Conversión RGB-YUV usando intrinsics.

Luis Arias, Guillermo López, Edward Umaña

Junio 2019

1. Formato de Imágenes

1.1. Imagen de entrada

La imagen de entrada debe corresponder a una imagen de extensión .rgb y sus dimensiones deben ser de 640x480 pixeles. El formato de la imagen debe ser RGB24, donde los pixeles deben estar almacenados cada uno como un paquete de 24 bits, con un formato de pixel BGRA; es decir, en la imagen cada pixel debe tener sus tres componentes R, G y B, de un byte cada una, y se almacenan de forma secuencial en un paquete de 3 bytes en el orden B, G y R.

1.2. Imagen de salida

La imagen de salida creada será una imagen de extensión .yuv y de dimensiones 640x480 pixeles. El formato de la imagen será YUV444, donde los pixeles serán almacenados en paquetes de 24 bits con un formato de pixel YUV; es decir, en la imagen cada pixel tendrá sus tres componentes Y, U y V, de un byte cada una, y se almacenan de forma secuencial en un paquete de 3 bytes en el orden Y, U y V.

2. Algoritmo

Inicialmente el algoritmo leerá la imagen de entrada completamente, creando una matriz de dimensiones 640x480 con los pixeles de la imagen. La lectura se hace por filas, de forma que se va leyendo la imagen fila por fila, a la vez que se almacena la información en la matriz.

Una vez leída completamente la imagen se procede a la conversión, esta se hace pixel por pixel. En primera instancia se toman los valores R, G y B del pixel y se mueven a un vector 1x4 de tipo float32 (con el cuarto valor del vector igual a cero). Posteriormente, se normaliza los valores para pasarlos de enteros de 0 a 255 a flotantes de entre 0 y 1; esto para poder llevar a cabo la conversión a YUV.

Por medio de la función intrinsics de neon vmulf32 se realizan las siguientes multiplicaciones vectoriales:

$$vect_Y = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \\ 0 \end{bmatrix} \tag{1}$$

$$vect_U = \begin{bmatrix} -0{,}14713 & -0{,}28886 & 0{,}436 & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \\ 0 \end{bmatrix} \tag{2}$$

$$vect_V = \begin{bmatrix} 0{,}615 & -0{,}51499 & -0{,}10001 & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \\ 0 \end{bmatrix} \tag{3}$$

Para obtener los valores Y, U y V se deben sumar todas los valores almacenados en los vectores vectY, vectU y vectV recién calculados. Para ello se usa la función vectorial vpadd que permite sumar los valores en de dos diferentes vectores a mismo tiempo.

Una vez obtenidos los valores Y, U y V. es necesario pasarlos a la misma escala, ya que Y es un valor entre 0 y 1, U es un valor entre -0.436 y 0.436 y V es un valor entre -0.615 y 0.615. Por lo tanto, el siguiente paso es escalar cada uno de lo valores para ser representados en un sólo byte, es decir, a valores de entre 0 y 255. En este proceso, se crea una nueva matriz con las mismas dimensiones que la anterior donde se van almacenando los pixeles en el formato YUV.

Finalmente, se hace la escritura de la los pixeles en formato YUV en la imagen de salida. Para ello se toman los pixeles empaquetados y se almacenan fila por fila hasta haber almacenado todos los valores de la matriz en la nueva imagen.

3. Resultados

3.1. Imágenes

En las Fig. 1 y 2 se muestran las imágenes de entrada y salida respectivamente. Para poder ser presentadas en este documento se usó la herramienta en

Figura 1: Imagen de entrada en formato RGB24.

Figura 2: Imagen de salida en formato YUV444

línea http://rawpixels.net/, la cual lee imágenes en formato RGB24 y YUV444 y las presenta gráficamente.

3.2. Tiempo de Ejecución

Para la medición del tiempo de ejecución se realizó una prueba que realiza la conversión de la imagen 100 veces, suma los tiempos, y los promedia. Se obtuvo un tiempo de ejecución promedio de 160 ms.