Complementos de Informática

***Docente*: Adrián Orellana**
***Alumno*: Vittorio Pontoriero**

Año 2022

1. INTODUCCIÓN

Como trabajo integrador se realizó una aplicación de escritorio con el uso del IDE Qt Creator. La aplicación consiste en un software para ser utilizado con la cámara Kinect V2, y realiza un seguimiento de los marcadores corporales seleccionados previamente, guardando sus posiciones y tiempo en un archivo de texto.

El proyecto tuvo en total 10 archivos: uno de extensión *.pro*, tres archivos fuentes (*articulación.h, mainwindow.h y dialogo1.h*), cuatro archivos ejecutables (*main.cpp, mainwindow.cpp, dialogo1.h y articulación.h)* y dos archivos de interfaz de usuario, de extensión *.ui* (*mainwindow.ui y dialogo1.ui).*

Se desarrolló una clase *articulación* donde se almacenarán los datos obtenidos mediante la cámara. Del total de 16 marcadores posibles, se crean 16 objetos articulación, de los cuales se habilitará la captura de datos de los cuales sean seleccionados. La selección de marcadores se realiza en una ventana emergente descrita por *dialogo1*.

2. DESARROLLO

Comenzando con la clase articulación, en su archivo de cabecera encontramos sus métodos y atributos públicos y privados:

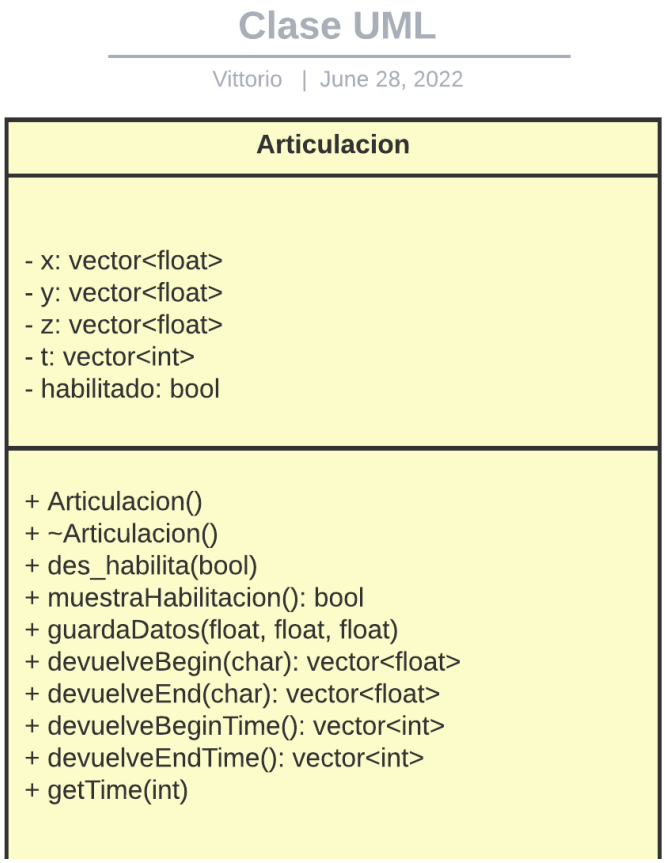


Figura 1: Clase Articulación.

- Los contenedores STL clase vector de tipo *float* corresponden las posiciones x, y, z del respectivo marcador.
- El contenedor STL clase vector tipo *int*, que corresponderá al tiempo.
- El booleano llamado *habilitado* (que es cargado con *false*) corresponde a la bandera que determinará que el marcador fue habilitado para la toma de sus datos.
- Método *des_habilita* cambia la bandera *habilitado*, según el argumento que entre en el método.
- Método *muestraHabilitacion* retorna el valor de la bandera *habilitado*.
- Dos métodos que retornan *std::vector<float>::iterator*, uno para el principio del iterador, y otro para el final. Estos métodos toman como argumento una variable tipo *char* que determinará de que vector quiere el principio o final del iterador (x, y o z).
- Un método llamado *getTime* que tiene de argumento un valor *int*, que se almacenará en el vector del tiempo.

Dentro del archivo *mainwindow.h* se crea un arreglo de la clase *Articulación* de tamaño 16 (total de posiciones). Sumándole un timer, un entero llamado *timeSet* donde se guardará el tiempo que el usuario desea tomar como base, un *string* donde se guardará el nombre del archivo que el usuario desee crear/guardar/abrir, una bandera de stop que será levantada al expirar el tiempo seteado, y una serie de métodos y slots que formaran al funcionamiento del sistema. Se cumple con la programación orientada a objetos, pues todos estos métodos y atributos son catalogados como privados, ya que el funcionamiento se hace en su respectivo ejecutable (*mainwindow.cpp*).

2.1) Interfaz

En la interfaz de usuario del software nos encontramos con diferentes opciones: un menú de herramientas, y en la ventana principal, opciones para el seteo del timer.

Dentro del menú de herramientas tenemos las opciones: *File, Camera, Data*.

En la opción *File* se despliega otro meno con: *New File, Open File, Save, Save As*, que, realizan su función según corresponde a su nombre. Para el caso de crear un nuevo archivo, lo que se hace es guardar el nombre que el usuario le da al archivo, para que luego de realizar una prueba el archivo sea cargado con los datos obtenidos mediante la opción de guardado. Para el caso de abrir un archivo, se tomará el nombre del archivo ya existente que el usuario introduce, permitiendo así al usuario a generar una nueva prueba que podrá ser guardada en el archivo abierto. Y para el caso de guardar como, se crea un nuevo archivo cuyo nombre el usuario introduce, y se guardan los datos ya obtenidos mediante el mismo método que el guardado normal (se utilizan los métodos de crear y guardar).

Para la opción de la cámara, se despliega un menú con una sola opción: la de abrirla (*Open*). Al clickear sobre esta opción, se abrirá una nueva ventana con la visualización que nos da la cámara de Kinect. En esta ventana, el usuario solo podrá cerrarla una vez que la actualización de la imagen se encuentre en pausa, esto pasará cuando se cumpla con el tiempo que el usuario configuró, o cuando el mismo presione la tecla ESCAPE.

Y, por último, para la opción *Data* se despliega una sola opción de *Select Markers*. Aquí se abrirá una nueva ventana, descrita por los archivos *dialog1.h* y *dialog1.cpp* donde el usuario podrá seleccionar los marcadores deseados: cabeza, manubrio esternal, hombro derecho, hombro izquierdo, codo derecho, codo izquierdo, muñeca derecha, muñeca izquierda, columna vertebral (punto vertical medio), cresta ilíaca derecha, cresta ilíaca izquierda, sínfisis púbica, rodilla derecha, rodilla izquierda, pie derecho y pie izquierdo. Para cada opción marcada, se verá reflejado en una imagen de un cuerpo humano al lado de los casilleros, un circulo rojo sobre la parte del cuerpo que refiere el marcador seleccionado.

Fuera de la barra de herramientas, sobre la ventana principal, nos encontramos con la configuración del timer. Aquí el usuario podrá configurar el tiempo que desea que dure la obtención de datos. Debe elegir un valor entero en segundos, y una vez que la cámara sea abierta, se visualizará en la misma ventana, un contador del timer en milisegundos que parará una vez que se alcance el tiempo deseado (al mismo momento se parará la visualización de la cámara, ventana que el usuario podrá cerrar en ese instante).

2.2) Mainwindow: estructura

En el constructor de la ventana *mainwindow*, declarado en su respectivo ejecutable (*mainwidnow.h*) se establecen las conexiones a los respectivos botones con los que cuenta el software.

Para la opción *Select Markers* se llama a un *slot* llamado *Select_Markers()* que lo que hace es mostrar la ventana (que ya fue creada como variable en la ventana principal).

Otra conexión establecida, es cuando la ventana de *Select Markers* fue terminada. Si el usuario presionó sobre el botón “OK” se llama a otro *slot* llamado *RefreshMarkers()*. Sobre este *slot* se lee los marcadores que el usuario seleccionó en la ventana de *SelectMarkers* y para cada marcador seleccionado, se levanta la bandera de habilitación de la articulación correspondiente (de las 16 posibles).

Se hace uso de un “spinBox” para la configuración del tiempo. En este spin box el usuario introducirá el tiempo en segundos para la toma de datos. En caso de que el usuario deje en cero este casillero, la adquisición de datos será continua hasta que el

usuario presione la tecla ESCAPE. El tiempo configurado será almacenado en la variable *timeSet* mediante el slot *setTime()*.

Para el timer se hace otra conexión que tomará la señal de *stop* del timer. Es decir, cuando el timer alcance el tiempo configurado, se llamará a un *slot* llamado *levantaBandera()* donde, como dice su nombre, lo único que se hace es cambiar el valor booleano de la *banderaStop* (variable de *mainwindow*) a *true.* Esta bandera será analizada en la visualización de la cámara, para que cuando sea levantada, no se siga con ese proceso. Si el tiempo configurado fue cero, el timer se cargará con un valor de 3600000, lo que significan 3600 segundos (una hora), esto es un tiempo exagerado que el software permite sin problemas, para que se pueda tomar los datos del tiempo sin tener uno establecido como máximo.

La conexión, si se quiere decir, más importante, es la del botón de *Open* para la cámara, con el *slot kinect_sk()*. Dentro de este slot encontramos el funcionamiento principal del software: apertura de la cámara, dibujo del esqueleto humano, seguimiento de la persona, y captura de datos.

Luego, como último en el constructor, tenemos las respectivas conexiones para el manejo del archivo (*New, Open, Save, Save As)*.

2.3) Kinect_sk: funcionamiento

Dentro de este método (en realidad, slot) es donde se cumple la verdadera función del software.

Primero, se hace una evaluación del control del sensor de Kinect: verifica que exista, se crean las variables para los futuros frames, y se trabaja la imagen que el sensor captará, con el uso de las librerías de openCV (principalmente en la creación de la matriz a la que se transformará la imagen).

Una vez que se crean las variables necesarias para la apertura de la cámara, antes de entrar en un bucle *while* que dará la imagen necesitada, se baja la bandera de *stop*, se inactiva el *spinBox* para que no se pueda cambiar el tiempo del timer durante la visualización de la imagen, y por último, se da *start* al timer (como fue explicado, con el tiempo configurado por el usuario, o con un valor exagerado).

Dentro del bucle *while* se toma la imagen captada, y se evalúa el reconocimiento del cuerpo. Si fue reconocido, se dibujan los marcadores corporales sobre la imagen. Aquí se llama a un método llamado *draw*, donde entran como argumentos: la imagen, dos marcadores contiguos, la imagen en 2D, y un entero que hace referencia al primer marcador de los dos contiguos.

En el método *draw* se grafica al esqueleto humano sobre la imagen visualizada (con el uso de la librería de openCV). Además, en este método se realiza la captura de datos, gracias a las variables cedidas por la librería de *Kinect.h*: se copian los valores de

las posiciones x, y, z que Kinect captura y guarda en la clase *Joint*, y se toma el tiempo del tiempo gracias al método del timer *remainingTime().* Para guardar el tiempo de manera creciente, se toma en el método *getTime* de la articulación la diferencia entre el tiempo seteado y el tiempo actual (ya que el timer cuenta de manera decreciente).

3. PRUEBA EJEMPLO

Para evaluar el funcionamiento del software, se probó sobre un paciente sometiéndolo a una resistencia sobre su extremidad izquierda.

Se hizo uso del software para evaluar la precisión de este. Los datos fueron exportados a otro software de estudio: MatLab.

Para esta prueba no se configuró tiempo alguno, sino que la prueba finalizó al presionar la tecla ESCAPE.

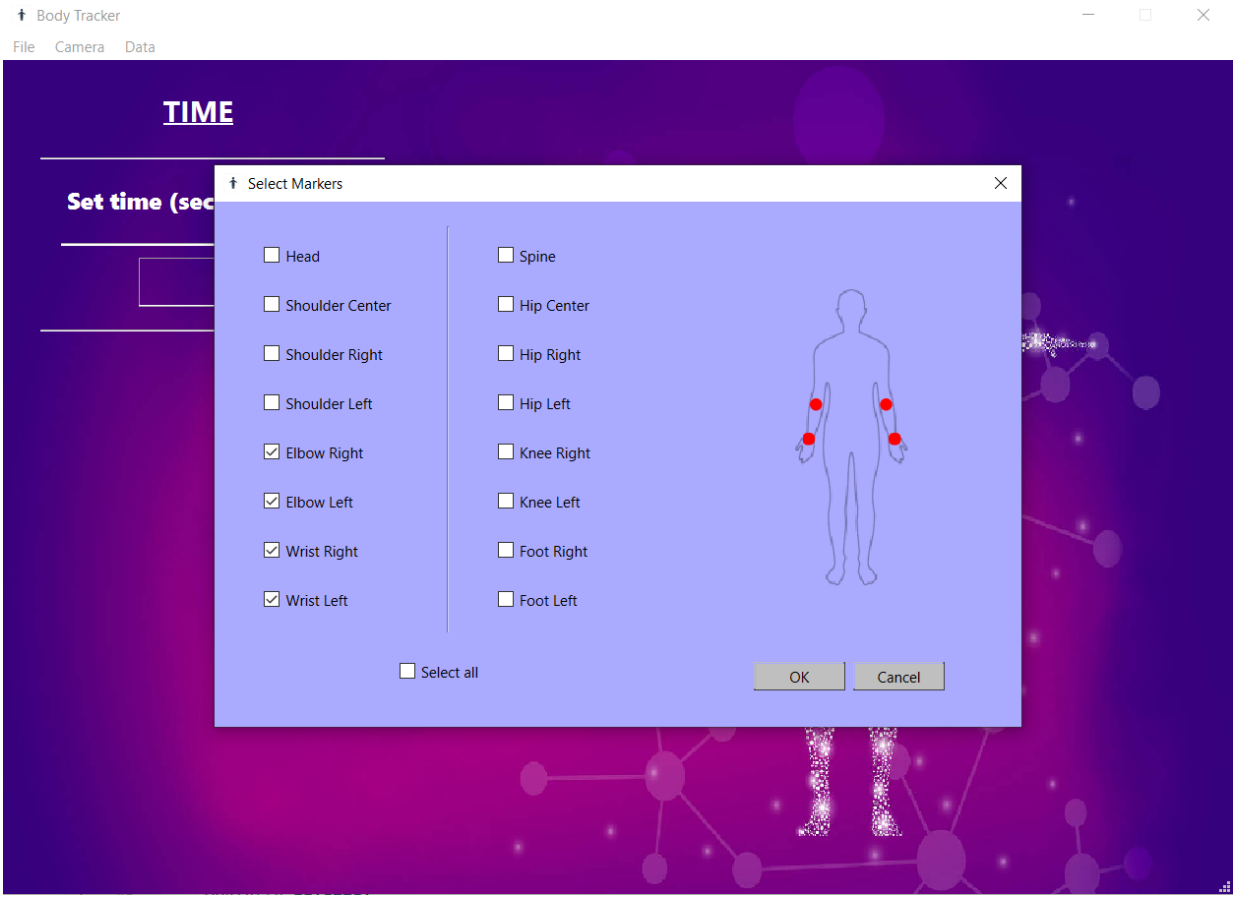


Figura 2: Selección de marcadores corporales.

Para el estudio sobre este paciente se seleccionaron ambos codos, y ambas muñecas. La idea era observar y analizar el movimiento del paciente y compararlo entre ambas extremidades, teniendo en cuenta que la resistencia se encuentra en su extremidad superior izquierda.

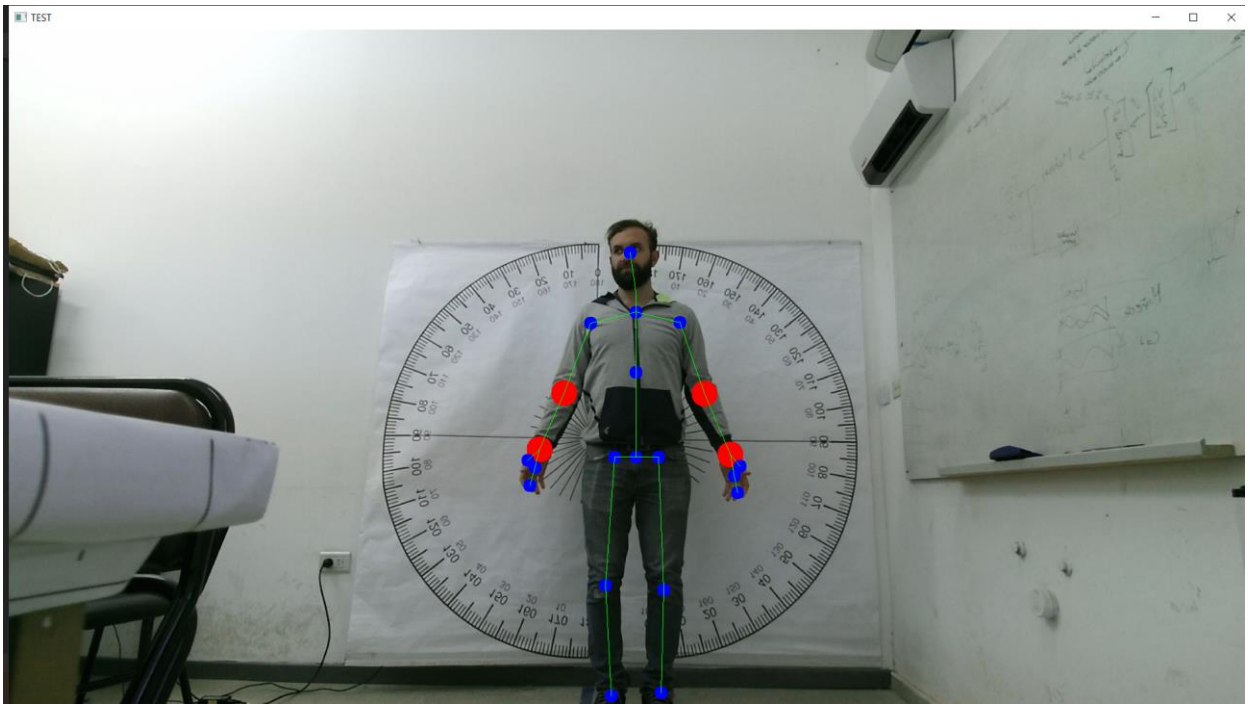


Figura 3: Visualización de la cámara del sensor Kinect.

Se le pidio al paciente que haga un movimiento hacia arriba máximo con ambas extremidades superiores. Una vez llegado a la máxima altura, que descienda. Una vez termianda la prueba se terminó la visualización manualmente (pues no se contaba con tiempo de base) y se guardaron los datos mediante la opción de *Save As…*

Los datos se obtuvieron en un archivo de texto, donde se encuentran las posiciones de los marcadores seleccionados en un plano tridimensional, es decir, conjuntos de tres puntos (uno para cada eje). Además, también se guarda el tiempo en el que se adquirió cada punto.

Se levantaron los datos en MatLab, y se graficó la trayectoria de todos los puntos en un plano tridimensional:

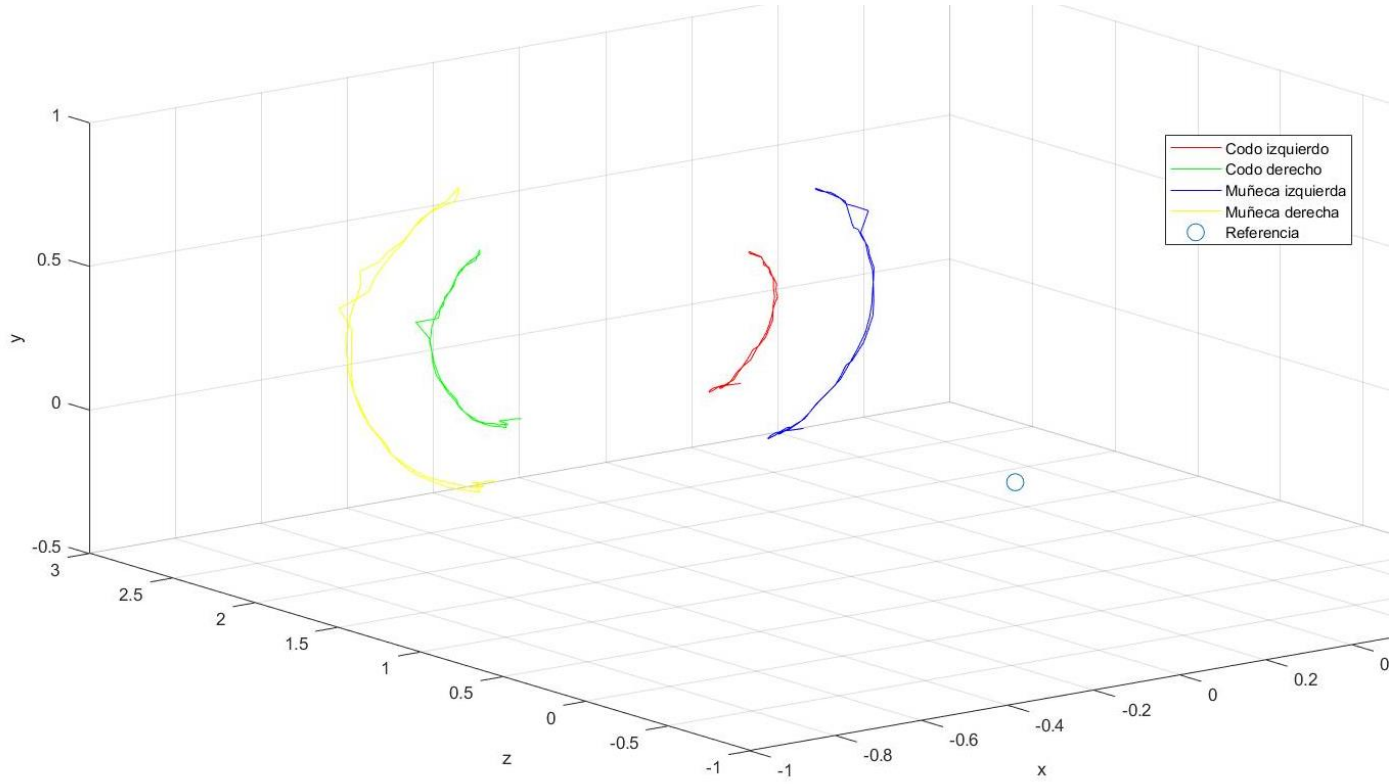


Figura 4: Grafica tridimensional.

La referencia graficada refiere a la cámara del sensor, éste sería el punto (0,0,0).

También se obtuvo para el codo derecho, la gráfica de su movimiento en función del tiempo:

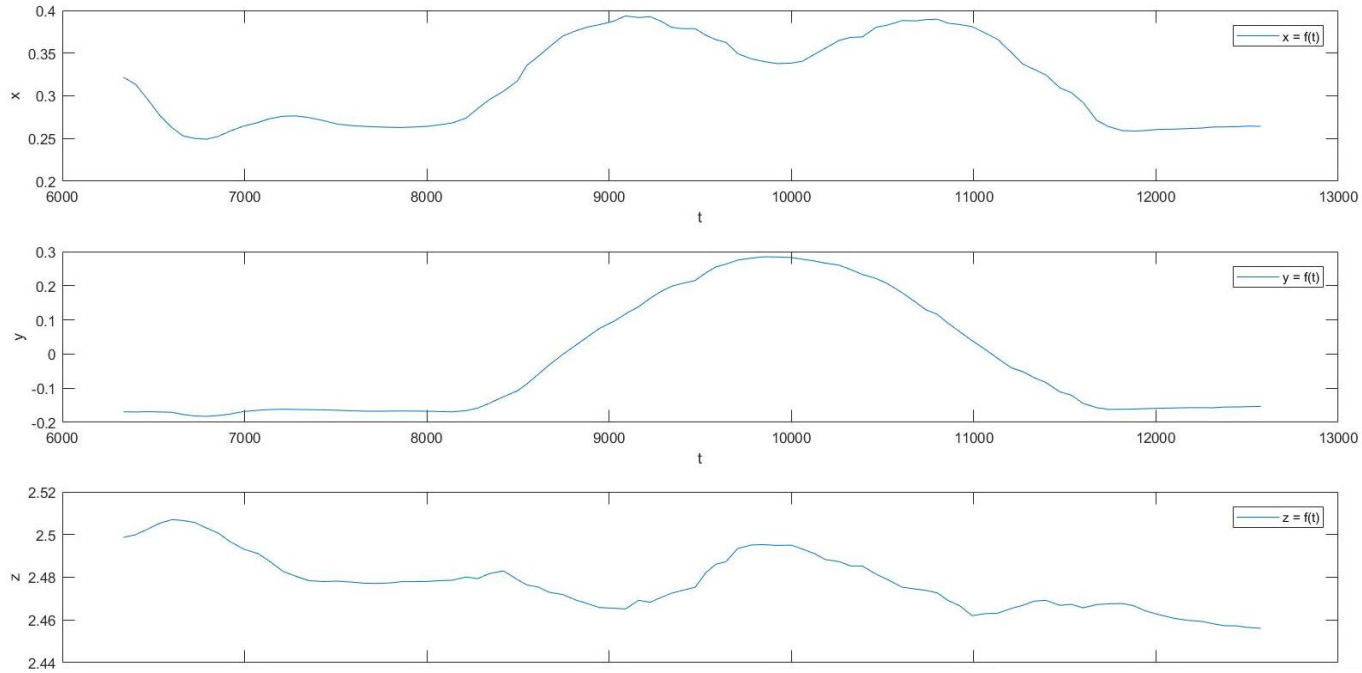


Figura 5: Posición en función del tiempo.

4. CONCLUSIÓN

Para finalizar, es necesario analizar la confiabilidad de las gráficas obtenidas y compararlas con lo esperado.

Se puede observar en la figura 3 que los marcadores izquierdos tomaron una trayectoria más corta en comparación a los derechos. Esto era de esperar, pues la resistencia se encuentra en esta extremidad.

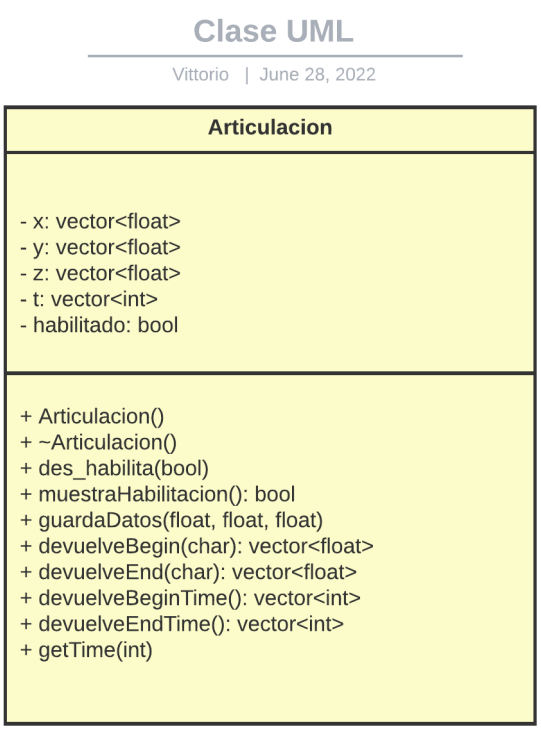
Lo que se analiza en la figura 4, graficas de posiciones en función del tiempo, es como en cada eje se toma coherencia con lo trazado sobre el tiempo. Por ejemplo, al ser un movimiento abducción-aducción, es esperable encontrar que en el eje Y encontremos una meseta. Esto es de verifcar en la figura 4.

En cuanto al proyecto realizado, se concluye con que hay muchos puntos a mejorar (o mejor dicho, a agregar).

Empezando con que sería de gran ayuda no depender de un software externo como MatLab, se puede agregar la posibilidad de obtener gráficas sobre el mismo software.

Además, se podría agregar mucho más analisis biomecánico. Por ejemplo, es de gran utilidad todo el analisis realizado sobre la excursión angular de algun punto del cuerpo sobre otro punto de referencia. Esto es solo un ejemplo y existen una infinitud de analisis realizable sobre el cuerpo humano en cuanto a su biomecánica.

Otro punto a agregar podría ser el tema de filtrar la señal. Esto no se realizó en el proyecto ya que hay gran confiabilidad sobre los puntos obtenidos por Kinect (esto se puede verificar en la figura 3) sin embarg, para una mejor intuición, es recomendable suavizar un poco las trayectorias.



Finalmente, el software cumplió su función principal, cumpliendo con la programación orientada a objetos, y haciendo uso de ella para un tema externo como lo es la biomecánica.