Real Micro Crystals - Data Engineering & Exploration

Michael Janus, June 2018

Use the functions on a real (small) data set.

For explanation and how to usage functions, see the notebook **imgutils_test_and_explain.ipynb**

1. Import the used modules, including the one with test functions:

In [46]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import matplotlib.pyplot as plt

import imgutils
import imgutils_test as tst
```

In [47]:
```
# Re-run this cell if you altered imgutils or imgutils_test
import importlib
importlib.reload(imgutils)
importlib.reload(tst)
```

Out[47]:
```
<module 'imgutils_test' from 'C:\\JADS\\SW\\Grad Proj\\realxtals1\\sources\\imgutils_test.py'>
```

1. Get image files

In [48]:
```
df_imgfiles = imgutils.scanimgdir('../data/Crystals_Apr_12/Tileset7', '.tif')
print(df_imgfiles)
```

```
                                        filename
0  ..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...
1  ..\data\Crystals_Apr_12\Tileset7\Tile_001-002-...
2  ..\data\Crystals_Apr_12\Tileset7\Tile_001-003-...
3  ..\data\Crystals_Apr_12\Tileset7\Tile_002-001-...
4  ..\data\Crystals_Apr_12\Tileset7\Tile_002-002-...
5  ..\data\Crystals_Apr_12\Tileset7\Tile_002-003-...
```

2. Get Image Slice Statistics

This set contains 6 images. Let's slice those up in 4 by 4; this will give total of 6 x 4 x 4 = 96 slices. And also apply the statistics on each slice.

In [49]:
```
statfuncs = [imgutils.img_min, imgutils.img_max, imgutils.img_range, imgutils.img_mean, imgutils.img_std,
imgutils.img_median]
df = imgutils.slicestats(list(df_imgfiles['filename']), 4, 4, statfuncs)
print("records: ", df.shape[0])
df.head()
```

```
records:  96
```

Out[49]:

	filename	s_y	s_x	n_y	n_x	alias	img_min	img_max	img_range	img_mean	ir
0	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	0	4	4	img0_0-0	5419.0	12927.0	7508.0	8955.557637	489.
1	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	1	4	4	img0_0-1	5248.0	12854.0	7606.0	8883.137305	501.
2	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	2	4	4	img0_0-2	6084.0	10737.0	4653.0	8786.996070	327.
3	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	3	4	4	img0_0-3	7105.0	12208.0	5103.0	8679.430512	273.
4	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	1	0	4	4	img0_1-0	4534.0	10926.0	6392.0	8982.867158	380.

Normalize the statistics using 'standarization'

In [50]:
```
stat_names = imgutils.stat_names(statfuncs)
print(stat_names)
```

```
['img_min', 'img_max', 'img_range', 'img_mean', 'img_std', 'img_median']
```

In [51]:
```
df.isnull().values.any()
```

Out[51]: False

In [52]:
```
imgutils.normalize(df, stat_names)
df.head()
```

Out[52]:

	filename	s_y	s_x	n_y	n_x	alias	img_min	img_max	img_range	img_mean	ir
0	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	0	4	4	img0_0-0	5419.0	12927.0	7508.0	8955.557637	489.
1	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	1	4	4	img0_0-1	5248.0	12854.0	7606.0	8883.137305	501.
2	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	2	4	4	img0_0-2	6084.0	10737.0	4653.0	8786.996070	327.
3	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	0	3	4	4	img0_0-3	7105.0	12208.0	5103.0	8679.430512	273.
4	..\data\Crystals_Apr_12\Tileset7\Tile_001-001-...	1	0	4	4	img0_1-0	4534.0	10926.0	6392.0	8982.867158	380.

In [53]:
```
stat_normnames = imgutils.normalized_names(stat_names)
print(stat_normnames)
```

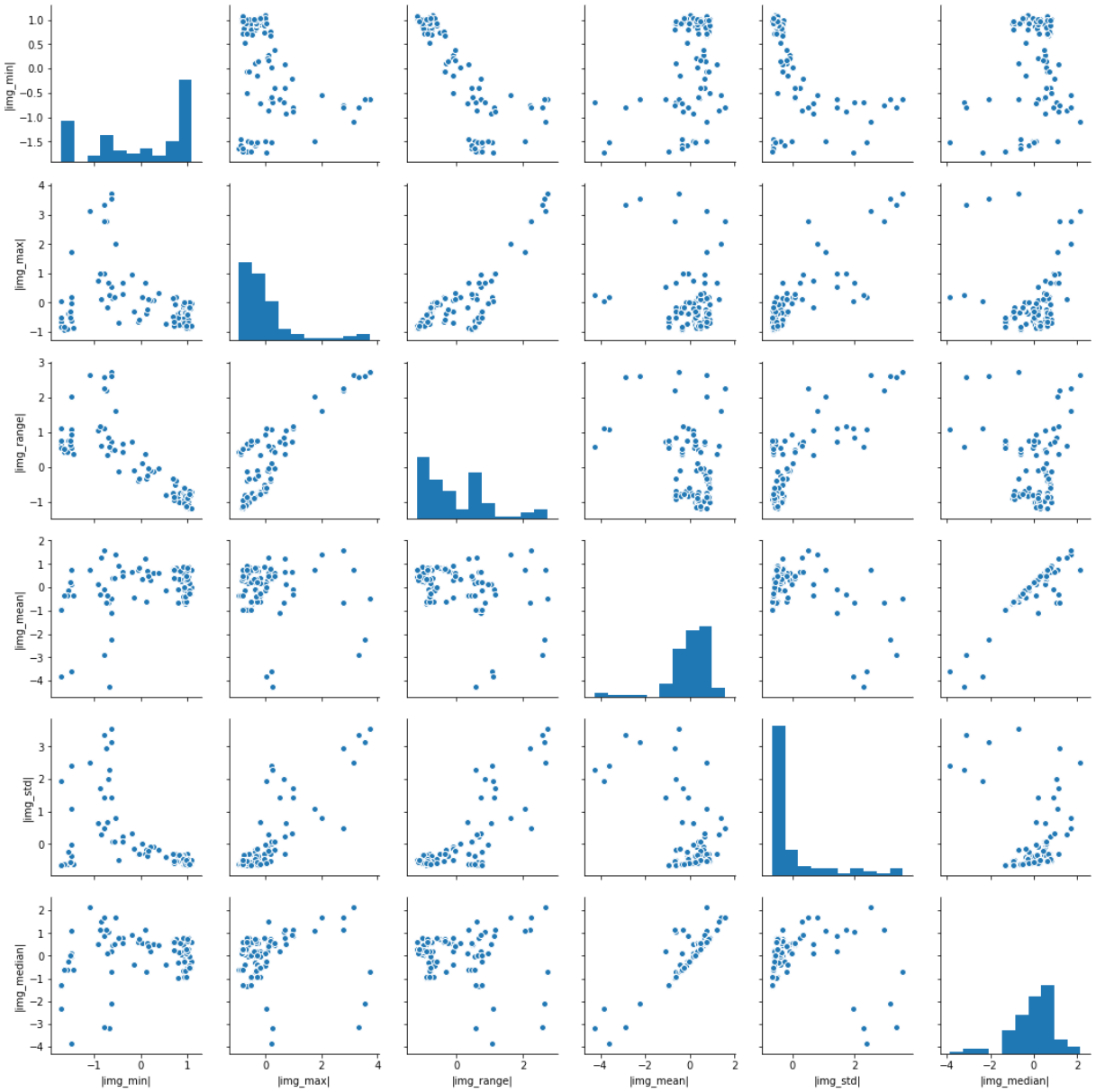
```
['|img_min|', '|img_max|', '|img_range|', '|img_mean|', '|img_std|', '|img_median|']
```

3. Check some combinations for patterns

(using the seaborn pairplot)

In [54]:
```
import seaborn as sb
```

In [56]:
```
%matplotlib inline
sb.pairplot(df, vars=stat_normnames)
plt.show()
```

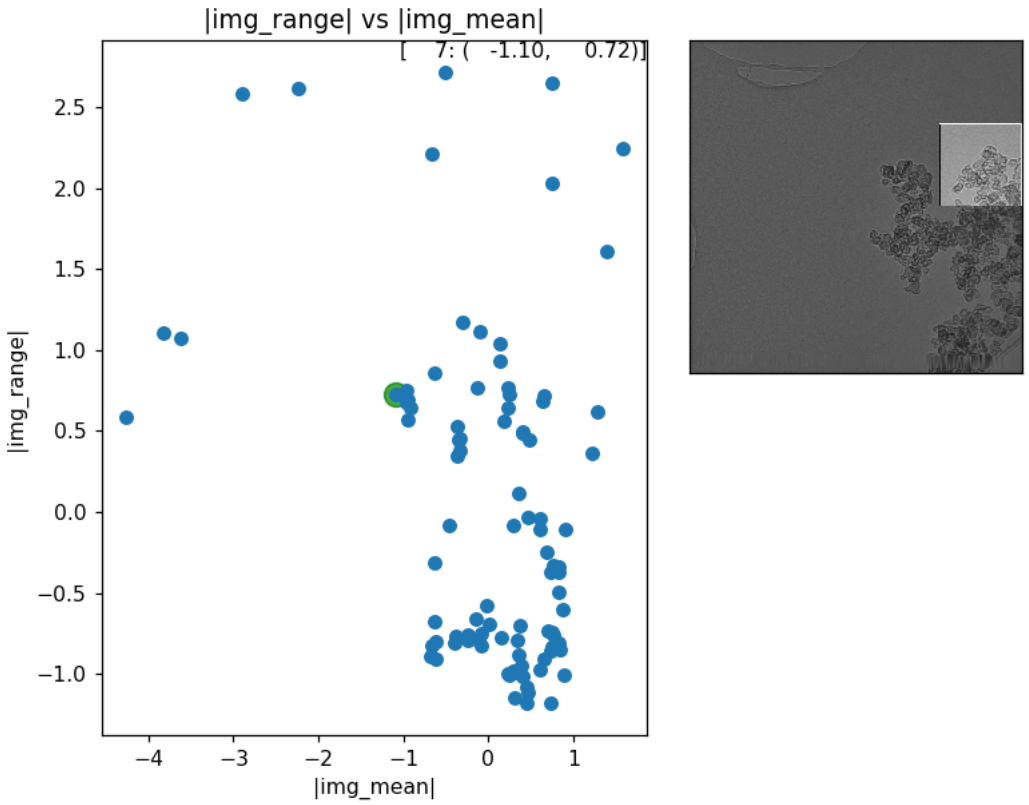


4. Inspect interactively

Let's inspect some combinations that have 'signs of clustering' in the interactive graph

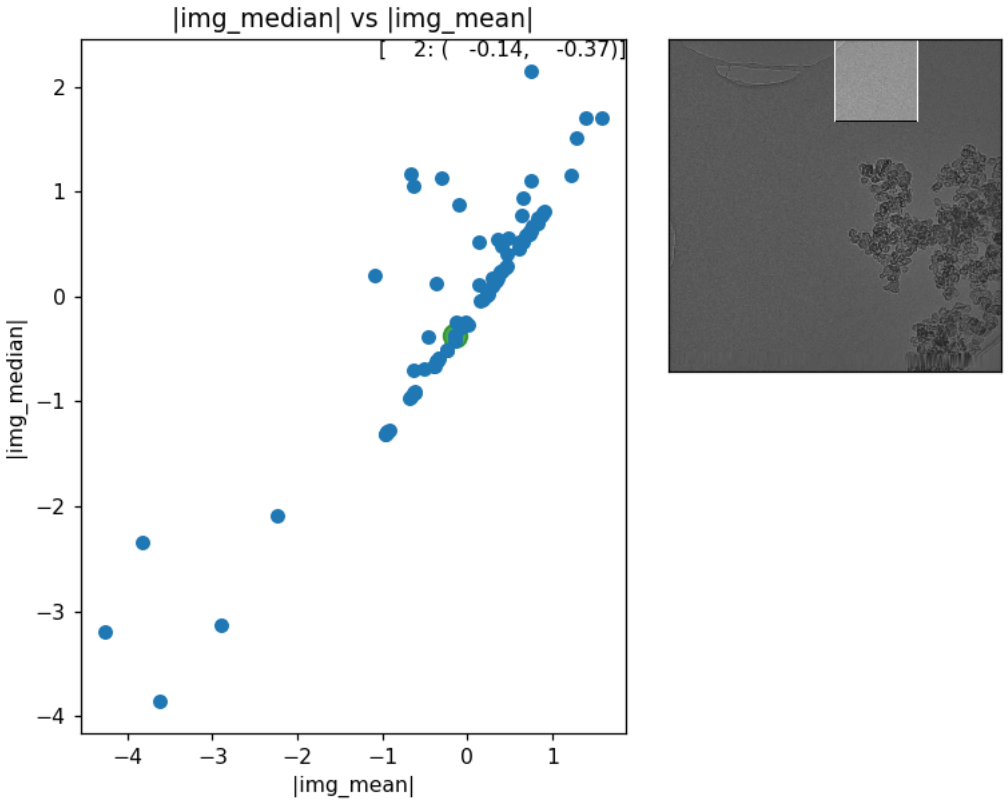
In [57]:
```
%matplotlib notebook
```

In [58]:
```
imgutils.plotwithimg(df, '|img_mean|', '|img_range|', imgutils.highlightimgslice, True)
```



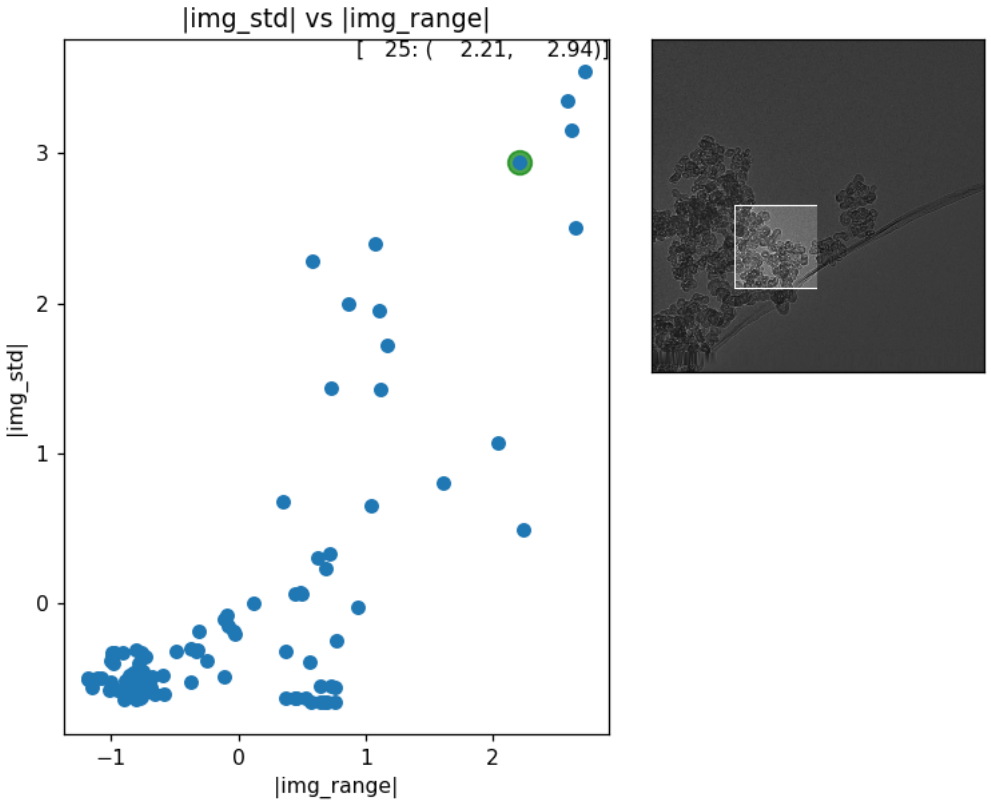
Looks likt the sort-of cluster in lower right are points without a crystal

In [59]:
```
imgutils.plotwithimg(df, '|img_mean|', '|img_median|', imgutils.highlightimgslice, True)
```



The separation is not representative, the group at top-left contains both with and without micro crystals

In [60]:
```
imgutils.plotwithimg(df, '|img_range|', '|img_std|', imgutils.highlightimgslice, True)
```



This looks better, bottom left are empty regions, top-left have crystals.

5. Heatmaps

Let's do an attempt to create a score for a heatmap. Looks like |img_std| is most infromative

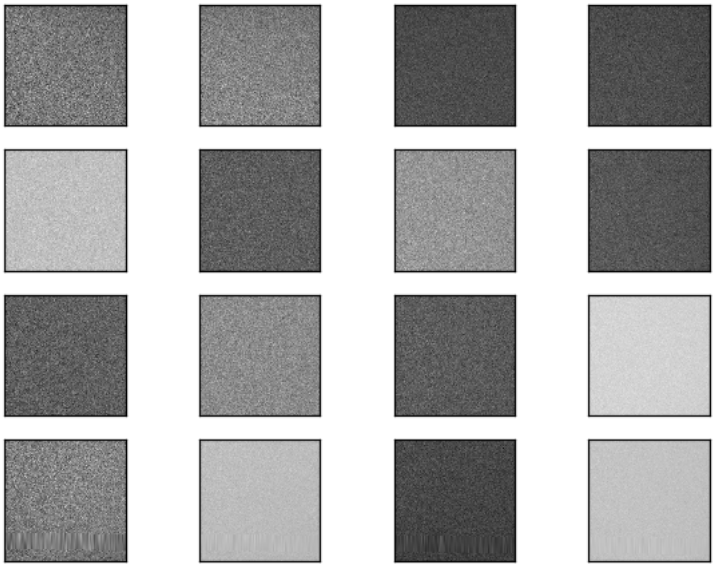
In [61]:
```
imgname = df_imgfiles.iloc[3]['filename']
print(imgname)
```

```
..\data\Crystals_Apr_12\Tileset7\Tile_002-001-000_0-000.tif
```

In [62]:
```
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
```

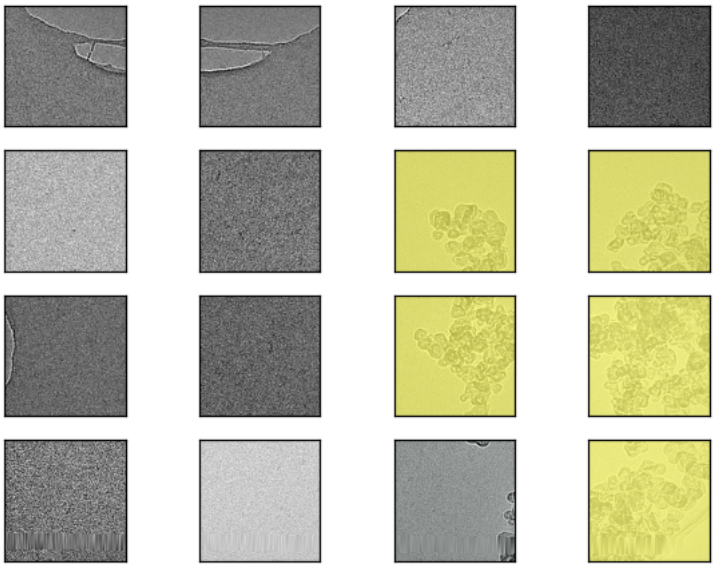
In [71]:
```
%matplotlib notebook
```

In [72]:
```
imgutils.showheatmap(imgs, heats)
```

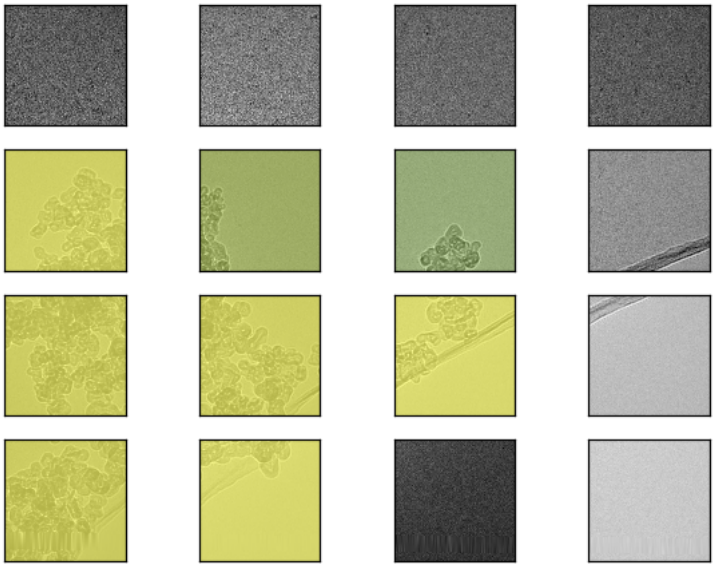


Yes, looks great!. Let's check for some other images as well

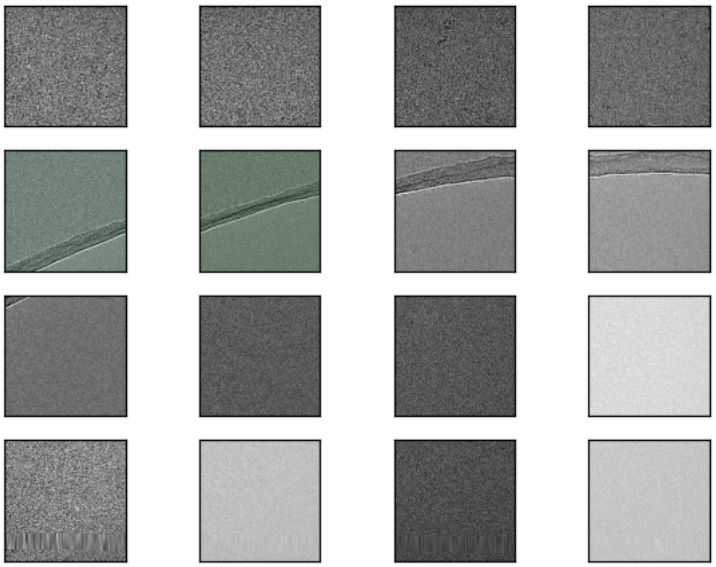
In [73]:
```
imgname = df_imgfiles.iloc[0]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



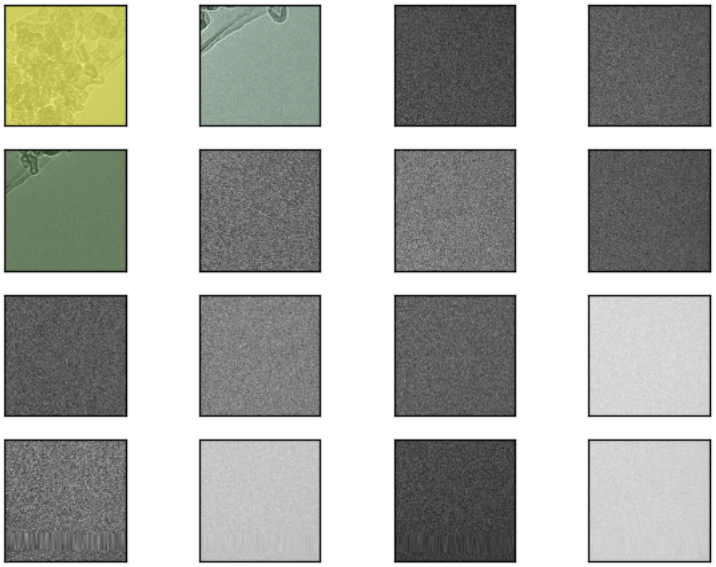
In [74]:
```
imgname = df_imgfiles.iloc[1]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



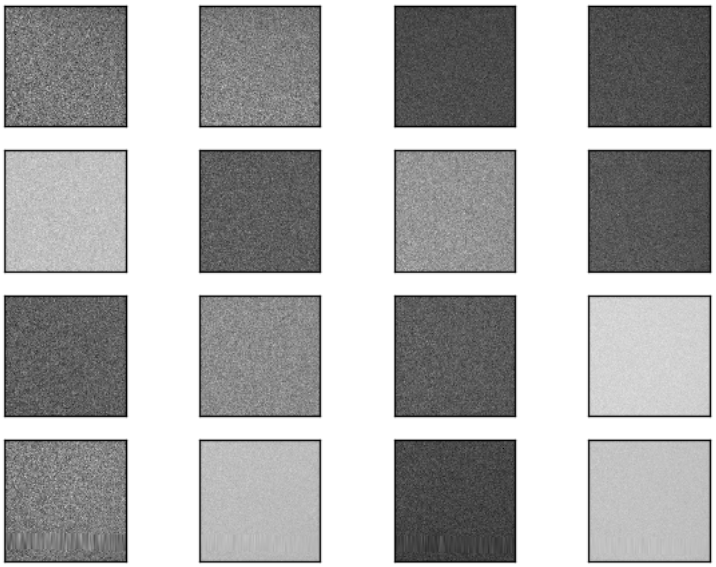
In [75]:
```
imgname = df_imgfiles.iloc[2]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



In [76]:
```
imgname = df_imgfiles.iloc[4]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



In [77]:
```
imgname = df_imgfiles.iloc[5]['filename']
imgs, heats = imgutils.getimgslices_fromdf(df, imgname, '|img_std|')
imgutils.showheatmap(imgs, heats, opacity=0.7)
```



6. Conclusions & Remarks

- The visualization and heatmap concept looks nice.
- Did not use real clustering, but from data exploration just used normalized standard deviation as indicator
- For larger or different sets (with outliers), I guess a combination of statistics is needed (which was the idea in the first place and let ML figure out what)

7. Next steps

- Export this data set and label it based on std-dev (e.g. 3 cats: none, some, full)
- Export this data set for unsupervised learning
- Repeat on bigger and more versatile set

Michael Janus, 15 June 2018