data_augmentation

June 16, 2020

1 Exemplo de pré-processamento e aumento de dados

[1]:
```
from create_dataset import *

from skimage import img_as_ubyte
import scipy
import rasterio.plot
```

[2]:
```
%matplotlib inline
```

[3]:
```
SAMPLE_INSTANCE = "c160418tscrfl_nfung_64"
FOLDER_SAMPLE =  "data/RAW/crop_tsc_2016_b2s"
```

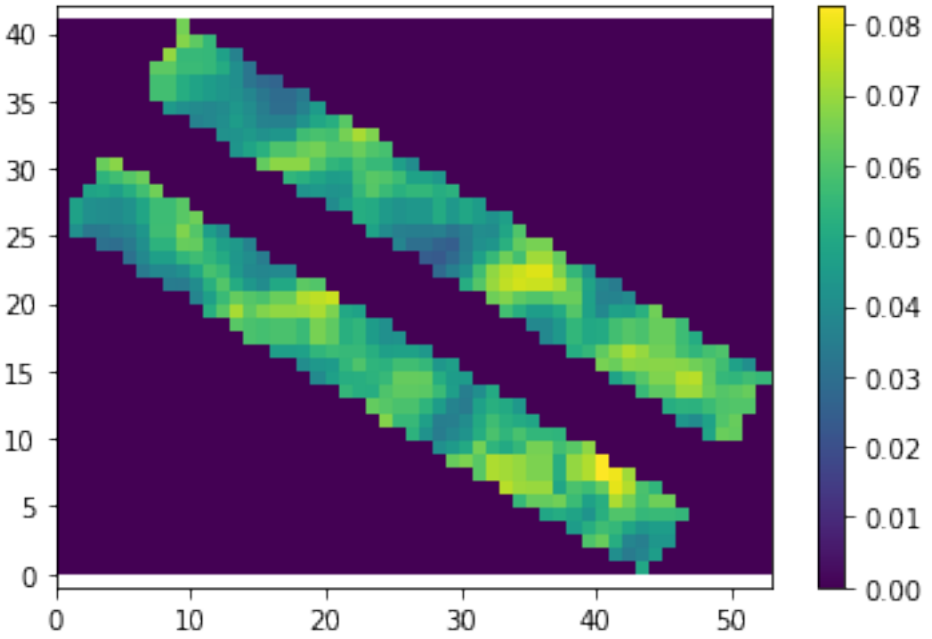
[4]:
```
def show(im):
    plt.pcolormesh(im)
    plt.axis('equal')
    plt.colorbar()
    plt.show()
    plt.close()

def print_histogram(im):
    plt.hist(im, bins='auto')
    plt.title("Histogram")
    plt.show()
    plt.close()
```

1.1 Pré processamento

[5]:
```
path = os.path.join(FOLDER_SAMPLE, SAMPLE_INSTANCE + '_B1.tif')
im_array = read_tif(path)
print(im_array.shape)
show(im_array)
```

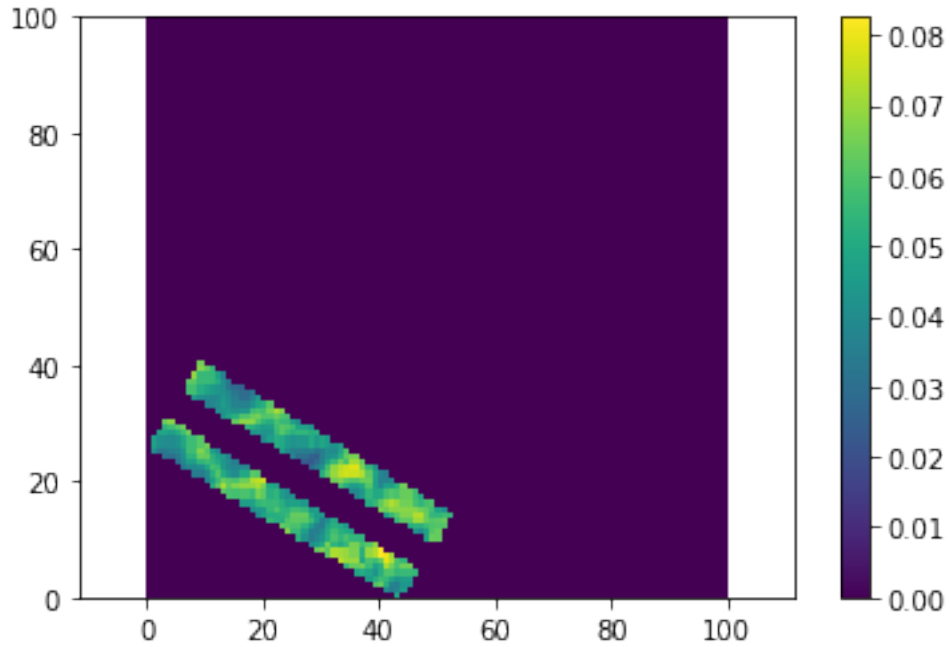
```
(41, 53)
```



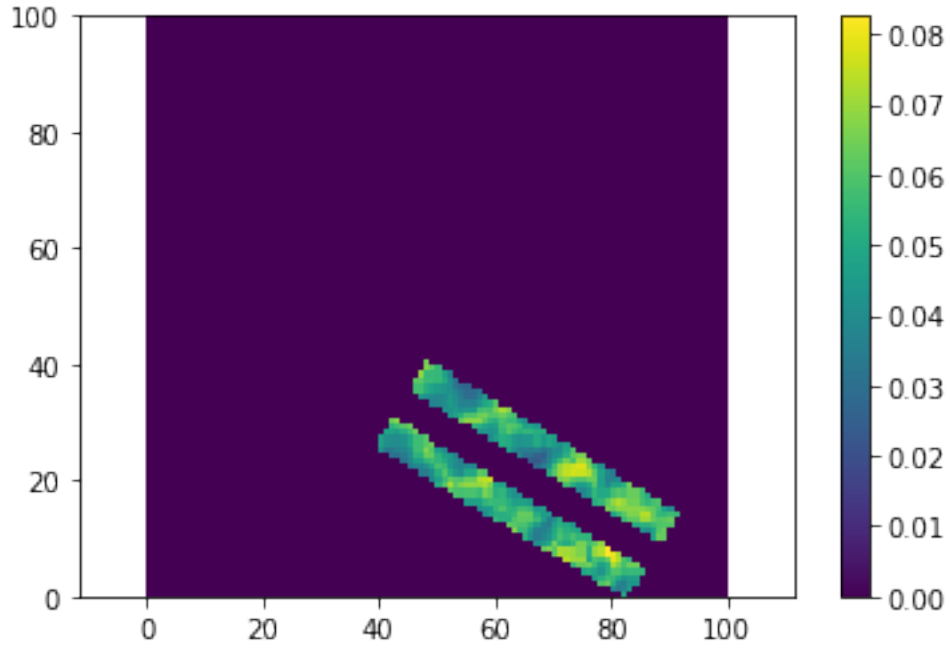
1.1.1 Padding

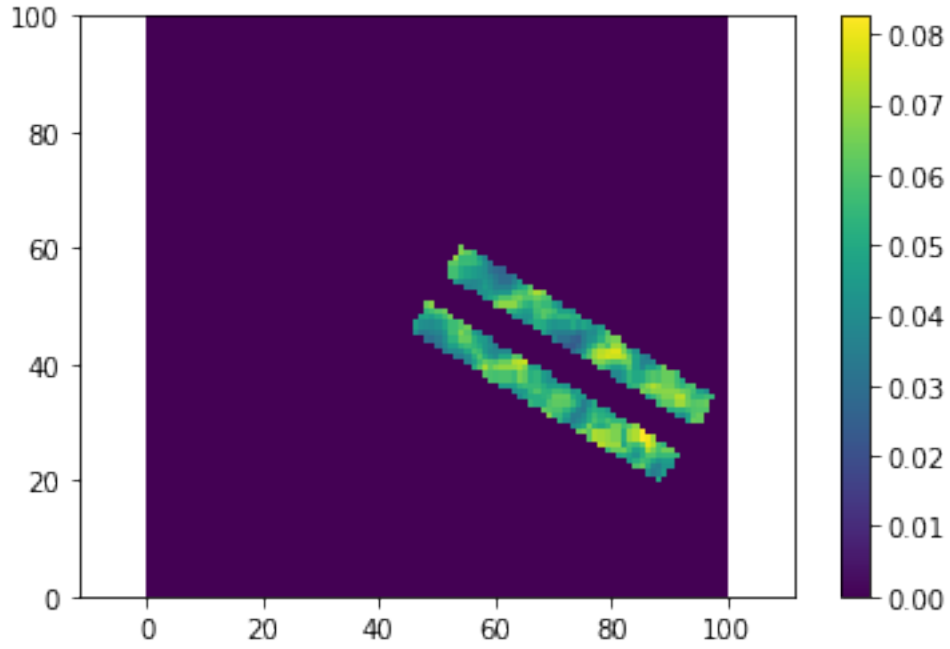
Algumas imagens apresentam shapes ligeiramente diferentes. Para garantir uma shape igual à todas as imagens, é recomendado realizar um padding.

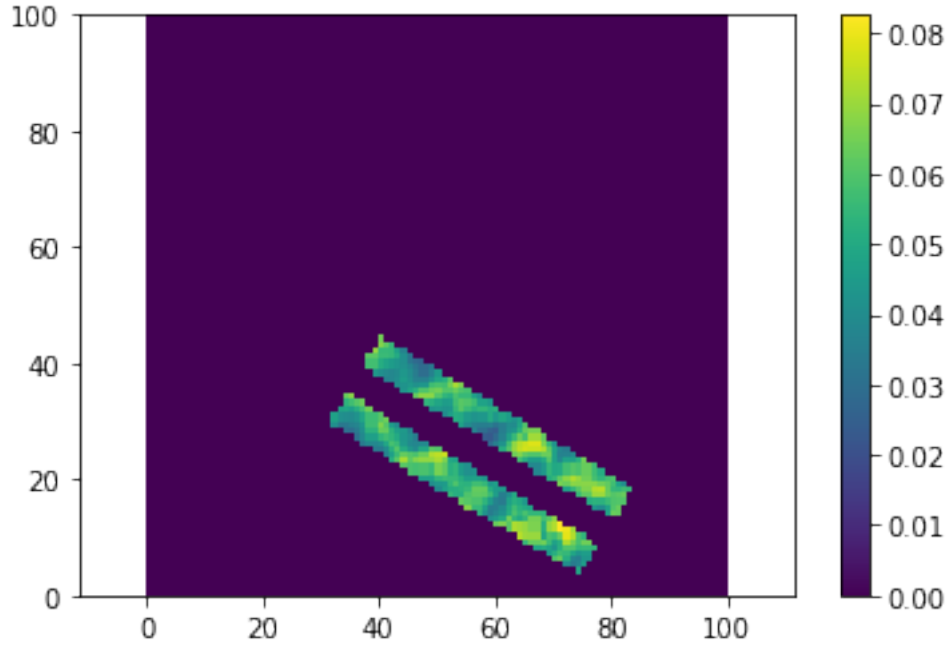
[6]:
```
show(pad_mat(im_array, shape=DEFAULT_SHAPE))
```



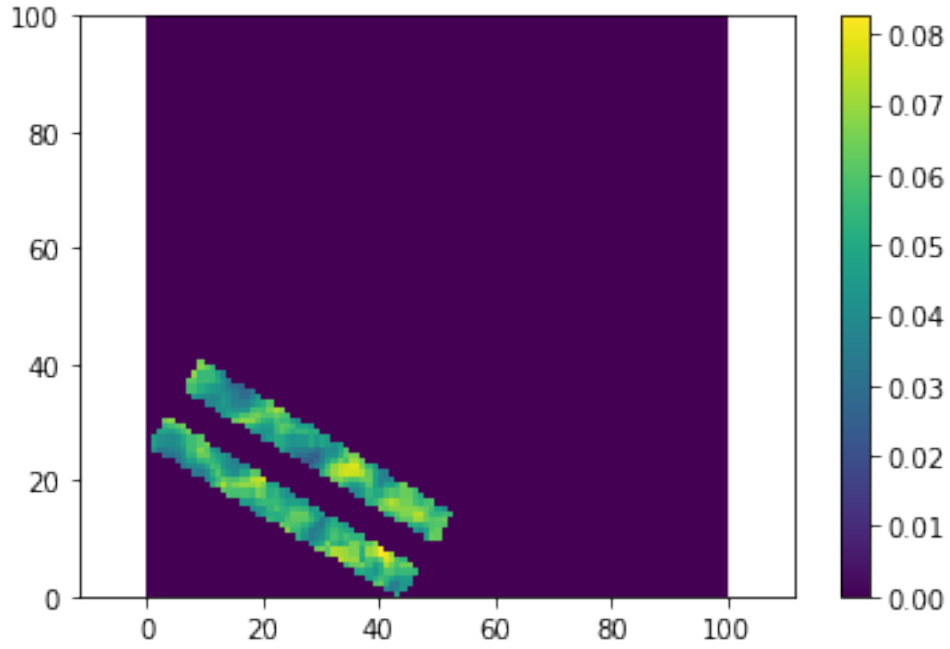
[7]:
```
show(pad_mat(im_array, start='random', shape=DEFAULT_SHAPE))
show(pad_mat(im_array, start='random', shape=DEFAULT_SHAPE))
show(pad_mat(im_array, start='random', shape=DEFAULT_SHAPE))
```







[8]:
```
im_array = pad_mat(im_array, DEFAULT_SHAPE)
show(im_array)
print(im_array.shape)
```



```
(100, 100)
```

1.1.2 Merge dos canais

[9]:
```
bands = []
for band in ['B4', 'B1', 'B2', 'B3']:
    print('Opening band', band)
    path = os.path.join(FOLDER_SAMPLE, SAMPLE_INSTANCE + '_' + band + '.tif')
    bands.append(pad_mat(read_tif(path), DEFAULT_SHAPE))
fullim = merge_channels(bands)
```

```
Opening band B4
Opening band B1
Opening band B2
Opening band B3
```

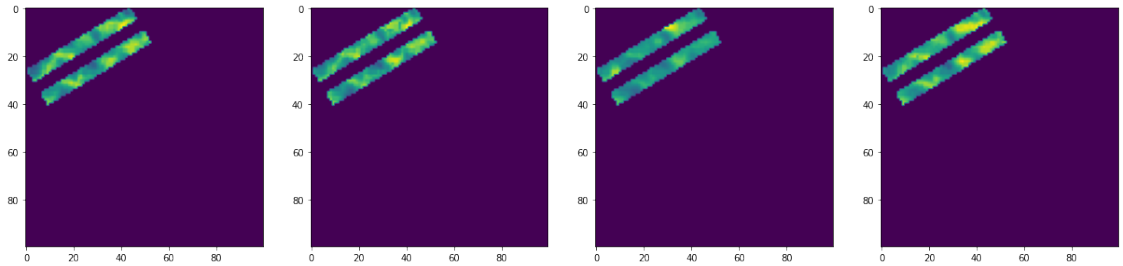
1.1.3 Visualização dos canais

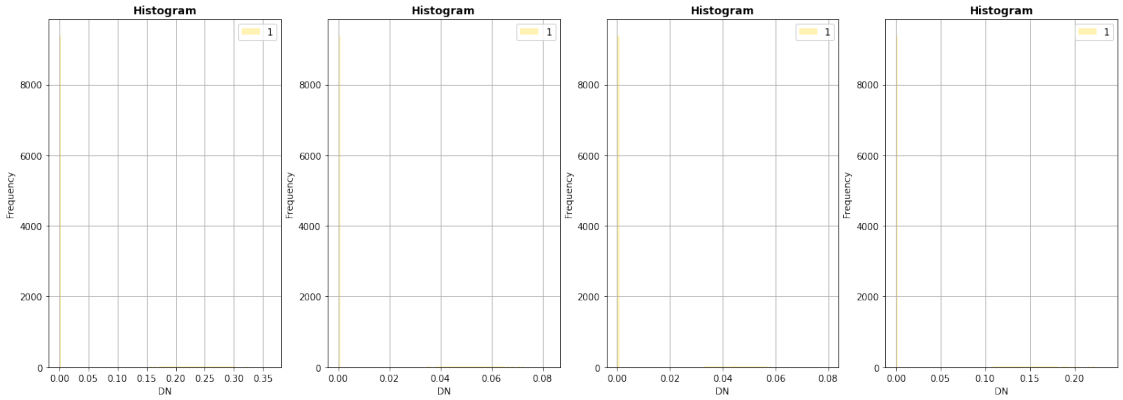
[10]:
```
def visualize(fullim, show_hist=False):
    print(fullim.shape)
    fig, subplots = plt.subplots(1, 4, figsize=(21, 7))
    for i in range(fullim.shape[-1]):
        rasterio.plot.show(fullim[:, :, i], ax=subplots[i])
    plt.show()
    plt.close()
    if show_hist:
        fig, subplots = plt.subplots(1, 4, figsize=(21, 7))
        for i in range(fullim.shape[-1]):
            rasterio.plot.show_hist(fullim[:, :, i], ax=subplots[i],
                                    bins=100, lw=0.0, stacked=False, alpha=0.3,␣
↪title="Histogram")
        plt.show()
        plt.close()
```

[11]:
```
# for i in range(fullim.shape[-1]):
#     show(fullim[:, :, i])
#     print_histogram(fullim[:, :, i])
```

[12]:
```
visualize(fullim, show_hist=True)
```

```
(100, 100, 4)
```





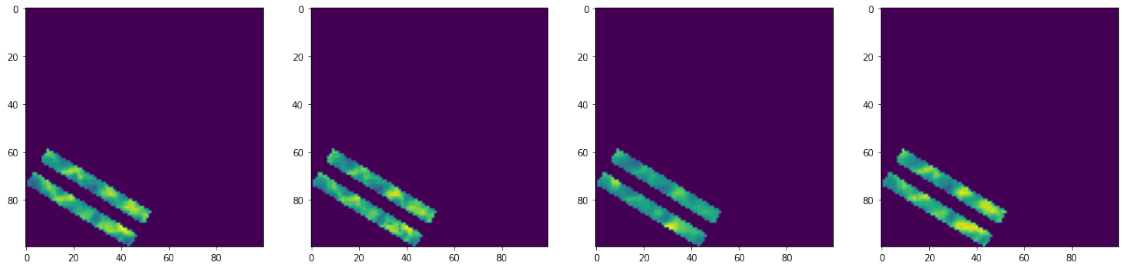
1.2 Técnicas de aumento de dados básicas

Técnicas que não alteram a qualidade da imagem.

1.2.1 Flip vertical

[13]:
```
im_flip_ver = vertical_flip(fullim, True)
visualize(im_flip_ver)
```

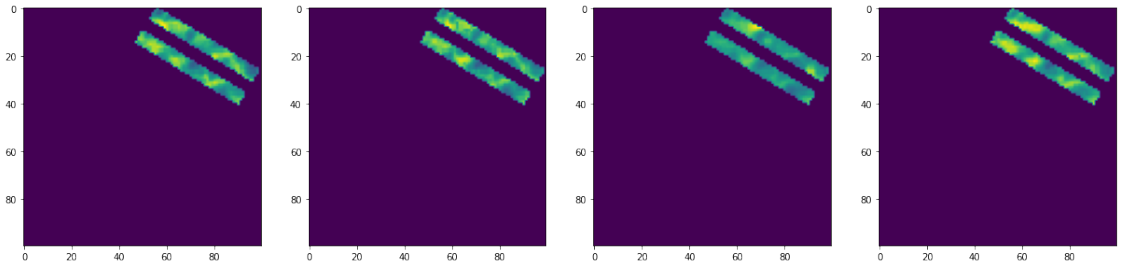
```
(100, 100, 4)
```



1.2.2 Flip horizontal

[14]:
```
im_flip_hor  = horizontal_flip(fullim, True)
visualize(im_flip_hor)
```

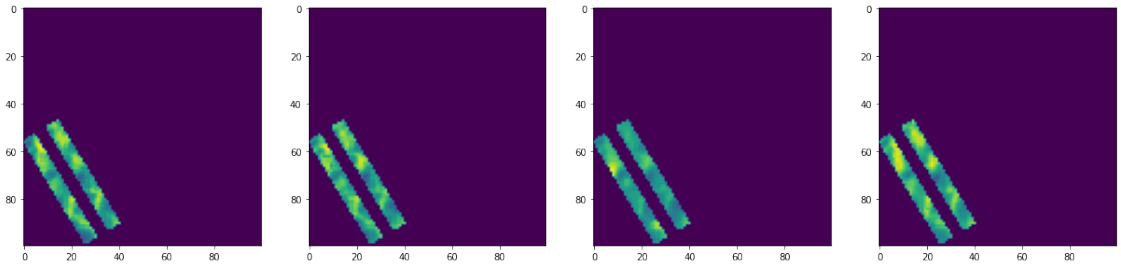
```
(100, 100, 4)
```



1.2.3 Rotação 90

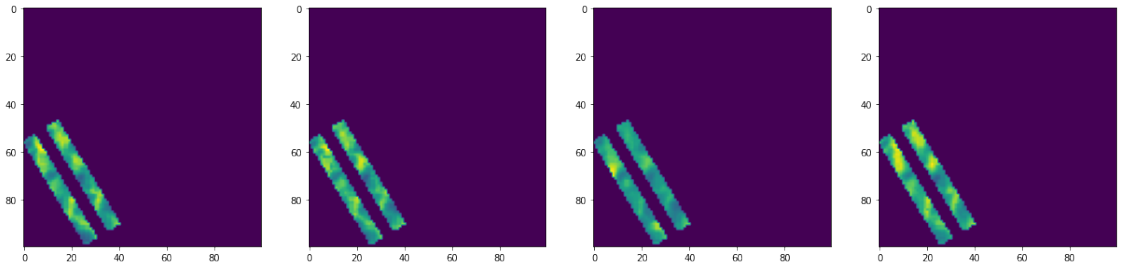
[15]:
```
print(fullim.shape)
im_rot = rot90(fullim)
visualize(im_rot)
```

```
(100, 100, 4)
(100, 100, 4)
```



[16]:
```
print(fullim.shape)
im_rot = rot90(fullim, k=1)
visualize(im_rot)
```

```
(100, 100, 4)
(100, 100, 4)
```



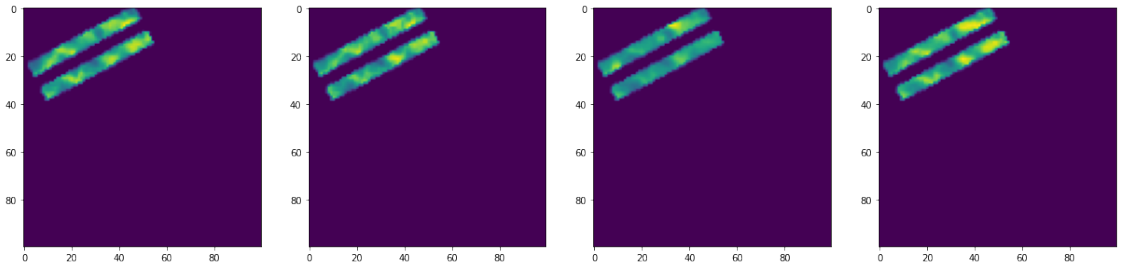
1.3 Técnicas de aumento de dados avaçadas

1.3.1 Rotação

A rotação pode cortar alguns pedaços da imagem (dependendo do ângulo).

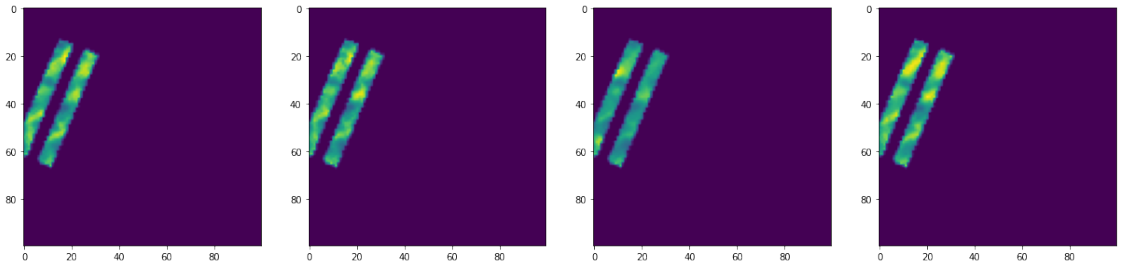
[17]:
```
im_rot30 = rotation(fullim, 30)
visualize(im_rot30)
```

```
(100, 100, 4)
```



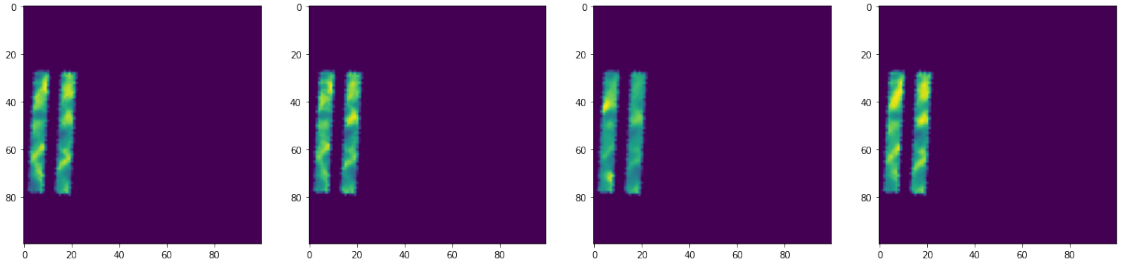
[18]:
```
im_rot90  = rotation(fullim, 90)
visualize(im_rot90)
```

```
(100, 100, 4)
```



[19]:
```
im_rot120  = rotation(fullim, 120)
visualize(im_rot120)
```

```
(100, 100, 4)
```

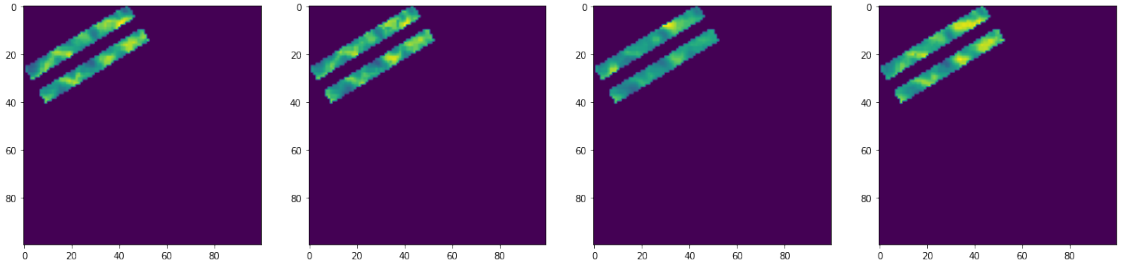


1.3.2 Zoom

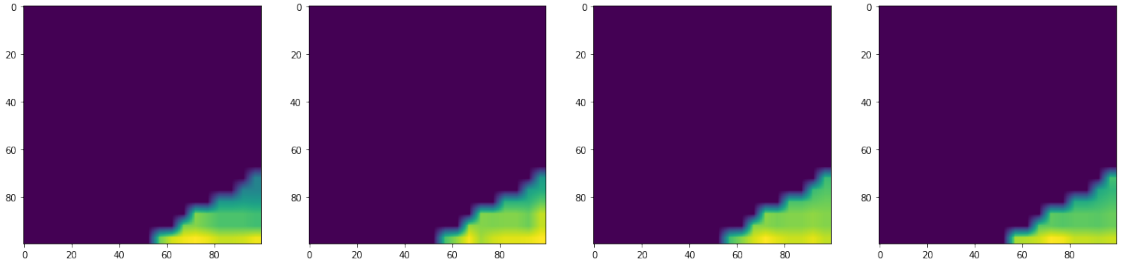
O zoom pode afetar a nitidez e cortar alguns pedaços da imagem.

[20]:
```
imzoom1 = zoom(fullim, .2)
visualize(fullim)
visualize(imzoom1)
print(fullim.shape, imzoom1.shape)
```

```
(100, 100, 4)
```



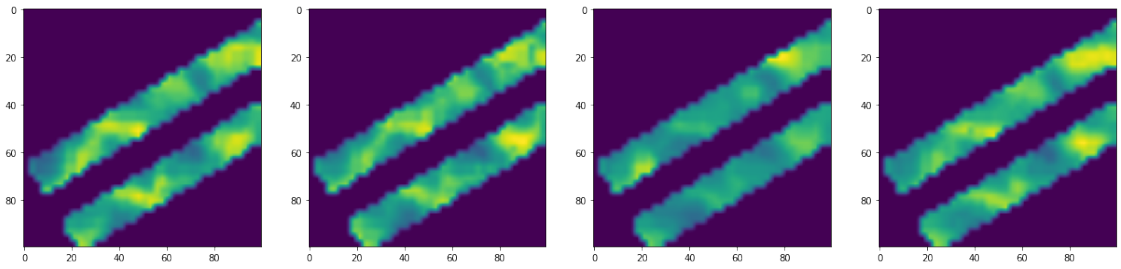
```
(100, 100, 4)
```



```
(100, 100, 4) (100, 100, 4)
```

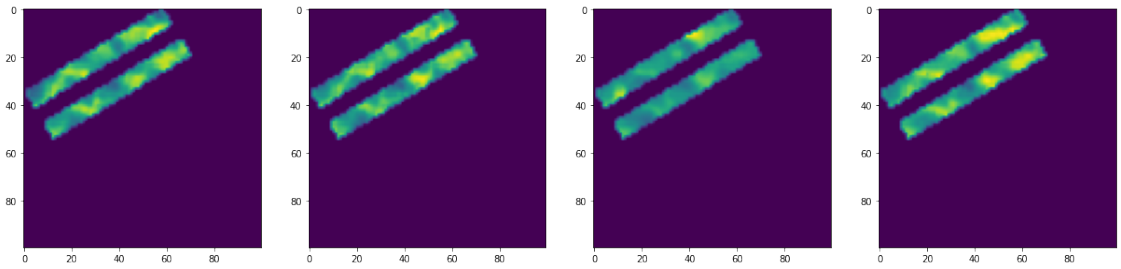
[21]:
```
imzoom2 = zoom(fullim, .4)
visualize(imzoom2)
```

```
(100, 100, 4)
```



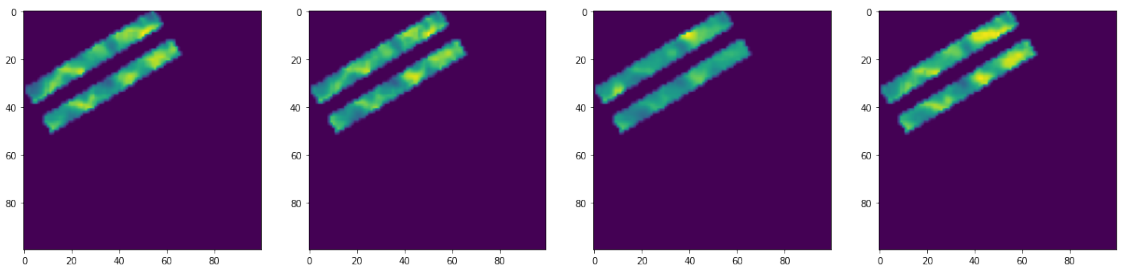
[22]:
```
imzoom3 = zoom(fullim, .75)
visualize(imzoom3)
```

```
(100, 100, 4)
```



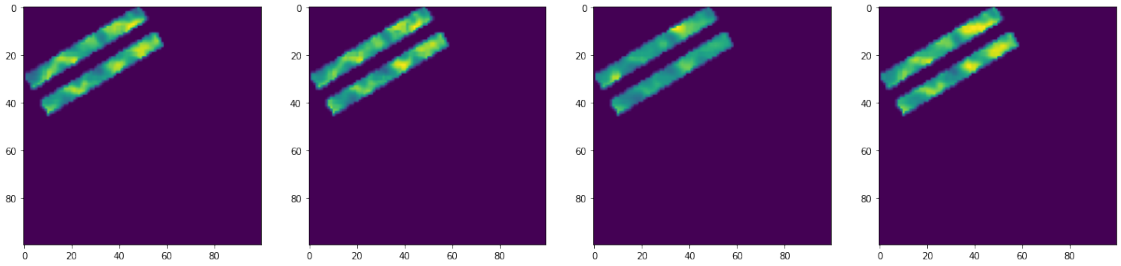
[23]:
```
imzoom3 = zoom(fullim, .8)
visualize(imzoom3)
```

```
(100, 100, 4)
```



[24]:
```
imzoom3 = zoom(fullim, .9)
visualize(imzoom3)
```

```
(100, 100, 4)
```

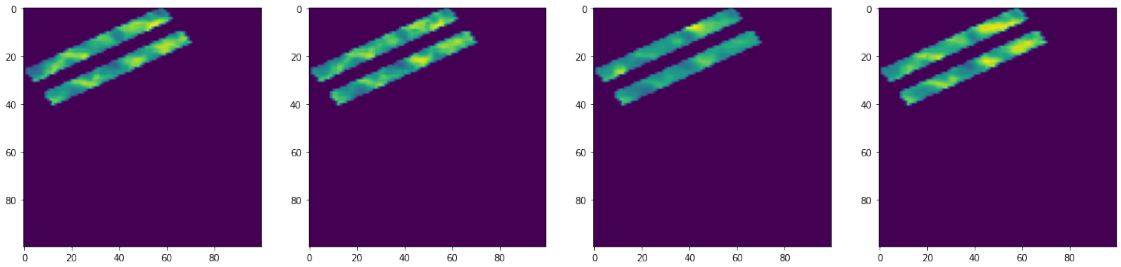


1.3.3 Shift vertical e horizontal

O Shift pode alongar a imagem, prejudicando sua nitidez, além de produzir cortes na imagem.

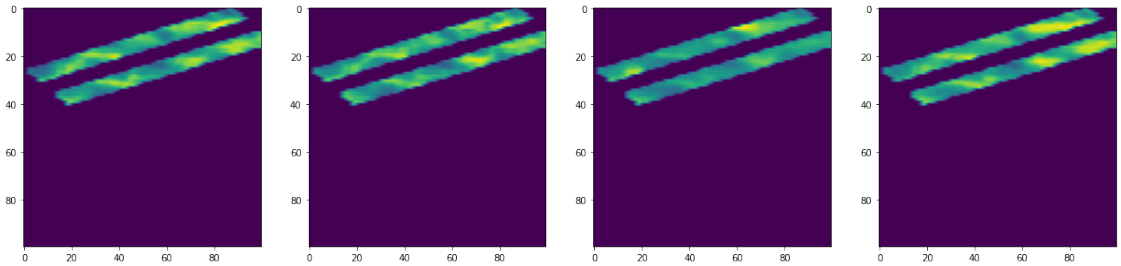
[25]:
```
im_shift = horizontal_shift(fullim, .25)
visualize(im_shift)
```

```
(100, 100, 4)
```



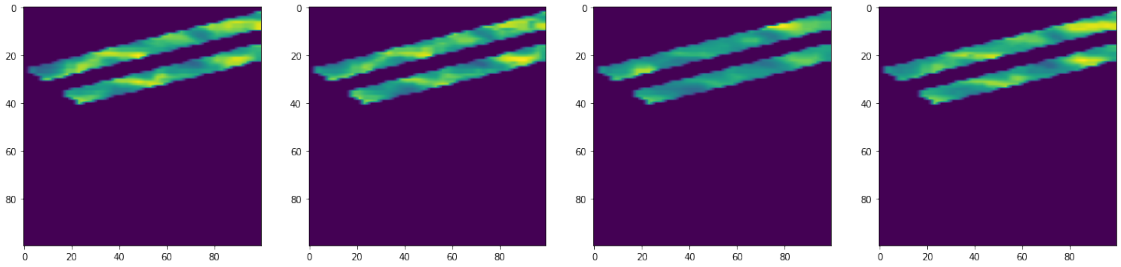
[26]:
```
im_shift = horizontal_shift(fullim, .50)
visualize(im_shift)
```

```
(100, 100, 4)
```



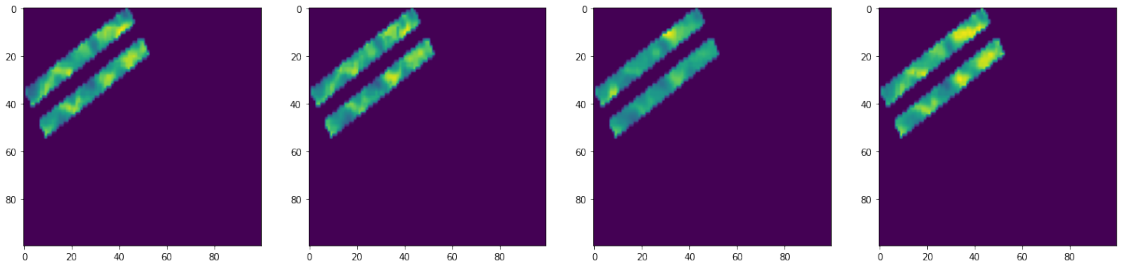
[27]:
```
im_shift = horizontal_shift(fullim, .60)
visualize(im_shift)
```

```
(100, 100, 4)
```



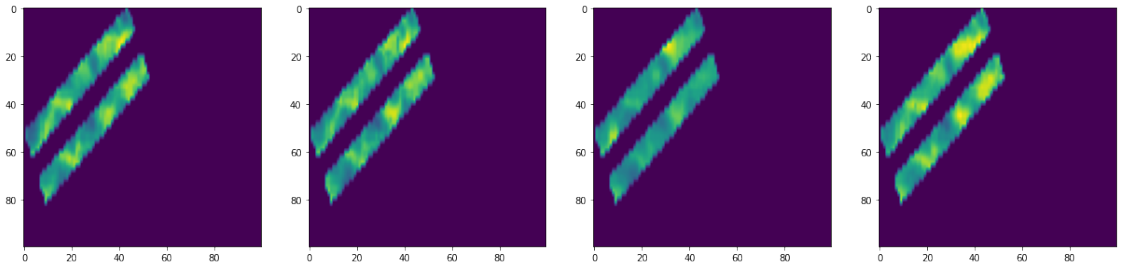
[28]:
```
im_shift = vertical_shift(fullim, .25)
visualize(im_shift)
```

```
(100, 100, 4)
```



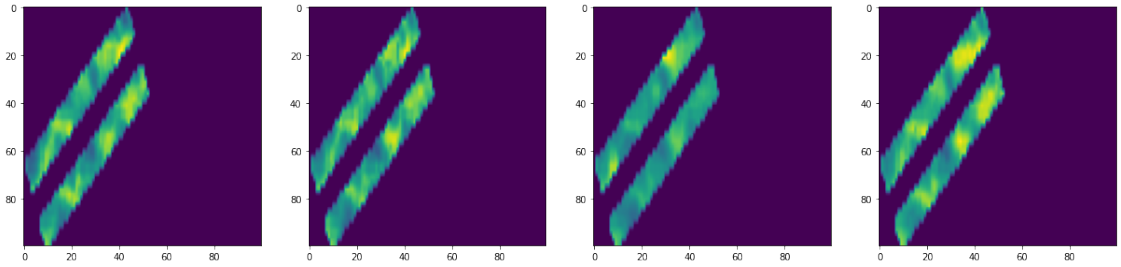
[29]:
```
im_shift = vertical_shift(fullim, .5)
visualize(im_shift)
```

```
(100, 100, 4)
```



[30]:
```
im_shift = vertical_shift(fullim, .60)
visualize(im_shift)
```

```
(100, 100, 4)
```

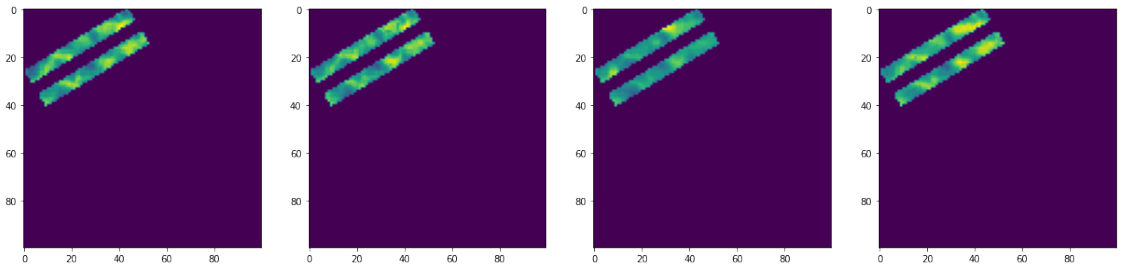


1.4 Exportando

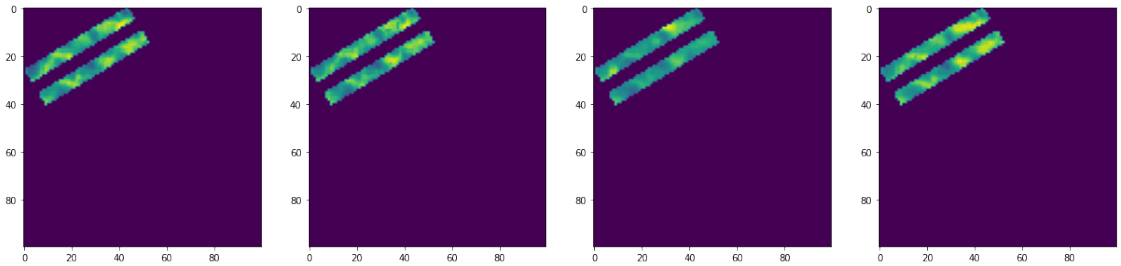
[31]:
```
export(fullim, 'export')
```

[32]:
```
im_array2 = np.load('export.npy')
visualize(im_array2)
visualize(fullim)
```

```
(100, 100, 4)
```

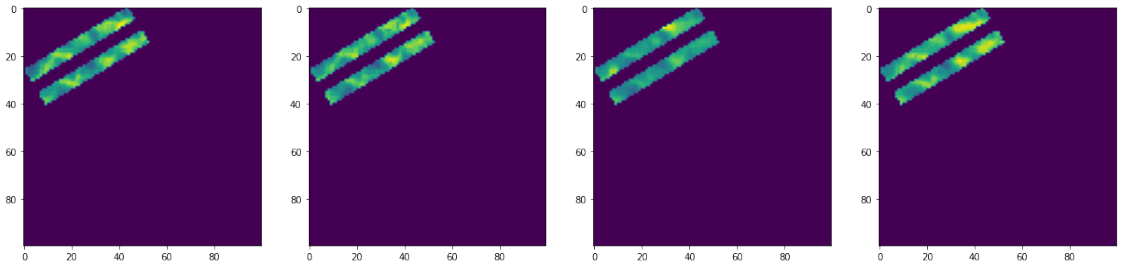


```
(100, 100, 4)
```



[33]:
```
from libtiff import TIFF
im_array2 = TIFF.open('export.tif').read_image()
print(im_array2.shape)
visualize(im_array2)
```

```
(100, 100, 4)
(100, 100, 4)
```



[ ]:
```

```