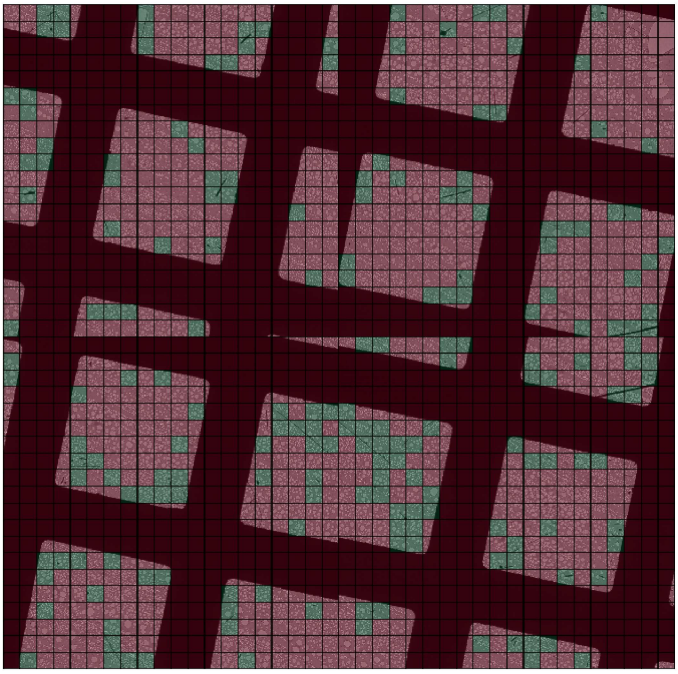
Out[406]:
```
0.0    1405
2.0     195
Name: heats2, dtype: int64
```

In [407]:
```
imgutils.show_large_heatmap(df3, 'heats', imgfiles3, n_rows=n_tiles_y3, n_cols=n_tiles_x3, fig_size=(16,12))
imgutils.show_large_heatmap(df3, 'heats2', imgfiles3, n_rows=n_tiles_y3, n_cols=n_tiles_x3, fig_size=(16,12))
```

Heats from: heats



Heats from: heats2



after filtering out the black ones, hierarchical clustering worked better than kmeans

This set needs smaller patches