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 = "imagedata04"
path = currDir + "/" + folder + "/"

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

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/imagedata04/
 ---- IMAGES ON THE FOLDER -------
Aluminum_1.TIF
Aluminum_2.TIF
Aluminum_3.TIF
Aluminum_4.TIF
Aluminum_5.TIF
Aluminum_6.TIF
Brass_1.TIF
Brass_2.TIF
Brass_3.TIF
CopperWire_1.TIF
CopperWire_2.TIF
CopperWire_3.TIF
CopperWire_4.TIF
Copper_1.TIF
Copper_2.TIF
Copper_3.TIF
Copper_4.TIF
Copper_5.TIF
Copper_6.TIF
Iron_1.TIF
Iron_2.TIF
Iron_3.TIF
Iron_4.TIF
Iron_5.TIF
PaintedIron_1.TIF
PaintedIron_2.TIF
StainlessSteel_1.TIF
StainlessSteel_2.TIF
StainlessSteel_3.TIF
StainlessSteel_4.TIF
```

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)
```

```
          Extremes : ((1, 255), (1, 255), (0, 255)) Med Extremes: 127.8333
3333333333
------------------------------------------------------------------------
Inf.File: StainlessSteel_4.TIF
Get Channel n:  0
Get Channel n:  1
Get Channel n:  2
Get main Color file: StainlessSteel_4.TIF
List Color: [[255, 255, 252], [255, 255, 252], [255, 255, 252]] Sum: 2286
Len: 3 Med: 254.0
List Extremes: ((1, 255), (1, 255), (0, 255)) Sum: 767 Len: 6 Med: 127.833
33333333333
StainlessSteel  Size: (2048, 1536)  Format: TIFF  Mode: RGB
          Sum array: 639.179701  Sum Ch 0: 214.73079  Sum Ch 1: 214.73079
Sum Ch 2: 209.718121
          Histog   : 94.37184   N.List elem: 768
          Color    : (255, 255, 252) Med Color   : 254.0
          Extremes : ((1, 255), (1, 255), (0, 255)) Med Extremes: 127.8333
3333333333
```

In [10]:

```
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[10]:

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_Cl
0	imagedata04	Aluminum_1.TIF	Aluminum	(2048, 1536)	TIFF	RGB	726.178839	243.7795
1	imagedata04	Aluminum_2.TIF	Aluminum	(2048, 1536)	TIFF	RGB	747.250139	250.7746
2	imagedata04	Aluminum_3.TIF	Aluminum	(2048, 1536)	TIFF	RGB	647.951011	217.5692
3	imagedata04	Aluminum_4.TIF	Aluminum	(2048, 1536)	TIFF	RGB	645.600269	216.8501
4	imagedata04	Aluminum_5.TIF	Aluminum	(2048, 1536)	TIFF	RGB	714.698479	239.8194
5	imagedata04	Aluminum_6.TIF	Aluminum	(2048, 1536)	TIFF	RGB	750.254375	251.7174
6	imagedata04	Brass_1.TIF	Brass	(2048, 1536)	TIFF	RGB	709.859100	238.2461
7	imagedata04	Brass_2.TIF	Brass	(2048, 1536)	TIFF	RGB	573.145720	192.8063
8	imagedata04	Brass_3.TIF	Brass	(2048, 1536)	TIFF	RGB	701.531680	235.4887
9	imagedata04	CopperWire_1.TIF	CopperWire	(2048, 1536)	TIFF	RGB	713.487175	239.4449
10	imagedata04	CopperWire_2.TIF	CopperWire	(2048, 1536)	TIFF	RGB	702.836070	235.9157
11	imagedata04	CopperWire_3.TIF	CopperWire	(2048, 1536)	TIFF	RGB	676.261871	227.0989
12	imagedata04	CopperWire_4.TIF	CopperWire	(2048, 1536)	TIFF	RGB	659.115091	221.3873
13	imagedata04	Copper_1.TIF	Copper	(128, 96)	TIFF	RGB	2.754645	0.9245
14	imagedata04	Copper_2.TIF	Copper	(2048, 1536)	TIFF	RGB	598.281918	201.0351

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_Cl
15	imagedata04	Copper_3.TIF	Copper	(2048, 1536)	TIFF	RGB	684.068746	229.6650
16	imagedata04	Copper_4.TIF	Copper	(2048, 1536)	TIFF	RGB	699.508259	234.8412
17	imagedata04	Copper_5.TIF	Copper	(2048, 1536)	TIFF	RGB	700.652871	235.2333
18	imagedata04	Copper_6.TIF	Copper	(2048, 1536)	TIFF	RGB	701.130022	235.3750
19	imagedata04	Iron_1.TIF	Iron	(2048, 1536)	TIFF	RGB	703.126571	236.0493
20	imagedata04	Iron_2.TIF	Iron	(2048, 1536)	TIFF	RGB	644.602235	216.4610
21	imagedata04	Iron_3.TIF	Iron	(2048, 1536)	TIFF	RGB	730.043419	244.9721
22	imagedata04	Iron_4.TIF	Iron	(2048, 1536)	TIFF	RGB	782.352207	262.3887
23	imagedata04	Iron_5.TIF	Iron	(2048, 1536)	TIFF	RGB	762.796121	255.8943
24	imagedata04	PaintedIron_1.TIF	PaintedIron	(2048, 1536)	TIFF	RGB	628.004650	211.0393
25	imagedata04	PaintedIron_2.TIF	PaintedIron	(2048, 1536)	TIFF	RGB	671.798350	225.6760
26	imagedata04	StainlessSteel_1.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	677.696081	227.5618
27	imagedata04	StainlessSteel_2.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	691.749295	232.1923
28	imagedata04	StainlessSteel_3.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	679.291137	228.1871
29	imagedata04	StainlessSteel_4.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	639.179701	214.7307

In [11]:

```
# 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 [12]:

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

Out[12]:

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extrems
0	Aluminum	726.178839	243.779562	243.779562	238.619715	254.000000	127.833333
1	Aluminum	747.250139	250.774651	250.774651	245.700837	254.000000	127.833333
2	Aluminum	647.951011	217.569200	217.569200	212.812611	254.333333	127.833333
3	Aluminum	645.600269	216.850148	216.850148	211.899973	254.000000	127.833333
4	Aluminum	714.698479	239.819486	239.819486	235.059507	254.333333	127.833333
5	Aluminum	750.254375	251.717483	251.717483	246.819409	254.000000	127.833333
6	Brass	709.859100	238.246129	238.246129	233.366842	254.000000	127.833333
7	Brass	573.145720	192.806339	192.806339	187.533042	254.000000	127.833333
8	Brass	701.531680	235.488771	235.488771	230.554138	254.333333	127.833333
9	CopperWire	713.487175	239.444931	239.444931	234.597313	254.333333	127.833333
10	CopperWire	702.836070	235.915798	235.915798	231.004474	254.333333	127.833333
11	CopperWire	676.261871	227.098992	227.098992	222.063887	254.000000	127.833333
12	CopperWire	659.115091	221.387333	221.387333	216.340425	254.000000	127.833333
13	Copper	2.754645	0.924552	0.924552	0.905541	254.000000	127.500000
14	Copper	598.281918	201.035122	201.035122	196.211674	254.000000	127.833333
15	Copper	684.068746	229.665048	229.665048	224.738650	254.000000	127.833333
16	Copper	699.508259	234.841215	234.841215	229.825829	254.000000	127.833333
17	Copper	700.652871	235.233367	235.233367	230.186137	254.000000	127.833333
18	Copper	701.130022	235.375054	235.375054	230.379914	254.000000	127.833333
19	Iron	703.126571	236.049335	236.049335	231.027901	254.000000	127.833333
20	Iron	644.602235	216.461098	216.461098	211.680039	254.000000	127.833333
21	Iron	730.043419	244.972145	244.972145	240.099129	254.333333	127.833333
22	Iron	782.352207	262.388791	262.388791	257.574625	254.000000	127.833333
23	Iron	762.796121	255.894397	255.894397	251.007327	254.333333	127.833333
24	PaintedIron	628.004650	211.039372	211.039372	205.925906	254.000000	127.833333
25	PaintedIron	671.798350	225.676073	225.676073	220.446204	254.000000	127.833333
26	StainlessSteel	677.696081	227.561825	227.561825	222.572431	254.000000	127.833333
27	StainlessSteel	691.749295	232.192312	232.192312	227.364671	254.333333	127.833333
28	StainlessSteel	679.291137	228.187119	228.187119	222.916899	254.000000	127.833333
29	StainlessSteel	639.179701	214.730790	214.730790	209.718121	254.000000	127.833333

In [13]:

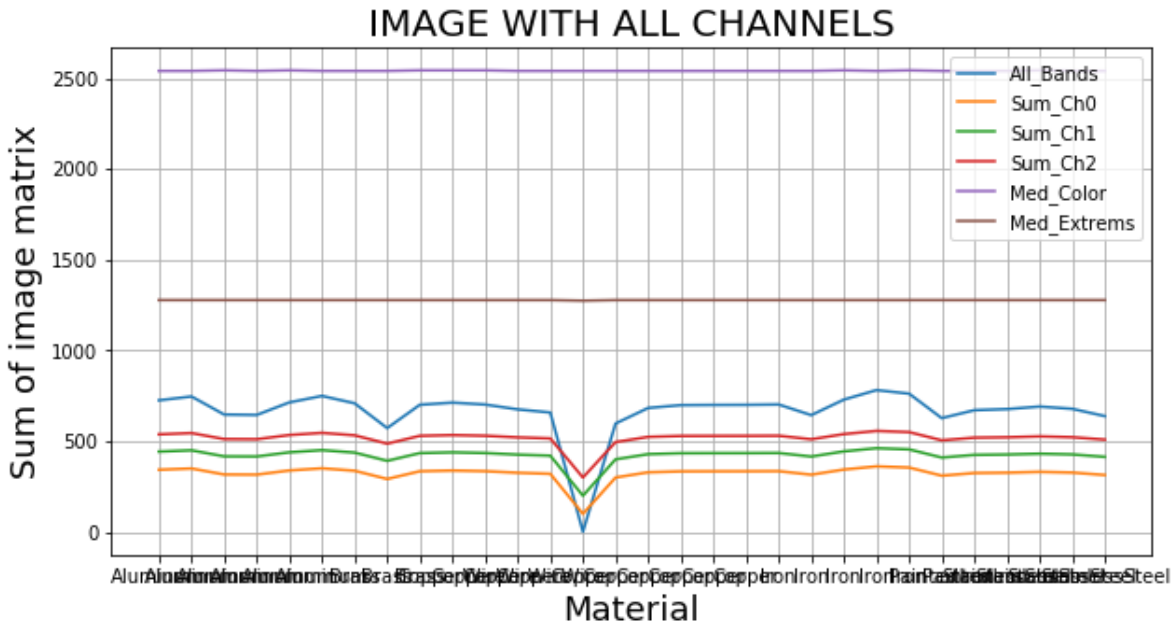
```
df_plot.Sum_Ch0 = df_plot.Sum_Ch0 + 100 # to have diference lines during plot
df_plot.Sum_Ch1 = df_plot.Sum_Ch1 + 200
df_plot.Sum_Ch2 = df_plot.Sum_Ch2 + 300
df_plot.Med_Color  = df_plot.Med_Color * 10
df_plot.Med_Extrems  = df_plot.Med_Extrems * 10
df_plot
```

Out[13]:

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extrems
0	Aluminum	726.178839	343.779562	443.779562	538.619715	2540.000000	1278.333333
1	Aluminum	747.250139	350.774651	450.774651	545.700837	2540.000000	1278.333333
2	Aluminum	647.951011	317.569200	417.569200	512.812611	2543.333333	1278.333333
3	Aluminum	645.600269	316.850148	416.850148	511.899973	2540.000000	1278.333333
4	Aluminum	714.698479	339.819486	439.819486	535.059507	2543.333333	1278.333333
5	Aluminum	750.254375	351.717483	451.717483	546.819409	2540.000000	1278.333333
6	Brass	709.859100	338.246129	438.246129	533.366842	2540.000000	1278.333333
7	Brass	573.145720	292.806339	392.806339	487.533042	2540.000000	1278.333333
8	Brass	701.531680	335.488771	435.488771	530.554138	2543.333333	1278.333333
9	CopperWire	713.487175	339.444931	439.444931	534.597313	2543.333333	1278.333333
10	CopperWire	702.836070	335.915798	435.915798	531.004474	2543.333333	1278.333333
11	CopperWire	676.261871	327.098992	427.098992	522.063887	2540.000000	1278.333333
12	CopperWire	659.115091	321.387333	421.387333	516.340425	2540.000000	1278.333333
13	Copper	2.754645	100.924552	200.924552	300.905541	2540.000000	1275.000000
14	Copper	598.281918	301.035122	401.035122	496.211674	2540.000000	1278.333333
15	Copper	684.068746	329.665048	429.665048	524.738650	2540.000000	1278.333333
16	Copper	699.508259	334.841215	434.841215	529.825829	2540.000000	1278.333333
17	Copper	700.652871	335.233367	435.233367	530.186137	2540.000000	1278.333333
18	Copper	701.130022	335.375054	435.375054	530.379914	2540.000000	1278.333333
19	Iron	703.126571	336.049335	436.049335	531.027901	2540.000000	1278.333333
20	Iron	644.602235	316.461098	416.461098	511.680039	2540.000000	1278.333333
21	Iron	730.043419	344.972145	444.972145	540.099129	2543.333333	1278.333333
22	Iron	782.352207	362.388791	462.388791	557.574625	2540.000000	1278.333333
23	Iron	762.796121	355.894397	455.894397	551.007327	2543.333333	1278.333333
24	PaintedIron	628.004650	311.039372	411.039372	505.925906	2540.000000	1278.333333
25	PaintedIron	671.798350	325.676073	425.676073	520.446204	2540.000000	1278.333333
26	StainlessSteel	677.696081	327.561825	427.561825	522.572431	2540.000000	1278.333333
27	StainlessSteel	691.749295	332.192312	432.192312	527.364671	2543.333333	1278.333333
28	StainlessSteel	679.291137	328.187119	428.187119	522.916899	2540.000000	1278.333333
29	StainlessSteel	639.179701	314.730790	414.730790	509.718121	2540.000000	1278.333333

In [14]:

```
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 [15]:

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

Out[15]:

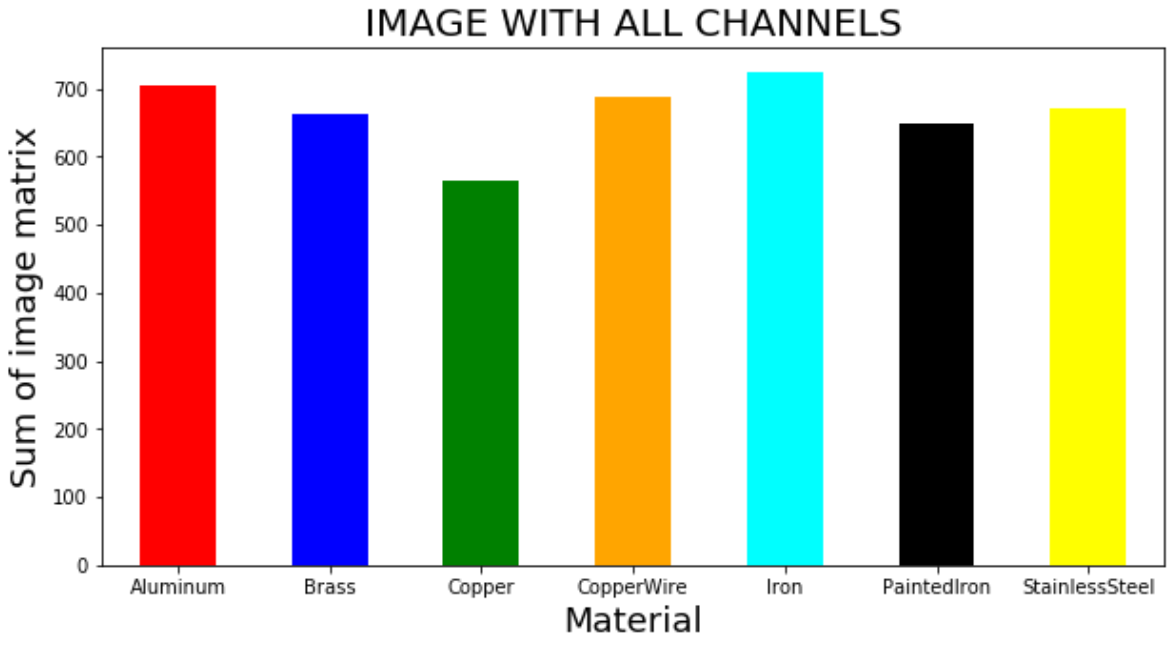
Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extrems
Aluminum	705.322185	336.751755	436.751755	531.818675	2541.111111	1278.333333
Brass	661.512167	322.180413	422.180413	517.151341	2541.111111	1278.333333
Copper	564.399410	289.512393	389.512393	485.374624	2540.000000	1277.777778
CopperWire	687.925052	330.961763	430.961763	526.001525	2541.666667	1278.333333
Iron	724.584111	343.153153	443.153153	538.277804	2541.333333	1278.333333
PaintedIron	649.901500	318.357722	418.357722	513.186055	2540.000000	1278.333333
StainlessSteel	671.979054	325.668012	425.668012	520.643031	2540.833333	1278.333333

In [16]:

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

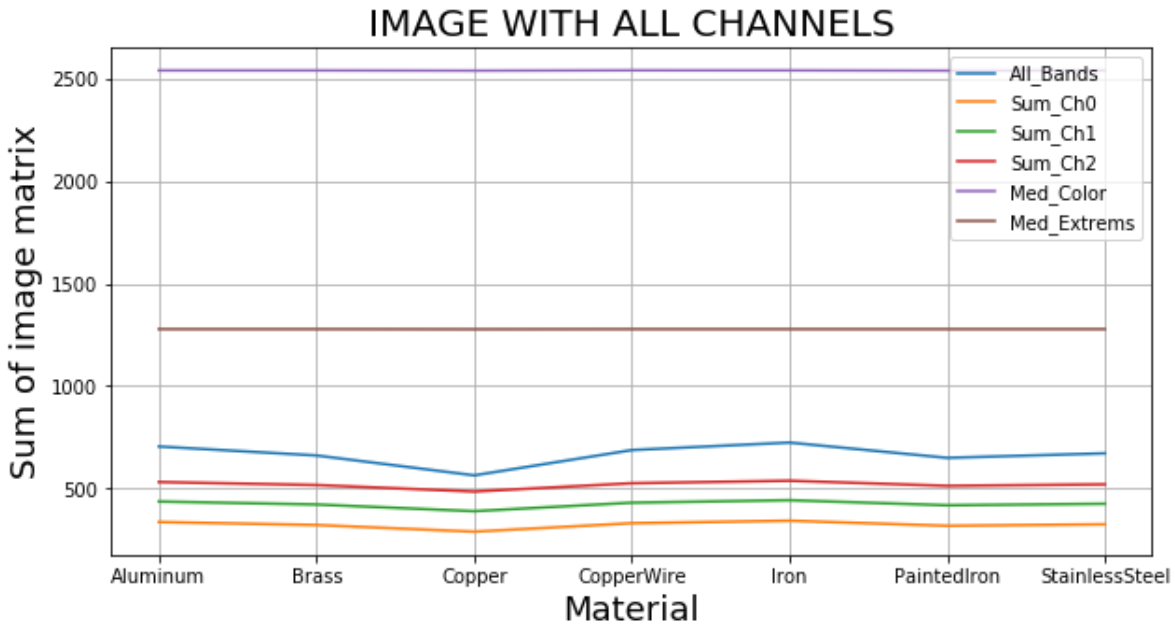
In [17]:

```
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 [18]:

```
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 [19]:

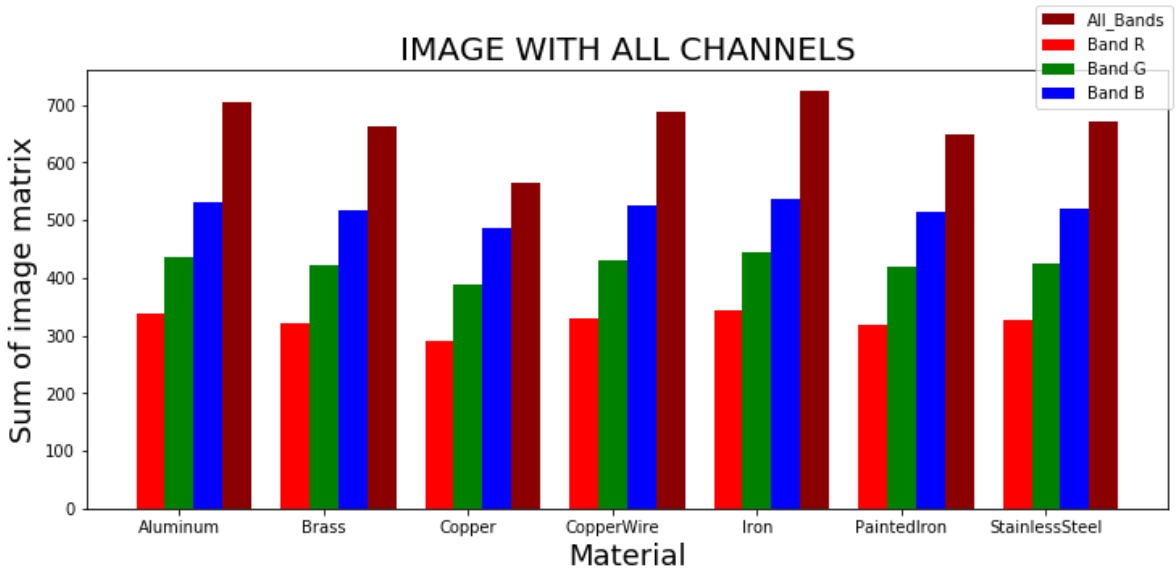
```
loc_Array_sum = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_r = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.3 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.5 # Offsetting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

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

In [ ]:

```

```

In [20]:

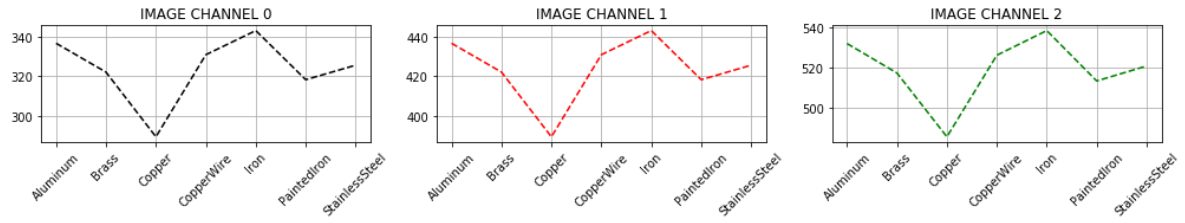
```
#Plot  Bar Graph
#df_plot1.plot(kind='bar', figsize=(12,5), grid=True, color='darkred',fontsize=18)
loc_Array_sum = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.3 # 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 [21]:

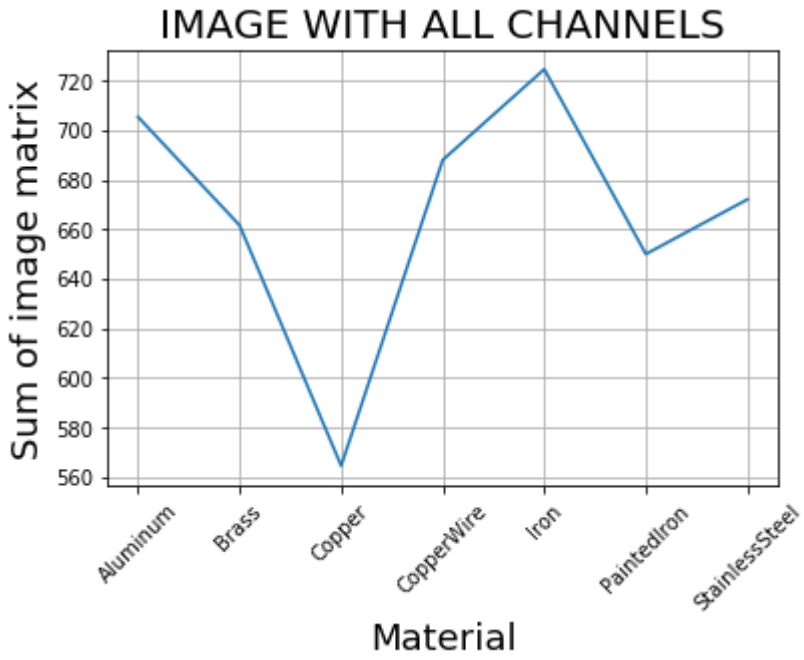
```
plt.figure(1)
plt.figure(figsize=(17, 4))
plt.tight_layout()
plt.subplot(231)
plt.title('IMAGE CHANNEL 0')
plt.xticks(rotation=45)
plt.grid(True)
plt.plot(df_plot1.Sum_Ch0, 'k--')

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>
```



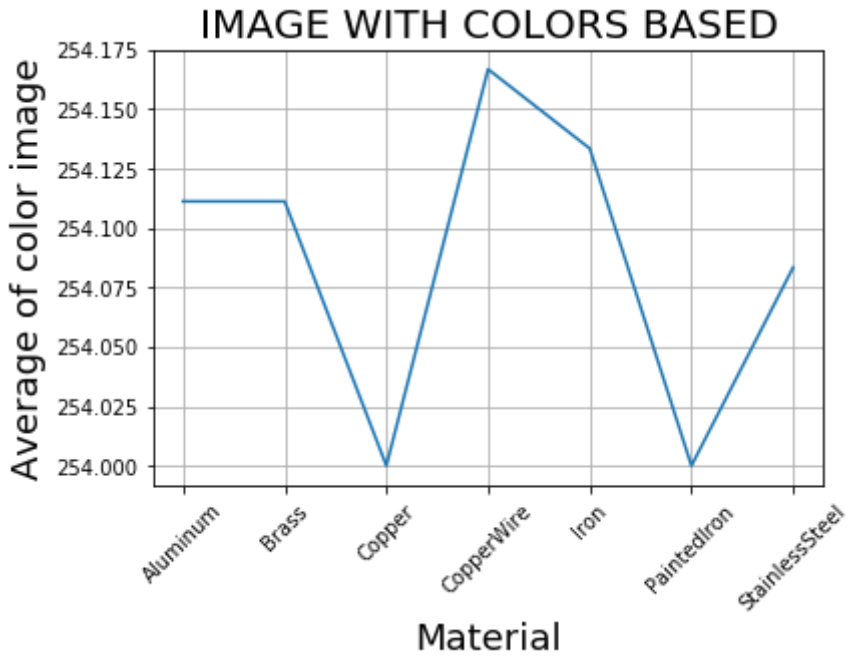
In [22]:

```
# 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()
```



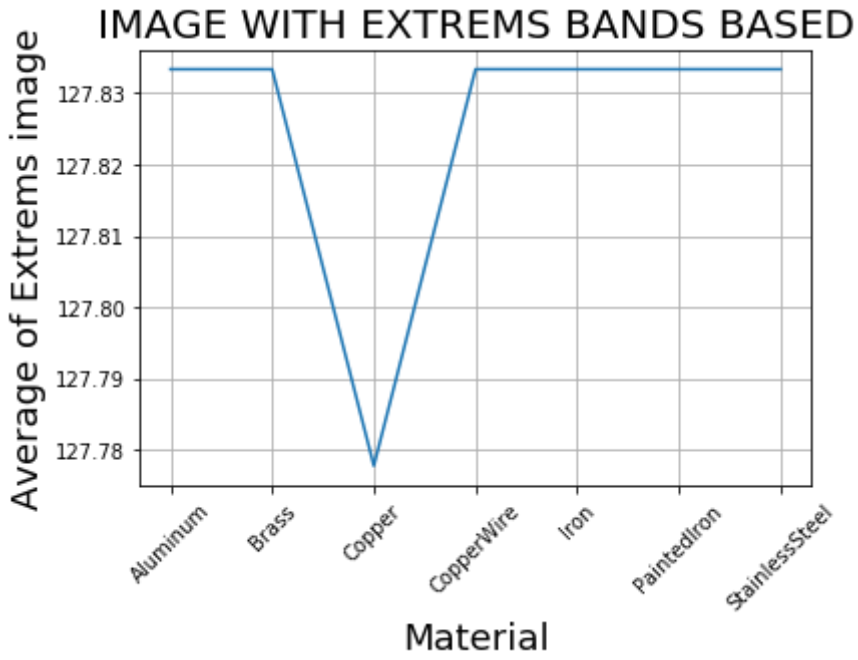
In [25]:

```
# 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 [26]:

```
# 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 [ ]: