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 = "imagedata02"
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/imagedata02/
 ---- IMAGES ON THE FOLDER -------
Aluminum_1.TIF
Aluminum_2.TIF
Aluminum_3.TIF
Brass_1.TIF
Brass_2.TIF
Brass_3.TIF
Brass_4.TIF
Brass_5.TIF
CooperWire_1.TIF
CooperWire_2.TIF
Copper_1.TIF
Copper_2.TIF
Copper_3.TIF
Iron_1.TIF
Iron_2.TIF
Iron_3.TIF
PaintedIron_1.TIF
StainlessSteel_1.TIF
StainlessSteel_2.TIF
StainlessSteel_3.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)
```

```
          Sum array: 751.938786  Sum Ch 0: 252.363792  Sum Ch 1: 252.36379
2  Sum Ch 2: 247.211202
          Histog   : 94.37184   N.List elem: 768
          Color    : (255, 255, 253) Med Color   : 254.33333333333334
          Extremes : ((1, 255), (1, 255), (0, 255)) Med Extremes: 127.8333
3333333333
------------------------------------------------------------------------
Inf.File: StainlessSteel_3.TIF
Get Channel n:  0
Get Channel n:  1
Get Channel n:  2
Get main Color file: StainlessSteel_3.TIF
StainlessSteel  Size: (2048, 1536)  Format: TIFF  Mode: RGB
          Sum array: 869.359917  Sum Ch 0: 291.41172  Sum Ch 1: 291.41172
Sum Ch 2: 286.536477
          Histog   : 94.37184   N.List elem: 768
          Color    : (255, 255, 253) Med Color   : 254.33333333333334
          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_(
0	imagedata02	Aluminum_1.TIF	Aluminum	(128, 96)	TIFF	RGB	2.917186	0.978
1	imagedata02	Aluminum_2.TIF	Aluminum	(2048, 1536)	TIFF	RGB	745.115335	250.209
2	imagedata02	Aluminum_3.TIF	Aluminum	(2048, 1536)	TIFF	RGB	844.400463	283.239
3	imagedata02	Brass_1.TIF	Brass	(2048, 1536)	TIFF	RGB	971.572582	325.441
4	imagedata02	Brass_2.TIF	Brass	(2048, 1536)	TIFF	RGB	813.656218	273.034
5	imagedata02	Brass_3.TIF	Brass	(2048, 1536)	TIFF	RGB	980.525819	328.435
6	imagedata02	Brass_4.TIF	Brass	(2048, 1536)	TIFF	RGB	964.217954	323.060
7	imagedata02	Brass_5.TIF	Brass	(2048, 1536)	TIFF	RGB	1006.157746	336.971
8	imagedata02	CooperWire_1.TIF	CooperWire	(2048, 1536)	TIFF	RGB	820.844704	275.483
9	imagedata02	CooperWire_2.TIF	CooperWire	(2048, 1536)	TIFF	RGB	761.900052	255.799
10	imagedata02	Copper_1.TIF	Copper	(2048, 1536)	TIFF	RGB	953.308521	319.368
11	imagedata02	Copper_2.TIF	Copper	(2048, 1536)	TIFF	RGB	952.844305	319.299
12	imagedata02	Copper_3.TIF	Copper	(2048, 1536)	TIFF	RGB	900.484212	301.924
13	imagedata02	Iron_1.TIF	Iron	(2048, 1536)	TIFF	RGB	845.280229	283.537
14	imagedata02	Iron_2.TIF	Iron	(2048, 1536)	TIFF	RGB	868.518814	291.102

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_(
15	imagedata02	Iron_3.TIF	Iron	(2048, 1536)	TIFF	RGB	946.388117	317.197
16	imagedata02	PaintedIron_1.TIF	PaintedIron	(2048, 1536)	TIFF	RGB	985.104052	329.969
17	imagedata02	StainlessSteel_1.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	908.287593	304.362
18	imagedata02	StainlessSteel_2.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	751.938786	252.363
19	imagedata02	StainlessSteel_3.TIF	StainlessSteel	(2048, 1536)	TIFF	RGB	869.359917	291.411

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	2.917186	0.978783	0.978783	0.959620	13.666667	128.333333
1	Aluminum	745.115335	250.209290	250.209290	244.696755	254.000000	127.833333
2	Aluminum	844.400463	283.239215	283.239215	277.922033	254.000000	127.833333
3	Brass	971.572582	325.441794	325.441794	320.688994	254.333333	127.833333
4	Brass	813.656218	273.034135	273.034135	267.587948	254.000000	127.833333
5	Brass	980.525819	328.435929	328.435929	323.653961	254.333333	127.833333
6	Brass	964.217954	323.060043	323.060043	318.097868	254.000000	127.833333
7	Brass	1006.157746	336.971231	336.971231	332.215284	254.333333	127.833333
8	CooperWire	820.844704	275.483117	275.483117	269.878470	254.000000	127.833333
9	CooperWire	761.900052	255.799359	255.799359	250.301334	254.000000	127.833333
10	Copper	953.308521	319.368378	319.368378	314.571765	254.333333	127.833333
11	Copper	952.844305	319.299286	319.299286	314.245733	254.000000	127.833333
12	Copper	900.484212	301.924460	301.924460	296.635292	254.000000	127.833333
13	Iron	845.280229	283.537171	283.537171	278.205887	254.000000	127.833333
14	Iron	868.518814	291.102653	291.102653	286.313508	254.333333	127.833333
15	Iron	946.388117	317.197385	317.197385	311.993347	254.000000	127.833333
16	PaintedIron	985.104052	329.969829	329.969829	325.164394	254.333333	127.833333
17	StainlessSteel	908.287593	304.362145	304.362145	299.563303	254.333333	127.833333
18	StainlessSteel	751.938786	252.363792	252.363792	247.211202	254.333333	127.833333
19	StainlessSteel	869.359917	291.411720	291.411720	286.536477	254.333333	127.833333

In [13]:

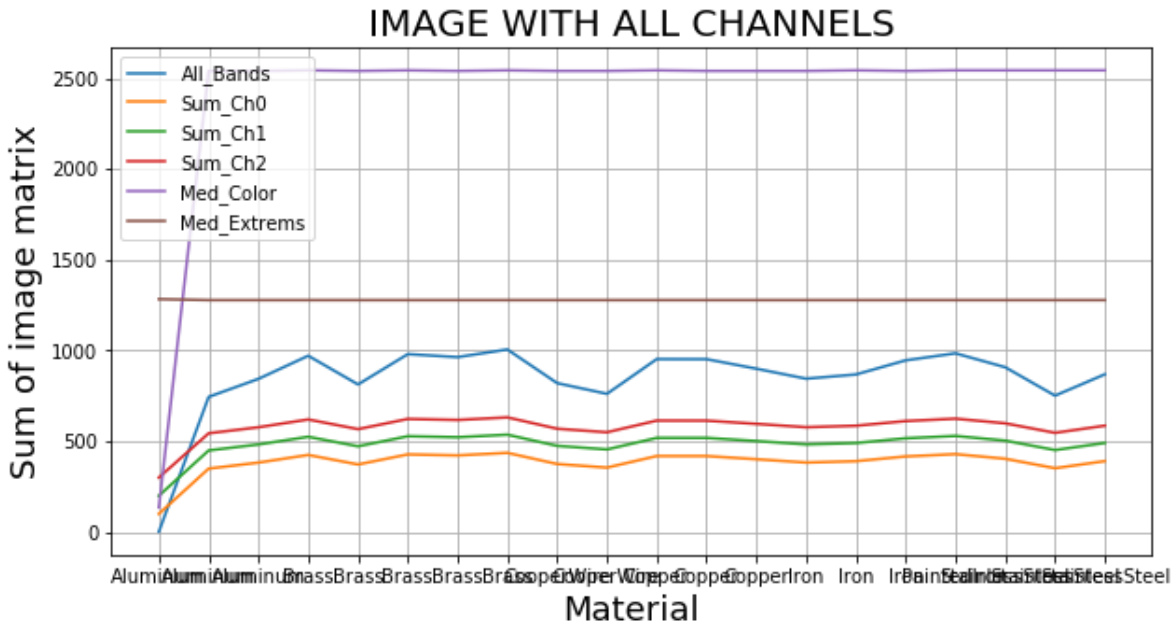
```
df_plot.Sum_Ch0 = df_plot.Sum_Ch0 + 100 # to have diference lines during plot
df_plot.Sum_Ch1 = df_plot.Sum_Ch1 + 200
df_plot.Sum_Ch2 = df_plot.Sum_Ch2 + 300
df_plot.Med_Color  = df_plot.Med_Color * 10
df_plot.Med_Extrems  = df_plot.Med_Extrems * 10
df_plot
```

Out[13]:

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extrems
0	Aluminum	2.917186	100.978783	200.978783	300.959620	136.666667	1283.333333
1	Aluminum	745.115335	350.209290	450.209290	544.696755	2540.000000	1278.333333
2	Aluminum	844.400463	383.239215	483.239215	577.922033	2540.000000	1278.333333
3	Brass	971.572582	425.441794	525.441794	620.688994	2543.333333	1278.333333
4	Brass	813.656218	373.034135	473.034135	567.587948	2540.000000	1278.333333
5	Brass	980.525819	428.435929	528.435929	623.653961	2543.333333	1278.333333
6	Brass	964.217954	423.060043	523.060043	618.097868	2540.000000	1278.333333
7	Brass	1006.157746	436.971231	536.971231	632.215284	2543.333333	1278.333333
8	CooperWire	820.844704	375.483117	475.483117	569.878470	2540.000000	1278.333333
9	CooperWire	761.900052	355.799359	455.799359	550.301334	2540.000000	1278.333333
10	Copper	953.308521	419.368378	519.368378	614.571765	2543.333333	1278.333333
11	Copper	952.844305	419.299286	519.299286	614.245733	2540.000000	1278.333333
12	Copper	900.484212	401.924460	501.924460	596.635292	2540.000000	1278.333333
13	Iron	845.280229	383.537171	483.537171	578.205887	2540.000000	1278.333333
14	Iron	868.518814	391.102653	491.102653	586.313508	2543.333333	1278.333333
15	Iron	946.388117	417.197385	517.197385	611.993347	2540.000000	1278.333333
16	PaintedIron	985.104052	429.969829	529.969829	625.164394	2543.333333	1278.333333
17	StainlessSteel	908.287593	404.362145	504.362145	599.563303	2543.333333	1278.333333
18	StainlessSteel	751.938786	352.363792	452.363792	547.211202	2543.333333	1278.333333
19	StainlessSteel	869.359917	391.411720	491.411720	586.536477	2543.333333	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]:

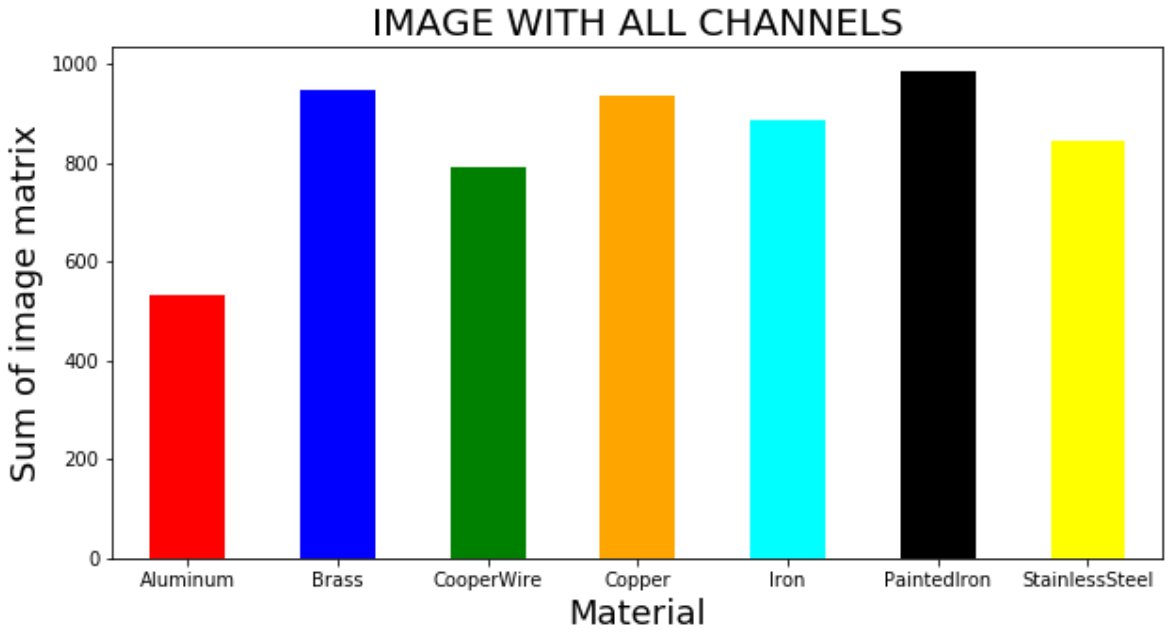
Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Med_Color	Med_Extrems
Aluminum	530.810995	278.142429	378.142429	474.526136	1738.888889	1280.000000
Brass	947.226064	417.388626	517.388626	612.448811	2542.000000	1278.333333
CooperWire	791.372378	365.641238	465.641238	560.089902	2540.000000	1278.333333
Copper	935.545679	413.530708	513.530708	608.484263	2541.111111	1278.333333
Iron	886.729053	397.279070	497.279070	592.170914	2541.111111	1278.333333
PaintedIron	985.104052	429.969829	529.969829	625.164394	2543.333333	1278.333333
StainlessSteel	843.195432	382.712552	482.712552	577.770327	2543.333333	1278.333333

In [16]:

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

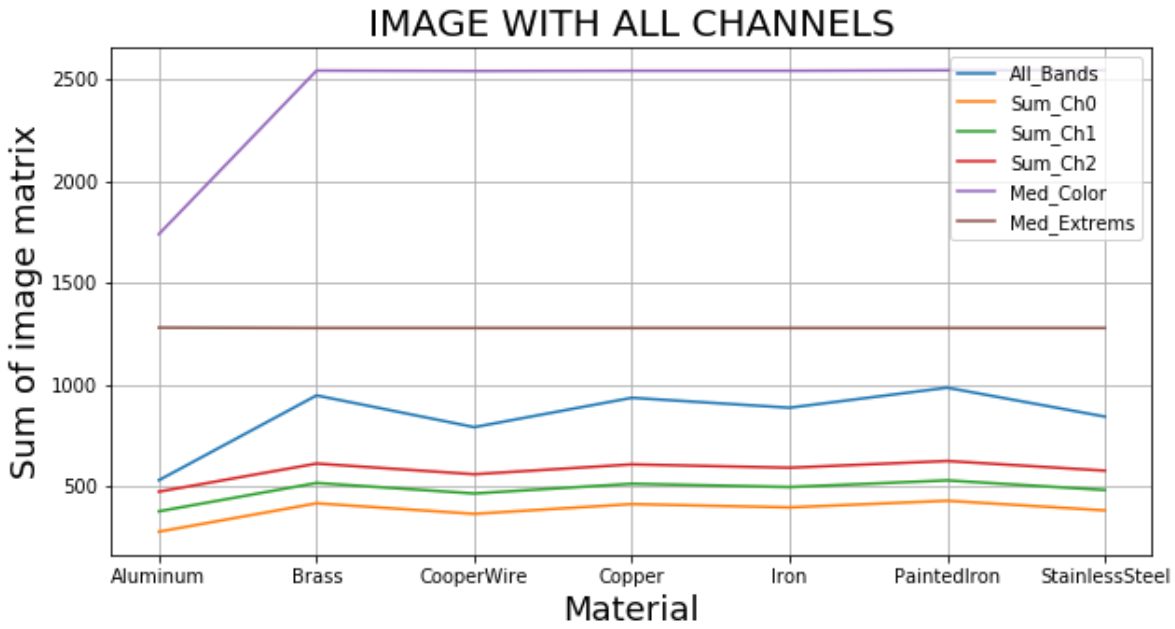
In [17]:

```
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]:

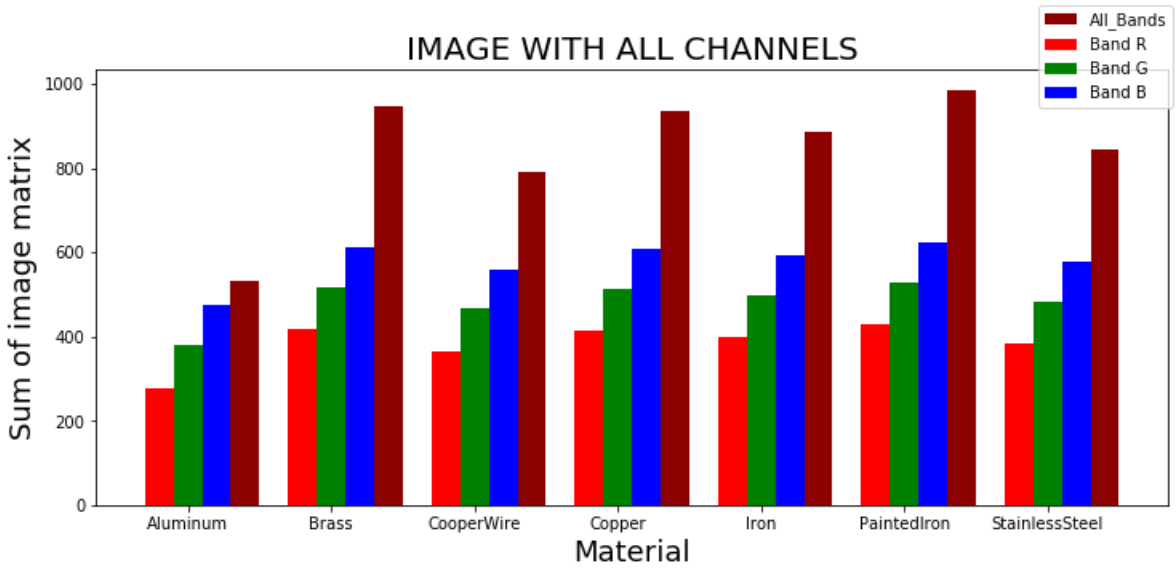
```
loc_Array_sum = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_r = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.3 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.5 # Offsetting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

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

In [ ]:

```

```

In [20]:

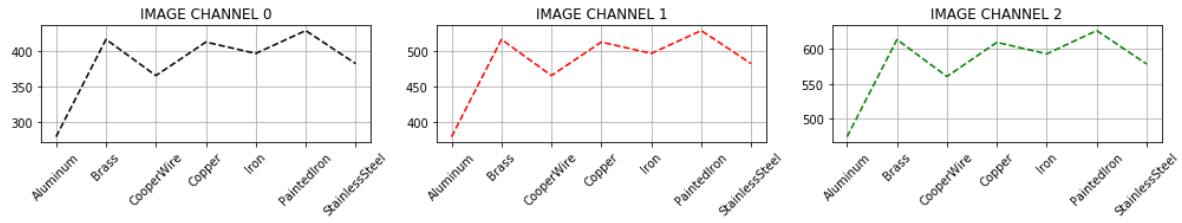
```
#Plot  Bar Graph
#df_plot1.plot(kind='bar', figsize=(12,5), grid=True, color='darkred',fontsize=18)
loc_Array_sum = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.3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

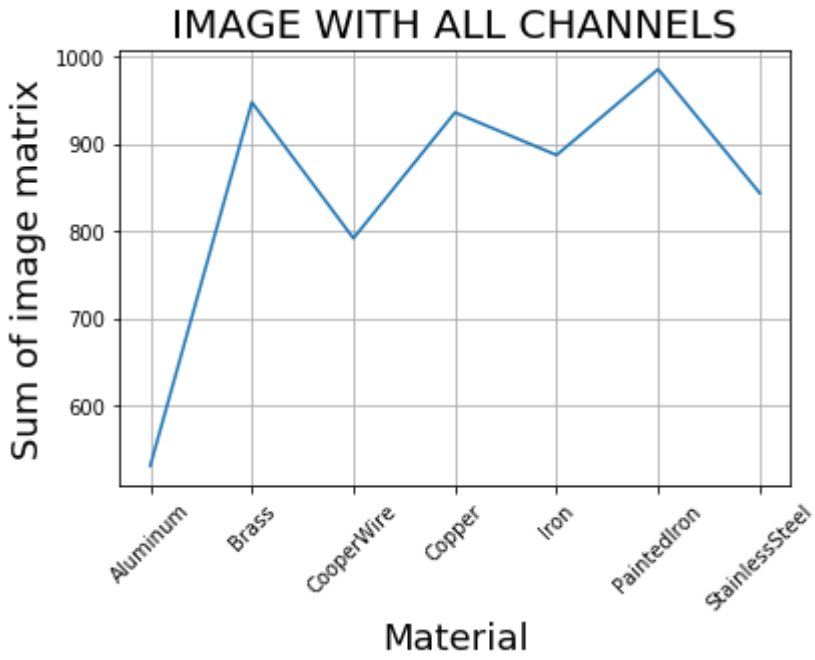
```
plt.figure(1)
plt.figure(figsize=(17, 4))
plt.tight_layout()
plt.subplot(231)
plt.title('IMAGE CHANNEL 0')
plt.xticks(rotation=45)
plt.grid(True)
plt.plot(df_plot1.Sum_Ch0, 'k--')

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>
```



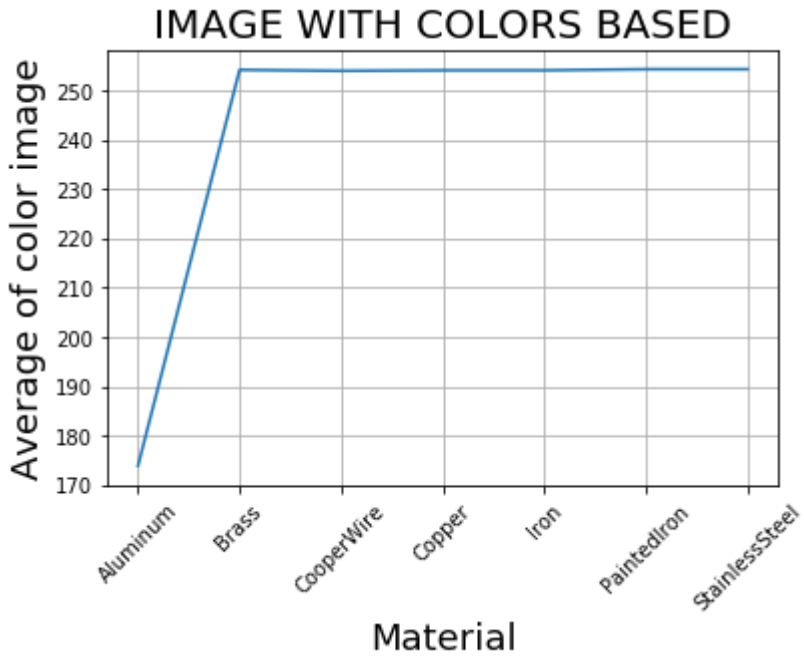
In [22]:

```
# 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()
```



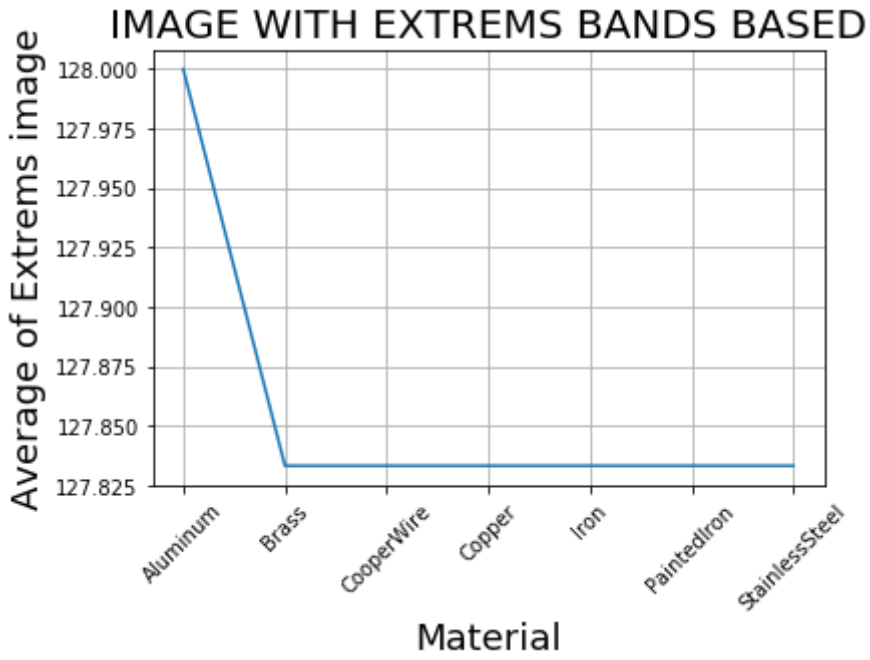
In [23]:

```
# 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 [24]:

```
# 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 [ ]: