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)

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 - 11-05-2019 Version 8

Add Libraries

In [1]:

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

In [2]:

```
# Clear all
os.system( 'cls' )

# 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 = "imagedata06"

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

# Upper End File
#end_file = end_file.upper()
path = currDir + "/" + folder + "/"
end_file
```

Out[3]:

```
'.tif'
```

Read images from folder

In [4]:

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

list_of_images = list()  # save all images on folder for further processing

for file in os.listdir(path):
    if file.endswith(end_file):
        print(os.path.join(file))
        list_of_images.append(file)   # save all images on folder for further processing
print('-')
```

```
C:\Users\manuel.robalinho\Google Drive\UPT_Portucalense\Trabalho final\Class
ificacao_Sucata\Jupyter_Notebook/imagedata06/
-
 ---- IMAGES ON THE FOLDER : imagedata06 ------- * .tif
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
Brass_4.tif
Brass_5.tif
Brass_6.tif
CopperWire_1.tif
CopperWire_2.tif
CopperWire_3.tif
CopperWire_4.tif
CopperWire_5.tif
CopperWire_6.tif
CopperWire_7.tif
CopperWire_8.tif
Copper_1.tif
Copper_2.tif
Copper_3.tif
Copper_4.tif
Iron_1.tif
Iron_2.tif
Iron_3.tif
Iron_4.tif
PaintedIron_1.tif
PaintedIron_2.tif
PaintedIron_3.tif
PaintedIron_4.tif
PaintedIron_5.tif
PaintedIron_6.tif
PaintedIron_7.tif
StainlessSteel_1.tif
StainlessSteel_2.tif
StainlessSteel_3.tif
StainlessSteel_4.tif
StainlessSteel_5.tif
StainlessSteel_6.tif
```

```
StainlessSteel_7.tif
StainlessSteel_8.tif
StainlessSteel_9.tif
-
```

In [5]:

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

Functions to the work

In [6]:

```
# Read image with PIL
from PIL import Image, ImageFilter, ImageOps
def read_pil_image(file1):
    #print('Reading PIL image:', file1)
    try:
        im_pil = Image.open(file1)
    except:
        print("-->Unable to load image",file1)
    return im_pil
```

In [7]:

```
# Read image with OPENCV
import cv2
def read_cv2_image(file1):
    #print('Reading CV image:',file1)
    try:
        im_cv = cv2.imread(file1)
    except:
        print("-->Unable to load image",file1)
    return im_cv
```

In [8]:

```
# 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 [9]:

```
# 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)
    file1 = path+file
    # Read image
    img = read_pil_image(file1)
    if img == None:
        print("-->Unable to load image",file1)

    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 [10]:

```
#!/usr/bin/python

# Return one 24-bit color value
def rgbToDecimal(x_rgb):
    r,g,b = rgbToRGB(x_rgb)
    rgb_dec = (r << 16) + (g << 8) + b
    #print('RGB Color:', x_rgb, '   Dec:', rgb_dec)
    return rgb_dec

# Convert 24-bit color value to RGB
def colorToRGB(c):
    r = c >> 16
    c -= r * 65536;
    g = c / 256
    c -= g * 256;
    b = c
    return [r, g, b]

def rgbToRGB(x_rgb):
    x_rgb = list(x_rgb)
    r = x_rgb[0]
    g = x_rgb[1]
    b = x_rgb[2]

    #print('rgbToRGB:',x_rgb, r,g,b)
    return r, g, b

def getRGBfromI(RGBint):
    blue =  RGBint & 255
    green = (RGBint >> 8) & 255
    red =   (RGBint >> 16) & 255
    return red, green, blue

def getIfromRGB(rgb):
    red = rgb[0]
    green = rgb[1]
    blue = rgb[2]
    #print('getIfromRGB:', red, green, blue)
    RGBint = (red<<16) + (green<<8) + blue
    return RGBint

# RGB to Hex Decimal
def rgb_to_hex(rgb):
    rgb_int = bytes(rgb).hex()
    rgb_dec = '#'+str(rgb_int)
    #print('RGB :',rgb, '  Hex Dec:', rgb_dec)
    return rgb_dec

# Test
#x_rgb = (254, 250, 255)
#rgb_hex = rgb_to_hex(x_rgb)
#rgb_dec = rgbToDecimal(x_rgb)
```

In [11]:

```
# https://github.com/conda-forge/webcolors-feedstock
# conda config --add channels conda-forge
# conda install webcolors
# It is possible to list all of the versions of webcolors available on your platform with:
#       conda search webcolors --channel conda-forge

# COLOR NAME
import webcolors
def get_color_name(rgb_x):
    min_colours = {}
    for key, name in webcolors.css21_hex_to_names.items():
        r_c, g_c, b_c = webcolors.hex_to_rgb(key)
        rd = (r_c - rgb_x[0]) ** 2
        gd = (g_c - rgb_x[1]) ** 2
        bd = (b_c - rgb_x[2]) ** 2
        min_colours[(rd + gd + bd)] = name
    print('Color name from RGB:',rgb_x,'  is :',min_colours[min(min_colours.keys())])
    return min_colours[min(min_colours.keys())]
```

In [12]:

```
# Get color name from RGB
# https://stackoverflow.com/questions/2453344/find-the-colour-name-from-a-hexadecimal-colou

colorof = {'#F0F8FF':"aliceblue",
'#FAEBD7':"antiquewhite",
'#00FFFF':"aqua",
'#7FFFD4':"aquamarine",
'#F0FFFF':"azure",
'#F5F5DC':"beige",
'#FFE4C4':"bisque",
'#000000':"black",
'#FFEBCD':"blanchedalmond",
'#0000FF':"blue",
'#8A2BE2':"blueviolet",
'#A52A2A':"brown",
'#DEB887':"burlywood",
'#5F9EA0':"cadetblue",
'#7FFF00':"chartreuse",
'#D2691E':"chocolate",
'#FF7F50':"coral",
'#6495ED':"cornflowerblue",
'#FFF8DC':"cornsilk",
'#DC143C':"crimson",
'#00FFFF':"cyan",
'#00008B':"darkblue",
'#008B8B':"darkcyan",
'#B8860B':"darkgoldenrod",
'#A9A9A9':"darkgray",
'#006400':"darkgreen",
'#BDB76B':"darkkhaki",
'#8B008B':"darkmagenta",
'#556B2F':"darkolivegreen",
'#FF8C00':"darkorange",
'#9932CC':"darkorchid",
'#8B0000':"darkred",
'#E9967A':"darksalmon",
'#8FBC8B':"darkseagreen",
'#483D8B':"darkslateblue",
'#2F4F4F':"darkslategray",
'#00CED1':"darkturquoise",
'#9400D3':"darkviolet",
'#FF1493':"deeppink",
'#00BFFF':"deepskyblue",
'#696969':"dimgray",
'#1E90FF':"dodgerblue",
'#B22222':"firebrick",
'#FFFAF0':"floralwhite",
'#228B22':"forestgreen",
'#FF00FF':"fuchsia",
'#DCDCDC':"gainsboro",
'#F8F8FF':"ghostwhite",
'#FFD700':"gold",
'#DAA520':"goldenrod",
'#808080':"gray",
'#008000':"green",
'#ADFF2F':"greenyellow",
'#F0FFF0':"honeydew",
'#FF69B4':"hotpink",
'#CD5C5C':"indianred",
```

```
'#4B0082':"indigo",
'#FFFFF0':"ivory",
'#F0E68C':"khaki",
'#E6E6FA':"lavender",
'#FFF0F5':"lavenderblush",
'#7CFC00':"lawngreen",
'#FFFACD':"lemonchiffon",
'#ADD8E6':"lightblue",
'#F08080':"lightcoral",
'#E0FFFF':"lightcyan",
'#FAFAD2':"lightgoldenrodyellow",
'#D3D3D3':"lightgray",
'#90EE90':"lightgreen",
'#FFB6C1':"lightpink",
'#FFA07A':"lightsalmon",
'#20B2AA':"lightseagreen",
'#87CEFA':"lightskyblue",
'#778899':"lightslategray",
'#B0C4DE':"lightsteelblue",
'#FFFFE0':"lightyellow",
'#00FF00':"lime",
'#32CD32':"limegreen",
'#FAF0E6':"linen",
'#FF00FF':"magenta",
'#800000':"maroon",
'#66CDAA':"mediumaquamarine",
'#0000CD':"mediumblue",
'#BA55D3':"mediumorchid",
'#9370DB':"mediumpurple",
'#3CB371':"mediumseagreen",
'#7B68EE':"mediumslateblue",
'#00FA9A':"mediumspringgreen",
'#48D1CC':"mediumturquoise",
'#C71585':"mediumvioletred",
'#191970':"midnightblue",
'#F5FFFA':"mintcream",
'#FFE4E1':"mistyrose",
'#FFE4B5':"moccasin",
'#FFDEAD':"navajowhite",
'#000080':"navy",
'#FDF5E6':"oldlace",
'#808000':"olive",
'#6B8E23':"olivedrab",
'#FFA500':"orange",
'#FF4500':"orangered",
'#DA70D6':"orchid",
'#EEE8AA':"palegoldenrod",
'#98FB98':"palegreen",
'#AFEEEE':"paleturquoise",
'#DB7093':"palevioletred",
'#FFEFD5':"papayawhip",
'#FFDAB9':"peachpuff",
'#CD853F':"peru",
'#FFC0CB':"pink",
'#DDA0DD':"plum",
'#B0E0E6':"powderblue",
'#800080':"purple",
'#FF0000':"red",
'#BC8F8F':"rosybrown",
'#4169E1':"royalblue",
'#8B4513':"saddlebrown",
```

```
'#FA8072':"salmon",
'#F4A460':"sandybrown",
'#2E8B57':"seagreen",
'#FFF5EE':"seashell",
'#A0522D':"sienna",
'#C0C0C0':"silver",
'#87CEEB':"skyblue",
'#6A5ACD':"slateblue",
'#708090':"slategray",
'#FFFAFA':"snow",
'#00FF7F':"springgreen",
'#4682B4':"steelblue",
'#D2B48C':"tan",
'#008080':"teal",
'#D8BFD8':"thistle",
'#FF6347':"tomato",
'#40E0D0':"turquoise",
'#EE82EE':"violet",
'#F5DEB3':"wheat",
'#FFFFFF':"white",
'#F5F5F5':"whitesmoke",
'#FFFF00':"yellow",
'#9ACD32':"yellowgreen"}


def get_rgb_color_name(rgb):

    hex_from_rgb = rgb_to_hex(rgb)  # transform RGB into hexadecimal
    hx = hex_from_rgb[1:8]
    #print(hx)
    # if color is found in dict
    if colorof.get(hx):return colorof[hx]

    # else return its closest available color
    m = 16777215
    k = '000000'
    for key in colorof.keys():
        key_color = key[1:8]
        #print(key_color)
        a = int(hx[:2],16)-int(key_color[:2],16)
        b = int(hx[2:4],16)-int(key_color[2:4],16)
        c = int(hx[4:],16)-int(key_color[4:],16)

        v = a*a+b*b+c*c # simple measure for distance between colors

        # v = (r1 - r2)^2 + (g1 - g2)^2 + (b1 - b2)^2

        if v <= m:
            m = v
            k = key

    return colorof[k], hex_from_rgb

# Test
#rgb_1 = (216, 220, 223)
#cname, hexdc = get_rgb_color_name(rgb_1)
#print('Found:',    cname, '  Hex:', hexdc)     # found in dict
```

In [13]:

```
# Increase the contrast image
# im - image
# xvalue = contrast value
# https://pillow.readthedocs.io/en/4.0.x/reference/ImageEnhance.html
from PIL import ImageEnhance
# Path + file name + numeric value to enhancement

def contrast(path, xfile, xvalue):
    print('   Enhance image:', xfile, '  Value:', xvalue)
    file1 = path + xfile
    # Read Image
    im = read_pil_image(file1)
    if im == None:
        print("-->Unable to load image",file1)

    enh = ImageEnhance.Contrast(im)
    # enh.enhance(1.0).show("30% more contrast")
    x_enh = enh.enhance(xvalue)
     # Create name file masked
    f2_file = 'Enh_' + xfile
    print('   Save enhanced file :', f2_file)
    x_enh.save(f2_file)  # save enhanced file
    return x_enh, f2_file
```

In [14]:

```
# Return RGB separately
def return_rgb_from_RGB(rgb):
    p_rgb = list(rgb)
    red   = p_rgb[0]
    green = p_rgb[1]
    blue  = p_rgb[2]
    return red, green, blue
```

In [15]:

```
# Return distance from 2 colors

# http://hanzratech.in/2015/01/16/color-difference-between-2-colors-using-python.html
# https://python-colormath.readthedocs.io/en/latest/delta_e.html#delta-e-cie-2000

from colormath.color_objects import sRGBColor, LabColor
from colormath.color_conversions import convert_color
from colormath.color_diff import delta_e_cie2000

def delta_2_colors(rgb_1, rgb_2):
    #print('   Delta colors: ', rgb_1, rgb_2)
    #---- first color
    xr, xg, xb = return_rgb_from_RGB(rgb_1)
    # Red Color
    color1_rgb = sRGBColor(xr, xg, xb)

    #--- other color
    rgb_1 = rgb_2
    xr, xg, xb = return_rgb_from_RGB(rgb_1)
    # Blue Color
    color2_rgb = sRGBColor(xr, xg, xb)

    # Convert from RGB to Lab Color Space
    color1_lab = convert_color(color1_rgb, LabColor)

    # Convert from RGB to Lab Color Space
    color2_lab = convert_color(color2_rgb, LabColor)

    # Find the color difference
    delta_e = delta_e_cie2000(color1_lab, color2_lab)

    #print("      The difference between the 2 color = ", delta_e)
    return delta_e
```

In [16]:

```
# Remove Background - Put red background
#https://stackoverflow.com/questions/29313667/how-do-i-remove-the-background-from-this-kind

import cv2
import numpy as np

def red_background(path, xfile):
    print('   Red background for image:', xfile)
    #== Parameters =======================================================================
    BLUR = 21
    CANNY_THRESH_1 = 10
    CANNY_THRESH_2 = 100
    MASK_DILATE_ITER = 10
    MASK_ERODE_ITER = 10
    MASK_COLOR = (0.0,0.0,1.0) # In BGR format

    #== Processing ========================================================================
    file1 = path + xfile
    #-- Read image -----------------------------------------------------------------------
    #img = cv2.imread(file1)
    # Read image
    img = read_cv2_image(file1)
    if img.any() == None:
        print("-->Unable to load image",file1)

    # Create GRAY Image
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    #-- Edge detection -------------------------------------------------------------------
    edges = cv2.Canny(gray, CANNY_THRESH_1, CANNY_THRESH_2)
    edges = cv2.dilate(edges, None)
    edges = cv2.erode(edges, None)

    #-- Find contours in edges, sort by area ---------------------------------------------
    contour_info = []
    _, contours, _ = cv2.findContours(edges, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
    for c in contours:
        contour_info.append((
            c,
            cv2.isContourConvex(c),
            cv2.contourArea(c),
        ))
    contour_info = sorted(contour_info, key=lambda c: c[2], reverse=True)
    max_contour = contour_info[0]

    #-- Create empty mask, draw filled polygon on it corresponding to largest contour ----
    # Mask is black, polygon is white
    mask = np.zeros(edges.shape)
    for c in contour_info:
        cv2.fillConvexPoly(mask, c[0], (255))

    #-- Smooth mask, then blur it
    mask = cv2.dilate(mask, None, iterations=MASK_DILATE_ITER)
    mask = cv2.erode(mask, None, iterations=MASK_ERODE_ITER)
    mask = cv2.GaussianBlur(mask, (BLUR, BLUR), 0)
    mask_stack = np.dstack([mask]*3)    # Create 3-channel alpha mask

    #-- Blend masked img into MASK_COLOR background
    mask_stack  = mask_stack.astype('float32') / 255.0
```

```
    img         = img.astype('float32') / 255.0
    masked = (mask_stack * img) + ((1-mask_stack) * MASK_COLOR)
    masked = (masked * 255).astype('uint8')

    cv2.imwrite(path+"MASK_"+xfile,masked)

    # Create name file masked
    f2_file = 'Mask_'+ xfile
    file2 = path + f2_file

    # Write masked image on disk
    print('   Save masked image with red background:', f2_file)
    cv2.imwrite(file2, masked)           # Save
    # Return name file masked and image masked
    return f2_file, masked

# Test
'''
xfile = 'Brass_001.tif'
f2_file, masked = red_background(path,xfile)
%matplotlib inline
plt.imshow(masked)
plt.title('Remove image background:'+xfile,fontsize=20)
plt.show()
'''
```

Out[16]:

```
"\nxfile = 'Brass_001.tif'\nf2_file, masked = red_background(path,xfile)\n%m
atplotlib inline\nplt.imshow(masked)\nplt.title('Remove image background:'+x
file,fontsize=20)\nplt.show()\n"
```

In [17]:

```
# https://convertingcolors.com/rgb-color-169_171_170.html

# return most_present RGB, RGB, color name, list RGB colors without RED, list RGB colors wi

import collections

def get_main_color_without_red_and_floor(path, f2_file):
    print('    Main color from image:', f2_file)
    file1 = path + f2_file

    # Read image
    img = read_pil_image(file1)
    if img == None:
        print("-->Unable to load image",file1)

    colors = img.getcolors( 1024*1024) #put a higher value if there are many colors in your
    #-----
    # Create list with colors without Background red color (near Background color)
    list_non_back = list()
    list_dec_back = list()   # List from decimal colors to list_non_back
    #
    print('...  List without excluded colors')
    # Convert list to decimal color
    for color in colors:
        # Diference between colors
       # print(color[1])
        rgb = color[1]

        excluded_rgb = False

        #Verify color name
        xt_color_name , hexdc = get_rgb_color_name(rgb)

        # Exclusion for some colors (Red Backgroud, Black foor, etc)
        if "red"   in xt_color_name:
             excluded_rgb = True
        if "black" in xt_color_name:
             excluded_rgb = True
        if "white" in xt_color_name:
            excluded_rgb = True
        if "cream" in xt_color_name:
            excluded_rgb = True

        # Force Only for non-tif files we do not delete anything
        if file.endswith('.tif'):
            excluded_rgb = False

        if  excluded_rgb == True:     # Exclude COLOR
            #print("Cor excluida", rgb, xt_color_name )
            excluded_rgb = True
        else:
            # OK COLOR - Save color in the list of correct colors (list_non_back)
            #print("Cor OK", rgb, xt_color_name )
            list_non_back.append(rgb)
            # Decimal color
            rgb_dec    = rgbToDecimal(rgb)
            list_dec_back.append(rgb_dec)

    #-----
```

```
    print('Count ocurrencies for color')
    most_present = 0

    # Most common color in the list - list_non_back
    x = collections.Counter(list_non_back)
    print('      4 Most common colors:', x.most_common(4))  # Five most common colors
    most_present = x.most_common(1)
    xrgb = list_non_back[0] # common color

    # ----- color name --
    #xt_color_name = get_color_name(xrgb)
    print('      Read color name:', xrgb)  # Color name from RGB
    xt_color_name , hexdc = get_rgb_color_name(xrgb)
    print('      Main Color file:', f2_file, ' RGB:', most_present, xrgb, ' Color name:', x

    return most_present, xrgb, xt_color_name, list_non_back, list_dec_back

# Test
#xfile = 'Copper_001.tif'
#most_present, xrgb, xt_color_name, list_non_back, \
#     list_dec_back = get_main_color_without_red_and_floor(path, xfile)
```

In [18]:

```
# https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histog
# Print histogram using Opencv
import cv2
import numpy as np
from matplotlib import pyplot as plt

def print_cv_hist(path, xfile):
    file1 = path + xfile
    print('Cv2 Hist from file:', file1)

    # Read image
    img_cv = read_cv2_image(file1)
    if img_cv.any() == None:
        print("-->Unable to load image",file1)

    # create a mask
    mask = np.zeros(img_cv.shape[:2], np.uint8)

    # define area to extract image from original
    #    Left:height , right:length
    mask[200:1400, 200:1800] = 255
    masked_img = cv2.bitwise_and(img_cv, img_cv ,mask = mask)

    # Calculate histogram with mask and without mask
    # Check third argument for mask
    hist_full = cv2.calcHist([img_cv],[0],None,[256],[0,256])
    hist_mask = cv2.calcHist([img_cv],[0],mask,[256],[0,256])

    plt.figure(figsize=(18,5))

    plt.subplot(141), plt.imshow(img_cv, 'gray')
    plt.title("Original")

    plt.subplot(142), plt.imshow(mask,'gray')
    plt.title('Mask')

    plt.subplot(143), plt.imshow(masked_img, 'gray')
    plt.title('Masked image')


    ax=plt.subplot(144), plt.plot(hist_full), plt.plot(hist_mask)
    ax = plt.gca()
    ax.grid(True)
    plt.title('Histogram')
    plt.xlim([0,256])

    plt.suptitle('IMAGE HISTOGRAM',fontsize=18)
    plt.xlabel('Image:'+xfile,fontsize=18)
    plt.ylabel('All chanels',fontsize=10)
    plt.savefig(path+'Hist_cv2_'+xfile)   # Save Histograme Figure
    plt.show()
    return

# Test
#xfile = 'Copper_001.tif'
#print_cv_hist(path, xfile)
```

In [19]:

```
# https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histog
# Print histogram using Opencv and matplotlib

import cv2
import numpy as np
from matplotlib import pyplot as plt

def print_matplot_hist(path, xfile):
    file1 = path + xfile
    print('Matplot Hist from file:', file1)

    # Read image
    img_mp = read_cv2_image(file1)
    if img_mp.any() == None:
        print("-->Unable to load image",file1)

    color = ('b','g','r')
    ax = plt.figure(figsize=(10,5))
    ax = plt.gca()
    ax.grid(True)

    for i,col in enumerate(color):
        histr = cv2.calcHist([img_mp],[i],None,[256],[0,256])
        plt.plot(histr,color = col, label='Band '+col.upper())
        plt.xlim([0,256])

    plt.title('Histogram of the image',fontsize=20)
    plt.xlabel('Image:'+xfile,fontsize=18)
    plt.ylabel('All chanels',fontsize=18)
    plt.legend(bbox_to_anchor=(.90,0.85),bbox_transform=plt.gcf().transFigure)
    plt.savefig(path+'Hist_'+xfile)   # Save Histograme Figure
    plt.show()

    return

# Test
#xfile = 'Copper_1.tif'
#print_matplot_hist(path, xfile)
```

In [20]:

```
# Max and Min value from Histogram and each position
#l = np.array(hist_full).tolist()  - Transform array in a list

import cv2
import numpy as np
from matplotlib import pyplot as plt

def histogram_max_min(path, xfile):

    file1 = path+xfile
    print('Histogram analisys:', file1)

    # Read image
    imgh = read_cv2_image(file1)
    if imgh.any() == None:
        print("-->Unable to load image",file1)

    # Calculate histogram without mask
    hist_full = cv2.calcHist([imgh],[0],None,[256],[0,256])

    # Transform array in a list
    hist_list = np.array(hist_full).tolist()

    # Valor maximo e minimo do Histograma e sua posição
    val_max  = max(hist_list)
    xval_max = int(val_max[0])

    val_avg  =  max(hist_list)
    xval_avg = int(val_avg[0]) / len(hist_list)
    xval_avg = int(xval_avg)

    val_min  = min(hist_list)
    xval_min = int(val_min[0])

    idx_max = hist_list.index(val_max)
    idx_min = hist_list.index(val_min)

    #print("Valor Max Histograma:", xval_max, '  Posição do valor Max:', idx_max)
    #print("Valor Min Histograma:", xval_min, '  Posição do valor Min:', idx_min)
    #print("Valor Avg Histograma:", xval_avg)

    return xval_max, idx_max, xval_min, idx_min

# Test
#xfile = 'Copper_001.tif'
#_,_,_,_ = histogram_max_min(path, xfile)
```

In [21]:

```
# Read image folder
import glob, os
def get_image_folder(xfile1):
    # Path to the image files
    path = currDir + "/" + folder + "/"
    # File
    file1 = path + xfile1
    print(file1)

    return file1
```

In [22]:

```
# Obtain percentage of channels R,G,B
import matplotlib.image as mpimg
def percent_rgb(path, xfile):
    print('    RGB percent from image:', xfile)
    emptyBlue = []
    emptyGreen= []
    emptyRed= []

    all_path = path + xfile
    # Read file
    img = mpimg.imread(all_path)
    imgplot = plt.imshow(img)
    # Mean of the array of each chanel
    RGBtuple = np.array(img).mean(axis=(0,1))

    averageRed = RGBtuple[0]
    averageGreen = RGBtuple[1]
    averageBlue = RGBtuple[2]

    percentageGreen = averageGreen/(averageRed+averageGreen+averageBlue) * 100
    percentageBlue = averageBlue/(averageRed+averageGreen+averageBlue) * 100
    percentageRed = averageRed/(averageRed+averageGreen+averageBlue) * 100

    emptyBlue+=[percentageBlue]
    emptyGreen+=[percentageGreen]
    emptyRed+=[percentageRed]
    print('     ------------------------------')
    print('     Percent Red',percentageRed)
    print('     Percent Green',percentageGreen)
    print('     Percent Blue',percentageBlue)
    print('     ------------------------------')
    return percentageRed, percentageGreen, percentageBlue
```

In [23]:

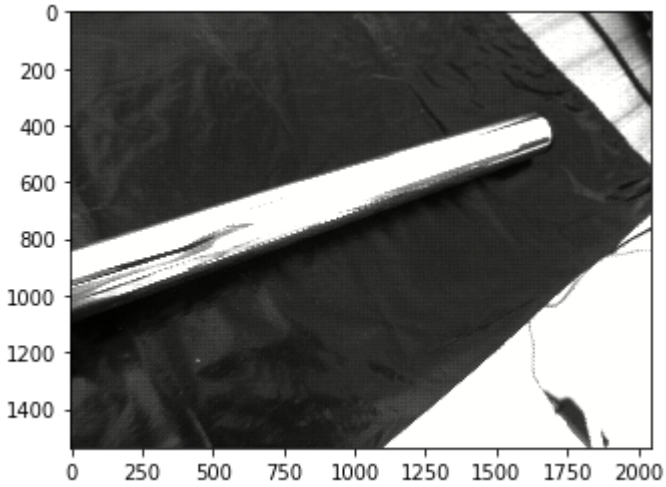
```
# Print all the informations from image, and create a pandas data frame with the relevant i

def print_file(path, xfile):
    print('--------------------------------------------------------------------------------')
    file1 = path + xfile

    # Read image
    tif_f1 = read_pil_image(file1)
    if tif_f1 == None:
        print("-->Unable to load image",file1)

    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)

    # Max and Min from Histogram
    xval_max, idx_max, xval_min, idx_min = histogram_max_min(path, xfile)

    # Percentage RGB
    perc_R, perc_G, perc_B = percent_rgb(path, xfile)

    # Get color
```

```
    # Enhancement Contrast color for better definition
    # f1_file has the file name saved enhanced
    xvalue = 2.0
    print('Enhancement color:', xfile, '  Value:',xvalue)
    x_enh, f1_file = contrast(path, xfile, xvalue)

    # Remove Background - Put red background
    # f2_file has the file name saved masked

    # Only red Background for NON tif files
    #xend_file = file.endswith('*.TIF').upper()
    if file.endswith('*.TIF'):
        f2_file = f1_file
        img_masked = tif_f1
    else:
        file1 = path+f1_file
        print('Red background:', path, f1_file)
        f2_file, img_masked = red_background(path, f1_file)

    # Get Main Color -
    print('Most common color:', path, f2_file)

    # most present color, RGB from most present color:
    # color name , Hex from rgb , list colors withour red, list colors without back, decima
    most_present, xrgb, xt_color_name, list_non_back,list_dec_back = get_main_color_without

    # HEX fom most present color
    hex_color  = rgb_to_hex(xrgb)

    # Decimal from most present color
    rgb_dec    = rgbToDecimal(xrgb)
    #----
    # 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
    print('          Histog   :',hsum ,'  N.List elem:', nlist, ' Max:', xval_max, 'Idx Max
    print('          Color    :',xt_color_name,'   RGB   :',xrgb, '   Hex color:', hex_colo
    print('          Extremes :',extr_a, 'Med Extremes:',med_extr)
    print('          Percentage R:', perc_R,'  Percentage G:', perc_B, '  Percentage B:', p

    # 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, xt_color_name, xrgb, hex_
                     rgb_dec, med_extr, xval_max, idx_max, xval_min, idx_min,
                     perc_R, perc_G, perc_B))

    return most_present, xrgb, xt_color_name, list_non_back, list_dec_back
```

Starting image analysis

In [24]:

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

xend_file = "*" + end_file
# change work to folder path
os.chdir(path)
print('Analysing Images from:',path, xend_file)

for file in glob.glob(xend_file):
    list_dec_back = list() # List with decimal colors in the image
    print(file)
    most_present, xrgb, xt_color_name, list_non_back, list_dec_back = print_file(path,file)
```



In [26]:

```
#list_dec_back ordered
order_list_dec = sorted(list_dec_back, key=int)
#order_list_dec
#list_non_back
```

In [27]:

```
'''
TESTS
# Read all list to see the color - obtain RGB from int
for x in order_list_dec:
    #print(x)
    # Get RGB from INT
    xrgb = getRGBfromI(x)
    #print('Int:, x,'  RGB: ',xrgb)
    xt_color_name , hexdc = get_rgb_color_name(xrgb)
    print('Int:', x,'  RGB: ', xrgb, xt_color_name)
'''
```

Out[27]:

```
"\nTESTS\n# Read all list to see the color - obtain RGB from int\nfor x in o
rder_list_dec:\n    #print(x)\n    # Get RGB from INT\n    xrgb = getRGBfrom
I(x)\n    #print('Int:, x,'  RGB: ',xrgb)\n    xt_color_name , hexdc = get_r
gb_color_name(xrgb)\n    print('Int:', x,'  RGB: ', xrgb, xt_color_name)\n"
```

In [28]:

```
df = pd.DataFrame(df_image,columns=['Folder','File','Material','Size','Format','Mode',
                                    'All_Bands', 'Sum_Ch0','Sum_Ch1','Sum_Ch2',
                                    'Histogram','Number_elements',
                                    'Color','Color_RGB', 'Color_hex','Color_dec','Med_Extre
                                    'Max_Histog', 'Idx_Max_Histog','Min_Histog', 'Idx_Min_H
                                    'perc_R', 'perc_G', 'perc_B'])
df.head(50)
```

Out[28]:

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_Ch0	Sum_Ch1
0	imagedata06	Aluminum_1.tif	Aluminum	(2048, 1536)	TIFF	RGB	658.137536	220.999978	220.999978
1	imagedata06	Aluminum_2.tif	Aluminum	(2048, 1536)	TIFF	RGB	693.024169	232.705276	232.705276
2	imagedata06	Aluminum_3.tif	Aluminum	(2048, 1536)	TIFF	RGB	696.532319	233.779842	233.779842
3	imagedata06	Aluminum_4.tif	Aluminum	(2048, 1536)	TIFF	RGB	658.137536	220.999978	220.999978
4	imagedata06	Aluminum_5.tif	Aluminum	(2048, 1536)	TIFF	RGB	693.024169	232.705276	232.705276
5	imagedata06	Aluminum_6.tif	Aluminum	(2048, 1536)	TIFF	RGB	696.532319	233.779842	233.779842

In [29]:

```
# Delete junk records
df = df[df.Material != 'MASK']
df = df[df.Material != 'Enh']
df
```

Out[29]:

	Folder	File	Material	Size	Format	Mode	All_Bands	Sum_Ch0	Sum_Ch1
0	imagedata06	Aluminum_1.tif	Aluminum	(2048, 1536)	TIFF	RGB	658.137536	220.999978	220.999978
1	imagedata06	Aluminum_2.tif	Aluminum	(2048, 1536)	TIFF	RGB	693.024169	232.705276	232.705276
2	imagedata06	Aluminum_3.tif	Aluminum	(2048, 1536)	TIFF	RGB	696.532319	233.779842	233.779842
3	imagedata06	Aluminum_4.tif	Aluminum	(2048, 1536)	TIFF	RGB	658.137536	220.999978	220.999978
4	imagedata06	Aluminum_5.tif	Aluminum	(2048, 1536)	TIFF	RGB	693.024169	232.705276	232.705276
5	imagedata06	Aluminum_6.tif	Aluminum	(2048, 1536)	TIFF	RGB	696.532319	233.779842	233.779842

Write statistics in excel book

In [30]:

```
# 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
```

Plot

In [31]:

```
df_plot = pd.DataFrame(df, columns=["Material", "All_Bands", "Sum_Ch0", "Sum_Ch1", "Sum_Ch2
                                    "Color", "Color_RGB", "Color_hex","Color_dec",
                                    "Med_Extrems", "Max_Histog", "Idx_Max_Histog","Min_Hist
                                    "Idx_Min_Histog","perc_R", "perc_G", "perc_B"])
df_plot
```

Out[31]:

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color	Color_RGB	Color_he
0	Aluminum	658.137536	220.999978	220.999978	216.137580	white	(255, 255, 255)	#ffff
1	Aluminum	693.024169	232.705276	232.705276	227.613617	white	(255, 255, 255)	#ffff
2	Aluminum	696.532319	233.779842	233.779842	228.972635	white	(255, 255, 255)	#ffff
3	Aluminum	658.137536	220.999978	220.999978	216.137580	white	(255, 255, 255)	#ffff
4	Aluminum	693.024169	232.705276	232.705276	227.613617	white	(255, 255, 255)	#ffff
5	Aluminum	696.532319	233.779842	233.779842	228.972635	white	(255, 255, 255)	#ffff
6	Brass	562.098115	189.033946	189.033946	184.030223	white	(255, 255, 255)	#ffff
7	Brass	449.396523	151.345791	151.345791	146.704941	white	(255, 255, 255)	#ffff
8	Brass	469.763357	158.316124	158.316124	153.131109	white	(255, 255, 255)	#ffff
9	Brass	562.098115	189.033946	189.033946	184.030223	white	(255, 255, 255)	#ffff
10	Brass	449.396523	151.345791	151.345791	146.704941	white	(255, 255, 255)	#ffff
11	Brass	469.763357	158.316124	158.316124	153.131109	white	(255, 255, 255)	#ffff
12	CopperWire	521.051218	175.360057	175.360057	170.331104	white	(255, 255, 255)	#ffff
13	CopperWire	634.394685	213.195874	213.195874	208.002937	white	(255, 255, 255)	#ffff
14	CopperWire	481.236521	162.041057	162.041057	157.154407	white	(255, 255, 255)	#ffff
15	CopperWire	564.848218	189.837464	189.837464	185.173290	white	(255, 255, 255)	#ffff
16	CopperWire	521.051218	175.360057	175.360057	170.331104	white	(255, 255, 255)	#ffff
17	CopperWire	634.394685	213.195874	213.195874	208.002937	white	(255, 255, 255)	#ffff
18	CopperWire	481.236521	162.041057	162.041057	157.154407	white	(255, 255, 255)	#ffff
19	CopperWire	564.848218	189.837464	189.837464	185.173290	white	(255, 255, 255)	#ffff

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color	Color_RGB	Color_he
20	Copper	447.139193	150.632053	150.632053	145.875087	white	(255, 255, 255)	#ffff
21	Copper	588.109624	197.635200	197.635200	192.839224	white	(255, 255, 255)	#ffff
22	Copper	447.139193	150.632053	150.632053	145.875087	white	(255, 255, 255)	#ffff
23	Copper	588.109624	197.635200	197.635200	192.839224	white	(255, 255, 255)	#ffff
24	Iron	641.517597	215.555740	215.555740	210.406117	white	(255, 255, 255)	#ffff
25	Iron	641.517597	215.555740	215.555740	210.406117	white	(255, 255, 255)	#ffff
26	Iron	601.674947	202.340947	202.340947	196.993053	white	(255, 255, 255)	#ffff
27	Iron	575.482113	193.527250	193.527250	188.427613	white	(255, 255, 255)	#ffff
28	PaintedIron	495.478663	166.811124	166.811124	161.856415	white	(255, 255, 255)	#ffff
29	PaintedIron	606.481866	203.824174	203.824174	198.833518	white	(255, 255, 255)	#ffff
30	PaintedIron	573.568470	192.841021	192.841021	187.886428	white	(255, 255, 255)	#ffff
31	PaintedIron	495.478663	166.811124	166.811124	161.856415	white	(255, 255, 255)	#ffff
32	PaintedIron	606.481866	203.824174	203.824174	198.833518	white	(255, 255, 255)	#ffff
33	PaintedIron	573.568470	192.841021	192.841021	187.886428	white	(255, 255, 255)	#ffff
34	PaintedIron	606.481866	203.824174	203.824174	198.833518	white	(255, 255, 255)	#ffff
35	StainlessSteel	654.906666	219.931246	219.931246	215.044174	white	(255, 255, 255)	#ffff
36	StainlessSteel	573.675257	192.847695	192.847695	187.979867	white	(255, 255, 255)	#ffff
37	StainlessSteel	654.906666	219.931246	219.931246	215.044174	white	(255, 255, 255)	#ffff
38	StainlessSteel	573.675257	192.847695	192.847695	187.979867	white	(255, 255, 255)	#ffff
39	StainlessSteel	638.936045	214.726624	214.726624	209.482797	white	(255, 255, 255)	#ffff
40	StainlessSteel	731.365569	245.627645	245.627645	240.110279	white	(255, 255, 255)	#ffff
41	StainlessSteel	595.649206	200.158029	200.158029	195.333148	white	(255, 255, 255)	#ffff
42	StainlessSteel	611.667734	205.504045	205.504045	200.659644	white	(255, 255, 255)	#ffff
43	StainlessSteel	860.058381	288.366525	288.366525	283.325331	white	(255, 255, 255)	#ffff

In [32]:

```
# Adjust values to plot
df_plot.Sum_Ch0        = df_plot.Sum_Ch0 + 500 # to have diference lines during plot
df_plot.Sum_Ch1        = df_plot.Sum_Ch1 + 1000
df_plot.Sum_Ch2        = df_plot.Sum_Ch2 + 1500
df_plot.All_Bands      = df_plot.All_Bands + 2000

df_plot.Color_dec      = df_plot.Color_dec / 1000
df_plot.Color_dec      = df_plot.Color_dec - 10000
df_plot.Med_Extrems    = df_plot.Med_Extrems + 500
df_plot.Max_Histog     = df_plot.Max_Histog / 1000
df_plot.Idx_Max_Histog = df_plot.Idx_Max_Histog + 1000
df_plot.Min_Histog     = df_plot.Min_Histog * 100
df_plot.Idx_Min_Histog = df_plot.Idx_Min_Histog * 10

df_plot.perc_R = df_plot.perc_R + 1000
df_plot.perc_G = df_plot.perc_G + 1100
df_plot.perc_B = df_plot.perc_B + 1200

df_plot
```

Out[32]:

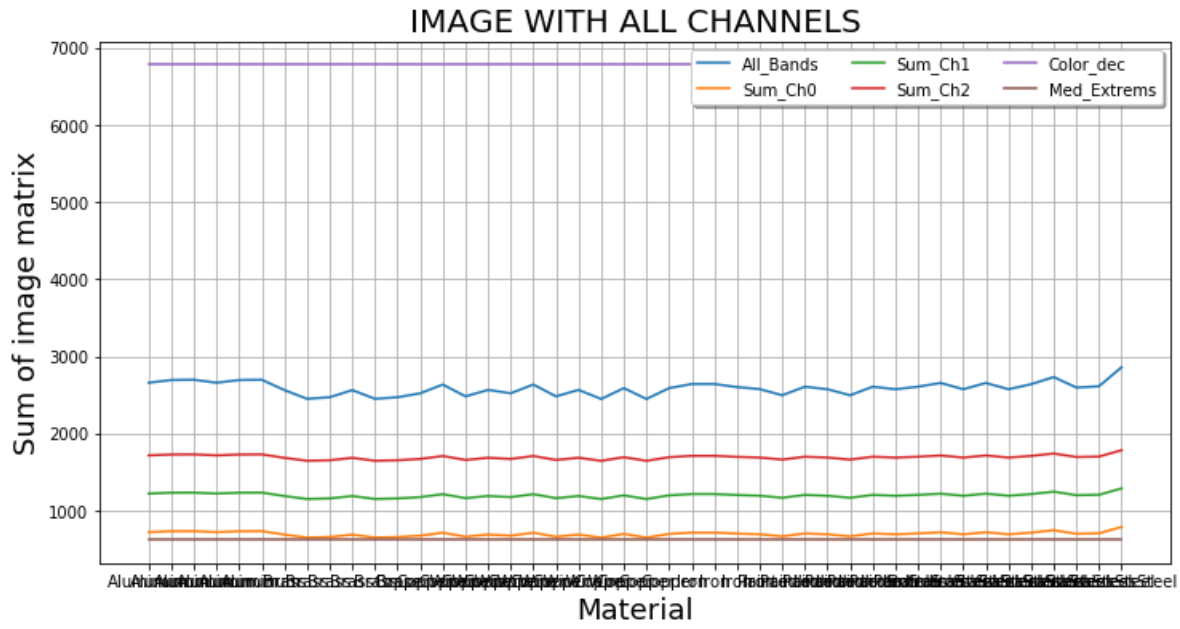
	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color	Color_RGB	Color
0	Aluminum	2658.137536	720.999978	1220.999978	1716.137580	white	(255, 255, 255)	
1	Aluminum	2693.024169	732.705276	1232.705276	1727.613617	white	(255, 255, 255)	
2	Aluminum	2696.532319	733.779842	1233.779842	1728.972635	white	(255, 255, 255)	
3	Aluminum	2658.137536	720.999978	1220.999978	1716.137580	white	(255, 255, 255)	
4	Aluminum	2693.024169	732.705276	1232.705276	1727.613617	white	(255, 255, 255)	
5	Aluminum	2696.532319	733.779842	1233.779842	1728.972635	white	(255, 255, 255)	
6	Brass	2562.098115	689.033946	1189.033946	1684.030223	white	(255, 255, 255)	
7	Brass	2449.396523	651.345791	1151.345791	1646.704941	white	(255, 255, 255)	
8	Brass	2469.763357	658.316124	1158.316124	1653.131109	white	(255, 255, 255)	
9	Brass	2562.098115	689.033946	1189.033946	1684.030223	white	(255, 255, 255)	
10	Brass	2449.396523	651.345791	1151.345791	1646.704941	white	(255, 255, 255)	
11	Brass	2469.763357	658.316124	1158.316124	1653.131109	white	(255, 255, 255)	
12	CopperWire	2521.051218	675.360057	1175.360057	1670.331104	white	(255, 255, 255)	
13	CopperWire	2634.394685	713.195874	1213.195874	1708.002937	white	(255, 255, 255)	

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color	Color_RGB	Color
14	CopperWire	2481.236521	662.041057	1162.041057	1657.154407	white	(255, 255, 255)	
15	CopperWire	2564.848218	689.837464	1189.837464	1685.173290	white	(255, 255, 255)	
16	CopperWire	2521.051218	675.360057	1175.360057	1670.331104	white	(255, 255, 255)	
17	CopperWire	2634.394685	713.195874	1213.195874	1708.002937	white	(255, 255, 255)	
18	CopperWire	2481.236521	662.041057	1162.041057	1657.154407	white	(255, 255, 255)	
19	CopperWire	2564.848218	689.837464	1189.837464	1685.173290	white	(255, 255, 255)	
20	Copper	2447.139193	650.632053	1150.632053	1645.875087	white	(255, 255, 255)	
21	Copper	2588.109624	697.635200	1197.635200	1692.839224	white	(255, 255, 255)	
22	Copper	2447.139193	650.632053	1150.632053	1645.875087	white	(255, 255, 255)	
23	Copper	2588.109624	697.635200	1197.635200	1692.839224	white	(255, 255, 255)	
24	Iron	2641.517597	715.555740	1215.555740	1710.406117	white	(255, 255, 255)	
25	Iron	2641.517597	715.555740	1215.555740	1710.406117	white	(255, 255, 255)	
26	Iron	2601.674947	702.340947	1202.340947	1696.993053	white	(255, 255, 255)	
27	Iron	2575.482113	693.527250	1193.527250	1688.427613	white	(255, 255, 255)	
28	PaintedIron	2495.478663	666.811124	1166.811124	1661.856415	white	(255, 255, 255)	
29	PaintedIron	2606.481866	703.824174	1203.824174	1698.833518	white	(255, 255, 255)	
30	PaintedIron	2573.568470	692.841021	1192.841021	1687.886428	white	(255, 255, 255)	
31	PaintedIron	2495.478663	666.811124	1166.811124	1661.856415	white	(255, 255, 255)	
32	PaintedIron	2606.481866	703.824174	1203.824174	1698.833518	white	(255, 255, 255)	
33	PaintedIron	2573.568470	692.841021	1192.841021	1687.886428	white	(255, 255, 255)	
34	PaintedIron	2606.481866	703.824174	1203.824174	1698.833518	white	(255, 255, 255)	
35	StainlessSteel	2654.906666	719.931246	1219.931246	1715.044174	white	(255, 255, 255)	
36	StainlessSteel	2573.675257	692.847695	1192.847695	1687.979867	white	(255, 255, 255)	
37	StainlessSteel	2654.906666	719.931246	1219.931246	1715.044174	white	(255, 255, 255)	
38	StainlessSteel	2573.675257	692.847695	1192.847695	1687.979867	white	(255, 255, 255)	

	Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color	Color_RGB	Color
39	StainlessSteel	2638.936045	714.726624	1214.726624	1709.482797	white	(255, 255, 255)	
40	StainlessSteel	2731.365569	745.627645	1245.627645	1740.110279	white	(255, 255, 255)	
41	StainlessSteel	2595.649206	700.158029	1200.158029	1695.333148	white	(255, 255, 255)	
42	StainlessSteel	2611.667734	705.504045	1205.504045	1700.659644	white	(255, 255, 255)	
43	StainlessSteel	2860.058381	788.366525	1288.366525	1783.325331	white	(255, 255, 255)	

In [33]:

```
df_plot.plot(y=["All_Bands","Sum_Ch0","Sum_Ch1", "Sum_Ch2","Color_dec","Med_Extrems"],
             figsize=(12,6), grid=True )

# 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.legend(loc='upper right', ncol=3, fancybox=True, shadow=True)
plt.savefig(folder+"_Line Graph all channels information.png")
plt.show()
```

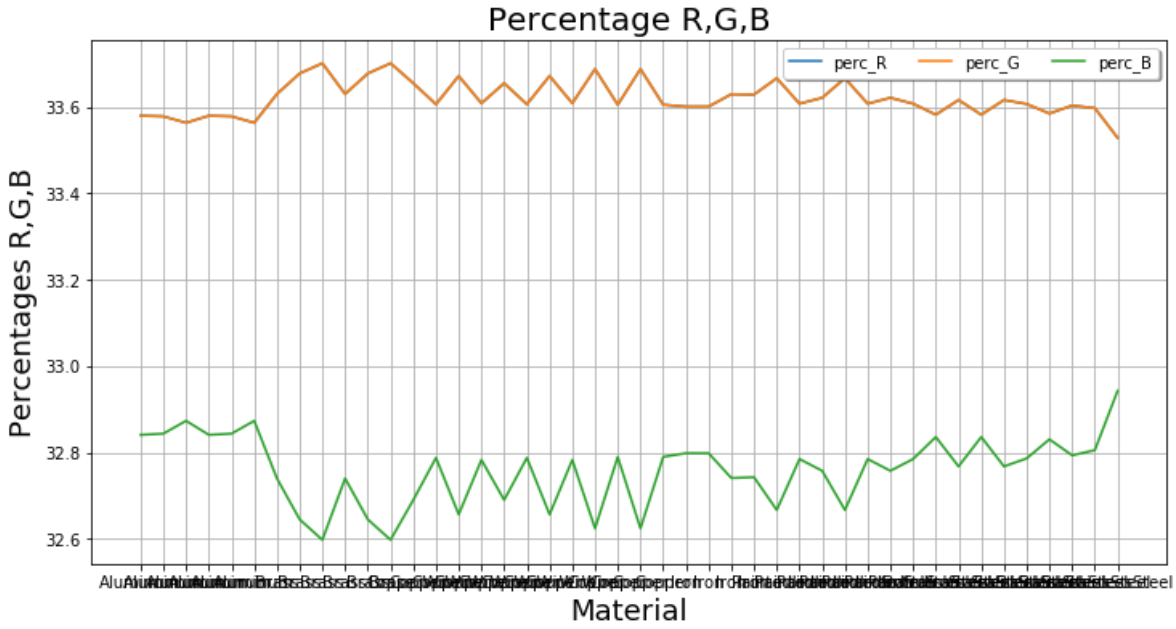


In [34]:

```
df_plot.perc_R = df_plot.perc_R - 1000
df_plot.perc_G = df_plot.perc_G - 1100
df_plot.perc_B = df_plot.perc_B - 1200
```

In [35]:

```
df_plot.plot(y=["perc_R","perc_G","perc_B"],
             figsize=(12,6), grid=True )

# 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('Percentage R,G,B',fontsize=20)
plt.ylabel('Percentages R,G,B',fontsize=18)
plt.xticks(xtick_loc, df_plot.Material, rotation=0)
plt.xlabel('Material',fontsize=18)
plt.legend(loc='upper right', ncol=3, fancybox=True, shadow=True)
plt.savefig(folder+"_Line Graph Percentage RGB.png")
plt.show()
```



In [36]:

```
# Create pivot table
df_plot1 = df_plot.groupby('Material')['All_Bands', 'Sum_Ch0','Sum_Ch1','Sum_Ch2','Color_de
                                       'Med_Extrems', 'Max_Histog', 'Idx_Max_Histog','Min_H
                                       'Idx_Min_Histog','perc_R','perc_G','perc_B'].mean()
df_plot1
```

Out[36]:

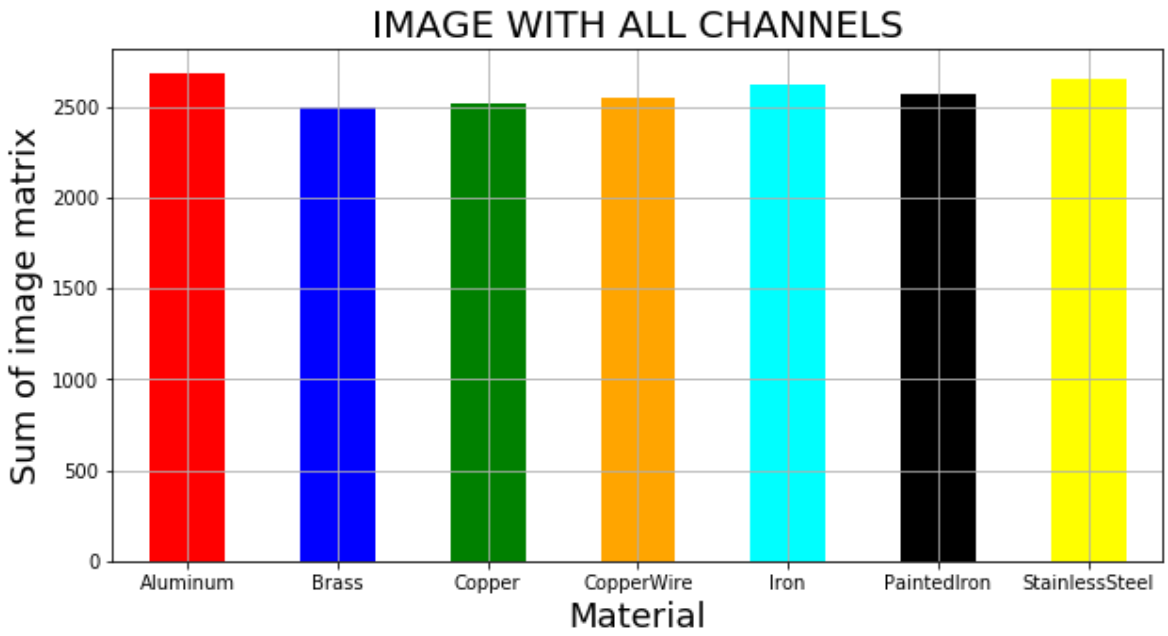
Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color_dec	Med_Extrems	M
Aluminum	2682.564675	729.161699	1229.161699	1724.241277	6777.215	627.833333	2
Brass	2493.752665	666.231954	1166.231954	1661.288758	6777.215	627.833333	1
Copper	2517.624408	674.133626	1174.133626	1669.357156	6777.215	627.833333	1
CopperWire	2550.382660	685.108613	1185.108613	1680.165434	6777.215	627.833333	1
Iron	2615.048063	706.744919	1206.744919	1701.558225	6777.215	627.833333	3
PaintedIron	2565.362838	690.110973	1190.110973	1685.140891	6777.215	627.833333	3
StainlessSteel	2654.982309	719.993417	1219.993417	1714.995476	6777.215	627.833333	4

In [37]:

```
color = ['red','blue','green','orange','cyan','black','yellow']
```

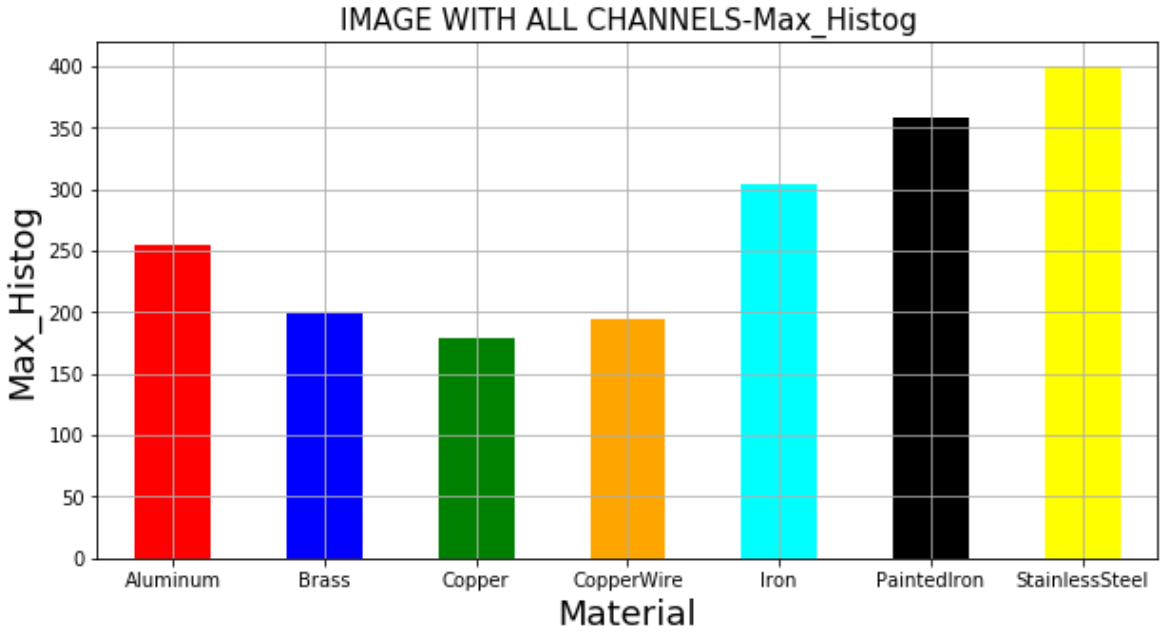
In [38]:

```
df_All_Bands = pd.DataFrame(df_plot1.All_Bands)

df_All_Bands.plot(kind='bar', y=0, color=color, legend=False, rot=0, figsize=(10,5))
plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.grid(True)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Sum of image matrix',fontsize=18)
plt.savefig(folder+"_Sum of image matrix.png")
plt.show()
```



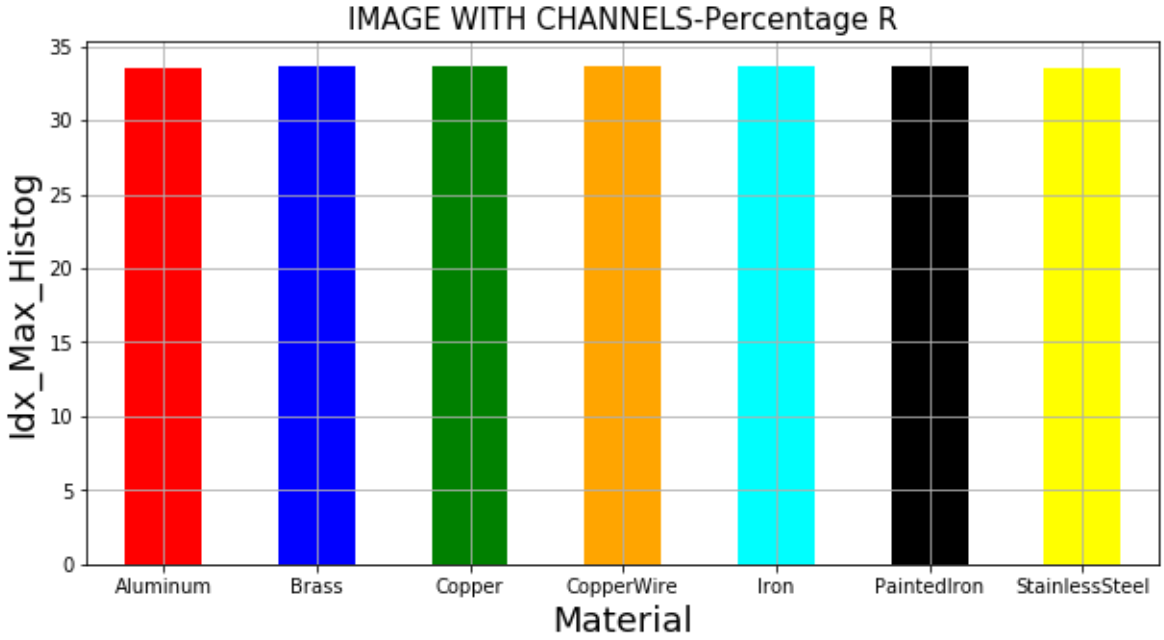
In [39]:

```
df_Max_Histog = pd.DataFrame(df_plot1.Max_Histog)

df_Max_Histog.plot(kind='bar', y=0, color=color, legend=False, rot=0, figsize=(10,5))
plt.title('IMAGE WITH ALL CHANNELS-Max_Histog',fontsize=15)
plt.grid(True)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Max_Histog',fontsize=18)
plt.savefig(folder+"_Max_Histog.png")
plt.show()
```



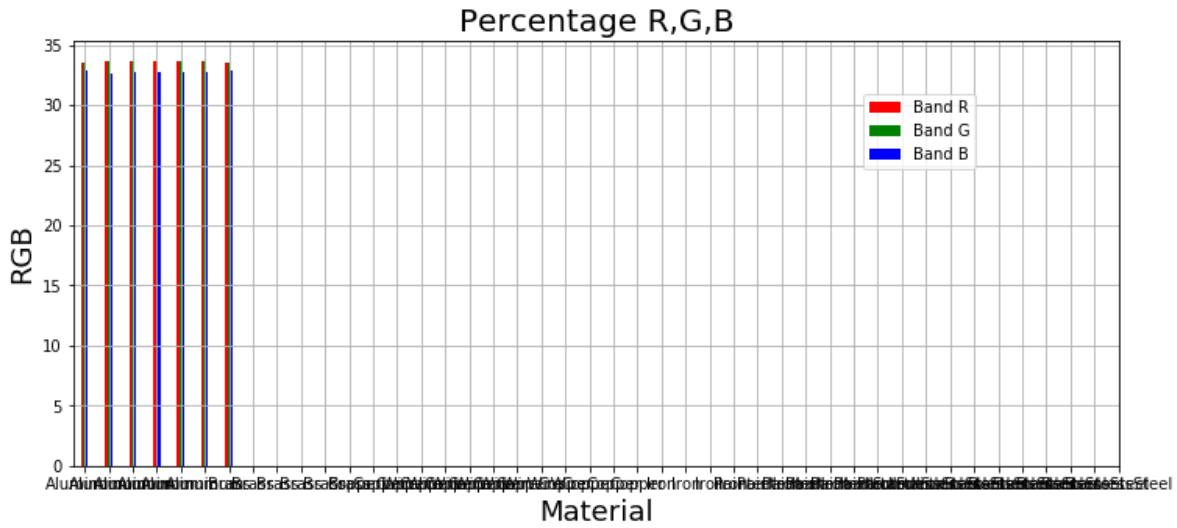
In [40]:

```
df_perc = pd.DataFrame(df_plot1.perc_R)

df_perc.plot(kind='bar', y=0, color=color, legend=False, rot=0, figsize=(10,5))
plt.title('IMAGE WITH CHANNELS-Percentage R',fontsize=15)
plt.grid(True)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Idx_Max_Histog',fontsize=18)
plt.show()
```



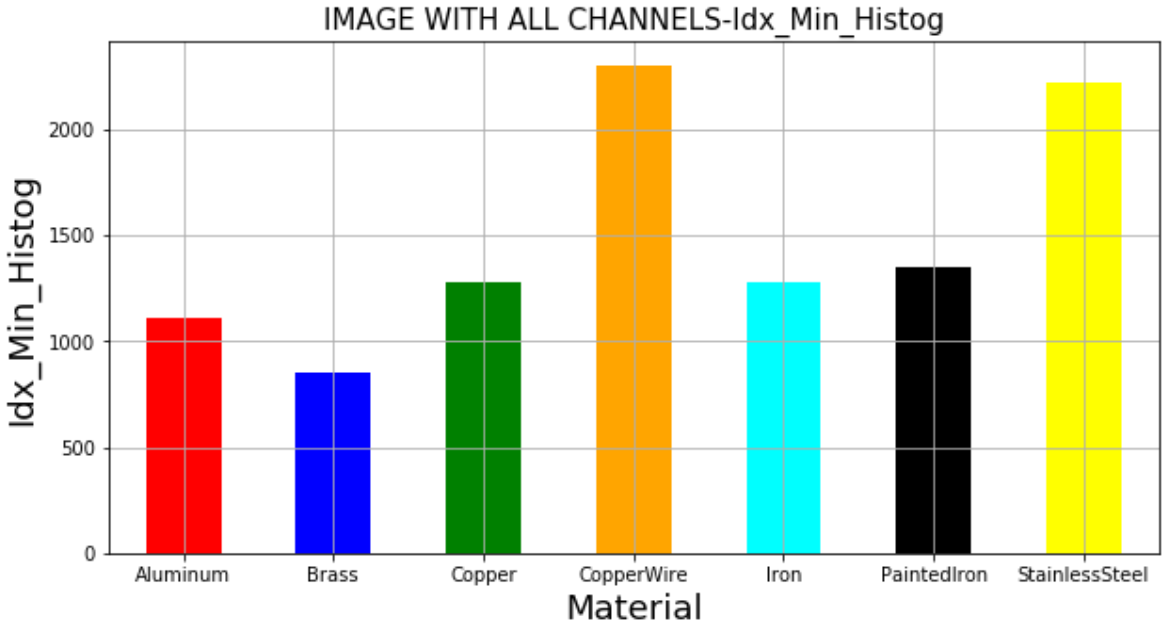
In [41]:

```
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))-0.0 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))+0.1 # 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_r, df_plot1.perc_R, color='red', width=0.1, label='Band R')
plt.bar(loc_g, df_plot1.perc_G, color='green', width=0.1,label='Band G')
plt.bar(loc_b, df_plot1.perc_B, color='blue', width=0.1,label='Band B')

plt.title('Percentage R,G,B',fontsize=20)
plt.xlabel('Material',fontsize=18)
plt.ylabel('RGB',fontsize=18)
plt.grid(True)
plt.xticks(xtick_loc, xticks, rotation=0)
plt.legend(bbox_to_anchor=(.8,0.8),\
    bbox_transform=plt.gcf().transFigure)
plt.savefig(folder+"_Bar Diagram_perc_RGB.png")
plt.show()
```



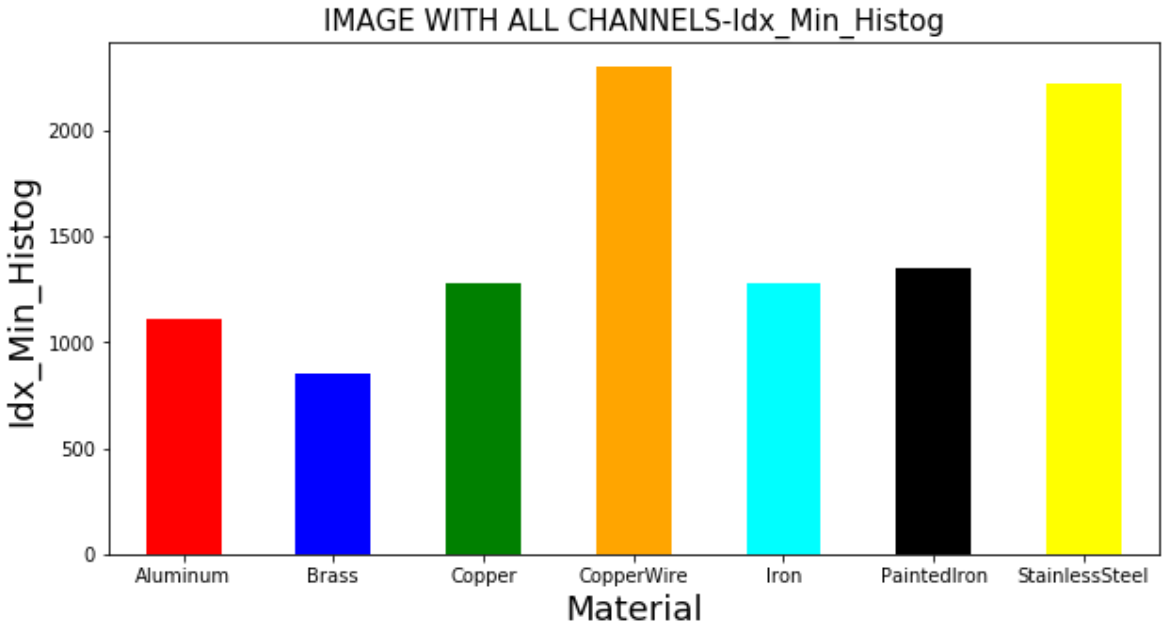
In [42]:

```
df_Idx_Min_Histog = pd.DataFrame(df_plot1.Idx_Min_Histog)

df_Idx_Min_Histog.plot(kind='bar', y=0, color=color, legend=False, rot=0, figsize=(10,5))
plt.title('IMAGE WITH ALL CHANNELS-Idx_Min_Histog',fontsize=15)
plt.grid(True)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Idx_Min_Histog',fontsize=18)
plt.savefig(folder+"_Idx_Min_Histogram.png")
plt.show()
```



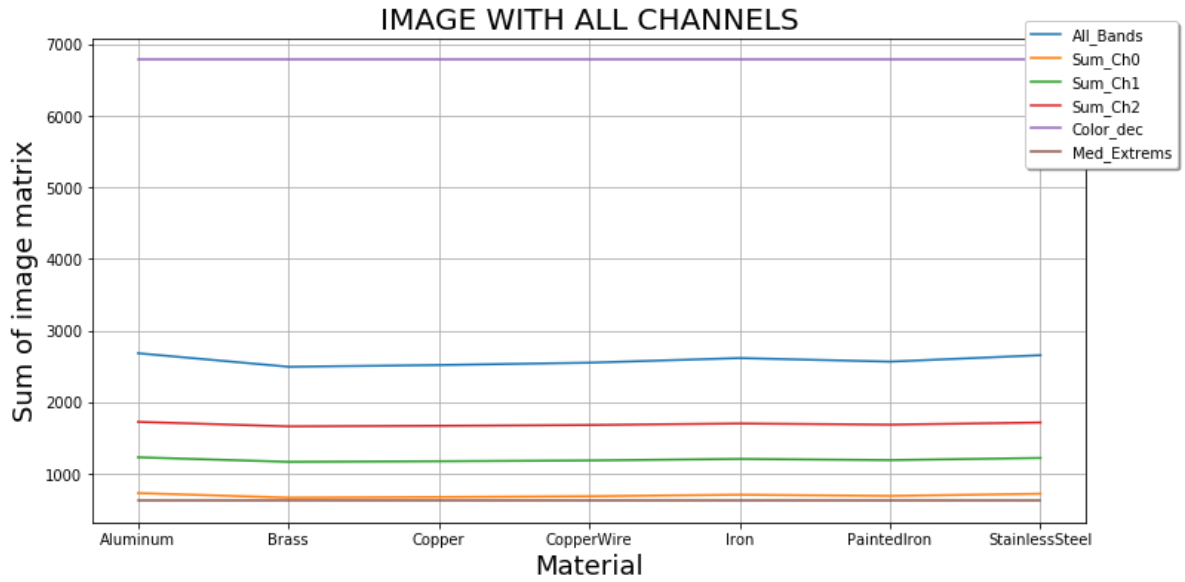
In [43]:

```
df_Idx_Min_Histog = pd.DataFrame(df_plot1.Idx_Min_Histog)

df_Idx_Min_Histog.plot(kind='bar', y=0, color=color, legend=False, rot=0, figsize=(10,5))
plt.title('IMAGE WITH ALL CHANNELS-Idx_Min_Histog',fontsize=15)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Idx_Min_Histog',fontsize=18)
plt.show()
```



In [44]:

```
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","Color_dec","Med_Extrems"],
              figsize=(12,6), grid=True )
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.legend(loc='upper right', ncol=1, fancybox=True, shadow=True,bbox_to_anchor=(1.1, 1.05))
plt.savefig(folder+"_resume all channels.png")
plt.show()
```



In [45]:

```
df_plot1
```

Out[45]:

Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color_dec	Med_Extrems	M
Aluminum	2682.564675	729.161699	1229.161699	1724.241277	6777.215	627.833333	2
Brass	2493.752665	666.231954	1166.231954	1661.288758	6777.215	627.833333	1
Copper	2517.624408	674.133626	1174.133626	1669.357156	6777.215	627.833333	1
CopperWire	2550.382660	685.108613	1185.108613	1680.165434	6777.215	627.833333	1
Iron	2615.048063	706.744919	1206.744919	1701.558225	6777.215	627.833333	3
PaintedIron	2565.362838	690.110973	1190.110973	1685.140891	6777.215	627.833333	3
StainlessSteel	2654.982309	719.993417	1219.993417	1714.995476	6777.215	627.833333	4

In [46]:

```
# Copy dataframe to arrange values
df_plot2 =  df_plot1.copy()
df_plot2
```

Out[46]:

Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color_dec	Med_Extrems	M
Aluminum	2682.564675	729.161699	1229.161699	1724.241277	6777.215	627.833333	2
Brass	2493.752665	666.231954	1166.231954	1661.288758	6777.215	627.833333	1
Copper	2517.624408	674.133626	1174.133626	1669.357156	6777.215	627.833333	1
CopperWire	2550.382660	685.108613	1185.108613	1680.165434	6777.215	627.833333	1
Iron	2615.048063	706.744919	1206.744919	1701.558225	6777.215	627.833333	3
PaintedIron	2565.362838	690.110973	1190.110973	1685.140891	6777.215	627.833333	3
StainlessSteel	2654.982309	719.993417	1219.993417	1714.995476	6777.215	627.833333	4

In [47]:

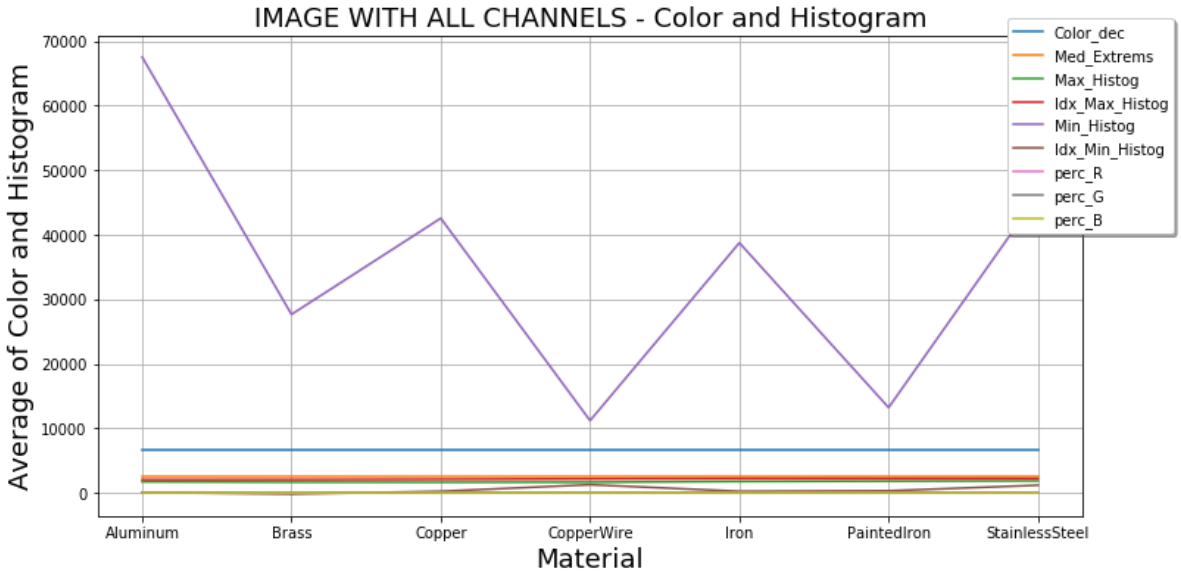
```
df_plot2.Med_Extrems    = df_plot2.Med_Extrems + 2000
df_plot2.Max_Histog     = df_plot2.Max_Histog  + 1500
df_plot2.Idx_Max_Histog = df_plot2.Idx_Max_Histog + 1000
df_plot2.Min_Histog     = df_plot2.Min_Histog + 500
df_plot2.Idx_Min_Histog = df_plot2.Idx_Min_Histog - 1000
df_plot2.head()
```

Out[47]:

Material	All_Bands	Sum_Ch0	Sum_Ch1	Sum_Ch2	Color_dec	Med_Extrems	Max
Aluminum	2682.564675	729.161699	1229.161699	1724.241277	6777.215	2627.833333	1755
Brass	2493.752665	666.231954	1166.231954	1661.288758	6777.215	2627.833333	1699
Copper	2517.624408	674.133626	1174.133626	1669.357156	6777.215	2627.833333	1678
CopperWire	2550.382660	685.108613	1185.108613	1680.165434	6777.215	2627.833333	1694
Iron	2615.048063	706.744919	1206.744919	1701.558225	6777.215	2627.833333	1803

In [48]:

```
loc_Array_sum = np.arange(len(df_plot2.index))
xtick_loc = list(loc_Array_sum)
xticks = list(df_plot1.index)

df_plot2.plot( y=['Color_dec','Med_Extrems', 'Max_Histog', 'Idx_Max_Histog',
                  'Min_Histog','Idx_Min_Histog','perc_R','perc_G','perc_B'],figsize=(12,6),
plt.xticks(xtick_loc, df_plot2.index, rotation=0)
plt.title('IMAGE WITH ALL CHANNELS - Color and Histogram',fontsize=18)
plt.xlabel('Material',fontsize=18)
plt.ylabel('Average of Color and Histogram',fontsize=18)
plt.legend(loc='upper right', ncol=1, fancybox=True, shadow=True,bbox_to_anchor=(1.1, 1.05)
plt.savefig(folder+"_color_and_histogram.png")
plt.show()
```



In [49]:

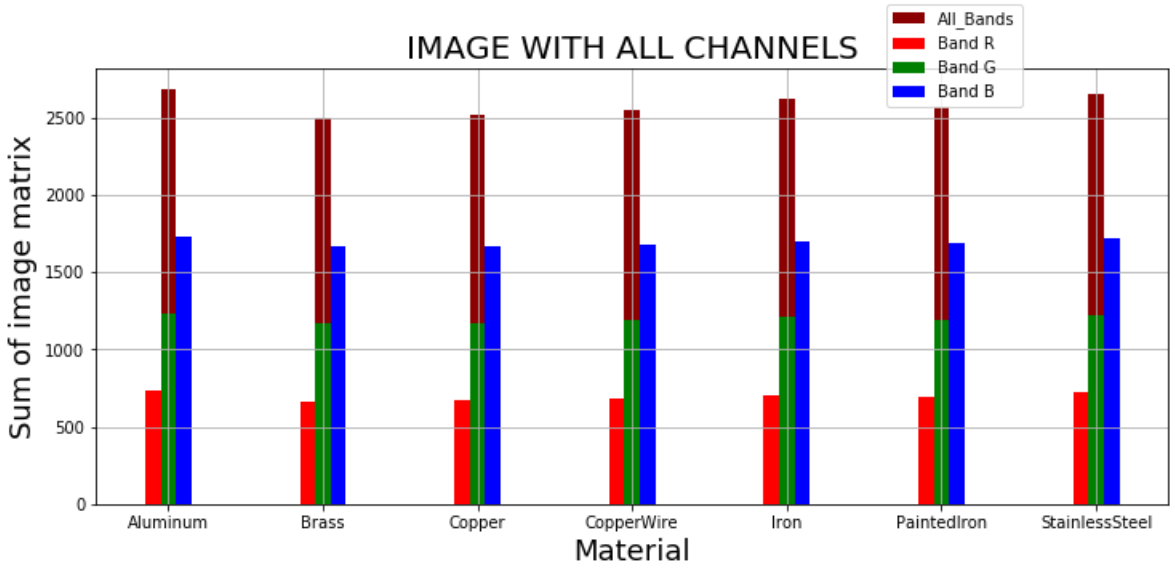
```
# Create Xlabels
loc_Array_sum = np.arange(len(df_plot1.index))+0.0 # Offsetting the tick-label location
loc_r = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.0 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))-0.1 # Offsetting the tick-label location

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

In [50]:

```
# Plot
```

In [51]:

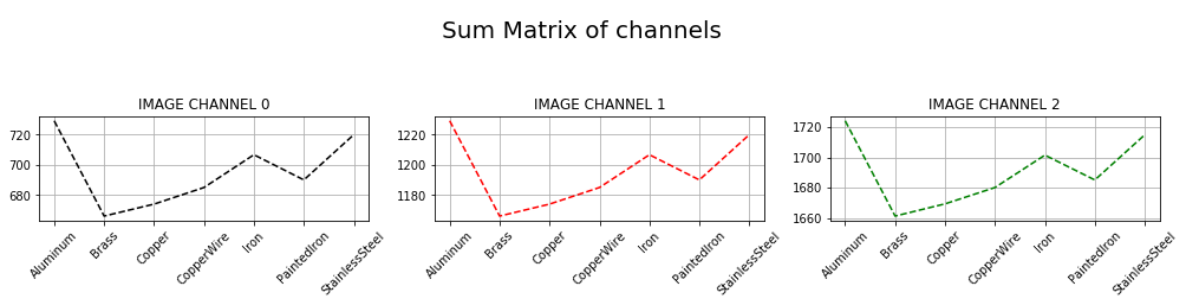
```
# Plot  Bar Graph
#df_plot1.plot(kind='bar', figsize=(12,5), grid=True, color='darkred',fontsize=18)
loc_Array_sum = np.arange(len(df_plot1.index))+0.0 # Offsetting the tick-label location
loc_b = np.arange(len(df_plot1.index))+0.1 # Offsetting the tick-label location
loc_g = np.arange(len(df_plot1.index))-0.0 # Offsetting the tick-label location
loc_r = np.arange(len(df_plot1.index))-0.1 # 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=0.1, label='All_Bands')
plt.bar(loc_r, df_plot1.Sum_Ch0, color=colors[1], width=0.1,label='Band R')
plt.bar(loc_g, df_plot1.Sum_Ch1, color=colors[2], width=0.1,label='Band G')
plt.bar(loc_b, df_plot1.Sum_Ch2, color=colors[3], width=0.1,label='Band B')

plt.title('IMAGE WITH ALL CHANNELS',fontsize=20)
plt.grid(True)
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.savefig(folder+"_all bands.png")
plt.show()
```



In [53]:

```
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.suptitle('Sum Matrix of channels',fontsize=20,y=1.08)
#plt.tight_layout()
plt.subplots_adjust(top=0.8)
plt.savefig(folder+"_Sum Matrix of channels.png")
plt.show()
```

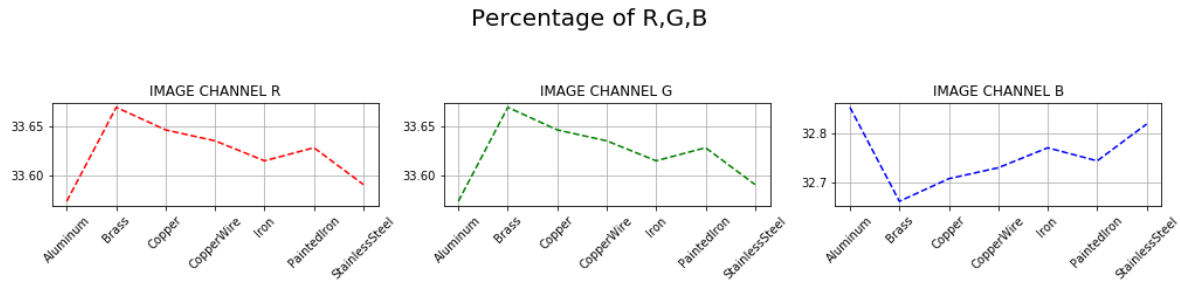
```
<Figure size 432x288 with 0 Axes>
```



In [55]:

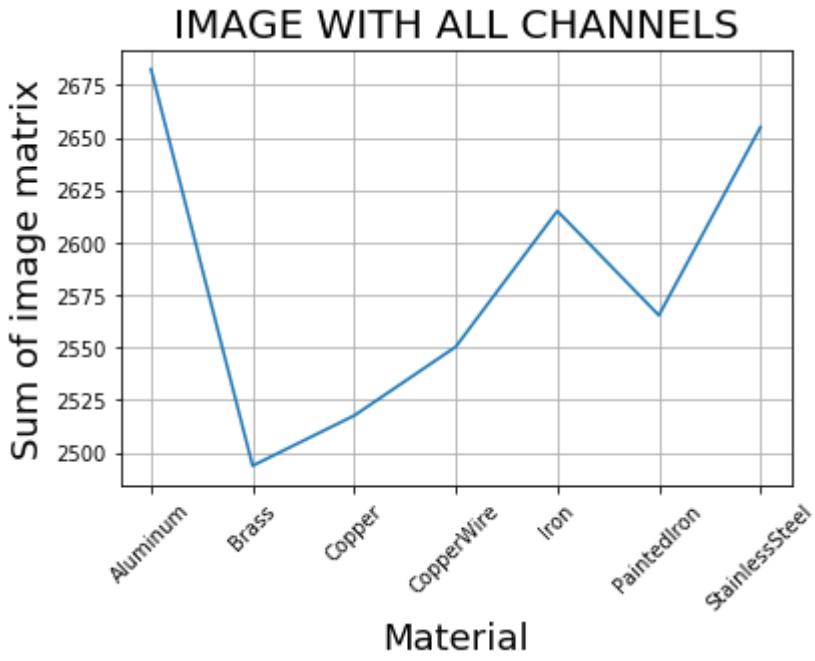
```
# Pèrcentage of R,G,B
plt.figure(1)
plt.figure(figsize=(17, 4))
plt.tight_layout()
plt.subplot(231)
plt.title('IMAGE CHANNEL R')
plt.xticks(rotation=45)
plt.grid(True)
plt.plot(df_plot1.perc_R, 'r--')

plt.subplot(232)
plt.title('IMAGE CHANNEL G')
plt.xticks(rotation=45)
plt.grid(True)
plt.plot(df_plot1.perc_G,  'g--')

plt.subplot(233)
plt.title('IMAGE CHANNEL B')
plt.xticks(rotation=45)
plt.plot(df_plot1.perc_B,  'b--')
plt.grid(True)

plt.suptitle('Percentage of R,G,B',fontsize=20,y=1.08)
#plt.tight_layout()
plt.subplots_adjust(top=0.8)
plt.savefig(folder+'_Percentage_RGB.png', bbox_inches='tight', pad_inches=0.0)
plt.show()
```

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



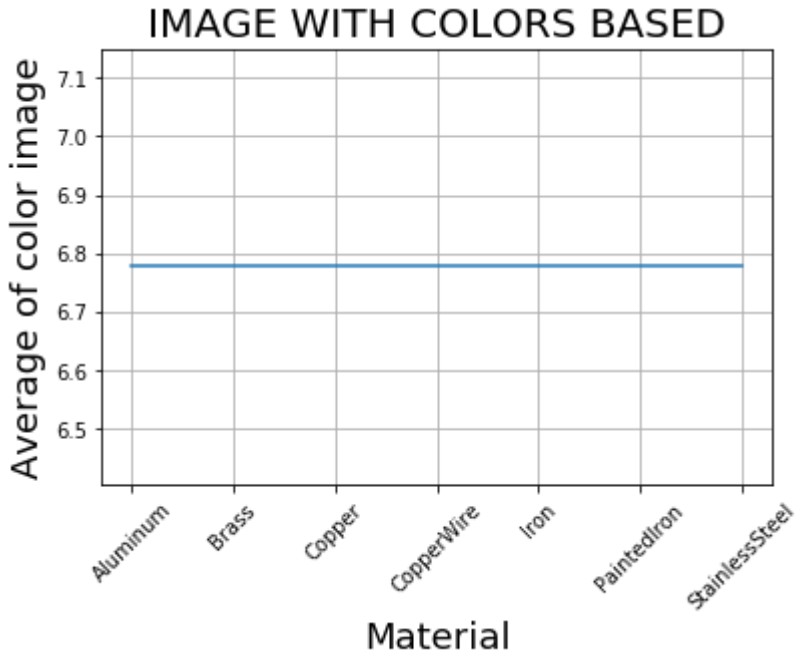
In [56]:

```
# 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.savefig(folder+'_Sum_all_channels.png', bbox_inches='tight', pad_inches=0.0)
plt.show()
```



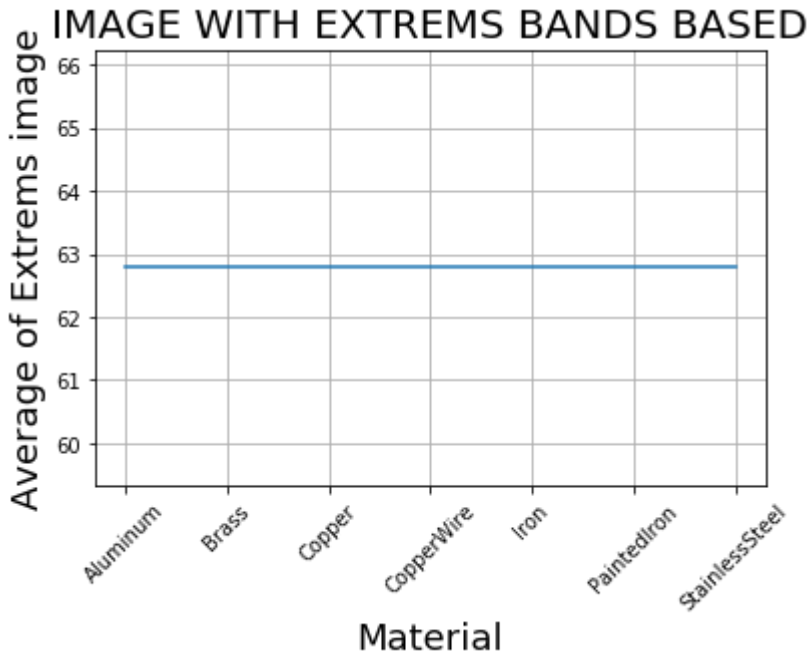
In [57]:

```
# Plot based on color
plt.plot(df_plot1.Color_dec/1000)
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 [58]:

```
# 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.savefig(folder+'_color_based.png', bbox_inches='tight', pad_inches=0.0)
plt.show()
```



Create Histograms

https://www.cambridgeincolour.com/pt-br/tutoriais/histograms1.htm (https://www.cambridgeincolour.com/pt-br/tutoriais/histograms1.htm)

https://www.cambridgeincolour.com/pt-br/tutoriais/image-noise.htm (https://www.cambridgeincolour.com/pt-br/tutoriais/image-noise.htm)

http://www2.ic.uff.br/~aconci/aula-2-2015-AI.pdf (http://www2.ic.uff.br/~aconci/aula-2-2015-AI.pdf)

https://www.ic.unicamp.br/~ra144681/misc/files/ApostilaProcDeImagesParteI.pdf (https://www.ic.unicamp.br/~ra144681/misc/files/ApostilaProcDeImagesParteI.pdf)

histograma, também conhecido como distribuição de frequências ou diagrama das frequências, é a representação gráfica, em colunas (retângulos), de um conjunto de dados previamente tabulado e dividido em classes uniformes.

Histogramas:

O histograma de uma imagem cinza e uma funcao discreta h(I) (vetor) que produz o numero de ocorrencias de cada nıvel de cinza na imagem. O histograma normalizado h(I)/|DI | representa a distribuicao de probabilidade dos valores dos pixels.

Imagens claras possuem histogramas com altas concentracoes de pixels de alto brilho. Imagens escuras possuem histogramas com altas concentracoes de pixels de baixo brilho. O contraste maior esta associado a um grau maior de dispersao do histograma.

No caso de imagens multiespectrais, cada banda e´ requantizada em um certo numero de intervalos, de forma que o espaco de caracterısticas Zk ´e dividido em hipercubos (bins do histograma). A contagem de cores em cada bin ´e usada no c´alculo do histograma. Assim, para cada bin, precisamos analisar os n´ıveis de cinza das 3 bandas da imagem colorida (RGB).

Entendendo Histogramas:

O histograma mostra a frequencia dos valores de brilho da imagem, ou seja, a quantidade de luz presente na imagem.

In [59]:

```
list_of_images
```

Out[59]:

```
['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',
 'Brass_4.tif',
 'Brass_5.tif',
 'Brass_6.tif',
 'CopperWire_1.tif',
 'CopperWire_2.tif',
 'CopperWire_3.tif',
 'CopperWire_4.tif',
 'CopperWire_5.tif',
 'CopperWire_6.tif',
 'CopperWire_7.tif',
 'CopperWire_8.tif',
 'Copper_1.tif',
 'Copper_2.tif',
 'Copper_3.tif',
 'Copper_4.tif',
 'Iron_1.tif',
 'Iron_2.tif',
 'Iron_3.tif',
 'Iron_4.tif',
 'PaintedIron_1.tif',
 'PaintedIron_2.tif',
 'PaintedIron_3.tif',
 'PaintedIron_4.tif',
 'PaintedIron_5.tif',
 'PaintedIron_6.tif',
 'PaintedIron_7.tif',
 'StainlessSteel_1.tif',
 'StainlessSteel_2.tif',
 'StainlessSteel_3.tif',
 'StainlessSteel_4.tif',
 'StainlessSteel_5.tif',
 'StainlessSteel_6.tif',
 'StainlessSteel_7.tif',
 'StainlessSteel_8.tif',
 'StainlessSteel_9.tif']
```

In [60]:

```
# Delete values from list - Bad image names
def remove_values_from_list(list_values, mask):
    list_new = list()
    for list_value in list_values:
        if(mask not in list_value):
            print(list_value)
            list_new.append(list_value)
    return list_new
```

In [61]:

```
# Remove from list names with 'MASK'
new_list = remove_values_from_list(list_of_images, 'MASK')
```

```
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
Brass_4.tif
Brass_5.tif
Brass_6.tif
CopperWire_1.tif
CopperWire_2.tif
CopperWire_3.tif
CopperWire_4.tif
CopperWire_5.tif
CopperWire_6.tif
CopperWire_7.tif
CopperWire_8.tif
Copper_1.tif
Copper_2.tif
Copper_3.tif
Copper_4.tif
Iron_1.tif
Iron_2.tif
Iron_3.tif
Iron_4.tif
PaintedIron_1.tif
PaintedIron_2.tif
PaintedIron_3.tif
PaintedIron_4.tif
PaintedIron_5.tif
PaintedIron_6.tif
PaintedIron_7.tif
StainlessSteel_1.tif
StainlessSteel_2.tif
StainlessSteel_3.tif
StainlessSteel_4.tif
StainlessSteel_5.tif
StainlessSteel_6.tif
StainlessSteel_7.tif
StainlessSteel_8.tif
StainlessSteel_9.tif
```

In [62]:

```
# Remove from list names with 'Enh'
new_list = remove_values_from_list(new_list, 'Enh')
```

```
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
Brass_4.tif
Brass_5.tif
Brass_6.tif
CopperWire_1.tif
CopperWire_2.tif
CopperWire_3.tif
CopperWire_4.tif
CopperWire_5.tif
CopperWire_6.tif
CopperWire_7.tif
CopperWire_8.tif
Copper_1.tif
Copper_2.tif
Copper_3.tif
Copper_4.tif
Iron_1.tif
Iron_2.tif
Iron_3.tif
Iron_4.tif
PaintedIron_1.tif
PaintedIron_2.tif
PaintedIron_3.tif
PaintedIron_4.tif
PaintedIron_5.tif
PaintedIron_6.tif
PaintedIron_7.tif
StainlessSteel_1.tif
StainlessSteel_2.tif
StainlessSteel_3.tif
StainlessSteel_4.tif
StainlessSteel_5.tif
StainlessSteel_6.tif
StainlessSteel_7.tif
StainlessSteel_8.tif
StainlessSteel_9.tif
```

In [63]:

```
list_of_images = new_list
```

In [64]:

```
path = mypath + '/' + folder + '/'
path
```

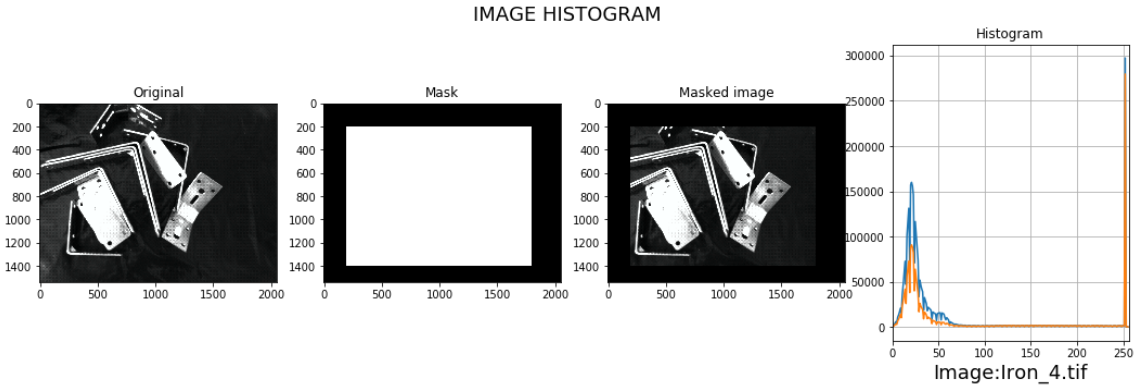
Out[64]:

```
'C:\\Users\\manuel.robalinho\\Google Drive\\UPT_Portucalense\\Trabalho final
\\Classificacao_Sucata\\Jupyter_Notebook/imagedata06/'
```

In [65]:

```
# HISTOGRAMS
# Print Histograms for all folder images
# list_of_images has all the name files

for x in list_of_images:
    print('Cv2 Histogram for File:', x)
    print_cv_hist(path, x)
```

```
Cv2 Histogram for File: Iron_4.tif
Cv2 Hist from file: C:\Users\manuel.robalinho\Google Drive\UPT_Portucalens
e\Trabalho final\Classificacao_Sucata\Jupyter_Notebook/imagedata06/Iron_4.
tif
```




```
Cv2 Histogram for File: PaintedIron_1.tif
Cv2 Hist from file: C:\Users\manuel.robalinho\Google Drive\UPT_Portucalens
```

In [66]:

```
# HISTOGRAMS
# Print Histograms for all folder images
# list_of_images has all the name files

for x in list_of_images:
    print('Matplot Histogram for File:', x)
    print_matplot_hist(path, x)
```

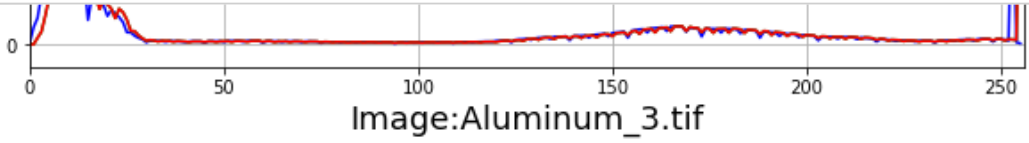
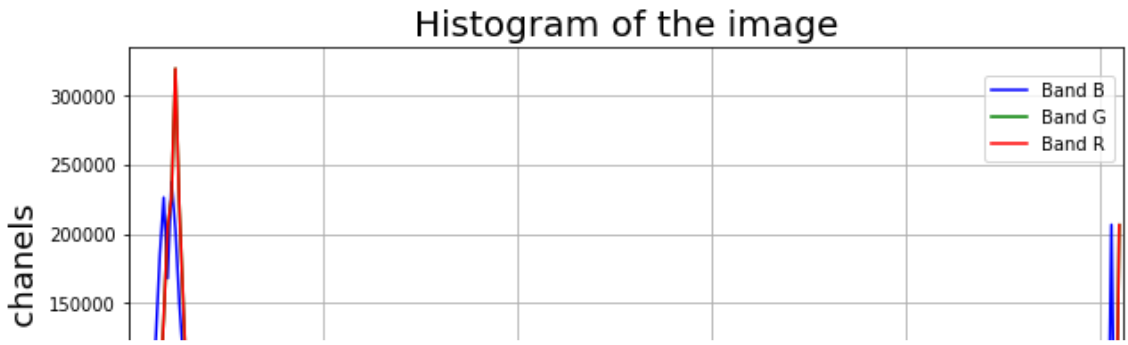


Image:Aluminum_3.tif

```
Matplot Histogram for File: Aluminum_4.tif
Matplot Hist from file: C:\Users\manuel.robalinho\Google Drive\UPT_Portuca
lense\Trabalho final\Classificacao_Sucata\Jupyter_Notebook/imagedata06/Alu
minum_4.tif
```



In [67]:

```
print('Finished')
```

```
Finished
```

In [ ]: