

**LABORATORIO**

**EQUIPOS Y SISTEMAS DE**

**AUDIO Y VIDEO**

**(ESAV)**

*Práctica Final (Borrador)*

***Analizador de video Vectorscopio para interfaces SDI***

Curso 2018-2019

Grupo 2:

Alba Blanco, Miguel Ángel

Durán Castro, Patricia

Rodrigo Castillo, Adrián

## Introducción.

**En este informe se presenta la arquitectura general y realización de un Vectorscopio para archivos SDI.**

**El sistema es capaz de realizar las transformaciones necesarias para representar en un diagrama con la información de color extraída del vector crominancia de un video digital.**

A continuación, se muestra un diagrama de bloques con los módulos necesarios los cuales se explicarán posteriormente.

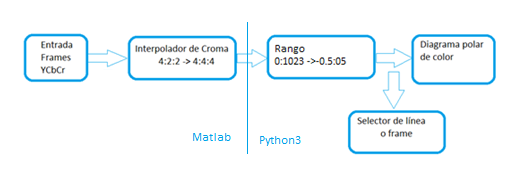


Ilustración 1: Diagrama de bloques.

* **Entrada de frames.** Este módulo es el encargado de recibir la información de vídeo de una trama SDI y extraer las componentes de luma y croma.
* **Interpolador de Croma.** Este módulo es el encargado de realizar una interpolación espacial de la croma en caso de que sea necesario.
* **Rango 0:1023🡪-0.5:0.5.** Este módulo aplican las transformaciones necesarias para poder representar los valores en el Vectorscopio.
* **Diagrama de color.** Representa gráficamente la información de color en un diagrama polar.
* **Selector de línea o frame.** Ofrece la posibilidad de poder representar en el vectorscopio un frame completo o una línea determinada.

|  |  |
| --- | --- |
| Ilustración 2: Imagen diagrama de barras laboratorio | Ilustración 3: Representación del diagrama de barras en Vectorscopio del laboratorio |

Todos los módulos presentados se desarrollan en lenguaje de programación Matlab excepto el módulo de visualización que es desarrollado en lenguaje de programación Python3.

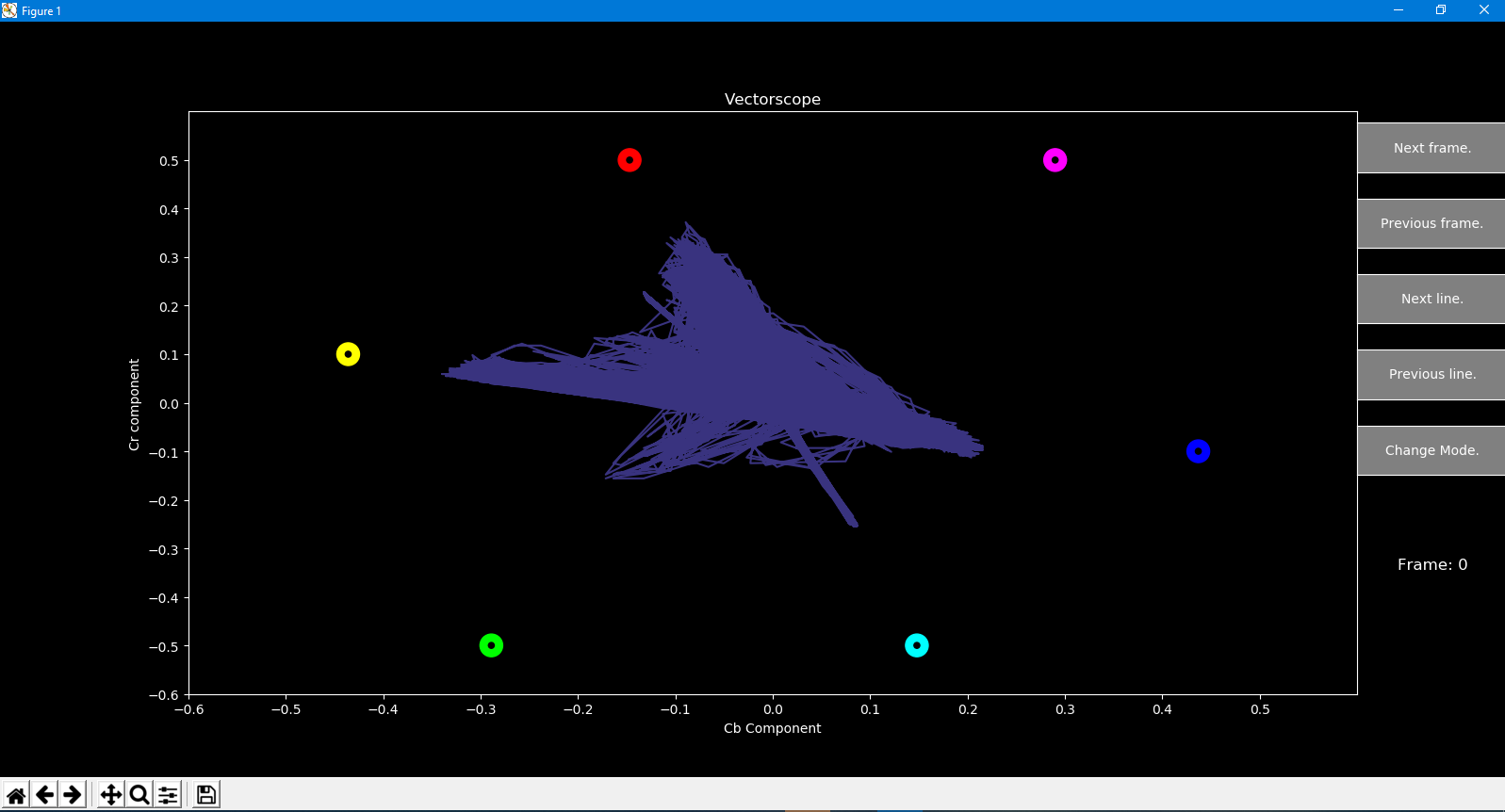


Ilustración . Captura de la ventana del Vectorscopio.

## Matlab

## Entrada de frames

La lectura de una trama SDI se realizará de la siguiente manera:

El bucle principal del programa **sdi\_reader.m** se puede configurar para que extraiga un número determinado de frames del fichero SDI. Por cada iteración, se leerá un frame entero de la trama SDI y se devolverá un objeto que contiene una estructura tipo Cell de Matlab. Esta estructura tiene 3 matrices: Y, Cb, Cr. Estas 3 matrices tienen la misma dimensión (720x576) que se corresponde con una secuencia SDI SD, que es el formato que acepta el programa.

* **Sdi\_reader** devuelve un array con los frames obtenidos.

Este procedimiento principal utilizará la función **read\_video\_frame** para obtener un frame por cada iteración del bucle for e interpola las componentes Cb y Cr, dado que la información original de vídeo es 4:2:2 a través de la función **cbcr2tocbcr4**. Para modificar la cantidad de frames a leer, solo hay que cambiar el rango de la variable :

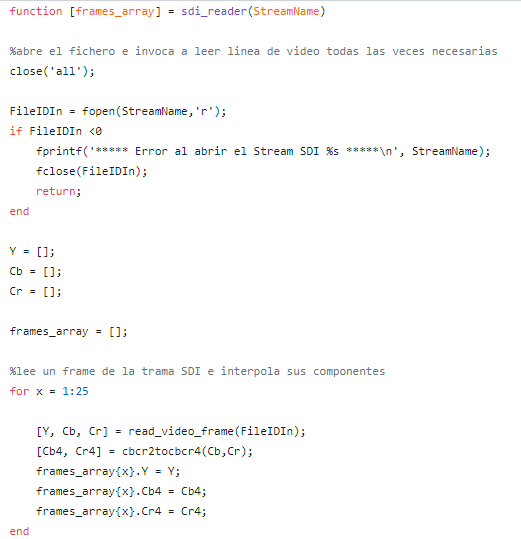


Ilustración . Estructura general de sdi\_reader

* La función **read\_video\_frame** lee un frame de la trama SDI y devuelve los arrays Y, Cb, Cr, iterando 625 veces, una vez por cada línea de video del formato SD. Esta función utiliza la función **read\_line\_from\_SDI** para obtener una línea de la trama. En función del tipo de línea que se ha encontrado, **read\_video\_frame** la guarda en el array Y, Cb y Cr si es de video activo o la descarta si pertenece al sincronismo vertical.

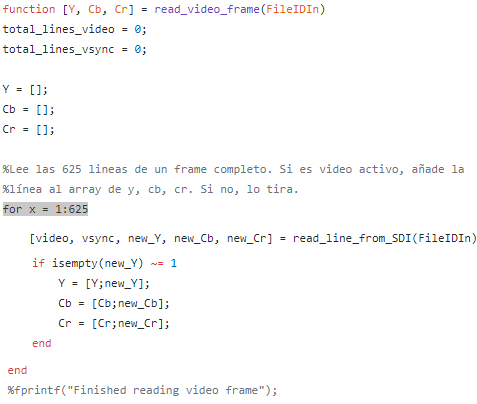


Ilustración . Estructura general de read\_video\_frame

* La función **read\_line\_from\_sdi** leeuna línea de la trama SDI y devuelve 2 flags que indican si la línea es de video activo o no y arrays de tamaño 720, 360 y 360 respectivamente de las componentes Y, Cb, Cr pertenecientes a la línea.

Esta función utiliza la función **find\_trs**, que se encarga de encontrar el primer TRS EAV, para que el puntero lector del fichero de video quede colocado en la posición correcta para poder leer la trama siempre a partir de un EAV. Esto permite leer tramas SDI que no empiecen justo en un EAV.

Cuando **read\_line\_from\_SDI** se está listo para la lectura, descarta la información de HANC y SAV, deja el puntero de fichero listo para leer el payload y activa el flag **video** o **vsync** dependiendo de si la línea es video activo o no. Esto se determina mediante el flag V de la parte XYZ del EAV.

Para leer el payload, se utiliza la función **read\_payload**.

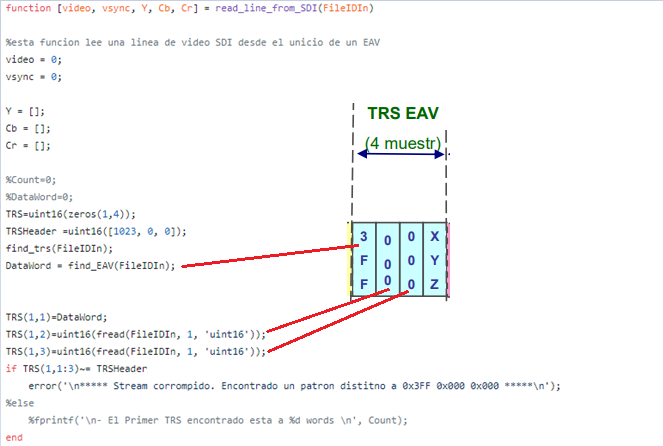


Ilustración . Estructura general de read\_line\_from\_SDI.

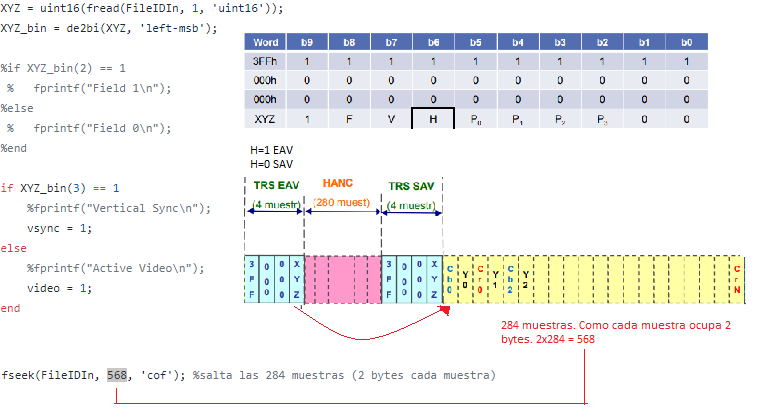


Ilustración . Estructura general de read\_line\_fom\_SDI.

* La función **read\_payload** lee las 1440 de video, en grupos de 4 muestras para poder iterar sobre todas las muestras y devuelve un array de dimensiones 720, 360 y 360 de las muestras Y, Cb, Cr respectivamente. Si el payload es de sincronismo vertical, todo se descarta, avanzando el puntero de fichero.

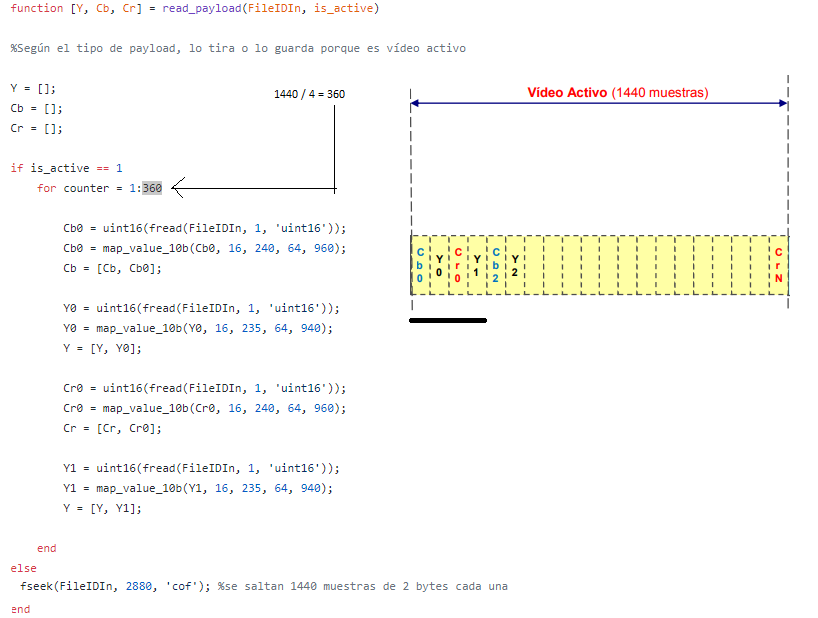
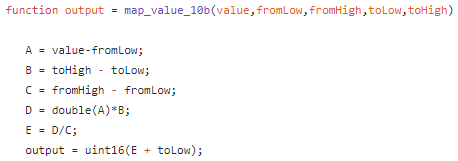


Ilustración . Estructura general de read\_payload.

Cabe mencionar que la función **map\_value\_10b()** es una conversión de profundidad de bit de

8 bit por pixel a 10. Esto es debido a que en este caso, las secuencias de test que se utilizaron

tenían una profundidad de pixel de 8 bits, aunque en la realidad toda trama acorde al estándar SD-SDI debería tener una profundidad de 10 bits por muestra.



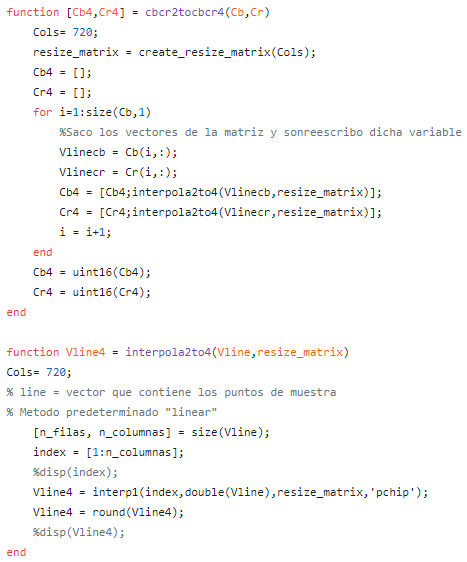
## Interpolador de croma.

Dado el hecho de que la información de vídeo en una trama SDI está submuestreada en 4:2:2,

es necesario interpolar Cb y Cr para obtener las 3 componentes Y, Cb y Cr con las mismas

dimensiones de 720x576.

Para ello, se utiliza la función **cbcr2tocbcr4.**



## Python3

## El programa principal (main()) de Python es el siguiente:

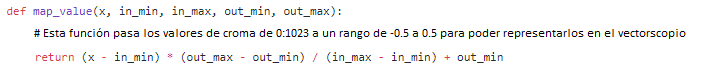
## 

Donde se llamaran las siguientes funciones:

* get\_video\_frame() 🡪 explicada en el apartado c).
* draw\_vectorscope() 🡪 explicada en el apartado b).
* draw\_video\_info() 🡪 explicada en el apartado b).
* button()🡪 explicada en el apartado c).

## Cambio de Rango.

La función **map\_value** está contenida dentrode **get\_video\_frame**(). Es la encargada de realizar la transformación de un rango de 0 a 1023 procedente de los 1024 valores correspondientes a los 10 bit de profundidad de croma al rango del vectorscopio el cual es entre -0.5 a 0.5.



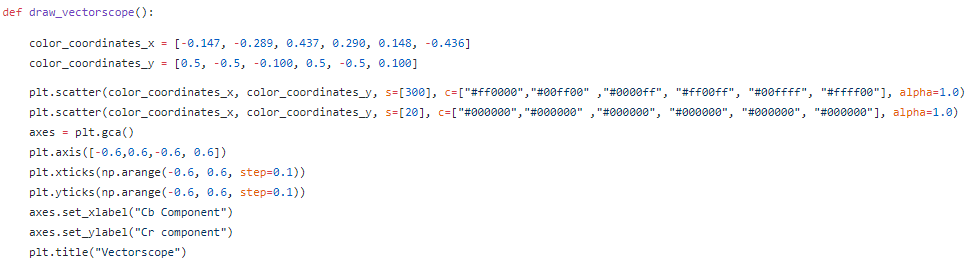
## Diagrama polar de color.

La función **draw\_vectorscope** representa en un plano el diagrama polar con los valores de Crominancia.

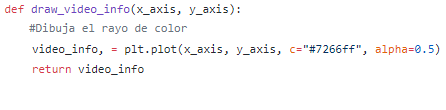
La función plt.scatter() se representa mediante círculos, los valores de los colores primarios y secundarios: R, G, B, M, Y, C.

s= [] representa el área del círculo para dichos colores. El valor de 300 sirve para poder percibir de manera apropiada el color y 20 el punto exacto en el que se centran.

El eje x representará los valores correspondientes a la componente Cb mientras que él eje y representará los valores correspondientes a Cr.



La función **draw\_video\_info** representa en el plano el rayo correspondiente al valor de crominancia del frame o línea seleccionado.



## Selector de línea o frame.

Get\_video\_frame()