Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

FUNCIONES BÁSICAS PARA RECONOCIMIENTO AUTOMÁTICO DE LENGUAJE DE SIGNOS UTILIZANDO LEAP MOTION.

Alumno: Adrián Garrote Núñez

Tutor: Prof. D. Raúl Mata Campos
Depto.: Ingeniería de Telecomunicación

Junio, 2016

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

Trabajo Fin de Grado

Funciones básicas para reconocimiento automático de lenguaje de signos utilizando Leap Motion.

Junio, 2016.

Alumno: **Adrián Garrote Núñez** Tutor: **Raúl Mata Campos**

ÍNDICE

1. RESUMEN

El principal objetivo de este trabajo es estudiar las funciones básicas para el reconocimiento automático de lenguaje de signos utilizando Leap Motion. Con el objetivo final de diseñar un traductor de lengua de signos para facilitar y agilizar la comunicación entre una persona sordomuda y una persona con capacidad auditiva que desconozca la lengua de signos.

Para ello, se va a adquirir nuevas técnicas avanzadas de procesamiento de imagen/señal para el reconocimiento de gestos utilizando las manos y dedos. En concreto, en este trabajo se pretende la implementación de una librería básica para Leap Motion que permita reconocer y grabar gestos correspondientes al lenguaje de signos. En total, el diccionario final que será proporcionado constará de 10 gestos y se usará el algoritmo Dynamic Time Warping (DTW) para el entrenamiento del sistema y la traducción. Además, este diccionario será modificable pudiendo añadir el usuario su propio diccionario. El dispositivo Leap Motion se encarga de capturar los movimientos de las manos y dedos con una alta precisión (unos 0.01 mm) y en un entorno de 3D.

Se tendrá entonces, una aplicación que va a permitir tanto grabar gestos para crear así un diccionario propio, como reconocer estos gestos para implementar un traductor de lengua de signos. El software ha sido desarrollado para PC y se ha empleado el lenguaje C++ para su desarrollo, por lo que sería sencillo añadirlo a otras posibles aplicaciones.

2. INTRODUCCIÓN

A lo largo del tiempo, se han creado mecanismos para facilitar la interacción entre humano y ordenador. Control mediante teclas y ratón, pantallas táctiles, reconocimiento de gestos, comandos de voz, etc. Sobre todo, se ha buscado dar solución y agilidad a determinados problemas en determinadas aplicaciones. Aunque cabe destacar también el gran auge que ha tenido el mundo de los videojuegos en este crecimiento que ha experimentado la interacción humano-ordenador, con mecanismos como la realidad virtual, control por gestos, etc.

Debido a este interés por la búsqueda de una integración más mecánica entre humano y ordenador, han surgido diversas tecnologías y dispositivos que se acercan un poco más a este objetivo. Entre ellos cabe destacar (debido a su relación calidad-precio) el dispositivo Kinect, que fue lanzado por Microsoft a finales de 2010 y que permite interactuar con nuestra consola o PC mediante gestos, comandos de voz e imágenes. También tiene su distinción el dispositivo **Leap Motion** (en el cual se centra este trabajo) lanzado al mercado en Julio de 2013 y que permite capturar el movimiento de manos y dedos en su totalidad, con una precisión mucho mayor a la de Kinect. Aunque ya se compararán y explicaran estos dispositivos en posteriores apartados. Surge por tanto un sinfín de aplicaciones que usan principalmente este tipo de dispositivos.

Antes de que salieran a la luz dispositivos como el Leap Motion, el lenguaje de signos tuvo un papel poco importante en este crecimiento debido al coste de los dispositivos y el coste del desarrollo que tendría un sistema con esta finalidad a nivel de usuario. El principal interés se ha centrado en videojuegos y tecnología puntera y llamativa para el usuario. Pero con esto, se ha dejado de lado una amplia población la cual podría verse enormemente beneficiada por esta tecnología. Cabe destacar que según la Confederación Estatal de Personas Sordas (CNSE) [4] en un estudio llevado a cabo en el año 2011, en España había un total de 1.064.000 personas sordas y con algún tipo de discapacidad auditiva (es decir, un 2,3% de la población total) para las cuales en su mayoría su lengua materna es la Lengua de Signos situando la lengua oral y escrita como segunda lengua.

Gracias al dispositivo Leap Motion que se centra en el movimiento de manos, su gran precisión y bajo coste (unos 90€ aproximadamente) es posible implementar una solución totalmente innovadora para agilizar la comunicación de estas personas con otras

que desconozcan la lengua de signos. Y además, se abren otras puertas relacionadas como podría ser el aprendizaje de esta lengua. Aunque de eso se hablará en posteriores capítulos.

2.1 Antecedentes

Los sistemas de signos han sido utilizados desde hace mucho tiempo no sólo por las personas sordas. Estos sistemas, como medio de comunicación para personas sordomudas, han sido estudiados durante años. Entre los primeros estudios, cabe destacar el de Juan de Pablo Bonet (1573 – 1633) el cual en 1620 escribió un libro [Figura 2.1], "Reducción de las letras y arte para enseñar a hablar a los mudos", que enseñaba una forma de comunicación por alfabeto de signos.

Figura 2.1. Páginas del libro de Juan de Pablo Bonet, citado anteriormente.

Más tarde, Charles-Michel de l'Epee (1712-1789) publicó un ejemplar en el que se basan los alfabetos de signos usados en la actualidad. Y así, numerosos estudios hasta conseguir el actual diccionario de la Lengua de Signos Española (LSE) regido por el Centro de Normalización Lingüística de la Lengua de Signos Española (CNLSE).

Desde entonces, se han hecho varios intentos de sintetizar el reconocimiento de gestos a través de sistemas informáticos. La mayoría de ellos, basados en la utilización de visión artificial (cámaras) analizando las imágenes captadas mediante el color, bordes,

movimiento, etc. o mediante el uso de guantes. Para ello, uno de los dispositivos que más se ha usado y estudiado ha sido Microsoft Kinect.

Microsoft Kinect [Figura 2.2] es una barra horizontal de unos 23 cm apoyada sobre una base articulada y circular. Este dispositivo tiene incorporado una cámara RGB, un sensor de profundidad, un micrófono para reconocimiento de voz y un procesador personalizado. Con esto capta el movimiento de todo el cuerpo en 3D. Dispone de un Kit de Desarrollo de Software que permite utilizarlo en Windows.

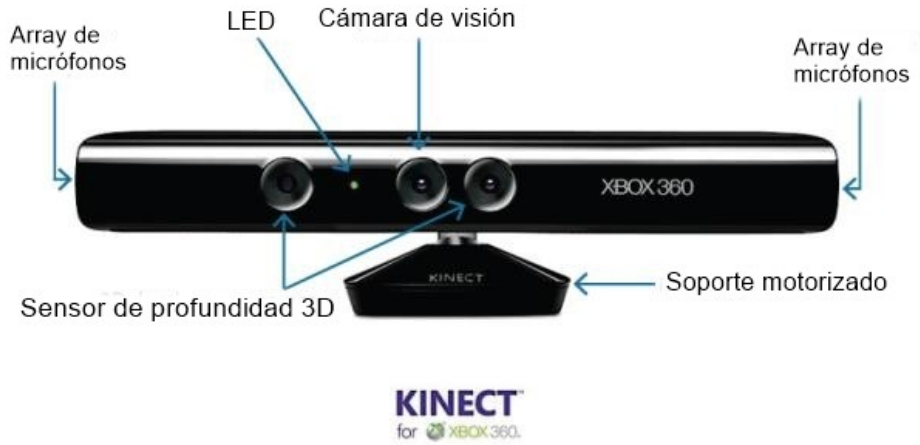

Figura 2.2. Arquitectura del dispositivo Kinect.

Muchas aplicaciones de reconocimiento de gestos han sido desarrolladas para Kinect por multitud de usuarios, incluso la división asiática de Microsoft Research ha estado trabajando en las posibilidades de Kinect para entender la lengua de signos [Figura 2.3]. Pero la falta de diferenciación en aspectos relativos a dedos, la flexión de huesos de la mano, muñeca, velocidad, y otras propiedades importantes de las manos en la lengua de signos, hacen que el dispositivo Kinect no haya tenido especial auge en su posible utilización para esta lengua.

Figura 2.3. Aplicación desarrollada por Microsoft Research para lengua de signos.

Esta falta de detalle e información de la mano que tiene Kinect, es fácilmente capturada con el dispositivo Leap Motion. Además de tener una mayor precisión que Kinect. Por eso, como bien se verá en posteriores apartados, Leap Motion es una potentísima herramienta para el reconocimiento de gestos realizados con las manos y fácil de integrar a la lengua de signos (a diferencia de Kinect).

3. OBJETIVOS Y MOTIVACIÓN

El principal objetivo, como bien se ha dicho anteriormente, es el estudio del dispositivo Leap Motion para su integración en la lengua de signos. La idea es diseñar un traductor de lengua de signos para facilitar y agilizar la comunicación entre una persona sordomuda y una persona con capacidad auditiva que desconozca dicha lengua.

Se va a desarrollar para ello un sistema que permitirá grabar un diccionario propio y usarlo en dicha comunicación. Además, el sistema de reconocimiento que se va a implementar queda abierto para numerosas aplicaciones futuras como podrían ser: enseñanza de lengua de signos, juegos interactivos usando dicha lengua, sistema de control mediante gestos definidos (automóviles, drones, domótica, brazos robóticos…). Es posible plantear por tanto una serie de objetivos fundamentales para alcanzar nuestra meta, y que quedarán definidos como los objetivos del TFG:

- Revisión bibliográfica y estudio de la documentación disponible para las técnicas de análisis, procesamiento y reconocimiento gesticular. Además de la documentación del dispositivo Leap Motion, como es obvio. Para ello, serán estudiadas técnicas que puedan proporcionar utilidad a este proyecto. Además, estás técnicas y/o librerías serán comparadas y se manejará la posibilidad de usarlas para el fin de este trabajo.

- Estudio de aplicaciones existentes y similares. Hay que conocer sus diferentes alcances, limitaciones y requisitos ya que será de ayuda para poder diseñar correctamente las funcionalidades que se van a pretender cubrir con la aplicación desarrollada en este TFG.

- Estudiar la captación, procesamiento y tratamiento de la información obtenida por el dispositivo Leap Motion. Este objetivo es fundamental ya que la aplicación que se pretende desarrollar se centra en el uso de este dispositivo. Se debe conocer cuánta información es posible capturar y cuál será únicamente la necesaria para la aplicación que se pretende desarrollar. Además, dependiendo de cómo sea esta información, se tomarán futuras decisiones sobre el algoritmo a usar, el modo de guardar dicha información, o la precisión del sistema entre otras.

- Diseño e implementación de algoritmos/herramientas/librerías que integrarán la parte de análisis y reconocimiento. Es fundamental conocer si hay disponible alguna herramienta que pueda ser de utilidad en este trabajo, y si está permitido su uso. Conociendo las herramientas disponibles y las que deben ser programadas y diseñadas, es posible desarrollar la aplicación siguiendo una metodología más dedicada.

- Diseño de una aplicación en consola, o en un entorno gráfico a partir del software creado y recopilado. Una vez se haya decidido qué herramientas serán desarrolladas y cuáles serán recopiladas, el siguiente paso será la implementación de estas en la aplicación. Para ello será necesario decidir ciertos parámetros y estudiar el modo de interconexión posible entre dichas herramientas. A primera vista, será necesario tener una interconexión entre la información capturada por Leap Motion y el algoritmo o librería dedicada a reconocer los gestos.

- Implementación y comprobación del correcto funcionamiento de la aplicación. Este apartado es fundamental para conocer si se ha alcanzado el principal objetivo de este TFG. Una vez desarrollada la primera versión de la aplicación, se comprobará especialmente si es funcional y si tiene unos resultados medianamente adecuados para el uso que se pretende dar. Conseguido esto, habrá que llevar a cabo pruebas y comprobaciones un poco más exigentes hasta que sea posible determinar que la aplicación desarrollada es adecuada y funcional para cumplir con el principal objetivo de este TFG.

- Desarrollo de la memoria, manuales y documentación habitual a la presentación de trabajo fin de grado. Para ello, habrá que cumplir con las normas básicas de estilo y estructura proporcionadas por la escuela. En ellas se determinan aspectos como pueden ser el tipo de letra, la numeración de figuras y tablas ó la norma a seguir en la redacción de referencias bibliográficas por ejemplo. Además, para este TFG, será necesaria la inclusión de un anexo que contenga un manual para el usuario en el que se explicará cómo debe ser instalado el dispositivo Leap Motion.

Siguiendo esta metodología, se habrá conseguido alcanzar el principal objetivo. Una vez conseguido, se pondrá la meta más allá para intentar lograr nuevos objetivos siempre con la intención de mejorar este trabajo.

Leap Motion es un sensor que puede revolucionar potencialmente la interacción humano-ordenador y que ofrece numerosas posibilidades de desarrollo. El interés por el estudio de este dispositivo, su gran poder innovador y la idea de poder dar una solución potente e innovadora a la población sordomuda, dan a este trabajo la motivación y dedicación suficiente para alcanzar los objetivos.

4. ESTADO DEL ARTE

En este capítulo, se va a desarrollar el estado del arte del dispositivo Leap Motion. Para ello, se hablará de los diferentes sensores que existen y que permiten de un modo u otro monitorizar el movimiento de alguna parte del cuerpo. Se hablará también de los modelos de representación usados en la monitorización de dicho movimiento, centrándose especialmente en las manos. Finalmente, se hará un estudio más detallado del dispositivo Leap Motion describiendo los aspectos de mayor interés relativos al hardware y al software del mismo.

4.1. Sensores

Hay varios tipos de dispositivos que son capaces de llevar a cabo un seguimiento en los movimientos de una persona y determinar qué gesto está realizando (tras previa programación). Estos tipos son los siguientes:

- **Dispositivos sin visión**: Usan tecnologías del tipo de: acelerómetros, pantallas multi-touch, guantes, brazaletes, sensores electromagnéticos, etc. [Figura 4.1] para detectar el movimiento. Básicamente no poseen cámara.

Figura 4.1 Ejemplo de dispositivos sin visión.

- **Dispositivos con visión**: Usan una o más cámaras para detectar el movimiento, además de otros elementos adicionales. Son del tipo: cámaras de vídeo, stereo cámaras, sensores infra-rojos, marcas de color RGB, etc. [Figura 4.2] **(Aquí se encuentra el dispositivo Leap Motion que será utilizado en este TFG).**

Figura 4.2. Ejemplo de dispositivos con visión.

4.2 Modelos de representación

Se va a detallar los distintos modelos de representación de gestos [Figura 4.4] que son proporcionados por los tipos de dispositivos descritos anteriormente. Estos modelos son:

→ **Modelo 3D**: Define la información espacial en 3D de las partes del cuerpo que detecte dicho dispositivo. Se subdivide en:

→ Modelos volumétricos: representan con gran exactitud el tamaño y forma de la mano o brazo, el cual se visualiza de forma muy similar a la realidad (tienen una mayor computación).

→ Modelos esqueléticos: modelo simplificado con la información más relevante de la mano o brazo. Como por ejemplo: huesos, dirección, posición, ángulos, etc. Necesita una menor computación (**Aquí se encuentra el dispositivo Leap Motion que será utilizado en este TFG**, aunque es posible diseñar un modelo volumétrico con software de programación gráfica en 3D).

- **Modelo basado en la apariencia (2D)**: No define la información espacial del objeto, sino que se basa en la forma y tamaño de las manos y brazos que se pueden apreciar en las imágenes captadas. Se pierde la información relativa a la profundidad, ya que es una imagen plana. Los modelos más usados son:
  - → Basado en el color: usa marcas de color en el cuerpo para seguir el movimiento.
  - → Basado en la silueta: usa las propiedades geométricas de la silueta de la mano, brazo, etc.
  - → Basado en el contorno: se centra solo en el contorno de la mano, brazo, dedos, etc.
  - → Basado en el movimiento: hace un seguimiento del movimiento de los píxeles que capta en la imagen para sacar un patrón de movimiento.

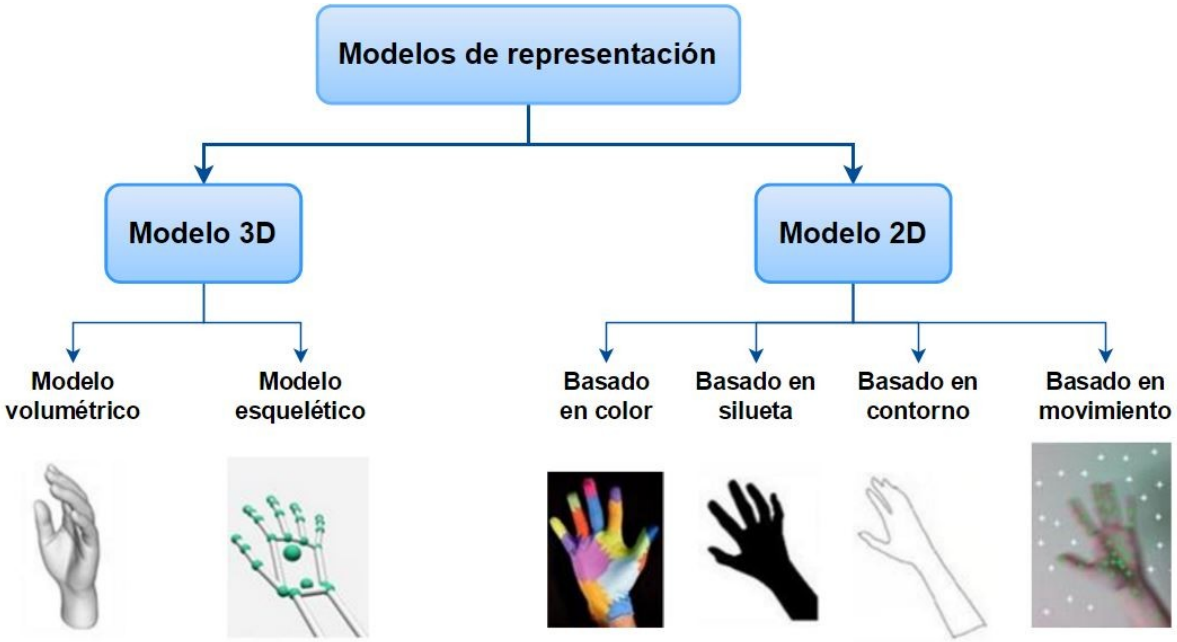

Figura 4.3. Esquema de modelos de representación.

4.3. Leap Motion

La compañía LeapMotion fue fundada en el año 2010 por Michael Buckwald y David Holz. El primer dispositivo comercial fue lanzado por primera vez en Julio del 2013 bajo el nombre “The Leap”. Aunque sus inicios y primerísimos prototipos se remontan al año 2008, cuando su cofundador David Holz estaba estudiando un doctorado en matemáticas.

Figura 4.4 Dispositivo Leap Motion actual.

Figura 4.5. Evolución Leap Motion

Se va a distinguir y analizar a continuación el hardware y el software del dispositivo Leap Motion, para una mejor estructuración del mismo. Se explicarán las características físicas y de uso del dispositivo y se analizará su SDK (Software Development Kit). Además de explicar cómo está constituido este SDK y su modo de funcionamiento, se explicarán las clases que principalmente serán usadas en este trabajo. Serán resaltadas además las facilidades que incorpora para diseñar aplicaciones en tiempo real, y la información que puede obtener de dedos y manos en cada *frame* recibido.

4.3.1. Hardware

Se trata de un pequeño dispositivo con forma rectangular. Tiene unas dimensiones de 75x25x11 mm (largo, ancho y alto) y un peso aproximado de 50g. Su precio es de unos 90€. El dispositivo tiene su alimentación por USB 2.0 o 3.0 a través del cual manda la información al driver del PC al que está conectado. Dependiendo de si se usa el USB 2.0 o el 3.0 y del rendimiento del PC, se tendrán diferentes velocidades en la transmisión de la información. Pero puede permitir un *framerate* de hasta 300 fps (frames per second) con una precisión de 0.01 mm.

Se sitúa principalmente delante del ordenador, apoyado en una superficie preferiblemente plana. De esta manera, crea un espacio de interacción en 3D invisible [Figura 4.6] llamado AirSpace ó Interaction Box, con forma piramidal invertida de aproximadamente unos 0.23 m3.

Figura 4.6. Interaction Box creado por Leap Motion

Este Interaction Box será el espacio de trabajo del dispositivo. Hace un seguimiento de las manos, dedos, y objetos "*pointables*" (similares a un dedo) situados dentro de este espacio con una precisión muy alta. Aquí se encuentra la primera gran diferencia con respecto a Kinect, el cual hace un seguimiento principalmente de todo el cuerpo en dimensiones de una habitación.

Internamente [Figura 4.7] el dispositivo cuenta con dos cámaras, tres LEDs de infrarrojos, un microcontrolador y un controlador USB.

- → Cada **cámara** cuenta con un sensor monocromático CMOS sensible a la luz infrarroja, capaz de trabajar a una velocidad de hasta 300 fps.
- → Los **LEDs** iluminan la Interaction Box por inundación. Esta iluminación va variando para asegurar una igualdad en resolución de imagen. Están separados por barreras de plástico asegurando así una iluminación uniforme.
- → El **microcontrolador** es un circuito integrado que funciona como BIOS. Controla el dispositivo y capta y envía la información de los sensores al driver o controlador del PC.
- → El **controlador USB** es de alta velocidad y permite que el PC reconozca al dispositivo.

Figura 4.7. Despiece del dispositivo Leap Motion

En la parte superior, cuenta con un plástico negro que funciona como filtro óptico dejando transmitir únicamente la luz infra-roja.

El uso de la CPU que hace el dispositivo es relativamente bajo para la alta precisión que se obtiene. Llega a ser de aproximadamente un 12% tras cierto tiempo empleando *tracking* de manos y dedos con un procesador i5. Los drivers y tecnología usada en el *tracking* para obtener resultados tan potentes sigue siendo un misterio guardado por la empresa. En la página oficial podemos encontrar los **requisitos mínimos** para el uso del dispositivo:

- Windows 7 + ó Mac OS X 10.7 +
- Procesador AMD Phenom II ó Intel Core i3/i5/i7
- 2 GB RAM
- Puerto USB 2.0/3.0
- Conexión a internet

4.3.2. Software

Leap Motion SDK es soportado por los sistemas operativos Windows, Macintosh y Linux. Cada vez son más las aplicaciones que aparecen en mercado cuyo control viene dado por el dispositivo Leap Motion. La empresa cuenta con un App Store [Figura 4.8] el cual se conecta con el "Leap Motion App Home" [Figura 4.9] que se añade como aplicación de escritorio en el ordenador al instalar el controlador del dispositivo. Aquí se pueden descargar aplicaciones gratuitas o de pago que son subidas por los desarrolladores. Son muchas las posibilidades que ofrece, como por ejemplo: control del escritorio Windows, jugar a videojuegos, instrumentos de aire, etc. Además están saliendo diversos plugins que permiten por ejemplo que sea posible navegar por Chrome usando el Leap Motion, o moverse por Google Earth. La interacción con imágenes médicas anatómicas en 3D, integración en smartphones, integración con gafas de realidad virtual…son algunos de los objetivos con vista al futuro de los desarrolladores que ya se pueden contemplar. Por tanto, como se puede deducir, el dispositivo cuenta con un SDK muy flexible y con el que se pueden desarrollar aplicaciones muy diversas.

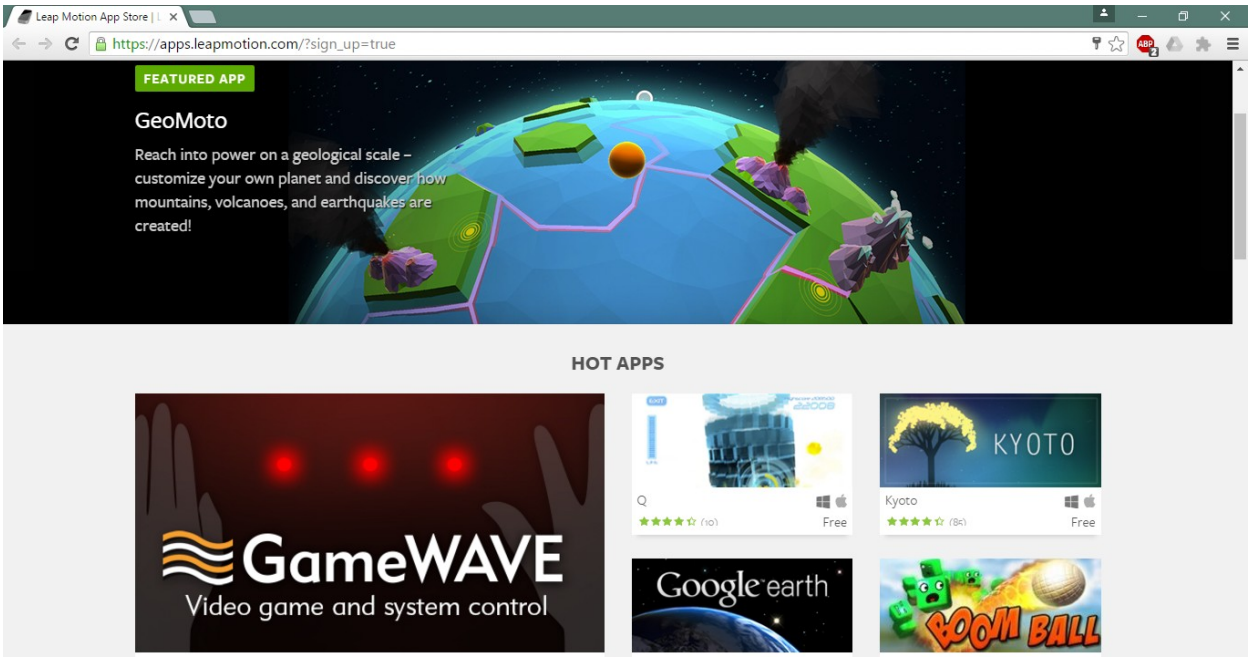

Figura 4.8. Página principal de la App Store de Leap Motion

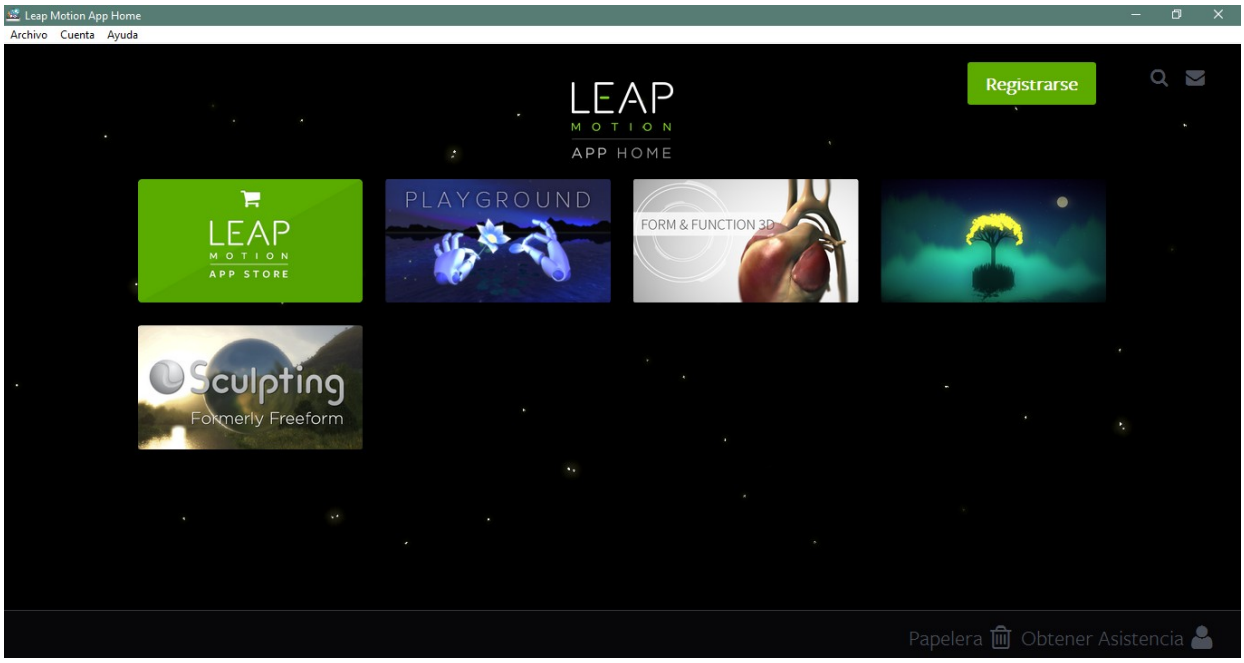

Figura 4.9. Pantalla principal de Leap Motion App Home

Para la puesta en marcha del dispositivo, es necesario descargar el "**LeapDeveloperKit**" para el correspondiente sistema operativo (Windows en el caso de este TFG) directamente de la página oficial de Leap Motion. En el paquete descargado se **incluye un instalador del controlador y una carpeta que contiene el SDK**. Al instalar el controlador, será instalado también el "Leap Motion App Home" citado anteriormente. El controlador cuenta principalmente con dos aplicaciones:

- **Panel de control de Leap Motion**: permite cambiar prácticamente cualquier opción del dispositivo. Como por ejemplo: área de interacción del dispositivo, mecanismo de ahorro de energía, recalibrar el dispositivo, resolución o informar acerca de problemas, configuración de rastreo, actualizaciones, etc. [Figura 4.8]. Además cuenta con un apartado "Acerca de" donde se puede encontrar información relativa a la versión del SDK y enlaces a las páginas web oficiales.

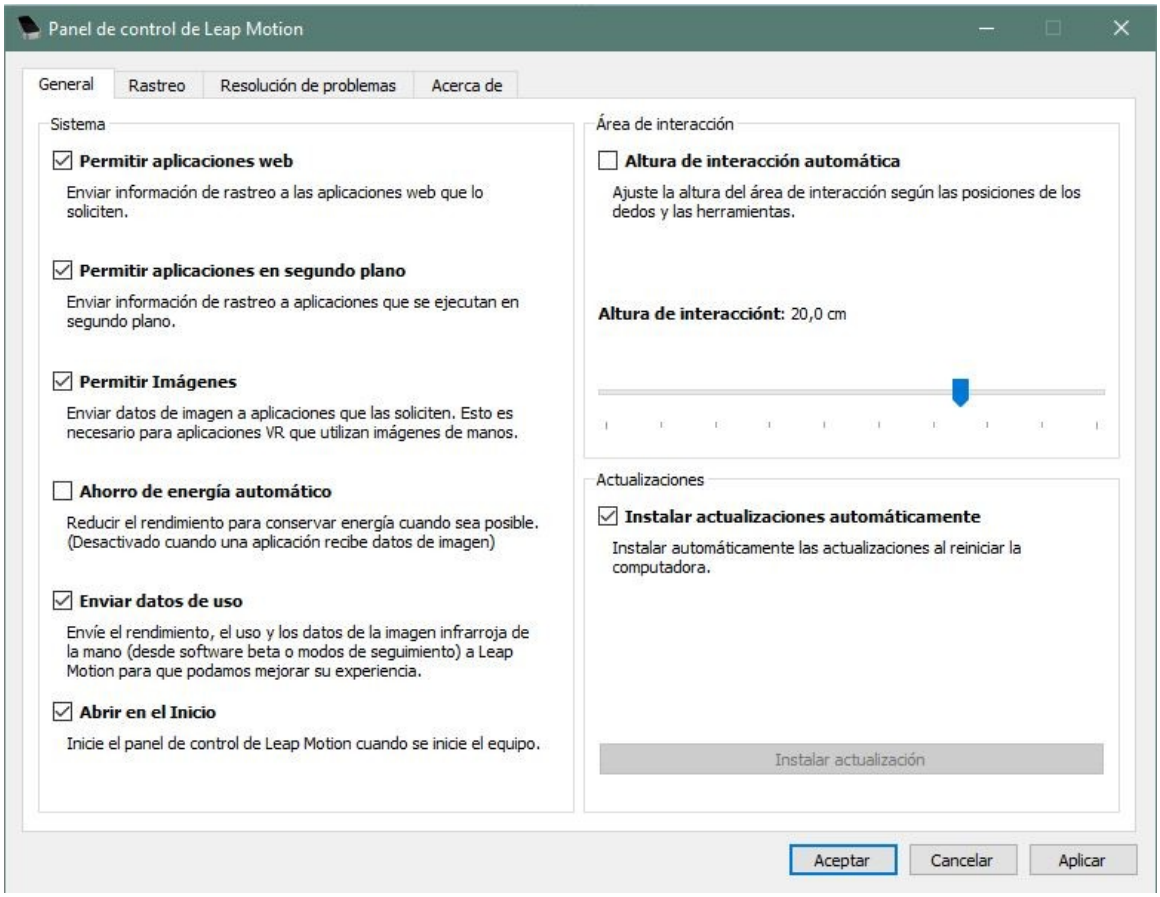

Figura 4.10. Panel de control de Leap Motion

- **Visualizador**: muestra un espacio gráfico 3D con un sistema de coordenadas cartesianas (x, y, z). El eje "x" pertenece a la parte horizontal, el "y" va verticalmente desde el sensor y el "z" perpendicular a la pantalla. En este gráfico 3D se monitorizan las manos, dedos y elementos *"pointables"* [Figura 4.9] siendo representadas y desplazándose tal y como el usuario las mueve. Además, ofrece la posibilidad de visualizar información relativa al muestreo, coordenadas, dibujar la Interaction Box, cambiar el tipo de representación, etc.

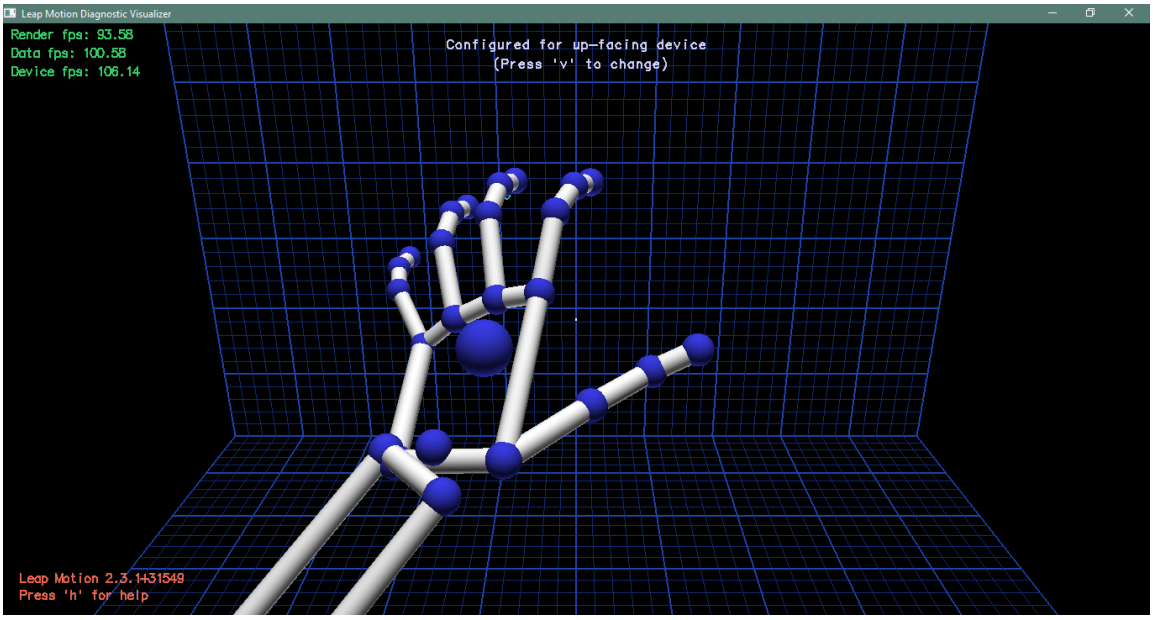

Figura 4.11. Visualizador de diagnóstico de Leap Motion

Para crear la aplicación que se quiere desarrollar en este trabajo, es fundamental conocer el SDK de Leap Motion. Sus clases, lenguajes, modo de trabajo, etc. Este SDK permite una gran variedad de lenguajes de programación entre los que están C++, C#, Unity, Objetive-C, Java, Java Script, Python y Unreal Engine. Para esta aplicación vamos a usar el lenguaje **C++.** Como bien se ha dicho anteriormente, el dispositivo crea un espacio 3D y tiene la información correspondiente a manos, dedos y objetos *"pointables"*. Para posicionarlos usa un sistemas de coordenadas cartesianas (x, y, z) [Figura 4.10] centrando el origen en el medio del dispositivo.

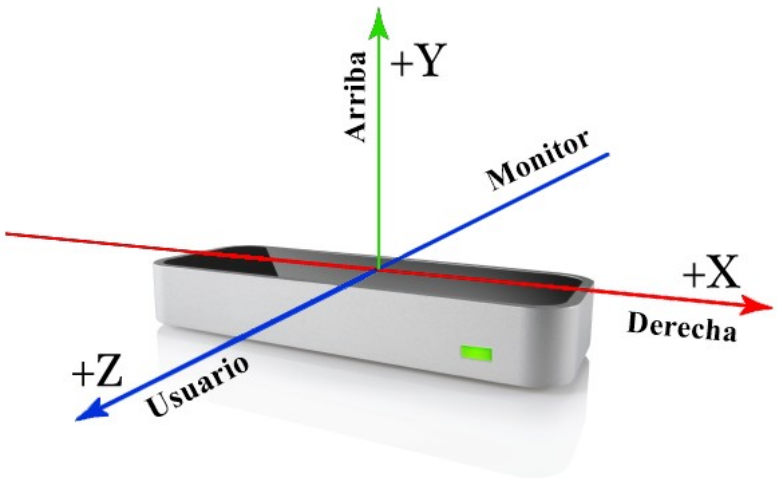

Figura 4.12. Sistema de coordenadas de Leap Motion

A continuación, se van a describir los objetos y clases esenciales de la API (Application Programming Interface) del dispositivo que van a ser fundamentales para poder programar la aplicación. Además se detallarán sus propiedades más características de las cuales algunas serán usadas en esta aplicación. Es importante conocer qué información es capaz de capturar el dispositivo para conocer las limitaciones y posibilidades que se tendrán a la hora de diseñar el sistema que se quiere implementar.

FRAMES

Es el contenedor principal de la información que captura Leap Motion. Cada *frame* contiene, en un instante determinado, la siguiente información [Figura 4.13]:

- ***Hand****:* Información de las manos, brazos y herramientas *"pointables"* (como dedos, por ejemplo).
- ***Finger*s**: Información de los dedos.
- ***Bones****:* Información de los huesos de los dedos.
- Posición de las articulaciones.
- Imágenes de las cámaras.

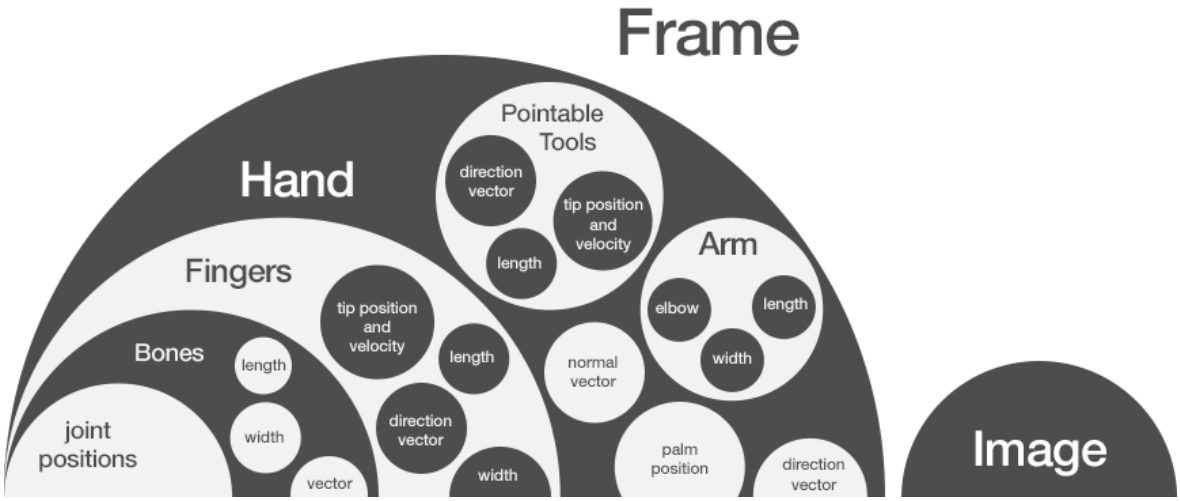

Figura 4.13. Información que contiene cada Frame

Hay que resaltar que cada objeto se asocia con otro objeto superior, siendo el más superior el *Frame*. Esta jerarquía es uno de los aspectos más importantes a tener en cuenta. Así para poder acceder por ejemplo a *Fingers* o *Pointable Tools*, se debe hacer a través del objeto

Hand dentro de un *Frame* determinado. Esto recuerda a una muñeca matrioska, ya que cada objeto está contenido dentro de otro superior, siendo el más superior el *Frame*.

Además de esta información (que es más relativa a manos y dedos), cada *Frame* también contiene información de identificación de *Frame*, *timestamp*, etc. El software analiza la información de un *Frame* actual y el anterior y sintetiza el eje de rotación, ángulo de rotación, matriz de rotación, factor de escala y translación para la monitorización de las manos. Por tanto, se basa en el cambio continuo de dicha información.

HANDS

Hand es la principal entidad de información (después de *frame*) ya que como se ha visto, a través de ella se accederá a los objetos inferiores. El controlador tiene un modelo de la mano humana y asocia la información de sus sensores con este modelo. Esto permite hacer un seguimiento de la posición de los dedos hasta cuando el dedo no es completamente visible. Puede distinguir entre mano derecha e izquierda. Para cada *hand*, se tiene los siguientes atributos:

- Posición del centro de la palma (***Palm Position***) en milímetros. Vector.
- Velocidad de la palma (***Palm Velocity***) en milímetros/segundo.
- Vector perpendicular a la palma (***Palm Normal***). Vector.
- Dirección de la mano (***Direction***). Vector.
- Información de la esfera que describe la postura de la mano (***grabStrength, pinchStrength***)
- Factores del movimiento como escala, rotación y translación (***Motion Factors***)

Se puede acceder a la orientación de la mano usando la dirección de la mano y los vectores normales. Su postura viene dada por una la información de una esfera que "encaja" en la mano [Figura 4.14].

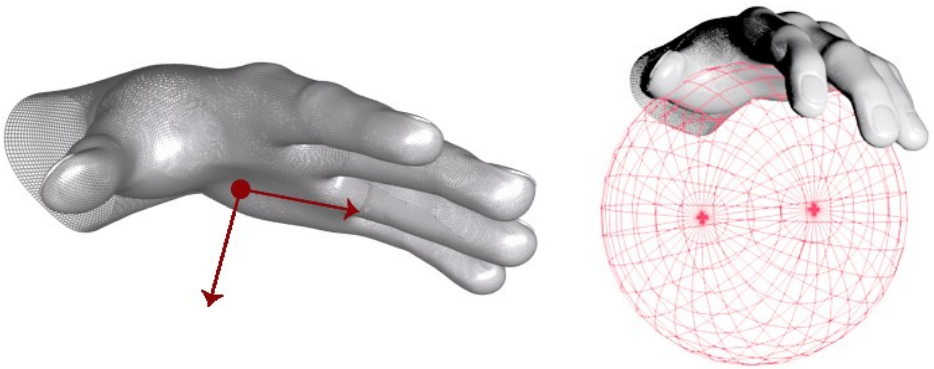

Figura 4.14. Algunos de los atributos del objeto Hand.

FINGERS (Pointables)

Tanto *Finger* como *Tool* se encuentran dentro de la clase *Pointable*, ya que ambos son objetos que pueden considerase "*pointables*". Esta aplicación se centra solo en *Finger,* ya que en la realización de gestos pertenecientes a la lengua de signos sólo son usadas las manos y dedos y no ningún objeto con forma alargada.

A través de *Hand* se accede a los *Fingers* asociados con esa determinada mano, la cual está disponible en un *Frame* concreto. Al acceder a los dedos de una mano, se devuelve un vector que contiene los 5 dedos de la mano. Será posible acceder al dedo en concreto que sea requerido [Figura 4.15] bajo la función *Finger::type()* usando los valores de la siguiente enumeración:

- `TYPE_THUMB = = 0` (Dedo pulgar)
- `TYPE_INDEX = = 1` (Dedo índice)
- `TYPE_MIDDLE = = 2` (Dedo corazón)
- `TYPE_RING = = 3` (Dedo anular)
- `TYPE_PINKY = = 4` (Dedo pequeño)

Los atributos que se tienen del objeto *Finger* son los siguientes [Figura 4.15]:

- Posición del extremo (***Tip position***) del dedo en milímetros. Vector.
- Velocidad del extremo (***Tip velocity***) del dedo en milímetros/sg.
- Posición estabilizada del extremo (***Stabilized tip position***) del dedo usando la velocidad y la posición en el *frame* anterior. Vector.
- Dirección (***Direction***) del dedo. Vector.
- Longitud (***Length***) del dedo.
- Anchura (***Width***) del dedo.

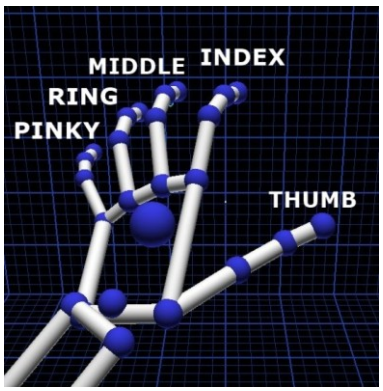

Figura 4.15. Algunos de los atributos del objeto Finger.

BONES

De forma similar a como se obtiene la información de los dedos en el apartado anterior, se accede a la información de los huesos de la mano. Así, para acceder a *Bone*, habrá que hacerlo a través de *Frame/Hand/Finger*. Y análogamente a los dedos, se devuelve un vector que contiene los cuatro huesos para cada dedo. En este caso, será posible acceder al hueso en concreto que sea requerido usando la función *Bone::type()* siendo su enumeración, ordenados de la base al extremo del dedo, la siguiente:

- `TYPE_METACARPAL = = 0` (Hueso que conecta con la muñeca).
- `TYPE_PROXIMAL = = 1` (Hueso de la base del dedo).
- `TYPE_INTERMEDIATE = = 2` (Hueso entre el extremo y la base del dedo).
- `TYPE_DISTAL = = 3` (Hueso en el extremo del dedo).

NOTA*: *El dedo pulgar no tiene hueso metacarpiano (type == 0), así que contiene un valor de longitud igual a cero.*

Los atributos que posee el objeto Bone son los siguientes [Figura 4.16]:

- Dirección del hueso (***Direction***).
- Longitud del hueso (***Length***).
- Anchura del hueso (***Width***).
- El punto medio del hueso (***Center***).

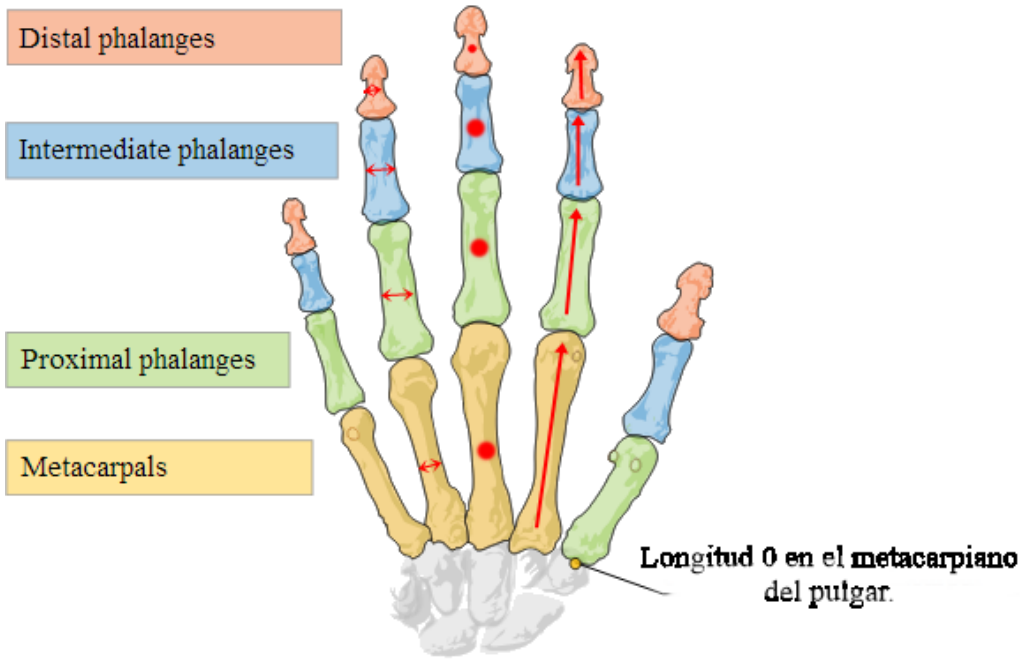

Figura 4.16. Algunos de los atributos del objeto Bone.

Hay que destacar que si el dispositivo permite un *framerate* de hasta 300 frames/segundo, está permitiendo un seguimiento prácticamente total y continuo de las manos. Si a esto se le suma su alta precisión, es apreciable su gran potencial y su ventaja en el desarrollo de aplicaciones en tiempo real.

Esta es por tanto [Figura 4.17], de forma resumida en forma de imagen, la información y componentes que se pueden captar de una mano con el dispositivo Leap Motion y que será actualizada para cada *frame*.

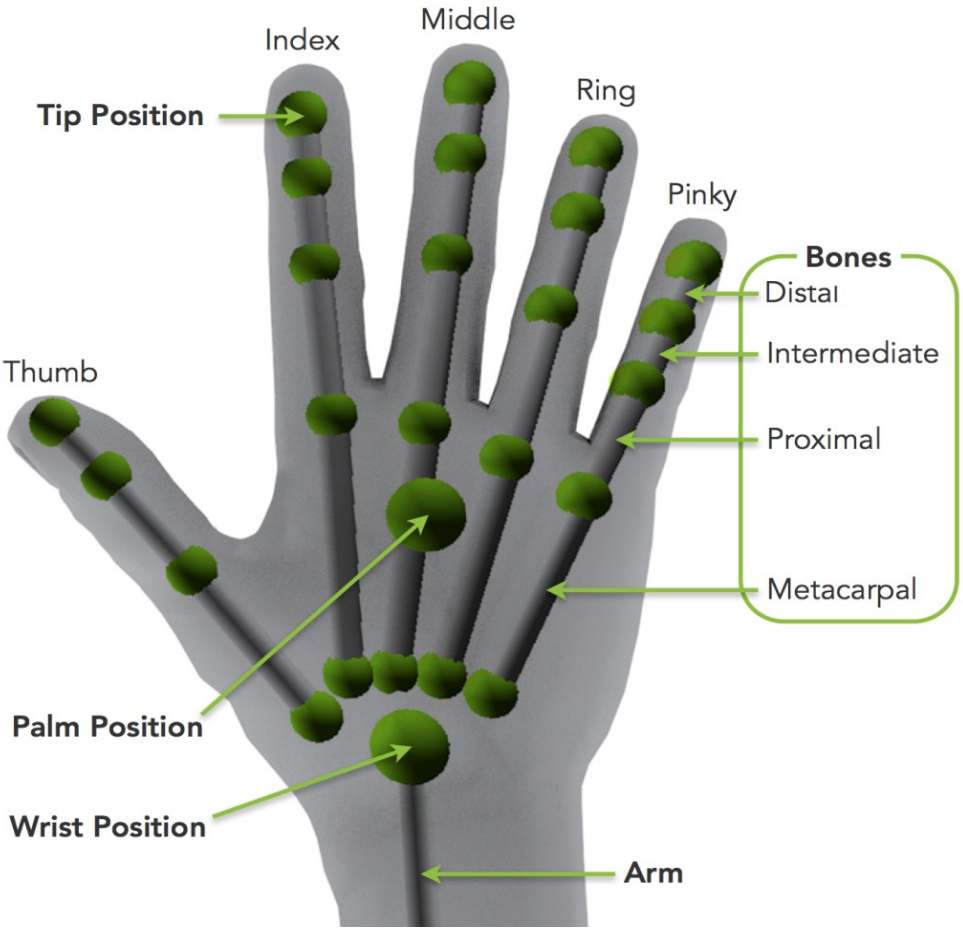

Figura 4.17. Resumen de los componentes e información capturada por Leap Motion.

5. DESARROLLO

En el siguiente capítulo se centra el desarrollo de la aplicación. Para llegar a tener una aplicación que sea adecuada al objetivo de este TFG y que tenga un correcto funcionamiento, se propone la siguiente metodología [Figura 5.1] a seguir en su desarrollo la cual constará de una serie de fases que serán claves para tal fin:

Fase 1) Se explicaran los requisitos o módulos deseados en el sistema que se pretende desarrollar. Es muy importante tener claras las funcionalidades que se quieren tener en la aplicación final antes de comenzar a programarla. Por tanto, se requiere una buena estructuración y división del sistema en módulos o requisitos.

Fase 2) Serán analizadas las herramientas y librerías ya disponibles con licencia MIT que puedan ser de utilidad para esta aplicación, dejando claro qué parte del sistema será recopilado (en caso de usar alguna de ellas) y qué parte será desarrollada en este trabajo. Dejando claro siempre el por qué de dicha elección.

Fase 3) Se va a explicar detalladamente el algoritmo de reconocimiento que se pretende usar, y su apropiada variante para la implementación en esta aplicación. Para llegar a entender el funcionamiento del sistema y tener en cuenta especialmente el tipo y la cantidad de información que se debe capturar del dispositivo, hay que conocer cómo funciona el algoritmo. Ya que este utilizará dicha información para el reconocimiento.

Fase 4) Teniendo en cuenta las prioridades de un diccionario personalizable, y una rápida computación, hay que decidir qué información será capturada por el dispositivo Leap Motion. Ya que este dispositivo da la posibilidad de capturar una gran cantidad de información, será necesario hacer un estudio para determinar qué información será la necesaria para el desarrollo de esta aplicación y por qué.

Fase 5) Se determinará la lógica del sistema. Una vez se hayan decidido las funcionalidades y herramientas con las que contará la aplicación, es necesario hacer un estudio de la lógica del sistema antes de comenzar a programarlo. Con esto se asegura una buena integración e interconexión de las diferentes librerías, algoritmos y herramientas que van a constituir la aplicación. Se pretende conseguir robustez,

dejando de lado posibles bucles infinitos o caminos sin salida debidos a la falta de interconexión entre los módulos.

Fase 6) Por último, se procederá a programar dicha aplicación. Se nombrará el IDE utilizado para tal fin y algunas de sus características. Por otro lado, y como aspecto más importante, se explicarán los elementos que conforman el proyecto. Cuáles han sido recopilados y cuáles desarrollados para este trabajo. Se explicará las distintas funciones que han sido programadas y su modo de funcionamiento. Además se explicará el menú que presentará la aplicación final ofrecida al usuario.

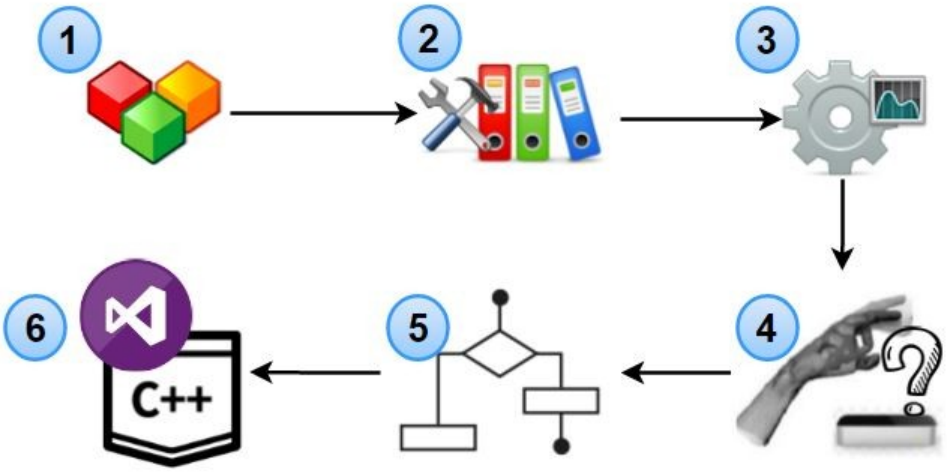

Figura 5.1. Metodología para el desarrollo de la aplicación.

5.1. Módulos del sistema

Al dividir el sistema en varios módulos o requisitos, se asegura una mejor integridad del sistema. Ya que será más fácil detectar y corregir un fallo si únicamente afecta a un módulo en concreto. Además será más fácil analizar su correcta implementación. Son necesarios por tanto, módulos que cumplan con los siguientes requisitos:

- ✓ **Módulo grabador de gestos**: Clave fundamental si se quiere un diccionario personalizable. El usuario podrá grabar sus gestos. Para ello, se guardará la información capturada por el dispositivo Leap Motion mientras se esté realizando dicho gesto. El usuario podrá nombrar los gestos para identificarlos a la hora de ser traducido.

- ✓ **Módulo para guardar el diccionario grabado**: crear una base de datos que contenga la información de todos los gestos grabados para poder cargarlos y usarlos en el sistema en cualquier momento.
- ✓ **Módulo de entrenamiento**: hay que entrenar el sistema con un algoritmo de clasificación para que pueda detectar cuál de los gestos del diccionario se ha realizado y traducirlo correctamente.
- ✓ **Módulo reconocedor**: es fundamental que se tenga un correcto funcionamiento en este módulo para que el sistema no confunda unos gestos con otros, ya que se estará intentando mantener una conversación con otra persona.
- ✓ **Módulo de audio**: Consiste en un conjunto de audios que reproducen el nombre del signo una vez sea reconocido. Se ha pensado para agilizar aún más la comunicación, ya que así el hablante que desconoce la lengua de signos no tendría que leer la pantalla para ver la traducción.

Estos serán en un principio los requisitos y módulos mínimos que se han planteado para que el sistema tenga una buena estructuración. Se irán añadiendo o modificando módulos en función de las facilidades o contratiempos que vayan surgiendo en su desarrollo.

5.2. Implementación de herramientas.

Se van a analizar algunas de las herramientas, programas, toolkit, librerías, etc. que se han encontrado y que podrían cubrir con alguna de las necesidades de esta aplicación.

5.2.1. Elección de herramientas recopiladas.

Existen varias aplicaciones que permiten grabar gestos usando el dispositivo Leap Motion. A continuación se detallará las que han parecido más interesantes y se decidirá si alguna puede ser adaptada a la aplicación que estamos desarrollando, o si por lo contrario, sería mejor desarrollarla.

- **JestPlay**: Escrito en Java Script, permite grabar el movimiento de la mano en un archivo del tipo JSON ó BVH. Una vez guardado, se puede abrir y ver su reproducción en 3D. Desarrollado exclusivamente para dispositivo Leap Motion.

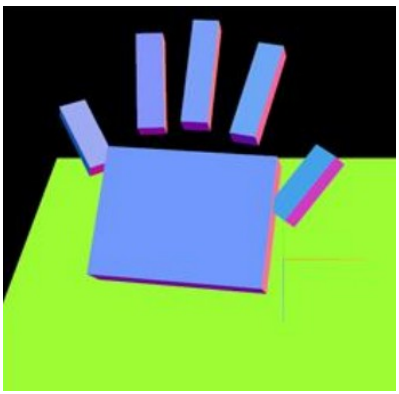

Figura 5.2. Captura de JestPlay

- **Playback**: Escrito en Java Script, permite grabar y visualizar el gesto grabado. Además se puede guardar el gesto en un archivo JSON. Funciona a través del navegador web y está desarrollado exclusivamente para el dispositivo Leap Motion.

Figura 5.3. Captura de Playback.

✓ **LeapTrainer.js:** Escrito en Java Script. Permite grabar varios gestos y posteriormente reconocerlos usando para ello un algoritmo algebraico que los compara. Es fácil de implementar en una aplicación web. Funciona a través del navegador y está desarrollado exclusivamente para Leap Motion.

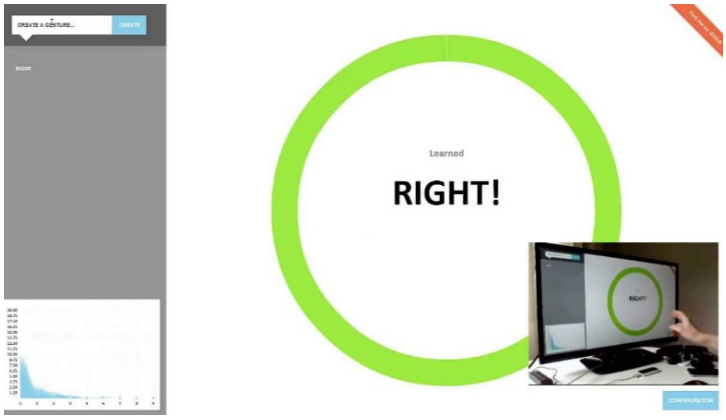

Figura 5.4. Captura de LeapTrainer.js

- ✓ **Gesture Recognition Toolkit (GRT):** Escrito en C++. Es una toolkit que permite crear un sistema de reconocimiento de gestos en tiempo real usando la información de cualquier tipo de sensor (no solo Leap Motion). Dispone de varios algoritmos de reconocimiento, procesamiento de la información y numerosas funciones extras. Es posible su integración en un proyecto escrito en C++.

Figura 5.5. Logotipo de la toolkit GRT

A continuación, se van a analizar las ventajas y desventajas que tendrían estas herramientas frente a la posibilidad de desarrollar una desde cero. Será comparada principalmente: la posibilidad de decisión en la información capturada de manos y dedos, el funcionamiento sin necesidad del navegador, el lenguaje de programación, la capacidad de reconocer gestos y el hecho de estar ya desarrollado.

	JestPlay	**Playback**	**LeapTrainer**	**GRT**	**Desarrollar una desde cero**
Decisión de información a capturar	NO (+0)	NO (+0)	NO (+0)	SÍ (+1)	SÍ (+1)
Funciona exclusivamente en navegador	NO (+1)	SÍ (+0)	SÍ (+0)	NO (+1)	NO (+1)
Lenguaje	JS (+0)	JS (+0)	JS (+0)	C++ (+1)	C++ (+1)
Reconoce gestos	NO (+0)	NO (+0)	SÍ (+1)	SÍ (+1)	SÍ (+1)
Desarrollado	SÍ (+1)	SÍ (+1)	SÍ (+1)	SÍ (+1)	NO (+0)
TOTAL	**2**	**1**	**1**	**5**	**4**

Tabla 5.6. Comparación de las diferentes herramientas.

Como se puede observar, la herramienta que reúne más cualidades y presenta más ventajas para satisfacer las necesidades del sistema es la Toolkit **GRT.**

El aspecto más importante de los comparados anteriormente que ha sido clave para la elección de esta herramienta ha sido la posibilidad de elección en la información a capturar. La aplicación que se pretende desarrollar es algo más específica que el simple hecho de reconocer el movimiento, y por eso se necesita tener una mayor diferenciación entre la posición de manos y dedos. Para ello es fundamental poder decidir qué información se quiere capturar. El hecho de que no funcione exclusivamente en el navegador, permite dar mayor integridad y robustez al sistema al no depender de fuentes externas. Que el lenguaje C++ sea una ventaja frente a Java Script, se debe a un mejor conocimiento del mismo.

Además, esta toolkit cuenta con una licencia MIT, lo que permite utilizarla con unas condiciones mínimas para el desarrollo de software libre.

5.2.2. Adaptación de la toolkit GRT.

En primer lugar, se va a realizar una pequeña descripción de esta toolkit y a continuación, se explicará cómo ha sido modificada y adaptada para su implementación en la aplicación que se está desarrollando.

Se trata de una Toolkit desarrollada por el ingeniero Dr. Nick Gillian; especializado en el reconocimiento de gestos y en *machine learning.* Esta librería esta bajo licencia MIT, por lo que impone unas limitaciones mínimas en su reutilización y posee una buena compatibilidad de licencia. No tiene licencia copyright, por tanto permite su modificación. Esta librería ha sido desarrollada para:

- ✓ Ser fácil de integrar en proyectos C++ existentes.
- ✓ Ser compatible con cualquier tipo de sensor.
- ✓ Poder entrenar el sistema con gestos propios.
- ✓ Ser extensible y adaptable a procesamiento y algoritmos propios.

Además, proporciona principalmente dos grandes y potentes herramientas útiles en la creación de un sistema de reconocimiento:

- Algoritmos de clasificación: Contiene numerosos algoritmos especializados para entrenar el sistema de reconocimiento. Depende de si está basado en gestos estáticos, dinámicos, el tipo de información que usamos, etc. será conveniente usar

uno u otro. Entre los algoritmos de los que dispone se encuentran: AdaBoost, Decision Tree, Dynamic Time Warping, Hidden Markov Models, Support Vector Machine, etc.

- Procesamiento de la información: contiene numerosos mecanismos de pre-proceso, post-proceso y extracción de la información para poder elegir el que mejor se adapte al sistema. Entre ellos se encuentran: filtro paso bajo y paso alto, filtro promedio de movimiento, FFT, derivada, análisis de las componentes principales, etc.

En la página web oficial se puede encontrar una gran cantidad de documentación acerca de la API que utiliza, tutoriales, foros y otras opciones que han sido de gran ayuda para aprender a implementar esta toolkit en la aplicación que se está desarrollando. En esta documentación pueden encontrarse los elementos que se consideran que debe de tener un sistema de reconocimiento [Figura 5.7] y algunos consejos acerca de su diseño. Estos elementos son los siguientes:

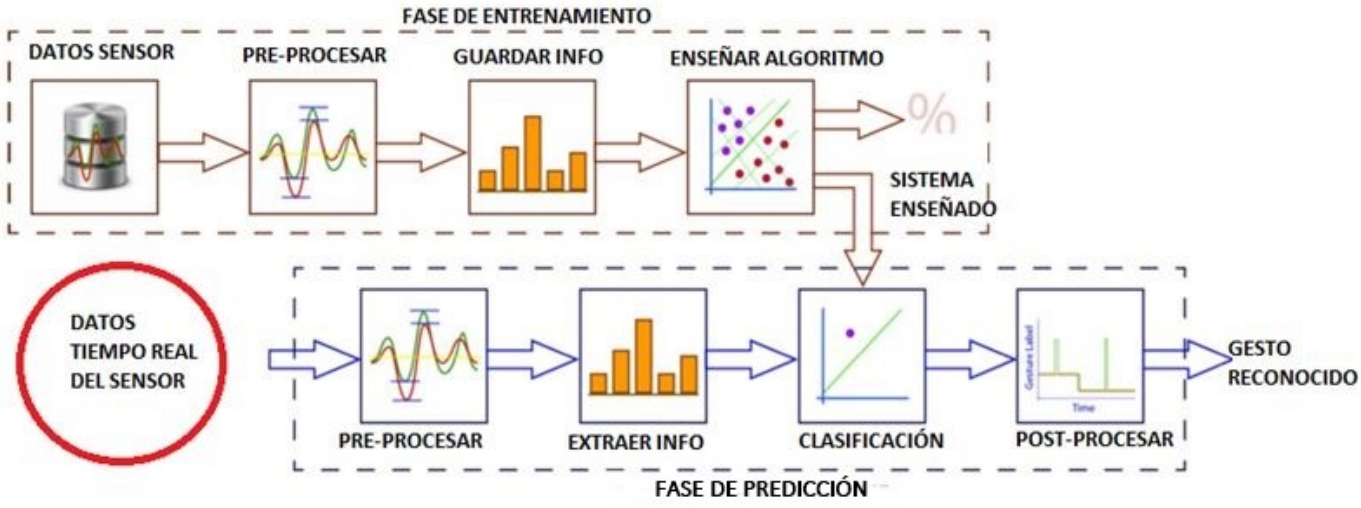

Figura 5.7. Estructura del sistema de reconocimiento considerado por la toolkit GRT.

Como se puede observar, el sistema tendrá dos fases: fase de entrenamiento y fase de predicción.

Se va a analizar ahora cómo ha sido modificada y adaptada esta toolkit para la aplicación que se está desarrollando. Para ello, se analizarán cada uno de los elementos propuestos y se decidirán cuáles serían los adecuados para el sistema de reconocimiento de gestos que se quiere desarrollar y cuáles han de ser las modificaciones que deben realizarse

para darle un correcto uso en este trabajo. Teniendo en cuenta que la finalidad es reconocer gestos pertenecientes a la lengua de signos y que el dispositivo de captura usado es el Leap Motion, se puede determinar lo siguiente:

1) El pre-procesamiento de la información puede ser obviado. Este dispositivo capta información relativa principalmente al posicionamiento y dirección de la mano y dedos en un espacio 3D, lo que se traduce en valores de coordenadas (x, y, z). Al no tener una frecuencia de trabajo determinada, un periodo y otras características relativas a una señal típica (por ejemplo sonidos, luminancia, etc.) aplicar filtros paso bajo, paso alto, transformadas o derivadas se convierte en algo innecesario.

2) Un requerimiento básico como bien ya se ha dicho, es el de poder guardar el diccionario grabado para poder usarlo posteriormente sin necesidad de volver a grabarlo. Por tanto este será un elemento fundamental el cual habrá que adaptar.

3) Se debe determinar el algoritmo de clasificación que sería apropiado usar. Para ello, se tendrán en cuenta especialmente tres aspectos:

- ✓ Se puede considerar que todos los gestos que se van a emplear en la lengua de signos son dinámicos ya que para realizar cualquier gesto hay que realizar cambios en las manos, y no cabo esperar encontrarse la mano inmóvil y bien colocada. Por tanto es necesario un algoritmo que trabaje con gestos dinámicos.
- ✓ No puede haber dependencia con la velocidad de realización del gesto ni con el tiempo empleado en realizarlo, ya que unas personas pueden tardar más o menos que otras en realizar el gesto completo.
- ✓ De igual manera, no puede haber dependencia con la longitud. La información captada no tiene porque ser de igual longitud a la que comparamos, precisamente por el tiempo y velocidad empleada.

Estas consideraciones fundamentales, hacen que se elija el algoritmo **Dynamic Time Warping (DTW)** para esta aplicación, ya que este algoritmo mide la similitud entre dos señales que pueden variar en velociad, tiempo y/o longitud. Más tarde se explicará con detalle cómo funciona este algoritmo, y se verá la modificación necesaria del mismo para esta aplicación.

4) Sería bueno incorporar un post-procesamiento de la información tras ser clasificada. Esto serviría por ejemplo para descartar falsos positivos e impedir confusión en

la comunicación. Para ello, se puede considerar una correcta traducción (reconocimiento) siempre y cuando se supere un umbral al restar la similitud del gesto traducido con la del siguiente gesto de mayor similitud.

Tras estas aclaraciones, y tras realizar las modificaciones y adaptaciones que han sido necesarias, es posible determinar como será el **sistema de reconocimiento de gestos que se empleará en la aplicación** [Figura 5.8]. La estructura diseñada para este sistema será la siguiente:

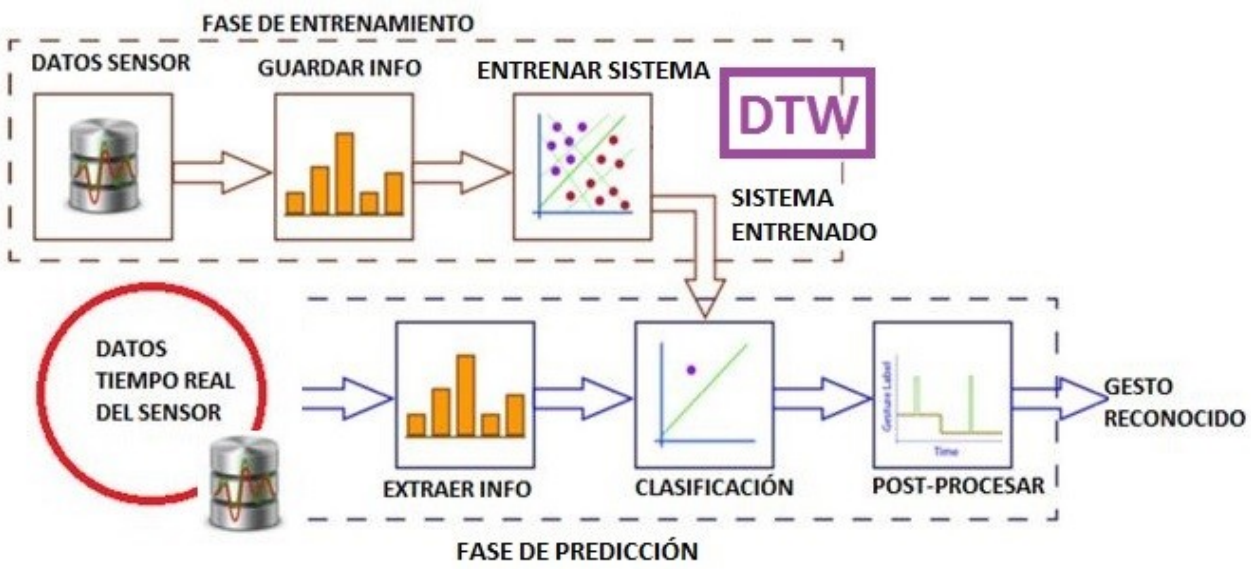

Figura 5.8. Estructura que tendrá el sistema de reconocimiento de gestos desarrollado.

Se tiene por tanto una nueva división en módulos o requisitos siguiendo esta estructura:

- **Módulo de extracción de información** deseada del Leap Motion (tanto en fase de entrenamiento como en fase de predicción)
- **Módulo grabador de gestos** usando dicha información.
- **Módulo para guardar diccionario** de gestos para futuro uso.
- **Módulo de entrenamiento** del sistema de reconocimiento (con algoritmo DTW).
- **Módulo reconocedor** gestos.
- **Módulo de post-procesamiento** para dar como valido el gesto reconocido.
- **Módulo de audio,** para el signo reconocido**.**

Se han añadido dos módulos más a los expuestos en el apartado 5.1 que harán una mejor división del sistema con la finalidad de ganar robustez.

5.3 Aplicación del algoritmo Dynamic Time Warping.

Este apartado se centra principalmente en el estudio del funcionamiento del algoritmo Dynamic Time Warping, que será usado en la aplicación que se está desarrollando. Además, se detallará la adaptación de dicho algoritmo que será finalmente la usada para este trabajo. Ha sido imprescindible el estudio de este algoritmo para su aplicación, tanto para determinar la manera de enviar la información a dicho algoritmo como el modo de extraerla.

DTW es un potente algoritmo para medir la similitud entre dos secuencias temporales que pueden variar en el tiempo, velocidad y/o longitud. Por ejemplo, similitudes en la trayectoria entre dos coches pueden ser detectadas usando DTW, aunque una de ellos se mueva más rápido que el otro. Este algoritmo fue usado originalmente para reconocimiento de voz, pero se puede aplicar a secuencias temporales de video, audio, gráficos, datos, etc. Cualquier secuencia temporal puede ser analizada por DTW.

Los gestos temporales que se emplearán en la aplicación, se pueden definir como una secuencia consecutiva de movimientos que ocurren en un periodo de tiempo variable. Por tanto se puede aplicar DTW sin ningún problema. Además, este algoritmo proporciona rechazo para una secuencia (gesto) que no pertenezca a los gestos para los que el sistema ha sido entrenado (diccionario).

- Un ejemplo de su funcionamiento sería el siguiente: Existen dos secuencias X e Y [Figura 5.9], de una dimensión, cuyos valores son:

X = x1, x2, x3,…xi,….xn.
Y = y1, y2, y3,..., yi,…yn.

Longitud de X → **|x|**
Longitud de Y → **|y|**

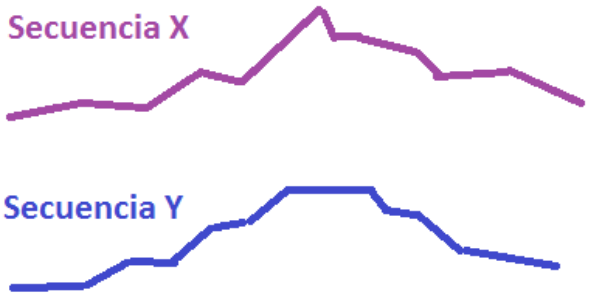

Figura 5.9. Secuencias X e Y

Ambas secuencias se alinean [Figura 5.10] sin tener en cuenta el desfase (de tiempo, velocidad, etc.) entre ellas.

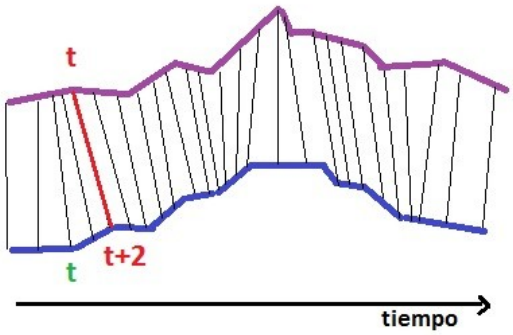

Figura 5.10. Secuencias alineadas

Para encontrar la similitud óptima entre ambas secuencias X e Y, es necesario trazar el camino P (p1, p2,…, ps,…, pn) de distancia, típicamente Euclidea (1), entre ambas secuencias.

$$DIST\left(w_{k_i}, w_{k_j}\right) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad u.d \quad (1)$$

Se usa una *matriz de costo* para facilitar el cálculo. Este camino (2) debe comenzar en p1 = (1,1) y debe acabar en pk = (|x|, |y|) = (xn, yn). Debe de ser continuo y presentar una monotonía, no puede ir hacia atrás.

$$DTW(x,y) = \sum_{k=1}^{|P|} DIST(w_{k_i}, w_{k_j}) \quad u.d \quad (2)$$

Hay multitud de caminos que cumplen estas condiciones, pero solo interesa el que minimiza las distancias entre las secuencias. La distancia entre p1 y pk tiene que ser mínima. Este camino P tiene el nombre de *warping path* (3).

$$P(warping\ path) = \min \sum_{k=1}^{|P|} DIST(w_{k_i}, w_{k_j}) \quad u.d \quad (3)$$

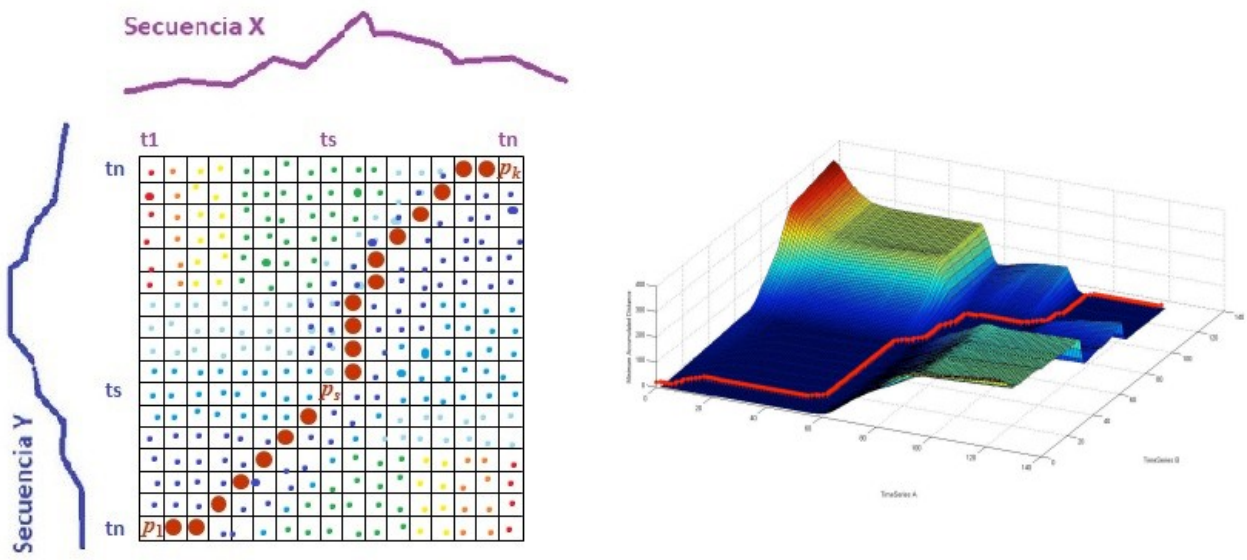

Figura 5.11. Matriz de costo y su "warping path". En 2D (izquierda) y 3D (derecha).

En el sistema que se está desarrollando, existen dos tipos de secuencias:

- La secuencia del gesto que se ha realizado (y que se quiere traducir).
- Las K secuencias de los K gestos que componen el diccionario.

Hay que comparar por tanto la secuencia del gesto realizado (y que se quiere traducir) con cada una de las K secuencias del diccionario, para determinar con cuál de ellas el camino (warping path) es el mínimo. Ya que el que presente un camino mínimo será el gesto traducido.

Además se debe tener en cuenta que las secuencias no son de una sola dimensión, sino que tienen una dimensión N que dependerá de la cantidad de información que sea captada por nuestro dispositivo.

***Por ejemplo**. Si captamos solo la posición de la mano izquierda (x_mi, y_mi, z_mi) y la posición de los dedos pulgar e índice (x_dp, y_dp, z_dm) y (x_di, y_di, z_di), las secuencias tendrán una dimensión de N = 9.*

Se debe usar por tanto **ND-DTW**, que es una implementación de DTW para secuencias de más de una dimensión.

La metodología usada en el reconocimiento de gesto que se usará en la aplicación, será por tanto la siguiente:

Se suman las distancias de error (distancia Euclidea) entre las N dimensiones de las ambas secuencias. Esta distancia total (4) es la que se usa para construir la *matriz de costo.*

$$DIST(i,j) = \sqrt{\sum_{n=1}^{N}(i_n - j_n)^2} \quad u.d \quad (4)$$

Cuando se calculen las *matrices de costo* (con sus respectivos *warping path*) obtenidas de comparar la secuencia del gesto realizado con las secuencias de cada uno de los gestos grabados en el diccionario, se determinará como gesto reconocido (traducido) aquel que proporciona el camino (*warping path*) mínimo.

$$Gesto\ reconocido = \min[ND_{DTW}(gesto\ realizado, diccionario)]$$

Se puede determinar por tanto, de forma gráfica, cómo funcionará el bloque que permite reconocer gestos [Figura 5.12] en este sistema mediante el algoritmo ND-DTW:

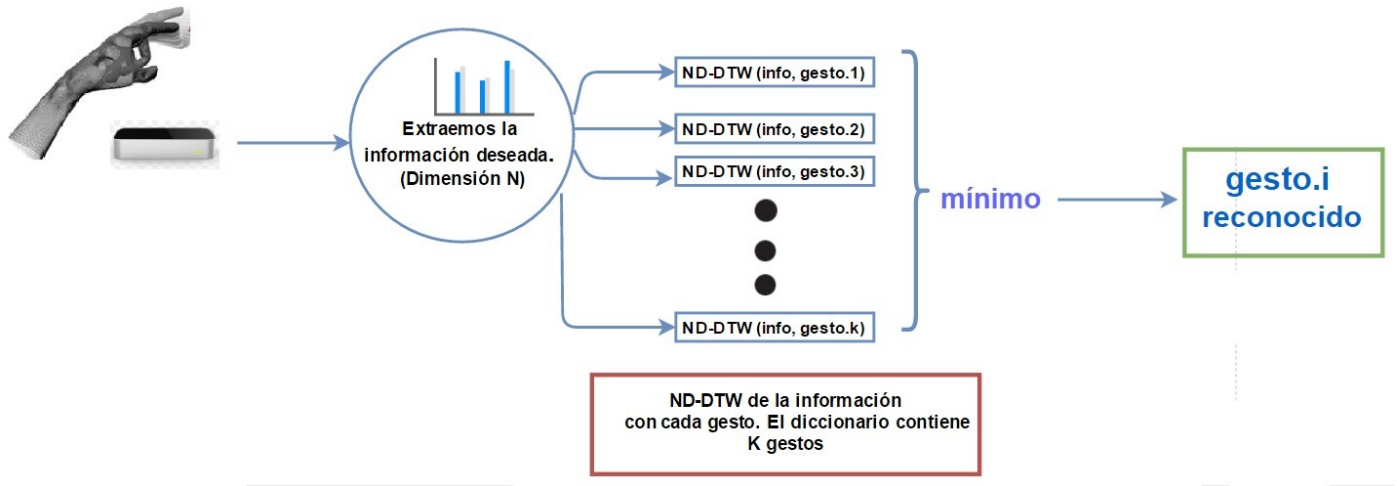

Figura 5.12. Funcionamiento del reconocedor de gestos del sistema.

5.4. Elección de la información capturada por Leap Motion.

En los apartados anteriores se ha concluido cual será la estructura que tendrá el sistema [Figura 5.13] que se está desarrollando, con su correspondiente separación en módulos o bloques:

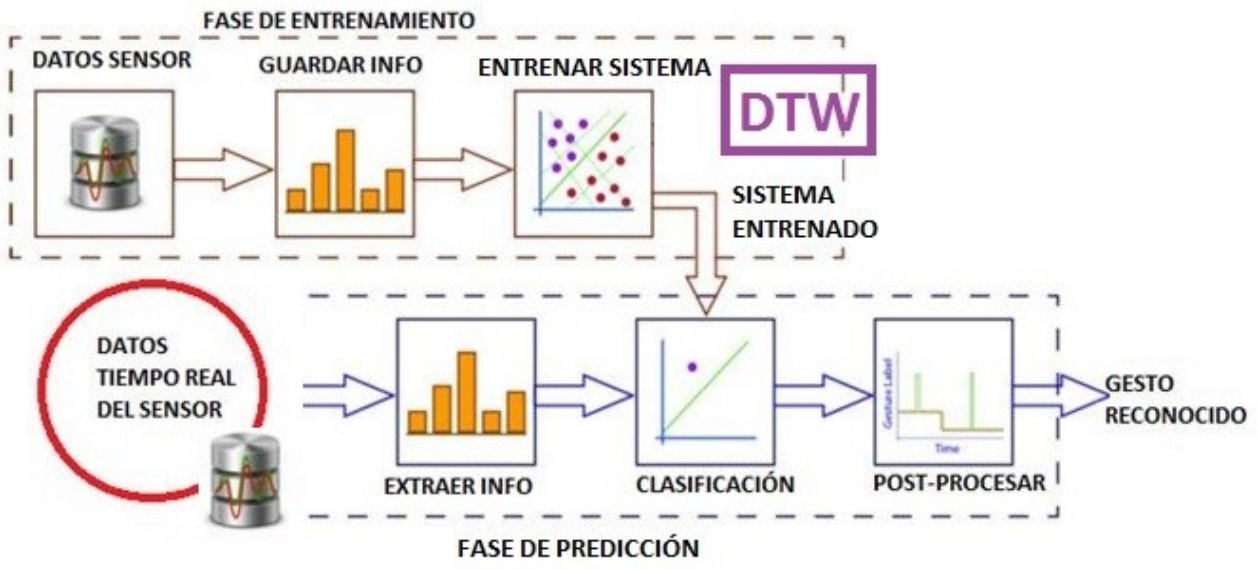

Figura 5.13. Estructura del sistema.

Además, se ha explicado cómo va a funcionar el módulo del reconocedor (clasificación) en el apartado *5.3* y el tipo de post-procesamiento que se llevará a cabo. Se ha explicado también en el apartado *4.3.2* qué información es capaz de capturar el dispositivo Leap Motion. Queda por tanto determinar qué información, de entre toda la que se podría captar, será la más adecuada para este sistema.

No se debe tener en cuenta toda la información que Leap Motion es capaz de capturar porque se aumentaría el uso de CPU, la dimensión de las secuencias y el tiempo de computación sin ser necesario. Y además, cogiendo solo la información precisa y necesaria el sistema funcionará correctamente.

Para decidir qué información es la que se debe tener en cuenta para la aplicación, se analizará un poco más los gestos empleados en la lengua de signos. Existe un conjunto de unidades simbólicas mínimas o propiedades en la lengua de signos que forman y caracterizan la mayoría de los signos. Estas propiedades son:

1) **Configuración**: Forma que adquiere la mano para realizar un signo. La forma de los dedos es la que más peso tiene en este aspecto.
2) **Orientación**: palma hacia arriba, hacia abajo, hacia el hablante…
3) **Zona de articulación**: lugar del cuerpo donde se realiza el signo; boca, frente, hombro…
4) **Movimiento**: tipo de movimiento que tienen las manos al realizar el signo; recto, giratorio, vaivén…
5) **Contacto**: Parte de la mano dominante (derecha en diestros, izquierda en zurdos) que toca otra parte del cuerpo; yemas de los dedos, palma de la mano…
6) **Plano**: Zona donde se realiza el signo y que dependerá de la distancia entre signo y cuerpo; Plano 1 → en contacto con el cuerpo y Plano 4 → lo más alejado posible, brazos estirados.
7) **Componente no manual**: Información que se trasmite por el resto del cuerpo; expresión facial, componentes orales, movimientos de hombros o tronco…

Será posible cubrir con todas las propiedades que caracterizan un signo excepto con la componente no manual, ya que está fuera de la información que el dispositivo Leap Motion es capaz de capturar. Aún así, si se consigue cubrir las otras seis de forma correcta, se conseguirá cubrir aproximadamente el 85% de las propiedades que caracterizan un signo. Quedan ya únicamente dos preguntas a las que dar respuesta:

¿Qué información se debe capturar entonces del dispositivo Leap Motion para cubrir estas características?

En la siguiente tabla, se exponen las propiedades que se deberían cubrir y qué información ha parecido ser la más adecuada para cada una. Así como la razón de su elección:

Propiedad	Información	Razón
Configuración	• Finger::tipposition(x,y,z) • Finger::direction (x,y,z)	Con la posición del extremo del dedo y su dirección será suficiente para determinar la configuración que adquiere la mano.
Orientación	• Hand::palmposition (x,y,z) • Hand::palmnormal (x,y,z)	Se recuerda que con la posición de la palma y el vector normal a esta, quedaba definida la orientación de la mano.
Zona de articulación	• Hand::palmposition (x,y,z) • Finger::tipposition (x,y,z) • Hand::direction (x,y,z)	Situando la mano y los dedos, podrá determinarse en qué parte del espacio 3D se encuentra la mano.
Movimiento	• Hand::palmposition (x,y,z) • Hand::palmnormal (x,y,z) • Hand::direction (x,y,z)	Conociendo el cambio que tiene la orientación y dirección durante la realización del gesto, se determina el tipo de movimiento.
Contacto	• Hand::palmposition (x,y,z) • Hand::direction (x,y,z) • Finger::tipposition (x,y,z)	Situando la mano y los dedos, se conocerá si hay algún punto en contacto.
Plano	• Hand::palmposition (x,y,z)	Situando la mano puede determinarse si está más o menos cerca del cuerpo.

Tabla 5.14. Información que cubre las propiedades de los signos.

Se observa por tanto que para poder distinguir con propiedad entre los diferentes signos, es necesario capturar la siguiente información procedente del sensor Leap Motion:

- ✓ Hand::palmposition
- ✓ Hand::palmnormal
- ✓ Hand::direction
- ✓ Finger::tipposition
- ✓ Finger::direction

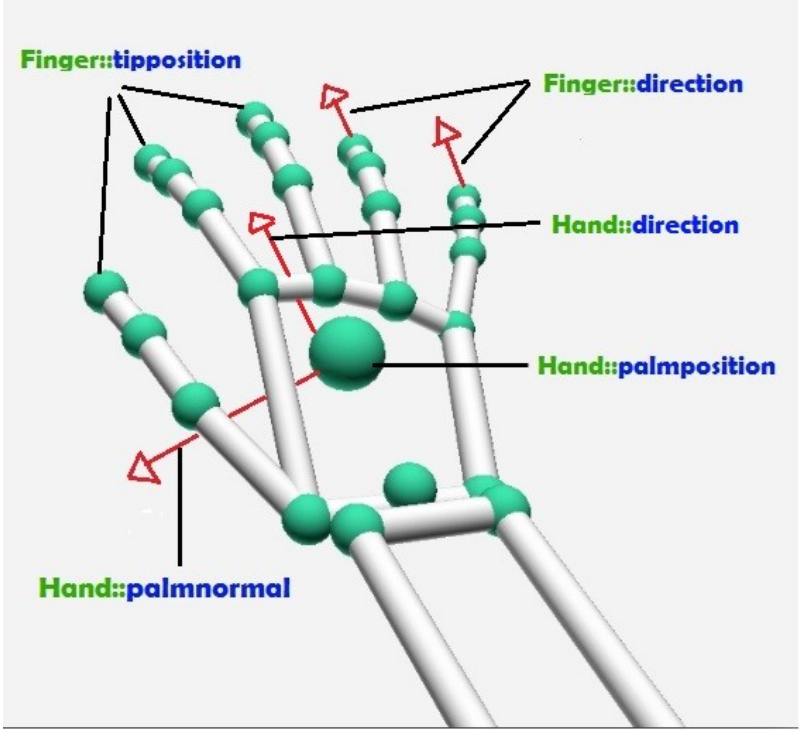

Figura 5.15. Información capturada por el dispositivo.

¿Qué dimensión tendrán entonces las secuencias grabadas en el diccionario y las secuencias que serán clasificadas (traducidas)?

Teniendo en cuenta que cada parámetro devuelve información en forma de coordenadas espaciales (x, y, z), cada uno de estos parámetros tendrá una dimensión de N = 3. Además, dependerá de si se está capturando una o dos manos. Y como cada mano cuenta con 5 dedos, los parámetros *Finger::tipposition* y *Finger::direction* cuentan para cada dedo (aparecen cinco veces para cada mano). Resumiendo, cada secuencia tendrá una dimensión N:

$$N = \begin{cases} Con\ 1\ mano \rightarrow\ 1\ mano\ x\ 9\ coordenadas + 5\ dedos\ x\ 6\ coordeandas \\ Con\ 2\ manos \rightarrow 2\ manos\ x\ 9\ coordeandas + 10\ dedos\ x\ 6\ coordenadas \end{cases} \quad (5)$$

Con lo que se determina que:

$$39\ \leq N \leq 78 \quad (6)$$

5.5. Lógica del sistema.

Ya ha quedado clara cuál va a ser la estructura del sistema que estamos desarrollando. Además, se ha explicado y detallado cada uno de los bloques de dicha estructura. Diseñado ya el sistema y sus diferentes módulos, se puede empezar a programar. Pero antes, es necesario determinar cuál será la lógica que debe seguir para conectar unos módulos con otros y que formen en su conjunto el sistema de reconocimiento que se ha desarrollado. Es esencial una correcta interconexión de las librerías, herramientas y algoritmos recopilados y/o desarrollados para que el sistema no conduzca a bucles infinitos, caminos sin salida, o a módulos inaccesibles. Para esto, la lógica propuesta es la siguiente:

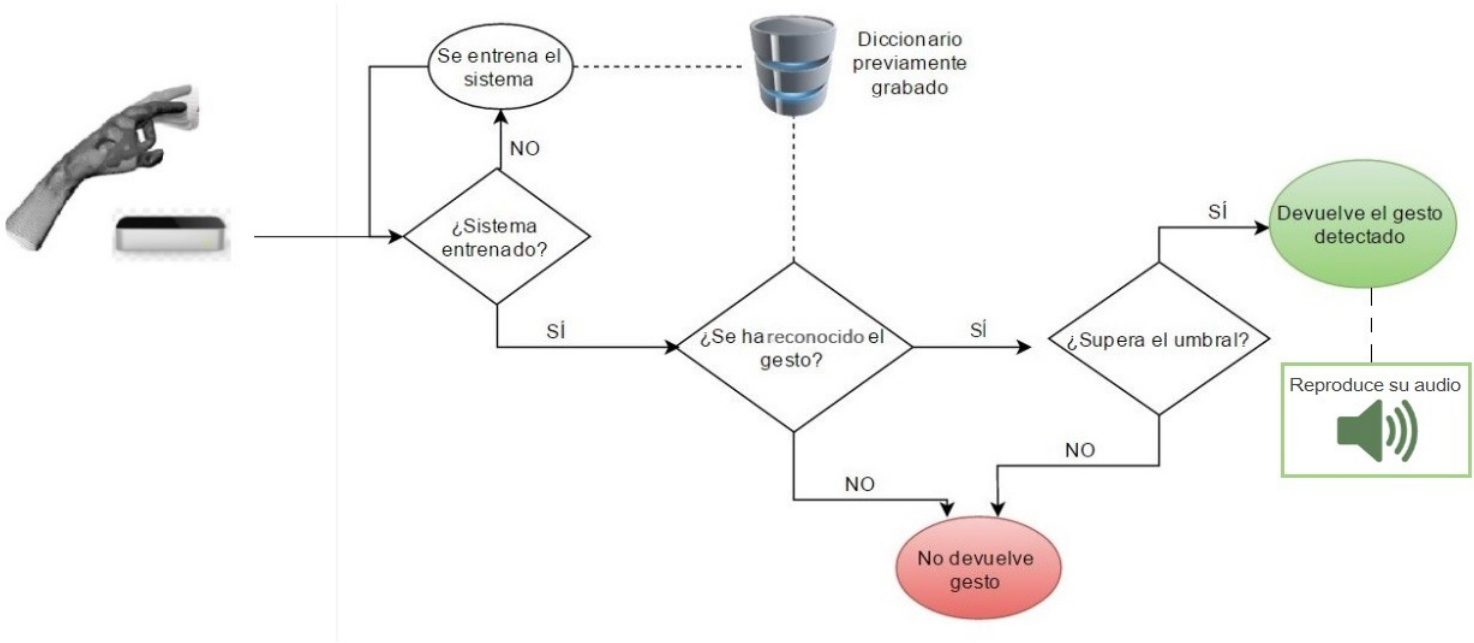

Figura 5.16. Lógica del sistema.

I. El dispositivo Leap Motion capta la información. Si el sistema no está entrenado, habrá que entrenarlo con el diccionario previamente grabado. Si el sistema está entrenado, se llevará la secuencia captada al módulo reconocedor.

II. Aquí se comparará con el diccionario mediante el algoritmo N-DTW y devolverá el gesto que se ha reconocido (traducido), o el valor 0 si no se ha detectado ningún gesto.

III. En el caso de reconocer gesto se pasará al módulo de post-procesamiento, donde se comprobará si la similitud devuelta supera el umbral que hayamos determinado. En

caso de superarlo, se devuelve el gesto traducido y se reproduce su audio correspondiente. En caso contrario, no devuelve gesto.

Con esto, queda determinada la lógica que tendrá el sistema y la interconexión de unos módulos con otros. Se puede por tanto dar paso a la programación.

5.6. Programación del sistema.

Se ha diseñado el sistema completo, explicando y detallando cada módulo que tendrá. Comienza ahora su programación. Se debe programar una aplicación con la que poder interactuar y que cuente con todos los requisitos planteados. En el siguiente apartado se explicarán los elementos que conforman el proyecto distinguiendo cuáles han sido recopilados y cuáles desarrollados para este trabajo. Se explicará las distintas funciones que han sido programadas y su modo de funcionamiento. Además se describirá el menú que presentará la aplicación final ofrecida al usuario

5.6.1. IDE empleado.

Para programar la aplicación, se va a utilizar el lenguaje C++. Además de ser el lenguaje empleado en la toolkit que se implementará (GRT), es el lenguaje nativo del dispositivo Leap Motion.

El IDE (Integrated Development Environment) que será empleado es Visual Studio [Figura 5.17]. Concretamente Visual Studio Community 2015. Este IDE es extensible y permite crear aplicaciones modernas para Windows, Android e iOS, además de aplicaciones web y servicios en la nube.

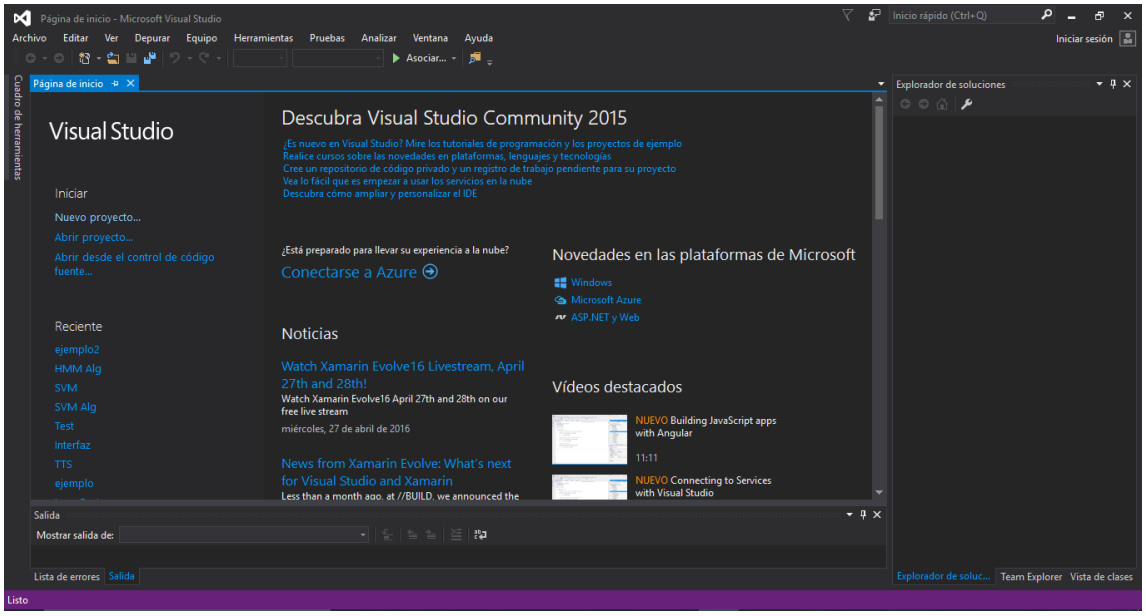

Figura 5.17. Pantalla principal de Visual Studio

5.6.2. Proyecto/solución

Visual Studio trabaja con "soluciones". Una solución es un entorno de trabajo en el que es posible tener varios proyectos. El proyecto creado para el desarrollo y programación de la aplicación se puede dividir en los siguientes componentes:

- **Archivos externos**: Se corresponden con las librerías y toolkit que vamos a usar (que se han recopilado). Como bien se ha expuesto con anterioridad, tendremos los siguientes:
    - → **Leap Motion SDK versión 2.3.1**: Contiene los archivos de cabecera y las librerías de código para crear aplicaciones nativas compatibles con Leap Motion. Para añadir esta SDK al proyecto, hay que seguir los pasos proporcionados en la documentación disponible en la web oficial, concretamente en el apartado de C++ llamado *"Setting Up a Project"*. Esta SDK será la que permita programar funciones que accedan y capturen la información deseada del dispositivo.
    - → **Gesture Recognition Toolkit**: Librería diseñada para reconocimiento de gestos. Cuenta con numerosos algoritmos de reconocimiento, además de otras utilidades para estos sistemas. Se encuentra bajo licencia MIT, lo que permite usarla sin problema.

- **Archivos propios**: Son los archivos de encabezado y de código fuente que han sido programados para desarrollar el sistema. Se encargan del interconexionado y combinación de los archivos externos y de dar funcionalidad al sistema. Estos archivos son los siguientes:
    - → **Funciones.h**: En este archivo de encabezado se ubica **el motor del sistema** y es sin ninguna duda el más importante de todos. En él se combinan los archivos externos tanto para extraer la información del Leap Motion como para crear el sistema de reconocimiento basado en la toolkit GRT. Además, contiene todas las funciones que se han programado para intentar cubrir los módulos del sistema. Más adelante se explicará el contenido de este archivo de una forma más extendida.
    - → **Main.cpp**: Archivo de código fuente que contiene el menú que permite llamar a las distintas funciones del archivo "Funciones.h" y que se

mostrará al usuario para utilizar el software diseñado. Este menú se explicará de una forma más extendida más adelante.

→ **Archivos de audio**: Conjunto de archivos de audio en formato WAV. Estos archivos son una reproducción del nombre del signo y serán reproducidos una vez que el signo sea reconocido. Se incluyen en un principio los 10 audios correspondientes a los 10 signos grabados para la comprobación del correcto funcionamiento de esta aplicación. Al añadir más gestos o grabar un nuevo diccionario, basta con generar sus correspondientes audios y añadirlos a la carpeta del proyecto.

- **Archivos generados**: Son los archivos que se generan con algunas de las funciones programadas y que serán utilizados en otras funciones. Se guardan en la carpeta principal del programa. Estos archivos son:
  - → **Diccionario.txt**: Contiene la información relativa a los gestos grabados y que serán usados para entrenar el sistema de reconocimiento. Esta información se guarda en forma de matrices de N columnas y M filas, siendo N la dimensión de la información capturada por el dispositivo (declarada en el apartado 5.4) y M el número de frames capturadas en los X segundos transcurridos durante la realización del gesto.
  - → **Nombres.txt**: En este archivo se escriben y se leen los nombres que han sido asignados a los gestos grabados y se relaciona con la información de "Diccionario.txt" para nombrarlos.
  - → **DTWModel.txt**: Aquí se guarda el sistema ya entrenado (con la información de "Diccionario.txt" y "Nombres.txt") y listo para la traducción. Cargando este archivo, no hay que volver a entrenar el sistema, con lo que se puede llegar a ahorrar bastante tiempo sobre todo si el diccionario posee muchos gestos y para cada uno de ellos han sido grabadas numerosas repeticiones.

En el siguiente diagrama se puede ver la composición del proyecto:

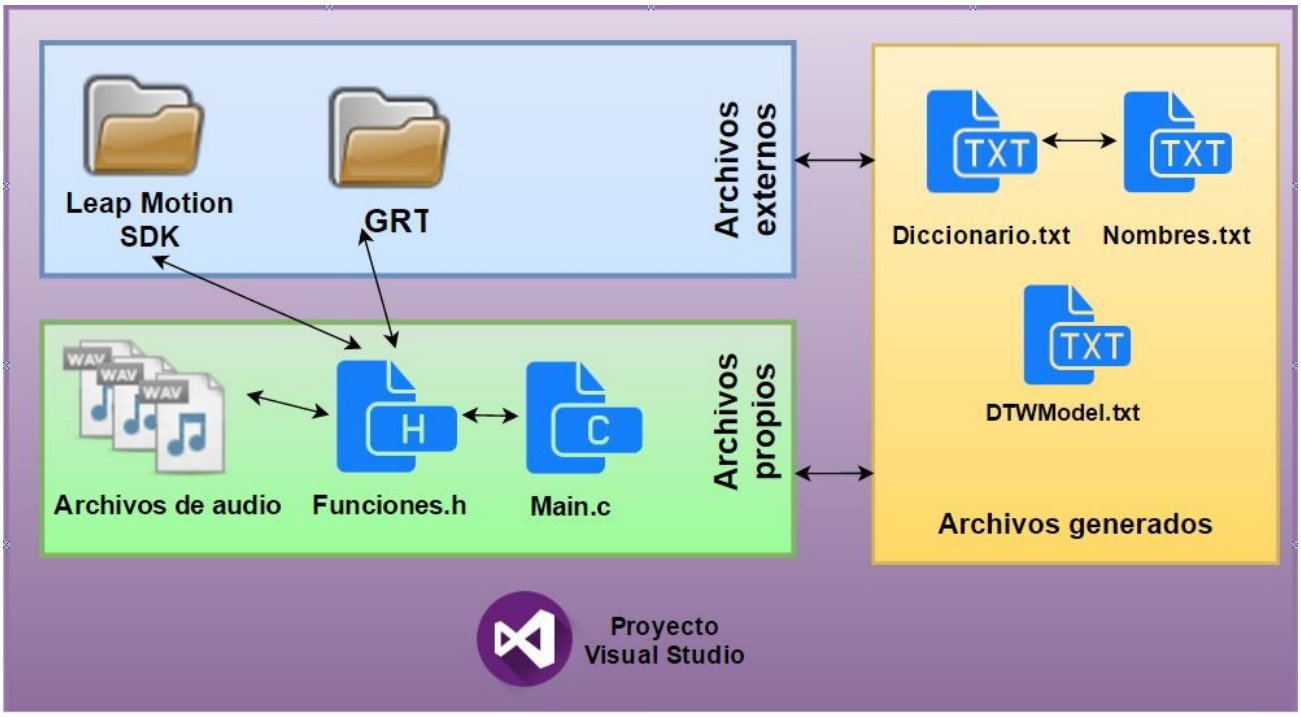

Figura 5.18. Elementos que componen el proyecto.

5.6.3. "Funciones.h", el motor del sistema.

Como bien se ha explicado anteriormente este archivo contiene el motor del sistema, por lo que se ha considerado importante profundizar en su explicación. En él se combinan los archivos externos tanto para extraer la información del Leap Motion como para crear el sistema de reconocimiento basado en la toolkit GRT. Además, contiene todas las funciones que se han programado para intentar cubrir los módulos del sistema. Se va a explicar en detalle qué hace cada función programada en este archivo, y cómo ha sido programada para dicho fin:

→ **Función para conectar con Leap Motion (SampleListener):** Esta función se llama de forma automática cada vez que haya un *Frame* disponible en el dispositivo Leap Motion siempre y cuando se haya añadido un "listener" al controlador. Se ha programado principalmente para almacenar la información que se captura (explicado en el apartado 5.4), para ello se usa principalmente la API del dispositivo. Esta información se guarda temporalmente en un vector de dimensión N llamado "sample". En caso de estar grabando un signo, ese vector copia dicha información a la matriz "timeseries" que conformará el diccionario. Por el contrario

en caso de estar en la fase de predicción, el vector copia dicha información a una matriz auxiliar "timeseries2" que será enviada al módulo reconocedor para su traducción.

→ **Setup( ):** Esta función se encarga de establecer la configuración del sistema. Se llama automáticamente al iniciar el programa. En ella se inicializan parámetros cómo por ejemplo la dimensión N de las matrices "timeseries" y "timeseries2". Además configura la toolkit GRT para usar el algoritmo DTW, y le pasa algunos parámetros como por ejemplo el coeficiente de rechazo.

→ **Grabar( ):** Esta función ha sido programada para grabar los gestos que se realizan y añadirlos al diccionario que se está grabando. Para ello, añade un "listener" y empieza a almacenar la información en la matriz "timeseries". Una vez realizado el gesto, esta matriz será añadida al diccionario y el "listener" será eliminado. Además, informa sobre el número de repetición que acaba de ser grabada.

→ **Gestomas( ):** Función sencilla que añade un gesto al diccionario para posteriormente grabar sus repeticiones. Tiene también la tarea de asignarle el nombre que se ha introducido por teclado a ese gesto en concreto, para ello se almacena en el vector "nombres" que posteriormente será escrito en el archivo "Nombres.txt".

→ **Gestomenos( ):** Función sencilla cuya única finalidad es la de eliminar el último gesto añadido al diccionario, para así poder rectificar en caso de error.

→ **Entrenar( ):** Esta función ha sido programada para entrenar el sistema de reconocimiento con el algoritmo DTW. Para ello invoca a una función de la toolkit GRT que nos permite realizar dicho entrenamiento. También se encarga de guardar el sistema entrenado en el archivo "DTWModel.txt" de forma automática.

→ **Guardar( ):** Esta función se encarga de guardar el diccionario en el archivo "Dicconario.txt" para poder darle un posterior uso. Además, es la encargada también de generar el archivo "Nombres.txt" donde se escribirá la información del vector "nombres".

→ **Cargar( ):** Esta función utiliza los archivos generados por la función guardar(). En ella se carga la información disponible en el archivo "Diccionario.txt" al diccionario del sistema, y se carga la información del archivo "Nombres.txt" al vector "nombres". Tendrá también la tarea de mostrar una lista en la que se enumeran los signos que han sido cargados.

→ **Predecir( ):** Se ha programado para ser la función encargada en la traducción y en la reproducción de los audios. Para ello, añade un "listener" y empieza a almacenar la información en la matriz auxiliar "timeseries2". Una vez realizado el gesto que se quiere traducir, esta matriz será enviada al módulo reconocedor y el "listener" será eliminado. Tras ser reconocido, nos devolverá el signo traducido y una lista que contiene la similitud con cada uno de los gestos del diccionario. Además, en caso de una correcta traducción, se reproducirá el audio correspondiente.

→ **Borrar( ):** Función sencilla cuya única finalidad es la de borrar el diccionario que ha sido grabado o cargado.

→ **Cargarpipeline( ):** Función sencilla cuya única finalidad es cargar el sistema ya entrenado que se encuentra en el archivo "DTWModel.txt".

El archivo "Funciones.h" será por tanto, el principal motor de la aplicación ya que se encarga de la interconexión de los distintos módulos.

5.6.4. Menú de usuario.

En este apartado se explicará el menú [Figura 5.19] que se ha diseñado para dicha aplicación y qué proporciona cada opción. Este será el que se pondrá a disposición del usuario final para utilizar el software, y se trata de una aplicación en consola.

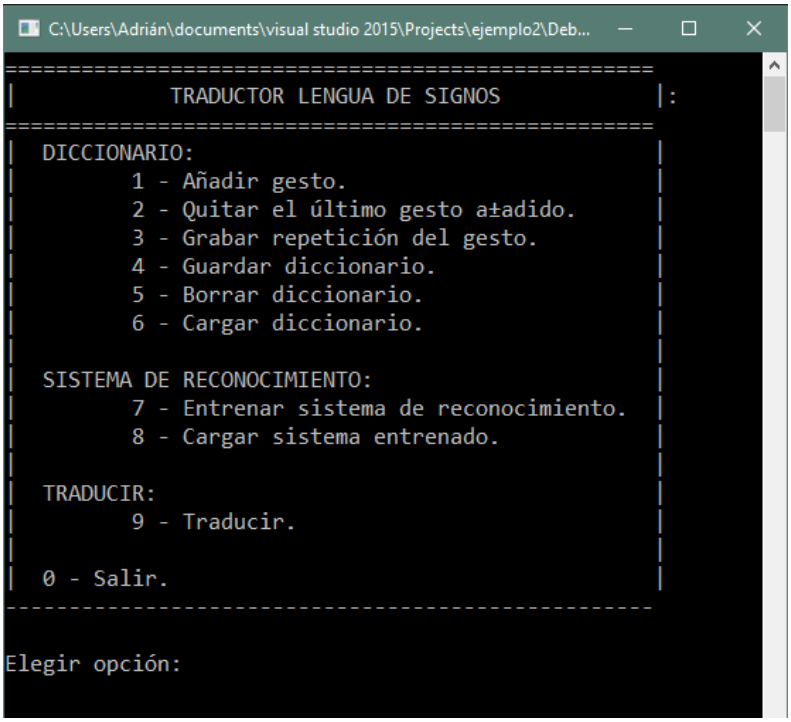

Figura 5.19 Menú de la aplicación.

Como se puede observar en la Figura 5.19, el menú se ha dividido en tres grandes secciones (DICCIONARIO, SISTEMA DE RECONOCIMINETO Y TRADUCIR) para que sea más fácil e intuitivo diferenciar la funcionalidad de las opciones.

Las opciones que permiten trabajar con el **DICCIONARIO** son:

1) Añadir gesto: permite añadir un signo al diccionario que se está grabando. Al seleccionar esta opción se pedirá escribir el nombre del signo (se guardará en el archivo "Nombres.txt"). Tras escribirlo, el signo será añadido y ya solo habrá que grabar repeticiones de este.

2) Quitar el último gesto añadido: permite quitar el último gesto que se ha añadido durante la grabación del diccionario. Es útil si hay algún tipo de equivocación al añadir un gesto, ya que podemos rectificarlo.

3) Grabar repeticiones del gesto: una vez añadido el gesto hay que grabar repeticiones. Para ello, una vez seleccionada esta opción, se realiza el gesto con naturalidad delante del dispositivo Leap Motion y se pulsa la tecla "1" y "Enter" una vez finalizado. Habrá que repetir este proceso para cada gesto el número de veces que sea aconsejable y necesario. Es importante procurar que todos los gestos tengan el mismo número de repeticiones para dar equidad al sistema.

4) Guardar diccionario: una vez completada la grabación del diccionario mediante las opciones 1), 2) y 3), esta opción permite guardarlo en el archivo "Diccionario.txt" para poder usarlo en cualquier momento.

5) Borrar diccionario: permite borrar el diccionario que acaba de ser grabado ó cargado para poder grabar uno nuevo.

6) Cargar diccionario: con esta opción es posible cargar al sistema el diccionario guardado bajo el nombre de "Diccionario.txt".

Las que nos permiten trabajar con el **SISTEMA DE RECONOCIMIENTO**:

7) Entrenar sistema de reconocimiento: esta opción se encarga de entrenar al sistema con la información de "Dicconario.txt" para su uso posterior en la traducción mediante el algoritmo DTW. El sistema ya entrenado se guarda en el archivo "DTWModel.txt".

8) Cargar sistema entrenado: permite cargar el sistema guardado en "DTWModel.txt". Esta opción ahorra tener que volver a entrenar el sistema, ya que si el diccionario contiene mucha información puede llevar cierto tiempo entrenarlo.

Y por último, dentro de la sección **TRADUCIR** tenemos**:**

9) Traducir: Al seleccionar esta opción, el Leap Motion empezará a capturar información. Realizamos el signo y pulsamos la tecla "1" y "Enter" una vez realizado. Automáticamente se devolverá el signo que se ha reconocido con su correspondiente audio, y la similitud con el resto del diccionario.

0) Salir: Con esta opción se sale del programa.

6. ENSAYOS Y RESULTADOS

Con la aplicación ya diseñada y desarrollada, es necesario probar su correcto funcionamiento y su exactitud para la finalidad de este TFG.

Para probar dicho sistema, se va a tomar como objetivo el correcto reconocimiento de un diccionario de 10 signos. Se tendrán que llevar a cabo por tanto los ensayos y mejoras que sean necesarias hasta que se alcance este objetivo, y se pueda determinar que la aplicación es exacta y funcional para este TFG. La metodología general que será usada en la realización de dichos ensayos será la siguiente:

1) Grabar diccionario de 10 signos ó usar el grabado en un ensayo previo.
2) Entrenar el sistema de reconocimiento.
3) Construir la matriz de correlación que determinará el nivel de acierto del sistema.
4) Comentarios y observaciones de los resultados y el proceso.
5) Decidir las mejoras que se harán en el siguiente ensayo en caso de que haya que mejorar, o si la aplicación ya es funcional.

Figura 6.1. Metodología general en la realización de ensayos.

Se van a detallar alguno de estos procesos en los siguientes apartados.

6.1. Diccionario.

Un aspecto importante en la comprobación del correcto funcionamiento de la aplicación, es la elección de un diccionario adecuado y que cuente con las características más destacables de la lengua de signos.

Para intentar cubrir con los aspectos y propiedades más comunes en los signos de dicha lengua, e intentar de algún modo simplificar sólo a diez signos todo un lenguaje para comprobar dicho funcionamiento, se han grabado principalmente tres variedades de signos:

- Signos basados en el movimiento de la mano.
- Signos basados en la posición de los dedos.
- Signos basados en el movimiento de la mano y posición de los dedos.

Si se consigue reconocer correctamente los diez signos pertenecientes a estas tres principales variedades, se dará por supuesto que el sistema será capaz de reconocer en mayor o menor medida cualquier otro signo grabado por el usuario, debido a que estos cuentan con distinción en la posición de los dedos, movimiento de la mano y ambos aspectos en conjunto.

Por tanto, el diccionario empleado en las pruebas estará formado por los siguientes diez signos:

***NOTA**: *Las fotos de los signos están tomadas desde una vista "alzado" (situando al observador en el monitor del PC) y desde una vista "planta" (siendo el observador nosotros mismos). Además, los signos han sido grabados utilizando la mano izquierda para su realización.*

Signo 1: COMPRENDER

Se corresponde con el signo "comprender" empleado en la Lengua de Signos Española (LSE). Al inicio, la mano está colocada verticalmente con la palma apuntando al oyente (en este caso el monitor) y se irán uniendo las yemas de los dedos unas con otras mientras la mano se mueve hacia la persona que realiza el signo. Este signo está basado tanto en el movimiento de la mano como en la posición de los dedos.

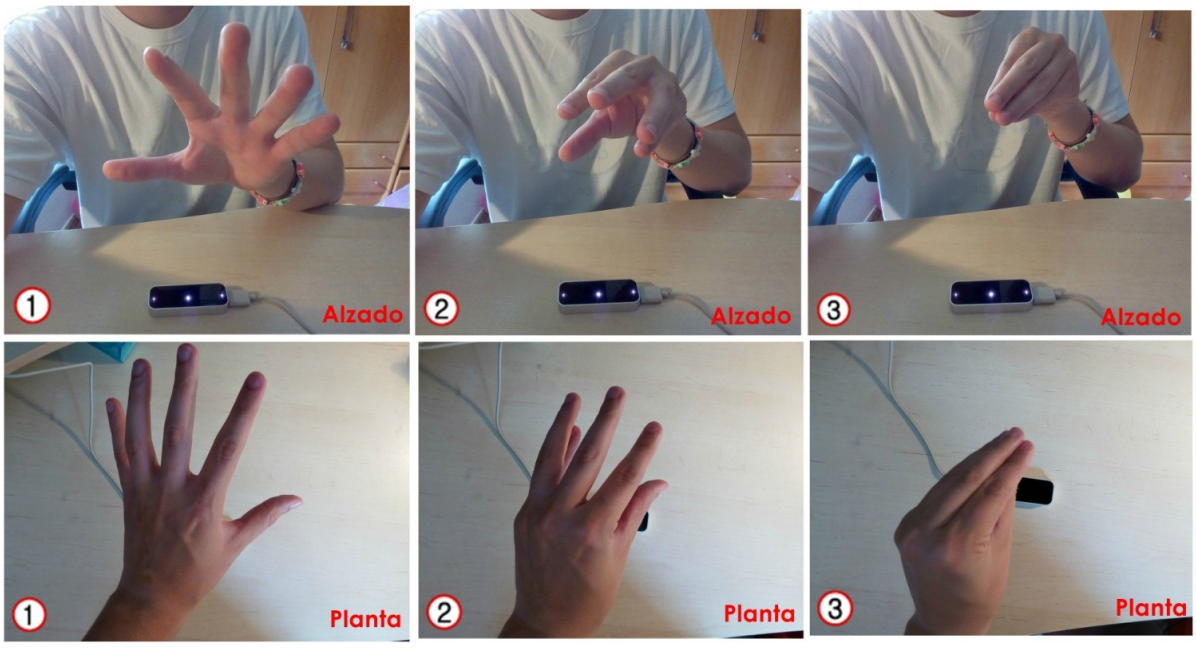

Figura 6.2. Signo "comprender" visto en alzado y planta

Signo 2: ROTAR

Consiste en un movimiento rotatorio de la mano. Al inicio, la mano está colocada horizontalmente con la palma hacia abajo. Tras un movimiento rotatorio de 180º la mano tomará una posición horizontal con la palma hacia arriba. Este signo está basado en el movimiento de la mano.

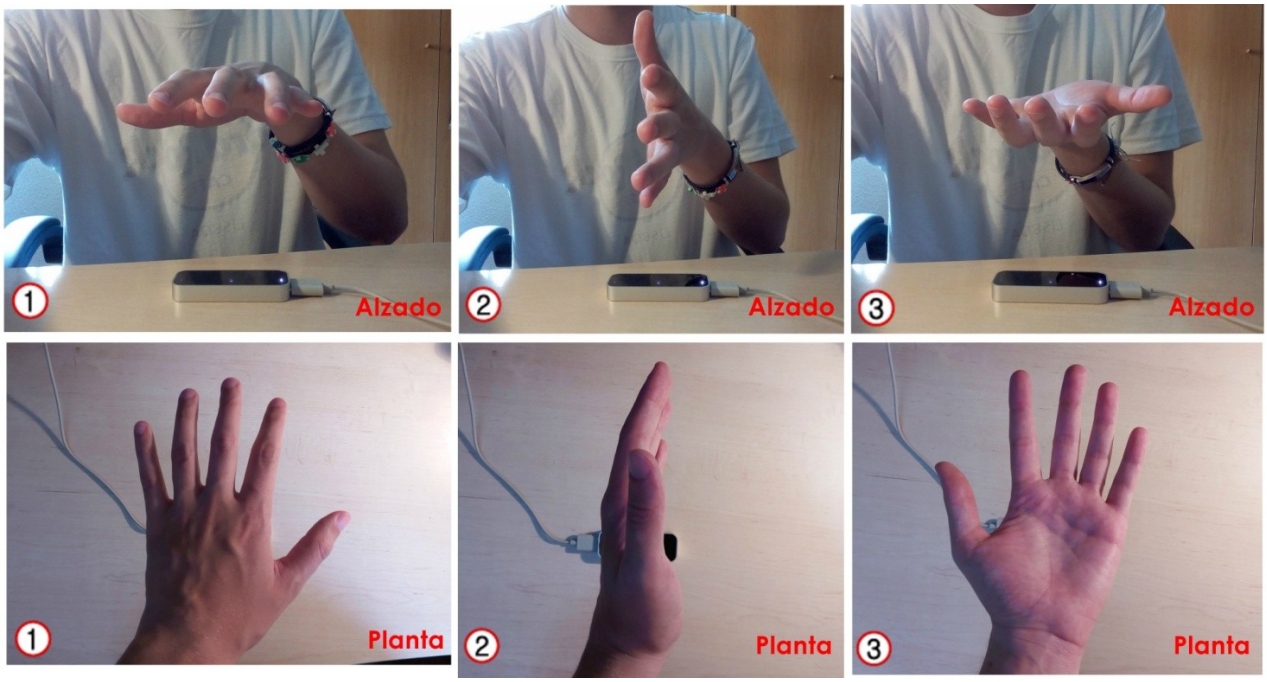

Figura 6.3. Signo "rotar" visto en alzado y planta.

Signo 3: TIJERAS

Corresponde con el signo "tijeras" utilizado en la Lengua de Signos Española (LSE). Consiste en la colocación de la mano en forma de tijeras y el movimiento de los dedos que simulan las hojas de la tijera. Este signo está basado en la posición de los dedos.

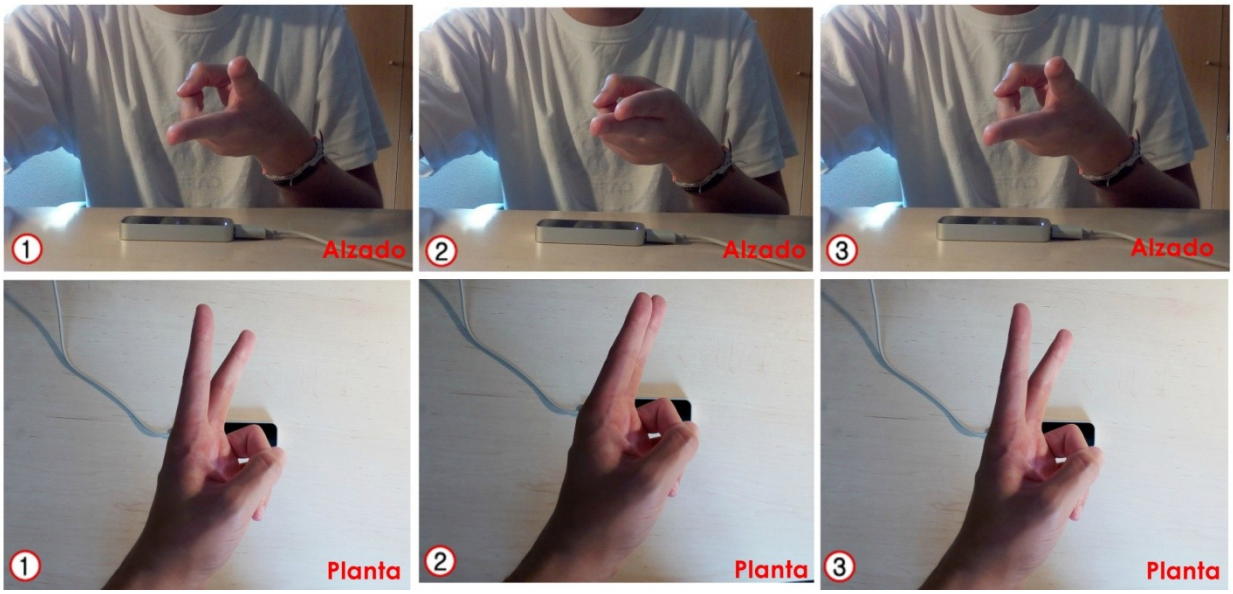

Figura 6.4. Signo "tijeras" visto en alzado y planta.

Signo 4: DERECHA

Consiste en el movimiento rectilíneo de la mano hacia la derecha. La mano está colocada horizontalmente con la palma hacia abajo. Este signo está basado en el movimiento de la mano.

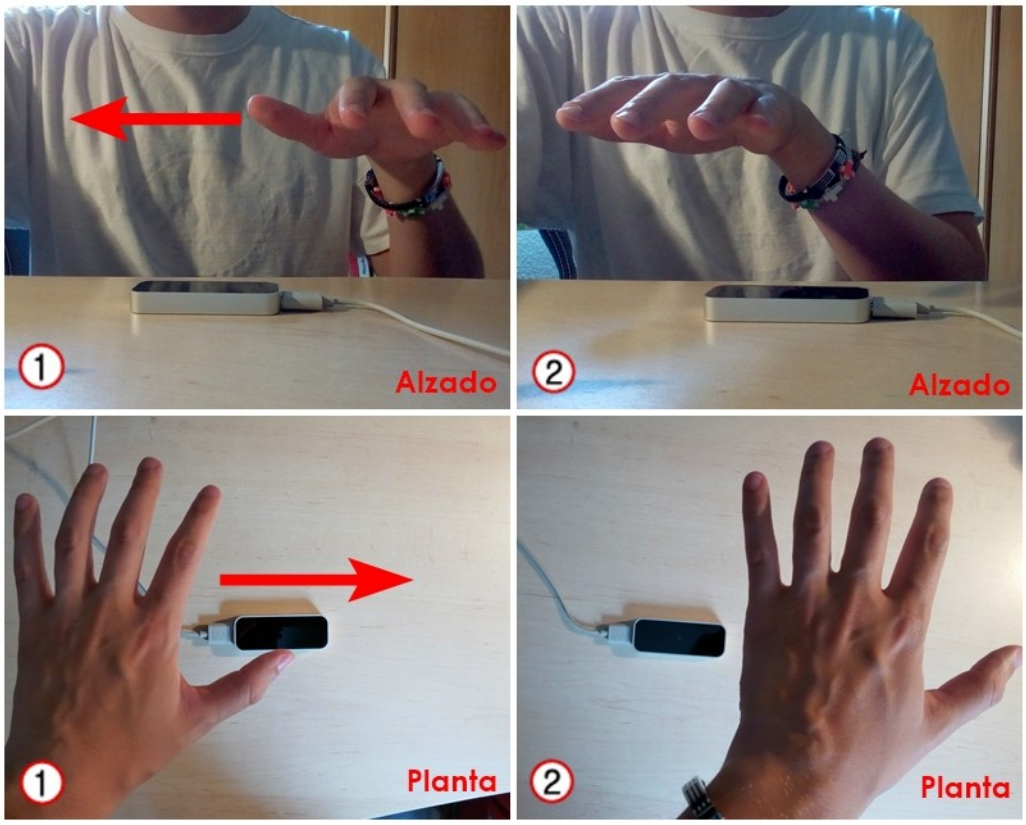

Figura 6.5. Signo "derecha" visto en alzado y planta.

Signo 5: IZQUIERDA

Consiste en el movimiento rectilíneo de la mano hacia la izquierda. La mano está colocada horizontalmente con la palma hacia abajo. Este signo está basado en el movimiento de la mano.

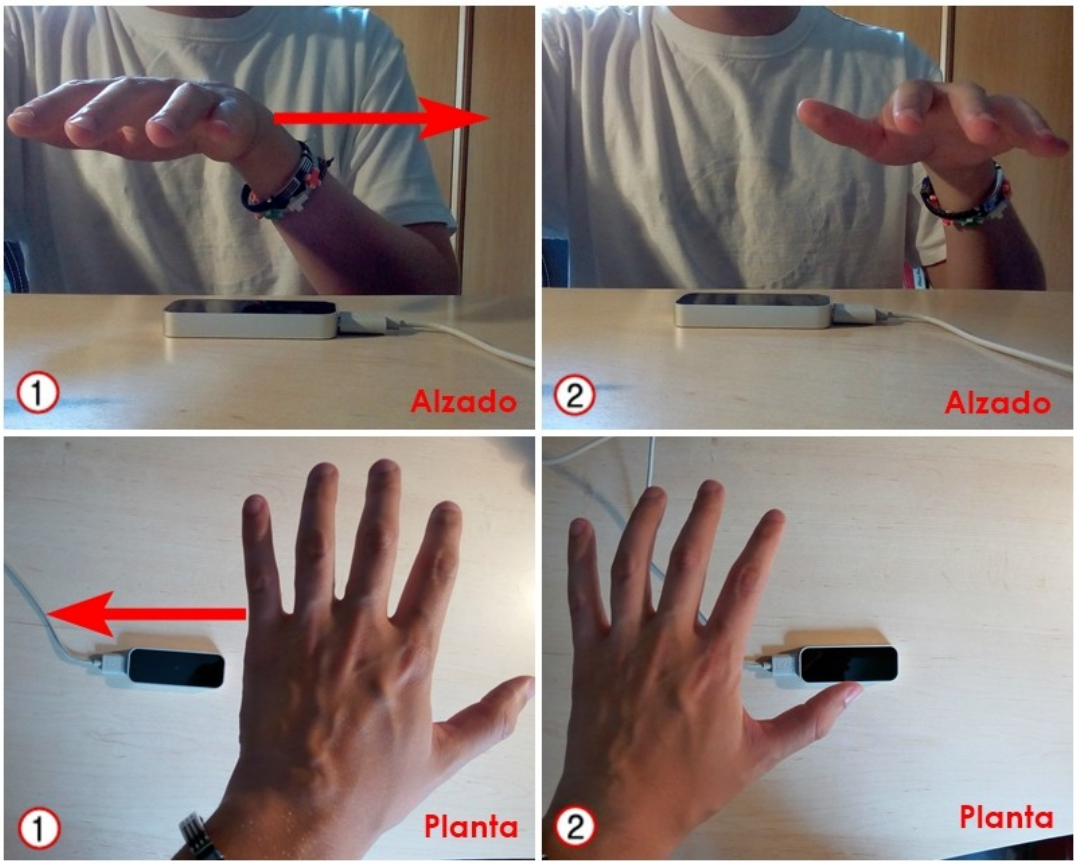

Figura 6.6. Signo "izquierda" visto en alzado y planta

Signo 6: TE QUIERO

Corresponde con la forma corta de decir "te quiero" en la Lengua de Signos Española (LSE). Consiste en la mano cornuta pero con el pulgar extendido. Y es la suma de las letras "i", "l" e "y" por separado (I love you). Este signo está basado en la posición de los dedos.

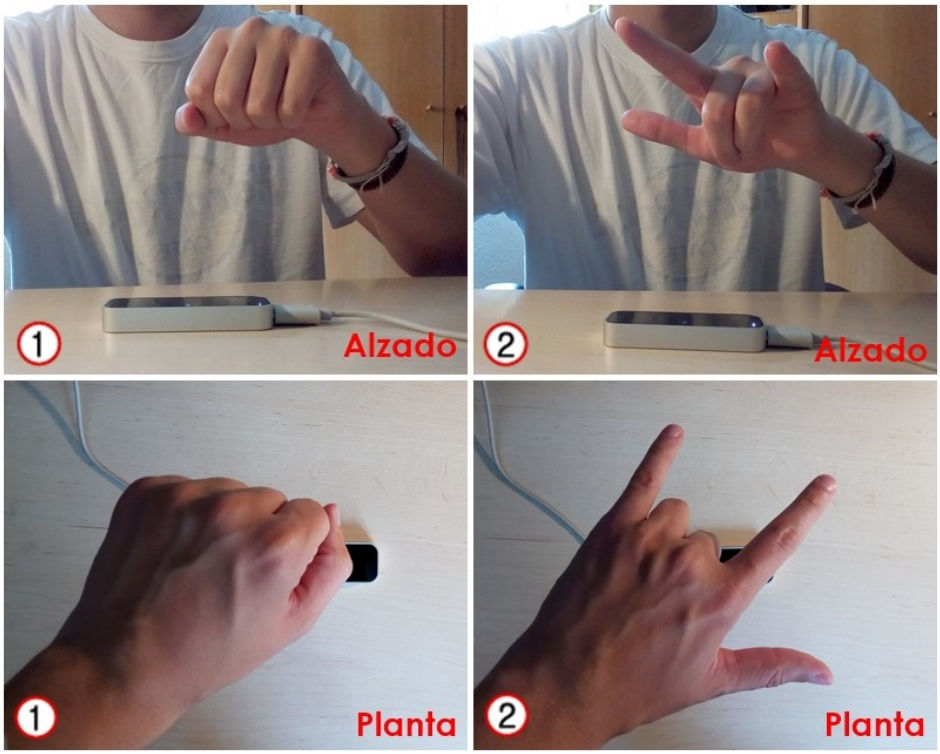

Figura 6.7. Signo "te quiero" visto en alzado y planta.

Signo 7: LETRA "L"

Corresponde con la letra "L" del alfabeto utilizado en la Lengua de Signos Española (LSE). Consiste en la colocación de la mano en forma de "L". Este signo está basado en la posición de los dedos.

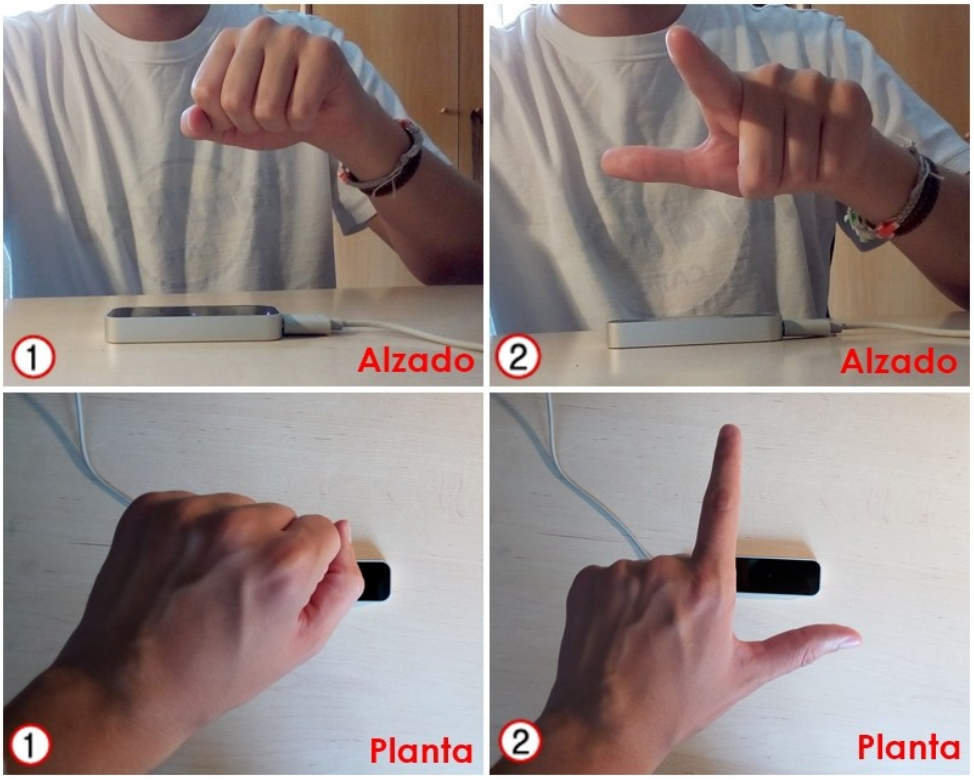

Figura 6.8. Signo "Letra L" visto en alzado y planta.

Signo 8: BUENAS

Corresponde con el signo “buenas” empleado en la Lengua de Signos Española (LSE). Al inicio, las yemas de los dedos están unidas y situadas cerca de la boca. Mientras la mano se va moviendo hacia abajo, se van extendiendo los dedos hasta quedar la mano horizontal con la palma hacia arriba. Este signo está basado tanto en el movimiento de la mano como en la posición de los dedos.

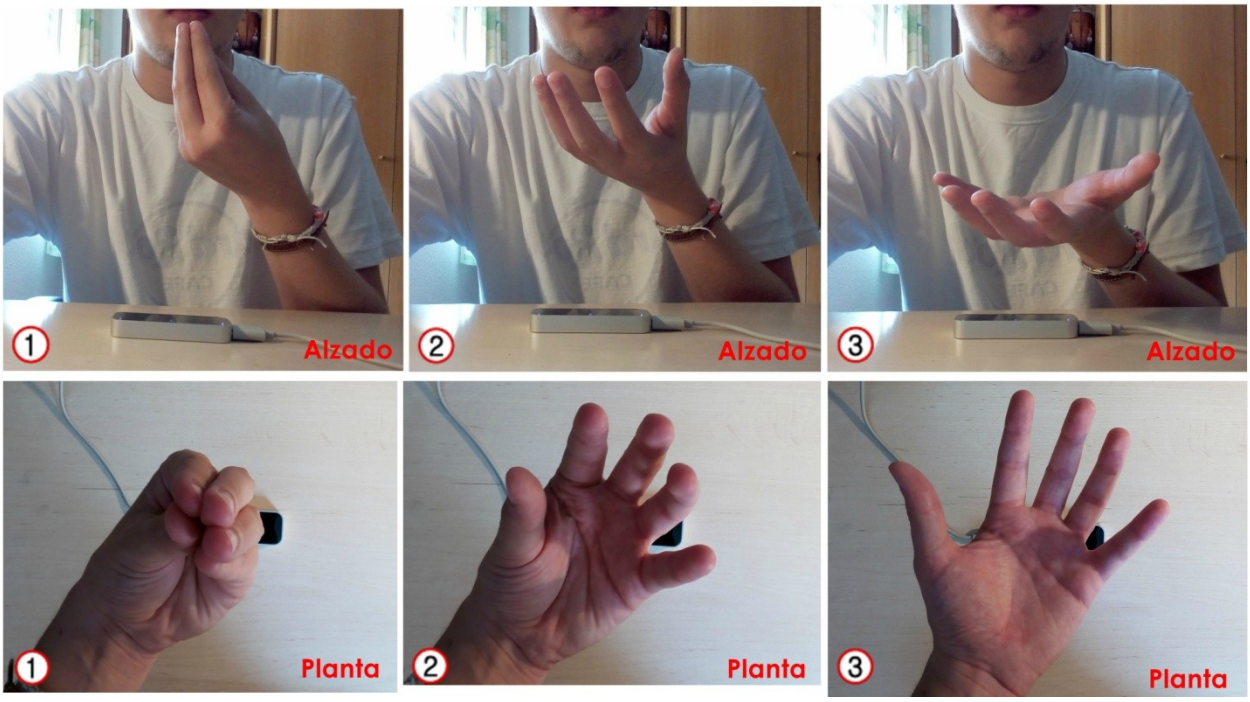

Figura 6.9. Signo “buenas” visto en alzado y planta.

Signo 9: HOLA

Corresponde con el signo “hola” empleado en la Lengua de Signos Española (LSE). Consiste en un movimiento de la mano desde nuestra cara (frente) hacia fuera con los cinco dedos extendidos. Este signo está basado tanto en el movimiento de la mano como en la posición de los dedos.

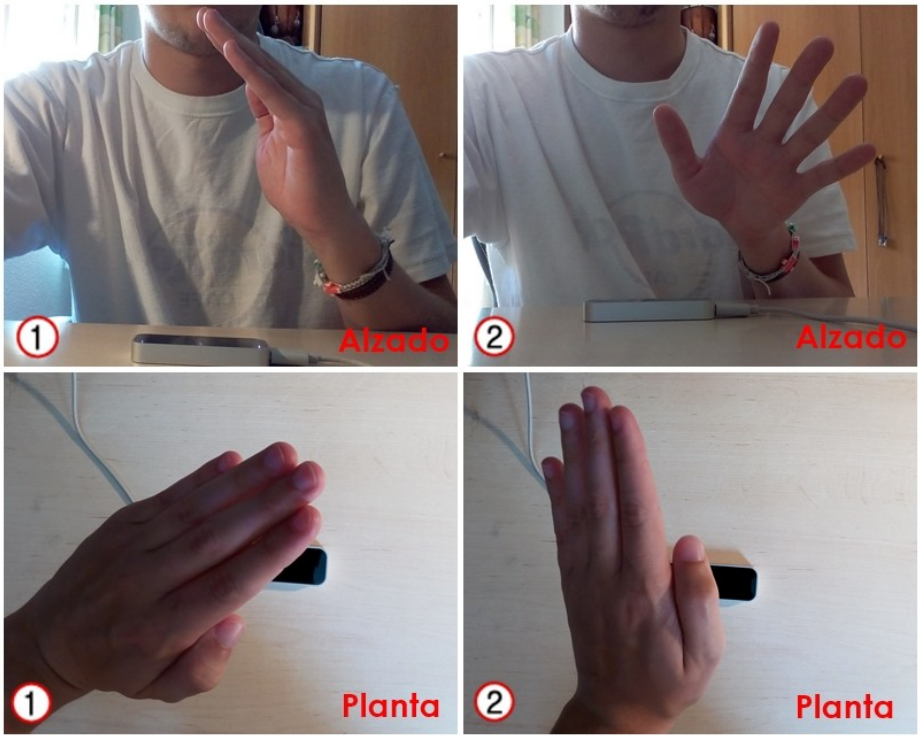

Figura 6.10. Signo “hola” visto en alzado y planta.

Signo 10: TARDES

Corresponde con el signo “tardes” empleado en el saludo “buenas tardes” en la Lengua de Signos Española (LSE). Los dedos índice y pulgar hacen un círculo mientras que los demás se colocan semi-extendidos. Esta posición se acompaña de un pequeño movimiento circular cerca de la boca.

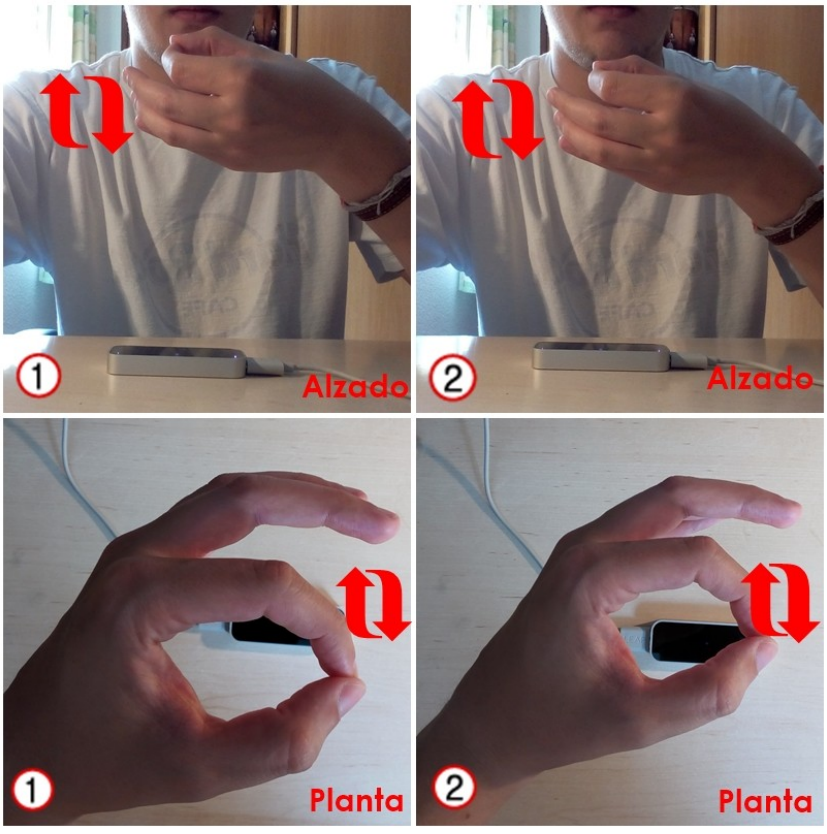

Figura 6.11. Signo “tardes” visto en alzado y planta.

Para las siguientes pruebas, se usarán estas abreviaciones del diccionario:

S1	S2	S3	S4	S5
COMPRENDER	ROTAR	TIJERAS	DERECHA	IZQUIERDA
S6	**S7**	**S8**	**S9**	**S10**
TE QUIERO	LETRA “L”	BUENAS	HOLA	TARDES

Tabla 6.12. Abreviación del diccionario

6.2. Matriz de correlación

Una vez grabado el diccionario y entrenado el sistema, habrá que comprobar cuál es el nivel de exactitud y acierto del sistema. Para ello, se va a usar las matrices de correlación.

Una matriz de correlación es una tabla de doble entrada para las variables, que muestra el correspondiente coeficiente de correlación entre cada pareja de cada celda. El modelo mide y muestra la relación entre cada pareja de variables y todas al mismo tiempo.

Para la aplicación desarrollada, el coeficiente de correlación que se usará es la **similitud** entre el signo realizado y los signos del diccionario expresado **en tanto por ciento**. Para obtener este coeficiente de correlación, se ha realizado (traducido) cada signo un total de 5 veces, y se ha hecho la media aritmética de la similitud del signo realizado con cada uno de los signos del diccionario. Para esto se ha ido exportando a un archivo de texto el vector de similitud con los gestos del diccionario que proporciona la librería GRT después de traducir el gesto, donde se encuentran dichas similitudes. Posteriormente, se ha pasado la información a una hoja de cálculo de Excel para calcular cada media aritmética de forma más rápida y sencilla.

La matriz de correlación tiene la siguiente forma: en la primera fila, se encuentran los signos realizados, y en la primera columna se encuentran los signos del diccionario (obviamente estos coincidirán). En las celdas restantes estará el coeficiente de correlación (similitud) del gesto realizado con cada gesto del diccionario. De forma, que la matriz de correlación que vamos a usar puede representarse de la siguiente manera [Tabla 6.13].

Siendo **SRi** el signo realizado i, y **Si** el Signo i del diccionario:

		Signos realizados			
		Signo realizado 1	**Signo realizado 2**	**Signo realizado 3**	**Signo realizado i**
Diccionario	**Signo 1**	**Similitud SR1 con S1**	Similitud SR2 con S1	Similitud SR3 con S1	Similitud SRi con S1
	Signo 2	Similitud SR1 con S2	**Similitud SR2 con S2**	Similitud SR3 con S2	Similitud SRi con S2
	Signo 3	Similitud SR1 con S3	Similitud SR2 con S3	**Similitud SR3 con S3**	Similitud SRi con S3
	Signo i	Similitud SR1 con Si	Similitud SR2 con Si	Similitud SR3 con Si	**Similitud SRi con Si**

Tabla 6.13. Modelo de matriz de correlación

Las celdas que son marcadas de color amarillo son las que presentan la mayor similitud de cada columna (cada gesto realizado). Lo ideal es que esta similitud sea mayor siempre en la diagonal de la matriz, ya que esto indicaría que cada signo que se ha realizado ha sido traducido de forma correcta. Por tanto, el objetivo es obtener una matriz que tenga las máximas similitudes en la diagonal.

Decidido el diccionario que se va a utilizar en las pruebas y el modelo de análisis que se va a seguir para evaluar el nivel de acierto, ya se pueden realizar dichos ensayos.

6.3. Ensayos.

En este apartado, se va a explicar cada una de los ensayos fundamentales que han sido necesarios hasta lograr el objetivo de un modelo funcional. Además de las que se van a describir en los siguientes apartados, han sido necesarias (como es obvio) numerosas pruebas hasta conseguir un sistema que integre las librerías usadas con un funcionamiento correcto y que sirva de pilar para lograr el objetivo final de este TFG.

6.3.1. Primer ensayo: Mano izquierda y 10 signos.

Para este primer ensayo se ha grabado cada signo un total de 30 veces, en diferentes posiciones y a diferentes velocidades. Para ello, se ha utilizado únicamente la mano izquierda y la información que se ha capturado ha sido la expuesta en el apartado 5.4.

Repeticiones grabadas de cada signo	30 repeticiones
Repeticiones totales	300 repeticiones
Mano empleada	Mano izquierda
Información capturada	Apartado 5.4.

Tabla 6.14. Resumen primer ensayo

Tras realizar cada gesto 5 veces y calcular la media aritmética de las similitudes, se han obtenido los siguientes resultados:

Signos realizados

Diccionario	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**
S1	11.3%	5.53%	8.17%	4.82%	9.74%	10.38%	12.7%	7.55%	10.38%	7.61%
S2	7.89%	**20.18%**	8.55%	8.29%	9.32%	9.40%	6.67%	8.66%	8.9%	8.11%
S3	7.86%	7.05%	13.21%	6.29%	7.42%	9.92%	10.63%	8.28%	9.08%	9.94%
S4	5.72%	6.88%	6.57%	**30.6%**	9.08%	7.74%	6.37%	6.52%	7.48%	6.34%
S5	7.01%	10.37%	8.15%	6.86%	**25%**	7.21%	5.71%	7.74%	7.09%	7.8%
S6	11.58%	16.42%	**13.41%**	14.41%	9.61%	**16.03%**	12.02%	7.64%	11.26%	14.14%
S7	13.9%	8.56%	12.74%	9.07%	8.48%	12.09%	**17.43%**	9.49%	11.74%	14.76**%**
S8	5.65%	6.5%	6.67%	5.47%	5.9%	7.47%	7.04%	14.42**%**	7.08%	6.22%
S9	**19.13%**	11.91%	10.6%	8.04%	9.21%	9.72%	8.65%	11.7%	**17.74%**	10.08%
S10	9.96%	6.6%	11.93%	6.15%	6.6%	10.04%	12.78%	**18%**	9.25%	**15%**

Tabla 6.15. Matriz de correlación del primer ensayo.

Se han traducido correctamente **7/10** signos.

Comentarios y observaciones

→ **Sobre los resultados**: Los signos que han fallado, son en los que hay que diferenciar especialmente la posición de los dedos. Además, dentro de los que sí son traducidos correctamente, se observa que hay una alta probabilidad de haber

detectado otro signo en el cual hay cierto parecido en la posición de los dedos. Los gestos que se basan principalmente en la posición de la mano (y no tanto en los dedos) son detectados sin problemas y además tienen la similitud más alta de la matriz (S4 y S5).

→ **Sobre el proceso**: Hay que emplear bastante tiempo para grabar las 300 repeticiones. Además, también tarda bastante tiempo en entrenar el sistema. Por otra parte, parece que no depende mucho la manera de realizar el signo a traducir con la manera que ha sido grabado el signo del diccionario. (Velocidad, posición inicial de la mano, etc.)

Mejoras para el siguiente ensayo.

- ✓ **Capturar más información de los dedos**. Por ejemplo, algún atributo de los huesos del dedo.
- ✓ **Grabar menos repeticiones** para cada signo. Ya que se ha observado que no depende mucho de esto a la hora de traducir. Además, puede que al tener un mayor rango de decisión en cada signo sea más difícil para el sistema decidir qué signo estamos realizando, y por tanto de lugar a fallo.

6.3.2. Segundo ensayo: Incremento de información capturada.

Para este segundo ensayo se ha grabado cada signo un total de 20 veces, en diferentes posiciones y a diferentes velocidades. Para ello, se ha utilizado únicamente la mano izquierda. Con respecto a la información que se ha capturado, a la expuesta en el apartado 5.4 se le ha añadido la dirección del hueso "distal" para cada dedo *(Bone::* `TYPE_DISTAL` *::direction(x, y, z))*.

¿Por qué el hueso distal?

Según se ha podido observar, al realizar un signo el hueso que más cambios presenta en su dirección es el hueso distal. ***Por ejemplo****, cuando cerramos el puño (sin contar movimiento en la posición espacial de la mano) este hueso presenta un cambio de 270º en su dirección. Mientras que el intermedio cambia 180º, el proximal 90º y el metacarpal 0º.* Por tanto, la información que será capturada ahora es la siguiente:

- ✓ Hand::palmposition
- ✓ Hand::palmnormal
- ✓ Hand::direction
- ✓ Finger::tipposition
- ✓ Finger::direction
- ✓ Bone::**`TYPE_DISTAL::`**direction

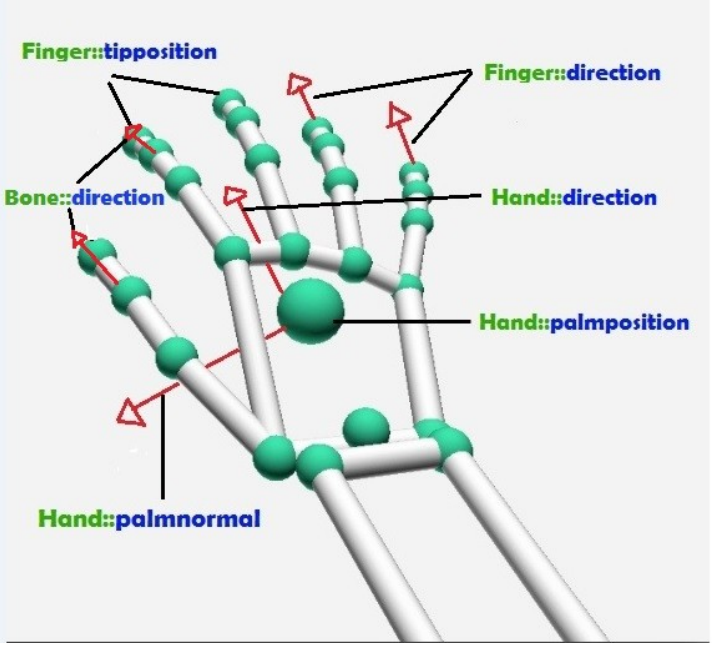

Figura 6.16. Nueva información capturada

Repeticiones grabadas de cada signo	20 repeticiones
Repeticiones totales	200 repeticiones
Mano empleada	Mano izquierda
Información capturada	Apartado 5.4. + Bone::direction

Tabla 6.17. Resumen segundo ensayo.

Tras realizar cada gesto 5 veces y calcular la media aritmética de las similitudes, se han obtenido los siguientes resultados:

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
S1	18,43%	3,65%	4,96%	3,36%	3,83%	9.82%	5,79%	4,64%	5,7%	5,00%
S2	10,94%	**20,02%**	10,96%	15,32%	12,84%	8,85%	6,55%	9,59%	9,02%	9,25%
S3	5,2%	7,92%	**16,62%**	5,04%	8,11%	8,71%	7,70%	6,16%	7,1%	12,49%
S4	11,83%	13,19%	11,46%	**27,84%**	11,45%	7,87%	7,58%	9,95%	9,33%	9,57%
S5	11,15%	13,39%	10,28%	11,71%	**27,41%**	5,86%	4,38%	9,32%	7,57%	8,86%
S6	6,26%	12,94%	7,68%	7,18%	14,64%	**16,06%**	19,17%	7,16%	14,6%	6,42%
S7	5,08%	11,88%	15,73%	11,93%	5,18%	14,52%	**25,74%**	12,71%	11,42%	10,9%
S8	6,42%	4,8%	6,48%	7,02%	4,61%	9,27%	8,46%	14%	7,11%	7,33%
S9	**19,2%**	7,81%	6,93%	6,77%	7,71%	10,24%	8,31%	9,72%	**23,13%**	4,17%
S10	5,49%	4,7%	8,9%	3,83%	4,22%	8,80%	6,32%	**18,63%**	5,02%	**26,01%**

Tabla 6.18. Matriz de correlación del segundo ensayo.

Se han traducido correctamente **8/10** signos.

Comentarios y observaciones

→ **Sobre los resultados**: Los signos que han fallado son basado tanto en la posición de los dedos como en el movimiento de la mano, lo importante es que ha sido por una diferencia de probabilidad muy pequeña. Además dentro de los que sí son traducidos correctamente, se observa una relativa mejoría con respecto a la prueba anterior que se refleja en un aumento de la similitud.

→ **Sobre el proceso**: Se ha reducido tanto el tiempo empleado en grabar el diccionario como el tiempo de entrenamiento del sistema, aunque sigue siendo un tanto elevado. Además parece que la similitud se ha mejorado también debido a la disminución del número de repeticiones (parece que es mejor tener un menor rango de decisión dentro de cada signo)

Mejoras para el siguiente ensayo:

- ✓ **Grabar menos repeticiones** para cada signo. Se sigue observando que no hay una gran dependencia en el modo que se realiza el gesto y el modo que ha sido grabado. Además, parece que la reducción del número de repeticiones también ha contribuido a la mejoría.

Con respecto a la información capturada de los dedos, se va a considerarla suficiente por ahora debido a la mejoría obtenida. Si en la siguiente prueba no tenemos una mejoría considerable, volveremos a tratar esta solución.

6.3.3. Tercer ensayo: Reducción del número de repeticiones del gesto.

Para este tercer ensayo se ha grabado cada signo un total de 5 veces, en diferentes posiciones y a diferentes velocidades. Para ello, se ha utilizado únicamente la mano izquierda. Con respecto a la información que se ha capturado, se trata de la misma utilizada en la segunda prueba.

Repeticiones grabadas de cada signo	5 repeticiones
Repeticiones totales	50 repeticiones
Mano empleada	Mano izquierda
Información capturada	Apartado 5.3 + Bone::direction

Tabla 6.19. Resumen tercer ensayo.

Tras realizar cada gesto 5 veces y calcular la media aritmética de las similitudes, hemos obtenido los siguientes resultados:

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
S1	**23,09%**	8,19%	7,84%	7,69%	6,6%	7,12%	7,12%	8,28%	7,69%	7,29%
S2	11,46%	**41,01%**	7,22%	7,77%	8,03%	7,06%	7,08%	7,97%	8,38%	7,46%
S3	5,67%	4,92%	**22,35%**	5,32**%**	5,32%	11,83%	9,21%	7,27%	7,48%	6,83%
S4	7,43%	6,33%	6,29%	**42,99%**	5,26%	6,72%	6,71%	6,02%	6,12%	4,63%
S5	7,6%	7,46%	6,36%	5,21%	**41,19%**	6,63%	6,1%	6,14%	7,28%	6,97%
S6	6,88%	5,85%	10,15%	6,45%	6,31%	**23,11%**	17,78%	8,49%	8,93%	8,23%
S7	7,45%	6,46%	10,45%	6,84%	6,67%	10,86%	**19,26%**	9,92%	10,05%	9,84%
S8	6,85%	5,58%	8,46%	4,77%	5,19%	6,79%	6,97%	**22,14%**	7,29%	12,45%
S9	17,07%	8,95%	12,99%	8,13%	9,57%	11,69%	12,23%	11,32%	**29,36%**	12,99%
S10	6,5%	5,25%	7,89%	4,83%	5,86%	8,19%	7,54%	12,45%	7,42%	**23,31%**

Tabla 6.20. Matriz de correlación del tercer ensayo.

Se han traducido correctamente **10/10** signos.

Comentarios y observaciones

→ **Sobre los resultados**: Se ha conseguido traducir correctamente el 100% de los signos. Además se puede observar un aumento general en el coeficiente de correlación de los signos que ya habían sido traducidos correctamente en la prueba anterior. Por tanto, no solo se ha conseguido un 100% de acierto, sino que también se ha conseguido aumentar la similitud.

→ **Sobre el proceso**: Se ha reducido enormemente tanto el tiempo empleado en grabar el diccionario como el tiempo de entrenamiento del sistema.

Puede concluirse entonces que la información capturada por Leap Motion es suficiente. Además, se ha observado con esta prueba que no es mejor tener un mayor número de repeticiones en cada signo ya que el espacio de decisión sería muy grande. Basta con tener las repeticiones necesarias para englobar a cada signo en el espacio, posición y velocidad donde se supone que será realizado.

Con esta prueba, se ha logrado el objetivo de traducir correctamente 10 gestos. Por se puede concluir que la aplicación desarrollada es exacta y funcional

7. CONCLUSIÓN

En este trabajo de fin de grado, se ha realizado un estudio intensivo de las funciones básicas para el reconocimiento de la lengua de signos utilizando el dispositivo Leap Motion. Se ha centrado principalmente en optimizar la inclusión de este dispositivo en el reconocimiento de signos pertenecientes a esta lengua ya que es necesaria una gran diferenciación entre posición de dedos y manos. Este dispositivo, sacado al mercado en 2013, posee una gran precisión de seguimiento para manos y dedos, además de un reducido tamaño y precio. Esto y el añadido de que no es nada parecido a lo que se había visto hasta ahora, hacen que este proyecto sea ambicioso e innovador.

Para lograr el objetivo de este TFG, se ha desarrollado una aplicación escrita en C++ que cuenta con todas las funciones diseñadas en el sistema de reconocimiento de gestos. Además, se han realizado una serie de ensayos para probar su correcta funcionalidad. Finalmente con todo este trabajo expuesto, se han logrado alcanzar todos los objetivos que fueron propuestos para este trabajo de fin de grado.

Durante la realización de este trabajo en estos casi cinco meses, he adquirido numerosos conocimientos totalmente nuevos para mí, y he reforzado otros que ya tenía.
He aprendido a programar con un nuevo dispositivo como es el Leap Motion, tras previo estudio de su extensa API y diversas pruebas de iniciación a su programación. La página oficial cuenta con una documentación totalmente acertada que para mí ha sido fundamental para llegar a entender con certeza el funcionamiento del Leap Motion. Además, el trabajar principalmente con vectores, números y coordenadas en el espacio ha sido otra ventaja añadida.

He estudiado también diversos aspectos de la Lengua de Signos, que era un mundo totalmente desconocido para mí y que ha sido fundamental para poder centrar el software desarrollado y su funcionalidad en el objetivo de crear un traductor de dicha lengua. Otro gran detonante en la motivación de este trabajo, ha sido la idea de poder desarrollar una aplicación con un fin social como este.
También he tenido que estudiar el lenguaje de programación C++ con el cual he desarrollado dicha aplicación. Era un lenguaje prácticamente desconocido para mí, pero me decanté por él al ser el lenguaje usado en la librería de reconocimiento y al ser el lenguaje nativo de Leap Motion.

He usado un IDE con el que no había trabajo como es Visual Studio, y he reforzado los conocimientos de inclusión de varias librerías en el mismo proyecto.

Durante el desarrollo, me he reforzado y mejorado mucho en la redacción debido a la memoria. Además de darme cuenta de la importancia de tener bien comentado el código para facilitar el trabajo presente y futuro, y la importancia de tener un plan de trabajo y marcar objetivos.

Por lo que a mí respecta, ha sido muy gratificante alcanzar los objetivos propuestos.

8. TRABAJO FUTURO

Este proyecto da la posibilidad de abrirse a multitud de líneas de trabajo. Son muchas las puertas que han quedado abiertas a pesar de haber desarrollado un extenso trabajo que cumple con todos los objetivos propuestos para este trabajo de fin de grado. Más allá de los objetivos que este proyecto tenía como finalidad, podemos encontrar las siguientes líneas de trabajo las cuales se intentarán abarcar en un futuro.

8.1. Desarrollo de una interfaz en 3D.

Uno de los principales objetivos futuros y que ya se encuentra en desarrollo, será la creación de una interfaz gráfica en 3D que permita visualizar y monitorizar las manos y dedos en la aplicación desarrollado. Además, esta interfaz será controlada principalmente por el dispositivo Leap Motion para agilizar la interacción.

Figura 8.1. Posibles interfaces gráficas

Además otro de los principales objetivos que sería apropiado abarcar, es el aumento de la precisión del sistema para reducir la tasa de fallos. Para ello, habrá que realizar los cambios en el sistema que sean necesarios hasta lograr que los signos sean traducidos con una similitud ideal de al menos un 90%.

8.2. Enseñanza y mini juegos.

Otro posible trabajo futuro muy interesante sería añadir un módulo a nuestra aplicación que permita aprender la lengua de signos. Un tipo de profesor virtual que enseñe

el correcto posicionamiento de manos y dedos en el vocabulario y sintaxis de esta lengua. Y que posea algún método de auto-evaluación que haga saber si se está o no progresando. Esta opción parece bastante interesante y de gran utilidad para facilitar la enseñanza a toda persona que quiera aprender la lengua.

Figura 8.2. Posible módulo de enseñanza.

Además para ayudar a esta enseñanza y poner a prueba los conocimientos adquiridos, o simplemente para el entretenimiento, otra futura línea de trabajo sería desarrollar mini juegos en los que sea necesario usar la lengua de signos para jugar. Por ejemplo, una sopa de letra, el juego de la batalla naval, etc.

Figura 8.3. Posible minijuego sopa de letras.

8.3. Nuevas plataformas.

En esta línea, la idea principal es el desarrollo de la aplicación para su uso en una tablet. Para ello, sería necesario optimizar el software desarrollado para esta plataforma e integrar el dispositivo Leap Motion en el hardware de la tablet. Esta nueva posibilidad

permitiría disponer de la aplicación prácticamente en cualquier lugar y en cualquier momento. Este es un aspecto que potenciaría un uso cotidiano y daría la movilidad necesaria para facilitar y agilizar aún más la comunicación entre una persona sordomuda y otra que desconozca la lengua de signos.

La compañía Motion Savvy comenzó en Noviembre de 2012 a desarrollar este prototipo y actualmente se encuentran en una fase muy avanzada en la que ya comercializan su propio producto.

Figura 8.4. Posible prototipo de la aplicación para tablet similar al usado por MotionSavvy

8.4. Realidad virtual.

El mundo de la realidad virtual está teniendo un gran crecimiento e interés actualmente. Son muchas las temáticas utilizadas en la realidad virtual. Desde videojuegos hasta visualizar y caminar por la casa de tus sueños antes de ser construida. Y como era de esperar, la compañía Leap Motion también ha entrado en este mundo. Además de una API y SDK especializada en la realidad virtual, están comercializando un módulo que permite incluir el dispositivo Leap Motion a unas gafas y cascos de realidad virtual. Su precio es de 19,99 € y se puede comprar en la página web oficial entre otros. Este módulo está optimizado especialmente para HTC Vive y para Oculus Rift CV1. El precio de las gafas de realidad virtual Oculus Rift es de 599$.

Figura 8.5. Oculus Rift + Leap Motion

¿**Qué tiene que ver la aplicación desarrollada en este TFG con esto?** Para crear dicha aplicación ha sido preciso desarrollar y diseñar un sistema de reconocimiento de gestos que use la información captada por Leap Motion, y que mediante el algoritmo Dynamic Time Warping (DTW) sea capaz de reconocer los gestos previamente grabados. Este sistema podría ser usado en el ámbito de la realidad virtual, haciendo posible llevar a cabo acciones dependiendo del gesto que estemos realizando.

9. BIBLIOGRAFÍA

[1] Developer Documentation. © 2016, Leap Motion, Inc. [en línea] [última consulta: Mayo 2016]. Disponible en: https://developer.leapmotion.com/

[2] STEVEN PARKHUST y DIANNE PARKURST. *Un estudio lingüístico: Variación de las lenguas de signos en España*. Revista Española de Lingüística de Lengua de Signos (RELLS), Madrid 2001. Promotora Española de Lingüística (PROEL) [última consulta: Marzo 2016].

[3] Microsoft Windows, Kinect [en línea] [última consulta: Marzo 2016]. Disponible en: https://developer.microsoft.com/es-es/windows/kinect

[4] CONFEDERACIÓN ESTATAL DE PERSONAS SORDAS (CNSE). Dossier de prensa: *75 años luchando por la igualdad de oportunidades para todas las personas sordas* [en línea]. Año 2011. 10 pág. [última consulta: Marzo 2016]. Disponible en: http://www.fesorcam.org/wp-content/descargas/2011/06/Dossier-CNSE-2011.pdf

[5] MICHAEL NOWICKI, OLGIERD PILAREZYK y otros. Bachelor's thesis: *Gesture recognition library for leap motion controller* [en línea]. Poznan, 2014 [última consulta: Febrero 2016]. Disponible en: http://www.cs.put.poznan.pl/wjaskowski/pub/theses/LeapGesture_BScThesis.pdf

[6] How Does the Leap Motion Controller Works? © 2016 Leap Motion, Inc. [en línea] [última consulta: Marzo 2016]. Disponible en: http://blog.leapmotion.com/

[7] Leap Motion Community [en línea] [última consulta: Mayo 2016]. Disponible en: https://community.leapmotion.com/

[8] Programación en C++. Wikilibros: libros libres para un mundo libre [en línea]. 24 Mayo 2016, 14:55 [última consulta: Abril 2016]. Disponible en: https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B

[9] Visual Studio. © 2016 Microsoft [en línea] [última consulta: Mayo 2016]. Disponible en: https://www.visualstudio.com/

[10] T. ARMOUR <jaanga en © 2016 GitHub, Inc.> *JestPlay* [en línea]. 20 Diciembre 2013 [última consulta: Marzo 2016]. Disponible en: https://github.com/jaanga/gestification/tree/gh-pages/cookbook/jest-play

[11] Leapjs-playback. < Leapmotion en © 2016 GitHub, Inc.> [en línea].20 Junio 2015 [última consulta: Marzo 2016]. Disponible en: https://github.com/leapmotion/leapjs-playback

[12] R. O'LEARY <roboleary en © 2016 GitHub, Inc. > *LeapTrainer.js*. [en línea].13 Noviembre 2013 [última consulta: Marzo 2016]. Disponible en: https://github.com/roboleary/LeapTrainer.js/

[13] NICHOLAS GILLIAN. *Gesture Recognition Toolkit* [en línea]. 21 Abril 2016, 17:16 [última consulta: Mayo 2016]. Disponible en: http://www.nickgillian.com/wiki/pmwiki.php/GRT/GestureRecognitionToolkit

[14] NICHOLAS GILLIAN, R.BENJAMIN KNAPP, y SILE O'MODHRAIN. *Recognition Of Multivariate Temporal Music Gestures Using N-Dimensional Dynamic Time Warping* [en línea]. NIME'11. Oslo, Norway. 30 Mayo 2011 [última consulta: Marzo 2016]. Disponible en: http://www.nickgillian.com/papers/Gillian_NDDTW.pdf

[15] PSB.UGENT. *DTW algorithm* [en línea] [última consulta: Marzo 2016]. Disponible en: http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm

[16] MÜLLER, Meinard. *Information retrieval for music and motion* [en línea]. 2007 [última consulta: Abril 2016]. *Capítulo 4: Dynamic Time Warping.* Disponible en: http://www.springer.com/cda/content/document/cda_downloaddocument/9783540740476-c1.pdf?SGWID=0-0-45-452103-p173751818.
ISBN 9783540740483.

[17] MARÍA ÁNGELES RODRÍGUEZ GONZÁLEZ. Lengua de Signos [en línea]. Biblioteca virtual Miguel de Cervantes. [última consulta: Abril 2016]. Disponible en: http://www.cervantesvirtual.com/servlet/SirveObras/signos/01473952099104051054480/index.htm

[18] SÉMATOS, el portal europeo de la lengua de signos [en línea]. Video diccionario de la lengua de signos española [última consulta: Mayo 2016]. Disponible en: http://www.sematos.eu/lse.html

[19] E MOYANO BAQUERO. *Análisis de las matrices de covarianza y correlación* [en línea]. 2011 [última consulta: Mayo 2016]. Disponible en: https://www.dspace.espol.edu.ec/bitstream/123456789/5775/8/Cap6.doc.

[20] Motion Savvy [en línea] [última consulta: Mayo 2016]. Disponible en: http://www.motionsavvy.com/

[21] Oculus Rift [en línea] [última consulta: Mayo 2016]. Disponible en: https://www.oculus.com/en-us/rift/

ANEXO 1. INSTALACIÓN LEAP MOTION.

En este anexo se va a explicar de forma detallada la puesta en marcha del dispositivo Leap Motion y la instalación de su correspondiente controlador para poder usarlo en el PC.

1) Primero hay que descargar el SDK del dispositivo directamente desde la página web oficial, en el apartado "Get Started". Es recomendable descargar la última versión disponible. Habrá que descargar la SDK correspondiente al sistema operativo que se vaya a utilizar: Windows, iOs o Linux.

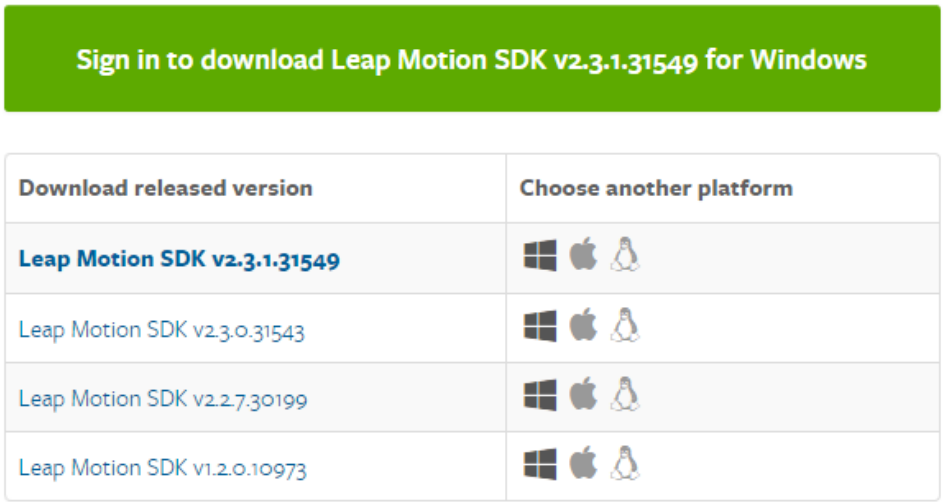

Figura 1.1. Descarga SDK Leap Motion

2) El segundo paso, es la instalación del controlador del dispositivo. Para ello, una vez descomprimido el archivo descargado en 1). Abrimos el ejecutable **Leap_Motion_Installer.**

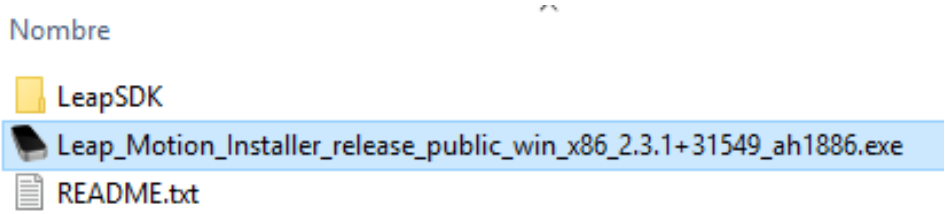

Figura 1.2. Archivos descargados con el SDK

3) Se abrirá un asistente de instalación. Tras pulsar la tecla "*Siguiente*", el controlador se instalará y ya se podrá disfrutar del dispositivo Leap Motion en el PC. Se aconseja elegir la opción: Iniciar experiencia Leap Motion para tener una primera toma de contacto con el dispositivo.

Figura 1.3. Asistente de instalación Leap Motion.

4) Con esto, se instalará automáticamente la aplicación "Leap Motion App Home" en la cual se podrán descargar las diversas aplicaciones que han sido desarrolladas para el dispositivo por los desarrolladores.

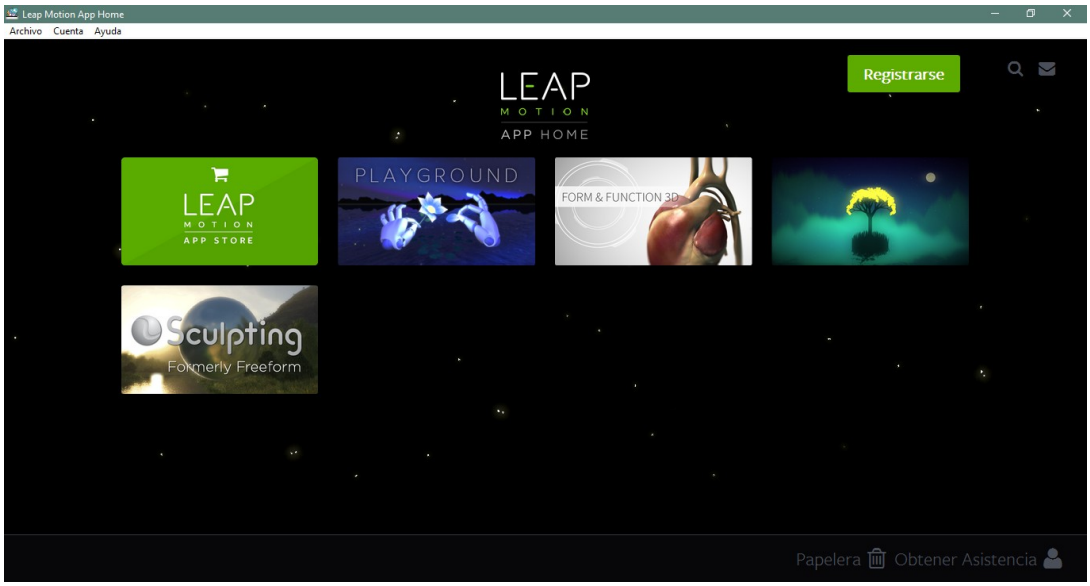

Figura 1.4. Pantalla principal de Leap Motion App Home.

5) Se recuerda, como ya se explicó en el apartado 4.3.2, que el controlador cuenta principalmente con dos aplicaciones:

- **Panel de control de Leap Motion**: permite cambiar prácticamente cualquier opción del dispositivo. Como por ejemplo: área de interacción del dispositivo, mecanismo de ahorro de energía, recalibrar el dispositivo, resolución o informar acerca de problemas, configuración de rastreo, actualizaciones, etc. Además cuenta con un apartado "Acerca de" donde se puede encontrar información relativa a la versión del SDK y enlaces a las páginas web oficiales.

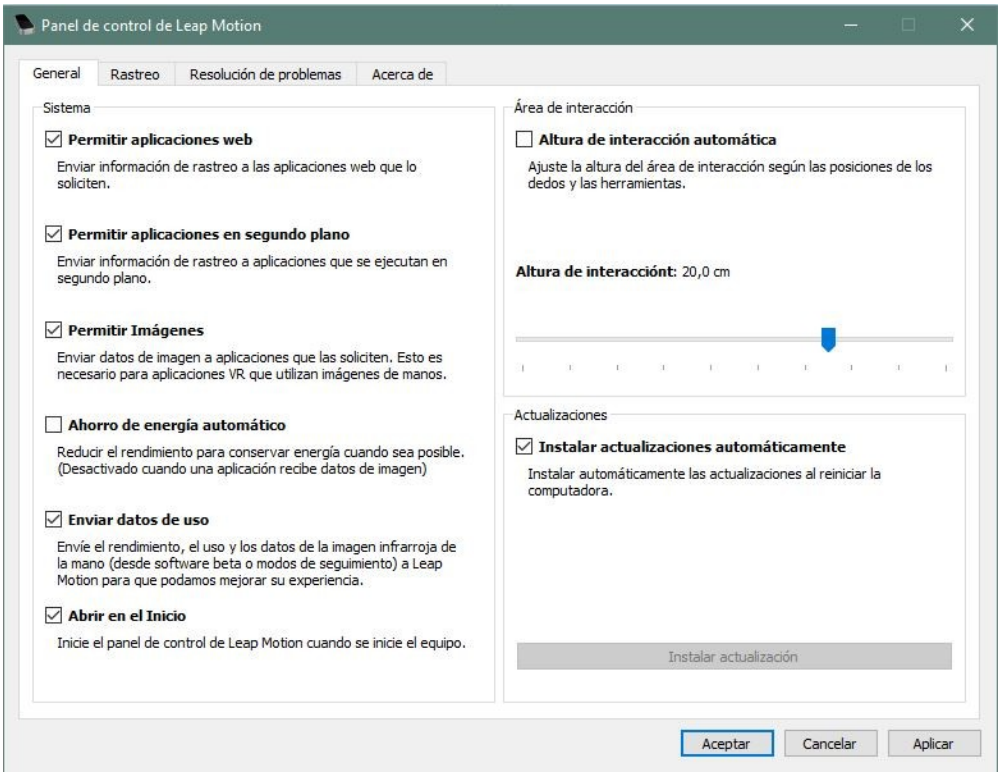

Figura 1.5 Panel de control de Leap Motion

- **Visualizador**: muestra un espacio gráfico 3D con un sistema de coordenadas cartesianas (x, y, z). El eje "x" pertenece a la parte horizontal, el "y" va verticalmente desde el sensor y el "z" perpendicular a la pantalla. En este gráfico 3D se monitorizan las manos, dedos y elementos *"pointables"* [Figura 4.9] siendo representadas y desplazándose tal y como el usuario las mueve. Además, ofrece la posibilidad de visualizar información relativa al muestreo, coordenadas, dibujar la Interaction Box, cambiar el tipo de representación, etc.

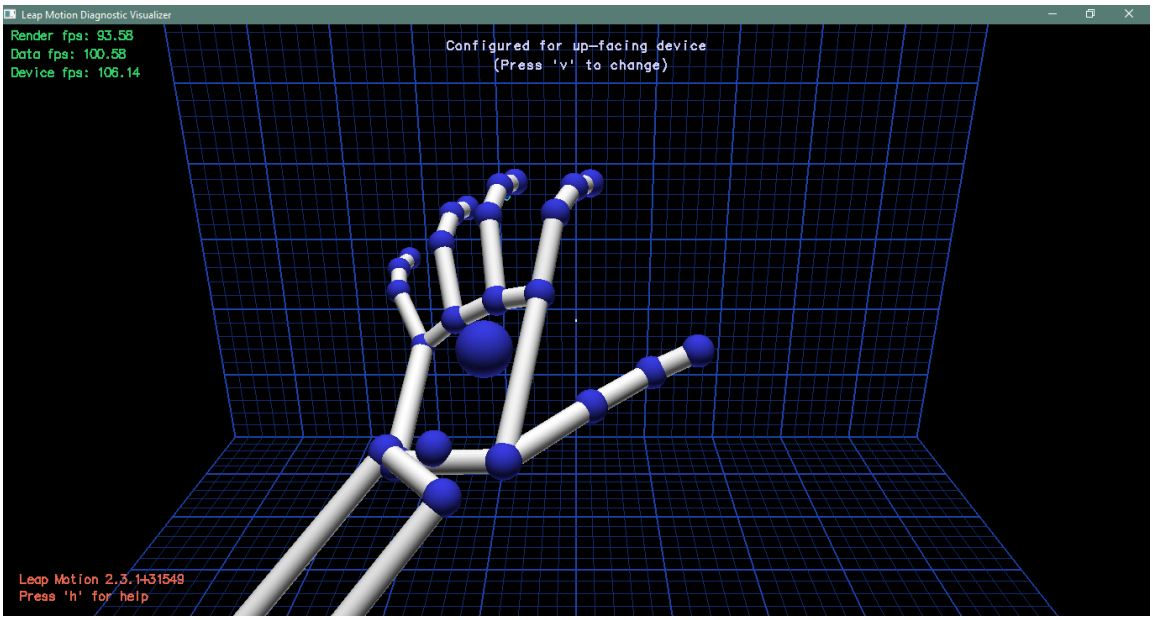

Figura 4.11. Visualizador de diagnóstico de Leap Motion