Reading Multi Spectral Images

https://nbviewer.jupyter.org/github/thomasaarholt/hyperspy-demos/blob/master/2_SVD_and_BSS.ipynb (https://nbviewer.jupyter.org/github/thomasaarholt/hyperspy-demos/blob/master/2_SVD_and_BSS.ipynb)

Bands and Wavelengths

When talking about spectral data, we talk from both, the electromagnetic spectrum and image bands. Spectral remote sensing data are collected by powerful camera-like instruments known as imaging spectrometers. Imaging spectrometers collect reflected light energy in “bands.”

A band represents a segment of the electromagnetic spectrum. For example, the wavelength values between 800 nanometers (nm) and 850 nm might be one band captured by an imaging spectrometer. The imaging spectrometer collects reflected light energy within a pixel area on the ground. Since an imaging spectrometer collects many different types of light - for each pixel the amount of light energy for each type of light or band will be recorded. So, for example, a camera records the amount of red, green and blue light for each pixel.

Often when we work with a multispectral dataset, the band information is reported as the center wavelength value. This value represents the center point value of the wavelengths represented in that band. Thus in a band spanning 800-850 nm, the center would be 825 nm.

Spectral Resolution

The spectral resolution of a dataset that has more than one band, refers to the spectral width of each band in the dataset. While a general spectral resolution of the sensor is often provided, not all sensors collect information within bands of uniform widths.

Spatial Resolution

The spatial resolution of a raster represents the area on the ground that each pixel covers. If you have smaller pixels in a raster the data will appear more “detailed.” If you have large pixels in a raster, the data will appear more coarse or “fuzzy.”

Multispectral Imagery

Images obtained with a ADC Lite - Tetracam's Lightweight ADC

I made pitures about:

> Aluminum , Copper, Brass, Iron, Stainless Steel, Painted Iron

http://tetracam.com/Products-ADC_Lite.htm (http://tetracam.com/Products-ADC_Lite.htm)

MRobalinho - 25-03-2019

In [1]:

```
# Add libraries
import glob, os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image
from openpyxl import load_workbook
```

In [2]:

```
# Verify my current folder
currDir = os.path.dirname(os.path.realpath("__file__"))
mypath = currDir
print(currDir)
```

```
C:\Users\manuel.robalinho\Google Drive\UPT_Portucalense\Trabalho final\Class
ificacao_Sucata\Jupyter_Notebook
```

In [3]:

```
# Path to the image files
folder = "imagedata05"
path = currDir + "/" + folder + "/"

# Part name of file to filter files
end_file = ".jpg"
```

In [4]:

```
# Read files from folder
print(path)
print(' ---- IMAGES ON THE FOLDER -------')

for file in os.listdir(path):
    if file.endswith(end_file):
        print(os.path.join(file))
```

```
C:\Users\manuel.robalinho\Google Drive\UPT_Portucalense\Trabalho final\Class
ificacao_Sucata\Jupyter_Notebook/imagedata05/
 ---- IMAGES ON THE FOLDER -------
Aluminum_1.jpg
Aluminum_2.jpg
Aluminum_3.jpg
Brass_1.jpg
Brass_2.jpg
Brass_3.jpg
Brass_4.jpg
Brass_5.jpg
CopperWire_1.jpg
CopperWire_2.jpg
CopperWire_3.jpg
Copper_1.jpg
Copper_2.jpg
Copper_3.jpg
Copper_4.jpg
Iron_1.jpg
Iron_2.jpg
Iron_3.jpg
PaintedIron_1.jpg
PaintedIron_2.jpg
StainlessSteel_1.jpg
StainlessSteel_2.jpg
StainlessSteel_3.jpg
```

In [5]:

```
# Create Data Frame with image information
df_image = []
```

In [6]:

```
# Look from an chanel from then image

def channel(img, n):
    """Isolate the nth channel from the image.

       n = 0: red, 1: green, 2: blue
    """
    a = np.array(img)
    a[:,:,(n!=0, n!=1, n!=2)] *= 0
#   a[:,:,n] *= 0
#   print(Image.fromarray(a), 'Get Channel n: ', n)

    print('Get Channel n: ', n)
    return Image.fromarray(a)

# def to resize
# Given parameters : image , number to divide (resize)
def imageResize(img, n):
    width, height = img.size

    print('Original size:', width, '/', height, 'Resize:',n)

    newWidth = int(width / n)
    newHeight = int(height / n)
    img.resize((newWidth, newHeight), Image.ANTIALIAS)
    print('New size:', newWidth, '/', newHeight)
    return img
```

In [7]:

```
# Obtain main color from image
# https://convertingcolors.com/rgb-color-169_171_170.html

def get_main_color(path, file):
    img = Image.open(path+file)
    colors = img.getcolors( 1024*1024) #put a higher value if there are many colors in your
    print('Get main Color file:', file)
    max_occurence, most_present = 0, 0
    try:
        for c in colors:
            if c[0] > max_occurence:
                (max_occurence, most_present) = c
        return most_present
    except TypeError:
        raise Exception("Too many colors in the image")
```

In [8]:

```
def print_file(path, xfile):
  print('------------------------------------------------------------------------')
  tif_f1 = Image.open(path+xfile)

  print('Inf.File:',xfile)

  # Transform Image to array
  aArray = np.array(tif_f1)
  # Array sum
  xsum = aArray.sum() / 1000000

  # Get channel 0
  x0_channel = channel(tif_f1, 0)
  aArray = np.array(x0_channel)
  xsum_0 = aArray.sum() / 1000000

  # Get channel 1
  x1_channel = channel(tif_f1, 1)
  aArray = np.array(x1_channel)
  xsum_1 = aArray.sum() / 1000000

  # Get channel 2
  x2_channel = channel(tif_f1, 2)
  aArray = np.array(x2_channel)
  xsum_2 = aArray.sum() / 1000000

  # Histogram from image
  aHist = tif_f1.histogram()
  hsum = sum(aHist) / 100000

  # Histogram channel 0
  aHist_0 = x0_channel.histogram()
  hsum_0 = sum(aHist_0) / 100000

  # Histogram channel 1
  aHist_1 = x1_channel.histogram()
  hsum_1 = sum(aHist_1) / 100000

  # Histogram chanel 0
  aHist_2 = x2_channel.histogram()
  hsum_2 = sum(aHist_2) / 100000

  # number elements on list
  nlist = len(aHist)

  # Get color
  main_color = get_main_color(path, xfile)
  # Transform tuple in a list
  pix_color_a = [list(main_color) for x in main_color]
  pix_color_b = [x for sets in pix_color_a for x in sets]
  # Sum the list and medium list pixel
  sum_color = sum(pix_color_b)
  med_color = sum_color / len(pix_color_b)
  #print('List Color:',pix_color_a,'Sum:',sum_color,'Len:', len(pix_color_a), 'Med:',med_co

  # Get Extrems of the image
  extr_a = tif_f1.getextrema()
  # Transform tuple in a list
  extr_b = [x for sets in extr_a for x in sets]
```

```
  # Sum the list
  sum_list = sum(extr_b)
  med_extr  = sum_list / len(extr_b)
  #print('List Extremes:',extr_a,'Sum:',sum_list,'Len:', len(extr_b), 'Med:',med_extr)


  # Obtain name file without extension
  sample_name = os.path.basename(xfile).split('_')[0]

  # Print information
  print(sample_name,' Size:',tif_f1.size, ' Format:',tif_f1.format, ' Mode:', tif_f1.mode)
  print('          Sum array:',xsum, ' Sum Ch 0:', xsum_0, ' Sum Ch 1:', xsum_1, ' Sum Ch 2
  print('          Histog   :',hsum ,'  N.List elem:', nlist )
  print('          Color    :',main_color,'Med Color   :',med_color)
  print('          Extremes :',extr_a, 'Med Extremes:',med_extr)

   # insert information in a Pandas Data Frame
  df_image.append((folder, xfile, sample_name, tif_f1.size, tif_f1.format, tif_f1.mode ,
                   xsum, xsum_0, xsum_1, xsum_2, hsum, nlist, main_color, med_color, med_ex
```

In [9]:

```
# Create Data Frame with image information
df_image = []

xend_file = "*" + end_file
os.chdir(path)
for file in glob.glob(xend_file):
 #   print(file)
    print_file(path,file)
```

```
Get main Color file: StainlessSteel_2.jpg
StainlessSteel  Size: (5312, 2988)  Format: JPEG  Mode: RGB
          Sum array: 1963.320855  Sum Ch 0: 2107.296881  Sum Ch 1: 2094.15
5125  Sum Ch 2: 2056.836145
          Histog   : 476.16768   N.List elem: 768
          Color    : (156, 158, 157) Med Color   : 157.0
          Extremes : ((0, 255), (0, 255), (0, 255)) Med Extremes: 127.5
------------------------------------------------------------------------
Inf.File: StainlessSteel_3.jpg
Get Channel n:  0
Get Channel n:  1
Get Channel n:  2
Get main Color file: StainlessSteel_3.jpg
StainlessSteel  Size: (5312, 2988)  Format: JPEG  Mode: RGB
          Sum array: 1840.495744  Sum Ch 0: 2073.778109  Sum Ch 1: 2056.03
4992  Sum Ch 2: 2005.649939
          Histog   : 476.16768   N.List elem: 768
          Color    : (151, 151, 149) Med Color   : 150.33333333333334
          Extremes : ((0, 255), (0, 255), (0, 255)) Med Extremes: 127.5
```

In [11]:

```
df = pd.DataFrame(df_image,columns=['Folder','File','Material','Size','Format','Mode',
                                    'All_Bands', 'Sum_Ch0','Sum_Ch1','Sum_Ch2',
                                    'Histogram','Number_list_elements','Color','Med_Color',
df.head(100)
```

Out[11]:

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_
0	imagedata05	Aluminum_1.jpg	Aluminum	(5312, 2988)	JPEG	RGB	2195.247943	2168.40
1	imagedata05	Aluminum_2.jpg	Aluminum	(5312, 2988)	JPEG	RGB	2257.032332	2187.40
2	imagedata05	Aluminum_3.jpg	Aluminum	(5312, 2988)	JPEG	RGB	2380.904071	2235.25
3	imagedata05	Brass_1.jpg	Brass	(5312, 2988)	JPEG	RGB	2347.971869	2256.69
4	imagedata05	Brass_2.jpg	Brass	(5312, 2988)	JPEG	RGB	3042.322194	2557.93
5	imagedata05	Brass_3.jpg	Brass	(5312, 2988)	JPEG	RGB	2343.831242	2255.41
6	imagedata05	Brass_4.jpg	Brass	(5312, 2988)	JPEG	RGB	2305.312821	2270.56
7	imagedata05	Brass_5.jpg	Brass	(5312, 2988)	JPEG	RGB	2491.403767	2301.78
8	imagedata05	CopperWire_1.jpg	CopperWire	(5312, 2988)	JPEG	RGB	267.892644	1556.58
9	imagedata05	CopperWire_2.jpg	CopperWire	(5312, 2988)	JPEG	RGB	400.634317	1608.22
10	imagedata05	CopperWire_3.jpg	CopperWire	(5312, 2988)	JPEG	RGB	555.143850	1658.52
11	imagedata05	Copper_1.jpg	Copper	(5312, 2988)	JPEG	RGB	2475.739855	2340.82
12	imagedata05	Copper_2.jpg	Copper	(5312, 2988)	JPEG	RGB	2512.756599	2351.11
13	imagedata05	Copper_3.jpg	Copper	(5312, 2988)	JPEG	RGB	3403.152445	2732.99
14	imagedata05	Copper_4.jpg	Copper	(5312, 2988)	JPEG	RGB	2524.999454	2367.93

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_
15	imagedata05	Iron_1.jpg	Iron	(5312, 2988)	JPEG	RGB	2125.839781	2155.84(
16	imagedata05	Iron_2.jpg	Iron	(5312, 2988)	JPEG	RGB	2058.051761	2137.30(
17	imagedata05	Iron_3.jpg	Iron	(5312, 2988)	JPEG	RGB	2034.603394	2123.15(
18	imagedata05	PaintedIron_1.jpg	PaintedIron	(5312, 2988)	JPEG	RGB	2388.517537	2242.17!
19	imagedata05	PaintedIron_2.jpg	PaintedIron	(5312, 2988)	JPEG	RGB	2406.412176	2244.08(
20	imagedata05	StainlessSteel_1.jpg	StainlessSteel	(5312, 2988)	JPEG	RGB	2169.327805	2168.96(
21	imagedata05	StainlessSteel_2.jpg	StainlessSteel	(5312, 2988)	JPEG	RGB	1963.320855	2107.29(
22	imagedata05	StainlessSteel_3.jpg	StainlessSteel	(5312, 2988)	JPEG	RGB	1840.495744	2073.77(

In [12]:

```
# Verify my current folder
path = mypath + r"/upt_data.xlsx"
print('Write statistics into file :', path)

# Block to Read excel old excel file
book = load_workbook(path)
writer = pd.ExcelWriter(path, engine = 'openpyxl')
writer.book = book
# ------------------------

# Write statistics into excel file
#writer = pd.ExcelWriter(path, engine = 'xlsxwriter') # only for new excelfile
df.to_excel(writer, sheet_name = folder)
writer.save()
writer.close()
```

```
Write statistics into file : C:\Users\manuel.robalinho\Google Drive\UPT_Port
ucalense\Trabalho final\Classificacao_Sucata\Jupyter_Notebook/upt_data.xlsx
```

In [13]:

```
df_plot = pd.DataFrame(df, columns=["Material", "All_Bands", "Sum_Ch0", "Sum_Ch1", "Sum_Ch2
df_plot
```

Out[13]:

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extren
0	Aluminum	2195.247943	2168.407625	2181.081817	2140.725797	145.333333	126.66666
1	Aluminum	2257.032332	2187.401438	2200.808299	2163.789891	148.333333	127.50000
2	Aluminum	2380.904071	2235.251031	2235.978671	2204.641665	150.333333	127.66666
3	Brass	2347.971869	2256.695208	2224.776302	2161.467655	161.000000	107.50000
4	Brass	3042.322194	2557.932421	2458.105183	2321.251886	170.000000	133.83333
5	Brass	2343.831242	2255.419735	2226.972316	2156.406487	163.000000	101.16666
6	Brass	2305.312821	2270.568593	2204.614105	2125.097419	174.333333	107.00000
7	Brass	2491.403767	2301.781831	2269.732149	2214.857083	165.000000	108.33333
8	CopperWire	267.892644	1556.587269	1522.765520	1483.507151	163.000000	105.00000
9	CopperWire	400.634317	1608.228710	1563.715691	1523.657212	161.000000	104.50000
10	CopperWire	555.143850	1658.528294	1611.116925	1580.465927	169.000000	102.33333
11	Copper	2475.739855	2340.824072	2233.916684	2195.966395	176.000000	103.66666
12	Copper	2512.756599	2351.116814	2245.775493	2210.831588	177.000000	102.16666
13	Copper	3403.152445	2732.991551	2548.093134	2417.035056	203.333333	135.16666
14	Copper	2524.999454	2367.930725	2244.869608	2207.166417	180.000000	106.83333
15	Iron	2125.839781	2155.840271	2139.247180	2125.719626	172.666667	122.00000
16	Iron	2058.051761	2137.300730	2122.972972	2092.745355	167.666667	115.00000
17	Iron	2034.603394	2123.156255	2114.419050	2091.995385	169.666667	126.50000
18	PaintedIron	2388.517537	2242.175234	2244.439610	2196.869989	148.333333	90.00000
19	PaintedIron	2406.412176	2244.088768	2251.928704	2205.362000	146.333333	89.66666
20	StainlessSteel	2169.327805	2168.962604	2165.083051	2130.249446	159.000000	127.50000
21	StainlessSteel	1963.320855	2107.296881	2094.155125	2056.836145	157.000000	127.50000
22	StainlessSteel	1840.495744	2073.778109	2056.034992	2005.649939	150.333333	127.50000

In [14]:

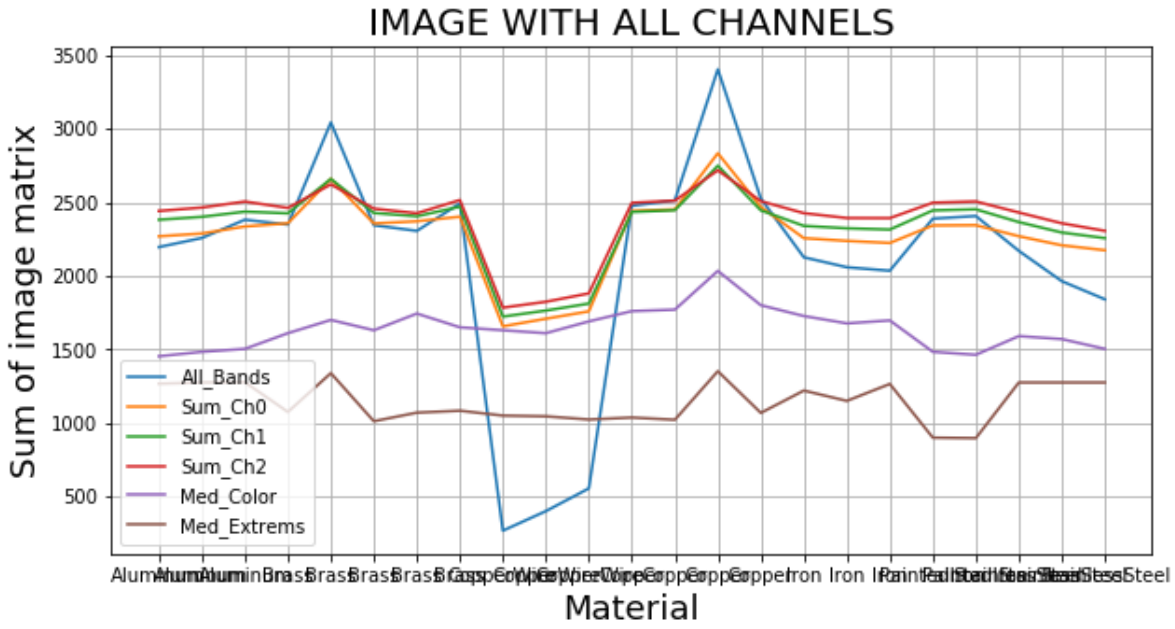
```
df_plot.Sum_Ch0 = df_plot.Sum_Ch0 + 100 # to have diference lines during plot
df_plot.Sum_Ch1 = df_plot.Sum_Ch1 + 200
df_plot.Sum_Ch2 = df_plot.Sum_Ch2 + 300
df_plot.Med_Color  = df_plot.Med_Color * 10
df_plot.Med_Extrems  = df_plot.Med_Extrems * 10
df_plot
```

Out[14]:

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extre
0	Aluminum	2195.247943	2268.407625	2381.081817	2440.725797	1453.333333	1266.666
1	Aluminum	2257.032332	2287.401438	2400.808299	2463.789891	1483.333333	1275.000
2	Aluminum	2380.904071	2335.251031	2435.978671	2504.641665	1503.333333	1276.666
3	Brass	2347.971869	2356.695208	2424.776302	2461.467655	1610.000000	1075.000
4	Brass	3042.322194	2657.932421	2658.105183	2621.251886	1700.000000	1338.333
5	Brass	2343.831242	2355.419735	2426.972316	2456.406487	1630.000000	1011.666
6	Brass	2305.312821	2370.568593	2404.614105	2425.097419	1743.333333	1070.000
7	Brass	2491.403767	2401.781831	2469.732149	2514.857083	1650.000000	1083.333
8	CopperWire	267.892644	1656.587269	1722.765520	1783.507151	1630.000000	1050.000
9	CopperWire	400.634317	1708.228710	1763.715691	1823.657212	1610.000000	1045.000
10	CopperWire	555.143850	1758.528294	1811.116925	1880.465927	1690.000000	1023.333
11	Copper	2475.739855	2440.824072	2433.916684	2495.966395	1760.000000	1036.666
12	Copper	2512.756599	2451.116814	2445.775493	2510.831588	1770.000000	1021.666
13	Copper	3403.152445	2832.991551	2748.093134	2717.035056	2033.333333	1351.666
14	Copper	2524.999454	2467.930725	2444.869608	2507.166417	1800.000000	1068.333
15	Iron	2125.839781	2255.840271	2339.247180	2425.719626	1726.666667	1220.000
16	Iron	2058.051761	2237.300730	2322.972972	2392.745355	1676.666667	1150.000
17	Iron	2034.603394	2223.156255	2314.419050	2391.995385	1696.666667	1265.000
18	PaintedIron	2388.517537	2342.175234	2444.439610	2496.869989	1483.333333	900.000
19	PaintedIron	2406.412176	2344.088768	2451.928704	2505.362000	1463.333333	896.666
20	StainlessSteel	2169.327805	2268.962604	2365.083051	2430.249446	1590.000000	1275.000
21	StainlessSteel	1963.320855	2207.296881	2294.155125	2356.836145	1570.000000	1275.000
22	StainlessSteel	1840.495744	2173.778109	2256.034992	2305.649939	1503.333333	1275.000

In [15]:

```
df_plot.plot(y=["All_Bands","Sum_Ch0","Sum_Ch1", "Sum_Ch2","Med_Color","Med_Extrems"],figsi

# Obtain legend (xticks) for X axis
loc_Array_sum = np.arange(len(df_plot.index))
# Position of X labels
xtick_loc = list(loc_Array_sum)
# Name of x labels
xticks = list(df_plot.Material)
#-------

#plt.plot(df_plot.Array_sum)
plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.ylabel('Sum of image matrix',fontsize=18)
plt.xticks(xtick_loc, df_plot.Material, rotation=0)
plt.xlabel('Material',fontsize=18)
plt.show()
```



In [16]:

```
# Create pivot table
df_plot1 = df_plot.groupby('Material')['All_Bands', 'Sum_Ch0','Sum_Ch1','Sum_Ch2','Med_Colc
df_plot1
```

Out[16]:

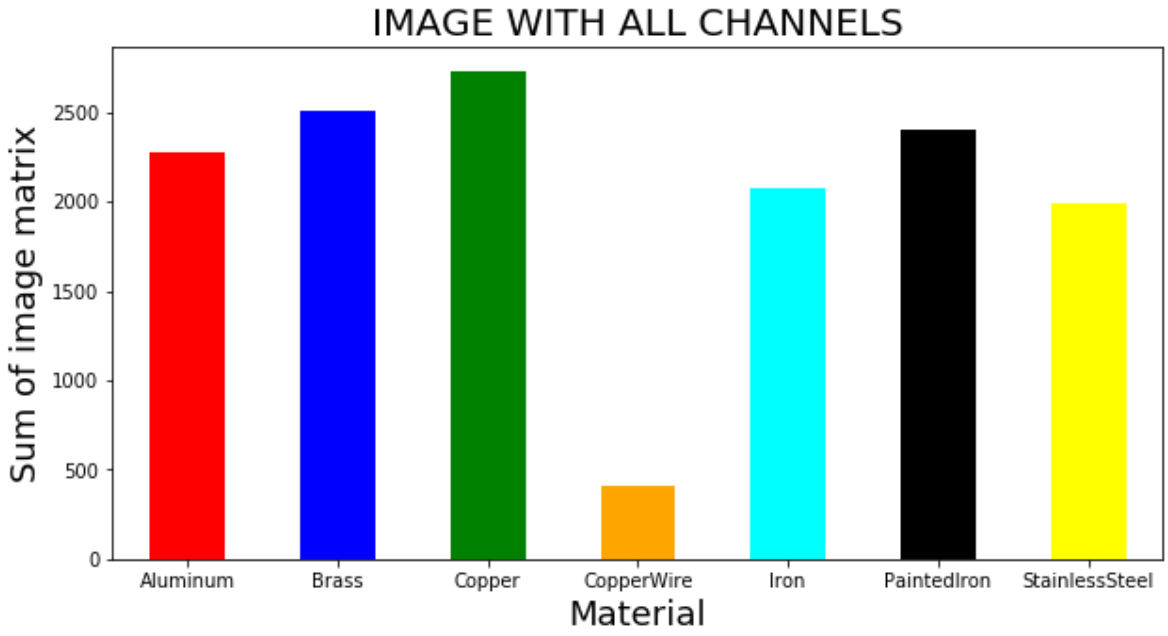
Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extrems
Aluminum	2277.728115	2297.020031	2405.956262	2469.719118	1480.000000	1272.777778
Brass	2506.168379	2428.479558	2476.840011	2495.816106	1666.666667	1115.666667
Copper	2729.162088	2548.215791	2518.163730	2557.749864	1840.833333	1119.583333
CopperWire	407.890270	1707.781424	1765.866045	1829.210097	1643.333333	1039.444444
Iron	2072.831645	2238.765752	2325.546401	2403.486789	1700.000000	1211.666667
PaintedIron	2397.464856	2343.132001	2448.184157	2501.115994	1473.333333	898.333333
StainlessSteel	1991.048135	2216.679198	2305.091056	2364.245177	1554.444444	1275.000000

In [17]:

```
df = pd.DataFrame(df_plot1.All_Bands)
color = ['red','blue','green','orange','cyan','black','yellow']
```

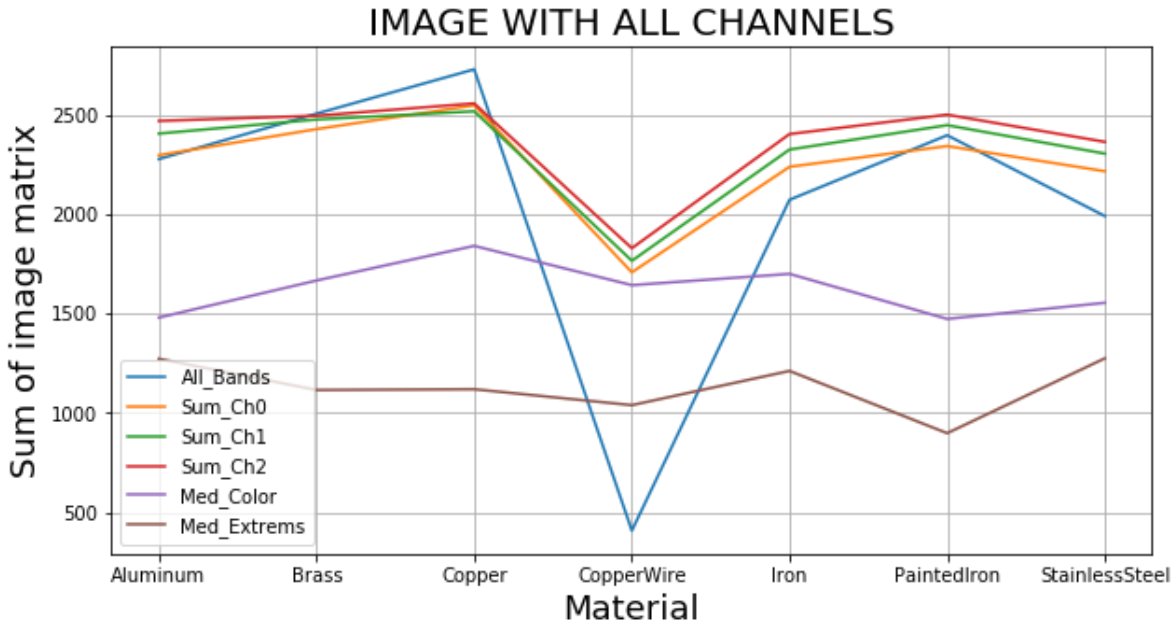
In [18]:

```
df.plot(kind='bar', y=0, color=color, legend=False, rot=0, figsize=(10,5))
plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Sum of image matrix',fontsize=18)
plt.show()
```



In [19]:

```
loc_Array_sum = np.arange(len(df_plot1.index))
xtick_loc = list(loc_Array_sum)
xticks = list(df_plot1.index)

df_plot1.plot( y=["All_Bands","Sum_Ch0","Sum_Ch1", "Sum_Ch2","Med_Color","Med_Extrems"],fig
plt.xticks(xtick_loc, df_plot1.index, rotation=0)
plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Sum of image matrix',fontsize=18)
plt.show()
```



In [20]:

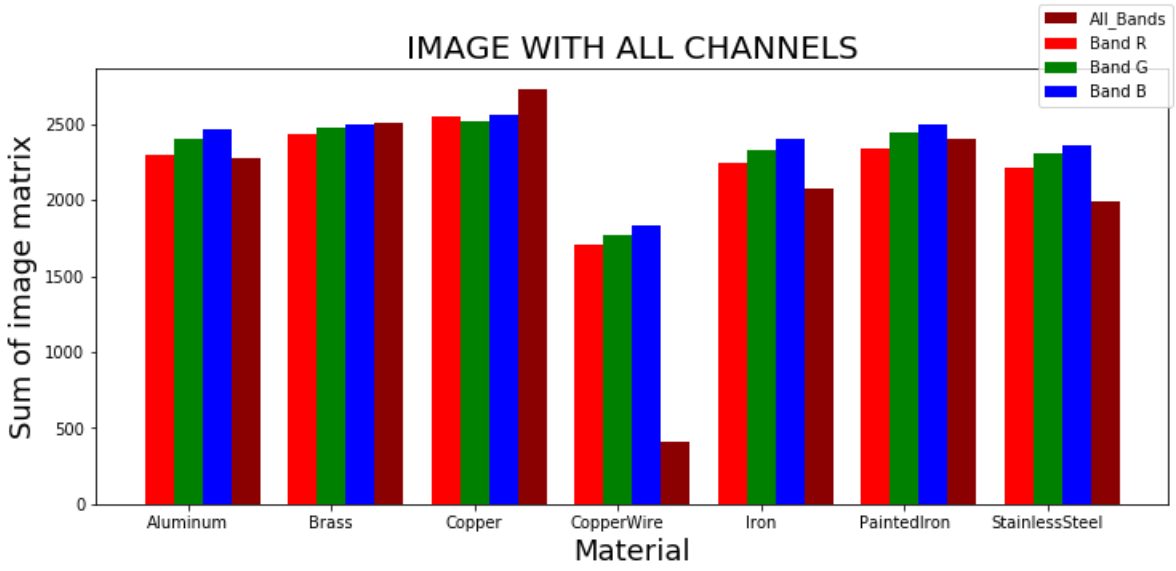
```
loc_Array_sum = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_r = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.3 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.5 # Offsetting the tick-label location

xtick_loc = list(loc_g)
xticks = list(df_plot1.index)
```

In [ ]:

```

```

In [21]:

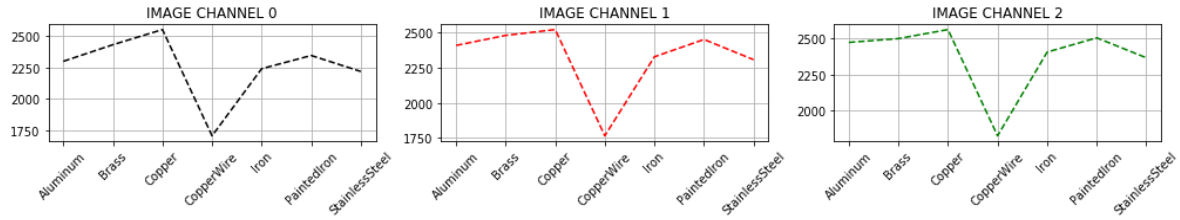
```
#Plot  Bar Graph
#df_plot1.plot(kind='bar', figsize=(12,5), grid=True, color='darkred',fontsize=18)
loc_Array_sum = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.3 # Offsetting the tick-label location
loc_r = np.arange(len(df_plot1.index))-0.5 # Offsetting the tick-label location

#xtick_loc = list(loc_Array_sum) + list(loc_r) + list(loc_g) + list(loc_b)
#xticks = list(selected.keys())+ list(rejected.keys())
colors = ['darkred','red','green','blue','orange','cyan','black','yellow']
plt.figure(figsize=(12,5))

plt.bar(loc_Array_sum, df_plot1.All_Bands, color=colors[0], width=0.2, label='All_Bands')
plt.bar(loc_r, df_plot1.Sum_Ch0, color=colors[1], width=0.2,label='Band R')
plt.bar(loc_g, df_plot1.Sum_Ch1, color=colors[2], width=0.2,label='Band G')
plt.bar(loc_b, df_plot1.Sum_Ch2, color=colors[3], width=0.2,label='Band B')

plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Sum of image matrix',fontsize=18)
plt.xticks(xtick_loc, xticks, rotation=0)
plt.legend(bbox_to_anchor=(.8,0.8),\
    bbox_transform=plt.gcf().transFigure)

plt.show()
```



In [22]:

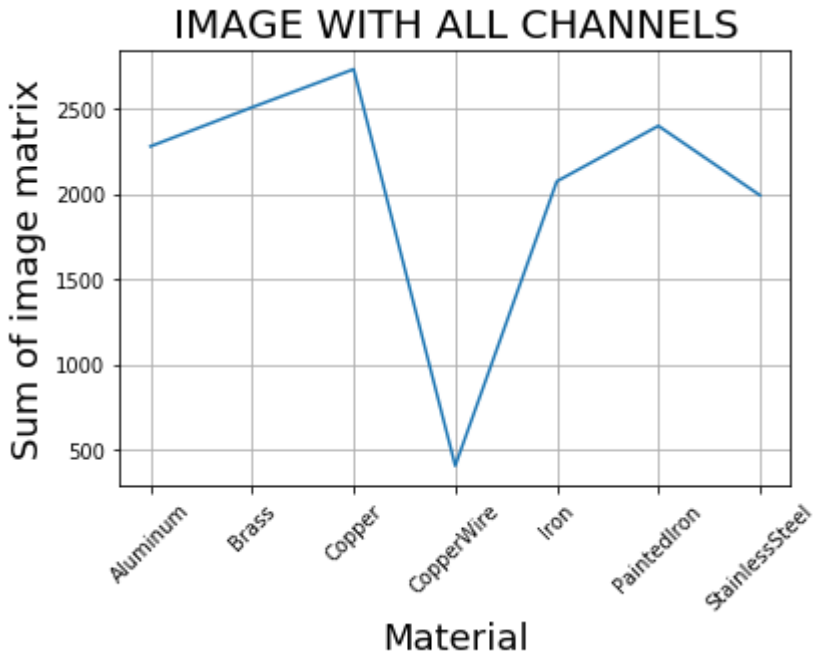
```
plt.figure(1)
plt.figure(figsize=(17, 4))
plt.tight_layout()
plt.subplot(231)
plt.title('IMAGE CHANNEL 0')
plt.xticks(rotation=45)
plt.grid(True)
plt.plot(df_plot1.Sum_Ch0, 'k--')

plt.subplot(232)
plt.title('IMAGE CHANNEL 1')
plt.xticks(rotation=45)
plt.grid(True)
plt.plot(df_plot1.Sum_Ch1,  'r--')

plt.subplot(233)
plt.title('IMAGE CHANNEL 2')
plt.xticks(rotation=45)
plt.plot(df_plot1.Sum_Ch2,  'g--')
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



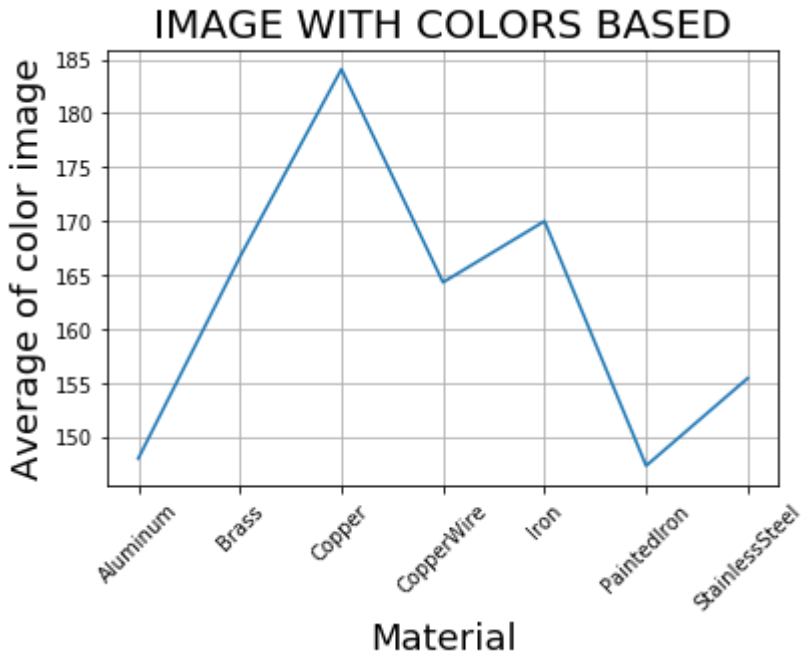
In [23]:

```
# Plot channel based
plt.plot(df_plot1.All_Bands)
plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Sum of image matrix',fontsize=18)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



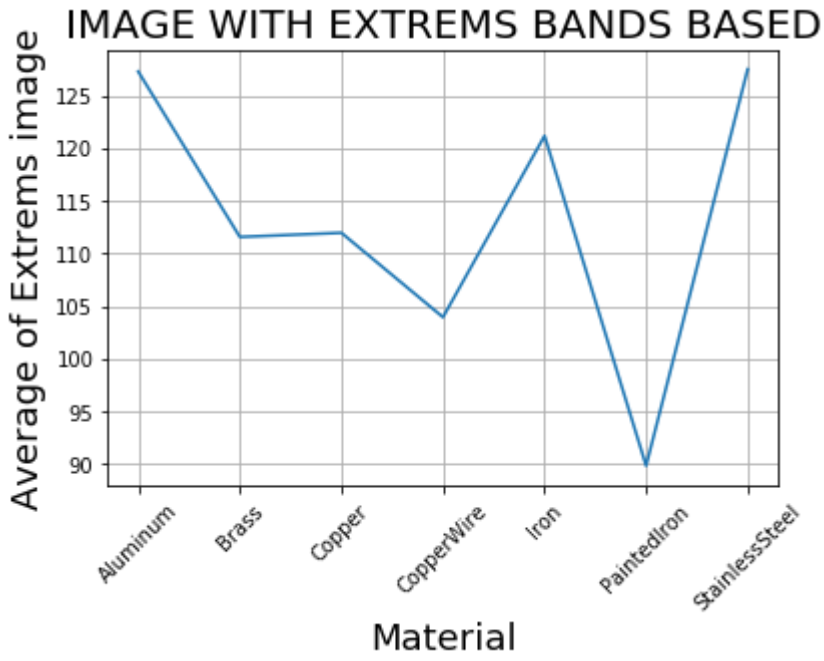
In [24]:

```
# Plot based on color
plt.plot(df_plot1.Med_Color/10)
plt.title('IMAGE WITH COLORS BASED',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Average of color image',fontsize=18)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [25]:

```
# Plot based on Extrems of the Bands
plt.plot(df_plot1.Med_Extrems/10)
plt.title('IMAGE WITH EXTREMS BANDS BASED',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Average of Extrems image',fontsize=18)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [ ]:

In [ ]: