GSA

December 6, 2023

1 Dataset

- Satellite- WorldView-2
- Ground sampling distance- 0.46m PAN 1.84m MS
- The number of bands for MS image- 8
- The radiometric resolution in bits- 11
- Location- Miami, US

2 Import packages

[1]:
```
import numpy as np
import cv2
import os
import scipy.io as sio
from scipy import ndimage
from scipy import signal
import scipy.misc as misc
from PIL import Image
from IPython.display import display, HTML
```

3 Utils

[2]:
```
def upsample_interp23(image, ratio):

    image = np.transpose(image, (2, 0, 1))

    b,r,c = image.shape

    CDF23 = 2*np.array([0.5, 0.305334091185, 0, -0.072698593239, 0, 0.
↪021809577942, 0, -0.005192756653, 0, 0.000807762146, 0, -0.000060081482])
    d = CDF23[::-1]
    CDF23 = np.insert(CDF23, 0, d[:-1])
    BaseCoeff = CDF23

    first = 1
    for z in range(1,int(np.log2(ratio))+1):
```

```
        I1LRU = np.zeros((b, 2**z*r, 2**z*c))
        if first:
            I1LRU[:, 1:I1LRU.shape[1]:2, 1:I1LRU.shape[2]:2]=image
            first = 0
        else:
            I1LRU[:,0:I1LRU.shape[1]:2,0:I1LRU.shape[2]:2]=image

        for ii in range(0,b):
            t = I1LRU[ii,:,:]
            for j in range(0,t.shape[0]):
                t[j,:]=ndimage.correlate(t[j,:],BaseCoeff,mode='wrap')
            for k in range(0,t.shape[1]):
                t[:,k]=ndimage.correlate(t[:,k],BaseCoeff,mode='wrap')
            I1LRU[ii,:,:]=t
        image = I1LRU

    re_image=np.transpose(I1LRU, (1, 2, 0))

    return re_image
```

4 GSA (Adaptive Gram Schmidt)

[3]:
```
def estimation_alpha(pan, hs, mode='global'):
    if mode == 'global':
        IHC = np.reshape(pan, (-1, 1))
        ILRC = np.reshape(hs, (hs.shape[0]*hs.shape[1], hs.shape[2]))

        alpha = np.linalg.lstsq(ILRC, IHC,rcond=None)[0]

    elif mode == 'local':
        patch_size = 32
        all_alpha = []
        print(pan.shape)
        for i in range(0, hs.shape[0]-patch_size, patch_size):
            for j in range(0, hs.shape[1]-patch_size, patch_size):
                patch_pan = pan[i:i+patch_size, j:j+patch_size, :]
                patch_hs = hs[i:i+patch_size, j:j+patch_size, :]

                IHC = np.reshape(patch_pan, (-1, 1))
                ILRC = np.reshape(patch_hs, (-1, hs.shape[2]))

                local_alpha = np.linalg.lstsq(ILRC, IHC)[0]
                all_alpha.append(local_alpha)

        all_alpha = np.array(all_alpha)
```

```
        alpha = np.mean(all_alpha, axis=0, keepdims=False)

    return alpha

def GSA(pan, hs):

    M, N, c = pan.shape
    m, n, C = hs.shape

    ratio = int(np.round(M/m))

    print('get sharpening ratio: ', ratio)
    assert int(np.round(M/m)) == int(np.round(N/n))

    #upsample
    u_hs = upsample_interp23(hs, ratio)

    #remove means from u_hs
    means = np.mean(u_hs, axis=(0, 1))
    image_lr = u_hs-means

    #remove means from hs
    image_lr_lp = hs-np.mean(hs, axis=(0,1))

    #sintetic intensity
    image_hr = pan-np.mean(pan)
    image_hr0 = cv2.resize(image_hr, (n, m), cv2.INTER_CUBIC)
    image_hr0 = np.expand_dims(image_hr0, -1)

    alpha = estimation_alpha(image_hr0, np.concatenate((image_lr_lp, np.
 ↪ones((m, n, 1))), axis=-1), mode='global')

    I = np.dot(np.concatenate((image_lr, np.ones((M, N, 1))), axis=-1), alpha)

    I0 = I-np.mean(I)

    #computing coefficients
    g = []
    g.append(1)

    for i in range(C):
        temp_h = image_lr[:, :, i]
        c = np.cov(np.reshape(I0, (-1,)), np.reshape(temp_h, (-1,)), ddof=1)
        g.append(c[0,1]/np.var(I0))
    g = np.array(g)

    #detail extraction
```

```
    delta = image_hr-I0
    deltam = np.tile(delta, (1, 1, C+1))

    #fusion
    V = np.concatenate((I0, image_lr), axis=-1)

    g = np.expand_dims(g, 0)
    g = np.expand_dims(g, 0)

    g = np.tile(g, (M, N, 1))

    V_hat = V + g*deltam

    I_GSA = V_hat[:, :, 1:]

    I_GSA = I_GSA - np.mean(I_GSA, axis=(0, 1)) + means

    #adjustment
    I_GSA[I_GSA<0]=0
    I_GSA[I_GSA>1]=1

    return np.uint8(I_GSA*255)
```

5 Load Data

[4]:
```
import imageio.v2 as imageio
import matplotlib.pyplot as plt

# Specify the path to your TIFF file
ms_tiff_path = 'PAirMax/W2_Miam_Urb/FR/MS_LR.tif'
pan_tiff_path = 'PAirMax/W2_Miam_Urb/FR/PAN.tif'

# Read the TIFF image
ms = imageio.imread(ms_tiff_path)
pan = imageio.imread(pan_tiff_path)

pan = np.expand_dims(pan, -1)
# ms= np.transpose(ms, (0, 1, 2))

print(pan.shape,ms.shape)

# Set the figure size
fig, axes = plt.subplots(1, 2, figsize=(14, 7))  # Adjust the figsize as needed

# Display the first image
axes[0].imshow(pan[:, :], cmap='gray')  # Assuming the image is grayscale
```

```
axes[0].set_title('High Resolution Panchromatic Image')
axes[0].axis('off')

# Display the second image
axes[1].imshow(ms[:, :, 1], cmap='gray')  # Assuming the image is grayscale
axes[1].set_title('Low Resolution Multispectral Image')
axes[1].axis('off')

# Adjust layout to prevent overlapping
plt.tight_layout()

# Show the plot
plt.show()



# preprocess
# pan = np.expand_dims(pan, -1)
# ms= np.transpose(ms, (1, 2, 0))

max_patch, min_patch = np.max(ms, axis=(0,1)), np.min(ms, axis=(0,1))
ms = np.float32(ms-min_patch) / (max_patch - min_patch)
max_patch, min_patch = np.max(pan, axis=(0,1)), np.min(pan, axis=(0,1))
pan = np.float32(pan-min_patch) / (max_patch - min_patch)
```

```
(2048, 2048, 1) (512, 512, 8)
```

6 Apply GSA Algorithm

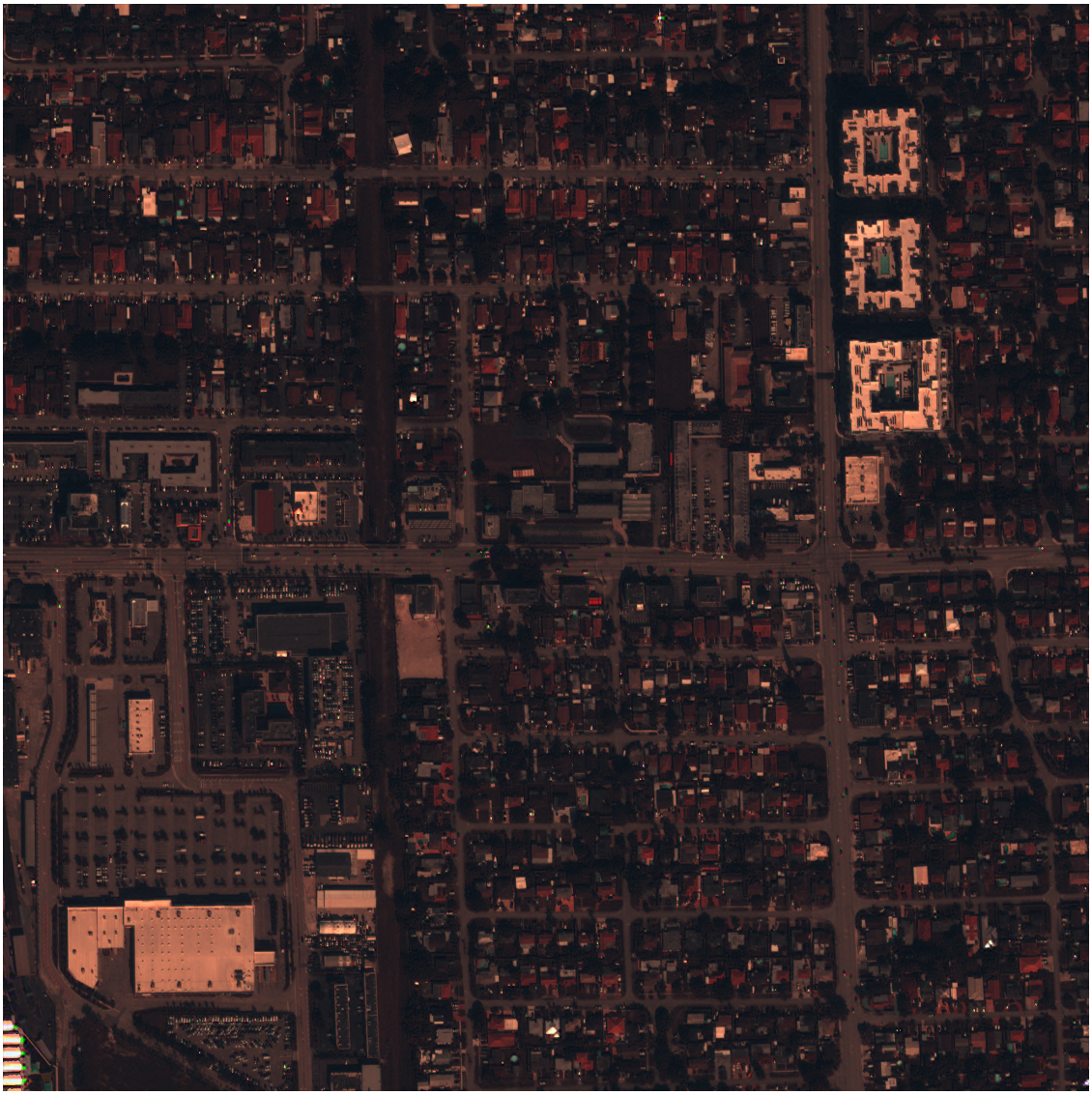
[5]:
```
fused_img = GSA(pan[:, :, :], ms[:, :, :])
print(pan.shape,ms.shape,fused_img.shape)

save_dir='./results/'
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)

cv2.imwrite(save_dir+'GSA.tiff', fused_img[:, :, [0,1,3]])
```

```
get sharpening ratio:  4
(2048, 2048, 1) (512, 512, 8) (2048, 2048, 8)
```

[5]:
```
True
```

7 Print the Resulting Image

[6]:
```
# Specify the path to your TIFF image
image_path = 'results/GSA.tiff'

# Open the TIFF image using PIL
image = Image.open(image_path)

# Display the image in the Jupyter Notebook with a centered title
display(image)
display(HTML("<div style='text-align: center;'><h1>High Resolution␣
 ↪Multispectral Image</h1></div>"))
```



```
<IPython.core.display.HTML object>
```

8 PSNR

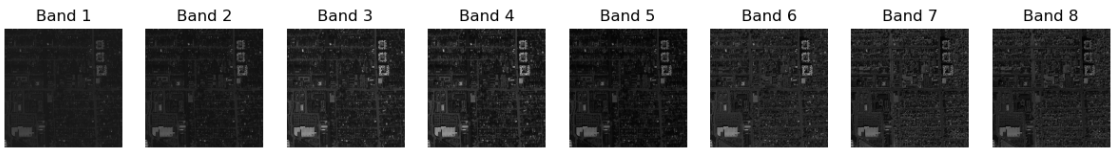
- PSNR $<$ 20 dB: Poor quality, significant loss of information.
- 20 dB $<$ PSNR $<$ 30 dB: Fair quality, moderate loss of information.
- 30 dB $<$ PSNR $<$ 40 dB: Good quality, acceptable for many applications.
- PSNR $>$ 40 dB: Very good to excellent quality.

[7]:
```python
def psnr(img1, img2, dynamic_range=255):
    """PSNR metric, img uint8 if 225; uint16 if 2047"""
    if not  img1.shape == img2.shape:
        raise ValueError('Input images must have the same dimensions.')
```

```
    img1_ = img1.astype(np.float64)
    img2_ = img2.astype(np.float64)
    mse = np.mean((img1_ - img2_)**2)
    if mse <= 1e-10:
        return np.inf
    return 20 * np.log10(dynamic_range / (np.sqrt(mse) + np.finfo(np.float64).
↪eps))
```

9 Load Reference Image

[8]:
```
import imageio
import matplotlib.pyplot as plt

# Specify the path to your TIFF image
reference_image_path = 'PAirMax/W2_Miam_Urb/FR/MS.tif'

# Read the TIFF image using imageio.v2
reference_image = imageio.v2.imread(reference_image_path)

# Get the number of bands in the image
num_bands = reference_image.shape[2]

# Display each band separately
fig, axes = plt.subplots(1, num_bands, figsize=(15, 5))

for i in range(num_bands):
    axes[i].imshow(reference_image[:, :, i], cmap='gray')
    axes[i].set_title(f'Band {i + 1}')
    axes[i].axis('off')

plt.show()
```



[9]:
```
# Normalization
max_patch, min_patch = np.max(reference_image, axis=(0,1)), np.
↪min(reference_image, axis=(0,1))
reference_image = np.float32(reference_image-min_patch) / (max_patch -␣
↪min_patch)
reference_image=np.uint8(reference_image*255)
```

[10]:
```
psnr(fused_img,reference_image)
```

[10]: 24.78061183549462