Normalized Difference Index for color vegetation with UAV-based imaging

Leonardo Andrés Ángeles Daza
Maestría en Tecnología de Cómputo
Centro de Innovación y Desarrollo Tecnológico en Cómputo
Mexico City, Mexico
0000-0002-4404-3171

***Abstract*—This exercise is a first glare to computer vision, where I practice basic operations and maintain admissible pixel values in an adequate quantization range as I use a normalized difference (NDI) index into an image obtained with a UAV of the *Escuela Superior de Computación* (ESCOM) to identify plant biomass versus soil and residue background using python for the image processing.**

***Index Terms*—python, computer vision, vegetation**

I. Introduction

Crop parameters, like biomass, are frequently used to assess crop health status, nutrient supply and effects of agricultural management practices. [3] An accurate vegetation index is required to identify plant biomass versus soil and residue backgrounds for automated remote sensing. The normalized difference vegetation index (NDI) has been proposed in [1] to separate plants from soil and residue background images.

$$NDI = \frac{G - R}{G + R} \tag{1}$$

The topic of this paper is to create a binary image which accurately separates plant regions from background as in [2].

II. Materials and methods

A. Test Site and Image to Process

The study site is based at the *Escuela Superior de Computación* (ESCOM) (19.5042° N, 99.1467° W), located in Mexico City, Mexico

Fig. 1. Original RGB image of ESCOM.

Figure 1 is originally an `uint8` RGB image type with the dimensions 1044 × 1155. Figure 1 has a maximum value of 255 and a minimum value of 0 in the red and blue channels, in the green channel the minimum value is 7.

B. Image Processing

The first thing we did was to split the red, green and blue channels and save them in independent nested lists of `float32` type. Then to maintain admissible pixel values in an adequate quantization range we use a weighted sum with the function `cv2.addWeighted()` which has the form

$$dst = \alpha src1 + \beta src2 + \gamma \tag{2}$$

In (1) the numerator is the addition of the red and green channel where both channels have the same weight, so $\alpha = \beta$, $scr1 = g$ where g is the value of the green channel in that pixel, $scr2 = r$ where r is the value of the red channel in that pixel and $\gamma = 0$ as we don't want to add any number to the operation.

In (1) the denominator is the subtraction of the red channel to the green channel the reason we convert the unit type of the channels was to be able to obtain negative values as the possible range of NDI is from -1.0 to 1.0. [4]

C. Threshold Method

Thresholding to obtain binary NDI index images was performed using the method of Otsu. Otsu's method is based on an analysis of the histogram of the tonal image resulting from the initial vegetative index mage calculation.

Thankfully OpenCV has a function where we can apply a python function which is `cv2.threshold()`, but before we can use it we have to rescale the values contained in the NDI image to be an `uint8` type once again using:

$$\frac{255\,(NDI - NDI_{minvalue})}{NDI_{maxvalue}]} \tag{3}$$

Once the values are scaled the NDI values are admissible for the threshold method using `cv2.THRESH_BINARY+cv2.THRESH_OTSU` as the *thresholdingTechnique* argument.

III. Results

After doing (1) and rescaling to `int` values, the gray scale image obtained is shown in 2 which reassembles to the image obtained by [2] shown in 3. Where the zones with healthy vegetation are shown with more illuminated pixels.

Fig. 2. NDI image converted to `uint8`.

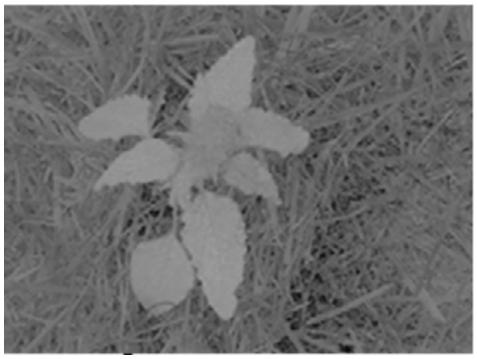


Fig. 3. NDI tonal image obtained by [2].

Finally, after thresholding with the Otsu's method the binary image obtained is shown in 4

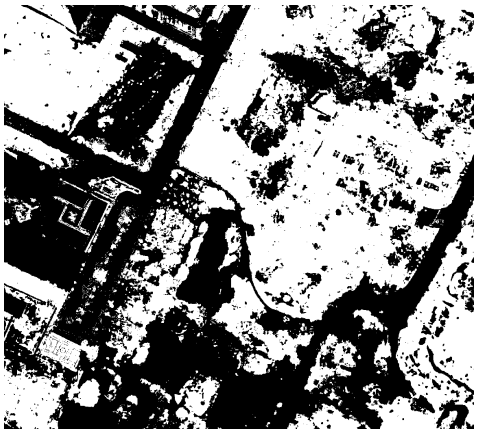


Fig. 4. NDI+Otsu binary image.

IV. Conclusion

The image vegetative index is an important critical step for machine vision identification of vegetation, in this exercise specifically for remote sensing. Its primary purpose is to provide the boundaries of the vegetation regions in the images obtained by the UAV.

This first exercise of computer vision was very illustrative to show how images can contain a lot of information and how we can access that information by processing images. This work only exemplifies one of the many applications there are for computer vision to work with.

References

[1] A.J. Perez, F. Lopez, J.V. Benlloch, and S. Christensen, "Color and shape analysis techniques for weed detection in cereal fields," Computer and Electronics in Agriculture, Volume 25, Issue 3, pp. 197-212, February 2000.

[2] George E. Meyer, and João Camargo Neto. "Verification of color vegetation indices for automated crop imaging applications," Computer and Electronics in Agriculture, Volume 63, pp. 282-29, 2008

[3] J. Bendig, et al., "Combining UAV-based plant height from crop surface models, visible, and near infrared vegetation indices for biomass monitoring in barley," International Journal of Applied Earth Observation and Geoinformation, Volume 39, pp. 79-87, 2015

[4] E.R. Hunt, M. Cavigelli, C.T. Daughtry, J. McMurtrey, and S.L. Walthall, "Evaluation of digital photography from model aircraft for remote sensing of crop biomass," Precision Agriculture, Volume 6, pp. 359-378, August 2005.

1 Miniproyecto 1: Índice NDI

La primer tarea será calcular el índice NDI que se utiliza como indicador de la salud de las plantas. Como imagen de entrada utilizaremos un mosaico de la ESCOM generado con vuelos aéreos. El objetivo de esta practica es que apliques las operaciones básicas que hemos visto y que seas capaz de mantener los valores de los pixeles en un rango de cuantización adecuado o que selecciones el tipo de dato necesario.

De acuerdo con [1], el NDI se calcula:

$$NDI = \frac{G - R}{G + R} \tag{1}$$

Nota, en el artículo de Meyer[1] la división está al revés, al parecer es un error del artículo.

Adicionalmente, puedes intentar calcular un indice más elaborado como es el MGRVI o el RGBVI [2].

1.0.1 Referencias

[1] Meyer, George E., and João Camargo Neto. Verification of color vegetation indices for automated crop imaging applications. Computers and electronics in agriculture 63.2 (2008): 282-293

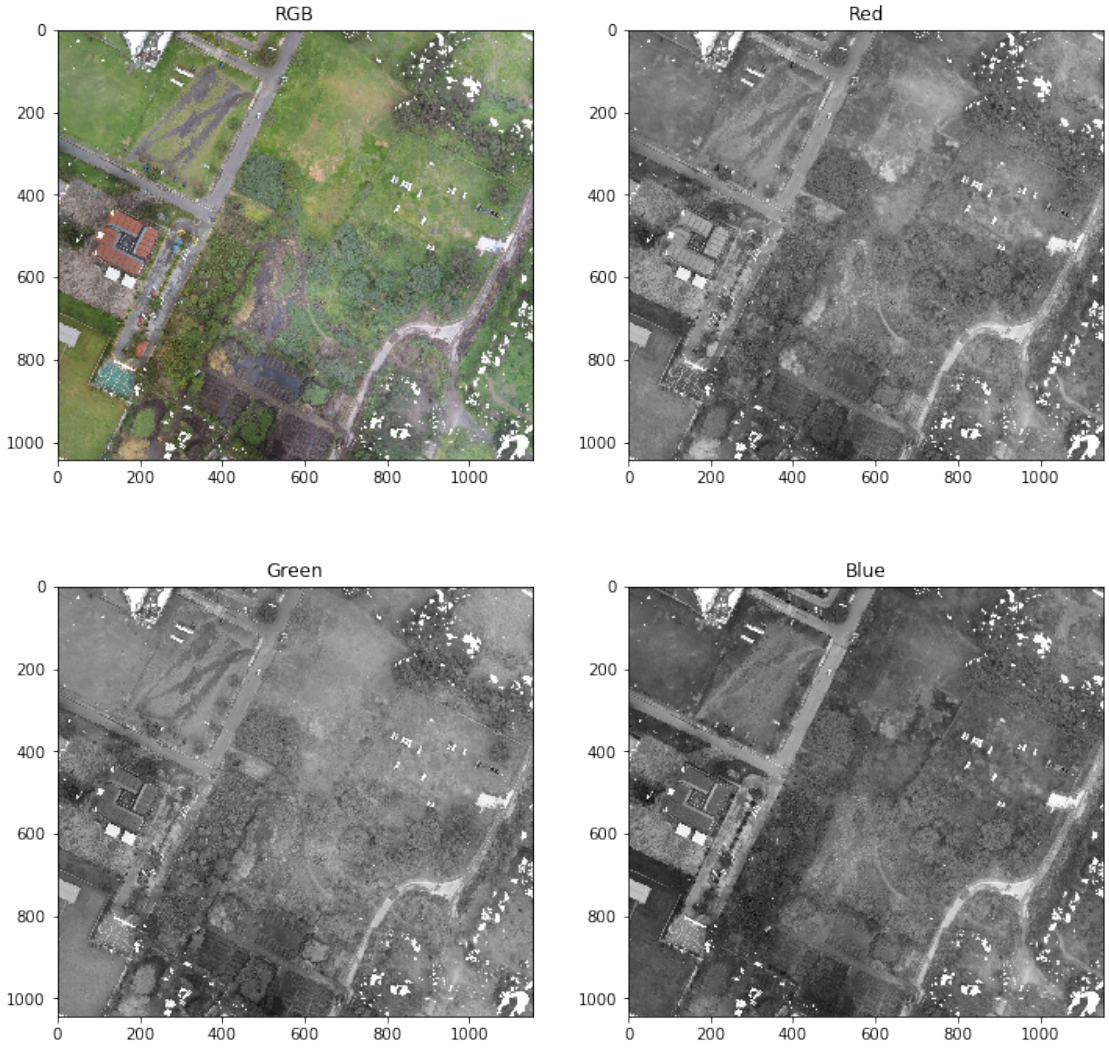
[2] Bendig, Juliane, et al. Combining UAV-based plant height from crop surface models, visible, and near infrared vegetation indices for biomass monitoring in barley. International Journal of Applied Earth Observation and Geoinformation 39 (2015): 79-87.

[1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import cv2

%matplotlib inline
```

[2]:
```
# Read Image
image = cv2.imread('ESCOM2_small.jpg')
print('Image type:', image.dtype, 'with dimensions:', image.shape)

# Obtaining rgb channels
b,g,r = cv2.split(image)

# Show channels
plt.figure(figsize=(12, 12))

plt.subplot(2,2,1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('RGB')

plt.subplot(2,2,2)
```

```
plt.imshow(r, cmap='gray')
plt.title('Red')

plt.subplot(2,2,3)
plt.imshow(g, cmap='gray')
plt.title('Green')

plt.subplot(2,2,4)
plt.imshow(b, cmap='gray')
plt.title('Blue')
plt.show()
```

```
('Image type:', dtype('uint8'), 'with dimensions:', (1044L, 1155L, 3L))
```



[3]:
```
print("\nMaximum value in the red channel is: %0.1f" % (np.amax(r)))
print("Minimum value in the red channel is: %0.1f" % (np.amin(r)))

print("\nMaximum value in the green channel is: %0.1f" % (np.amax(g)))
print("Minimum value in the green channel is: %0.1f" % (np.amin(g)))

print("\nMaximum value in the blue channel is: %0.1f" % (np.amax(b)))
print("Minimum value in the blue channel is: %0.1f" % (np.amin(b)))
```

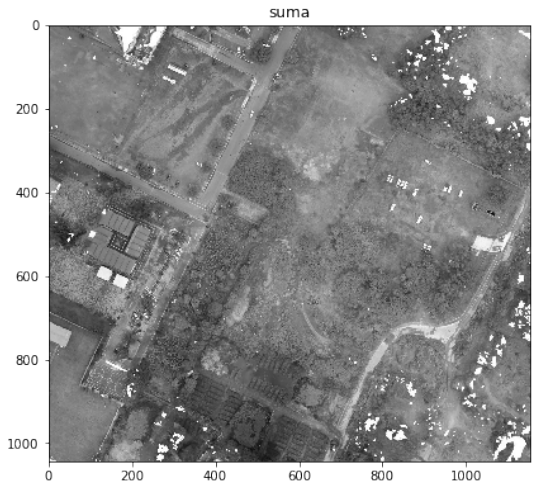
```
Maximum value in the red channel is: 255.0
Minimum value in the red channel is: 0.0

Maximum value in the green channel is: 255.0
Minimum value in the green channel is: 7.0

Maximum value in the blue channel is: 255.0
Minimum value in the blue channel is: 0.0
```

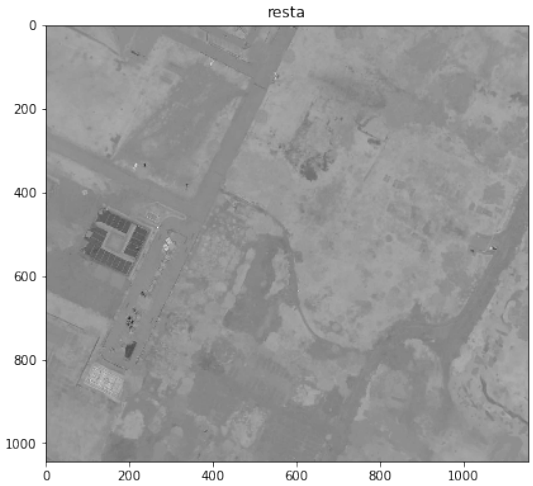
[4]:
```
alpha = 1

r_float=r.astype('float32')
g_float=g.astype('float32')
b_float=b.astype('float32')

# sumamos y mentenemos en el rango
suma = cv2.addWeighted(g_float, alpha, r_float, alpha, 0)

# restamos y mentenemos en el rango
resta = cv2.addWeighted(g_float, alpha, r_float, -alpha, 0)

# División
NDI = resta/suma

plt.figure(figsize=(15, 15))

plt.subplot(2,2,1)
plt.imshow(suma, cmap='gray')
plt.title('suma')

plt.subplot(2,2,2)
plt.imshow(resta, cmap='gray')
plt.title('resta')


plt.figure(figsize=(15, 15))
plt.subplot(2,2,1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

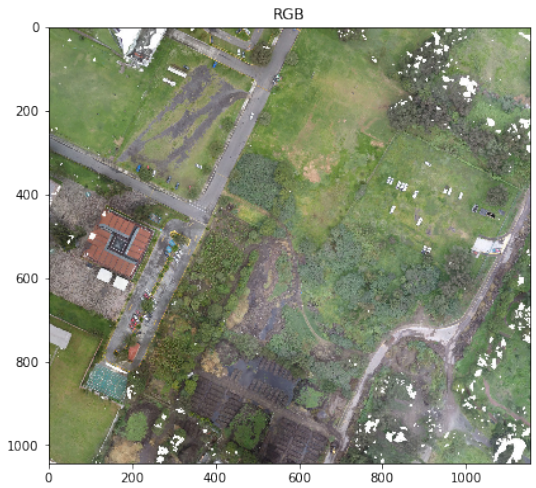
```
plt.title('RGB')

plt.subplot(2,2,2)
plt.imshow(NDI, cmap='gray')
plt.title('NDI')



# convertir a enteros para usar Otsu
NDI_scaled = NDI.copy()
NDI_scaled = NDI_scaled - NDI.min()
NDI_scaled = NDI_scaled * (255/NDI.max())
NDI_int = NDI_scaled.astype('uint8')

cv2.imwrite("TRY.png", NDI_int)

print(suma)
print(resta)

print(np.amin(NDI))
print(np.amax(NDI))
print(NDI)
```

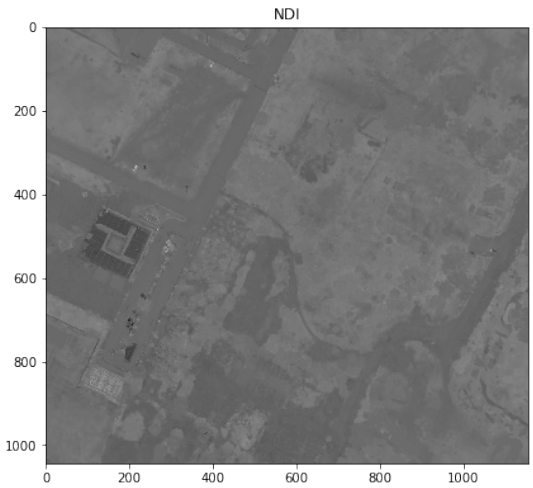
```
[[369. 325. 272. ... 312. 304. 296.]
 [254. 238. 225. ... 308. 300. 292.]
 [244. 237. 233. ... 319. 310. 298.]
 ...
 [508. 509. 507. ... 249. 243. 241.]
 [467. 491. 510. ... 251. 245. 243.]
 [289. 347. 356. ... 253. 247. 243.]]
[[ 15.  15.  18. ...  30.  30.  30.]
 [ 18.  18.  19. ...  30.  30.  30.]
 [ 24.  25.  25. ...  31.  30.  30.]
 ...
 [ -2.  -1.   3. ...   3.   3.   3.]
 [ -9.  -5.   0. ...   3.   3.   3.]
 [-13.  -9.  -4. ...   3.   3.   3.]]
-0.71900827
1.0
[[ 0.0406504   0.04615385  0.06617647 ...  0.09615385  0.09868421
   0.10135135]
 [ 0.07086615  0.07563026  0.08444444 ...  0.09740259  0.1
   0.10273973]
 [ 0.09836066  0.10548523  0.10729614 ...  0.09717868  0.09677419
   0.10067114]
 ...
 [-0.00393701 -0.00196464  0.00591716 ...  0.01204819  0.01234568
   0.01244813]
```

```
[-0.01927195 -0.0101833   0.          ...  0.01195219  0.0122449
  0.01234568]
[-0.0449827  -0.0259366  -0.01123596 ...  0.01185771  0.01214575
  0.01234568]]
```





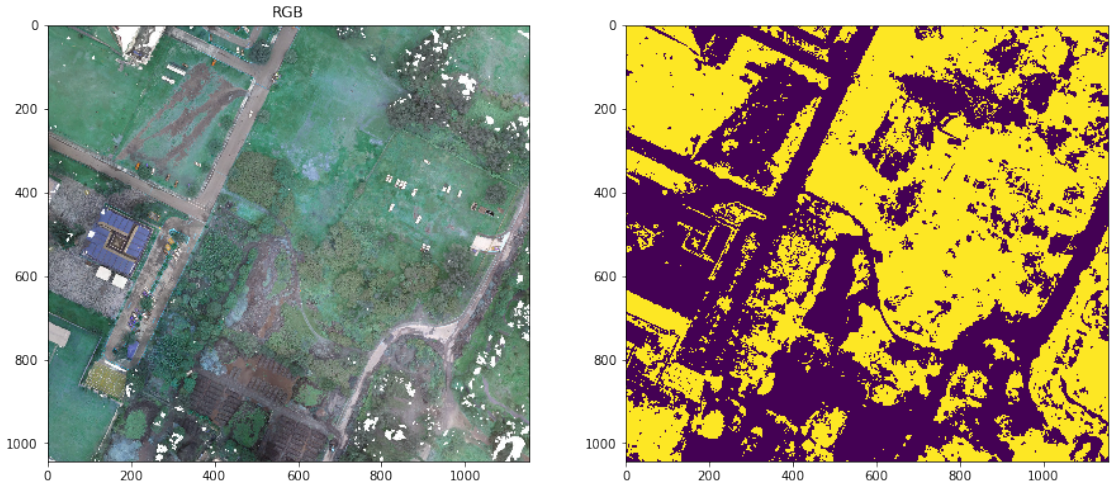




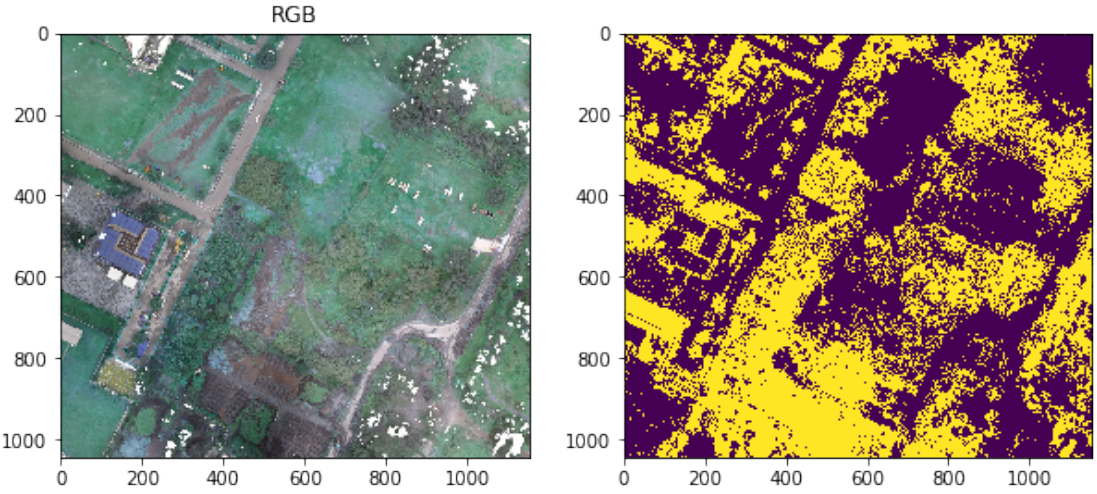
[5]:
```
# Thresholding
ret2,th1 =  cv2.threshold(NDI_int, 0, 255.0, cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Comparación
plt.figure(figsize=(15, 15))

plt.subplot(2,2,1)
plt.imshow(image)
plt.title('RGB')
```

```
plt.subplot(2,2,2)
plt.imshow(th1)

cv2.imwrite("binary.png", th1)
```

[5]: True



Finalmente si todo funcionó debes visualizar algo como lo siguiente:



Ve más allá. ¿Puedes obtener los resultados del paper[2]?