will it blend? - paint and snake

November 19, 2024

[173]:
```
import noise

from pdesign import canvas, shapes, lines
from pdesign import transforms as trans
from pdesign import smooth as smooth_lib


import numpy as np
from shapely.geometry import MultiLineString, LineString, Point, Polygon,␣
↪MultiPoint
from shapely.geometry import box as Box
from shapely.ops import unary_union
from shapely import affinity
import matplotlib.pyplot as plt

from skimage.draw import line, circle_perimeter
from skimage import draw

from ipywidgets import widgets
from ipywidgets import interact, interact_manual, interactive
```

[174]:
```
from scipy.ndimage import filters
from scipy import signal
from scipy.stats import multivariate_normal

import skimage
from skimage.feature import shape_index

from skimage import draw
from sklearn import preprocessing
from scipy import interpolate

from plottermagic.io import io

from scipy.ndimage import filters
from scipy import ndimage, misc
```

```
from skimage import exposure

from skimage.transform import rescale
```

[175]:
```
from skimage.morphology import disk
from skimage.filters import rank, unsharp_mask

from skimage.transform import pyramids
from skimage import transform
```

[176]:
```
img_a = io.load_image("/Users/gnb/dtd/images/smeared/smeared_0027.jpg",␣
 ↪as_type='grey').astype(float)
img_b = io.load_image("/Users/gnb/dtd/images/scaly/scaly_0156.jpg",␣
 ↪as_type='grey').astype(float)

img_a = pyramids.img_as_float(img_a)
img_b = pyramids.img_as_float(img_b)

print(np.percentile(img_a, [0, 50, 100]), np.percentile(img_b, [0, 50, 100]))
```

```
[ 21.  91. 245.] [  0. 107. 255.]
```

[177]:
```
np.percentile(exposure.adjust_gamma(img_b/255, 1.5), [0, 50, 100])
```

[177]:
```
array([0.        , 0.27180998, 1.        ])
```

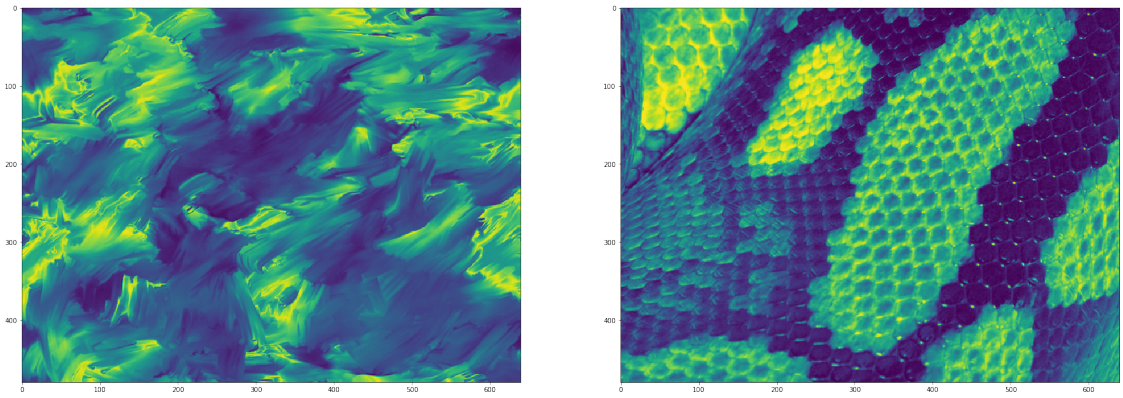
[178]:
```
a_asp = img_a.shape[0]/img_a.shape[1]
b_asp = img_b.shape[0]/img_b.shape[1]

if (a_asp<1 and b_asp>1) or (a_asp>1 and b_asp<1):

    img_b = np.transpose(img_b)

print(img_a.shape, img_b.shape)
```

```
(480, 640) (480, 640)
```

[ ]:
```

```

[191]:
```
#img_a = io.load_image("/Users/gnb/dtd/images/smeared/smeared_0027.jpg",␣
 ↪as_type='grey').astype(float)
img_a = 255*exposure.adjust_gamma(img_a/255, 0.9)


fig, axes = plt.subplots(ncols=2, figsize=(30,20))

axes[0].imshow(img_a, )
axes[1].imshow(img_b,)
```

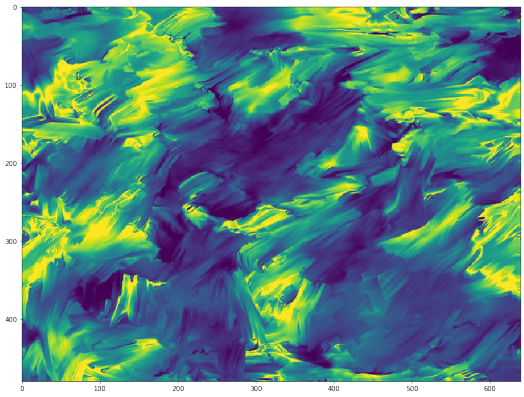
[191]: <matplotlib.image.AxesImage at 0x134b9e110>

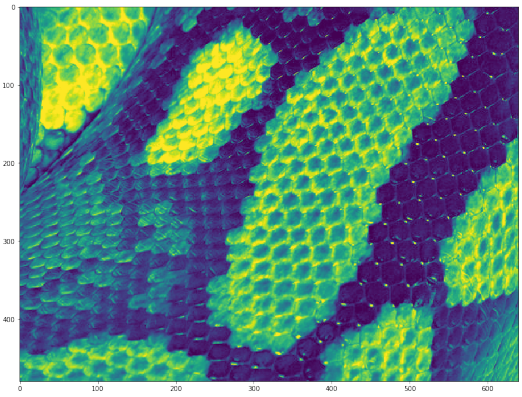


[ ]:

[192]:
```
n_layers = 4

sharpen_radius = 10
sharpen_mag = 1.5
```

[193]:
```
fig, axes = plt.subplots(ncols=2, figsize=(30,20))

for ii, img in enumerate([img_a, img_b]):
    g = [e for e in pyramids.pyramid_gaussian(img, max_layer=n_layers)]
    l = [e for e in pyramids.pyramid_laplacian(img, max_layer=n_layers)]

    sum_total = g[-1].copy()
    for i, ld in enumerate(l[::-1][1:]):
        sum_total = transform.rescale(sum_total, 2, anti_aliasing=False) + ld

    normalized = (sum_total-np.min(sum_total))/(np.max(sum_total) - np.
 ↪min(sum_total))

    sharpened = 255*unsharp_mask(normalized, radius=sharpen_radius,␣
 ↪amount=sharpen_mag)

    axes[ii].imshow(sharpened)
```

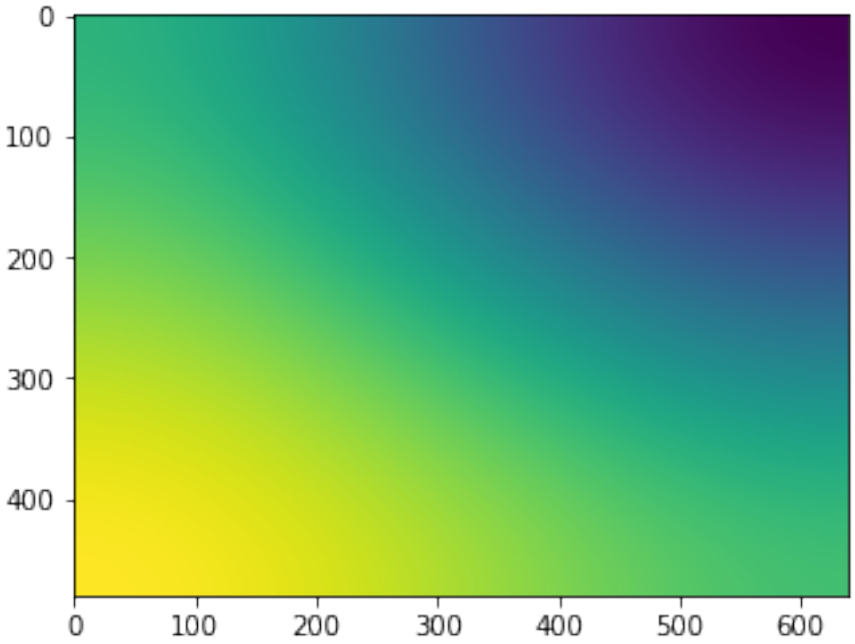




[255]:
```
"""
blend_radius = 50
mask = np.zeros_like(img_a).astype(float)
tri_x, tri_y = draw.polygon([0, 0, img_a.shape[0]], [0, img_a.shape[1], img_a.
 ↪shape[1]], shape=img_a.shape)
mask[tri_x, tri_y] = 1.0
mask = mask**10


blend_radius = 20
mask = np.zeros_like(img_a).astype(float)
chunk_size = 80
for start in np.arange(0, mask.shape[0], chunk_size)[::2]:
    mask[start:start+chunk_size] = 1


"""



blend_radius = 200
mask = np.zeros_like(img_a).astype(float)
tri_x, tri_y = draw.polygon([0, 0, img_a.shape[0]-150], [0, img_a.shape[1],␣
 ↪img_a.shape[1]], shape=img_a.shape)
mask[tri_x, tri_y] = 1.0


mask = 1 - mask


mask = mask**2


mask = filters.gaussian_filter(mask, sigma=blend_radius)
```

```
mask_layers = [g for g in pyramids.pyramid_gaussian(mask, max_layer=n_layers)]

plt.imshow(mask)
```

[255]: <matplotlib.image.AxesImage at 0x140a04350>



[ ]:

[ ]:

[256]:
```
mask.shape
```

[256]: (480, 640)

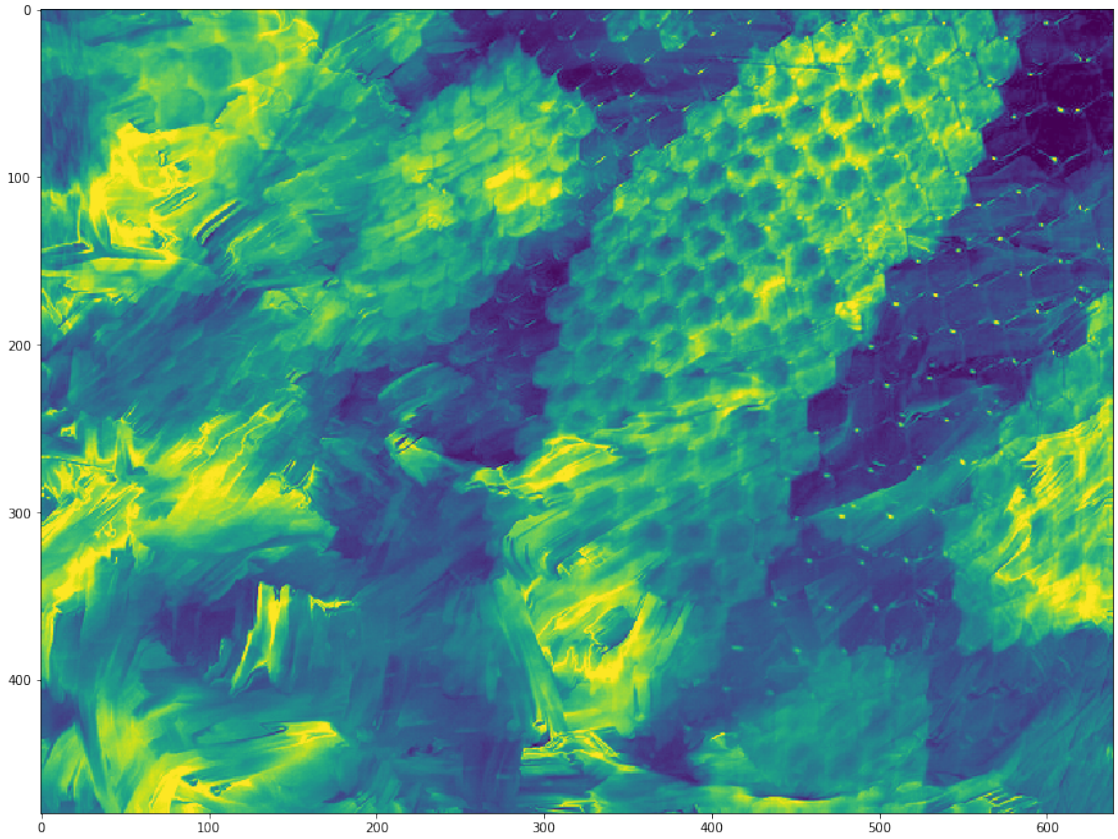
[257]:
```
np.arange(0, mask.shape[0], 80), np.arange(0, mask.shape[0], 80)[::2]
```

[257]: (array([0, 80, 160, 240, 320, 400]), array([0, 160, 320]))

[258]:
```
g1, g2 = [e for e in pyramids.pyramid_gaussian(img_a, max_layer=n_layers)], [e␣
↪for e in pyramids.pyramid_gaussian(img_b, max_layer=n_layers)]
l1, l2 = [e for e in pyramids.pyramid_laplacian(img_a, max_layer=n_layers)], [e␣
↪for e in pyramids.pyramid_laplacian(img_b, max_layer=n_layers)]
```

```
blended_img = g1[-1]*mask_layers[-1]+g2[-1]*(1-mask_layers[-1])


l1, l2, masks = l1[::-1][1:], l2[::-1][1:], mask_layers[::-1][1:]

for i in range(len(masks)):
    m = masks[i]
    lp1, lp2 = l1[i], l2[i]
    blended_img = transform.rescale(blended_img, 2, anti_aliasing=False) +␣
 ↪lp1*m+lp2*(1-m)



normalized = (blended_img-np.min(blended_img))/(np.max(blended_img) - np.
 ↪min(blended_img))
sharpened = 255*unsharp_mask(normalized, radius=sharpen_radius,␣
 ↪amount=sharpen_mag)

blended_img = sharpened

plt.figure(figsize=(15,15))
plt.imshow(blended_img)
```

[258]: <matplotlib.image.AxesImage at 0x1405e12d0>



[259]:
```
def get_fft_smoothed(img, sigma):
    fft = np.fft.fftn(img)
    smoothed_fft = ndimage.fourier_gaussian(fft, sigma=sigma)
    smoothed = np.abs(np.fft.ifftn(smoothed_fft))

    return smoothed, smoothed_fft


def get_derivs(img):
    d1 = ndimage.sobel(img, axis=0)
    d2 = ndimage.sobel(img, axis=1)
    d_len = np.hypot(d1,d2)
    return d1, d2, d_len
```

[260]:
```
im_smooth_factors = [1, 2, 5, 10,]
smoothed_weights = [0.12, 0.08, 0.4, 1.5]


gamma = 1
grad_speed = 12
```

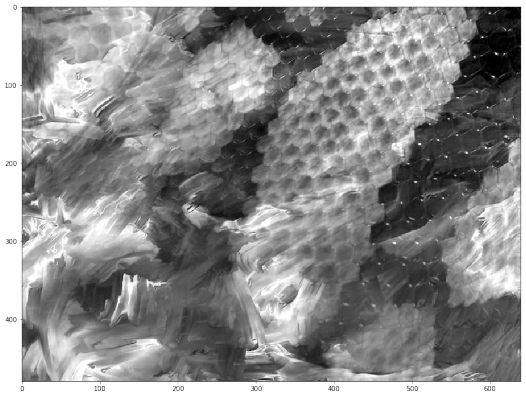
```
grad_smooth = 0.25
alpha = -1

image_bounds = (0, 255)

"""
params = [
    (1, 1, 10.),
]
im_smooth_factors = [1,2, 5, 10,]
smoothed_weights = [0.12, 0.08, 0.4, 1.5]
np.clip(original_image, 3, 200)



params = [
    (0.08, 0, 25),
]
im_smooth_factors = [1,2, 5, 10]
smoothed_weights = [0.12, 0.08, 0.4, 1.5]
gc = exposure.adjust_gamma(np.clip(original_image, 3, 255), im_gamma)
"""



gc = exposure.adjust_gamma(np.clip(blended_img, *image_bounds), gamma)

d1, d2 = np.zeros_like(gc).astype(float), np.zeros_like(gc).astype(float)

for fft_s, w in zip(im_smooth_factors, smoothed_weights):

    z, _ = get_fft_smoothed(gc, fft_s)

    dx = ndimage.sobel(z, axis=0)
    dy = ndimage.sobel(z, axis=1)

    d1 += grad_speed*w*(2/255)*dx
    d2 += grad_speed*w*(2/255)*dy

d1 = ndimage.gaussian_filter(d1, sigma=grad_smooth)
d2 = ndimage.gaussian_filter(d2, sigma=grad_smooth)
```

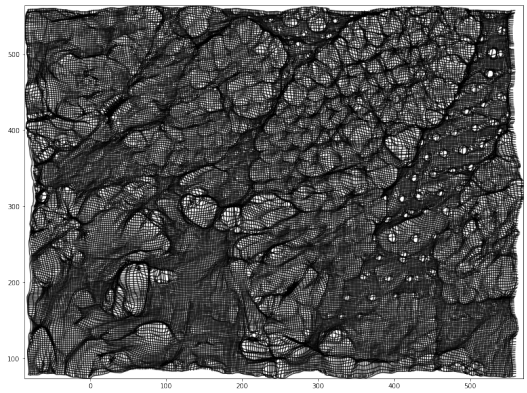
[261]:
```
v_lines = []
h_lines = []
```

```
for c1 in np.arange(0, blended_img.shape[0], 1):

    v2 = np.arange(0, blended_img.shape[1], 1)
    v1 = c1*np.ones_like(v2)

    s1 = v1+alpha*d1[c1, v2]
    s2 = v2+alpha*d2[c1, v2]

    h_lines.append(np.stack([s1, s2]).transpose())

for c2 in np.arange(0, blended_img.shape[1], 1):

    v1 = np.arange(0, blended_img.shape[0], 1)
    v2 = c2*np.ones_like(v1)


    s1 = v1+alpha*d1[v1, c2]
    s2 = v2+alpha*d2[v1, c2]
    v_lines.append(np.stack([s1, s2]).transpose())
```

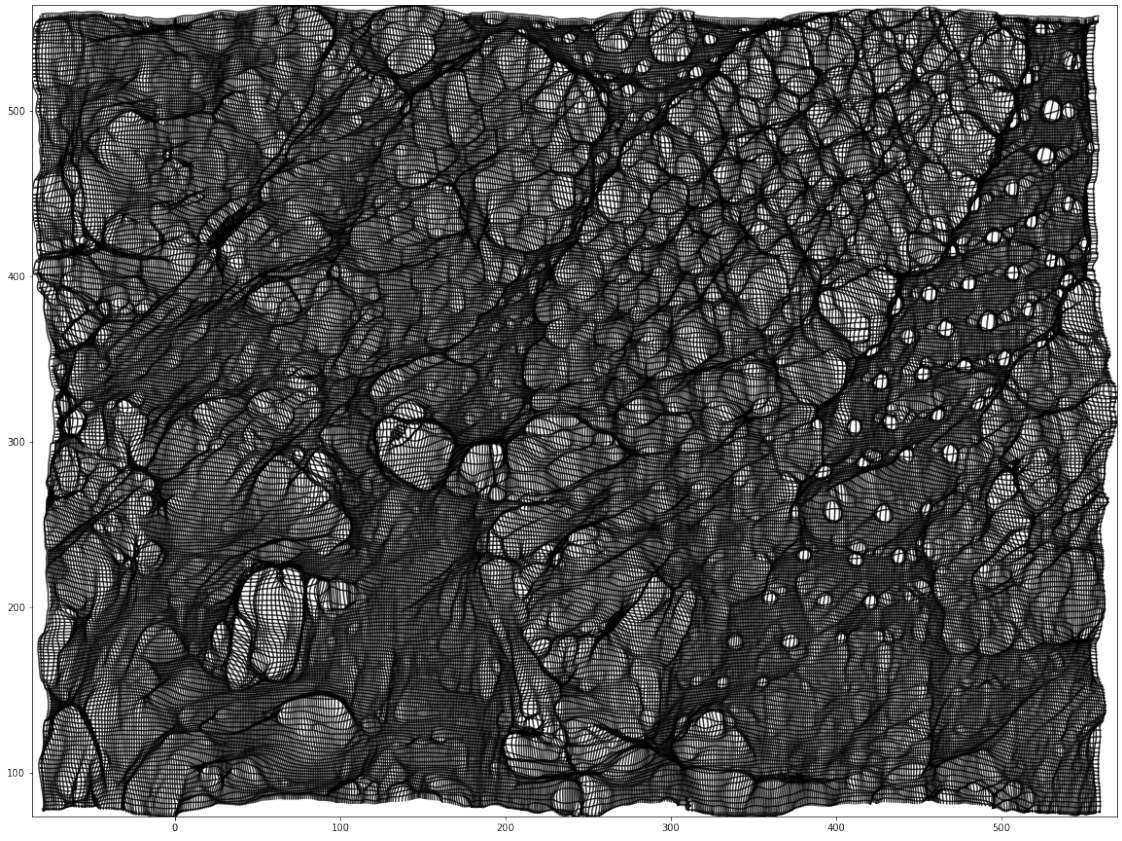
[262]:
```
mls_sets = [affinity.rotate(MultiLineString(c[::2]), -90) for c in [h_lines,␣
 ↪v_lines]]

fig, axes = plt.subplots(ncols=2, figsize=(30,20))
axes[0].imshow(blended_img, cmap='gray')

axes[1].add_collection(canvas.shapely_objs_to_lines(mls_sets, color='black'))
axes[1].set_xlim(mls_sets[0].bounds[0], mls_sets[0].bounds[2])
axes[1].set_ylim(mls_sets[0].bounds[1], mls_sets[0].bounds[3])
axes[1].set_aspect('equal')

print('xbounds', mls_sets[0].bounds[0], mls_sets[0].bounds[2])
print('ybounds', mls_sets[0].bounds[1], mls_sets[0].bounds[3])
```

```
xbounds -86.36557751877956 570.2341959940795
ybounds 73.59500768559997 564.0868180087962
```





[263]:
```
mls_sets = [affinity.rotate(MultiLineString(c), -90) for c in [h_lines[::2],␣
 ↪v_lines[::1]]]

fig, ax = plt.subplots(figsize=(20,20))

ax.add_collection(canvas.shapely_objs_to_lines(mls_sets, color='black'))
ax.set_xlim(mls_sets[0].bounds[0], mls_sets[0].bounds[2])
ax.set_ylim(mls_sets[0].bounds[1], mls_sets[0].bounds[3])
ax.set_aspect('equal')
```



[ ]:

[ ]:

[ ]:

[ ]: