

OpenCV – Spazio di Colore e accesso ai Pixel C++/Python-Part. II

Spazi di Colore

Prima di introdurre la gestione dei spazi di colore e dei pixel di OpenCV, cerchiamo di dare qualche accenno su cos'è uno **Spazio di Colore**.

Uno **Spazio di Colore** non è altro che una mappatura del colore che una periferica è in grado di riprodurre, ovvero una descrizione precisa di un colore e di come dovrebbe essere riprodotto.

Uno Spazio Colore si basa a sua volta su un **Modello di Colore**, che corrisponde ad una *rappresentazione astratta* del colore.

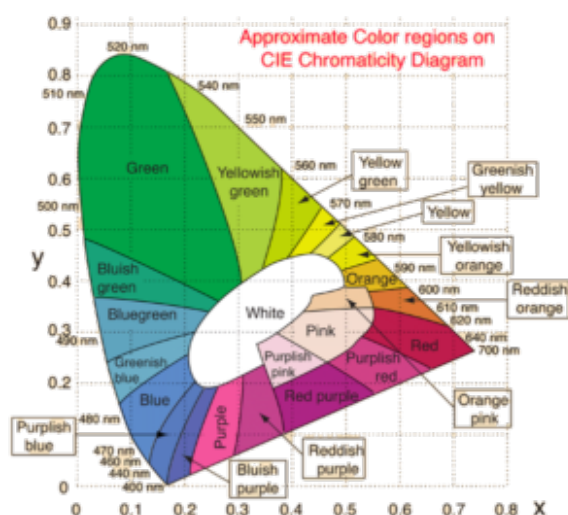


Diagramma di Cromaticità

Ne
l
19
31
la
CIE
Commission
Internat
io

**na
le
De
L'
Ec
la
ir
ag
e
(C
om
mi
ss
io
ne
In
te
rn
az
io
na
le
pe
r
l'
Il
lu
mi
na
zi
on
e)
de
fi
nì
un
o**

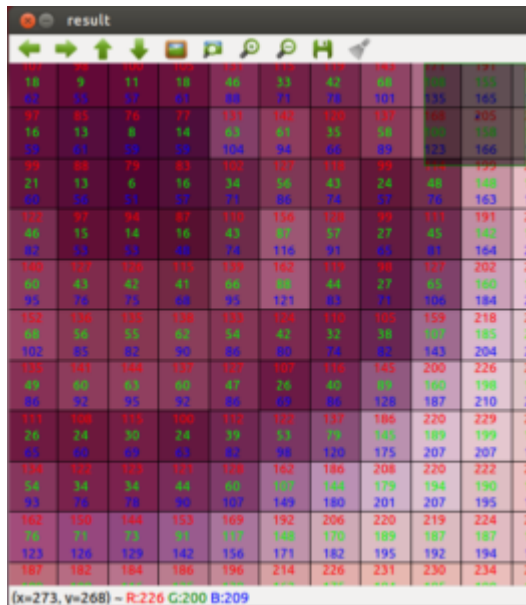
sp
az
io
bi
di
me
ns
io
na
le
ch
ia
ma
to
Di
ag
ra
mm
a
di
Cr
om
at
ic
it
à
ch
e
co
mp
re
nd
ev
a
tu
tt
e

le
ti
nt
e
vi
si
bi
li
da
ll
'o
cc
hi
o
um
an
o.

Il modello CIE del 1931 si basa sull'utilizzo di **3 colori primari** che, opportunamente miscelati tra loro, permettono di ottenere tutti i colori che l'occhio umano può percepire.

Attualmente, la maggior parte dei Spazi colore utilizzati sono basati su 3 colori primari: lo spazio di colore più conosciuto, che corrisponde anche a quello standard con il quale OpenCV elabora l'immagine alla sua apertura, è quello **RGB (Red-Green-Blue)** sviluppato da **Adobe** nel 1998.

Me
nt
re
pe
r
la
Sc
al
a
di
Gr
ig
i
ad
og
ni
pi
xe
l
è
as
so
ci
at
o
un
by
te
,
ci
oè
un
va
lo
re
da
0



Zoom su immagine di Leena in OpenCV

a
25
5,
in
un
'i
mm
ag
in
e
a
co
lo
re
pe
r
og
ni
pi
xe
l
as
so
ci
am
o
3
by
te
,
og
nu
no
co
n
ra
ng

e
di
va
lo
re
tr
a
0
e
25
5.
(P
er
ra
gi
on
i
st
or
ic
he
,
l'
or
di
ne
de
i
by
te
no
n
è
RG
B
ma
BG

R)

.

Come si può notare nell'immagine a destra, ingrandendo su un'immagine aperta a colori in OpenCV si può notare come ogni pixel sia formato dai tre valori RGB discussi precedentemente.

Passiamo al codice

Vediamo come accedere ai singoli pixel di un'immagine in OpenCV sia in C++ che in Python altre operazioni sui canali **RGB**.

Supponiamo di voler aprire un'immagine a colori, quindi in RGB, e di voler stampare a schermo i valori di Red, Green e Blue di un determinato pixel (Ovviamente, prima di scegliere il pixel, verificate che la grandezza dell'immagine presa in input lo permetta).

```
#include <opencv2/opencv.hpp>
#include <iostream> using namespace std;
using namespace cv; int main(int argc, char **argv)
{   Mat   immagineOriginale   =   imread(argv[1]);
if(immagineOriginale.data == nullptr) { cerr << "Error Open
Image" << endl; return(-1); }   namedWindow("Immagine
Originale", CV_WINDOW_AUTOSIZE); imshow("Immagine Originale",
immagineOriginale); cout << "Numero di righe dell'immagine: "
<< immagineOriginale.rows << endl; cout << "Numero di colonne
dell'immagine: " << immagineOriginale.cols << endl; Vec3b
pixel = immagineOriginale.at<Vec3b>(500, 400); cout << "B: "
<< int(pixel.val[0]) << endl; cout << "G: " <<
int(pixel.val[1]) << endl; cout << "R: " << int(pixel.val[2])
<< endl; waitKey(0);
return(0);
}
```

Come si può notare, in **C++** la matrice **Mat** che corrisponde all'immagine aperta è formata da pixel rappresentati dal tipo di OpenCV **Vec3b** che permette di accedere ai valori **BGR** attraverso l'uso di indici.

In **Python** la situazione è simile:

```
import cv2
import sys
image = cv2.imread(sys.argv[1])
if image is None:
    print("Errore apertura immagine")
    sys.exit(-1)
cv2.imshow("Immagine Originale", image)
pixel = image[500, 400]
print("B: ", pixel[0])
print("G: ", pixel[1])
print("R: ", pixel[2])
cv2.waitKey(0)
```

OpenCV ci permette anche di **splittare** i tre canali Red – Green – Blue e di operare su loro singolarmente, come se fossero semplici **Mat** in scala di grigi utilizzando la funzione **split()** sia in C++ che in Python:

```
...
Mat channels[3];
split(immagineOriginale, channels);
...
```

In Python è possibile anche utilizzare una soluzione senza **splittare** l'immagine ma accedendo ai singoli canali.

```
...
B, G, R = cv2.split(image)
...
B = image[:, :, 0]
G = image[:, :, 1]
R = image[:, :, 2]
```

Nel parte III verrà mostrato come si scorrono tutti i pixel della matrice di un'immagine e introdurremo l'**Istogramma**.

Per qualsiasi informazione o ulteriore chiarimento non esitate e commentate!

<https://italiancoders.it/opencv-spazio-di-colore-e-accesso-ai-pixel-c-python-part-ii/>