Universidad Zaragoza
1542

Trabajo Fin de Grado

Grado en Ingeniería informática

Definición e implementación de un vocabulario de signos para la interacción con distintos dispositivos

Autor

Sergio Igea Bruch

Director

Carlos Sagüés Blázquiz

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

Septiembre 2014

Definición e implementación de un vocabulario de signos para la interacción con distintos dispositivos

Resumen

En los últimos años han aparecido nuevas técnicas de interacción con dispositivos electrónicos, como el control por voz, o el movimiento de nuestro cuerpo capturado por sensores. El control a través del movimiento del cuerpo se ha utilizado especialmente en el ámbito del ocio, en videoconsolas, concretamente Kinect o ASUS Xtion, pero se ha desarrollado poco en el ámbito domótico. Con el fin de aplicar esta nueva forma de interactuar en el ámbito domótico, en este trabajo se ha estudiado el sensor de características similares a Kinect, Leap Motion y las posibilidades que ofrece.

Inicialmente, con el propósito de comprender la técnica de visión que es utilizada por Leap Motion para la obtención de los datos de profundidad, se han analizado y procesado los datos en crudo que son capturados por el sensor. Estos datos consisten en las dos imágenes capturadas por las cámaras del sensor únicamente, que se asemejan a la visión estero, utilizadas para construir el mapa de profundidad. Una vez realizada esta identificación, se ha desarrollado un identificador gestual utilizando las imágenes obtenidas.

Después de entender el funcionamiento, se ha procedido a la definición de un vocabulario de gestos, a partir de la información capturada por Leap Motion utilizando sus propias librerías. Definido el vocabulario, se ha realizado un estudio de la *precision* y el *recall* de cada gesto en los diferentes modos de funcionamiento del Leap. Además se ha analizado su variación con el cambio de alguna condición externa, como por ejemplo diferentes alturas de realización del gesto. Se ha concluido que de los cuatro modos de funcionamiento del sensor el mejor modo es el *modo equilibrado*, pero siendo el *modo robusto* el menos sensible al cambio de altura.

Finalmente, con el fin de evaluar una aplicación que utilice el Leap Motion, se ha realizado la implementación del vocabulario definido para controlar una placa de inducción. Se ha realizado un estudio para identificar cuáles son los mejores gestos para este propósito. Una vez identificados se ha implementado de dicha aplicación.

Índice

1. **INTRODUCCIÓN** **1**

1.1. Objetivos y alcance **2**

1.2. Marco de trabajo **2**

1.3. Herramientas de trabajo **3**

2. **VISIÓN 3D** **5**

2.1. Técnicas pasivas **5**

2.1.1. Visión estéreo 5

2.1.2. Forma a partir de textura y movimiento 7

2.2. Técnicas activas **8**

2.2.1. Luz estructurada 8

2.2.2. Telemetría laser 9

3. **LEAP E INFORMACIÓN 3D** **11**

3.1. Cálculo de la profundidad **12**

3.2. Separación de manos y brazos **14**

3.2.1. Método Otsu [8] 15

3.3. Reconocimiento de gestos estáticos **17**

4. **RECONOCIMIENTO DE GESTOS USANDO LEAP** **25**

4.1. Información capturada **25**

4.2. Gestos estáticos **26**

4.3. Gestos dinámicos **27**

5. **ANÁLISIS DE GESTOS 31**

5.1. Introducción 31

5.2. Gestos de interés y modos de funcionamiento 32

5.2.1. Palma abierta 33
5.2.2. Dos palmas 33
5.2.3. Puño 34
5.2.4. Círculo vertical derecha 34
5.2.5. Círculo vertical izquierda 35
5.2.6. Palma vertical 35
5.2.7. Círculo horizontal derecho 36
5.2.8. Círculo horizontal izquierda 36
5.2.9. KeyTap derecha 37
5.2.10. KeyTap izquierda 37

5.3. Sensibilidad al cambio de condiciones 38

5.4. Particularidades 40

6. **APLICACIÓN DESARROLLADA 43**

6.1. Gestos elegidos para la aplicación 44

6.2. Diferentes configuraciones elegidas 45

6.3. Análisis de la usabilidad, robustez y facilidad de aprendizaje de la aplicación 47

6.3.1. Resultados obtenidos sin fase de entrenamiento 47
6.3.2. Fase de entrenamiento 49
6.3.3. Resultados obtenidos después de fase de entrenamiento 49

6.4. Aplicación 51

7. **CONCLUSIONES Y TRABAJO FUTURO 55**

8. **BIBLIOGRAFÍA 59**

Capítulo 1

Introducción

Desde la aparición de los primeros aparatos electrónicos se ha buscado una forma sencilla e intuitiva para interactuar con ellos. La primera forma de interacción se remonta a tarjetas perforadas o a comandos de texto, los cuales eran introducidos con un teclado. Esta forma de interacción fue la utilizada en las primeras computadoras. Con el paso del tiempo se produjo la aparición del ratón en 1968, un apuntador que facilitaba la interacción.

Desde entonces los cambios en la forma de interactuar entre una persona y una maquina no han sido muy significativos. Hasta los últimos años, cuando se ha producido un gran cambio que se pone de manifiesto con la aparición de las pantallas táctiles, en las cuales se utilizan las manos para interactuar con el dispositivo. Esto supone la existencia de un contacto entre el dispositivo y la persona, pero existen situaciones en las cuales no es adecuado el contacto, o el reconocimiento del mismo resulta dificultoso. Surgen también en estos últimos años los controles por voz, y a través de gestos. El uso de estos últimos se ha extendido sobre todo en el área de entretenimiento, con el uso de dispositivos que se manejan a través de gestos realizados directamente por nuestro cuerpo. Se pueden utilizar dispositivos de captura como Kinect, más adecuado para el cuerpo completo y Leap Motion, orientado específicamente a la identificación de gestos hechos únicamente con las manos.

Esta última forma de interacción con los dispositivos ha sido aplicada poco en otros entornos como la domótica o la interacción con electrodomésticos. Esta podrá llegar a sustituir completamente a los mecanismos que hacen posible la interacción, como los botones, ruletas o los paneles táctiles, debido a que puede solucionar ciertos problemas que existen actualmente, además de mejorar la accesibilidad.

1.1. Objetivos y alcance

Con el fin de aplicar una nueva forma de interactuar en el ámbito domótico, el objetivo de este trabajo es el estudio del funcionamiento del sensor Leap Motion, analizando la información en crudo que es capturada. Utilizando técnicas de visión, se desarrollará un identificador gestual, con la información en crudo obtenida.

Siguiendo con el estudio del sensor, otro de los objetivos es la definición de un vocabulario de gestos que puedan ser identificados a partir de los datos capturados por este. Se realizará el estudio de cada uno de estos gestos, analizando como de bien son identificados, con una variación tanto de las condiciones externas, como del modo de funcionamiento del Leap.

El último objetivo es implementar una aplicación para el control de una placa de inducción usando Leap Motion. Para ello se realizará la implementación del vocabulario de gestos definido para la interacción. Todos aquellos gestos que puedan ser utilizados en la interacción con la placa tienen que poseer la característica de ser robustos, ya que, en ese contexto, se podrían generar situaciones peligrosas. Esta información será obtenida del análisis de cada uno de los gestos.

Para poder cumplir con los objetivos del trabajo, este ha sido estructurado en las siguientes fases.

I. Estudio de la documentación del sensor que se va a utilizar.
II. Estudio de la codificación de datos obtenidos por el sensor.
III. Estudio de técnicas y librerías para el procesamiento de estas imágenes.
IV. Análisis de los diferentes modos de funcionamiento del sensor.
V. Análisis de las características de los diferentes gestos capturados.
VI. Desarrollo e implementación de una aplicación que permita interactuar con una cocina de inducción.

1.2. Marco de trabajo

Este trabajo se realiza en colaboración con la empresa BSH, dedicada al diseño y fabricación de electrodomésticos. El objetivo final es poder llegar a interactuar con los electrodomésticos a través de gestos, realizando la implementación del vocabulario de gestos sobre un microprocesador de capacidades de memoria y cómputo limitadas, ya sean capturados por el Leap Motion o bien otro sensor de características similares. De esta manera poder solucionar algunos de los problemas que existen con las interfaces que se utilizan actualmente en este ámbito.

1.3. Herramientas de trabajo

En este trabajo se utiliza el sensor Leap Motion, y las librerías que el propio sensor proporciona para la implementación de aplicaciones. Una computadora de carácter general en la cual se conecta el sensor y se realizan todas las operaciones de procesado de información, pero con el objetivo de poder sustituirlo en un futuro por un microprocesador de capacidades limitadas.

Tanto la parte de implementación de la aplicación final como el procesado de la información en crudo proporcionada por el sensor, se ha realizado íntegramente en el lenguaje C++, utilizando el entorno de desarrollo Visual C++ 2010 Express.

Para la implementación de la aplicación final se ha utilizado la biblioteca de utilidades GLUT (*OpenGL Utility Toolkit),* utilizada para rutinas de dibujo, la biblioteca GLUI *(OpenGL User Interface),* utilizada para el control de ventanas y elementos de control. Para el procesado de imágenes se ha empleado la versión 2.4.3 de librería OpenCV.

Capítulo 2

Visión 3D

Las diferentes técnicas ópticas para obtener información 3D [1][2][3], se suelen clasificar en grupos, dependiendo de las características que poseen las fuentes de luz utilizadas. Se dividen en dos grandes grupos, técnicas pasivas y activas. En ambos grupos se tiene control sobre la iluminación de la escena, pero en las técnicas pasivas no es necesario conocer la posición de la fuente de luz.

2.1. Técnicas pasivas

Las técnicas pasivas se pueden aplicar en más situaciones que las técnicas activas, pero debido a que no tienen en cuenta la posición de la fuente de luz se limita la exactitud de la medida. La visión estéreo y la obtención de la forma a partir de textura y movimiento son las técnicas pasivas principales.

2.1.1. Visión estéreo

Las técnicas de visión estéreo por computador tratan de emular el sistema de visión humano, en el que se analizan las diferencias de la proyección de la escena en dos imágenes tomadas desde dos posiciones diferentes.

El proceso de visión estereoscópica se puede dividir en las siguientes etapas,

- Modelado de la cámara
- Adquisición de imágenes
- Extracción de características
- Correspondencia de características
- Determinación de la profundidad

De las fases que se han citado anteriormente, la más compleja es la correspondencia de características, y depende fuertemente de las características que se ha decidido extraer de las imágenes.

Las imágenes pueden ser adquiridas de forma simultánea, a través de la utilización de múltiples cámaras en diferentes posiciones, o en varios momentos temporales con una sola cámara desde puntos distintos. Hay que tener cuidado con esta última forma debido a que si la diferencia temporal es muy grande, pueden haber variado las características de la escena de una forma notable, sobre todo si se tratan de espacios exteriores.

El modelo de cámara más simple (Figura 1) es aquel que posee dos cámaras cuyos ejes ópticos son paralelos. La distancia que los separa es la *línea base*, quedando los ejes ópticos perpendiculares y las líneas epipolares paralelas a esta línea. Las líneas epipolares quedan definidas por la intersección del plano epipolar, plano resultante de la unión de un punto en el espacio con los centros ópticos de las dos cámaras, con el plano de proyección de la cámara. Todos los puntos cuyas proyecciones izquierdas se encuentren en una línea epipolar, sus proyecciones derechas también deben estar contenidas en una misma línea epipolar. La búsqueda de correspondencias entre los puntos se tiene que realizar tanto en el eje "x" como en el eje "y", para disminuir la complejidad, se hace coincidir el eje "y" de las cámaras, quedando el problema reducido a una búsqueda unidireccional.

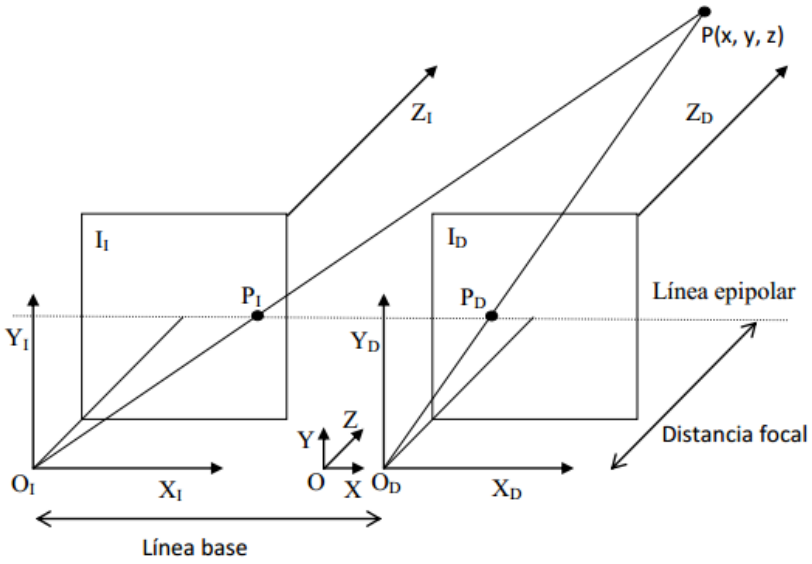

Figura 1: Modelo de cámara.

Con este modelo se obtiene un valor de disparidad d, para cada par de puntos emparejados PI (xI, yI) y PD (xD, yD) dado por d = xI – xD, siendo este el desplazamiento horizontal que se produce. Con el valor de disparidad de cada punto de la imagen se construye una matriz o mapa de disparidad, en el que cada punto de la imagen contiene su valor de disparidad.

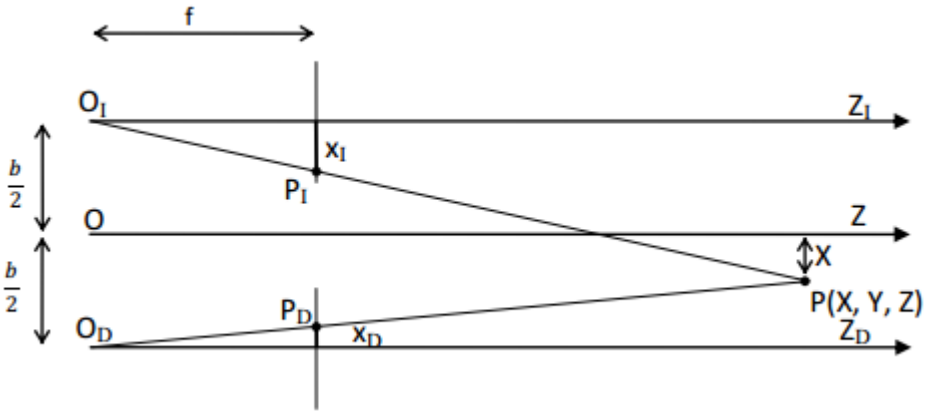

Figura 2: Modelo geométrico de cámara estereoscópica visto desde arriba. El valor de la línea base está determinado por el valor b, la distancia focal es f.

En la figura 2 se puede ver la semejanza de triángulos, con la información dada por el mapa de disparidad. Los cálculos para calcular la profundidad se muestran a continuación,

$$Imagen\ izquierda: \frac{\frac{b}{2}+x}{z} = \frac{x_I}{f} \Rightarrow x_I = \frac{f}{z}\left(x+\frac{b}{2}\right)$$

$$Imagen\ derecha: \frac{\frac{b}{2}-x}{z} = \frac{x_D}{f} \Rightarrow x_D = \frac{f}{z}\left(x-\frac{b}{2}\right)$$

$$d = x_I - x_D = \frac{f*b}{z} \Rightarrow z = \frac{f*b}{d}$$

2.1.2. Forma a partir de textura y movimiento

Las técnicas que intentan obtener la información tridimensional utilizando la textura, se basan en dos efectos que se producen cuando se observa un objeto que posee una textura con patrón regular. El primer efecto es el ángulo con el que se observa la superficie del objeto. Este ángulo puede distorsionar la unidad mínima de textura. El análisis de esta distorsión permite determinar el ángulo que forman las superficies con el sensor del sistema. El segundo efecto es el tamaño de los elementos de la textura, que varía en función de la distancia al observador. El estudio de esta variación sirve para determinar la distancia a la que se encuentran los objetos.

Las técnicas de captura de información a través del movimiento se basan en la propiedad de rigidez de los objetos, de tal manera que los puntos que se encuentren más cercanos al sensor se moverán más rápidos que los que se encuentren más alejados. Estas técnicas tienen la limitación que no pueden ser aplicadas a objetos flexibles o que cambien de aspecto a lo largo del tiempo.

2.2. Técnicas activas

Las técnicas activas son más interesantes que las técnicas pasivas debido a que ofrecen una mayor precisión, pero su implementación es más costosa. Aun así estas técnicas son las más utilizadas en el ámbito industrial.

2.2.1. Luz estructurada

Es la técnica más sencilla que se puede utilizar para obtener información tridimensional de la escena a partir de puntos reconocibles en ella. Para que estos puntos puedan ser reconocidos, se proyecta un patrón de luz estructurada a la escena, normalmente a través de una fuente de luz láser. Esta técnica se basa en el cálculo de triángulos semejantes entre los elementos sensor óptico, emisor de luz y objeto de la escena (Figura 3).

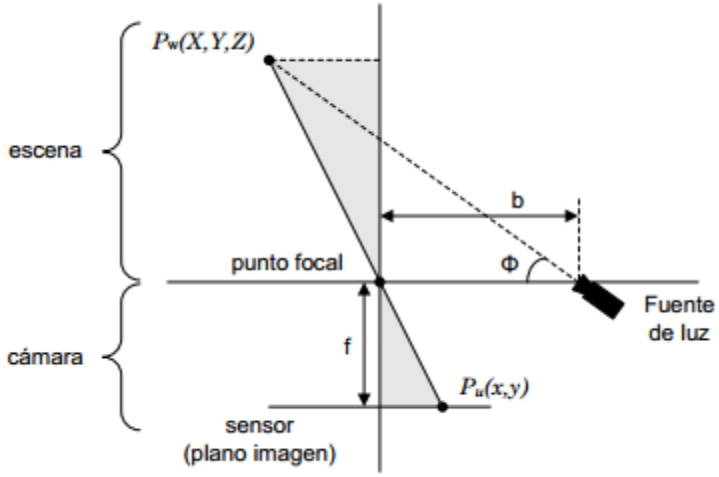

Figura 3: Muestra la semejanza entre triángulos para el punto P.

Con la semejanza de triángulos se puede calcular la posición del punto P con las siguientes ecuaciones, que se obtienen de la semejanza de triángulos (x * Z = X * f) y (y * Z = Y * f).

$$X = \frac{b}{f * \cot(\varphi - x)} * x$$

$$Y = \frac{b}{f * \cot(\varphi - x)} * y$$

$$Z = \frac{b}{f * \cot(\varphi - x)} * f$$

2.2.2. Telemetría laser

La telemetría laser[4] consiste en medir el tiempo de recorrido de un rayo luminoso, hasta la superficie medida. Se puede medir de dos maneras, con la medida del tiempo de vuelo, y el cálculo por diferencia de fase.

A. **Tiempo de vuelo:** el tiempo de vuelo determina la distancia a la escena cronometrando el tiempo del viaje de ida y vuelta de un pulso de luz (Figura 4). La velocidad de la luz es conocida, el tiempo de viaje determina la distancia del viaje de la luz, que es dos veces la distancia entre el escáner y la superficie.

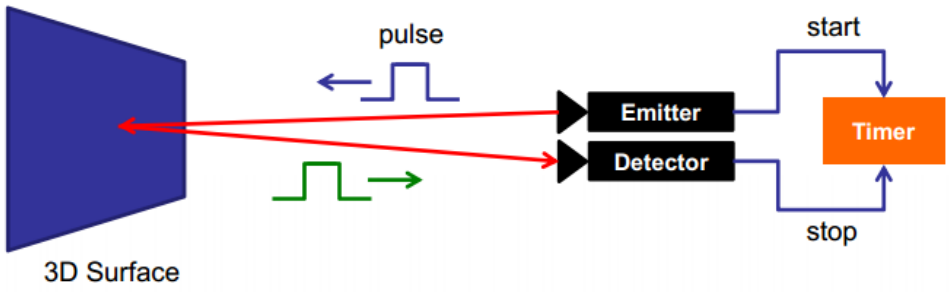

Figura 4: Esquema de funcionamiento de la técnica de tiempo de vuelo.

Esta técnica tiene como ventaja que se mide directamente el tiempo de vuelo del pulso. Sin embargo, tiene bastantes inconvenientes, se requiere una medida del tiempo de gran precisión, debido a que la luz recorre 1 milímetro en 3.3 picosegundos. Además, la medida del pulso de luz es inexacta debido al *scattering*.

B. **Diferencia de fase:** con el cálculo de la diferencia de la fase de la onda que se emite desde el sensor y la que se captura después de que esta rebote en la escena, se puede calcular la distancia de la escena al sensor. En este caso en vez de la emisión de un pulso de luz, se realiza la emisión continua de un haz de luz. Se trata de una técnica que proporciona una buena precisión en distancias medias, hasta los 200 metros, poseyendo un error de 2 milímetros por cada 25 metros.

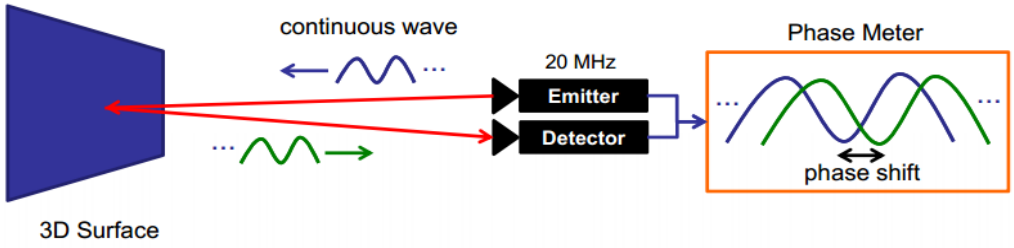

Figura 5: Esquema de funcionamiento de la técnica de diferencia de fase.

Esta técnica permite la utilización de una gran variedad de fuentes de luz debido a que no se necesitan pulsos de luz cortos y fuertes. Tiene la desventaja de que tiene un alcance más limitado que el tiempo de vuelo.

Capítulo 3

Leap e información 3d

Leap Motion [5][6] es un periférico USB diseñado para ser colocado en un escritorio mirando hacia arriba. Se trata de un sensor que está especialmente desarrollado para la captura de información de las manos. Está compuesto por dos cámaras monocromáticas IR, y tres LEDs infrarrojos (Figura 7). Este dispositivo observa aproximadamente en una semiesfera a una distancia de un metro (Figura 6). Las cámaras son capaces de capturar hasta 300 frames por segundo.

Figura 7: Hardware Leap Motion.

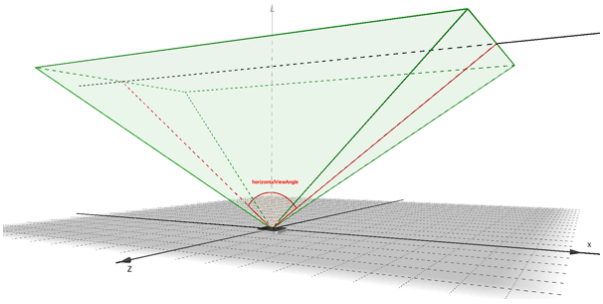

Figura 6: Cono de detección de Leap Motion.

Para poder saber la técnica que utiliza el sensor para la adquisición de información 3D, se ha capturado la información en crudo que envía al ordenador, sin la utilización de las librerías que proporciona el propio sensor. La información que se transmite a través del USB no se encuentra codificada. Para cada uno de los pixeles existen dos componentes, uno por imagen obtenida, que están formados por 8 bits. Las imágenes obtenidas son de un tamaño de 640x240 en el caso de que se trate del modo en el cual los LEDs se encuentran encendidos continuamente. Si se trata del *modo robusto*, se obtienen imágenes de 640x120.

Con la información obtenida en cada frame, se reconstruyen las dos imágenes de las cámaras. La reconstrucción no es perfecta debido a que durante la captura de la información que se transmite por el USB se puede producir la perdida de algún paquete, así que algún frame presenta errores al no poseer la información completa. En figura 8 se muestran las imágenes capturadas por las cámaras en la misma imagen, poniendo en rojo la imagen capturada por la cámara derecha y en verde la capturada por la cámara izquierda. En la parte amarilla hay superposición de las imágenes.

Figura 8: Imagen obtenidas del Leap Motion.

Se puede observar que las imágenes que se obtienen están desplazadas una respecto a la otra. Esto se debe a la posición de las cámaras. Obtenida esta información, se observa que la técnica usada para la obtención de información de profundidad de la escena es visión estereoscópica. Además gracias a la luz infrarroja emitida por los leds que posee, la imagen se obtiene en escala de grises, la cual se usa para calcular la profundidad de los puntos de interés.

3.1. Cálculo de la profundidad

Para el cálculo de la profundidad, inicialmente se tiene que realizar la calibración de cada una de las cámaras, que permite calcular los parámetros de distorsión. Los parámetros de distorsión son k_1, k_2, k_3, los cuales sirven para corregir la distorsión radial y poder así transformar las imágenes obtenidas para eliminar las distorsiones.

Para la eliminar la distorsión radial, tanto la distorsión de barril como la de cojín (en este caso las cámaras del Leap poseen distorsión de barril) se aplicarán las siguientes fórmulas.

$$x_{corregida} = x_c + (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)(x_{dis} - x_c)$$

$$y_{corregida} = y_c + (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)(y_{dis} - y)$$

Donde X_{dis}, y_{dis} son la posición del pixel en la imagen distorsionada, x_c indica la posición del centro de la cámara y $r = \sqrt{(x_{dis} - x_c)^2 + (y_{dis} - y_c)^2}$.

Una vez calibradas las cámaras se procede a buscar los diferentes puntos de interés de la imagen. Para obtener los puntos de interés se pueden utilizar diferentes descriptores como pueden ser SURF o SIFT, ambos descriptores son invariantes a escala. En este caso esta propiedad es innecesaria debido las dos imágenes van a ser tomadas a la misma distancia e instante de tiempo.

A continuación, se debe realizar el emparejamiento de cada uno de los puntos con su correspondiente en la otra imagen. Debido a la disposición de las cámaras en el Leap, el emparejamiento de los puntos de interés encontrados en la fase anterior es más sencillo que en otras situaciones. En este caso hay que recorrer todos los puntos de interés de la imagen de la derecha o de la izquierda, de tal manera que únicamente se busca el punto de interés de la primera imagen en la segunda imagen sobre la línea epipolar correspondiente (Figura 9).

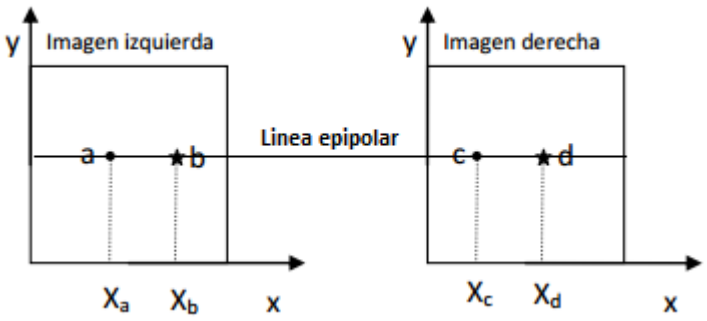

Figura 9: Búsqueda de emparejamientos de entre las imágenes obtenidas por Leap.

En este proceso y para evitar que se produzcan emparejamientos erróneos, se buscan los dos mejores puntos de la segunda imagen que pueden corresponder con el punto de la primera imagen al que se le quiere buscar correspondencia. De tal manera que cuando la distancia que tenga el primer descriptor al punto de la primera imagen no sea menor que la distancia del segundo multiplicada por 0.8, es decir que haya una diferencia como mínimo de un 20%, el emparejamiento no se tiene en cuenta.

```
void match_descriptor(Mat descriptors1, Mat descriptors2, vector< DMatch >
&matches, int tipo_matcher=3){

	Ptr<DescriptorMatcher> DescriptorMatcher;
	if (tipo_matcher == 1){
		DescriptorMatcher= DescriptorMatcher::create("FlannBased");
	} else {
		DescriptorMatcher= DescriptorMatcher::create("BruteForce");
	}

	vector<vector<DMatch>> initialMatch;
	(*DescriptorMatcher).knnMatch( descriptors1, descriptors2, initialMatch,2
);
	for(int i = 0; i< initialMatch.size(); i++){
		if(initialMatch[i].at(0).distance< 0.8 *
initialMatch[i].at(1).distance){
			matches.push_back(initialMatch[i].at(0));
		}
	}

}
```

Figura 10: Muestra del código que devuelve los emparejamientos, utilizando el método de fuerza bruta o Flann y teniendo en cuenta que la distancia la distancia al segundo vecino tiene que ser al menos un 20 % mayor.

Una vez que se han calculado estas correspondencias, sabiendo la distancia que existe entre las cámaras, se calcula la profundidad a la que se encuentra cada uno de los puntos a través de triangulación, como se ha explicado en el capítulo 2.

Una vez obtenidos los puntos de interés se puede realizar una interpolación teniendo en cuenta el color de gris que posee los pixeles que no son puntos de interés y conseguir así un mapa de profundidad completo.

3.2. Separación de manos y brazos

Para poder realizar el reconocimiento de gestos[14][15] es preciso realizar una segmentación para separar las manos y los brazos, del resto de la imagen obtenida por el sensor.

Para realizar la segmentación se pueden utilizar distintas técnicas. Inicialmente se pensó en utilizar el color de la piel[7]. Se puede pasar la imagen al modelo HSV, siglas del inglés *Hue, Saturation, Value,* que significan, tonalidad, saturación, y valor, y una vez en este modelo encontrar los valores óptimos para cada uno de los canales que permitan detectar la piel. Este modelo es el menos sensible a la luminosidad, y esta es la razón de su elección. Sin embargo, las imágenes que se obtienen de las cámaras son en escala de grises, por lo que esta técnica no puede ser usada.

Por esta razón se pensó en utilizar un método de segmentación sin utilizar el color de la piel, un método de segmentación por umbral. Como todos los métodos de segmentación, consiste en asignar cada pixel a un cierto grupo, siendo el umbral un valor que determina a qué grupo pertenece el pixel. Ante esta posibilidad surgió un problema cuando en el alcance del sensor no había ningún objeto. La segmentación

era errónea. Entonces se pensó en complementar el método de segmentación por umbral, añadiendo un rango de grises que pueden ser considerados posibles manos. A través de diferentes pruebas este umbral se ha establecido desde el valor gris 0, es decir blanco, hasta el valor de gris 34.

Para realizar la segmentación utilizando un método de umbralización, es necesario que la imagen se encuentre en escala de grises, así que se ha utilizado la imagen que se obtiene directamente del sensor.

Existe una gran cantidad de métodos que permiten calcular un umbral, pero no se obtienen buenos resultados con todos cuando se trabaja con imágenes del mundo real como es en este caso. Esto es debido a la presencia de ruido, histogramas planos o iluminación inadecuada. Ante estos inconvenientes el método Otsu[8] proporciona buenos resultados, y se ha elegido este método para realizar la segmentación. El principal problema de este método es el tiempo computacional. Esto se debe a que realiza una búsqueda exhaustiva del umbral para maximizar la varianza entre clases, y si el número de clases aumenta, es ineficiente. En este caso como solo se desea hacer la separación de dos clases, el tiempo es aceptable.

3.2.1. Método Otsu [8]

Una imagen en escala de grises está formada por N pixeles y un número de tonos de gris que se encuentran entre 1 y L. La probabilidad de ocurrencia de un nivel de gris en la imagen está dada por

$$p_i = \frac{f_i}{N}$$

Donde f_i es el número de pixeles que poseen el nivel de gris i.

En una umbralización en dos niveles, los pixeles quedan divididos en, la clase C_1 que contiene los niveles de gris desde 1 a t, y la clase C_2 que contiene del t+1 al L, siendo t el valor del umbral.

$$C_1 : \frac{p_1}{w_1(t)}, \dots, \frac{p_t}{w_1(t)}$$

$$C_2 : \frac{p_{t+1}}{w_2(t)}, \dots, \frac{p_L}{w_2(t)}$$

Donde

$$w_1 = \sum_{i=1}^{t} p_i$$

$$w_2 = \sum_{i=t+1}^{L} p_i$$

También es necesaria la media de cada una de las clases y la media de intensidad de toda la imagen, que quedan definidas,

$$\mu_1 = \sum_{i=1}^{t} \frac{i * p_i}{w_1(t)}$$

$$\mu_2 = \sum_{i=t+1}^{L} \frac{i * p_i}{w_2(t)}$$

$$w_1 * \mu_1 + w_2 * \mu_2 \ = \ \mu_T$$

Usando análisis discriminante, Otsu definió la variancia entre clases de una imagen umbralizada como

$$\sigma_B^2 = \ w_1 * \ (\mu_1 - \ \mu_T)^2 + \ w_2 * \ (\mu_2 - \ \mu_T)^2$$

Con la varianza entre clases se calcula el valor óptimo

$$t = \max_t\{\sigma_B^2(t)\} \qquad 1 \leq t \leq L$$

Una vez calculado el mejor umbral, la segmentación en dos clases se realiza de manera sencilla.

$$g(x,y) = \begin{cases} 1 \ \Leftrightarrow \ f(x,y) > T \\ 0 \ \Leftrightarrow \ f(x,y) \leq T \end{cases}$$

Con la aplicación de estas técnicas se obtiene una imagen en blanco y negro (Figura 11), quedando las manos de color negro, y el fondo de color blanco.

En la imagen se pueden encontrar elementos de tamaño reducido que no son manos. Este problema es fácil de solucionar, a partir de operadores morfológicos. Los que se mantienen en la imagen pueden ser ignorados en pasos posteriores, y en particular en la identificación de los contornos de la imagen.

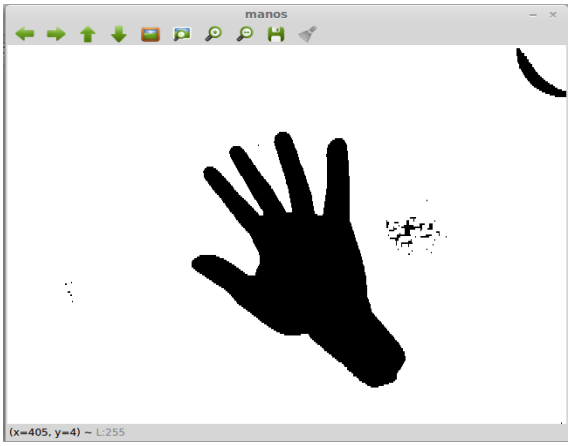

Figura 11: Segmentación de las manos.

3.3. Reconocimiento de gestos estáticos

Una vez realizada la segmentación de las manos ya se pueden utilizar diferentes técnicas para identificar los gestos. En este trabajo únicamente se ha implementado el reconocimiento de ciertos gestos estáticos, debido a que este no es el objetivo principal del proyecto.

Para poder identificar estos gestos se parte de la imagen obtenida por el sensor a la que ya se le ha realizado la segmentación, para tener únicamente los brazos y manos. El siguiente paso es la separación de cada una de las figuras que haya en la imagen, en este caso los brazos y manos, para el cual se tiene que definir una conectividad como punto de partida. Los tipos más importantes de conectividad son 4-conectividad, 8-conectividad, y 6-conectividad (Figura 12).

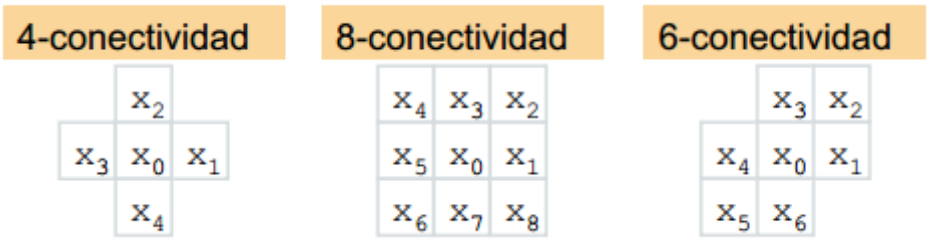

Figura 12: Muestra de pixeles que se consideran conectados con el pixel x0 para distintos tipos de conectividad.

El algoritmo que determina qué pixel pertenece a cada figura en una imagen consiste en recorrer la imagen por filas, realizando las siguientes operaciones.

Descripción formal del algoritmo:

a) **Paso 1:** Si existe 1-pixel (de color negro) a procesar.
   a. Si uno solo de sus vecinos (superior e izquierdo) 1-pixel, copiar su etiqueta.
   b. Si ambos los son y tienen la misma etiqueta, copiarla.
   c. Si ambos los son y tienen etiqueta distinta, copiar la del superior, y marcar etiquetas como equivalentes.
   d. Si ninguno de sus vecinos es 1-pixel, asignar etiqueta nueva.
b) **Paso 2:** Si existen más pixeles, volver al paso uno.

Una vez recorrida toda la imagen es necesaria una segunda pasada. Se reetiquetan los pixeles, eligiendo para cada segmento el menor de sus etiquetas, y sustituyendo las etiquetas equivalentes por la menor.

En el caso de este trabajo se ha elegido usar 4-conectividad, por lo que habrá que tenerlo en cuenta a la hora de determinar si un pixel es vecino o no (Figura 13).

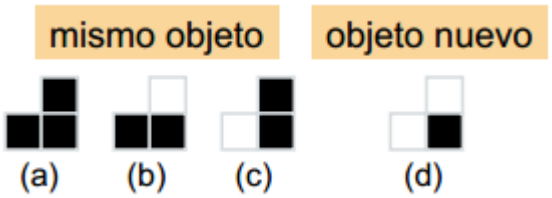

Figura 13: Muestra de los pixeles que considerados vecinos en el caso de 4-conectividad.

Una vez se tienen separadas cada una de las figuras que posee la imagen se pasa a calcular el contorno para cada uno de ellas[9]. Se comienza en el primer pixel que se ha encontrado, y se va recorriendo a los pixeles que le rodean y pertenecen a la figura, en sentido contrario a las agujas del reloj, empezando por la izquierda (Figura 14).

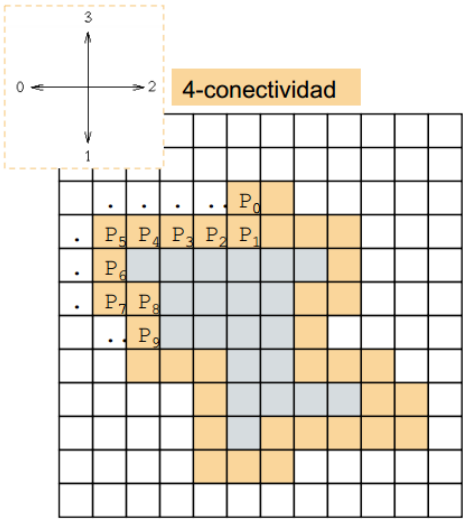

Figura 14: Ejemplo de cálculo del contorno con 4-conectividad.

La implementación de todo este proceso se ha realizado a través del uso de la librería OpenCV y con la llamada a la función **findContours**. Pasándole como parámetros la imagen en la que hay que calcular los contornos, un vector de vectores de puntos para almacenar los contornos por figura, y la constante CV_CHAIN_APPROX_SIMPLE para que utilice 4-conectividad.

Como se ha citado anteriormente se pueden encontrar figuras que no son manos. Estas figuras pueden ser ignoradas, considerando la longitud del contorno detectado. Si es muy pequeño no será utilizada en las siguientes operaciones.

Una vez que se han calculado los contornos, se procede al cálculo del envolvente convexo de dichos contornos. Encontrar el envolvente convexo de un conjunto de puntos es un problema de geometría computacional. Para calcularlo existen varios métodos, la librería de OpenCV utiliza el algoritmo de Sklansky[10]. Este algoritmo es conocido como el de las tres monedas.

Descripción formal del algoritmo

a) **Paso 1:** Encontrar un vértice convexo y etiquetarlo como l_0.
b) **Paso 2:** Etiquetar el resto de los vértices en sentido de la agujas del reloj, comenzando por l_1.
c) **Paso 3:** Poner monedas en los vértices l_0, l_1, l_2, y marcarlos como última, centro, primera.
d) **Paso 4:** Hacer
    a. **Si** las tres monedas tienen un giro a la derecha, la última pasa al siguiente vértice de la moneda primera, y la última pasa a ser primera, la centro, última, y la primera, centro.
    b. **Sino** la moneda centro pasa a la anterior de la última, se borra el vértice que ocupaba última, última pasa a ser centro y centro a última.

    Hasta que primera sea l_0.

Los vértices y aristas resultantes del algoritmo forman un polígono convexo que incluye todos los vértices que se pasan inicialmente.

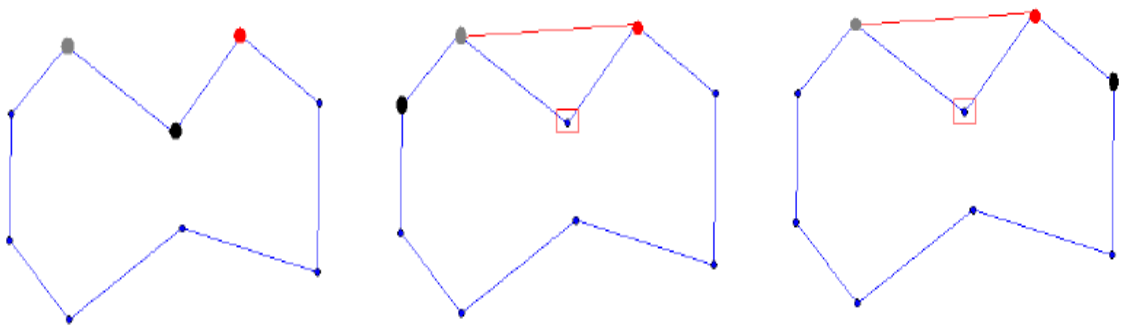

Figura 15: Demostración del funcionamiento del algoritmo de las tres monedas cuando una de las tres monedas no tiene un giro a derecha.

A través de la librería de OpenCV[11][12] se consigue el envolvente convexo de un conjunto de puntos llamando a la función **convexHull.**

Figura 16: Envolvente convexo de las figuras con un contorno superior a un mínimo establecido.

A partir del envolvente convexo y el contorno de la figura (Figura 16), para poder identificar cada uno de los dedos, se calculan los defectos de convexidad, es decir aquellos puntos en los cuales haya una distancia significativa entre el envolvente convexo calculado y el contorno de la figura. Cada uno de estos defectos de convexidad está definido por tres puntos, el comienzo del defecto, el punto más alejado del envolvente convexo calculado, y el punto final. Un ejemplo de esto se puede observar en la figura 17.

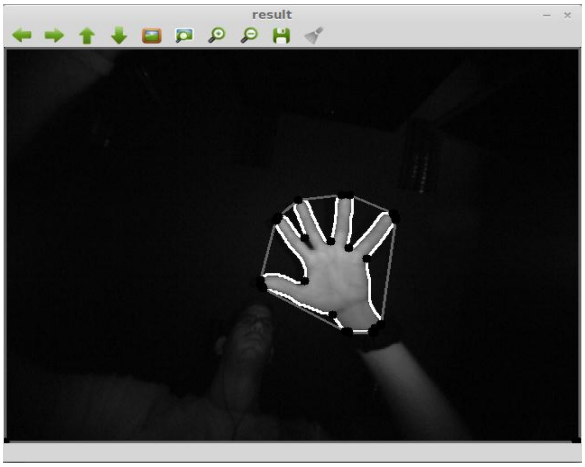

Figura 17: Defectos de convexidad, todos los puntos mostrados.

Se puede observar en la figura 17 que hay una gran cantidad de puntos. De todos estos, para detectar cual es el número de dedos que posee la mano, únicamente se necesitan los puntos que marcan el punto central del defecto, los puntos de comienzo y fin de cada uno de los defectos se pueden ignorar. El resultado se ve en la figura 18.

Figura 18: Muestra de los defectos de conexidad punto central.

Aun así se puede ver que siguen sobrando puntos. La solución a esto es realizar un filtrado. El filtro consiste en establecer una distancia mínima entre el envolvente convexo y el contorno de la mano, y un ángulo máximo entre los puntos de comienzo del defecto y finalización (Figura 19). En concreto esta limitación es 80^{o}.

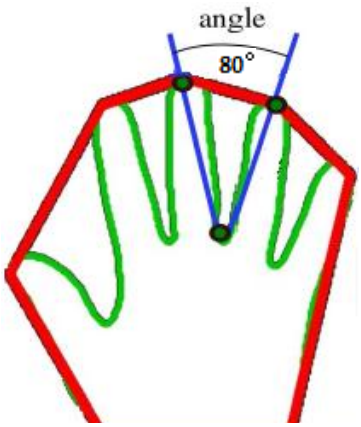

Figura 19: Muestra el Angulo existente entre los puntos que marcan un defecto de convexidad.

Con los defectos de convexidad se reconoce el número de dedos de una mano siempre y cuando no sean uno o cero, en cuyo caso, no existe defecto de convexidad que cumpla el filtro anterior.

Figura 20: Detección correcta del número de dedos a través de los defectos de convexidad.

En la figura 20 se ve el funcionamiento en el caso de reconocer una mano con más de un dedo. Hay que tener en cuenta que el número de dedos reconocidos es igual al número de defectos de convexidad detectados más uno.

Figura 21: Imposibilidad de diferenciar uno y cero dedos con los defectos de convexidad.

En la figura 21 se muestra el problema existente al intentar reconocer el cero o un dedo a través de la técnica de defectos de convexidad. El número de defectos de convexidad es cero, por lo que no es posible utilizar este método para estos casos.

Para poder realizar el reconocimiento de estos gestos, se ha utilizado otro método, que consiste inicialmente en el cálculo del círculo de la palma, que en principio es el círculo más grande del brazo.

Definición formal del algoritmo

a) **Paso 1:** Calculo del rectángulo que envuelve el contorno detectado.

b) **Paso 2:** Este contorno se divide en dos por el lado más largo, y desde cada punto perteneciente al contorno se calcula la distancia mínima al rectángulo que lo contiene.

c) **Paso 3:** Se busca el punto más alejado del contorno en cada una de las mitades del rectángulo, y este es el centro de la palma.

d) **Paso 4:** De los dos puntos obtenidos será el centro del círculo de la palma aquel que se encuentre a una mayor distancia, siendo la distancia su radio.

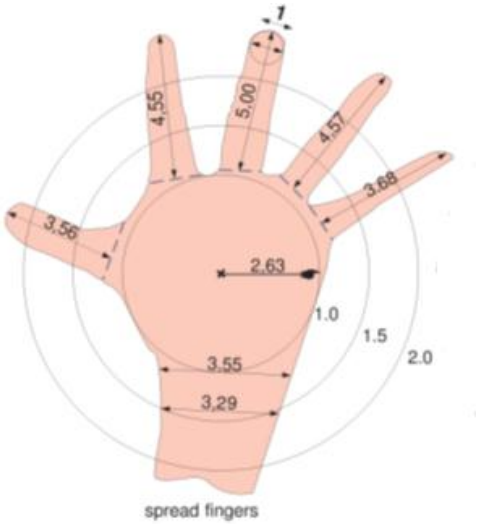

Figura 22: Anatomía de la mano humana.

Una vez que se ha calculado el centro de la palma, y el radio de esta, debido a la anatomía de la mano (Figura 22), la distancia máxima que puede haber entre el centro de la palma y el final del dedo es 2.87 veces el radio de la palma. Así que para tener más seguridad se pondrá como límite 3 veces el radio de la palma. En la figura 23 se muestran las circunferencias de la palma la situada a 2 veces el radio, y la situada a 3.

Figura 23: Se observa como no sobresale ningún dedo fuera de la esfera, en caso contrario se puede ver que el dedo sobresale de la esfera, entonces es posible diferenciar estos dos.

Con esta información ya es posible realizar la diferenciación entre cero y un dedo. Así que en el caso de que el número de defectos de convexidad sea igual a cero, se buscará si existe algún vértice del envolvente convexo fuera y a una distancia en 2 y 3 veces el radio de la palma. En ese caso quiere decir que existe un dedo extendido. En el caso de que no haya ningún vértice en ese rango de distancias, significará que no hay dedo extendido.

Capítulo 4

Reconocimiento de gestos usando Leap

Una vez identificada la técnica de visión que utiliza Leap Motion para obtener la información tridimensional, se van a analizar los gestos que pueden llegarse a detectar con las librerías del propio sensor[13]. Leap proporciona una gran cantidad de datos que deben ser analizados para interpretarlos como un gesto determinado.

La cantidad de gestos que se dan como resultado de la combinación de todos los datos obtenidos es muy elevada. La *precision* de cada uno de los datos capturados por el Leap depende del modo de funcionamiento. Leap tiene varios modos de funcionamiento, que hacen que el número de frames capturados por el sensor varíe. En concreto posee cuatro modos de funcionamiento, el *modo alta velocidad*, cuya prioridad es obtener el mayor número de frames posibles; el *modo precision*, que sacrifica el número de frames a cambio de una gran precisión; el *modo equilibrado*, que es un punto intermedio entre el *modo alta velocidad* y el *modo precisión*. Por último el *modo robusto*, se activa automáticamente cuando las condiciones de luminosidad son malas.

4.1. Información capturada

Como se ha citado anteriormente Leap es un sensor capaz de capturar gran información de las manos de tal manera que con su análisis se pueden definir un vocabulario de gestos. La información devuelta es de cada una de las manos detectadas, así como de cada uno de los objetos *pointables* detectados (objetos orientables que tiene una longitud elevada y una pequeña anchura).

De cada una de las manos devuelve una lista con los dedos que ha detectado y que pertenecen a esa mano, así como otra lista de objetos *pointables,* en la cual se incluyen los dedos. Además se obtiene la posición y la normal de esta en cada uno de los ejes de coordenadas, la velocidad de su desplazamiento, el ángulo de rotación, y el cambio de posición entre dos instantes de tiempo.

También se captura la información sobre la curvatura de la mano, y el centro de la esfera que forma, así como su radio (Figuras 24 y 25).

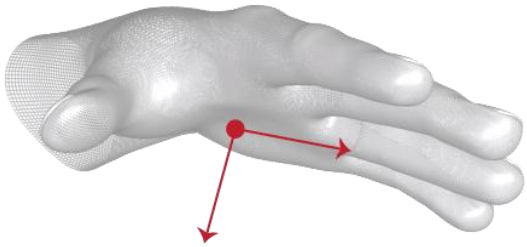

Figura 24: Detección de la normal de la mano.

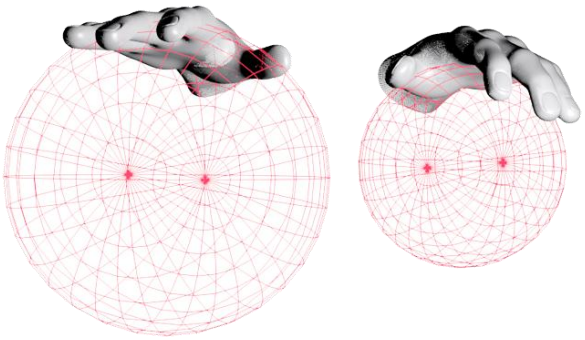

Figura 25: Detección de la curvatura de la esfera formada por la mano.

De cada uno de los dedos y por lo tanto para el resto de herramientas detectadas, se da también una gran cantidad de información. Se captura para cada uno de estos objetos, su dirección, su longitud (Figura 26) y su velocidad de desplazamiento.

Figura 26: Detección de la dirección de cada uno de los dedos de la mano.

4.2. Gestos estáticos

Con toda la información que se ha citado anteriormente, se pueden definir un conjunto de gestos estáticos, realizando combinaciones de estos.

- **Palma abierta:** El número de manos detectadas es una, que posee una normal en el eje "y" con valor -1, y el número de dedos detectados es igual a 5
- **Palma vertical:** El número de manos detectadas es una, y posee una normal en el eje "z" con un valor de -1 ó 1, el número de dedos detectados es igual a 0.
- **Puño:** El número de manos detectadas es una, la cual posee una normal en el eje "y" con un valor de -1, y el número de dedos detectados es igual a 0.
- **Dos palmas:** El número de manos detectadas es igual a 2, y el número de dedos que deben poseer cada mano es de 5.

Además de estos gestos definidos, existen otras combinaciones posibles. Las explicadas anteriormente son únicamente un ejemplo. Si se añaden más manos y se tienen en cuenta todos los parámetros, la cantidad de gestos que se pueden definir es bastante elevada.

4.3. Gestos dinámicos

El Leap es capaz de identificar varios gestos dinámicos, a partir de la información que obtiene, es decir, basándose en la posición de cada uno de los elementos que detecta, ya sean los dedos, herramientas, o simplemente la mano, en cada uno de los frames que captura.

- **Gesto círculo:** Detecta la realización de un círculo en el aire a través de un dedo o con una herramienta (Figura 27), es posible detectar el círculo independientemente de la dirección de su normal. De esta manera es posible diferenciar entre los círculos horizontales y verticales, así como dentro de un mismo tipo de círculo, la dirección del giro, debido a que se produce un cambio de la dirección de la normal.

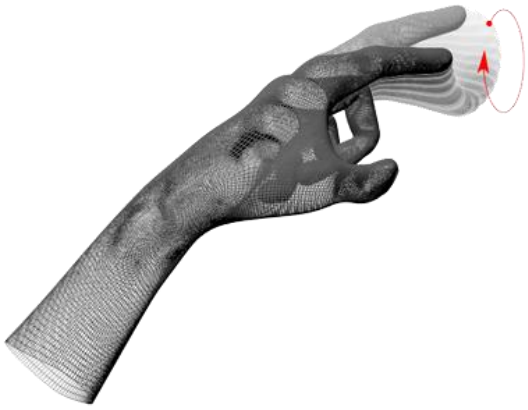

Figura 27: Detección de la realización de un círculo.

Una vez que el gesto círculo se detecta por el Leap se obtiene información de éste, como puede ser el centro del círculo, su normal, la duración del gesto, el progreso del círculo. Este valor es siempre positivo indicando el número de vueltas; si es 0.5, se ha dado media vuelta, si es 3 se han dado tres vueltas

- **KeyTap:** Detecta la realización de un clic en el aire (Figura 28), es decir hacer bajar el dedo o herramienta hasta una posición inferior y volver a la posición inicial, tardando menos de un tiempo determinado, sin una variación de la posición de la mano

Figura 28: Detección de la realización de un keyTap.

Para que este gesto sea detectado, se tiene que tardar menos 0.1 segundo en realizarlo, con velocidad mínima de 50mm/s, y que se produzca un desplazamiento mínimo de 3 mm. Al igual que ocurre con el círculo, se obtiene más información, como la duración del gesto y la dirección del keyTap.

- **ScreenTap:** Detecta la pulsación de un botón en el aire, es decir el atravesar un plano vertical imaginario (Figura 29). El dedo o herramienta comienza en una posición y se desplaza hacia delante con una velocidad determinada, y a continuación vuelve a la posición inicial.

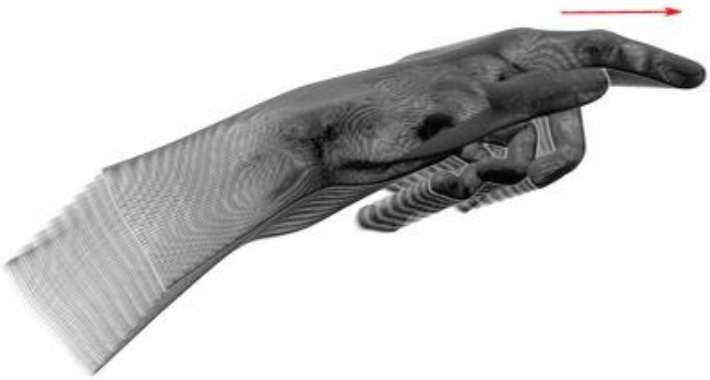

Figura 29: Detección de la realización de un ScreenTap.

Para que sea detectado se tiene que tardar en realizarlo menos de 0,1 segundos, a una velocidad mínima de 50mm/s, y que se produzca un desplazamiento mínimo de 5 mm. La información que se obtiene de este gesto es idéntica a la que se obtiene del keyTap.

- **Swipe:** Detecta la realización de un barrido en el aire con la mano o herramienta (Figura 30), es decir el desplazamiento desde una posición inicial a una posición final con una velocidad determinada.

Figura 30: Detección de la realización de un barrido.

Capítulo 5

Análisis de gestos

Siguiendo con los objetivos de este trabajo, se va a realizar un estudio de los datos capturados por Leap mediante una aplicación que permite obtener información de cada gesto. Posteriormente con esta información se obtendrán los parámetros que resulten más significativos para identificar cada uno de los gestos, así como para cada uno de los parámetros su rango óptimo para la identificación.

La aplicación ofrece una interface en el que se indican los gestos sucesivamente, de los cuales se obtiene toda la informacion. Cada uno de los gestos se realiza durante un periodo de 5 segundos, durante el cual se capturan los datos del sensor.

5.1. Introducción

Inicialmente la aplicación indica que se introduzca el nombre del usuario que la va a utilizar (Figura 32), y presione el boton “Start”. A continuación se le indicará el gesto que tiene que realizar (Figura 31).

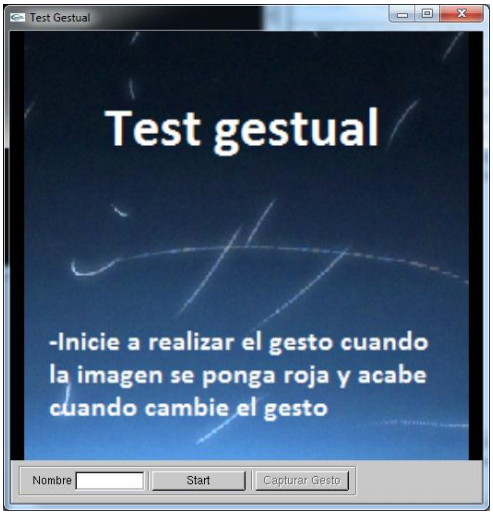

Figura 32: Muestra del gesto a realizar.

Figura 31: Pantalla inicial de la aplicación de análisis.

Figura 34: Pantalla que indica que se debe comenzar la realización del gesto.

Figura 33: Pantalla que indica que se está capturando información.

Cuando se pulsa en el boton capturar, la pantalla se pone de color rojo (Figura 33), y el ususario tiene que comenzar a realizar el gesto. Pasados tres segundos la pantalla cambia de color, a verde (Figura 34), y se empieza a guardar información.

Con la ejecución de esta aplicación y a través del análisis de los datos obtenidos se identifican cuáles son los parámetros más significativos para realizar la identificación. Se trata del número de manos y dedos detectados, la posición de las manos en el eje "x", junto con la normal en el eje "x" e "y" de estas, el número de keyTaps, así como la normal de los círculos detectados en el eje "z", e "y".

Una vez identificados cuales son los parámetros más significativos se ejecuta de nuevo la aplicación para detectar el mejor rango, que permita la identificación de cada gesto. Esto se realiza a partir de la *precision* y del *recall*, intentando maximizar los dos valores, dando una ligera prioridad a la *precision*, para que no se produzcan situaciones erróneas.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Donde TP, verdaderos positivos, FP, falsos positivos, y FN, falsos negativos.

5.2. Gestos de interés y modos de funcionamiento

Leap Motion tiene tres modos de funcionamiento principales, *modo precision*, en el que se da más importancia a la precisión de las medidas que al número de frames que son capturados; *modo alta velocidad*, con el cual se quiere obtener el mayor número

de frames por segundo, sacrificando así un poco la precisión del sensor; y por último el *modo equilibrado* que es un término medio entre ambos.

Se ha realizado una comparación entre los tres modos de funcionamiento para cada uno de los gestos, con la aplicación anteriormente explicada, una vez ya detectados los parámetros más identificativos y sus mejores rangos. Estos datos han sido obtenidos también de personas que han sido previamente entrenadas para realizar los gestos y que los resultados sean más fiables.

5.2.1. Palma abierta

Figura 35: Gesto palma abierta.

	Precision	Recall
Modo presicion	0.905297	0.98947368
Modo equilibrado	1	1
Modo alta velocidad	0.87045362	1

Tabla 1: Datos palma abierta.

Se puede observar en la tabla 1 que el mejor modo es el *modo equilibrado*, cuya *precision* y *recall* es igual a uno, es decir en dicho modo de funcionamiento no se producen ni falsos positivos ni falsos negativos. En los otros dos modos esto sí que ocurre. En el *modo alta velocidad* se puede ver un *recall* mayor. Esto quiere decir que no posee falsos negativos, por lo que no hay ningún gesto de palma abierta que no sea identificado. Sin embargo, tiene la *precision* más baja que el *modo precision*. Así que el número de falsos positivos es mayor en el *modo alta velocidad* y por tanto entre estos dos modos el mejor es el *modo precision*.

5.2.2. Dos palmas

Figura 36: Gestos dos palmas.

	Precision	Recall
Modo precision	1	0.68070175
Modo equilibrado	1	1
Modo alta velocidad	1	0.54084507

Tabla 2: Datos 2 palmas.

En este gesto el mejor sigue siendo el *modo equilibrado*, que también tiene una *precision* y *recall* de uno. En este gesto todos los modos poseen una *precision* de uno y esto quiere decir que ninguno de ellos posee falsos positivos. Quitando el *modo equilibrado* los otros modos poseen un bajo *recall*, así que muchos de los gestos de dos palmas no son reconocidos como tal.

5.2.3. Puño

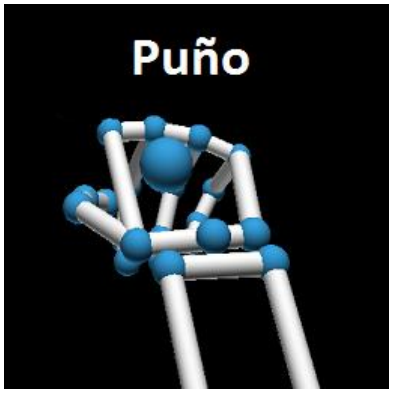

Figura 37: Gesto puño.

	Precision	*Recall*
Modo precision	0.261588	1
Modo equilibrado	0.2457956	1
Modo alta velocidad	0.35720275	1

Tabla 3: Datos puño.

Se puede ver que todos los gestos puño que son realizados se reconocen como tal. Pero tiene una gran cantidad de falsos positivos, es decir que hay muchas veces que se está realizando un gesto que no es el puño y sí que es reconocido como tal. Con los diferentes datos obtenidos en el análisis se observa que los falsos positivos corresponden a las veces que se realiza el gesto del keytap. Esto se debe a que los parámetros que se utilizan para identificarlos poseen un gran grado de solape. Una posible solución a este problema sería cambiar el rango del número de dedos para la identificación del puño, poniéndolo a cero, pero entonces el *recall* bajaría de forma notable.

5.2.4. Círculo vertical derecha

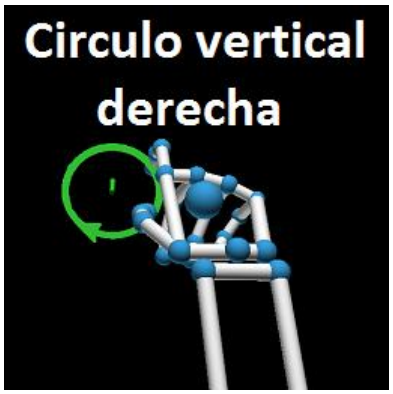

Figura 38: Gesto círculo vertical derecha.

	Precision	Recall
Modo precision	1	0.98245614
Modo equilibrado	1	0.95624752
Modo alta velocidad	1	0.717840376

Tabla 4: Datos círculo vertical derecha.

Para este gesto el mejor caso es el *modo precision*, debido a que todos tienen la *precision* a 1. El parámetro diferenciador es el *recall*, que es superior para este modo.

5.2.5. Círculo vertical izquierda

Figura 39: Gesto círculo vertical izquierda.

	Precision	Recall
Modo precision	1	0.67017544
Modo equilibrado	1	0.9608696
Modo alta velocidad	1	0.349765258

Tabla 5: Datos círculo vertical izquierda.

El mejor modo es el *modo equilibrado*. Al igual que ocurría con el *modo precision*, en el gesto anteriormente citado, ninguno de los modos posee falsos positivos y el *modo equilibrado* es el que menos falsos negativos posee.

Comparando estos dos gestos, se puede observar que el *modo alta velocidad* no es un buen modo para la detección de gestos dinámicos, debido a la cantidad falsos positivos.

5.2.6. Palma vertical

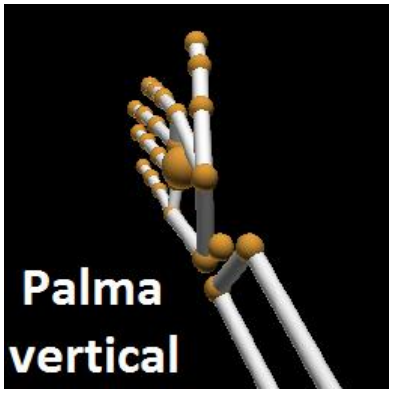

Figura 40: Gestos palma vertical.

	Precision	Recall
Modo precision	0.666667	1
Modo equilibrado	1	1
Modo alta velocidad	1	1

Tabla 6: Datos palma vertical.

En este gesto es tan bueno el *modo equilibrado* como el *modo alta velocidad*. Ninguno de estos modos posee falsos positivos ni falsos negativos. El *modo precision* queda descartado debido a que posee una gran cantidad de falsos positivos.

5.2.7. Círculo horizontal derecho

Figura 41: Gesto círculo horizontal derecha.

	Precision	Recall
Modo precision	1	0.48947368
Modo equilibrado	1	0.624561404
Modo alta velocidad	1	0.30046948

Tabla 7: Datos círculo horizontal derecha.

El mejor modo es el *modo equilibrado*. Ninguno de ellos posee falsos positivos, pero el *modo equilibrado* posee menos falsos negativos que el resto de los modos. Si se realiza una comparación entre los datos obtenidos de los círculos verticales y horizontales se observa que la *precision* se mantiene. Todos los gestos que son clasificados como círculos los son. Pero los círculos horizontales poseen un menor *recall*. Esto se debe a la dificultad que posee el sensor para reconocer este gesto en comparación con los círculos verticales.

5.2.8. Círculo horizontal izquierda

Figura 42: Gesto círculo horizontal izquierda.

	Precision	Recall
Modo precision	1	0.69824561
Modo equilibrado	1	0.75175439
Modo alta velocidad	1	0.38685446

Tabla 9: Datos círculo horizontal izquierda.

En este gesto ocurre exactamente lo mismo que en el gesto anterior. El mejor modo es el *modo equilibrado* debido a que posee una *precision* como el resto, pero su *recall* es mayor.

5.2.9. KeyTap derecha

Figura 43: Gesto keyTap derecha.

	Precision	Recall
Modo precision	0.928571	1
Modo equilibrado	0.98	1
Modo alta velocidad	0.87096774	1

Tabla 10: Datos keyTap derecha.

El mejor modo para este gesto es el *modo equilibrado* debido a que posee el *recall* igual a uno como el resto de los modos, todos los keyTap derechos que se realizan son reconocidos como tales, pero posee una mayor *precision* es decir, menos falsos positivos. En este gesto no se produce la confusión con el puño debido a que posee un parámetro más que lo define, el número de keyTap detectados.

5.2.10. KeyTap izquierda

Figura 44: Gesto de keyTap izquierda.

	Precision	Recall
Modo precision	1	1
Modo equilibrado	1	1
Modo alta velocidad	1	1

Tabla 11: Datos keyTap izquierda.

En este gesto es igual el modo de funcionamiento que se elige del Leap, debido a que en todos los modos no se producen falsos positivos ni falsos negativos.

Con todos los datos analizados en la parte anterior se llega a la conclusión que el mejor de los modos es el *modo equilibrado* debido a que es el modo que posee una mayor *precision* y *recall* para la mayoría de los gestos.

Además como se ha citado anteriormente, el *modo alta velocidad* es el que peor resultados proporciona para los gestos dinámicos. Sin embargo los gestos estáticos se detectan bien este modo.

5.3. Sensibilidad al cambio de condiciones

Leap Motion, además de los tres modos de funcionamiento que se han citado anteriormente, posee un cuarto modo, el *modo robusto*. Entra en funcionamiento automáticamente cuando las condiciones de luminosidad son malas, ya sea debido a la poca luz existente o bien porque hay una fuente demasiado luminosa incidiendo sobre el sensor.

A continuación se ha realizado un estudio de la sensibilidad del Leap Motion tanto en *modo robusto* como en *modo equilibrado* cuando se produce una variación de la altura, para comprobar las ventajas que proporciona dicho modo. Al mismo tempo es interesante identificar cual es la mejor altura para realizar los gestos. Para ello, se han obtenido datos con la aplicación anterior, realizando los gestos a una altura superior a 250 mm del sensor, y a una distancia inferior a esa distancia.

Los resultados que se han obtenido cuando los gestos se realizan desde una altura baja (inferior a 250 mm) y alta (superior a 250) en *modo robusto*, se ven en las tablas 12 y 13.

Altura < 250mm	Precision	Recall
Palma abierta	1	1
Dos palmas	1	1
Puño	0.06346899	0.45964912
Círculo vertical derecha	1	0.84444444
Círculo vertical izquierda	1	0.83684211
Palma vertical	1	1
Círculo horizontal derecha	1	0.82368421
Círculo horizontal izquierda	1	0.73333333
KeyTap derecha	1	0.95348837
KeyTap izquierda	1	1

Tabla 12: Datos *modo robusto* a una altura inferior a 250 mm.

Altura > 250 mm	Precision	Recall
Palma abierta	1	0.36842105
Dos palmas	1	0.23508772
Puño	0.14540816	1
Círculo vertical derecha	1	0.36491228
Círculo vertical izquierda	0.98666667	0.25964912
Palma vertical	1	1
Círculo horizontal derecha	1	0.04561404
Círculo horizontal izquierda	1	0.02105263
KeyTap derecha	1	0.8
KeyTap izquierda	1	1

Tabla 13: Datos *modo robusto* a una altura superior a 250 mm.

Si se realiza una comparación entre la tabla 12 y 13 se observa como la *precision* entre las dos alturas es similar, es decir no se producen apenas falsos positivos. Sin embargo, considerando el *recall* se ve que se produce una gran mejoría cuando los gestos se realizan a una altura inferior.

Realizando la misma prueba con el *modo equilibrado*. Las tablas 14 y 15 muestran los valores obtenidos realizando los gestos a una altura baja, y alta.

Altura < 250mm	Precision	Recall
Palma abierta	1	1
Dos palmas	1	1
Puño	0.22646007	1
Círculo vertical derecha	1	1
Círculo vertical izquierda	1	1
Palma vertical	1	1
Círculo horizontal derecha	1	0.99473684
Círculo horizontal izquierda	1	0.88947368
KeyTap derecha	0.95833333	0.95833333
KeyTap izquierda	1	1

Tabla 14: Datos *modo equilibrado* a una altura inferior a 250 mm.

Altura > 250 mm	Precision	Recall
Palma abierta	1	0.99473684
Dos palmas	0	0
Puño	0.17896389	1
Círculo vertical derecha	1	0.4754386
Círculo vertical izquierda	1	0.16538462
Palma vertical	0	0
Círculo horizontal derecha	0	0
Círculo horizontal izquierda	0	0
KeyTap derecha	1	0.94736842
KeyTap izquierda	1	1

Tabla 15: Datos *modo equilibrado* a una altura superior a 250 mm.

Ocurre lo mismo que sucede en el *modo robusto*, pero si se realiza la comparación de ambos modos de funcionamiento se ve claramente que el *modo equilibrado* es más sensible a la altura que el *modo robusto*, obteniéndose tanto una peor *precision* como un peor *recall*.

De las tablas anteriores también se puede deducir que los gestos dinámicos son más sensibles a la altura de realización, debido a que el descenso de la cantidad de gestos que son reconocidos como tal es menor que en el caso de los gestos estáticos. Si se compara cuál de los modos de funcionamiento es mejor se observa que el *modo robusto* identifica mejor los gestos dinámicos a una altura superior, pero en el caso de que la altura sea inferior, el *modo equilibrado* es mejor.

5.4. Particularidades

Con todos los datos que se han obtenido para realizar los análisis anteriores se observa que la realización de círculos horizontales es crítica. Siendo este el gesto con el que peor *recall* se obtiene, por lo que se ha decidido realizar un estudio más en profundidad de este gesto y tratar de encontrar la mejor manera para realizarlo.

Se han pensado diferentes maneras de realizar el círculo horizontal (Figura 45), para intentar mejorar los datos obtenidos, incrementando tanto la *precision* como el *recall*.

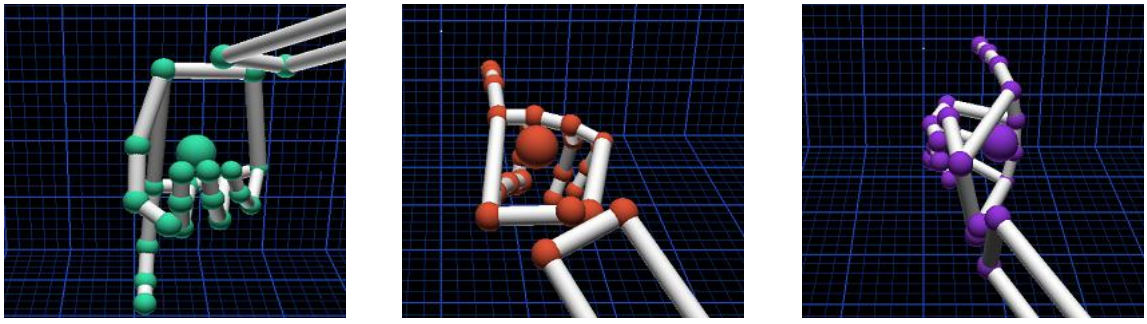

Figura 45: Distintas maneras de realizar el círculo horizontal, a) manera 1, b) manera 2, c) manera 3.

Los resultados que se obtienen de este estudio son los siguientes

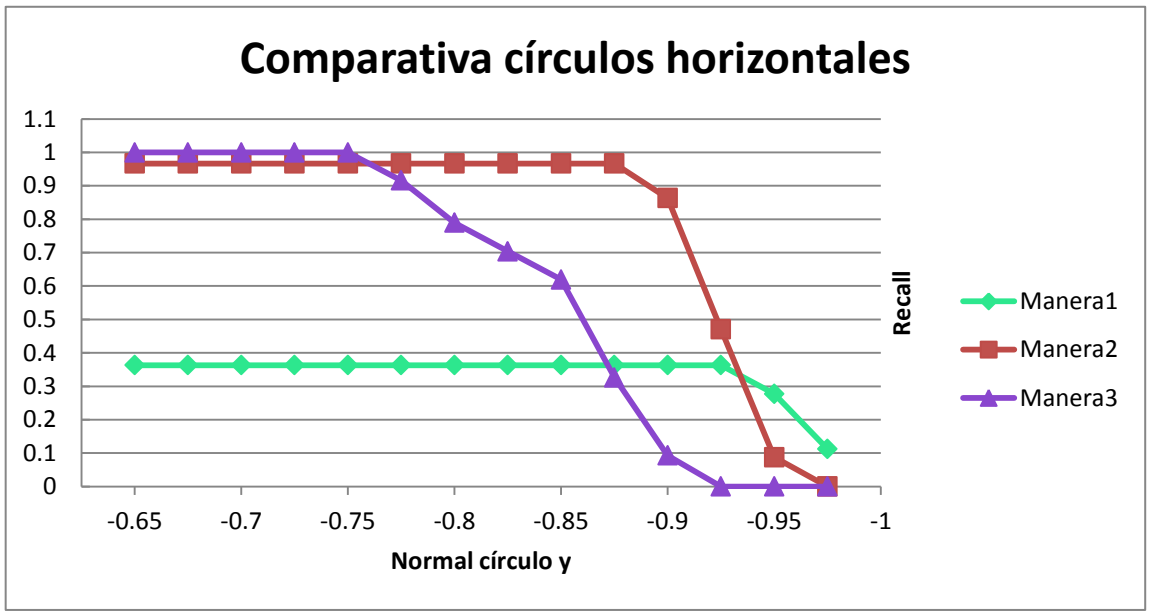

Figura 46: Comparativa de las diferentes maneras de realizar un círculo horizontal.

Se puede ver en la figura 46 que el *recall* es algo mayor si se realiza el gesto de la forma número 3. Sin embargo, conforme se aumenta el valor de la normal este valor disminuye muy rápidamente, es decir hay más gestos realizados que no se identifican como círculos horizontales. En el segundo caso, se mantiene durante más rato con un *recall* superior, pero al final también acaba cayendo. Por último la primera forma tiene un *recall* muy malo pero lo mantiene casi hasta que la normal en el eje "y" del círculo trazado es -1. Esta prueba ha sido realizada tanto en *modo equilibrado* como en *modo robusto* y el resultado es muy similar.

Capítulo 6

Aplicación desarrollada

Se ha utilizado el sensor Leap Motion, que ha sido analizado en los capítulos anteriores, para desarrollar una aplicación que sirve para el control de una cocina de inducción. Se trata de una aplicación de ámbito domótico que puede solucionar problemas que existen con los dispositivos utilizados actualmente para dicho control. Uno de los problemas con los actuales *touchpad* capacitivos es que cuando se derrama algún líquido sobre la superficie, esta no responde. Ocurre lo mismo cuando se está cocinando y se llevan las manos mojadas, el dispositivo actual no responde. A través de la aplicación desarrollada se consigue controlar la placa incluso en esas condiciones.

La aplicación se basa en la definición de un vocabulario de gestos, donde cada gesto se utiliza para la ejecución de una acción en la placa. Se ha elegido el sensor Leap Motion debido a que se trata de un sensor especializado para el reconocimiento de gestos realizados con las manos. Este sensor permite detectar tanto gestos estáticos, reconociendo el número de dedos que se muestran, el número de manos, posición de cada uno de los dedos, como gestos dinámicos, barridos ya sea horizontales o verticales, movimiento de apertura de la mano, círculos, y clic en el aire.

Se trata de una aplicación que puede funcionar aisladamente, es decir sin la ayuda de ningún otro dispositivo, controlar la placa únicamente a través de dicha aplicación, o también puede trabajar junto al actual sistema de interacción con la placa, mejorando problemas de usabilidad.

La principal característica que debe tener esta aplicación es la robustez y esto se debe a que el entorno en el que se va a ejecutar posee un cierto peligro. No se deben producir falsos positivos, ya que no se acepta que se dé por bueno un gesto cuando este no se ha realizado. Esto puede llevar a una situación peligrosa o puede hacer que sea imposible cocinar de manera adecuada. Es preferible que no se sean detectados ciertos gestos cuando estos sean realizados, a que se detecten cuando no han sido ejecutados. Hay que llegar, por tanto, a un compromiso entre robustez y usabilidad ya

que no es adecuado que buscando que no se produzcan falsos reconocimientos se tenga que realizar un gesto muchas veces para que este sea detectado como tal.

6.1. Gestos elegidos para la aplicación

De todos los gestos que se pueden detectar con el sensor Leap, se ha elegido un limitado grupo con los cuales se puedan realizar todas las operaciones a realizar sobre una placa de cocina. Las operaciones sobre la placa para las cuales se han definido gestos son los siguientes

- Encender y apagar la placa
- Seleccionar un fuego para variar su potencia
- Deseleccionar fuego
- Subir y bajar la potencia de un fuego

Inicialmente para cada una de estas acciones se pensaron varios gestos. Los gestos que se aplican a este entorno tienen que ser fáciles de realizar, para que la interacción con la placa sea fácil y a la vez tienen que tener una gran robustez, para que no se produzcan falsos positivos. Los gestos elegidos se muestran a continuación

- Una palma
- Dos palmas
- Círculo vertical
    - Hacia la derecha
    - Hacia la izquierda
- Círculo horizontal
    - Hacia la derecha
    - Hacia la izquierda
- Palma vertical
- Puño
- KeyTap
    - En el lado derecho del sensor
    - En el lado izquierdo del sensor

Para cada uno de estos gestos se deben establecer los parámetros oportunos para que sean reconocidos como tal. Se ha de tener en cuenta que puede darse el caso de que el sensor detecte algún elemento que sea considerado como dedo u otra mano. O bien que no se detecte algún dedo o mano cuando en realidad sí que se encuentran dentro del alcance del sensor. Esto puede suceder debido a que se produzca oclusiones. Estos son los errores de percepción que puede tener el Leap pero también se debe tener en cuenta que la persona que va a interactuar con el sensor no lo va a hacer de manera perfecta. Si tiene que hacer el gesto de palma vertical, lo ideal sería que la normal de la

mano en el eje “x”, fuera 1 o -1. Sin embargo, el usuario nunca la va a situar de manera completamente vertical, sino lo hará en un rango de valores considerados admisibles en los que este gesto tendrá que ser detectado como tal.

6.2. Diferentes configuraciones elegidas

Con los datos obtenidos en la fase de análisis de los gestos se pueden establecer distintas configuraciones de gestos para realizar todas las acciones de control de la placa. Estas configuraciones tienen que tener en cuenta los datos anteriores. Por ejemplo no se puede utilizar el gesto del puño y el clic, porque estos se confundirían. Algunas posibles configuraciones serían las siguientes

	Configuración 1	Configuración 2	Configuración 3
Encender	Palma abierta	Dos palmas abiertas	Dos palmas abiertas
Apagar	Palma abierta	Dos palmas abiertas	Dos palmas abiertas
Selección fuego	---	Palma abierta sobre fuego	Palma abierta sobre fuego
Deseleccionar fuego	---	Palma abierta sobre fuego	Palma abierta sobre fuego
Subir potencia	Círculo vertical derecha	Scroll hacia arriba con el puño	Círculo vertical derecha || clic derecho
Bajar potencia	Círculo vertical izquierda	Scroll hacia abajo con el puño	Círculo vertical izquierda || clic izquierdo
Cambiar de fuego	Círculo horizontal	---	Círculo horizontal

Tabla 16: Distintas configuraciones para la aplicación.

En la primera configuración se ha utilizado el gesto de palma abierta, que es muy robusto para encender y apagar la placa. Este gesto tiene que ser realizado durante un tiempo determinado para que se dé por válida la acción. Se ha elegido este gesto debido a que cuando se está cocinando no se suele estar más de 3 segundos con la mano abierta encima de la placa. Cuando se produce el encendido, automáticamente se selecciona el fuego situado en la parte superior izquierda de la placa. Sobre este fuego se van a realizar todas las acciones a ejecutar.

El principal problema en esta configuración es que no hay posibilidad de deseleccionar un fuego, por lo que si se realiza sin querer un gesto sobre la placa, y este es identificado, desemboca en una acción que no se quería realizar, pudiendo ser peligroso. El resto de los gestos utilizados tienen una gran robustez, y no producen falsos positivos. Además como ocurre con el gesto de la palma abierta, estos no se suelen realizar mientras se está cocinando.

En la segunda configuración se ha solucionado el problema que existe en la primera, debido a que hay un gesto para seleccionar el fuego sobre el que se desea realizar una acción. Ahora al encender la placa ya no es seleccionado un fuego automáticamente. El gesto que es el elegido en la configuración anterior para el encendido de la placa, se utiliza en esta configuración para seleccionar un fuego. Se mantienen las características de la primera configuración para el reconocimiento, pero se tiene en cuenta la posición de la mano para saber que fuego va a ser seleccionado. El gesto elegido en esta configuración para encender la placa es mantener durante un tiempo determinado las dos palmas abiertas.

Los problemas que aparecen en esta configuración están asociados a cómo de subir y bajar la potencia. Esto se debe a que se utiliza el scroll vertical con el puño cerrado. El problema de este gesto es su sensibilidad. Puede ser difícil establecer la potencia deseada. Además hay que remarcar las limitaciones que posee el sensor, ya que debe existir una mínima distancia entre la mano y el sensor para que esta sea capturada, igual que hay una distancia máxima a la cual es capturada. Esto se traduce en el problema de que depende a que altura se comience el gesto no se podrá llegar a alcanzar la potencia deseada. Esto puede ocurrir si la mano no sea detectada porque se encuentra a una distancia mayor que el alcance o porque se encuentra demasiado cerca del sensor.

La tercera configuración intenta solucionar todos los problemas que se han encontrado en las anteriores, utilizando cada uno de los puntos fuertes de las anteriores. En esta configuración para encender se utilizan las dos palmas abiertas, y no se produce una selección automática de un fuego. Se realiza la selección de un fuego poniendo la palma abierta sobre él. Además una vez seleccionado un fuego se puede pasar a un fuego contiguo realizando un círculo horizontal hacia la derecha, para seleccionar el fuego de la derecha o a la izquierda, para seleccionar el de la izquierda. También se puede seleccionar otro fuego poniendo la palma encima directamente. Para deseleccionar un fuego se utiliza el mismo gesto. Para realizar la subida de la potencia, se puede hacer con dos gestos, realizando un círculo vertical hacia la derecha, o haciendo un clic en el aire en la parte derecha del sensor. Para realizar la bajada de potencia se hace con un círculo vertical hacia la izquierda o bien un clic en el aire en la izquierda del sensor. Tanto para subir y bajar la potencia de un fuego es necesario que esté seleccionado.

6.3. Análisis de la usabilidad, robustez y facilidad de aprendizaje de la aplicación

Con el fin de evaluar tanto la usabilidad, la robustez de la aplicación, y la facilidad de aprendizaje se ha realizado un estudio divido en tres fases. En la primera se comprueban como de fáciles de realizar son los gestos la primera vez. A continuación estas personas se han sometido a una fase de entrenamiento que tiene una duración aproximada de 5 minutos. Finalmente a estas personas se les ha sometido a las pruebas iniciales y los resultados son comparados. Todos los datos, tanto de la fase antes del entrenamiento como los de después, se obtienen a través del uso de la aplicación que se ha explicado en el capítulo anterior.

6.3.1. Resultados obtenidos sin fase de entrenamiento

En esta primera fase se ha sometido a diferentes usuarios que no han realizado los gestos con anterioridad a la aplicación de robustez anteriormente citada, para averiguar como de intuitivos son estos gestos. A través del análisis de toda la información, se puede ver cuáles son los rangos óptimos para los parámetros que identifican los gestos. Serán aquellos que permitan alcanzar una mayor *precision* y *recall*.

	TP	FP	FN	TN	Precision	Recall
Palma abierta	2426	94	689	22867	0.96269841	0.7788122
Dos palmas	2295	0	0	23781	1	1
Punyo	2212	1305	638	21921	0.62894512	0.77614035
Círculo vertical derecha	1982	0	583	23511	1	0.77270955
Círculo vertical izquierda	2105	0	460	23511	1	0.82066277
Palma vertical	2503	139	62	23372	0.94738834	0.97582846
Círculo horizontal derecha	619	0	1946	23511	1	0.24132554
Círculo horizontal izquierda	355	0	2210	23511	1	0.13840156
Clic derecha	26	1	2	26047	0.96296296	0.92857143
Clic izquierda	77	2	0	25997	0.97468354	1
					0.94766784	**0.74324519**

Tabla 17: Resultados obtenidos con usuarios sin entrenamiento, con los mejores rangos de parámetros posibles.

En los datos de la tabla 17 se observa que para todos los gestos se consigue una muy buena *precision*, es decir que no hay casi falsos positivos. Eso quiere decir que estos gestos no son malos para ser utilizados en la aplicación final. No se producirían casos en los que se realizaría un gesto y se interpretaría como otro lo que puede producir una situación peligrosa. El único gesto que tiene una *precision* que es bastante mala es el puño. Esto se debe a que se confunde con el gesto del keyTap. Así que según estos datos no se tiene que utilizar el gesto de puño con los otros dos gestos en keyTap.

Si se observan los datos del *recall* se puede ver que los gestos estáticos tienen un valor bastante alto, lo que quiere decir que la mayoría de las veces que se realiza el gesto es reconocido como tal. En los círculos horizontales se ve que su valor es muy malo. Esto

es debido a que la forma de realizar el gesto para que este sea detectado es distinta a la que se realiza de manera intuitiva.

Los rangos de los parámetros que finalmente han sido elegidos para maximizar la *precision* y el *recall* se muestran en la tabla 18.

	#HANDS	#FINGERS	X pos (1)	Y norm (1)	X norm (1)	#KeyTap	Z norm Circ	Y norm Circ
Palma Abierta	=1	>=3						
Dos palmas	>=2	>=7						
Punyo	=1	<=1		<-0.97				
Círculo vertical derecha	=1	<=2					<-0.85	
Círculo vertical izquierda	=1	<=2					>0.875	
Palma vertical_1	=1	<=2			<-0.81			
Palma vertical_2					>0.81			
Círculo horizontal derecha	=1	<=2						<-0.67
Círculo horizontal izquierda	=1	<=2						>0.7
Clic derecha	=1	<=2	>0			=1		
Clic izquierda	=1	<=2	<0			=1		

Tabla 18: Rangos para obtener los mejores resultados antes de entrenamiento.

Estos datos se han obtenido ajustando los parámetros de filtrado que identifican un gesto para intentar maximizar el *recall* y la *precision* sin perder ninguna de ellas. Cuando más bajos se establecen estos parámetros de filtrado se consigue un mayor *recall*, es decir se reconocen más gestos que son realizados como tal, pero se disminuye la *precision*, debido a que aparecen más falsos positivos. Esto se puede observar claramente en los datos que se obtienen del gesto puño.

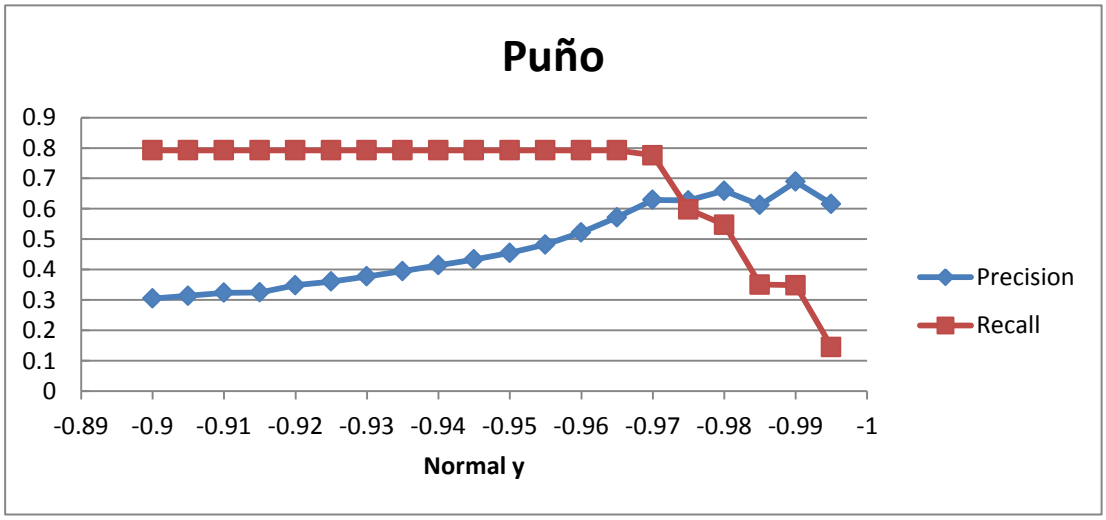

Figura 47: Muestra como conforme aumenta la *precision* el *recall* disminuye.

En la figura 47 se observa que conforme aumenta el valor que debe tener la normal del eje "y" para que el puño sea reconocido, la *precision* aumenta, es decir se tiene un menor número de falsos positivos. Al principio el *recall* no se ve comprometido pero llega un momento que para una ganancia pequeña de *precision* se produce un gran descenso del *recall*. Por tanto, el mejor valor de la normal de la mano en el eje "y" es 0.97.

6.3.2. Fase de entrenamiento

La fase de entrenamiento consiste en enseñar a los usuarios finales como deben realizar cada uno de los gestos. En el caso de usuarios que no han realizado estos gestos con anterioridad, se ve que el *recall* es bajo, sobre todo en los círculos horizontales. Se ha observado que la forma más intuitiva para realizar el gesto es diferente a la mejor forma para que sea reconocido por el Leap Motion. Durante esta fase de entrenamiento se corregirá la forma de realizar este gesto. Este proceso se ha de realizar con todos los gestos.

6.3.3. Resultados obtenidos después de fase de entrenamiento

Después de realizar la fase de entrenamiento los usuarios han repetido las pruebas. En la tabla 18 se ven los resultados obtenidos

	Sin Entrenamiento		Con entrenamiento	
	Precision	**Recall**	**Precision**	**Recall**
Palma abierta	0.96269841	0.7788122	0.98022356	1
Dos palmas	1	1	1	1
Punyo	0.62894512	0.77614035	0.58074376	1
Círculo vertical derecha	1	0.77270955	1	0.94989011
Círculo vertical izquierda	1	0.82066277	1	0.8877193
Palma vertical	0.94738834	0.97582846	1	0.97982456
Círculo horizontal derecha	1	0.24132554	1	0.91447368
Círculo horizontal izquierda	1	0.13840156	1	0.95263158
Clic derecha	0.96296296	0.92857143	1	1
Clic izquierda	0.97468354	1	1	1
	0.94766784	**0.74324519**	**0.95609673**	**0.96845392**

Tabla 18: Comparación de *precision* y *recall* antes y después del entrenamiento.

Como se ve la *precision* apenas ha sufrido modificación. Donde sí que se puede apreciar una gran modificación es en el *recall*. Esto es debido a que ahora los usuarios sí que han realizado los gestos de la forma que mejor son capturados. Las principales diferencias se encuentran en los círculos, tanto en los verticales como en los horizontales pero esa mejoría es mayor en los círculos horizontales. Como se ha explicado anteriormente, se puede observar en la figura 46 que dependiendo de la forma de realizar el gesto en los círculos horizontales marca mucho el resultado. La forma utilizada por los usuarios inicialmente era como la figura 45, "manera a". Durante el entrenamiento se les ha explicado la mejor forma de realizar este gesto y la mejoría ha sido notable. En la ejecución de los círculos verticales también hay mejoría.

Los usuarios dejaban fija la mano trazando el círculo únicamente con el dedo. La mejor forma de detectar este gesto es dejando el dedo fijo y haciendo círculos con la mano completa, pero este gesto no es tan crítico como lo es el círculo horizontal.

En la figura 48 se muestra la comparación del *recall* en los círculos horizontales antes y después del entrenamiento. La *precision* es uno en los dos casos.

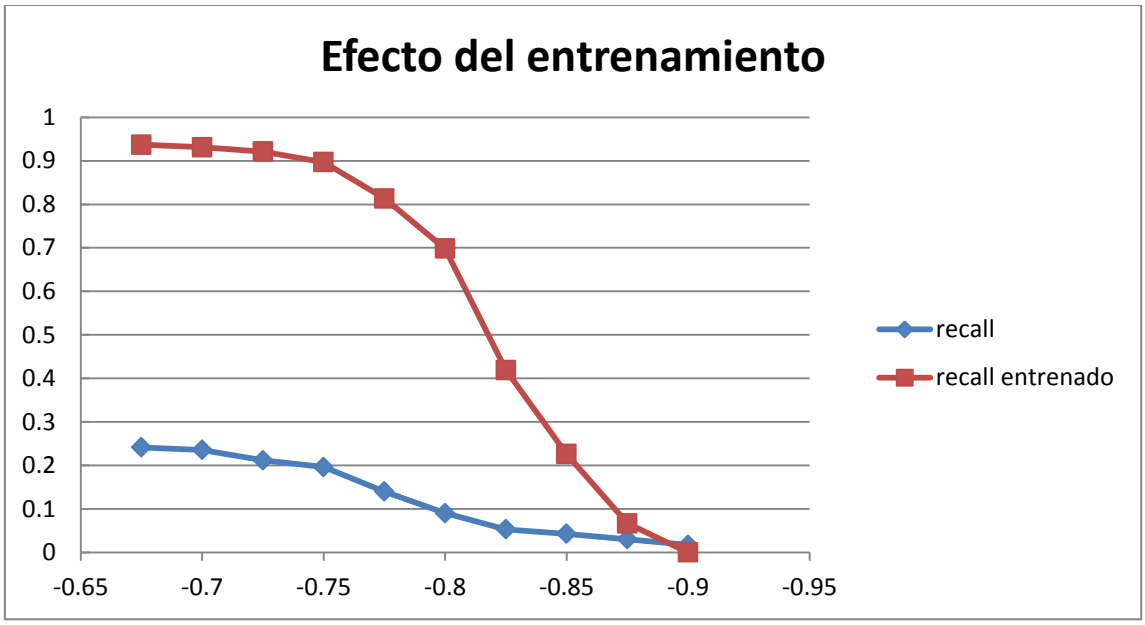

Figura 48: Comparación del *recall* obtenido para los círculos horizontales antes y después del entrenamiento.

La elección de los parámetros que determinan la detección de cada uno de los gestos se ha realizado de la misma manera que en la fase anterior, y los resultados se muestran en la tabla 19

	#HANDS	#FINGERS	X pos (1)	Y norm (1)	X norm (1)	#KeyTap	Z norm Circ	Y norm Circ
Palma Abierta	=1	>=4						
Dos palmas	>=2	>=7						
Punyo	=1	<=1		<-0.965				
Círculo vertical derecha	=1	<=2					<-0.85	
Círculo vertical izquierda	=1	<=2					>0.85	
Palma vertical_1	=1	<=3			<-0.95			
Palma vertical_2					>0.96			
Círculo horizontal derecha	=1	<=2						<-0.63
Círculo horizontal izquierda	=1	<=2						>0.63
Clic derecha	=1	<=2	>0			=1		
Clic izquierda	=1	<=2	<0			=1		

Tabla 19: Rangos para obtener los mejores resultados después de entrenamiento.

6.4. Aplicación

Con todos los datos que se han obtenido y las ventajas e inconvenientes de las distintas configuraciones se ha desarrollado una aplicación. Consta de un interfaz gráfico que representa la placa sobre la cual se va a interactuar, además de la implementación de cada uno de los gestos necesarios para interactuar.

La configuración elegida para interactuar con la placa es la tercera, debido a que ofrece un mayor número de formas para realizar las mismas acciones, además de ser la que menos problemas presenta. Los gestos estáticos que se han elegido como pueden ser palma abierta, dos palmas abiertas, o cualquiera de los restantes, necesitan un número determinado de frames para ser reconocidos. Esto es debido a que si el gesto fuera reconocido con un solo frame, se tendrían problemas, por la gran cantidad de frames que el sensor es capaz de capturar. Además de este problema existe otro, debido la sensibilidad del sensor. Puede haber un momento en el que se esté realizando un determinado gesto, y el sensor en un frame determinado reconozca otro gesto, de tal manera que si se esperan todos los frames necesarios seguidos, el gesto no será reconocido. Este problema se ha solucionado a través de un algoritmo que comprueba si el 80% de los frames que han sido capturados anteriormente son de un mismo gesto y en ese caso el gesto es reconocido.

Descripción formal del algoritmo:

a) **Paso 1:** Mientras que(NumFramesCapturados < tiempo * frameRate) hacer
   - **a.** Aumentar el contador del frames correctos del gesto correspondiente
   - **b.** Capturar nuevo frame
   - **c.** NumFramesCapturados = NumFramesCaputarados + 1;

   Fin
b) **Paso 2:** Si hay algún gesto que tiene más del 80% de los frames capturados
   - **a.** Ejecuta la acción correspondiente a dicho gesto
c) **Paso 3**: Sino
   - **a.** Contadores de gestos correcto a 0, excepto el del último gesto capturado

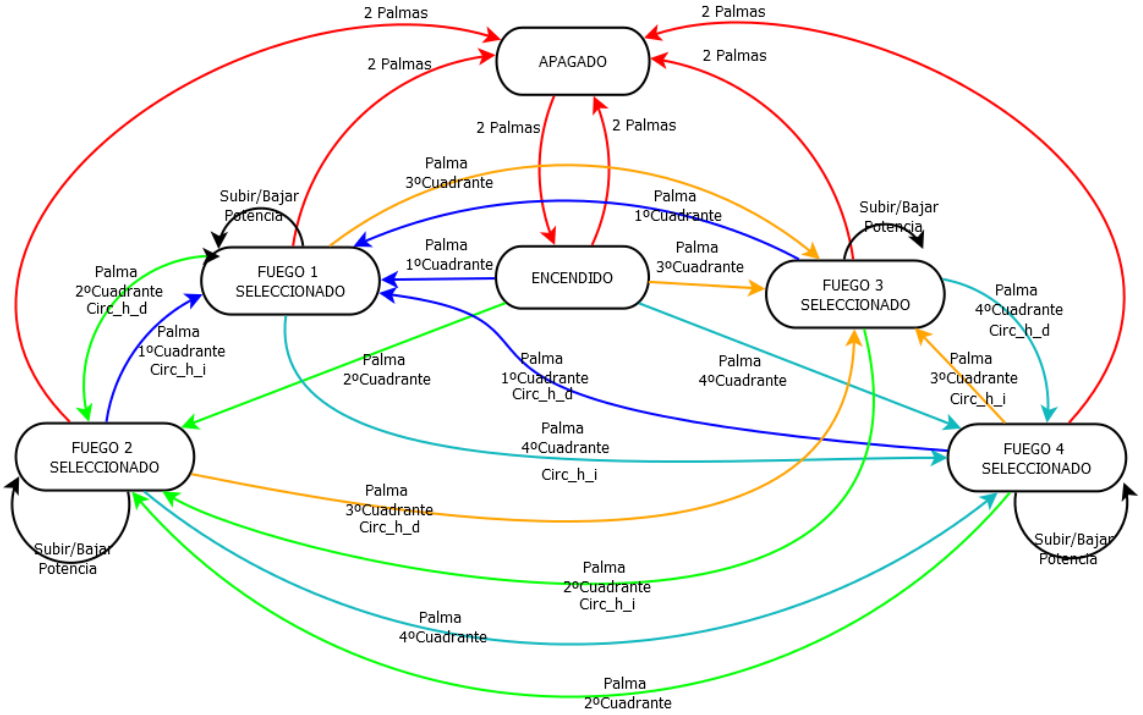

Figura 49: Diagrama de estados de la aplicación.

En el diagrama de estados de la figura 49 se puede observar que cuando la placa está apagada el único gesto que produce un cambio de estado es dos palmas. Cuando este gesto sea reconocido, entendiendo como reconocido que se haya realizado durante el tiempo correspondiente y el 80% de los frames que se hayan capturado durante ese tiempo sean correctos, la placa se encenderá, mostrando una potencia inicial para los fuegos de 0. Una vez la placa esté encendida son reconocidos tanto el gesto de dos palmas con el cual se apagaría la placa y el gesto de palma abierta con el cual se selecciona uno de los fuegos de la placa. La selección de un fuego depende de la posición en la que se ponga la mano. El alcance del sensor se ha dividido en cuatro cuadrantes, de tal manera que el fuego seleccionado será el uno si la palma abierta se sitúa en el cuadrante arriba izquierdo, el dos arriba derecha, el tres abajo izquierda y el cuatro abajo derecha. Al igual que ocurre con el gesto de dos palmas, este también es necesario realizarlo durante un tiempo determinado para que sea reconocido. Como se puede observar este gesto queda dividido en cuatro subgestos, los cuales son diferenciados gracias a la posición de la mano.

Una vez que se tiene seleccionado el fuego se pueden realizar otras acciones. Así se puede deseleccionar el fuego, realizando el gesto de palma abierta sobre el fuego seleccionado o seleccionar otro fuego de la misma forma que se realizó la selección del primero. También se puede seleccionar uno de los fuegos contiguos, realizando el gesto de círculo horizontal derecha o izquierda. También se puede dar potencia o quitársela al fuego seleccionado. Para ello se utilizan los gestos de círculo vertical derecha y círculo vertical izquierda, para subir y bajar la potencia respectivamente.

A la hora de implementar esta forma de subir y bajar la potencia, aparece el problema del tiempo que se tiene que realizar el gesto. Este problema surge debido a que el sensor devuelve la normal del círculo en cuanto detecta un movimiento que se le parece, aunque no se haya realizado el círculo completo. Para solucionar este problema, se ha estudiado la documentación que ofrece el sensor. Viendo toda la información que devuelve del círculo, inicialmente se hizo una primera aproximación, a través de la suma del ángulo avanzado entre un frame y otro y al llegar a los 360^{o} la potencia era subida o bajada. Pero no se consiguieron buenos resultados, así que se utilizó la información del progreso del círculo, que es un número entero positivo, que devuelve el número de vueltas realizadas. Con esta información se desarrolló un algoritmo que da mejores resultados que el anteriormente citado.

Descripción formal del algoritmo:

a) **Paso 1:** Si gesto == círculo
   - **a.** Si circulo.estado() == start
     - **i.** procesoAnterior = 0;
   - **b.** Sino si circulo.estado() == update
     - **i.** procesoActural = circulo.progreso();
     - **ii.** Si(procesoActual – procesoAnterior >= 1)
       1. Si(circulo.normal().z > 0.8 && circulo.normal().z <= 1)
          - **a.** potenciaFuego = potenciaFuego +1;
       2. Sino si(circulo.normal().z>- 0.8 && circulo.normal().z<=-1)
       3.
          - **a.** potenciaFuego = potenciaFuego -1;
     - **iii.** Sino
       1. procesoAnterior = 0;

     Fsi

   Fsi

Fsi

Para determinar si se debe aumentar o disminuir la potencia, se tiene en cuenta el valor de la normal en el eje “z”, de tal manera que si se encuentra entre 0.8 y 1, se aumenta, ya que el giro se ha realizado hacia la derecha, y si se encuentra entre -0.8 y -1, se disminuye, ya que el giro se ha realizado hacia la izquierda.

Este algoritmo permite saber cuándo se debe aumentar y disminuir, pero con una pequeña ampliación, también permite identificar cuando se ha realizado un círculo horizontal hacia la derecha o izquierda para cambiar de fuego seleccionado, teniendo en cuenta además del valor de la normal en el eje “z”, en el eje “y”.

Se ha implementado también otra manera alternativa para subir y bajar la potencia, con el gesto keyTap, si este gesto es realizado en la parte derecha del sensor entonces la potencia se incrementa. Si por el contrario el gesto se realiza en el lado izquierdo del sensor la potencia se decrementa.

Por último también se puede quitar toda la potencia del fuego que está seleccionado utilizando el gesto de palma vertical, al igual que con los gestos de palma abierta y dos palmas. Para que este gesto sea reconocido es necesario que se realice durante un tiempo determinado. Con este gesto lo que se hace es poner la potencia a 0 del fuego seleccionado.

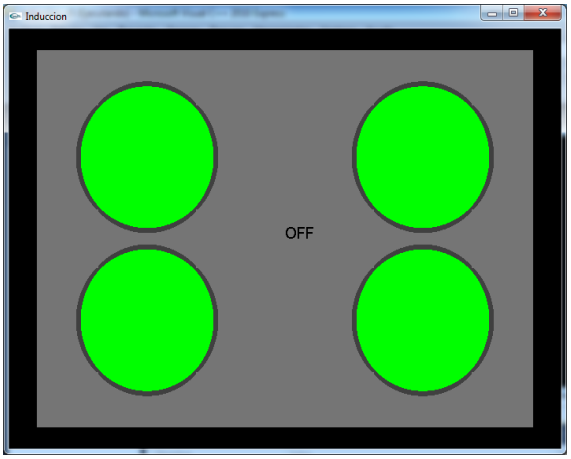

Figura 50: la placa se encuentra apagada.

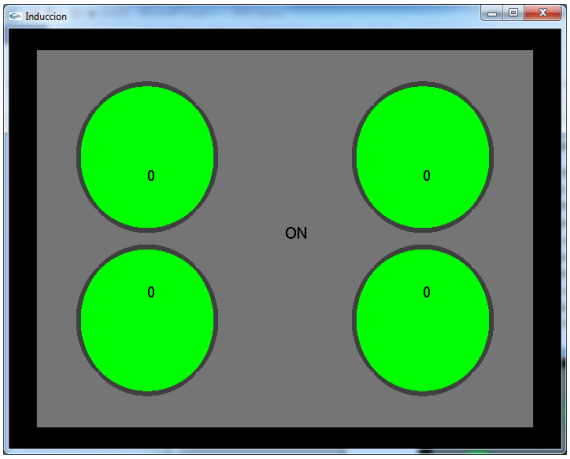

Figura 51: La placa se encuentra encendida.

Inicialmente cuando la placa se encuentra apagada se puede ver en el centro de la placa que pone “OFF” (Figura 50) y cuando es enchufada se cambia por “ON” apareciendo encima de cada fuego la potencia que tiene, inicialmente 0 (Figura 51). La selección de un fuego queda reflejada a través del cambio de color del fuego, pasando de color verde a azul (Figura 52). Si la potencia es mayor que 0 y el fuego no está seleccionado entonces el fuego se pone de color rojo (Figura 53).

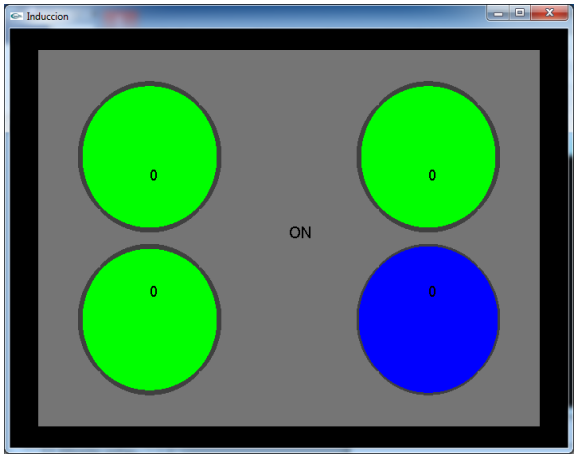

Figura 52: Fuego seleccionado.

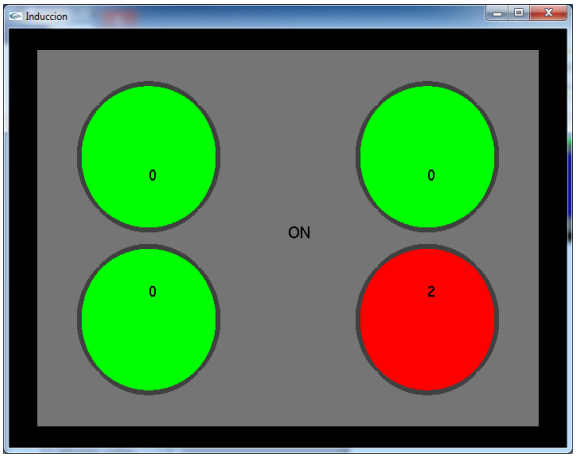

Figura 53: Fuego con potencia distinta de uno.

Capítulo 7

Conclusiones y trabajo futuro

Los objetivos que se han planteado para este trabajo se han cumplido satisfactoriamente. El primero de los objetivos que se había establecido era determinar la técnica de percepción utilizada por Leap Motion para la obtención de la información 3D. Se ha llegado a la conclusión de que es un sistema estéreo. Con dos imágenes obtenidas de las cámaras que posee. Junto con la utilización de los diferentes niveles de gris obtenidos gracias a sus tres leds para completar el mapa de profundidad. Una vez identificado la técnica el siguiente objetivo era el desarrollo de un sistema básico de reconocimiento de gestos. Se ha desarrollado un sistema capaz de identificar gestos estáticos, los cuales se diferencian por el número de dedos que se encuentran estirados.

Otro de los objetivos de este trabajo era la definición de un vocabulario de gestos a partir de los datos capturados por Leap Motion. Se ha realizado un estudio de la *precision* y el *recall* de los gestos en cada uno de los modos de funcionamiento de Leap. Se ha llegado a la conclusión que el *modo equilibrado* es el mejor modo para el reconocimiento de gestos tanto estáticos como dinámicos. Si las condiciones para el reconocimiento son malas, se establece automáticamente el *modo robusto*, con el cual se produce una peor detección de los gestos dinámicos, pero es mucho menos sensible a la distancia del sensor a la que se realicen los gestos.

El último objetivo era el desarrollo de una aplicación para el control de una cocina de inducción a través de gestos realizados con las manos. Para ello se ha realizado un estudio de como de fácil de realizar y aprender cada gesto, con el objetivo de decidir que gestos que deben ser usados en la aplicación. Una vez realizado este estudio se han establecido diferentes configuraciones, y se han analizado las ventajas e inconvenientes. Una vez detectada la mejor configuración se ha llevado a cabo la implementación de esta aplicación utilizando el lenguaje C++.

Este es un ámbito en el cual no se han desarrollado muchas aplicaciones y como trabajo para el futuro se podría completar el desarrollo de la aplicación de reconocimiento gestual, de tal manera que no se dependa de las bibliotecas del sensor

Leap. Otro objetivo que se puede abordar en un futuro es la definición e implementación de otros gestos, que permitan una interacción más intuitiva que la conseguida en este trabajo además de la optimización del código desarrollado, con el fin de poder utilizar dicha aplicación en un micro de capacidades reducidas.

Bibliografía

[1] Computer vision-http://www.escet.urjc.es/~visionc/VisionPorComputador.pdf

[2] Vision 3D: http://www.escet.urjc.es/~visiona/tema6.pdf

[3] Computer vision-http://www.escet.urjc.es/~visionc/VisionPorComputador.pdf

[4] Time-of-Flight-- http://campar.in.tum.de/twiki/pub/Chair/TeachingSs11Kinect/2011-DSensors_LabCourse_Kinect.pdf

[5] Leap Official Site -- https://www.leapmotion.com

[6] Blog oficial - http://blog.leapmotion.com/

[7] B. Deimel and S. Schr□oter, Improving Hand-Gesture Recognition via Video BasedMethods for the Separation of the forearm from the Human Hand. In Gesture Workshop, 1999.

[8] Nobuyuki Otsu. *"A threshold selection method from gray-level histograms"*. IEEE Trans. Sys., Man., Cyber. 9 (1): 62–66, 1979.

[9] Suzuki, S. and Abe, K., "*Topological Structural Analysis of Digitized Binary Images by Border Following"*. CVGIP 30 1, pp 32-46,1985

[10] Sklansky, J., "*Finding the Convex Hull of a Simple Polygon"*. PRL 1 $number, pp 79-83,1982

[11] OpenCV. OpenCV Documentation. [en línea] disponible en: http://opencv.itseez.com/index.html

[12] OpenCV. OpenCV Tutorials. [en línea] disponible en: http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc.html

[13] Leap C++ API -- https://developer.leapmotion.com/documentation/cpp/api/Leap_Classes.html

[14] 3D Imaging for Hand Gesture Recognition: Exploring the Software-Hardware Interaction of Current Technologies – Frol Periverzov, Horea T. Ilies. 3D Research, 03, 2012.

[15] X. WenWenWen, Y. Niu , "A Method for Hand Gesture Recognition Based on Morphology and Fingertip-Angle", The 2nd International Conference on Computer and Automation Engineering (ICCAE) vol. 1, pp. 688 -691, 2010