**Recolección de datos con Kinect usando la librería PyKinect en Python. Análisis de gestos.**

**(Data collection with Kinect using the PyKinect library in Python. Gesture analysis.)**

Introducción.

Objetivos.

General.

Especifico.

***Librería PyKinect.***

Esta librería esta implementada para visual y Python, no viene integrada en el paquete de herramientas de visual, sin embargo, se puede encontrar en la página oficial como PTVS, desde aquí se integran todas las herramientas de Python en visual. Los desarrolladores de VB crean un *wrap* del SDK4Kinect para su implementación en Python.

Esta librería permite el acceso a la matriz de micrófonos, al Angulo de elevación, a las cámaras RGB y *Depth,* Por último nos da espacio para hacer otro tipo de modificaciones. Coordenadas de la nube de puntos para su posterior visualización o incluso extraer los datos de acelerómetro del sensor, si se tiene un poco mas de conocimientos de la programación propia del Kinect, ya que esta librería tiene un comportamiento similar al del SDK, de esta forma basta con leer la guía de programación de Kinect para saber cuales son las funciones, la implementación es similar.

Para extraer los datos del Kinect se debe:

1. Iniciar el Kinect.

2.

3. Extraer la imagen RGB/Depth.

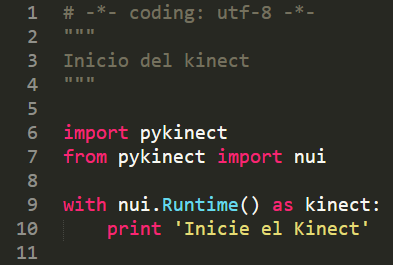
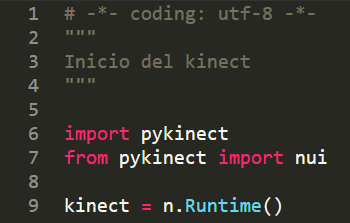
4. Dibujar las imágenes.

***Iniciar el Kinect.***

La librería Pykinect se comporta de manera similar al SDK original, ya que es una envoltura (Wrap). Para iniciar el Kinect debemos importar de la librería pykinect el método ***“nui”*** en este se encuentra contenido todo lo que necesitamos del Kinect, como la elevación del Angulo de la cámara, la recolección de la imagen RGB, de la imagen Depth y del esqueleto.

El *nui* es un paquete contenido en la librería *pykinect* contiene una clase en su interior llamada Runtime(), esta es la encargada de representar el Kinect para su posterior uso.

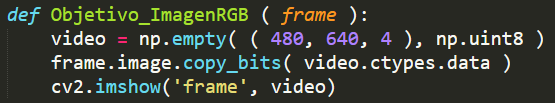
Para iniciar el Kinect debemos construir una estructura que nos permita trabajar con el dispositivo, como se muestra en cualquiera de los ejemplos de la figura.

***Recolección de la imagen RGB.***

Debido a que Runtime de la librería contiene tres funciones que serán las que permitirán actualizar los datos de las cámaras e incluso el *skeletonTraking* (Este ultimo con mayor interes). La función *VideoFrameReady()* permite correr la información de esta imagen y es ejecutada en un hilo independiente, solo hay que agregarle una función objetivo y esta la tomara y la correrá en su propio bucle.

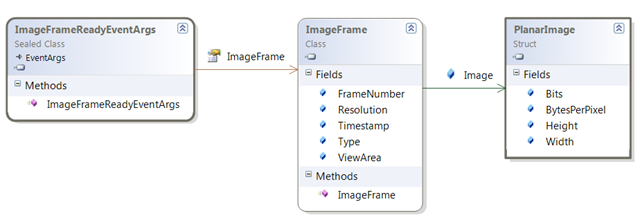
La función que describiremos a continuación es la función objetico, esta podrá ser representada de la siguiente forma:



Esta función crea una variable que contendrá a la imagen en su composición RGBW, por lo tanto, va a ser una matriz del tamaño de la imagen (480X640), por cuatro capas que son las que contienen los componentes RGBW (Uno por capa), dejando asi una matriz de 480X640X4 y caben en un entero de 8 bit.

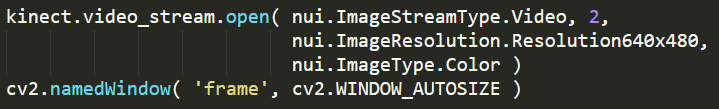
El objeto *frame* es el que va a contener los datos necesarios, este contenedor viene como requerimiento en la función raíz de la librería. Por lo cual desde ella se hace copiar la imagen a la variable *video*. En la última línea, se dibuja la imagen en la ventana seleccionada.

Hay que tener en cuenta como es la estructura de los argumentos del contenedor, la figura descompone dichos argumentos para que se puedan tener en cuenta.

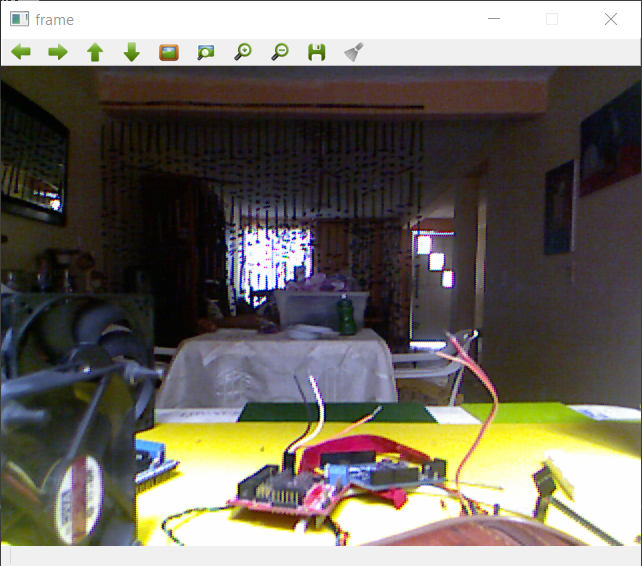


Por último, se asigna la función objetivo a la función de la librería en una función *main*, y se asignan las características de video necesarias.



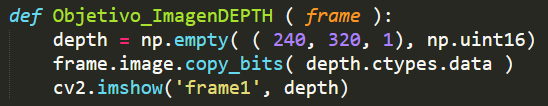


Obteniendo el resultado de la figura.



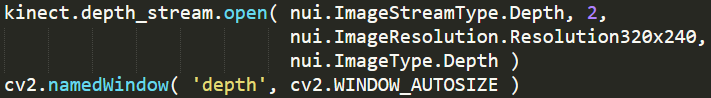
***Recolección de la imagen de profundidad:***

Siguiendo pasos similares a la recolección de la imagen RGB, creamos la función objetivo del *Depth Frame Ready* de la librería pykinect, como en la figura

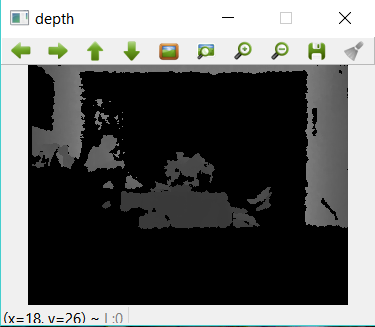


Se hacen las asignaciones necesarias y las modificaciones a las características.



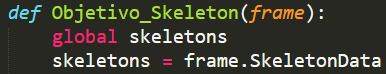


Se obtiene los resultados de la figura.



***Recolección del SkeletonTraking.***

Para recolectar los datos del esqueleto también se debe fijar una función objetivo como las de *Video Frame Ready y Depth Frame Ready,* esta se llamada por el mismo estilo de las anteriores *skeleton Frame Ready.* Aquí el contenedor frame se adapta a los datos del *Skeleton Engine,* dejando que los datos del *Skeleton* queden guardados en la variable con el mismo nombre como se muestra en la figura.

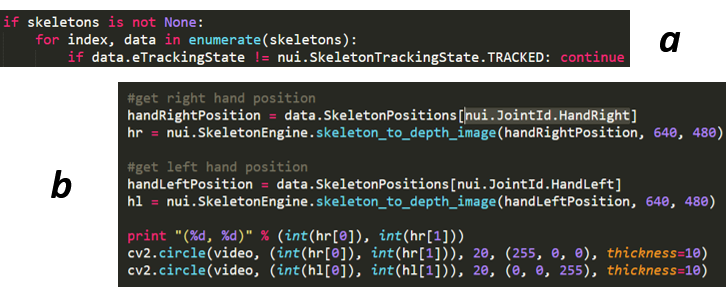


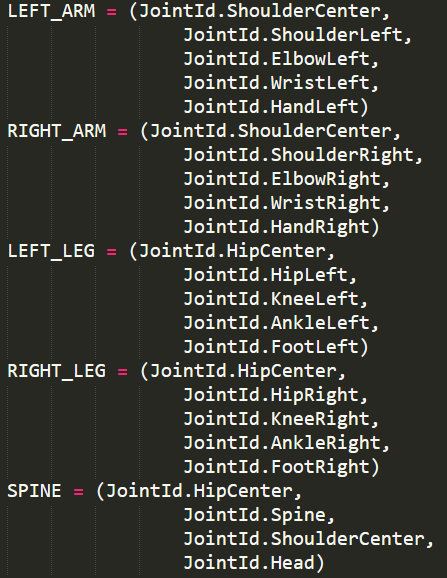
Una vez descrita la función de recolección se procede a hacer las asignaciones en el main, por lo cual antes de todo hay que activar el *Skeleton Engine*, que nos permitirá trabajar con los datos de cada punto anatómico.



Paso posterior seria la visualización que no es mas complicada pero si se tienen que tener en cuenta algunos pasos:

1. Comprobar si hay un *Sekeleton* o no, esto nos dice si hay una persona dispuesta en frente, ver *figura -a*.
2. Estraer del objeto *Skeleton* los índices y los datos de posición de cada punto en cada fotograma, segunda linea de la *figura -a*.
3. Extraer los datos de cada punto, colocando su respectivo nombre, esto no es complicado, la clase *nui* de la librería incluye un método que devuelve los valores de cabecera de cada punto llamado *JointID* (Ver *figura*)*.* Para que luego sean implementados en el método *SkeletonPositions* incluido en la data extraída del objeto *Skeleton* como se muestra en la *figura* -b.





De los datos de la figura se obtienen los resultados de las figuras.

***Recolección de la nube de puntos.***

Son dos cosas muy distintas una nube de puntos a la imagen de profundidad. Una nube de puntos se puede representar como una tupla de vectores de 4 posiciones (x,y,z,w), que representan las coordenadas de cada punto de proyección del patrón de luz infrarroja. Devolviéndonos una representación en puntos de la imagen que el sensor de profundidad esta captando. Mientras que la imagen de profundidad, es una representación 2D en escala de grises de lo que el sensor está captando con su cámara infrarroja. Por lo cual se parte de una para recolectar los datos en la otra.

La salida de la imagen de profundidad del Kinect tiene 11 bit con 2048 niveles de profundidad, al menos eso es lo descrito por el dato arrojado cuando ejecutamos esta parte de nuestro código []. Por lo que el representación en 3D de la imagen de profundidad se puede representar por:

Donde *d* es la nube de puntos traducida a espacio cartesiano y ya no como un valor en escala de grises, ,