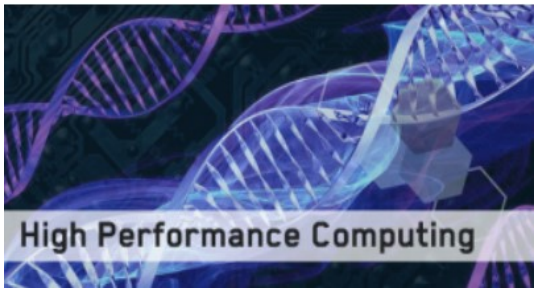


PYTHON

Manual de usuario Filtro de realce de color R, G ó B

VERSIÓN 1.0

Índice

Introducción	2
Configurar entorno	3
Armado del ambiente	4
Ejecución CPU	5
Ejecución GPU	7
Métricas y conclusiones	9

Introducción

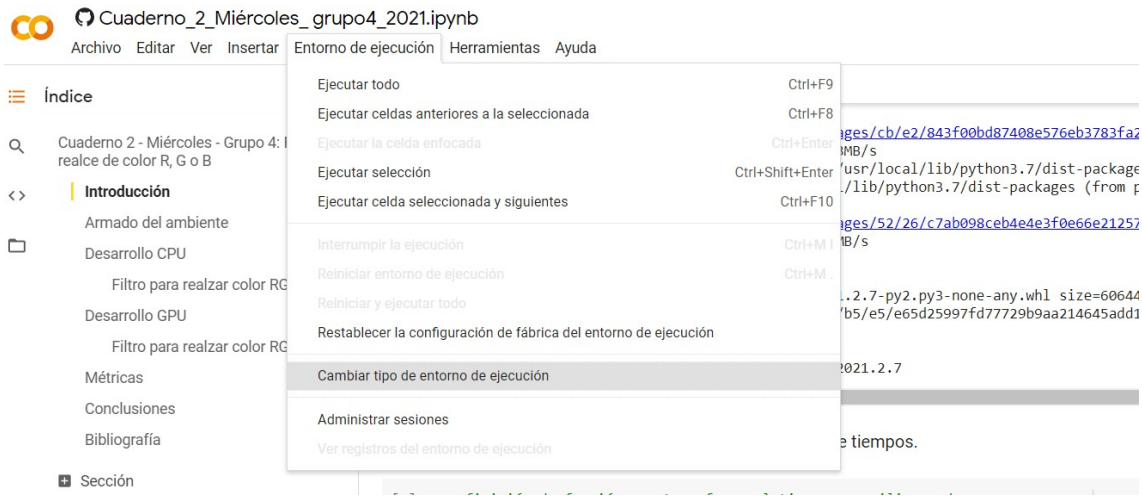
El cuaderno de Colab de Realce de color R, G o B, permite realizar pruebas de dicho filtro, el cual consiste en realzar el color seleccionado (rojo, verde o azul) y al resto de los colores pasarlos a escala de grises. Asimismo se puede realizar la comparativa entre una ejecución con CPU y otra con GPU.

Para poder utilizarlo es necesario ejecutar el cuaderno en Colab. La dirección de github del mismo es:

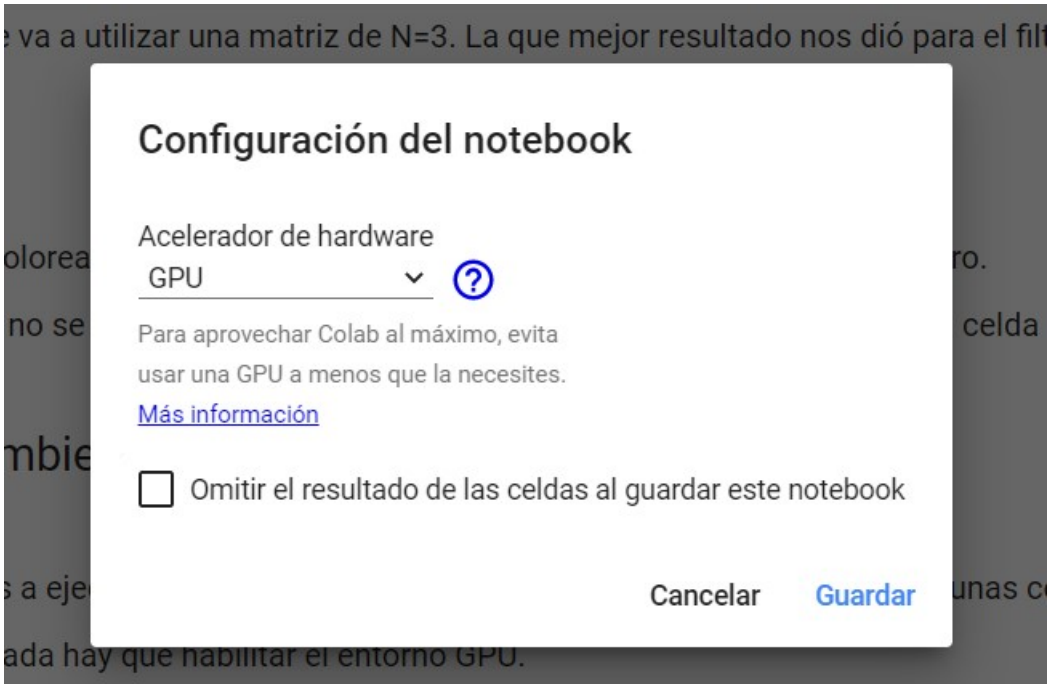
https://github.com/MartinRomano-S/soa-tp2-tp3-grupo4/blob/master/HPC/Cuaderno_2_Mi%C3%A9rcoles__grupo4_2021.ipynb/

Configurar entorno

Al acceder, lo primero que se debe hacer es configurar el entorno, cambiando el entorno de ejecución a Solo GPU.
Primero, cliquear “Entorno de ejecución”.

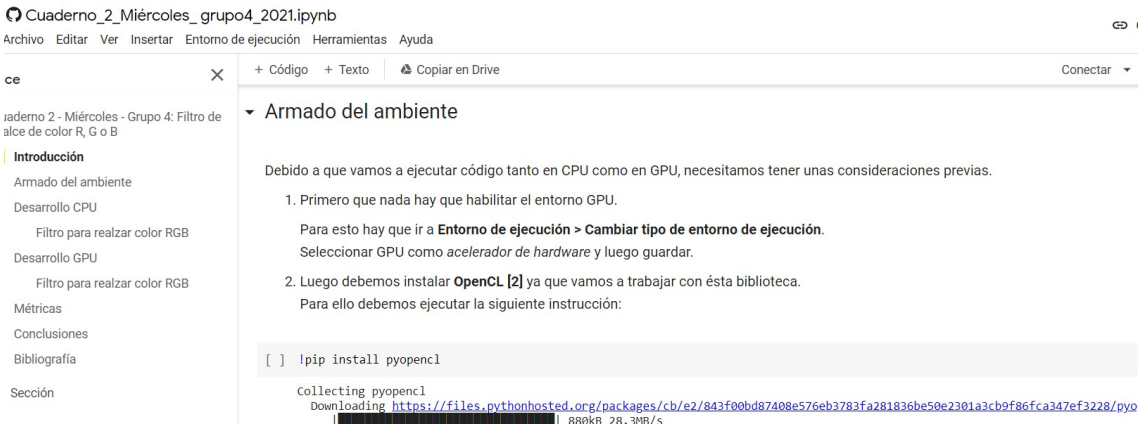


A continuación, se debe cambiar a “Solo GPU” y guardar la configuración.



Armado del ambiente

A continuación, se procede a armar el ambiente. Para ello se debe ejecutar las siguientes líneas:



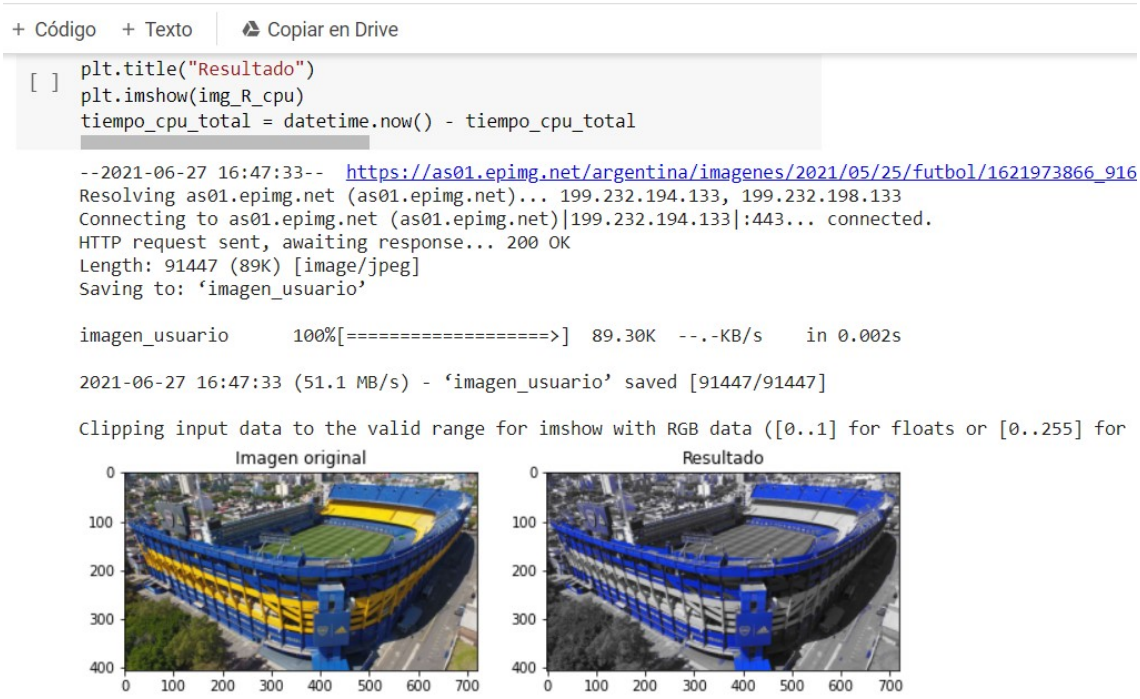
Definimos una función lambda para mostrar las métricas de tiempos.

```
[ ]  # Definición de función que transforma el tiempo en  milisegundos
     tiempo_en_ms = lambda dt:(dt.days * 24 * 60 * 60 + dt.seconds) * 1000 + dt.microseconds / 1000.0
```

Ejecución CPU

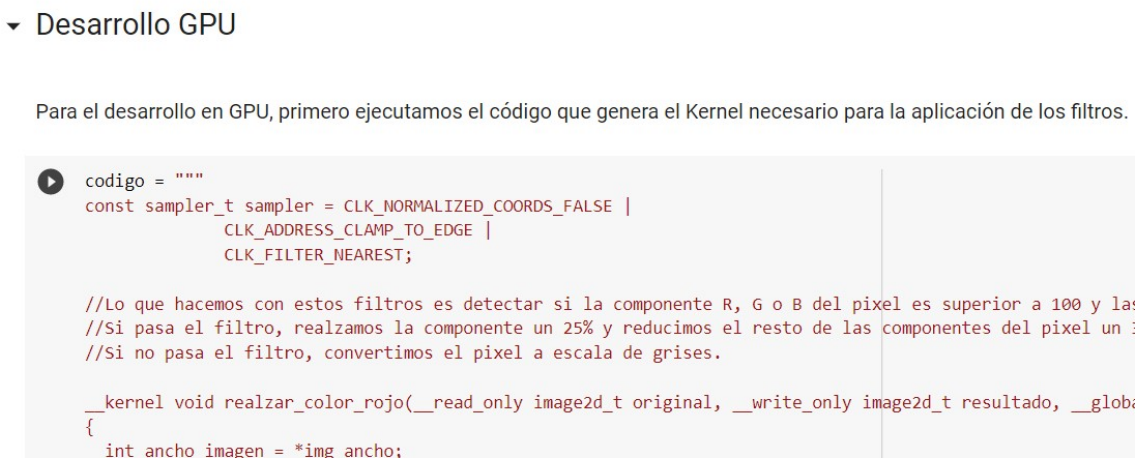
En la opción Desarrollo CPU, al cliquear en el botón de “play”, se ejecutará el filtro de realce R, G o B bajo CPU pudiéndose especificar una imagen y seleccionar el color a realzar.

Al finalizar la ejecución veremos el resultado final en modalidad CPU.



Ejecución GPU

En la opción Desarrollo GPU, se procederá a ejecutar el filtro R, G o B bajo GPU.
Primero se ejecutará un código para poder generar el kernel.



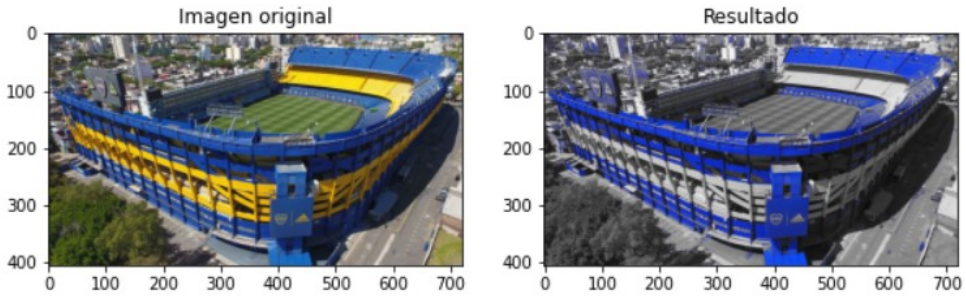
A continuación, se ejecutará el código en el cual se puede especificar una imagen y seleccionar el color a realzar.

Al terminar, se verá el resultado de la imagen.

```
--2021-06-27 16:47:47--  https://as01.epimg.net/argentina/imagenes/2021/05/25/futbol/
Resolving as01.epimg.net (as01.epimg.net)... 199.232.194.133, 199.232.198.133
Connecting to as01.epimg.net (as01.epimg.net)|199.232.194.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 91447 (89K) [image/jpeg]
Saving to: ‘imagen_usuario’

imagen_usuario      100%[===================>]  89.30K  --.-KB/s    in 0.003s

2021-06-27 16:47:47 (28.2 MB/s) - ‘imagen_usuario’ saved [91447/91447]
```



Métricas y conclusiones

Finalmente, se pueden ver las métricas a presionar el botón de “play” en dicha sección para poder comparar los tiempos tanto de CPU como GPU y además más abajo se podrán apreciar las métricas.

▾ Métricas

```
[ ]  print (' _____________________________________________________')
     print ("|{:<15}|{:<20}|{:<20}|".format('Tiempos','Implementación CPU','Implementación GPU'))
     print ('|---------------|--------------------|--------------------|')
     print ("|{:<15}|{:>20}|{:>20}|".format('Tiempo CPU', str(tiempo_en_ms(tiempo_cpu)) + 'ms', tiempo_gpu_gpu_total_string + 'ms'))
     print ("|{:<15}|{:>20}|{:>20}|".format('Tiempo GPU','0.0ms', str(tiempo_en_ms(tiempo_gpu)) + 'ms'))
     print ("|{:<15}|{:>20}|{:>20}|".format('Tiempo Total',  str(tiempo_en_ms(tiempo_cpu_total)) + 'ms', tiempo_gpu_total_string +
     print ('|_______________|____________________|____________________|')
```

```
 _____________________________________________________
|Tiempos        |Implementación CPU  |Implementación GPU  |
|---------------|--------------------|--------------------|
|Tiempo CPU     |          3617.422ms|            222.31ms|
|Tiempo GPU     |             0.0ms|            35.969ms|
|Tiempo Total   |          3674.834ms|           258.279ms|
|_______________|____________________|____________________|
```

▾ Conclusiones

A partir de los resultados obtenidos en las métricas, se ve que el tiempo de ejecución total del programa utilizando la GPU para la imagen propuesta es de tan sólo el 7.02% del tiempo de ejecución total utilizando sólo la CPU. Es decir, aproximadamente 14 veces menos. Ésto nos da una clara demostración de la mejora en tiempo que nos proveen los threads.