In [1]:

```
import PIL
from PIL import Image
import time

import pycuda.driver as cuda
import pycuda.autoinit
from pycuda.compiler import SourceModule
import numpy

def blackWhite(inPath , outPath , mode = "luminosity",log = 0):

    if log == 1 :
        print ("----------> SERIAL CONVERSION")
    totalT0 = time.time()

    im = Image.open(inPath)
    px = numpy.array(im)

    getDataT1 = time.time()

    print ("-----> Opening path :" , inPath)

    processT0 =  time.time()
    for x in range(im.size[1]):
        for y in range(im.size[0]):

            r = px[x][y][0]
            g = px[x][y][1]
            b = px[x][y][2]
            if mode == "luminosity" :
                val =  int(0.21 *float(r)  + 0.71*float(g)  + 0.07 * f

            else :
                val = int((r +g + b) /3)

            px[x][y][0] = val
            px[x][y][1] = val
            px[x][y][2] = val

    processT1= time.time()
    #px = numpy.array(im.getdata())
    im = Image.fromarray(px)
    im.save(outPath)

    print ("-----> Saving path :" , outPath)
    totalT1 = time.time()

    if log == 1 :
        print ("Image size : ",im.size)
        print ("get and convert Image data  : " ,getDataT1-totalT0 )
        print ("Processing data : " , processT1 - processT0 )
        print ("Save image time : " , totalT1-processT1)
        print ("total  Execution time : " ,totalT1-totalT0 )

def CudablackWhite(inPath , outPath , mode = "luminosity" , log = 0):

    if log == 1 :
        print ("----------> CUDA CONVERSION")

    totalT0 = time.time()

    im = Image.open(inPath)
    px = numpy.array(im)
    px = px.astype(numpy.float32)

    getAndConvertT1 = time.time()

    allocT0 = time.time()
    d_px = cuda.mem_alloc(px.nbytes)
    cuda.memcpy_htod(d_px, px)

    allocT1 = time.time()

    #Kernel declaration
    kernelT0 = time.time()

    #Kernel grid and block size
    BLOCK_SIZE = 1024
    block = (1024,1,1)
    checkSize = numpy.int32(im.size[0]*im.size[1])
    grid = (int(im.size[0]*im.size[1]/BLOCK_SIZE)+1,1,1)

    #Kernel text
    kernel = """

    __global__ void bw( float *inIm, int check ){

        int idx = (threadIdx.x ) + blockDim.x * blockIdx.x ;

        if(idx *3 < check*3)
        {
        int val = 0.21 *inIm[idx*3] + 0.71*inIm[idx*3+1] + 0.07 * inIm

        inIm[idx*3]= val;
        inIm[idx*3+1]= val;
        inIm[idx*3+2]= val;
        }
    }
    """
    #Compile and get kernel function
    mod = SourceModule(kernel)
    func = mod.get_function("bw")
    func(d_px,checkSize, block=block,grid = grid)

    kernelT1 = time.time()
    #Get back data from gpu
    backDataT0 = time.time()

    bwPx = numpy.empty_like(px)
    cuda.memcpy_dtoh(bwPx, d_px)
    bwPx = (numpy.uint8(bwPx))

    backDataT1 = time.time()

    #Save image
    storeImageT0 = time.time()
    pil_im = Image.fromarray(bwPx,mode ="RGB")

    pil_im.save(outPath)
    print ("-----> Saving path :" , outPath)

    totalT1 = time.time()

    getAndConvertTime = getAndConvertT1 - totalT0
    allocTime = allocT1 - allocT0
    kernelTime = kernelT1 - kernelT0
    backDataTime = backDataT1 - backDataT0
    storeImageTime =totalT1 - storeImageT0
    totalTime = totalT1-totalT0

    if log == 1 :
        print ("Image size : ",im.size)
        print ("get and convert Image data to gpu ready : " ,getAndConv
        print ("allocate mem to gpu: " , allocTime )
        print ("Kernel execution time : " , kernelTime)
        print ("Get data from gpu and convert : " , backDataTime)
        print ("Save image time : " , storeImageTime)
        print ("total  Execution time : " ,totalTime )
```

実行

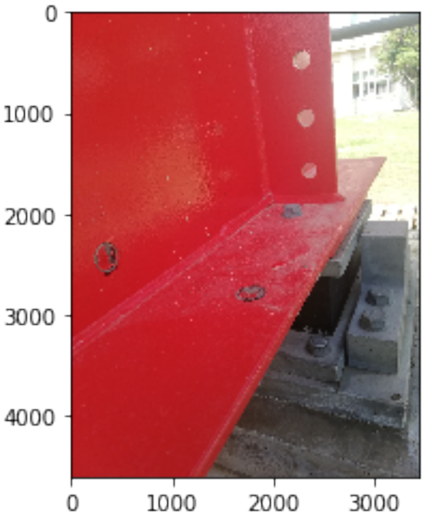
In [2]:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from matplotlib import rcParams
```

大きいサイズ[3456×4608]の画像の場合

In [3]:

```
img=mpimg.imread('test.jpg')
imgplot = plt.imshow(img)
plt.show()
```



In [4]:

```
inPath = "test.jpg"
outPath = "test_cpuout.jpg"
blackWhite(inPath , outPath , mode = "luminosity",log = 1)
```

```
----------> SERIAL CONVERSION
-----> Opening path : test.jpg
-----> Saving path : test_cpuout.jpg
Image size :  (3456, 4608)
get and convert Image data  :  0.11483478546142578
Processing data :  44.07874512672424
Save image time :  0.07214546203613281
total  Execution time :  44.26579523086548
```

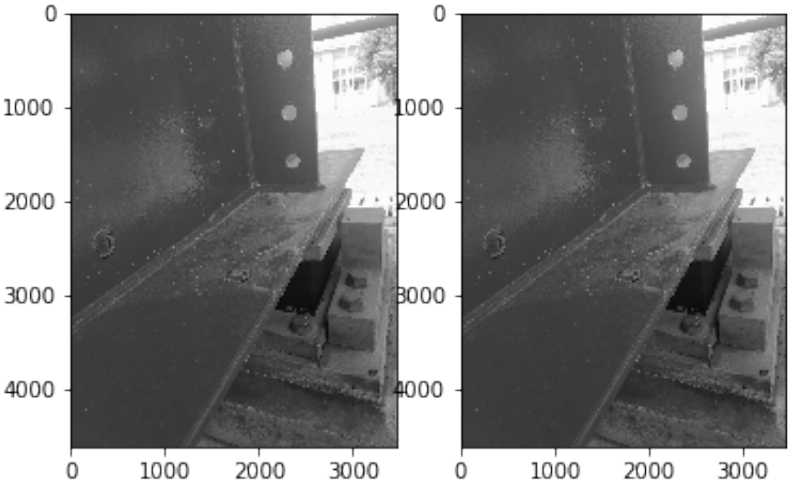
In [5]:

```
inPath = "test.jpg"
outPath = "test_gpuout.jpg"
CudablackWhite(inPath , outPath , mode = "luminosity" , log = 1)
```

```
----------> CUDA CONVERSION
-----> Saving path : test_gpuout.jpg
Image size :  (3456, 4608)
get and convert Image data to gpu ready :  0.14509034156799316
allocate mem to gpu:  0.036179304122924805
Kernel execution time :  0.010811805725097656
Get data from gpu and convert :  0.061189889907836914
Save image time :  0.07258486747741699
total  Execution time :  0.32585740089416504
```

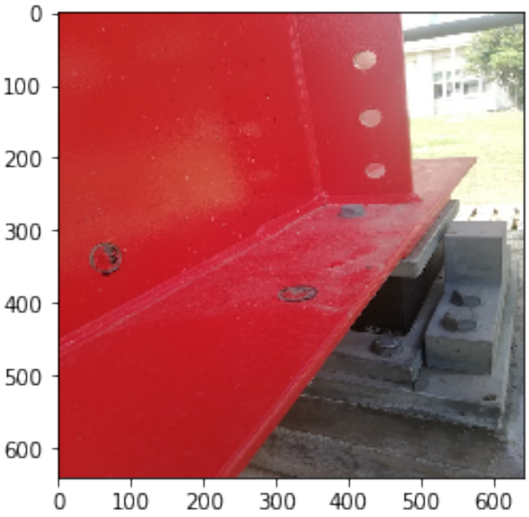
In [6]:
```
img1 = mpimg.imread('test_cpuout.jpg')
img2 = mpimg.imread('test_gpuout.jpg')

fig, ax = plt.subplots(1,2)
ax[0].imshow(img1);
ax[1].imshow(img2);
```



小さいサイズ[640×640]の画像の場合

In [7]:
```
img=mpimg.imread('test2.jpg')
imgplot = plt.imshow(img)
plt.show()
```



In [8]:

```
inPath = "test2.jpg"
outPath = "test2_cpuout.jpg"
blackWhite(inPath , outPath , mode = "luminosity",log = 1)
```

```
----------> SERIAL CONVERSION
-----> Opening path : test2.jpg
-----> Saving path : test2_cpuout.jpg
Image size :  (640, 640)
get and convert Image data  :  0.006237030029296875
Processing data :  1.1614701747894287
Save image time :  0.002123117446899414
total  Execution time :  1.1698954105377197
```

In [9]:

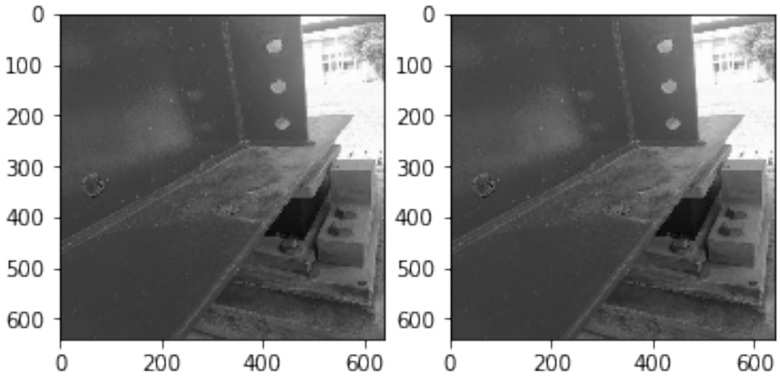
```
inPath = "test2.jpg"
outPath = "test2_gpuout.jpg"
CudablackWhite(inPath , outPath , mode = "luminosity" , log = 1)
```

```
----------> CUDA CONVERSION
-----> Saving path : test2_gpuout.jpg
Image size :  (640, 640)
get and convert Image data to gpu ready :  0.006748676300048828
allocate mem to gpu:  0.005264759063720703
Kernel execution time :  0.0009810924530029297
Get data from gpu and convert :  0.0015521049499511719
Save image time :  0.0018432140350341797
total  Execution time :  0.016391277313232422
```

In [10]:

```
img1 = mpimg.imread('test2_cpuout.jpg')
img2 = mpimg.imread('test2_gpuout.jpg')

fig, ax = plt.subplots(1,2)
ax[0].imshow(img1);
ax[1].imshow(img2);
```



In [ ]: