CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE BIOELECTRÓNICA

Diseño y desarrollo de un sistema infrarrojo de registro de movimiento de instrumental neuroendoscópico.

Tesis que presenta

Moisés Alberto Domínguez Sánchez

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de la Tesis Daniel Lorias Espinoza

Ciudad de México. Febrero 2017

Dedicatoria

A mi familia, el núcleo central de mi existencia. A mis padres, Fidel y Carmen, por su cariño, su apoyo incondicional y por siempre creer en mí. A mis hermanos, Álvaro, Sergio y Andrea por sus consejos y enseñanzas a lo largo de mi vida.

A Claire Closset por su cariño y apoyo constante durante mi estancia en la Ciudad de México.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado durante el periodo de este proyecto.

Al Centro de Investigación y de Estudios Avanzados del IPN por permitirme ser estudiante de la institución.

Al Dr. Daniel Lorias Espinoza por permitirme ser parte de su equipo de investigación, por su apoyo y enseñanzas durante toda la Maestría.

A mis profesores por brindarme sus conocimientos y enseñanzas que contribuyeron a mi desarrollo personal y profesional.

A mis compañeros de generación, por brindarme experiencias inolvidables. A Víctor Contreras Machado por su amistad durante mi estadía en la Ciudad de México.

Resumen

En este proyecto se describe el diseño y desarrollo de un sistema de seguimiento de movimiento basado en las cámaras infrarrojas que se encuentran embebidas en los controles del Nintendo Wii, que son utilizados para estimar la posición en el espacio tridimensional de marcadores infrarrojos. Si bien, existen distintas aplicaciones en las que un sistema de *tracking* puede ser utilizado, el presente proyecto se realizó, pensando en utilizar el sistema para el seguimiento de instrumental neuroendoscópico. Se utilizaron dos controles de Nintendo Wii en configuración ortogonal e independientes entre ellos, de tal modo que, cada cámara tiene su propio sistema de coordenadas y observa su propio plano de imagen. Esta configuración permite el registro de 3 grados de libertad completos.

Se propone un sistema activo de tipo *outside-in* en el cual, los sensores se encuentran en una posición y orientación fijas y detectan un objetivo en movimiento. Las cámaras de los Wiimotes poseen un motor de seguimiento multi-objetos que provee el seguimiento de hasta 4 marcadores infrarrojos a una frecuencia de 100 Hz, se utiliza el protocolo de comunicación Bluetooth como método de intercambio de información entre la computadora y los sensores.

El sistema de tracking se integró en una interfaz de usuario que provee información en tiempo real de ciertos parámetros de las manipulaciones del instrumental neuroendoscópico tales como ángulo axial, ángulo lateral, profundidad de inserción y las coordenadas x, y, z del instrumental en el espacio. Además, se muestran las vistas axial, anteroposterior y lateral de la región lumbar de la columna como retroalimentación visual.

Abstract

In this research, we present the design and implementation of a tracking system based on the infrared cameras of Nintendo Wii Remotes, used for estimation of position and orientation in the 3D space of infrared markers. Even though, there is a wide variety of applications for using a tracking system, we propose a surgical instrumental tracking approach for spine fixation procedures. We used an orthogonal setup with two Wiimotes independent from each other with their own coordinate system and their own image plane. This setup handles 3 degrees of freedom.

We present an active outside-in system were the sensors are fixed in position and orientation and detect an object in movement. The Wii Remote camera contains a Charged Coupled Device and an infra-red filter with a hardware resolution of 128x96 pixels, with a reported virtual resolution of 1024x768 pixels at 100Hz. The Wiimote is a wireless communications device that transmits and receives data via a Bluetooth link.

A friendly user interface allows the visualization of the spine anatomy in Posteroanterior, Lateral and Axial planes for visual. A trajectory line referenced to the long axis of the probe is superimposed in the above-mentioned views, as the probe is moved through the pedicle, the position of the trajectory line will change accordingly. Furthermore, the systems has an alert that allows the user to know when the surgical instrument its inside the spinal cord.

Índice

Índice de Figuras.

Índice de Tablas.

Capítulo 1

Introducción

El seguimiento de objetos en movimiento, es una actividad que se realizaba desde tiempos remotos. Las personas en sociedades antiguas solían seguir el movimiento de sus presas para cazarlas y poder alimentarse, inventaron formas de seguir el movimiento de las estrellas para propósitos de navegación y predecir cambios climáticos en sus entornos. El seguimiento de objetos ha sido una tarea esencial para la supervivencia y progreso de la humanidad.

Por muchos años, investigadores han explorado sistemas basados en tecnologías mecánicas, magnéticas, acústicas, ópticas, entre otras, para capturar el movimiento de objetos. La proliferación de ordenadores de alta potencia, la disponibilidad de cámaras de bajo costo y alta calidad, además de la creciente necesidad de análisis de video, ha generado un creciente interés en algoritmos para tracking, por lo que numerosos enfoques han sido propuestos. El seguimiento o *tracking* tiene una amplia variedad de áreas en las que se puede aplicar, incluyendo medicina, entretenimiento, tecnologías de control, aplicaciones militares, videojuegos por mencionar algunas, en donde se requiere conocer la posición y orientación de un objeto físicamente real [1,2].

En su forma más simple puede ser definido como el problema de estimar la trayectoria de un objetivo, utilizando sensores, en un plano de imagen mientras que éste se mueve alrededor de una escena. Un sensor puede ser cualquier dispositivo de medida que pueda ser utilizado para obtener información acerca de objetos dentro de un ambiente, tales como, sonares, radares, cámaras, sensores infrarrojos, micrófonos, ultrasonidos, o cualquier otro dispositivo que sirva para estos propósitos. Esencialmente, es un problema de estimación del estado de un objeto de interés como por ejemplo su velocidad, posición o aceleración, además permite determinar el número de objetos en una escena, sus identidades o características [3].

Existen numerosas fuentes de incertidumbre en el problema de seguimiento de objetos que lo hacen una tarea complicada y no trivial. Por ejemplo, los objetos en movimiento con frecuencia se encuentran sujetos a perturbaciones aleatorias, pueden ser no detectados por los sensores o el número de objetos en el campo de visión de un sensor pueden cambiar al azar. Además, las mediciones de los sensores se encuentran sujetas a ruidos y el número de mediciones recibidas por el sensor pueden cambiar de un marco a otro impredeciblemente. Los objetos en una misma escena pueden estar juntos y las mediciones recibidas pueden no distinguir entre ellos. En ocasiones, los sensores proveen información cuando no existe objeto alguno en su campo de visión [4]. Un sistema de seguimiento típico se muestra en la figura 1:

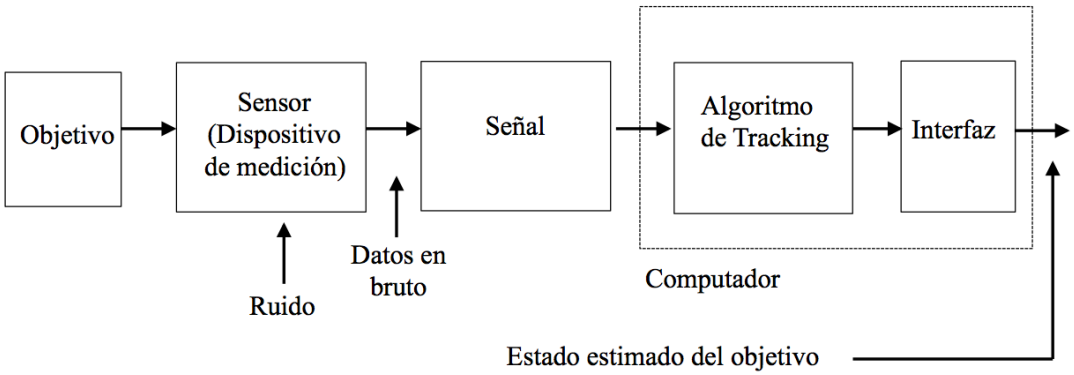


Fig. 1 Diagrama general a bloques de un sistema de seguimiento

El objetivo en el contexto de este proyecto es diseñar un sistema de registro de movimiento para realizar una tarea de seguimiento de instrumental quirúrgico orientada a procedimientos de fijación de columna mediante instrumentación que permita en un futuro cercano a residentes de neurocirugía llevar a cabo simulaciones de los procedimientos realizados en quirófano y que además les permita conocer parámetros importantes.

La cirugía de columna es un procedimiento complejo que requiere amplia experiencia y manipulaciones muy precisas de los instrumentos quirúrgicos. Para mejorar la seguridad y la exactitud de estos procedimientos, los cirujanos tienen que pasar por un proceso de formación

a largo plazo. El entrenamiento, de un estudiante o residente, en algún punto debe realizarse con pacientes bajo la supervisión de un experto, el cual es un proceso lento, inherentemente subjetivo y carece de objetividad, evaluación cuantitativa y rendimiento quirúrgico. Sin embargo, existe la obligación de proveer un entrenamiento optimo, así como también garantizar la seguridad del paciente. Por estas razones, la enseñanza, aprendizaje y evaluación basados en simuladores representa una opción viable para satisfacer esas necesidades mediante el desarrollo de habilidades, conocimientos y actitudes de los cirujanos mientras se protege al paciente de riesgos innecesarios [5].

1.1 Planteamiento del problema.

La mayoría de los sistemas de tracking ópticos utilizan cámaras de video que son susceptibles a interferencias externas del ambiente como la luz. Además, las frecuencias a la que trabajan las cámaras comerciales son bajas (30 Hz) para tareas de procesamiento de imagen en tiempo real. Una tecnología útil para el seguimiento óptico de movimiento son las cámaras infrarrojas (IR) que pueden informar las coordenadas proyectadas en 2D de las fuentes de luz IR. Las coordenadas proyectadas en 2D de un arreglo 3D específico conocido de marcadores IR se procesan a continuación para estimar la posición y la orientación de un objeto en el espacio tridimensional usando algoritmos de visión computarizados apropiados. Las cámaras infrarrojas no se ven afectadas por la luz ambiental y generalmente trabajan a frecuencias mucho mayores (100 – 150 Hz). El Wiimote es un dispositivo de juego que permite al usuario manipular objetos en un entorno virtual, con la capacidad de detectar gestos a través de su acelerómetro y sensores ópticos. Además proveen el seguimiento de hasta 4 marcadores infrarrojos a una frecuencia de 100 Hz.

Es por ello que, se propone el diseño e implementación de un sistema de seguimiento de movimiento orientado a ser utilizado en procedimientos de cirugía de columna. El sistema se basa en las mediciones obtenidas de los sensores infrarrojos que se encuentran en los mandos del Nintendo Wii ®. La ventaja de utilizar este tipo de cámaras radica en que ofrece

comunicación vía Bluetooth y es posible superponer marcadores infrarrojos en el instrumental sin necesidad de cables de comunicación o cables de alimentación que limiten el movimiento del instrumental, además, el campo del procedimiento quirúrgico no se ve alterado por dispositivos externos.

1.2 Objetivos

1.2.1 Objetivo General.

Diseñar e implementar un sistema infrarrojo de seguimiento de movimiento que permita el registro de 3 grados de libertad del instrumental neuroendoscópico.

1.2.2 Objetivos Particulares

- Utilizar las cámaras infrarrojas de los controles de Nintendo Wii ® para implementar un algoritmo de tracking que permita obtener la posición en el espacio tridimensional del instrumental quirúrgico.

- Desarrollar un algoritmo que permita calcular parámetros de las manipulaciones del instrumental como la profundidad y ángulo de inserción.

- Diseño del circuito para los marcadores infrarrojos.

- Implementar un sistema de alerta que permita al usuario conocer cuándo se ha introducido el instrumental en el conducto raquídeo.

- Desarrollar una interfaz de usuario que permita una retroalimentación visual del procedimiento de inserción del instrumental, en las vistas anteroposterior, lateral y axial.

1.3 Estructura de la tesis.

- Antecedentes: describe los conceptos básicos en lo que a cirugía de columna se refiere, además se describen diversos sistemas de entrenamiento que existen en el área, sistemas de seguimiento, así como también, las características principales de los controles de Nintendo Wii ® y las librerías utilizadas para realizar la comunicación de los controles con la computadora.

- Desarrollo: describe las características del sistema de seguimiento utilizado, las formulas, algoritmos y matemáticas necesarias, la medición y evaluación de las métricas, caracterización del sistema, así como el diseño de la interfaz de usuario.

- Pruebas: explica la metodología de las pruebas a las que se sometió el sistema.

- Resultados: indica los resultados obtenidos de las pruebas del sistema.

- Discusión: análisis y comparación de resultados.

- Conclusiones y perspectivas: resolución de la eficacia y la visión a futuro del sistema.

Capítulo 2

Antecedentes.

2.1 Seguimiento de objetos (*Object Tracking*)

El seguimiento o *tracking* de objetos es uno de los componentes más importantes para aplicaciones de visión por computador, tales como seguridad, vigilancia, interacción humano-computador, realidad aumentada, control de tráfico e imágenes médicas. El objetivo es localizar uno o varios objetos en movimiento en tiempo real, y proveer información respecto a su posición en el espacio [6]. Existen distintos métodos para el seguimiento de objetos, algunos de ellos son:

I. Seguimiento Óptico (Optical Tracking).

Los métodos generales para el seguimiento óptico de objetos están clasificados en 3 categorías:

i. *Point Tracking:* (Seguimiento de puntos) Los objetos detectados en imágenes consecutivas están representados cada uno por uno o varios puntos y la asociación de éstos está basada en el estado del objeto en la imagen anterior, que puede incluir posición y movimiento. Se requiere de un mecanismo externo que detecte los objetos de cada fotograma [7].

ii. *Kernel Tracking:* (Seguimiento del núcleo) realizan un cálculo del movimiento del objeto, el cual está representado por una región inicial, de una imagen a la siguiente. El movimiento del objeto se expresa en general en forma de movimiento paramétrico (translación, rotación, afín...) o mediante el campo de flujo calculado en los siguientes fotogramas [7].

iii. *Silhouette Tracking:* (Seguimiento de siluetas) Estas técnicas se realizan mediante la valoración de la región del objeto en cada imagen utilizando la información que

contiene. Esta información puede ser en forma de densidad de aspecto o de modelos de forma que son generalmente presentados con mapas de márgenes [7].

La mayoría de estas técnicas utilizan cámaras de video para obtener información acerca de la posición del objeto en el espacio. El seguimiento de objetos puede ser un proceso lento debido a la gran cantidad de datos que contiene un video. Además, existen distintos factores que incrementan la complejidad del seguimiento de objetos utilizando cámaras de video normales [8]. Algunos de estos factores son:

- *Cambios de posición*. El objeto móvil de interés varia su aspecto cuando se proyecta sobre el plano de la imagen, por ejemplo, al girar.

- *Iluminación ambiente.* La dirección, la intensidad y el color de la luz de ambiente influyen en el aspecto del objeto de interés. Asimismo, los cambios en la iluminación global son con frecuencia un reto en las escenas al aire libre.

- *Ruido.* El proceso de adquisición de imágenes introduce en la señal de la imagen un cierto grado de ruido que depende de la calidad del sensor. Las observaciones del objeto de interés pueden dañarse y por tanto afectar al rendimiento del seguidor.

- *Oclusiones*. Puede ser que un objeto de interés no se observe bien cuando sea parcial o totalmente tapado por otros objetos en la escena.

- *Resolución*. La resolución de una cámara de video convencional no es suficiente para capturar los detalles requeridos.

- *Frecuencia*. La frecuencia de las cámaras a la que se muestran las secuencias de imágenes no permite capturar movimientos rápidos. Comúnmente tienen una frecuencia de 30 o 60 cuadros por segundo.

Existe una clasificación adicional para los sistemas ópticos.

- *Outside-in* (de afuera hacia adentro): Se dice que un sistema es de tipo *outside-in* si sus sensores (por ejemplo, cámaras) se encuentran fijos (posición y orientación) en el entorno como se muestra en la Fig. 2. Los objetivos a seguir tienen marcadores activos o pasivos. El numero o la forma de los marcadores que se necesitan para cada objeto se encuentran relacionados con el número de grados de libertad [3].

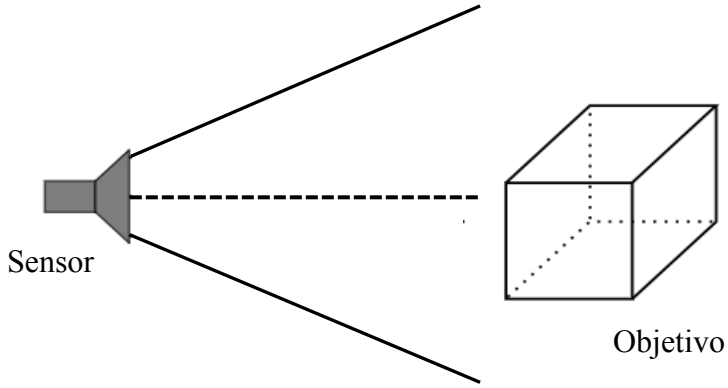


Fig. 2 Sistema de seguimiento óptico tipo *outside-in.*

- *Inside-out* (de adentro hacia fuera): Se dice que un sistema es de tipo *inside-out* si la posición y orientación se determinan mediante sensores que se encuentran directamente unidos al objeto a seguir como se muestra en la figura 3 [3].

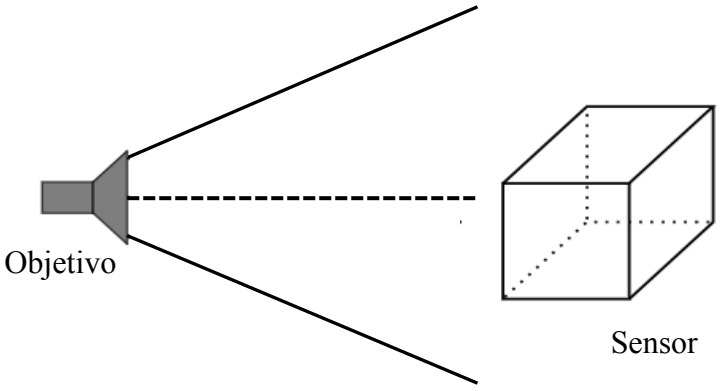


Fig. 3 Sistema de seguimiento óptico tipo *inside-out.*

En este caso los marcadores se encuentran en el entorno y los sensores superpuestos en el objeto a seguir los observan. El número y la forma de los marcadores que se necesitan para cada objeto pueden escogerse del mismo modo que para los sistemas de tipo *outside-in.*

II. Seguimiento Magnético (*Magnetic Tracking*)

Utiliza un dispositivo fuente que genera un campo magnético. Un sensor reporta la posición y orientación relativas al campo magnético generado por el dispositivo. Múltiples dispositivos y múltiples sensores permiten configuraciones que realizan un seguimiento con mayor precisión. Los sistemas magnéticos que se han desarrollado soportan frecuencias de hasta 100 Hz. Sin embargo, objetos con mucha mayor permeabilidad magnética como los metales, pueden causar interferencia en los campos magnéticos generados por los dispositivos, lo que conlleva a errores en los cálculos de la posición y orientación del sensor [6].

III. Seguimiento Acústico (*Acoustic Tracking*)

Estos sistemas utilizan dispositivos fuente que envían ultrasonido montados en el punto de interés, y utilizan receptores en el ambiente para medir el tiempo que ha tomado el sonido en llegar a ellos. Con esta información el sistema es capaz de triangular la posición de los dispositivos fuente. Desafortunadamente, superficies acústicamente reflectoras pueden causar interferencias con el sistema [6].

Se propone un sistema activo para el seguimiento de marcadores infrarrojos colocados en el instrumental quirúrgico, utilizando las cámaras infrarrojas de los controles del Nintendo Wii ®. Las características de este dispositivo se describen a continuación, así como también las ventajas de su utilización para este proyecto.

2.2 Nintendo Wii Remote ®

Wii es una videoconsola producida por Nintendo y estrenada en el año 2006, pertenece a la séptima generación de consolas. La característica más distintiva de la consola es su mando inalámbrico. El Wii Remote (Wiimote) que se puede observar en la Fig. 4 es el mando principal del Wii.

Fig. 4 Diferentes vistas del Wii Remote

Utiliza una combinación de acelerómetros y detección infrarroja (IR) para sentir su posición en un espacio tridimensional cuando es apuntado a los LEDs en el interior de la barra de sensores. Este diseño permite a los usuarios controlar el juego mediante gestos físicos, así como presionar los botones clásicos de un controlador estándar [9].

Sin embargo, las especificaciones oficiales de los mandos no fueron publicadas, a pesar de ello, la comunidad mundial de programadores ha obtenido, mediante ingeniería inversa, una parte significativa de la información técnica relativa a su funcionamiento.

2.2.1 Especificaciones Técnicas del Wii Remote ®.

El Wiimote es un dispositivo sumamente complejo y para los objetivos de esta investigación, los componentes más importantes son la cámara infrarroja y la conectividad Bluetooth por lo que se hará una breve descripción de ellos.

2.2.1.1 Cámara Infrarroja.

En la punta de cada Wiimote, hay una cámara infrarroja fabricada por PixArt Imaging que se puede observar en la Fig. 5.

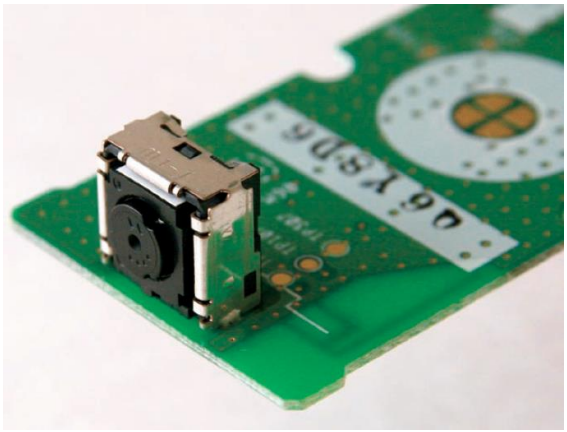


Fig. 5 Chip de la cámara IR fabricado por PixArt

La cámara posee un Motor de Seguimiento Multi-Objetos (Multiobject Tracking Engine: MOT) que provee el seguimiento de hasta 4 LEDs infrarrojos a una frecuencia de 100Hz, además posee un Dispositivo con Carga Acoplada (CCD) y un filtro infrarrojo que sensa dentro del espectro infrarrojo una longitud de 940 nm. Tiene una resolución de hardware de 128x96 pixeles, sin embargo, utiliza sub-pixel análisis para crear una resolución virtual de 1024x768 [10].

Los mandos del Nintendo Wii, pueden comprarse en México a un precio aproximado de $700 MXN. Estas especificaciones superan a cámaras web del mismo precio, las cuales típicamente proveen una resolución de 640x480 a 30Hz. Las webcams usualmente requieren una mayor carga computacional para realizar tareas en tiempo real. Además de su bajo costo, el Wiimote no se ve afectado por la iluminación del ambiente ya que esta cámara tiene incorporado un procesador de imágenes que puede analizar la imagen de la cámara en bruto, identificar un punto brillante (sin importar el fondo de la imagen) y calcular su posición (x, y) en el plano de imagen de la cámara. Otra de las características, descrita en el apartado 2.3.1.3, que hacen al Wii Remote adecuado para esta aplicación es su conectividad Bluetooth.

2.2.1.2 Campo de visión del Wiimote.

Una de las características importantes de la cámara infrarroja es su campo de visión. Wronski [11] y Lourenco [12] determinaron experimentalmente las características intrínsecas necesarias para determinar la posición de un marcador infrarrojo mediante un algoritmo. Estas características son el campo de visión horizontal y el campo de visión vertical. Mediante la medición con un marcador infrarrojo de los cambios en los ejes horizontal y vertical a diferentes distancias de la cámara, encontraron que los ángulos de visión eran 41º en horizontal y 31º en vertical y las coordenadas reportadas para "x" y "y" se encuentran en un rango de 0 a 1023 y de 0 a 767 pixeles respectivamente. Diferentes valores de estos dos ángulos se pueden encontrar en la literatura, además de que pueden variar de un Wiimote a otro, por esta razón es necesario estimar el valor de los ángulos correspondientes a cada dispositivo.

2.2.1.3 Conectividad Bluetooth.

La comunicación se establece a través de una conexión inalámbrica vía Bluetooth. La conexión utiliza un chip Broadcom 2042. El Wiimote se reconoce como un dispositivo de interfaz humana (HID) clase periférica, y el estándar Bluetooth HID permite que los dispositivos se auto-describan utilizando un bloque descriptor HID. Este bloque incluye una enumeración de los informes que se transmite a la central a través de la conexión inalámbrica al comienzo de la comunicación [9]. Es importante mencionar que, cuando se establece la conexión entre el Wiimote y el ordenador a través de un programa es posible conectar al Wiimote como un dispositivo HID. Después de esta conexión, se establece la comunicación hacia y desde el Wiimote.

2.2.2 Selección de Software.

Debido a las librerías disponibles para poder comunicar el Wiimote con la computadora, y de esta forma, obtener la posición del instrumental, se deicidio utilizar el lenguaje C#, utilizando

también XAML para crear una aplicación WPF en el entorno de Microsoft Visual Studio 2015 en un computador con Windows 7. El lenguaje C#, o C Sharp, es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Aunque es posible escribir código para la plataforma .NET en otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes, es por ello que se decidió utilizar la librería WiimoteLib de .NET. A continuación, se describe brevemente la librería utilizada.

2.2.2.1 WiimoteLib.

WiimoteLib es una librería de .NET para usar el Wii Remote de Nintendo y las extensiones de éste desde una aplicación .NET. El proyecto comenzó su vida como un artículo en sitio web de Microsoft Codin4fun, publicado por Brian Peek. En él, Peek muestra como conectar y usar un Wiimote desde C# y VB.NET. El resultado final es una API para el Wiimote bastante sencilla de usar que puede ser utilizada en cualquier aplicación gestionada. El proyecto WiimoteLib escrito en C# contiene cinco archivos .cs, donde se definen las estructuras y funciones que serán usadas para manejar el Wiimote y sus extensiones. También incluye una carpeta donde se adjunta información de ayuda sobre la API. Y, como todos los proyectos en C#, las propiedades y las referencias. Antes incluso de preguntarse cómo funciona la librería, habría que conseguir conectar el Wiimote con nuestro PC vía Bluetooth, por el momento es necesario saber que se necesita un programa para ello. El Wiimote es un dispositivo de entrada. Cuando se trabaja con esta librería lo que se persigue es localizar el dispositivo y tener en todo momento acceso a la información que reporta, para que a partir de ella la aplicación que se esté tratando realice una toma de decisiones y ejecute el código pertinente. Una de las piezas fundamentales de esta herramienta es la clase *WiimoteState*. Un objeto de esta clase contiene información del estado de todos los periféricos que componen el Wiimote: sus once botones, su acelerómetro de tres ejes, su cámara infrarroja y lo que posiblemente tenga conectado en su puerto de expansión están recogidos aquí. En la misma llamada al método que realiza la conexión, se realiza una primera lectura asíncrona, y siempre que se acabe con una se empieza con otra. Con los datos

recibidos en cada llamada (report) se actualiza el estado del Wiimote, de forma automática a través del método *ParseInput*. En definitiva, disponemos constantemente de un miembro *WiimoteState* que informa en todo momento que lo consultemos de lo que está ocurriendo en el mando. [13].

De esta forma, el usuario no tiene más que comprobar a su antojo el estado del Wiimote y actuar en consecuencia. A parte, se dispone de una serie de funcionalidades con las que el usuario puede:

- Conectarse y desconectarse del mando: usando los métodos *Connect* y *Disconnet*, respectivamente.

- Elegir el tipo de report con el que el Wiimote contesta: mediante *SetReportType*.

- Configurar el estado de los LEDs del mando: a través de *SetLEDs*.

- Activar o desactivar el motor de vibración: con el método *SetRumble*.

- Obtener el estado solicitándolo explícitamente: haciendo una llamada a *GetStatus.*

- Leer y escribir en los registros y memorias del mando (aunque la mayoría de las veces esto es necesario, por ejemplo, para inicializar extensiones o cámara IR se hace de forma automática): usando *ReadData* y *WriteReport.*

- Y liberar la memoria de ciertos recursos no gestionados: mediante *Dispose.*

- Cuando se quiere trabajar con múltiples Wiimotes entra en juego una pequeña clase llamada *WiimoteCollection* que contiene dos metodos: *FindAllWiimotes* y *WiimoteFound*.

2.3 Tratamientos quirúrgicos.

Para contribuir a reducir el dolor en la espalda, el médico puede utilizar primero un tratamiento no quirúrgico. Esto muchas veces implica limitar determinadas actividades y realizar fisioterapia. Es posible que se receten también medicamentos. Si estos tratamientos no mejoran la calidad de vida del paciente, el médico puede recomendar una cirugía. Con frecuencia, las patologías de la columna provocan dolor cuando los cambios óseos presionan la médula o los nervios. También pueden limitar el movimiento. El tratamiento varía según la enfermedad, pero algunas veces incluyen aparatos ortopédicos para la espalda y cirugía. En general en los tratamientos para la región lumbar se necesita acceder a la columna. En el caso de las hernias y opresión a los nervios se necesita restaurar la integridad de los discos intervertebrales o descomprimir el nervio dañado a diferencia del traumatismo [14].

2.3.1 Cirugía para fijación: Instrumentación mediante tornillos.

Esta técnica puede ser realizada mediante procedimientos abiertos o de mínima invasión. La fijación de columna mediante tornillos intrapediculares ha ganado bastante aceptación como un método confiable y efectivo para estabilización de la columna. Sin embargo, debido a las variaciones en la anatomía de cada paciente, el posicionamiento seguro y preciso de los tornillos puede ser una tarea complicada. La colocación errónea de los tornillos puede resultar en un daño a la medula espinal. Estas complicaciones pueden ser minimizadas si se le provee al cirujano con una orientación espacial exacta de cada pedículo previa a la inserción de los tornillos. Para ello, es utilizada la fluoroscopia, la cual, es una técnica de imagen que utiliza rayos X para obtener imágenes en tiempo real de las estructuras internas del cuerpo humano [5].

Si bien, provee imágenes en tiempo real, las vistas generadas representan vistas bidimensionales de una compleja estructura tridimensional, además, representa un peligro para la salud la utilización continua de radiación. La navegación espacial guiada por imagen puede ser utilizada en lugar de la fluoroscopia debido a que provee una vista de la anatomía de la columna y no

utiliza radiación. El tornillo intrapedicular entra desde una de las láminas que forman el arco vertebral y pasa a través del pedículo hasta el cuerpo vertebral como lo muestra la Fig. 6 [15]. Para la colocación adecuada es necesario planear la trayectoria del tornillo e identificar el punto de entrada, el ángulo de inserción y la profundidad de inserción.

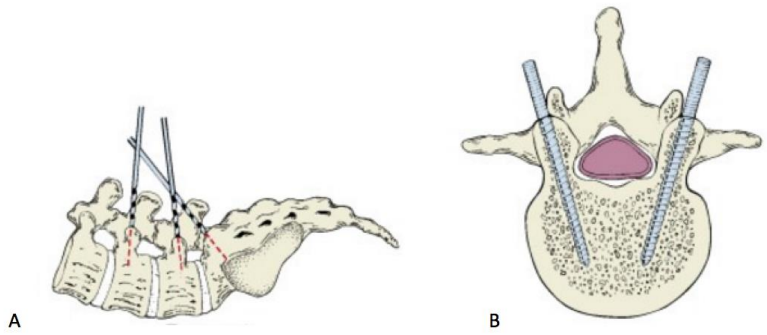


Fig. 6 Vistas lateral (A) y axial (B) de los tornillos dentro de la vértebra.

Sin embargo, es importante destacar que existen algunas características críticas a considerar durante la colocación de los tornillos intrapediculares, como lo son la anchura sagital y transversal de los pedículos, la longitud y ángulo del pedículo y la longitud de la cuerda que puede tener el tornillo tal y como se observa en la Fig. 7.

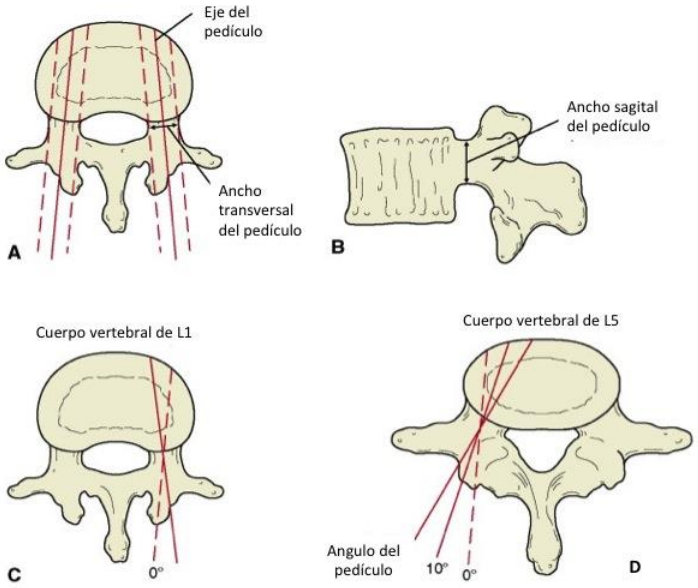


Fig. 7 La vista axial de las vértebras lumbares muestra el ancho del pedículo y la diferencia anatómica entre L1 y L5.

Estas dimensiones varían a lo largo de la columna vertebral entre las diferentes regiones e incluso entre las vértebras de una misma región. En promedio, el diámetro transversal medidos mediante tomografías y especímenes anatómicos reales es de aproximadamente 9mm en L1 e incrementa hasta 18 mm como máximo en L5. El ancho sagital en la región lumbar es relativamente constante, manteniéndose en un promedio entre 14 y 15 mm; El ángulo del eje del pedículo generalmente aumenta en la región lumbar con un promedio de 11º en L1, 15º en L3 y 20º en L5 [16].

Finalmente, el ángulo de inserción del tornillo es de importancia crítica en la medida en que la forma de las vértebras lumbares cambia drásticamente de L1 a L5. Debido a que la distancia entre pedículos es mayor en L5 y el diámetro anteroposterior del cuerpo vertebral es efectivamente menor en esa región, la longitud de la cuerda puede variar drásticamente con el ángulo de inserción. Es decir, si los tornillos son colocados perpendicularmente a la corteza posterior a través del eje 0º la profundidad promedio es de 45 mm mientras que en L5 es tan solo de 35 mm. Incrementando el ángulo de inserción a 10º o 15º, o el ángulo del eje del pedículo, se puede incrementar la distancia de corteza a corteza como máximo 5 mm en L1 (teniendo así 50 mm) y 15 mm en L5 (teniendo 50 mm) [16]. La Fig. 8 muestra el punto de entrada para la inserción de los tornillos en el pedículo se localiza en la intersección de una línea vertical tangencial al borde lateral de la apófisis articular y una línea horizontal que divide la apófisis transversa.

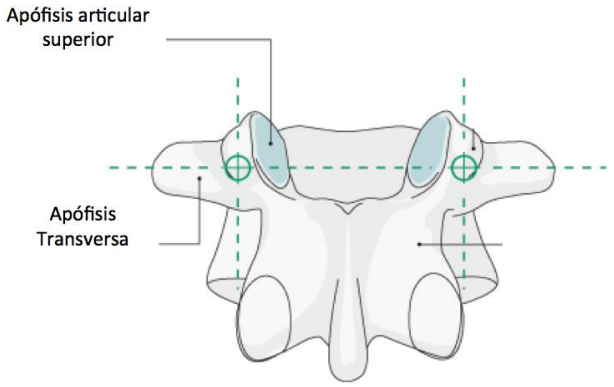


Fig. 8 Punto convencional de entrada para el posicionamiento de los tornillos en los pedículos representado por el circulo verde en la intersección de las dos líneas punteadas.

2.4 Estado del arte.

A continuación, se hace una revisión de algunos sistemas de seguimiento comerciales y aquellos no comerciales que utilizan mandos de Nintendo Wii.

2.4.1 Sistemas de seguimiento comerciales.

En párrafos anteriores se mencionó que existen distintas aplicaciones para los sistemas de seguimiento, desde aplicaciones militares hasta aplicaciones en el área de la medicina. En este apartado se menciona un sistema de tipo comercial cuya aplicación es utilizada para procedimientos de fijación d columna.

Brainlab Spinal Navigation.

Desarrollado por la compañía Brainlab ® es un sistema que permite el posicionamiento preciso de los tornillos intrapediculares en el pedículo de las vértebras y que, además, reduce drásticamente el uso de rayos x durante la cirugía. Utiliza cámaras infrarrojas para detectar marcadores pasivos que se encuentran en el instrumental. Ver Figura 9 [17].

Fig. 9 Sistema de tracking desarrollado por Brainlab

2.4.2 Sistemas de seguimiento que utilizan el Wiimote.

El Wiimote ha sido utilizado para diversas aplicaciones [18] tales como: Seguimiento de cabeza (Head Tracking), seguimiento de manos (Hand Tracking), sistema de seguimiento para personas que utilizan el sistema braille de lectura, seguimiento de vehículos aéreos no tripulados etc.

A continuación, se realiza una revisión de algunos sistemas de tracking que utilizan Wiimotes.

A Low-Cost System for Indoor Motion Tracking of Unmanned Aerial Vehicles

Trabajo desarrollado por estudiantes de la Escuela de Ingeniería e Informática (ECE) que se encuentra en París, Francia en el cual se describe un Sistema de seguimiento de movimiento de vehículos aéreos no tripulados. Utilizando al menos dos Wiimotes en configuración estereoscópica es posible calcular posición y orientación además se diseñó un circuito de marcadores activos colocados en el objetivo a seguir que constan de 7 leds infrarrojos [19]. Ver Figura 10.

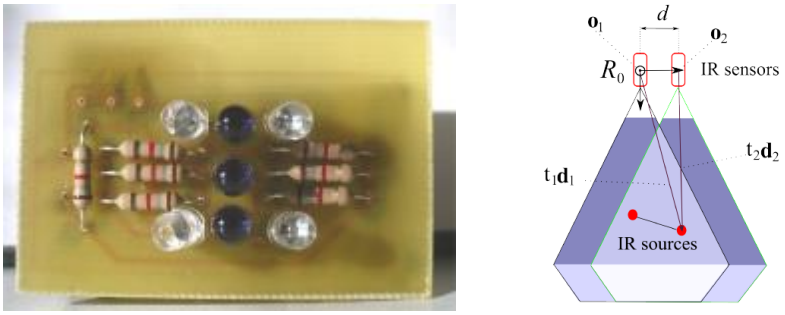


Fig. 10 Marcadores Infrarrojos (izq.). Configuración estereoscópica (der.).

Design and Implementation of Hand Tracking Interface using the Nintendo Wii Remote

En este trabajo de tesis se describe el diseño e implementación de un sistema de seguimiento para el reconocimiento de gestos de las manos, se utilizó una configuración estereoscópica, con

dos Wiimotes en paralelo. El sistema permite 6 grados de libertad completos. Trabajo realizado en la Universidad Cape Town, Sudáfrica [11]. Ver Figura 11.

Fig. 11. Marcadores Infrarrojos colocados en los dedos del usuario.

Nintendo Wii remote controllers for head posture measurement: accuracy, validity, and reliability of the infrared optical head tracker

Desarrollado en la Universidad Nacional de Seúl, Corea. Se presenta un sistema seguidor del movimiento de cabeza utilizando dos Wiimote en paralelo. El sistema utiliza un marcador compuesto por cuatro LEDs infrarrojos para la medición del ángulo de postura de la cabeza [20]. Ver Figura 12.

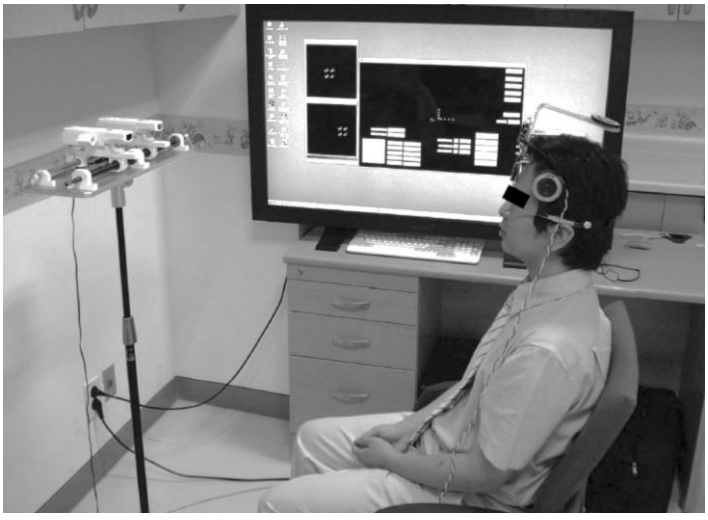


Fig.12 Seguidor de movimiento de cabeza.

Nintendo Wii remotes provide a reconfigurable tool-changing unit with an automatic calibration capability.

Desarrollado en la Universidad de KwaZulu-Nata en Sudáfrica. Un sistema de posicionamiento económico fue producido usando controles remotos de Nintendo Wii. Este artículo presenta el desarrollo, implementación y pruebas de un sistema de seguimiento 3D en tiempo real para detectar alteraciones en la posición del husillo de un robot Fanuc al que estaba suministrando las herramientas [21]. Ver Figura 13.

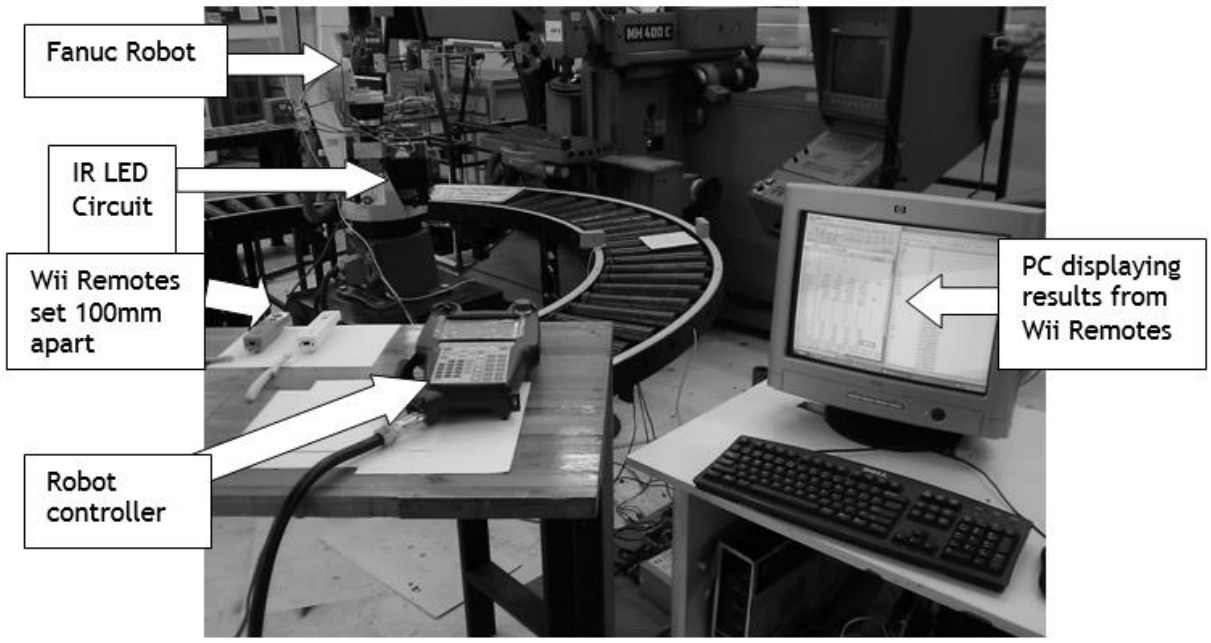


Fig. 13 Wiimotes como sistema de seguimiento de un robot Fanuc.

Wii Remote-enhanced Hand-Computer Interaction for 3D Medical Image Analysis

Desarrollado en la Universidad de Naples, Italia. El sistema utiliza Wiimotes para manipular imágenes médicas dentro de un entorno virtual mediante el reconocimiento de gestos de las manos [22]. Ver Figura 14

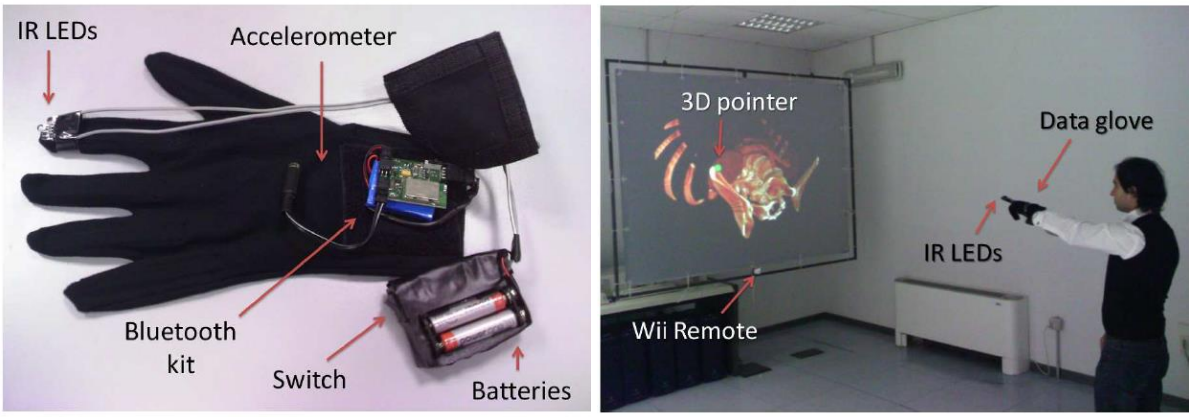


Fig. 14 Sistema de seguimiento de mano.

En la Tabla 1 se presenta un resumen y comparación de diferentes sistemas que utilizan controles de Nintendo Wii para el seguimiento de LEDs infrarrojos, así como sus diferentes características.

Tabla 1. Comparación de sistemas de seguimiento que utilizan el Wiimote.

Aplicación	No. de Referencia	Costo	**Precisión**	No. de Cámaras	**Modelo de cámara**	No. de LEDs infrarrojos
Head Tracking	[3]	Alto	6-DOF	4	Estacionario	4
Hand Tracking	[11]	Medio	6-DOF	2	Estacionario	2
Hand Tracking	[12]	Medio	6-DOF	2	Estacionario	2
Tracking of UAV	[19]	Medio	6-DOF	2	Estacionario	2
Head Tracking	[20]	Medio	3-DOF	2	Estacionario	4
Tracking for rehabilitation	[23]	Bajo	1-DOF	1	Estacionario	1
Head tracking	[24]	Medio	2/3-DOF	2	Estacionario	4
Head Tracking	[25]	Bajo	6-DOF	1	Estacionario	8
Human computer interaction	[26]	Bajo	2/3-DOF	1	Estacionario	1
Finger Tracking	[27]	Bajo	1-DOF	1	Estacionario	1
Wiimote Projects	[28]	Bajo	2/3-DOF	1	Estacionario	2
Wiimote Tracking	[29]	Bajo	6-DOF	1	Movimiento	4

Alto 100 USD, Medio 50 USD, Bajo 25 USD.

Capítulo 3

Desarrollo

3.1 Propuesta

La propuesta consiste en desarrollar un sistema que permita registrar el movimiento de marcadores infrarrojos. Se busca utilizar el sistema para el seguimiento del instrumental utilizado en un entrenador de cirugías de fijación de columna mediante tornillos intrapediculares. El objetivo es proporcionar una herramienta de registro al mismo tiempo que permite desarrollar habilidades como: ubicación espacial para interpretar imágenes en 2D de un espacio 3D, coordinación mano-ojo. El diagrama general de la solución propuesta, se muestra en la Fig. 15.

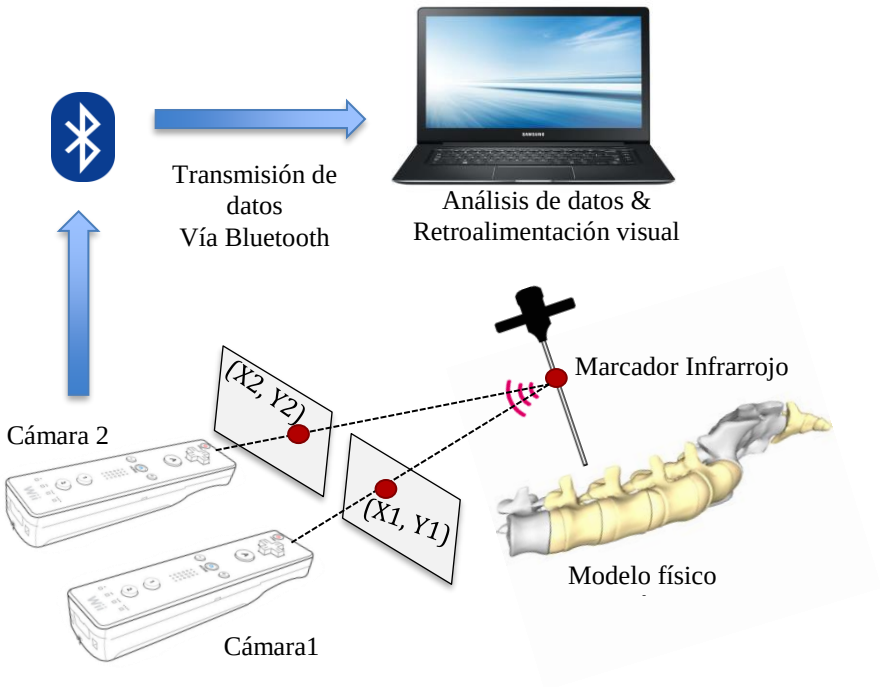


Fig. 15 Diagrama General de la solución propuesta. El alcance del proyecto abarca el sistema de tracking y la interfaz de usuario.

Como se puede observar en la imagen anterior, la información del marcador infrarrojo colocado en el instrumental, es sensada por las cámaras infrarrojas de los Wiimotes. Esta información es enviada a la computadora vía Bluetooth, que se encuentra integrado en los controles. Una vez en la computadora, los datos son utilizados para obtener la posición del marcador en el espacio, así como el ángulo en el que se encuentra el instrumental, el cual, es un parámetro importante para las cirugías de columna. Es importante mencionar que la configuración de la imagen anterior fue una de las utilizadas y no necesariamente la que se utilizó al final del proyecto. En la Fig. 16 se muestra el diagrama a bloques del proyecto.

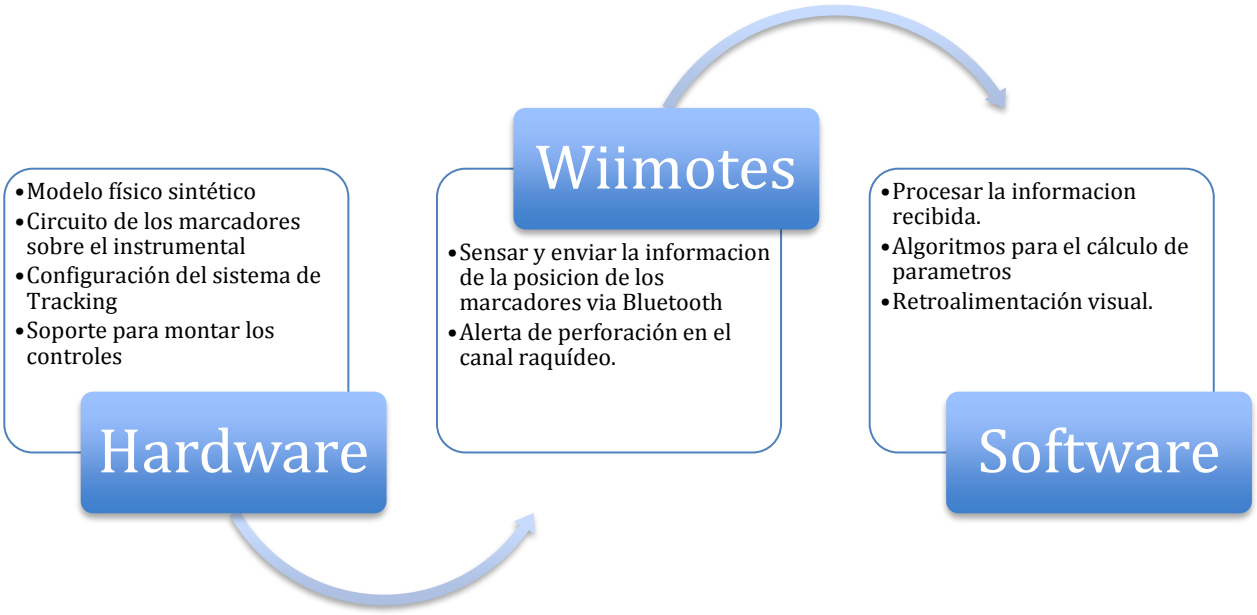


Fig. 16 Diagrama a bloques de la solución propuesta.

3.2 Hardware

3.2.1 Modelo Físico Sintético.

El sistema de seguimiento (*tracking*) es utilizado sobre un modelo de columna lumbar desarrollado en el laboratorio de trabajo. El modelo se realizó con resinas y poliuretanos, además, está basado en el modelo comercial A74 anatómico de la región lumbar de tamaño real para un adulto sin patologías de la marca Scientifi.

3.2.2 Diodo Emisor de Luz (LED) Infrarrojo

El sistema de LEDs es construido lo más simple posible para maximizar su usabilidad, la vida de la batería y más allá de la estética, que el sistema no interfiera con el manejo del instrumental. Se utilizaron diversos diodos emisores de luz de diversas marcas y tamaños e incluso colores, para determinar cuál resultaba óptimo para su uso. Se encontró que el diodo emisor de luz infrarroja de la marca Steren de 5mm de diámetro a una longitud de onda de 940 nm, 1.3 V típicos en polarización directa, 1.7 V máximos y a un ángulo de 12º permite una detección a mayores distancias. El brillo de un LED se mide en mili watts (mW). A pesar de que la luminiscencia de un LED depende de la cantidad de energía con que se alimente, la mayoría de los LED producen 20 mW de luz en su punto máximo y cerca de 1 mW de luz a un nivel de operación promedio. En la Fig. 17 se muestra el diseño del circuito.

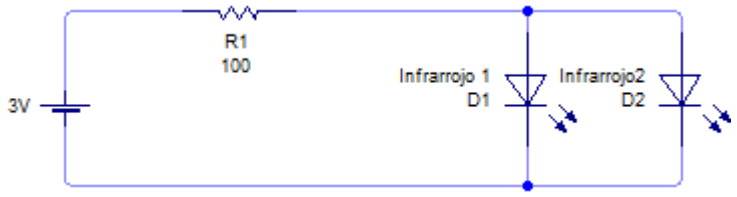


Fig. 17 Circuito para los marcadores infrarrojos

El circuito es construido con 2 LEDs infrarrojos alimentados por una batería de 3V, una resistencia para limitar la corriente, es importante aclarar que la longitud de onda de los LEDs infrarrojos es invisible para el ojo humano.

3.2.3 Configuración del sistema de seguimiento y cálculo de coordenadas.

Para el sistema de *tracking* del instrumental, el mando del Nintendo Wii resultó ser una opción económica. Es capaz de proveer en tiempo real un sistema de seguimiento a un costo mucho menor que los sistemas comerciales. Diversas configuraciones fueron probadas para el sistema, cada una con sus ventajas y desventajas. De acuerdo con los Grados de Libertad (GDL) que puede manejar el sistema, al menos 3 configuraciones pueden ser utilizadas.

3.2.3.1 Configuraciones con 2 Wiimotes.

La primera configuración utiliza dos Wiimotes y únicamente un marcador infrarrojo. Es posible utilizar una configuración ortogonal o una configuración estereoscópica.

i. Configuración Ortogonal.

En esta configuración dos Wiimotes se encuentran a 90º uno del otro, además cada mando debe observar 2 LEDs en su propio plano de imagen, por lo que se utilizan 4 LEDs en total como lo muestra la Fig. 18. Cada mando reporta las coordenadas (x, y) de cada LED por lo que es posible obtener la distancia de los Wiimotes al marcador infrarrojo utilizando uno de los mandos como cámara complementaria.

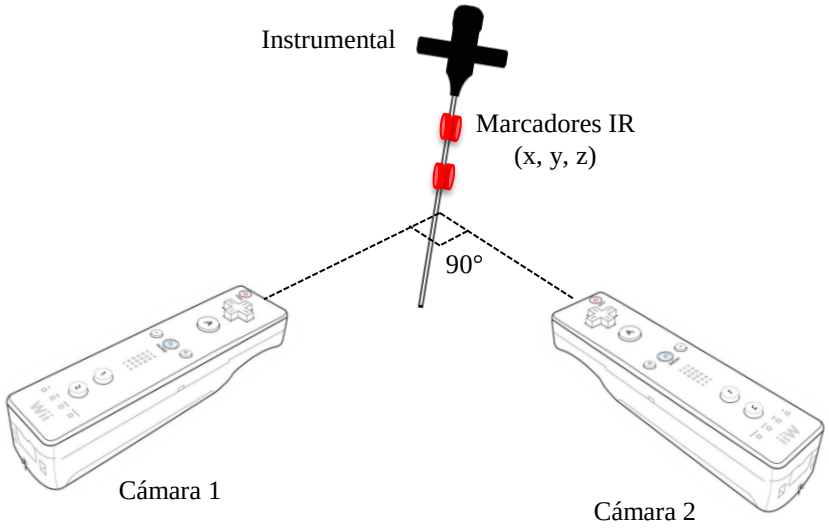


Fig. 18 Configuración ortogonal.

El Wiimote 1 reporta las coordenadas "x" e "y" del instrumental, mientras que el Wiimote 2 reporta la coordenada de profundidad "z" realizando las asignaciones de tal forma que es posible obtener las coordenadas globales del instrumental (x, y, z).

$$\begin{aligned} x &= x_1 \\ y &= y_{1=}\, y_2 \\ z &= x_2 \end{aligned} \tag{1}$$

Si bien, los cálculos necesarios para obtener la posición 3D son relativamente sencillos, la carga computacional es mayor debido a que cada Wiimote debe calcular la posición de 2 marcadores infrarrojos en tiempo real. Una ventaja de esta configuración son los 3 grados de libertad que maneja.

ii. Configuración estereoscópica.

Esta configuración es una de las más complicadas debido a que requiere el uso de procesos matemáticos complejos, lo que se traduce en una carga computacional mayor que las demás configuraciones. Se utilizan dos Wiimotes paralelos a una distancia fija entre sí, y mediante técnicas de estereoscopia y la intersección de dos rayos en el espacio es posible calcular la posición 3D de un marcador. La Fig. 19 muestra esta configuración.

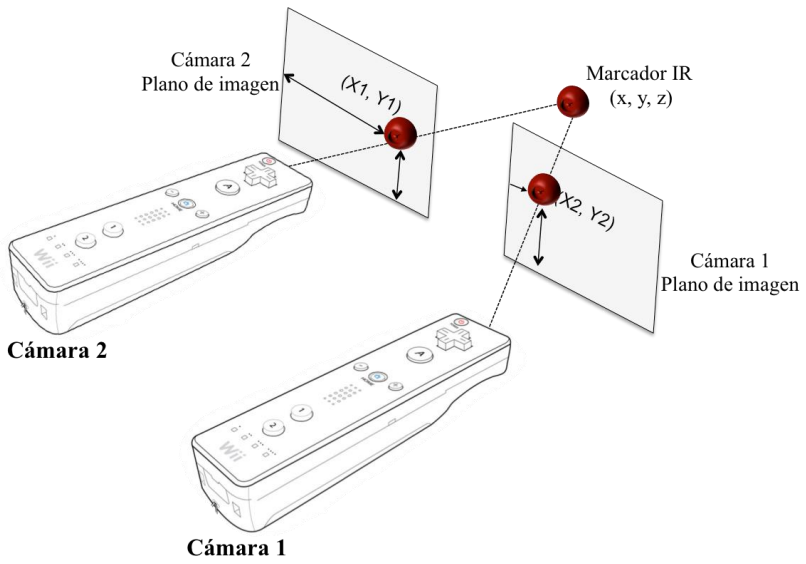


Fig. 19 Configuración estereoscópica

.

La configuración anterior puede ser utilizada con uno o dos marcadores infrarrojos implicando con ello, que la complejidad para calcular la posición de cada marcador se incrementa considerablemente. El marcador infrarrojo se proyecta en el plano de imagen de cada cámara

como un rayo y calculando su intersección es posible obtener la posición, sin embargo, en la realidad estos rayos nunca se intersectan por lo que se utiliza una aproximación. Los cálculos necesarios se muestran en el Apéndice A. Una de las ventajas de esta configuración es que permite tener un sistema de 6 grados completos de libertad. Sin embargo, debido a las limitaciones físicas de la configuración, que ya se mencionaron anteriormente, el cálculo de la posición es una tan solo aproximación, por lo que se trabajó con la siguiente configuración en busca de un mejor resultado.

3.2.3.2 Configuraciones con un Wiimote.

i. Configuración con un Wiimote y 2 LEDs.

Esta configuración utiliza únicamente un Wiimote y 2 LEDs. La configuración que se observa en la Fig. 20 permite el cálculo de la coordenada "z" así como también el cálculo del ángulo del instrumental utilizando matemáticas relativamente sencillas.

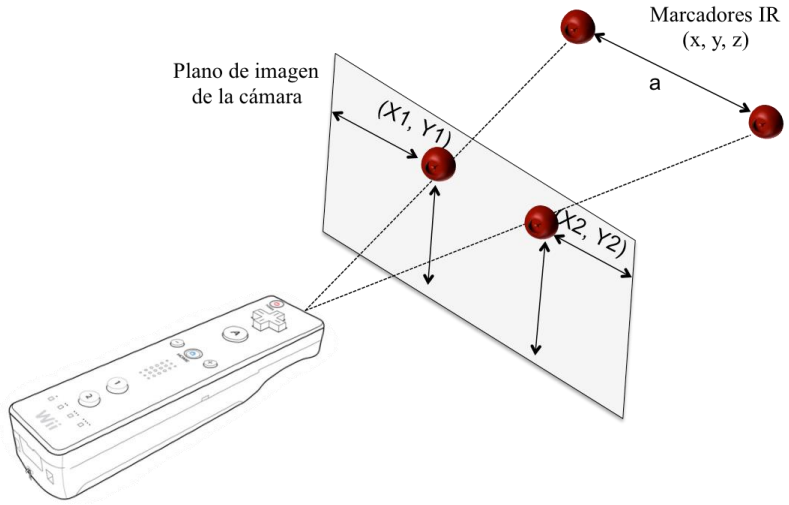


Fig. 20 Configuración con un Wiimote y 2 LEDs.

Esta configuración permite 4 grados de libertad, que incluye los 3 grados de libertad de traslación, así como el giro angular del objeto. La configuración anterior puede ser utilizada

con un único LED, sin embargo, es bastante simple y no es posible calcular la posición, distancia o ángulo del instrumental con ella, es una configuración que permite 2 grados de libertad es por ello que no se revisó en este proyecto. Para calcular la coordenada "z" que hace referencia a la distancia entre el Wiimote y el marcador infrarrojo, se utilizaron dos métodos que son descritos a continuación.

a) Por triángulos similares y distancia focal.

El primero de ellos es utilizando el teorema de Pitágoras, análisis de triángulos similares mediante los campos de visión y la distancia focal "b" de la cámara infrarroja. La distancia focal se puede calcular mediante la siguiente formula, en donde r_h es la resolución horizontal y a_h el ángulo de visión horizontal:

$$b = \frac{0.5(r_h)}{\tan\left(\frac{\pi(a_h)}{180}\right)} = \frac{0.5(1016)}{\tan\left(\frac{\pi(40.6)}{180}\right)} = 1369.41 \qquad (2)$$

La posición de cada uno de los diodos infrarrojos es reportada por sus coordenadas (x_1, y_1) y (x_2, y_2), la distancia relativa "a" en pixeles entre los diodos puede ser calculada mediante el teorema de Pitágoras:

$$a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (3)$$

Una vez conocida la distancia en pixeles entre los LEDs, la longitud focal de la cámara, y la distancia real en centímetros entre los LEDs "c", la coordenada "z" entre los Wiimotes y el marcador infrarrojo, puede ser conocida utilizando un análisis de triángulos similares como el que se muestra en la Figura 21:

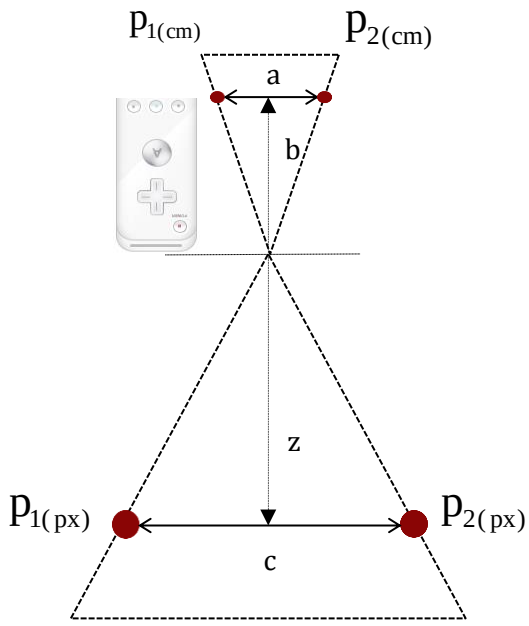


Fig. 21 Calculo de las coordenadas x, y, z por triángulos similares.

Por triángulos similares:

$$\frac{z}{b} = \frac{c}{a} \rightarrow z = \frac{cb}{a} \tag{4}$$

La distancia entre los LEDs en pixeles, varia respecto a la distancia, si los diodos se encuentran cerca de la cámara, la distancia es mayor que cuando se encuentran lejos, por lo que el parámetro "a" será variable siempre. Si la distancia real entre los diodos es de 3 cm, el cálculo de la coordenada z quedará en función del parámetro a, que se calcula automáticamente en el software, de tal forma que:

$$z = \frac{3(1369.41)}{a} \rightarrow \frac{px(cm)}{px} \tag{5}$$

Es posible observar que del análisis dimensional el resultado final se encuentra en centímetros. Además, para el cálculo de las coordenadas *x, y* en centímetros se realizan los siguientes cálculos a partir de la Fig. 29:

$$x = \frac{z\left(p_{1x} + p_{2x}\right)}{2b}$$
$$y = \frac{z\left(p_{1y} + p_{2y}\right)}{2b} \qquad (6)$$

b) Utilizando el teorema de Pitágoras y el campo de visión angular por pixel.

En la Fig. 22 se muestra el triángulo que se forma entre las cámaras y los dos marcadores. Los diodos se encuentran entre sí a una distancia física “c” en cm y a una distancia “r” en pixeles.

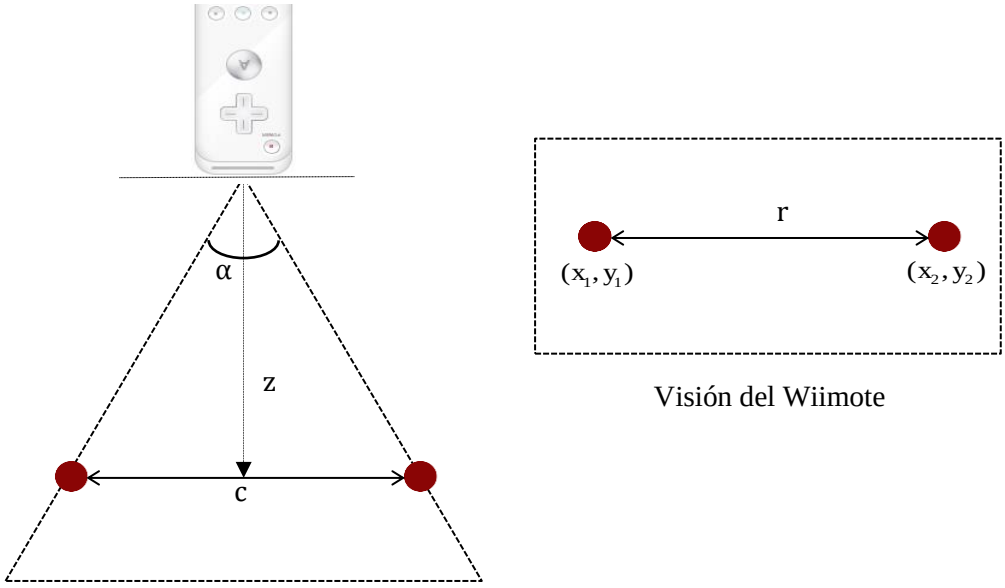


Fig. 22 Calculo de la coordenada z utilizando el campo de visión angular por pixel.

Dados los campos de visión de la cámara en las direcciones horizontal (H_{fov}) y vertical (V_{fov}) y la resolución de la cámara en las mismas direcciones (H_r, V_r) es posible determinar el campo de visión angular por pixel θ:

$$\theta = \frac{\left(\frac{H_{fov}}{H_r} + \frac{V_{fov}}{V_r}\right)}{2} = \frac{\left(\frac{40.6}{1016} + \frac{30.7}{760}\right)}{2} = 0.04017 \qquad (7)$$

La distancia r entre los marcadores puede ser calculada mediante sus coordenadas individuales mediante el teorema de Pitágoras:

$$r = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2} \tag{8}$$

Así, el ángulo entre los LEDs y la cámara del Wiimote está dado por:

$$\alpha = \frac{r\theta}{2} = \frac{\left(\frac{H_{fov}}{H_r} + \frac{V_{fov}}{V_r}\right)\left(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right)}{4} \tag{9}$$

por lo tanto, la coordenada "z" puede ser calculada mediante:

$$z = \frac{c}{2\tan(\alpha)} = \frac{c}{2\tan\left(\frac{\left(\frac{H_{fov}}{H_r} + \frac{V_{fov}}{V_r}\right)\left(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right)}{4}\right)} \tag{10}$$

3.2.3.3 Elección de la configuración final del sistema de seguimiento.

En la Tabla 2 se muestra una comparación de las configuraciones revisadas, ventajas y desventajas, de tal forma que permita utilizar la configuración más apropiada.

Se decidió utilizar una combinación de dos de las configuraciones anteriormente explicadas. Se utilizaron dos Wiimotes en configuración ortogonal e independientes uno del otro. Uno de los controles ve en su plano de imagen la vista axial de la vértebra. El segundo control ve en su plano de imagen la vista lateral de la región lumbar de la columna vertebral, por lo que en la interfaz se despliega una línea que corresponde al movimiento del instrumental por cada plano.

Tabla 2. Comparación de configuraciones para el sistema de seguimiento.

Configuración	No. Wiimotes	No. LEDs	DOF	Precisión	Costo $	Complejidad de Cálculos	Parámetros
Ortogonal	2	4	3	Alta	Alto	Media	Ángulo, Profundidad, Coord. z
Estereoscópica	2	1/2	3/6	Media	Alto	Alta	Coord. z
1 Wiimote 2 LEDs	1	2	3	Alta	Bajo	Media	Ángulo, profundidad, Coord. z
1 Wiimote 1 LED	1	1	2	Baja	Bajo	Baja	x

Alto 100 USD, Medio 50 USD, Bajo 25 USD.

Es decir, las coordenadas del instrumental no se calculan utilizando una cámara complementaria, cada Wiimote calcula las coordenadas del instrumental respecto a si mismo, de tal modo que el campo de visión de cada cámara es independiente y el espacio de trabajo no se ve reducido por el área en la que se intersectan los campos de visión de los dos Wiimotes. Se utilizaron 2 marcadores para cada cámara en su propio plano de imagen. Esta configuración tiene las siguientes ventajas:

- Permite calcular la coordenada z.
- Permite calcular el ángulo del instrumental.
- Permite calcular la profundidad del instrumental.
- Permite un mayor volumen de trabajo.
- 3 Grados de Libertad.

Algunas de las restricciones de la configuración estereoscópica son:

- Un cambio mínimo en la orientación de un Wiimote, puede llevar a errores en los cálculos.
- La configuración con un solo LED no permite el cálculo del ángulo del instrumental.

3.2.3.4 Soporte y montura para los Wiimotes.

Una vez seleccionada la configuración de los Wiimotes, es necesario realizar un soporte para montar los controles. Para ello se utilizó un soporte de tipo tripié de la marca Insignia de 5.5 pulgadas y expandible hasta 8 pulgadas. El soporte se muestra en la figura 23.

Fig. 23 Soporte para cámara tipo tripié.

Además, fue necesario diseñar un adaptador como el que se muestra en la Figura 24 para montar el Wiimote en el tripié. Para ello se utilizó el software SolidWorks y una impresora 3D.

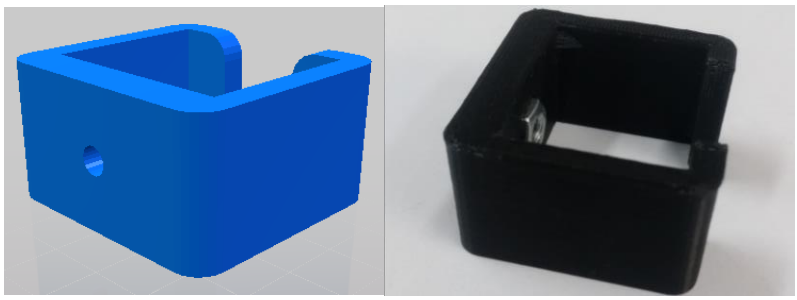


Fig. 24 Diseño del adaptador.

Cada Wiimote cuenta con su propio adaptador y tripié. Ver Figura 25

Fig. 25 Wiimote montado en tripié.

3.3 Conexión y envió de información.

3.3.1 Conexión Bluetooth de los Wiimotes.

Toda la comunicación con el Wiimote se realiza a través de Bluetooth. Este es proporcionado por un chip de Broadcom BCM2042. Todo el Wiimote está construido en torno a este chip ya que es la única manera para el Wiimote de comunicarse con el mundo exterior. El chip está diseñado para ser utilizado con Dispositivo de Interfaz Humana Bluetooth (HID). Sin embargo, el Wiimote no se emparejará y se comunicara de manera satisfactoria con todos los dispositivos Bluetooth que existen. Se utilizó el *"Toshiba Bluetooth Stack"* que permite conectar dispositivos Bluetooth a la computadora

Se utilizó un adaptador Bluetooth genérico como el que se muestra en la Fig. 26 en una computadora con Windows 7.

Fig. 26 Adaptador Bluetooth.

Uno de los principales problemas al utilizar los mandos del Nintendo Wii es conectarlos con la computadora. Es por ello que en el Apéndice B se describen los pasos necesarios para conectar satisfactoriamente el Wiimote a la computadora.

3.3.2 Envió de información.

El Protocolo de Descubrimiento de Servicio (SDP por sus siglas en inglés) permite la detección automática de dispositivos Bluetooth. Utilizando la información de dispositivo, la conexión mediante uno o más dispositivos Bluetooth puede ser realizada. Como todos los dispositivos Bluetooth HID, el Wiimote reporta su propio bloque descriptor HID con información única por cada Wiimote. Cuando el mando es consultado por el SDP reporta la información de la Tabla 3. Es importante mencionar que la librería buscara dispositivos con estos atributos, por lo que es necesario modificarla en caso de tener valores diferentes de VID y PID.

Tabla 3. Información del mando Wiimote

Atributo	Información
Nombre	Nintendo RVL-CNT-01
Vendor ID (VID)	0x57e
Product ID (PID)	0x0306

Los valores en la tabla anterior son únicos para el Wiimote y lo identifican entre todos los demás dispositivos Bluetooth. Si bien, la información presentada en la Tabla 3 es para la primera generación de Wiimotes, los dispositivos con terminación –TR tienen sus propios atributos únicos que pueden ser consultados en Windows 7 en la sección de características de dispositivos Bluetooth. Una vez conectados los mandos, se puede obtener información acerca de la posición de los marcadores infrarrojos, sin embargo, es necesario conocer las características propias de cada uno de los controles.

3.4 Software.

Se utilizó la herramienta de software Microsoft Visual Studio 2015 versión 14.0.24720.00 y la versión de Microsoft.NET Framework 4.6.01055. La computadora utilizada cuenta con Windows 7 y se utilizó la librería WiimoteLib versión 1.7.0.0 para comunicar el Wiimote con la computadora. Para el desarrollo de la interfaz gráfica de usuario se trabajó con el marco Windows Presentation Foundation (WPF) debido a que ofrece una amplia infraestructura y potencia gráfica con lo que es posible desarrollar aplicaciones visualmente atractivas con facilidad de interacción. La plataforma de desarrollo WPF usa el lenguaje de marcado de aplicaciones extensibles XAML para proporcionar un modelo declarativo para la programación de aplicaciones.

El código en C# es la interfaz visual con el usuario, y en él, se realizan todos los cálculos de parámetros, así como también el almacenamiento de registros. La siguiente figura es una captura de pantalla de la ventana principal de la interfaz gráfica de usuario.

3.4.1 Ventana principal del sistema ortogonal

De la figura 27 se pueden apreciar paneles diferentes: registro de métricas, vistas axial, anteroposterior y lateral, además de una ventana adicional llamada “instrumental tracking” que permite observar el movimiento de los marcadores infrarrojos para asegurar que son visibles.

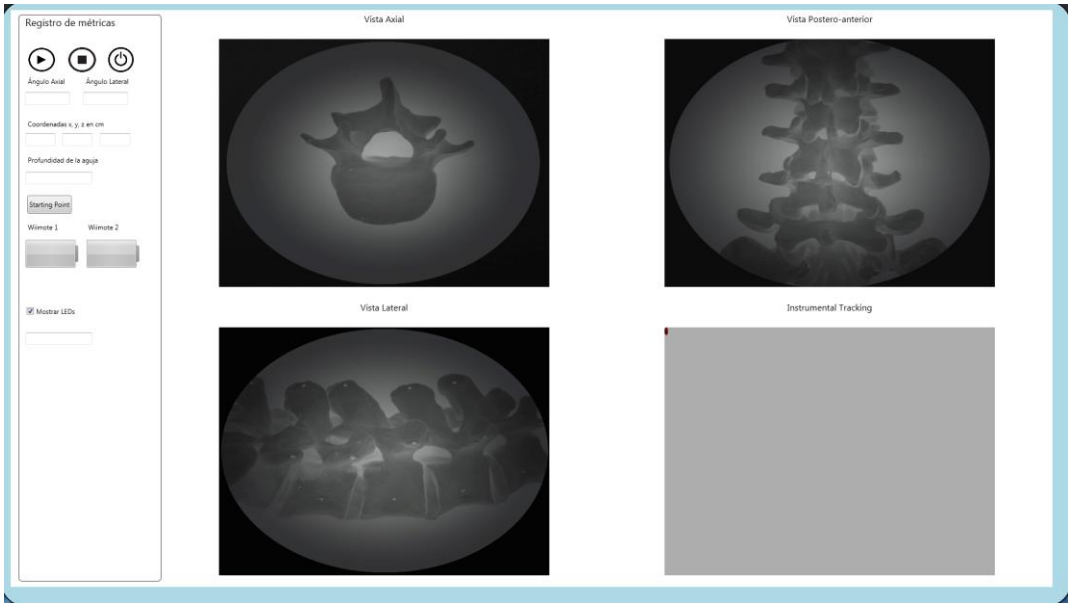


Fig. 27 Ventana principal de la interfaz de usuario.

3.4.2 Ventana de Inicio.

En la figura 28 se muestra la ventana de inicio del programa. Se presenta un menú desplegable para elegir un usuario anteriormente registrado, o un hipervínculo para el registro de un nuevo usuario. Una vez realizado el registro o seleccionado el usuario, se presiona el botón iniciar para pasar a la siguiente ventana.

Fig. 28 Ventana de inicio de la interfaz de usuario.

3.4.3 Ventana de registro de nuevo usuario.

Para registrar un nuevo usuario es necesario llenar todos los campos, de lo contrario el proceso de registro no se puede llevar acabo. La figura 29 muestra la ventana de registro de usuario.

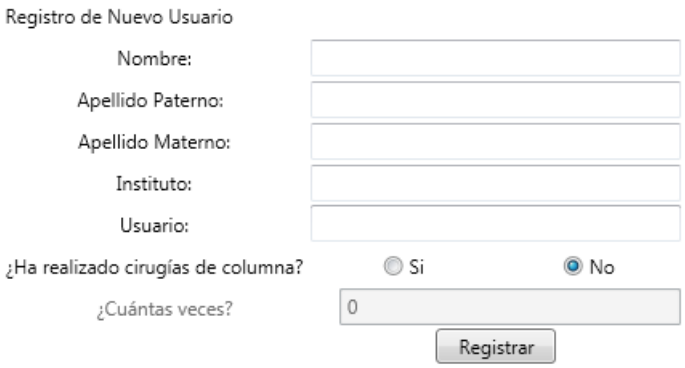
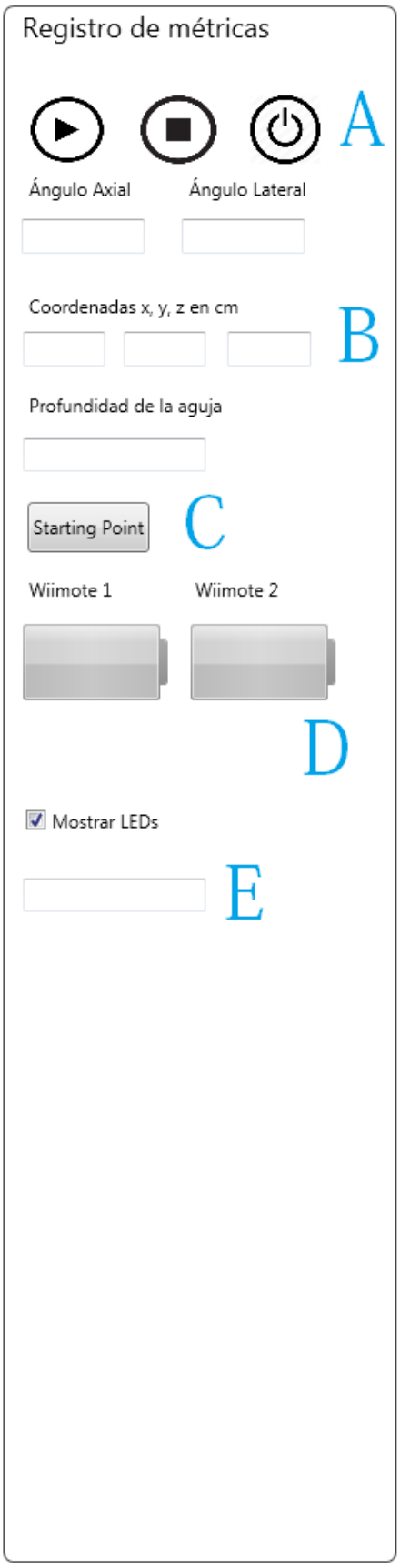


Fig. 29 Ventana de inicio registro de nuevo usuario.

3.4.4 Panel Registro de parámetros.

De la figura 28 se pueden apreciar paneles diferentes: registro de parámetros, vistas axial, anteroposterior y lateral, además de una ventana adicional llamada “instrumental tracking” que permite observar el movimiento de los marcadores infrarrojos para asegurar que son visibles.

Dentro del panel registro de métricas es posible observar diferentes botonerías, parámetros y elementos visuales. A continuación, se describen brevemente los elementos contenidos dentro de dicho panel.



A. Botonería de control.

Es posible observar una botonería de control de inicio, pausa o salir.

B. Registro de parámetros.

En este panel se observan todos los parámetros calculados. Se observan las coordenadas del instrumental respecto al Wiimote principal que es el que se encuentra en la vista axial, se observan también los ángulos lateral y axial, así como también la profundidad de inserción del instrumental.

C. *Starting Point.*

Una vez definido el punto de entrada del instrumental en el pedículo de la vértebra se presiona el botón *starting point* para guardar esas coordenadas y poder proceder a calcular la profundidad del instrumental a partir del punto de entrada.

D. Estado de batería de Wiimotes.

Se muestra gráficamente el porcentaje de batería que tiene cada Wiimote.

E. Cronometro.

Se implementó un contador de tiempo que registra cuánto tarda el usuario en realizar la tarea.

3.4.5. Calculo de parámetros.

i. Calculo del ángulo del instrumental

Utilizando la configuración ortogonal con los Wiimotes independientes se calculó el ángulo del instrumental para cada Wiimote. La Fig. 30 muestra el análisis necesario para obtener el ángulo del instrumental.

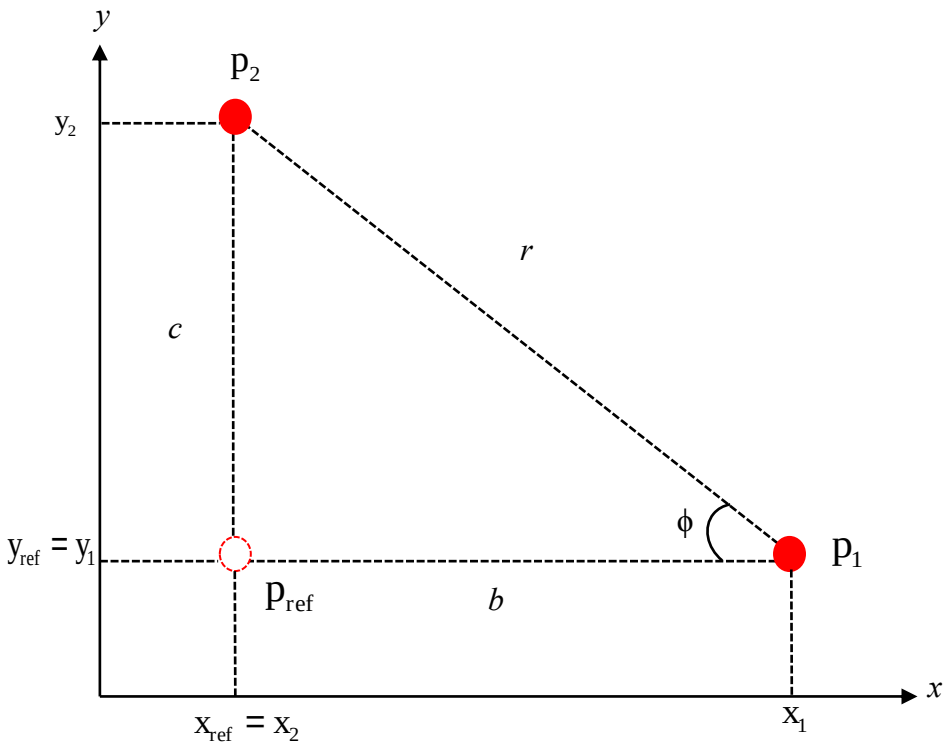


Fig. 30 Calculo del ángulo del instrumental

En la figura anterior se pueden observar los dos marcadores infrarrojos separados por una distancia r en pixeles. Además, se observa un punto de referencia, que está construido a partir de las siguientes coordenadas: $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ y $p_{ref} = (x_{ref}, y_{ref}) = (x_2, y_1)$.

El punto de referencia está construido a partir de la coordenada en *y* del punto 1 y la coordenada en *x* del punto 2, de tal manera que siempre que las coordenadas del punto 1 y el punto 2 cambien, las coordenadas del punto de referencia cambiaran de acuerdo a ellas.

Ahora bien, es posible realizar el cálculo del ángulo ϕ utilizando la tangente y las distancias c y b que se encuentran en pixeles mediante los siguientes cálculos:

$$c = \left| y_2 - y_{ref} \right| \tag{11}$$

$$b = \left| x_{ref} - x_1 \right| \tag{12}$$

$$\phi = \tan^{-1}\left(\frac{c}{d} \right) \tag{13}$$

ii. Calculo de la profundidad del instrumental

Una vez conocida la distancia en pixeles y en centímetros entre los marcadores, se puede conocer la longitud del instrumental en pixeles mediante una simple regla de tres, esto es posible debido a que, se presenta un comportamiento lineal en relación a la distancia y los pixeles vistos por las cámaras. De tal modo que:

$$inst_{px} = \frac{\left(LED_{px} \right)\left(inst_{px} \right)}{LED_{cm}} \tag{14}$$

donde la longitud en pixeles del instrumental ($inst_{px}$) es igual a la distancia en pixeles entre los marcadores (LED_{px}) multiplicado por la longitud en centímetros del instrumental ($inst_{px}$) dividido entre la distancia en centímetros entre los marcadores (LED_{cm}). Del análisis dimensional es posible observar que el resultado final se encuentra en pixeles.

Esto permite calcular la posición de la punta del instrumental mediante ciertas consideraciones. La posición de la punta del instrumental puede ser conocida mediante un análisis de la Figura 31.

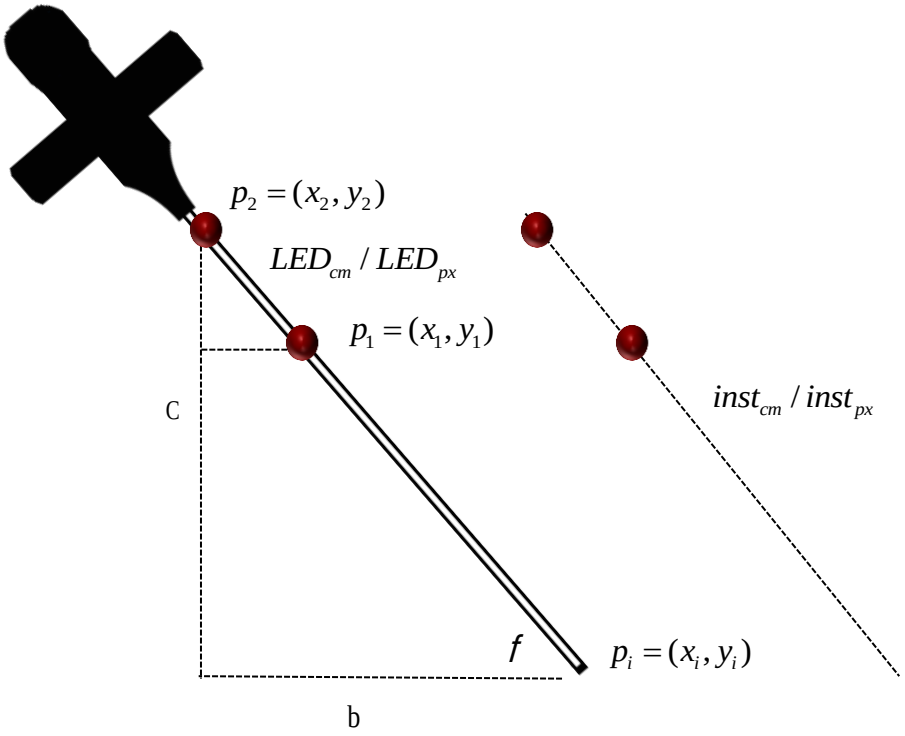


Fig. 31 Calculo de la longitud del instrumental

Es necesario conocer las distancias b y c que se calculan mediante:

$$b = inst_{px}\left(sen(\phi)\right) \tag{15}$$

$$c = inst_{px}\left(\cos(\phi)\right) \tag{16}$$

Una vez conocidas las distancias b y c se puede calcular la posición de $p_i = (x_i, y_i)$, de tal forma que:

$$x_i = \begin{cases} x_2 + b & \forall\left(x_1 < x_2\right) \\ x_2 - b & \forall\left(x_1 \geq x_2\right) \end{cases} \tag{17}$$

$$y_i = x_2 + c \tag{18}$$

Los cálculos anteriores sirven principalmente para conocer la longitud extracorpórea del instrumental, para posteriormente calcular la longitud intracorpórea del instrumental, en otras palabras, la parte del instrumental que se introduce en las vértebras durante el procedimiento.

Ya que se conoce la posición de la punta del instrumental en el espacio se procede a calcular la longitud del instrumental dentro de la vértebra, esta longitud es el parámetro profundidad. Para calcular la profundidad se utiliza el mismo análisis de la Fig. 32, en donde se tendrán dos puntos, el punto de inserción que corresponde a la punta del instrumental antes de entrar en la vértebra, se guardan las coordenadas de dicho punto y se fijan como el punto de entrada y un segundo punto que corresponde a la punta del instrumental moviéndose dentro del pedículo de la vértebra.

3.4.6 Alerta por invasión al conducto raquídeo.

Si bien, el entero propósito del sistema es desarrollar un sistema de tracking, se pretende utilizar el sistema para proporcionar una herramienta de práctica para la simulación de cirugías de fijación de columna, sin embargo, las complicaciones de introducir un tornillo intrapedicular en el conducto raquídeo pueden ser fatales. Es por ello que se implementó un algoritmo que permita alertar al usuario cuando esto sucede. Una vez que el instrumental se encuentre en el canal raquídeo ambos mandos vibran generando así una alerta. Para ello se implementó un algoritmo conocido como *point-in-polygon*. En geometría computacional, este problema pregunta si un punto dado en un plano se encuentra dentro, afuera, o en los límites de un polígono. Para el presente proyecto se utilizó un algoritmo llamado *ray casting*, el cual consiste en probar cuantas veces un rayo, que comienza desde el punto a probar y va hacia cualquier dirección fija, intersecta el borde del polígono [30].

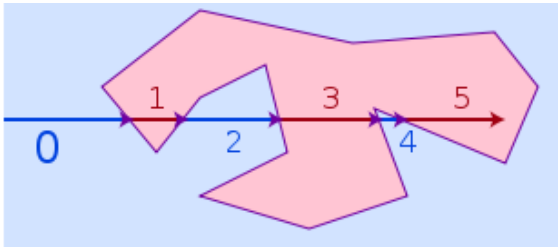


Fig. 32 Representación gráfica del algoritmo *ray casting*

De la figura 32 se deduce que si un punto se encuentra fuera del polígono el rayo intersecta el borde un numero par de veces. De lo contrario, si el punto se encuentra dentro del polígono, el

rayo intersecta el borde un número impar de veces [30]. Se procedió a definir un polígono construido a partir de puntos conocidos con coordenadas (x, y) sobre la imagen de la vértebra en el plano axial que corresponde al área del conducto raquídeo, de tal modo que sea posible aplicar el algoritmo sobre el polígono. Así, cuando la punta del instrumental, se encuentra dentro del polígono, ambos controles comienzan a vibrar

3.4.7 Flujo del programa.

La figura muestra los pasos que seguirá un usuario para realizar una tarea.

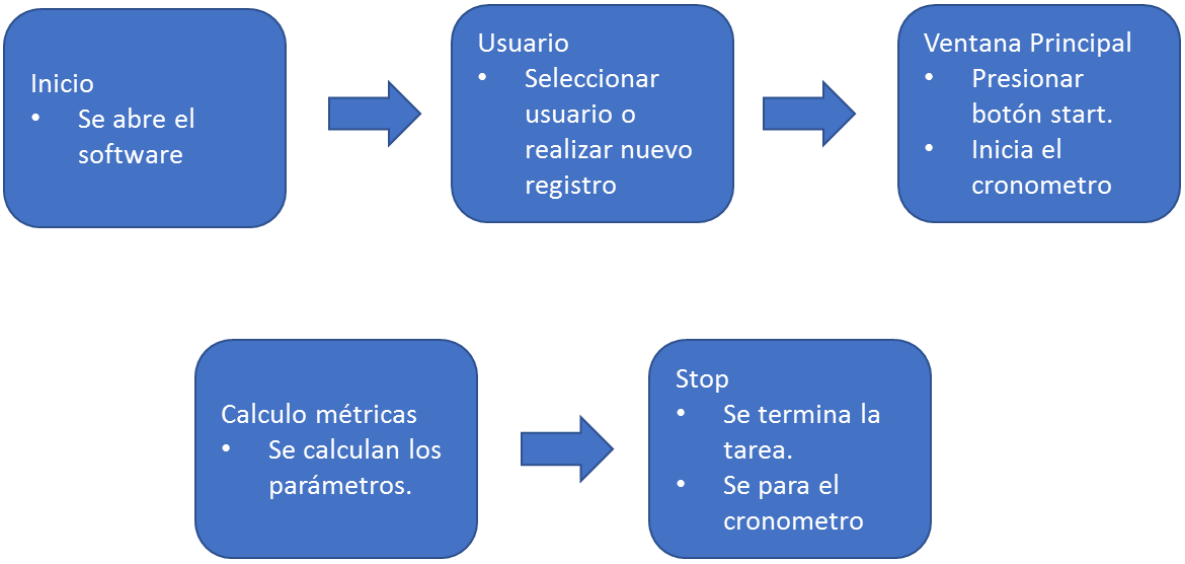


Fig. 33 Pasos para realizar una tarea.

A continuación, se presenta el diagrama de flujo que seguirá el programa para registrar una tarea desde que se inicia el software hasta que termina de registrarse la tarea y se cierra el software.

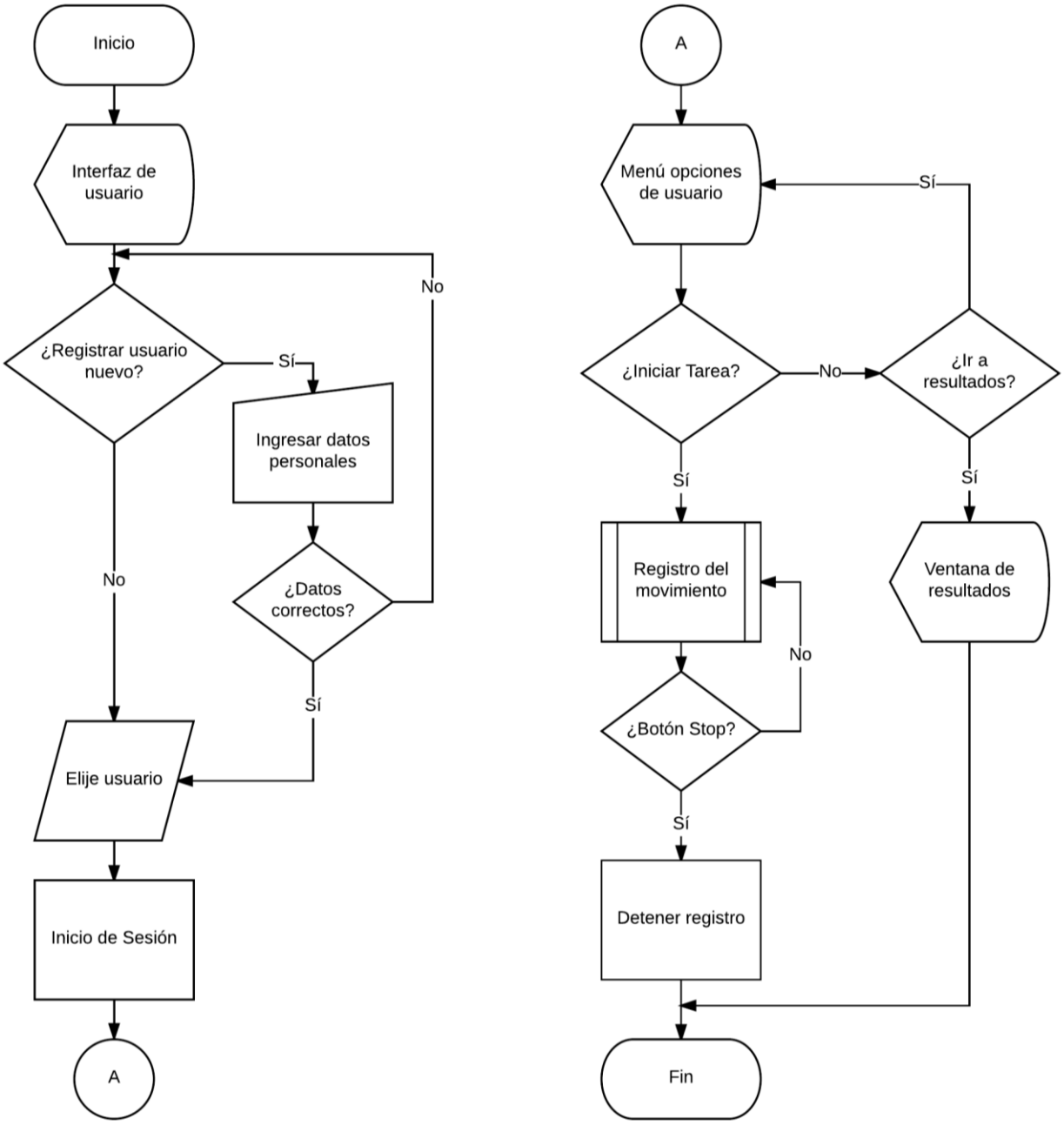


Fig. 34 Diagrama de flujo normal del programa principal desde que se inicia hasta que se cierra.

3.4.7.1 Diagrama de flujo: Registro del movimiento.

El flujo del bloque de "registro de movimiento" se detalla en la siguiente figura. Comprende desde la adquisición de la información mediante los Wiimotes, la transferencia de datos vía Bluetooth, cálculo de parámetros y el despliegue de la interfaz principal que proyecta la aguja intraósea.

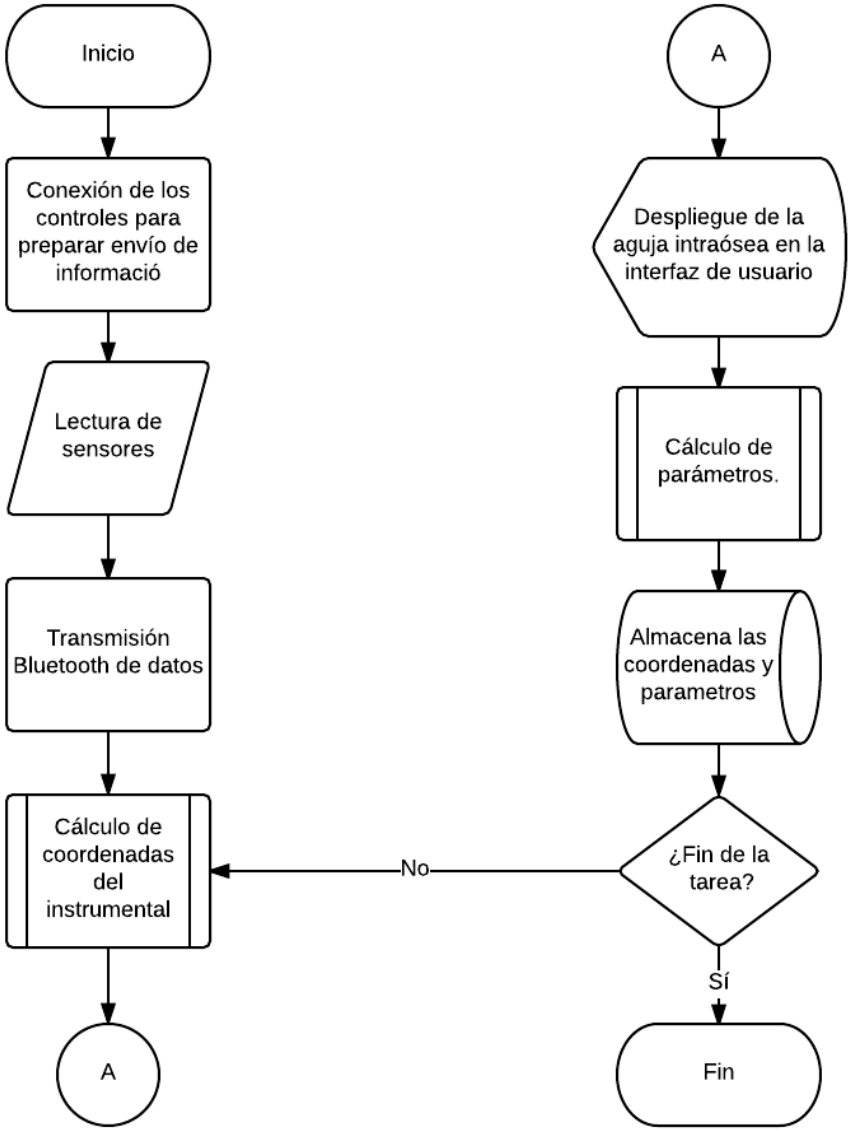


Fig. 35 Diagrama de flujo normal del programa de la ventana principal que registra el movimiento del instrumental desde que se inicia hasta que se cierra.

3.4.7.2 Diagrama de flujo: Cálculo de coordenadas del instrumental.

En esta sección se describe el diagrama de flujo correspondiente para el algoritmo que se encarga de obtener las coordenadas (x, y, z) del instrumental en centímetros.

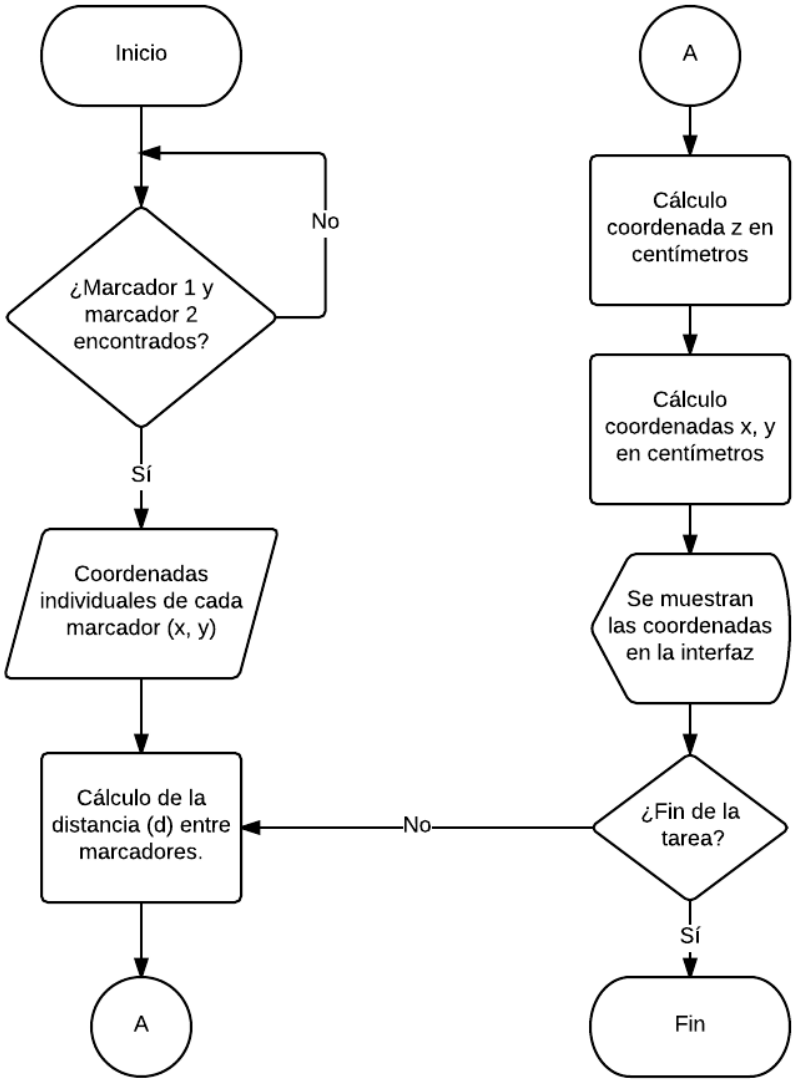


Fig. 36 Diagrama de flujo l del programa de la ventana principal que calcula las coordenadas del instrumental.

3.4.7.3 Diagrama de flujo: Cálculo de parámetros.

Una vez iniciada la tarea, el cronometro comienza a contar el tiempo en que el usuario tarda en realizar el procedimiento, se calcula el ángulo de inclinación y la profundidad de inserción del instrumental. El diagrama de flujo del algoritmo que lleva acabo el procedimiento anterior se muestra a continuación.

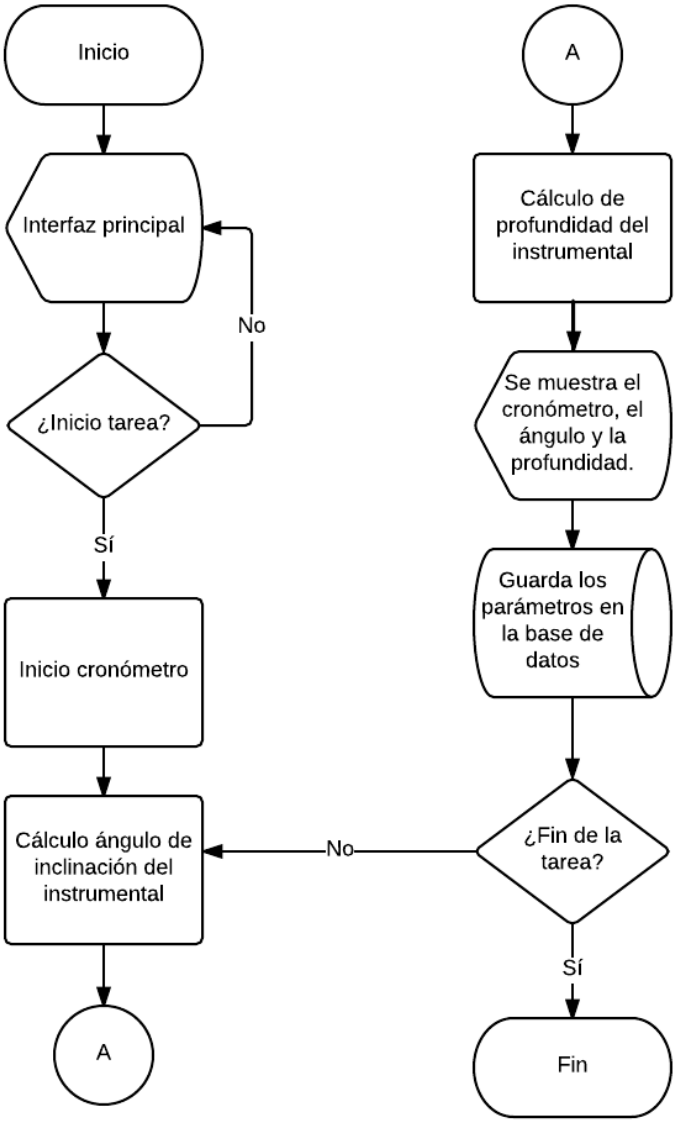


Fig. 37 Diagrama de flujo del programa de la ventana principal que calcula los parámetros.

3.5 Calibración del sistema.

La interfaz gráfica proporciona al usuario retroalimentación visual de la columna vertebral en distintos planos, sin embargo, para que exista una relación entre el espacio físico y las imágenes sobre las que se superpone la aguja intraósea en la interfaz de usuario, es necesario caracterizar dichas imágenes. Para caracterizar el sistema, se utilizó un método de caracterización manual, en el que se define un punto medio en los 3 planos, y a partir de ahí, se escalan las imágenes tomadas del modelo para que exista una relación 1:1 entre los centímetros que se desplaza el instrumental en el mundo real, y los centímetros que se desplaza en la interfaz gráfica. A pesar de que es una caracterización poco ortodoxa, permite manejar libremente el instrumental sobre toda la región de la columna vertebral.

Además las imágenes se modificaron para aparentar imágenes de fluoroscopia, en este sentido, fotografías del modelo sintético fueron tomadas para las vistas Anteroposterior, lateral y axial y después fueron procesadas utilizando filtros a blanco y negro, balance de blancos, *windowing* o ventaneo, que permite realizar un efecto de degradado sobre la imagen, y por ultimo una multiplicación de matrices para obtener la imagen final en la que se puede observar el circulo que aparenta una imagen obtenida de un fluoroscopio.

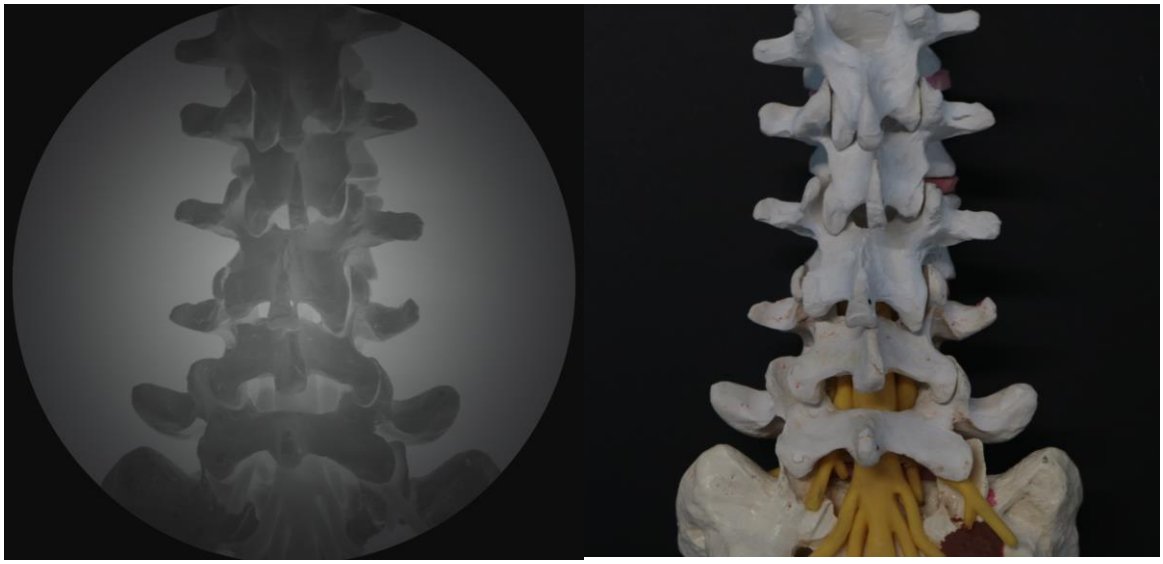


Fig. 38 Vista Anteroposterior del modelo de columna (*izq.*) y su apariencia de fluoroscopia (*der.*).

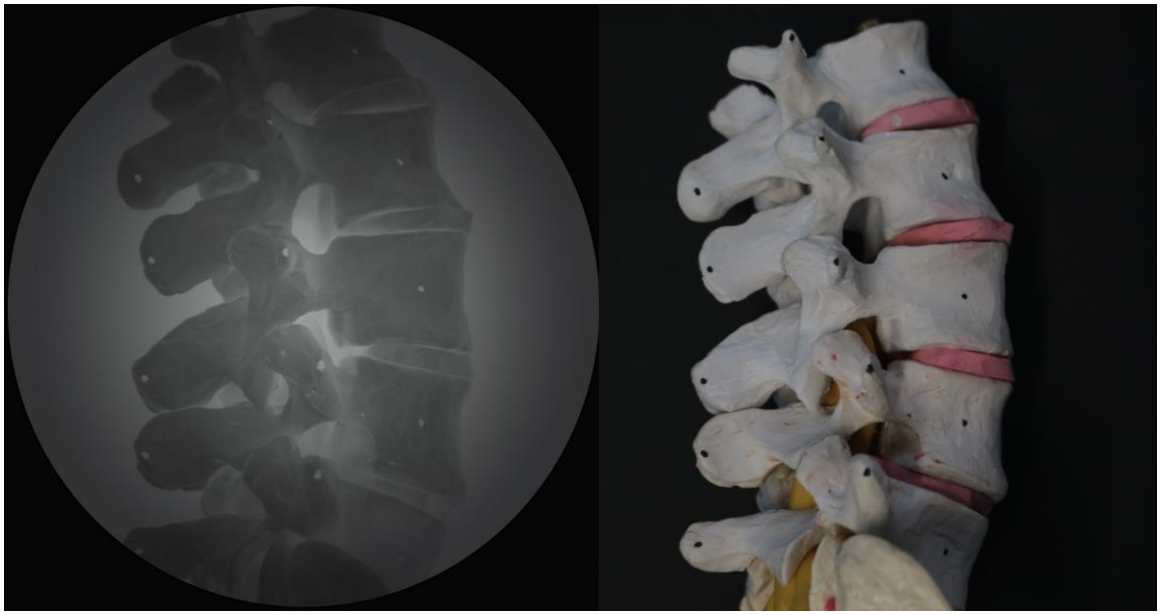


Fig. 39 Vista lateral del modelo de columna (*izq*.) y su apariencia de fluoroscopia (*der*.).

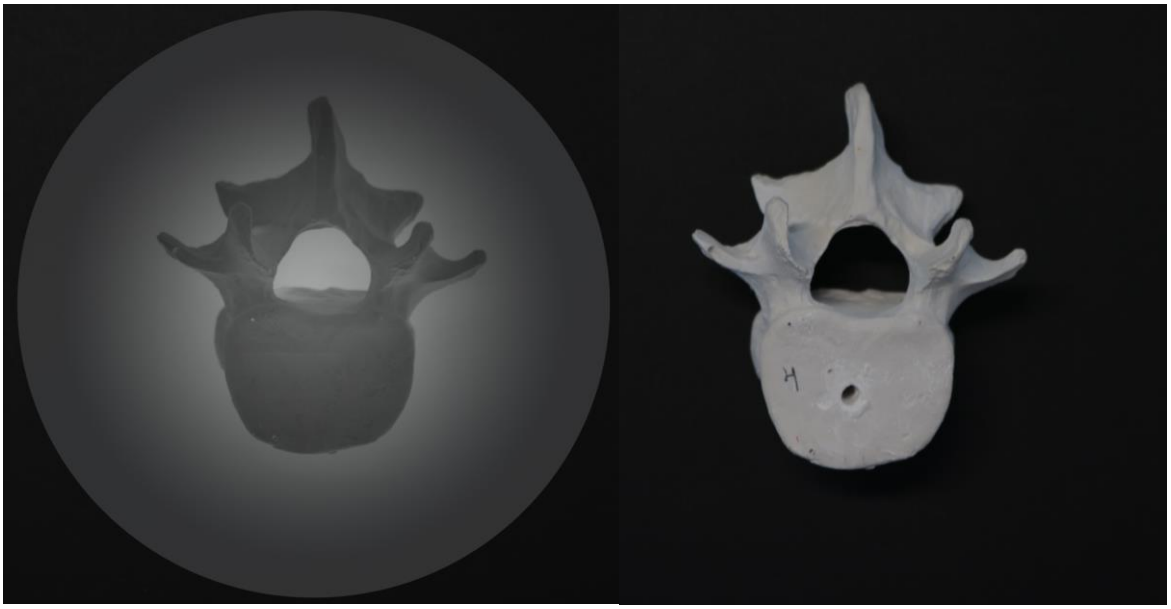


Fig. 40 Vista Axial de la vértebra L4 (*izq*.) y su apariencia de fluoroscopia (*der*.).

Capítulo 4

Pruebas

Se describen las pruebas a las que se sometió el sistema. En este proyecto el Wiimote es extensivamente probado, por lo que ciertos atributos cuantitativos son medidos. Detalles de las pruebas realizadas son presentadas en este capítulo.

4.1 Material.

- Se utilizaron replicas creadas en el laboratorio del instrumental neuroendoscópico utilizado en cirugías de columna. El sistema de tracking se limita a los siguientes instrumentos:

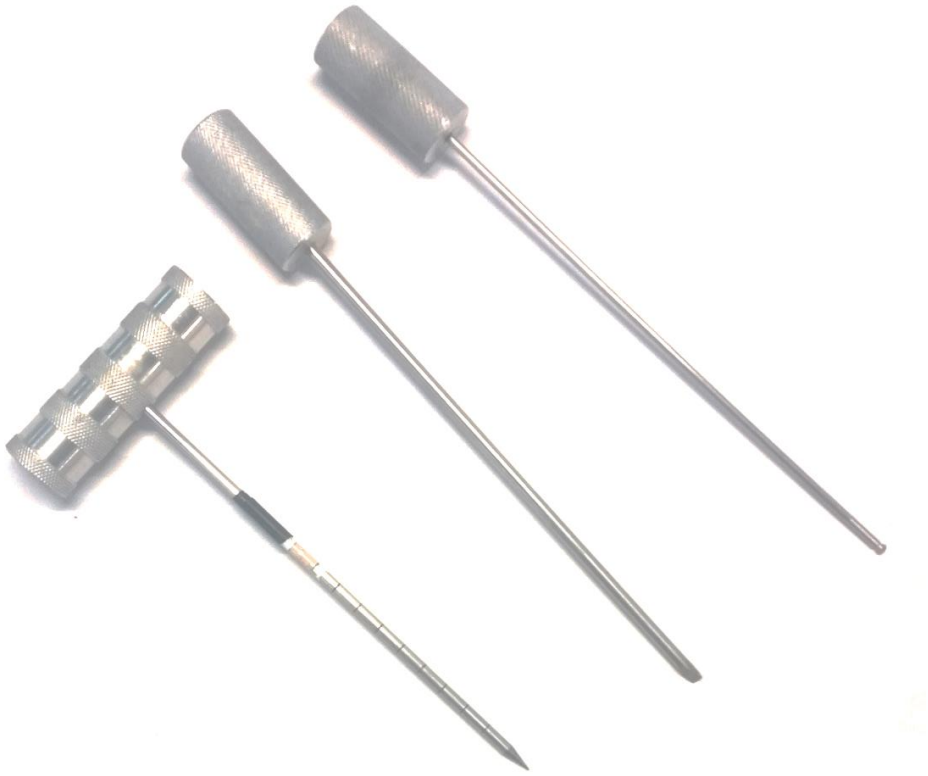


Fig. 41 Instrumentación utilizada. De izquierda a derecha: Punzón, Lezna, Palpador.

- En el instrumental se encuentran montados 4 marcadores infrarrojos cuyas características se describieron anteriormente.

- Se utilizaron dos Wiimotes en configuración ortogonal.

Fig. 42 Wiimotes utilizados montados en sus correspondientes tripiés.

- Modelo sintético de la región lumbar desarrollado en el laboratorio de trabajo.

Fig. 43 Modelo sintético de la región lumbar.

- Computadora de escritorio con Windows 7 y el software de registro.

Para asegurarnos del funcionamiento del sistema se realizaron una serie de mediciones dentro del laboratorio. En la siguiente figura se puede apreciar cómo se realiza una tarea.

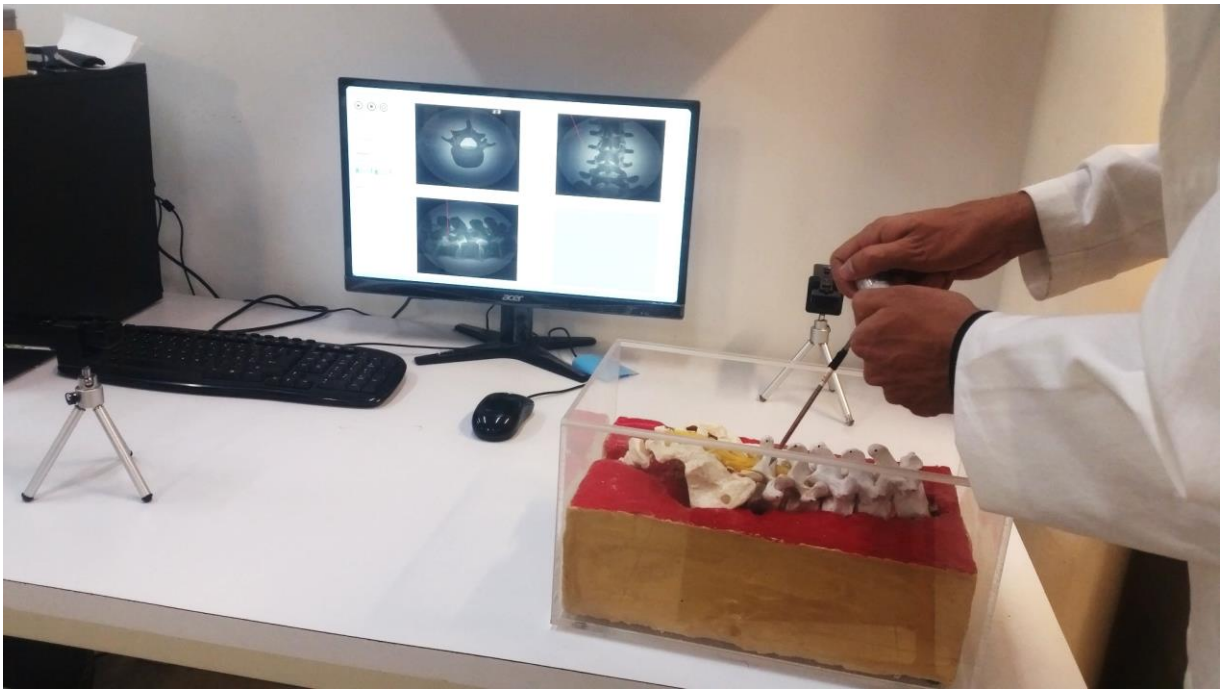


Fig. 44 Posible aplicación del sistema de tracking; en la figura se observa el instrumental, el modelo sintético, los Wiimotes en configuración ortogonal y la interfaz de usuario.

4.2 Pruebas para caracterizar el campo de visión del Wiimote.

En capítulos anteriores se mencionó que, la literatura reporta que los ángulos de visión del Wiimote son 41º en horizontal y 31º en vertical y las coordenadas reportadas para “x” e “y” se encuentran en un rango de 0 a 1023 y de 0 a 767 pixeles respectivamente. Estas características pueden variar de un Wiimote a otro, por lo que es necesario estimar el valor de los ángulos correspondientes a cada dispositivo.

Sin embargo, las limitaciones de la cámara en cuanto a medición en distancia y ángulo de visión durante el seguimiento de marcadores IR son desconocidas. Estas incluyen distancias de medición mínimas o máximas para un funcionamiento efectivo de la cámara y cualquier cambio

no lineal en el campo de visión a medida que el usuario se aleja de la cámara. Estos cambios pueden o no ser significativos, y se atribuyen a la distorsión focal de la lente. Las pruebas se describen a continuación.

- Distancia de detección efectiva mínima de la cámara.
- Alcance máximo de detección para marcadores IR.
- Campos de visión horizontales y verticales.

Para estimar los campos de visión propios de los mandos utilizados en el presente proyecto, se han llevado a cabo experimentos midiendo los cambios en los campos de visión horizontal y vertical encontrando sus límites a diferentes distancias.

Una vez conocidos los límites de visión de la cámara infrarroja, se pueden obtener los ángulos de visión utilizando el teorema de Pitágoras para calcular los ángulos horizontal y vertical. Se realizaron aproximadamente 10 mediciones en incrementos de 10 cm en el eje z, desde el control hasta el LED infrarrojo, se calculó el respectivo ángulo y el resultado final fue el promedio de todas las mediciones. En la sección de resultados se muestran los rangos y ángulos obtenidos.

4.3 Pruebas para el Algoritmo de tracking.

Para asegurarnos del correcto funcionamiento del sistema, se llevaron a cabo las siguientes pruebas:

El algoritmo de seguimiento fue puesto a prueba mediante la medición de coordenadas arbitrarias a partir de los limites horizontal y vertical de la cámara, de los cuales se tomó como origen el limite (0, 0) pixeles, que en el espacio real correspondería a (0, 0) centímetros. Una vez definidos los limites, se colocó un marcador infrarrojo en coordenadas arbitrarias conocidas y se compararon estos datos con los valores calculados mediante el algoritmo. Si bien el Wiimote puede detectar LEDs infrarrojos hasta más de dos metros de distancia, es de suma importancia utilizar LEDs que puedan emitir a largas distancias

Los resultados obtenidos de las pruebas anteriores se muestran en el Capítulo 5 de resultados para el sistema de seguimiento

4.4 Pruebas para el cálculo del ángulo del instrumental.

El algoritmo para el cálculo del ángulo del instrumental fue puesto a prueba. Se fijó el instrumental en ángulos conocidos de 0, 30, 90, 150 y 180 grados y se procedió a comparar el ángulo calculado con el ángulo real. Los resultados se encuentran en el siguiente capítulo. Además, se dejó correr el sistema durante cinco minutos con el instrumental a 90º para observar si existía algún cambio en la medición con respecto al tiempo.

4.5 Invariancia en el tiempo.

Un Wiimote permanece invariable durante un período de tiempo prolongado para establecer si el paso del tiempo tiene algún efecto sobre la exactitud de las mediciones o la estabilidad del sistema. El efecto del uso prolongado en la duración de la batería se analiza para determinar el tiempo operativo máximo del sistema entre los cambios de batería. Esto incluye el tiempo máximo de encendido y las tasas de descarga de ambos Wiimotes, y del sistema LED adjunto al instrumental.

4.6 Espacio de trabajo.

Se realizaron pruebas para determinar el espacio de trabajo en el que los marcadores infrarrojos son detectados por los controles. Los resultados se muestran en la siguiente sección.

Capítulo 5

Resultados

Nuestro sistema registra 3 grados de libertad de movimiento utilizando marcadores activos y sensores de detección infrarroja. Se realiza el registro a una frecuencia de muestreo de 100 Hz y se utiliza el protocolo de comunicación Bluetooth como método de intercambio de información entre la computadora y los sensores infrarrojos. El sistema calcula y registra el tiempo de duración de una tarea, ángulos axial y lateral, así como profundidad de inserción del instrumental.

Tabla 4. Características del sistema de seguimiento.

Características	Valor
Tipo de Sistema	*Outside-in*
Tipo de configuración	Ortogonal
Tipo de marcadores	Activos (infrarrojos)
Comunicación	Bluetooth
Grados de Libertad	3 GDL
Métricas	Tiempo, Angulo axial y lateral, profundidad, coordenadas del instrumental.
Exactitud	> 96%
Precisión ángulo	± 0.102°Ang. axial. ± 0.26 Ang. Axial
Frecuencia de muestreo	100 Hz

5.1 Resultados del Campo de visión del Wiimote.

De las pruebas descritas en el capítulo anterior, se encontró un comportamiento lineal entre la relación pixeles-centímetros vistos por el Wiimote para el eje x y para el eje y, en las siguientes

figuras se puede observar dicho comportamiento, de tal modo que, la relación de los datos reales pixeles-centímetros es casi lineal. Una cámara ideal no tiene distorsión en su campo de visión, sin embargo, es importante aclarar que es posible que existan errores tanto sistemáticos como aleatorios en las mediciones ya que se hicieron de manera manual utilizando un flexómetro. Las mediciones se hicieron cada 2 centímetros para los dos ejes, a una distancia del Wiimote de 30 centímetros.

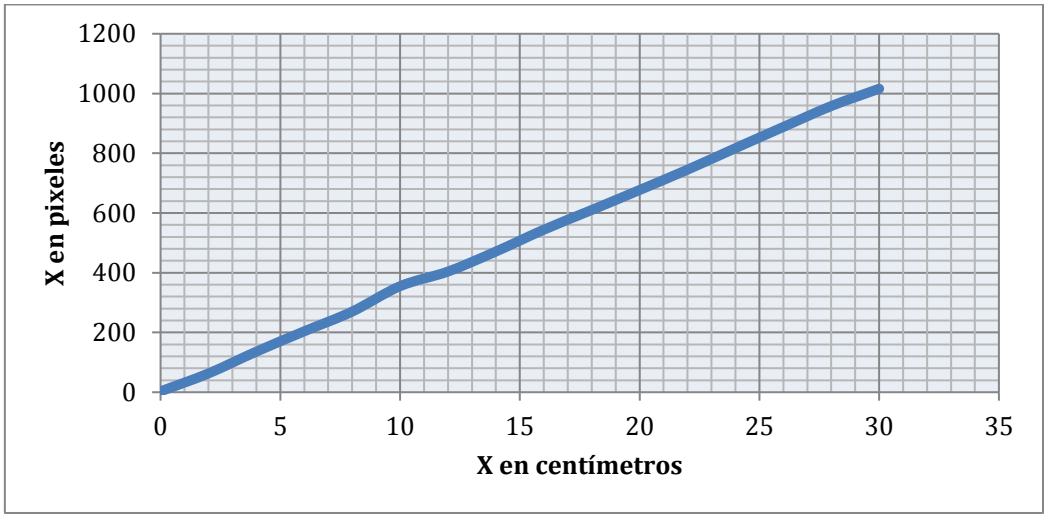


Fig. 45 Relación pixeles-centímetros en el eje x visto por el Wiimote

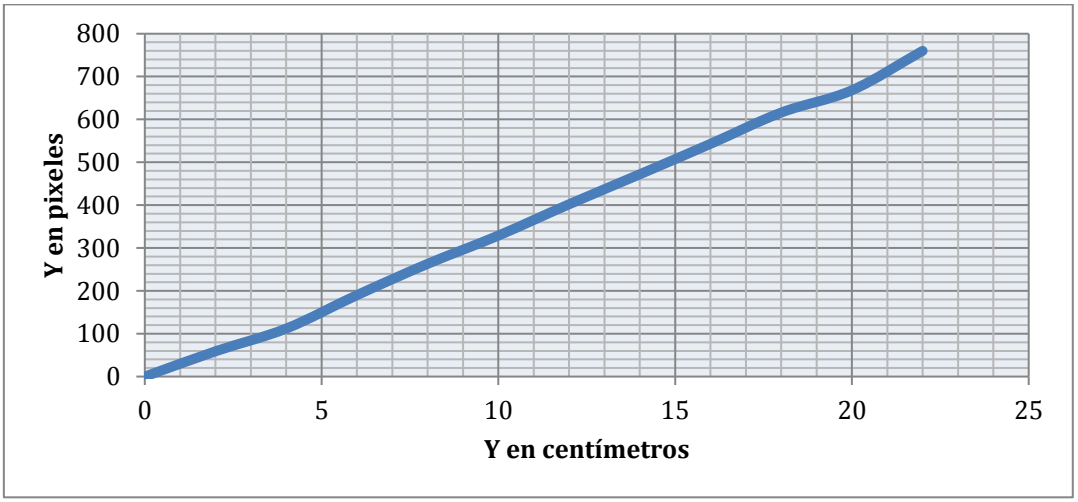


Fig. 46 Relación pixeles-centímetros en el eje y visto por el Wiimote

De lo anterior se encontró que el rango de visión para el eje “x" va de 0 a 1016 pixeles y para el eje “y" va de 0 a 760 pixeles. Se establece que para cualquier fuente de luz a menos de 15 cm de la lente de la cámara no es posible calcular su posición con precisión. Esto es más probable debido a la distorsión de los datos de los marcadores IR almacenados en el CCD. El rango máximo de detección de la cámara depende en gran medida del tamaño, luminiscencia y ángulo del marcador IR. Los experimentos se realizaron con los LEDs infrarrojos seleccionados para el sistema. Sin embargo, los LEDs no se detectan de forma fiable más de 3 metros. La distancia máxima recomendada para utilizar el sistema es de 60 a 80 centímetros de la cámara infrarroja. El ángulo calculado para el campo de visión horizontal es de 40.6° y para el campo de visión vertical de 30.6°

5.2 Exactitud del sistema de seguimiento.

Los resultados de las pruebas realizadas en el capítulo anterior son presentados en la siguiente tabla, en donde todos los valores se encuentran en centímetros.

Tabla 5. Calculo del error relativo para el sistema de seguimiento

Valor real (x, y, z)	**Valor calculado (x, y, z)**	**Error absoluto**
(5, 7.5, 20)	(4.82, 7.66, 20.23)	± (0.18, 0.16, 0.23)
(10, 10.5, 30)	(9.91, 10.49, 30.19)	(0.09, 0.01, 0.19)
(15, 13, 40)	(14.92, 13.61, 40.66)	(0.08, 0.61, 0.66)
(8, 16, 50)	(7.9, 16.4, 50.59)	(0.01, 0.40, 0.59)
(25, 20, 60)	(24.96, 19.49, 60.88)	(0.04, 0.51, 0.88)
(30, 22, 70)	(29.40, 22.08,70.79)	(0.60, 0.08, 0.79)
(40, 25, 80)	(40.16, 25.89, 80.39)	(0.16, 0.89, 0.39)

En la tabla anterior se observa que se calculó el error absoluto entre las mediciones lo que nos permite obtener una medida de la exactitud del sistema. En promedio, para el eje x se tiene un error absoluto aproximado de ± 0.165 cm, para el eje y ± 0.38 cm, y por ultimo para el eje z se tiene un error absoluto aproximado de 0.53 cm. Los marcadores se colocaron a una distancia aproximada de 60-80 centímetros respecto a los controles.

5.3 Resultados para el cálculo del ángulo del instrumental.

Los resultados de las pruebas realizadas en el capítulo anterior para el cálculo del ángulo del instrumental con la cámara en la vista axial son presentados en la siguiente tabla.

Tabla 6. Calculo del error absoluto para el ángulo axial.

Valor real	Valor calculado (x, y, z)	Error absoluto
0º	0.39	0.39
30º	30.06	0.06
90º	89.97	0.02
150º	150.04	0.04
180º	180	0

De los datos de la tabla anterior se obtuvo una varianza de 0.02113 y una desviación estándar de 0.1453°. Cuanta más pequeña sea la desviación estándar mayor será la concentración de datos alrededor de la media. El error absoluto en promedio es aproximadamente de ±0.102°. Es posible observar en la Figura 47 el registro de tiempo que marca los 5 minutos, además del que se muestra que el ángulo calculado fue igual a 90º y además, se aprecia que los dos LEDs que se encuentran en el instrumental se encuentran en la misma posición en x (480 px) por lo que se deduce que el ángulo calculado fue exacto.

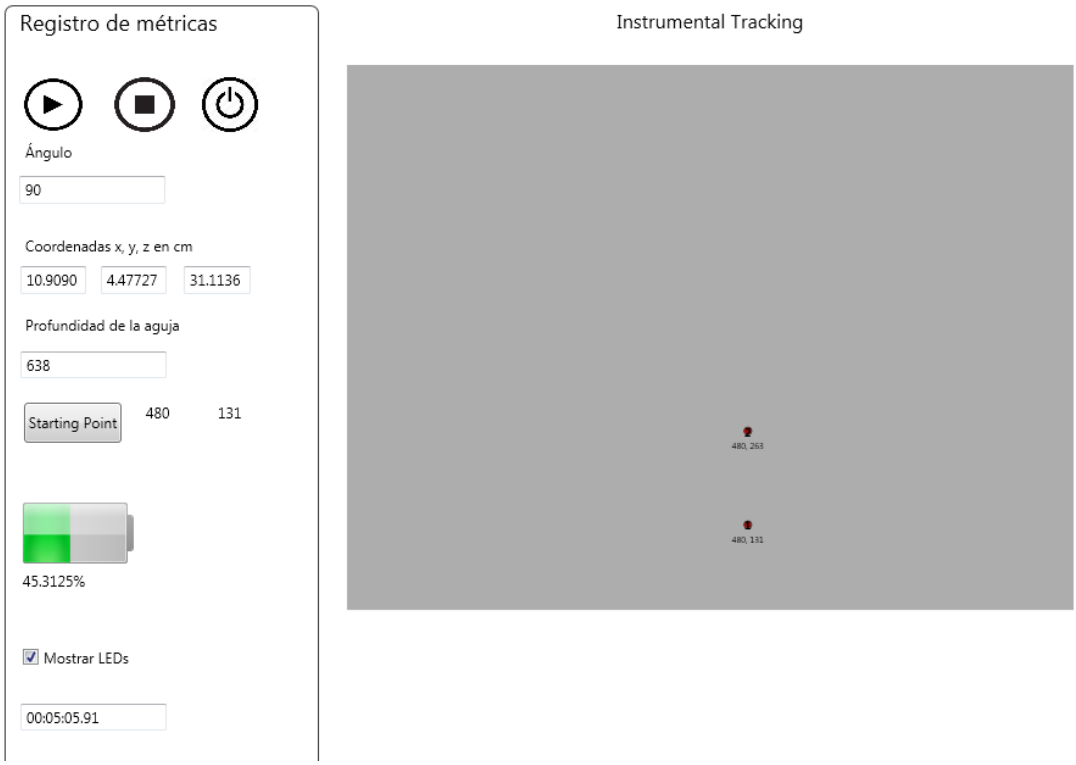


Fig. 47 Pruebas para el cálculo del ángulo.

Los resultados de las pruebas realizadas en el capítulo anterior para el cálculo del ángulo del instrumental con la cámara en la vista lateral son presentados en la siguiente tabla.

Tabla 7. Calculo del error absoluto para el ángulo lateral.

Valor real	**Valor calculado (x, y, z)**	**Error absoluto**
0º	0.19	0.19
30º	30.18	0.18
90º	90.07	0.07
150º	150.76	0.76
180º	179.90	0.10

De los datos de la tabla anterior se obtuvo una varianza de 0.0646 y una desviación estándar de 0.2541°.

Cuanta más pequeña sea la desviación estándar mayor será la concentración de datos alrededor de la media. El error relativo en promedio es aproximadamente de ±0.26°.

5.4 Invariancia en el tiempo.

Un solo Wiimote fue conectado a la computadora para capturar continuamente la posición de los marcadores infrarrojos. Durante un período de 1 hora, las baterías se descargaron en un 8%, medido por el informe de estado de la batería. Esto se traduce en aproximadamente 12.5 horas de tiempo de uso con un nuevo par de pilas AA. Durante este período de tiempo, no hubo fluctuación o cambio en la localización de las fuentes de luz IR, por lo que se concluye que el sistema es invariante en el tiempo.

5.5 Espacio de trabajo.

La siguiente figura muestra el análisis que se realizó para obtener el espacio de trabajo en el cual los marcadores infrarrojos pueden ser detectados por el instrumental. Se calculó para una distancia de 80 centímetros respecto a las cámaras.

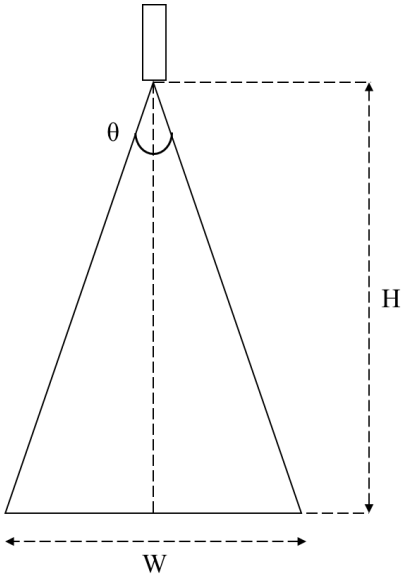


Fig. 48 Espacio de Trabajo.

A una distancia de 80 centímetros y utilizando el ángulo de visión horizontal calculado anteriormente se procede a conocer W mediante la siguiente formula:

$$W = 2H \tan\frac{\theta}{2} \quad (19)$$

Se obtiene como resultado aproximadamente 59 centímetros para el eje horizontal, sin embargo, para el eje vertical se realiza el mismo procedimiento, pero utilizando el ángulo de visión vertical.

Capítulo 6

Discusión

La mayoría de los sistemas de tracking utilizan cámaras de video para obtener información acerca de la posición del objeto en el espacio. El seguimiento de objetos puede ser un proceso lento debido a la gran cantidad de datos que contiene un video. Además, existen distintos factores que incrementan la complejidad del seguimiento de objetos utilizando cámaras de video normales. Sin embargo, la utilización de los sensores infrarrojos presenta ciertas ventajas que se menciona a continuación:

- *Iluminación ambiente.* Los sistemas ópticos que utilizan cámaras de video se ven fuertemente afectados por la iluminación del ambiente, sin embargo, los sensores infrarrojos de los controles del Nintendo Wii detectan longitudes de onda de 940 nm por lo que no se ven afectados por la luz ambiental.

- *Resolución*. La resolución de una cámara de video convencional no es suficiente para capturar los detalles requeridos. La resolución del Wiimote es de 1024 x 767 pixeles aproximadamente que es mucho mayor a la de las cámaras convencionales.

- *Frecuencia*. La frecuencia de las cámaras a la que se muestran las secuencias de imágenes no permite capturar movimientos rápidos. Comúnmente tienen una frecuencia de 30 o 60 cuadros por segundo. El Wiimote provee una frecuencia de 100 Hz muy por encima de las cámaras convencionales.

- *Sistema magnético*. Una de las principales desventajas de los sistemas magnéticos es que objetos con mucha mayor permeabilidad magnética como los metales, pueden causar interferencia en los campos magnéticos generados por los dispositivos, lo que conlleva a errores en los cálculos de la posición y orientación del sensor. Problema que no se presenta en los sistemas infrarrojos.

- *Seguimiento acústico.* El principal problema con los sistemas acústico es que desafortunadamente, superficies acústicamente reflectoras pueden causar interferencias con el sistema. Problema que no se presenta en los sistemas infrarrojos.

Tabla 8. Ventajas y desventajas de distintos sistemas de tracking.

Tipo de Sistema	Ventajas	Desventajas
Óptico (cámara webcam)	No necesita alterar el objetivo al que se realiza el tracking	Interferencias por luz ambiental, baja frecuencia (30-60 Hz) de muestreo, baja resolución, cables por conexión de cámaras.
Infrarrojo (Wiimote)	No hay interferencias por luz ambiental, alta frecuencia de muestreo (100-150 Hz) sistema inalámbrico, alta resolución (1024 x 768 px.)	Los marcadores infrarrojos tienen que estar siempre apuntando hacia la cámara
Acústico	Rango de tracking entre 100-1000 m	Interferencias por superficies acústicamente reflectoras
Magnético	Alta frecuencia (100 Hz), múltiples sensores permiten mayor precisión.	Objetos con mucha mayor permeabilidad magnética como los metales, pueden causar interferencia en los campos magnéticos generados por los dispositivos

La configuración ortogonal utilizada para los Wiimotes también representa ciertas ventajas sobre las otras configuraciones que se mencionan a continuación. Cada Wiimote calcula las coordenadas del instrumental respecto a si mismo, de tal modo que el campo de visión de cada cámara es independiente y el espacio de trabajo no se ve reducido por el área en la que se intersectan los campos de visión de los dos Wiimotes.

Tabla 9. Comparación de configuraciones para el sistema de seguimiento.

Configuración	No. Wiimotes	No. LEDs	DOF	Precisión	Costo $	Complejidad de Cálculos	Parámetros
Ortogonal	2	4	4	Alta	Alto	Media	Ángulo, Profundidad, Coord. z
Estereoscópica	2	1/2	3/6	Media	Alto	Alta	Coord. z
1 Wiimote, 2 marcadores	1	2	4	Alta	Bajo	Media	Ángulo, profundidad, Coord. z
1 Wiimote, 1 marcador	1	1	2	Baja	Bajo	Baja	x

Alto 100 USD, Medio 50 USD, Bajo 25 USD.

De la Tabla 8 es posible observar que la configuración ortogonal en combinación con la configuración de 1 Wiimote 2 LEDs presenta ventajas superiores a las demás configuraciones.

Sin embargo, la utilización del Wiimote también tiene sus desventajas. El estrecho campo de visión hace que sea incómodo de usar ya que tiene que mantener el objetivo visible para la cámara en todo momento. Se puede escalar el tamaño físico del marcador si desea realizar un seguimiento desde una distancia más lejana. Sin embargo, independientemente de la escala física por lo general comienza a obtener falsos verdaderos en cuanto a la posición de los marcadores una vez que la imagen proyectada es más pequeña que 1/10 del plano de imagen. Esto es probablemente porque la resolución real de la cámara es realmente 128 × 96 (mediante análisis del subpixel se consigue una resolución 1024 × 768).

Capítulo 7

Conclusiones y perspectivas

En este trabajo, una nueva técnica de seguimiento de instrumental neuroendoscópico ha sido presentada y probada. La técnica propuesta utiliza los campos de visión del Wiimote para seguir los 3 grados de libertad del instrumental. Este enfoque requiere sólo un par de LEDs y puntos de referencia que es bastante ventajoso, además, esta plataforma es relativamente barata, móvil y utilizable para otras aplicaciones.

Se pretende utilizar el sistema de tracking para aplicaciones en neurocirugía ya que cuenta con las características adecuadas para el seguimiento del instrumental utilizado en estos procedimientos, como, por ejemplo, para cirugías de fijación de columna mediante tornillos intrapediculares. Se espera que, en un futuro a corto plazo, se realicen pruebas con el sistema para ser implementado en un entrenador para cirugías de esta índole. Una de las principales ventajas es que, el sistema puede ser utilizado para cualquier aplicación que requiera conocer la posición y orientación de un objeto en el espacio.

Uno de los objetivos a futuro es cambiar la forma en la que se caracterizó el sistema, tanto las imágenes en la interfaz de la columna para que exista una relación entre las imágenes y el espacio físico real y el sistema sobre el que se montan los Wiimotes. Una de las posibles soluciones es utilizar una cámara de video de tal forma que sea posible para el sistema conocer exactamente donde se encuentra el modelo de columna respecto al Wiimote. Además, también es necesario calibrar el montaje del sistema, ya que si los Wiimotes se colocan a una altura mayor o menor que la necesaria produce un efecto negativo sobre los resultados.

El sistema se mostró fiable y preciso durante las pruebas. El objetivo del proyecto se vería cumplido puesto que se ha podido realizar un sistema que es capaz de realizar el seguimiento de objetos con errores aceptables y que pueden reducirse si la caracterización de la parte física del sistema se mejora.

La utilización de marcadores infrarrojos presento ciertas ventajas respecto a los sistemas ópticos que utilizan cámaras de video, ya que no se ve alterado por la iluminación del ambiente, sin embargo, es importante destacar que los sensores infrarrojos de las cámaras de los controles del Nintendo Wii se ven afectadas por la iluminación solar que contiene componentes en el espectro del infrarrojo.

La interfaz de usuario desarrollada permite al usuario una retroalimentación visual de las manipulaciones que realiza con el instrumental en tiempo real y le permite conocer además, parámetros de dichas manipulaciones como los ángulos lateral y axial del instrumento.

Referencias

[1]. Alper Yilmaz, Omar Javed, Mubarak Shah. "Object Tracking: A Survey" ACM Computing Surveys, Vol. 38, No. 4, Article 13, Publication date: December 2006.

[2]. David S. Bolme J. Ross Beveridge Bruce A. Draper Yui Man Lui "Visual Object Tracking using Adaptive Correlation Filters," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010.

[3]. S. De Amici, A. Sanna, F. Lamberti, B. Pralio "A Wii remote-based infrared-optical tracking system" Entertainment Computing 1,pp 119–124, 2010.

[4]. Sudha Challa, Mark R. Morelande, Darko Musicki "Fundamentals of Object Tracking" Cambridge University Press, Jul 28, 2011.

[5]. Krames Staywell "Cirugia de la Columna Lumbar" Krames Patient Education. The Staywell Company, 2010.

[6]. Sandeep Kumar Patel, Agya Mishra "Moving Object Tracking Techniques: A Critical Review" Indian Journal of Computer Science and Engineering (IJCSE). Vol. 4 No. 2. 2013

[7]. Jayashree Baskaran, Ruvi Subban "Compressive Object Tracking – A Review and Analysis" IEEE. International Conference on Computational Intelligence and Computing Research. 2014

[8]. Yi Wu, Jongwoo Lim, Ming-Hsuan Yang. "Online Object Tracking: A Benchmark" IEEE. Computer Vision and Pattern Recognition (CVPR). Junio 2013

[9]. Thomas Schlomer, Benjamin Poppinga, Niels Henze, Susanne Boll1 “Gesture Recognition with a Wii Controller ” Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction. 2008

[10]. Johnny Chung Lee. “Hacking the Nintendo Wii Remote” Pervasive Computing IEEE, Vol.7, pp. 39-54. 2008

[11] Michael Wronski. (2008) “Design and Implementation of Hand Tracking Interface using the Nintendo Wii Remote, Master’s thesis, University of Cape Town.

[12] Joa Lourenco. (2010) “Wii3D: Extending the Nintendo Wii Remote into 3D,” Bachelor’s Degree, Rhodes University,

[13] Brian Peek. “Managed library for nintendo's Wiimote”. Julio 2016 [En red],. Disponible en: https://channel9.msdn.com/coding4fun/articles/Managed-Library-for-Nintendos-Wiimote.

[14]. Burak Ozgur, Edward Benzel, Steven Garfin “Minimally Invasive Spine Surgery: A Practical Guide to Anatomy and Techniques” Springer Science + Bussines Media, LLC 2009.

[15] Browner et al, Skeletal Trauma, Low Lumbar Fractures. [En red]. Disponible en: http://z0mbie.host.sk/Low-Lumbar-Fractures.html

[16] Browner et al, Skeletal Trauma, Low Lumbar Fractures. [En red]. Disponible en: http://z0mbie.host.sk/Low-Lumbar-Fractures.html

[17] Brainlab, Brainlab Sppinal Navigation. [En red]. Disponible en: https://www.brainlab.com/en/surgery-products/overview-spinal-trauma-products/

[18] Yashar Deldjoo, Reza Ebrahimi Atani "A Low-Cost Infrared-Optical Head Tracking Solution for Virtual 3D Audio Environment Using the Nintendo Wii-Remote". Entertainment Computing. 2015 Elsevier B.V.

[19] Sylvain Bertrand , Julien Marzat , Mathieu Carton. "A Low-Cost System for Indoor Motion Tracking of Unmanned Aerial Vehicles." Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing. 2011.

[20] Jongshin Kim, Kyoung Won Nam, Ik Gyu Jang, Hee Kyung Yang, Kwang Gi Kim, and Jeong-Min Hwang, "Nintendo Wii Remote Controllers for Head Posture Measurement: Accuracy, Validity, and Reliability of the Infrared Optical Head Tracker" Investigative Ophthalmology & Visual Science, , Vol. 53, No. 3. March 2012

[21] J.E.T. Collins & G. Bright "Nintendo Wii remotes provide a reconfigurable tool-changing unit with an automatic calibration capability." South African Journal of Industrial Engineering Vol 25(2), pp 135-147. August 2014

[22] Luigi Gallo & Mario Ciampi. "Wii Remote-enhanced Hand-Computer Interaction for 3D Medical Image Analysis." International Conference on the Current Trends in Information Technology (CTIT), 2009

[23] H. Zhoua, H. Hu, "Human motion tracking for rehabilitation—a survey", Biomed. Signal Process. Control 3 (1) (2008) 1–18.

[24] Ryan A. Pavlik, Judy M. Vance, A modular implementation of Wii remote head tracking for virtual reality, in: ASME 2010 World Conference on Innovative Virtual Reality, American Society of Mechanical Engineers, 2010, pp. 351–359.

[25] M. Ubila, D. Mery, R.F. Cadiez "Head tracking for 3D audio using the Nintendo Wiimote". International Computer Music Conference. 2010, pp. 494 497

[26] L. Bharath, S. Shashank, V.S. Nageli, Sangeeta Shrivastava, S. Rakshit, Tracking method for human computer interaction using Wii remote, in: 2010 Internationa Conference on Emerging Trends in Robotics and Communication Technologies (INTERACT), IEEE, 2010, pp. 133–137.

[27] Ross C. Williams, Finger tracking and gesture interfacing using the Nintendo wiimote, in: Proceedings of the 48th Annual Southeast Regional Conference, ACM, 2010, p. 11.

[28] Johnny Lee, "Wii Remote projects". 2016. [En red]. Disponible en: http://johnnylee.net/projects/wii/urlhttp://johnnylee.net/projects/wii/

[29] O. Kreylos, "Wiimote Hacking" 2016. [En red]. Disponible en: http://graphics.cs.ucdavis.edu/okreylos/ResDev/Wiimote/index.html

[30] Shimrat, M., "Algorithm 112: Position of point relative to polygon", Communications of the ACM Volume 5 Issue 8, Aug. 1962

Apéndices

Apéndice A – Cálculos por Estereoscopia.

La herramienta principal para la reconstrucción de la posición 3D es la intersección de dos rayos **[19, 20].** Si uno de los Wiimotes es seleccionado como el origen $o_1 = [0,0,0]^T$ entonces el segundo Wiimote se encuentra a una distancia fija y conocida "d" sobre el eje x del marco de referencia $o_2 = [d,0,0]^T$. En esta configuración paralela, las coordenadas del punto a seguir están dadas por: $= [x,y,z]^T$. Es posible obtener un vector desde la cámara hasta el marcador infrarrojo utilizando los campos de visión horizontal y vertical. Estos vectores pueden ser vistos como rayos, en este sentido, cualquier punto detectado va a reportar las mismas coordenadas sobre todo el rayo. Estos rayos pueden ser definidos como:

$$r_1 = o_1 + t_1 d_1 \quad \text{(A1)}$$

$$r_2 = o_2 + t_2 d_2 \quad \text{(A2)}$$

donde d_1 y d_2 son vectores dirección desde las cámaras hasta los marcadores, mientras que t_1 y t_2 son las distancias desconocidas entre el marcador y cada una de las cámaras. Para calcular los vectores dirección es necesario calcular los valores normalizados de los pixeles para cada una de las cámaras que reportan las coordenadas bidimensionales denotadas por (x_1, y_1) y (x_2, y_2). La Fig. A1 muestra el campo de visión de las dos cámaras, así como los dos rayos construidos a partir de un solo marcador infrarrojo en cada uno de los planos de imagen de las cámaras.

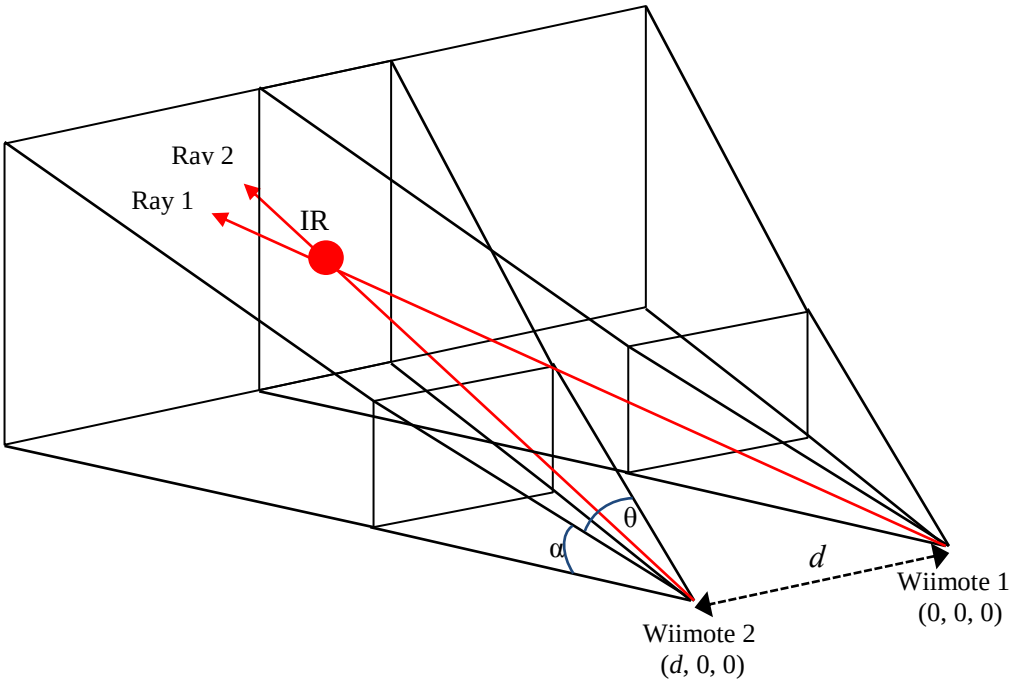


Fig. A1 Intersección de 2 rayos.

La resolución para ambos Wiimotes utilizados en este proyecto es de 1016 x 760 pasando estas coordenadas a un rango de [-1; 1] es posible normalizar los pixeles:

$$\hat{x}_{1,2} = 2\frac{x_{1,2}}{1016} - 1 \qquad \text{(A3)}$$

$$\hat{y}_{1,2} = 2\frac{y_{1,2}}{760} - 1 \qquad \text{(A4)}$$

Ahora es posible calcular los vectores dirección utilizando las coordenadas normalizadas. Se asume que el Wiimote apunta en la dirección $[0,0,-1]^T$ en su propio eje de coordenadas. Las coordenadas de cada rayo están dadas por:

$$d_1 = \begin{bmatrix} \hat{x}_1 \cdot \tan\left(\frac{a}{2}\right) \\ \hat{y}_1 \cdot \tan\left(\frac{q}{2}\right) \\ 1 \end{bmatrix} \qquad d_2 = \begin{bmatrix} \hat{x}_2 \cdot \tan\left(\frac{a}{2}\right) \\ \hat{y}_2 \cdot \tan\left(\frac{q}{2}\right) \\ 1 \end{bmatrix} \qquad \text{(A5)}$$

donde α and θ son los campos de visión horizontal y vertical respectivamente. Dado que en el sistema existe errores de mediciones de las constantes como la distancia entre los Wiimotes, existen inexactitudes y como consecuencia la intersección de los rayos nunca será perfecta por lo que es necesario calcular estimaciones igualando r_1 y r_2 de tal modo que:

$$o_1 + t_1 d_1 = o_2 + t_2 d_2 \tag{A6}$$

Calculando el producto cruz de la expresión anterior para d_1 y d_2 se obtiene:

$$t_1 = \frac{\det\left(o_2 - o_1, d_2, d_1 \times d_2\right)}{\left\| d_1 \times d_2 \right\|^2} \tag{A7}$$

$$t_2 = \frac{\det\left(o_2 - o_1, d_1, d_1 \times d_2\right)}{\left\| d_1 \times d_2 \right\|^2} \tag{A8}$$

Una coordenada final puede ser calculada utilizando las longitudes de cada vector como el promedio de dos puntos:

$$\hat{x} = \frac{r_1 + r_2}{2} \tag{A9}$$

Si bien, es necesario calibrar las cámaras infrarrojas para lograr calcular con exactitud la posición del marcador, para los fines de este proyecto se asume que las cámaras son un sistema lineal homogéneo por lo que no se considera la distorsión de la lente. Dado que las cámaras se encuentran paralelas entre ellas y a una distancia fija no existe rotación, y únicamente se considera una simple traslación en un eje.

Apéndice B - Conexión Bluetooth del Wiimote con la computadora.

A continuación, se describen los pasos necesarios para conectar satisfactoriamente el Wiimote a la computadora.

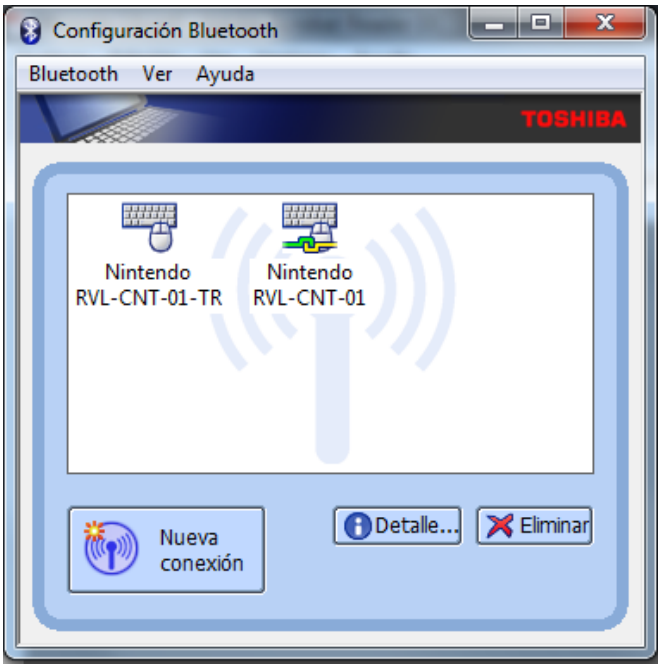


Fig. B1 Conexión de dispositivos Bluetooth.

i. En la ventana “Configuración Bluetooth” presionar el botón “Nueva conexión”.

ii. Aparecerá una ventana llamada “Asistente para agregar nueva conexión”. Seleccionar el modo exprés y presionar siguiente.

iii. Una nueva ventana se muestra buscando el nuevo dispositivo. Presionar el botón “*sync*” que se encuentra en la parte trasera del Wiimote.

iv. En la ventana de “configuración Bluetooth” aparecerá el nombre del dispositivo. En el caso de los Wiimotes aparecerá el nombre Nintendo RVL-CNT-01 para la primera versión de los mandos, sin embargo, para los mandos nuevos el nombre aparecerá como Nintendo RVL-CNT-01-TR. Aparecerá una línea verde conectada a una línea amarilla, lo que significa que el dispositivo se ha conectado satisfactoriamente con la computadora.

Apéndice C - Programa correspondiente al menú inicio.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Diagnostics;
using System.Data.SQLite;

namespace SpineTrainer
{
    /// <summary>
    /// Interaction logic for MenuInicio.xaml
    /// </summary>
    public partial class MenuInicio : Page
    {
        string dbConnectionString = @"Data Source=SurgeryTrainer.sqlite;Version=3;";
        public MenuInicio()
        {
            InitializeComponent();
        }


        private void Hyperlink_RequestNavigate(object sender, RequestNavigateEventArgs
e)
        {
            Process.Start(new ProcessStartInfo(e.Uri.AbsoluteUri));
            e.Handled = true;
        }

        private void Page_Loaded_1(object sender, RoutedEventArgs e)
        {
            SQLiteConnection sqliteCon = new SQLiteConnection(dbConnectionString);
            sqliteCon.Open();
            //string Consulta = "select usuario from registro";
            //verifica si ya existe el usuario
            string Consulta = "SELECT usuario from registro";
            SQLiteCommand Comando = new SQLiteCommand(Consulta, sqliteCon);
            SQLiteDataReader dr = Comando.ExecuteReader();
            while (dr.Read())
            {
                cbUsuarios.Items.Add(dr.GetString(0));
            }

        }

        private void btnIniciar_Click(object sender, RoutedEventArgs e)
```

```
        {
            //this.NavigationService.Navigate(new Uri("OpcionesUsuario.xaml",
UriKind.Relative));
            if (cbUsuarios.SelectedIndex == -1)
            {
                MessageBox.Show("Elige un usuario");
                return;

            }
            string usuario = cbUsuarios.SelectedItem.ToString();
            this.NavigationService.Navigate(new OpcionesUsuario(usuario));
        }

        private void btnSalir_Click(object sender, RoutedEventArgs e)
        {
            var parentWindow = this.Parent as Window;
            if (parentWindow != null)
            {
                parentWindow.Close();
            }

        }
        protected override void OnMouseLeftButtonDown(MouseButtonEventArgs e)
        {
            base.OnMouseLeftButtonDown(e);

            // Begin dragging the window
            var parentWindow = this.Parent as Window;
            if (parentWindow != null)
            {
                parentWindow.DragMove();
            }


        }


        private void comboBox1_DropDownClosed(object sender, EventArgs e)
        {
            cbUsuarios.SelectedIndex = -1;
            cbUsuarios.Text = "";
        }



    }
}
```

Programa en XAML

```
<Page x:Class="SpineTrainer.MenuInicio"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:Microsoft_Windows_Themes="clr-
namespace:Microsoft.Windows.Themes;assembly=PresentationFramework.Aero"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"

       Title="MenuInicio" Loaded="Page_Loaded_1"  >
    <Border Background="LightBlue"
          BorderBrush="Black"
          BorderThickness="2"
          CornerRadius="45"
          Padding="25">
        <Grid Background="White"  >
            <!--<Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition Height="0*"/>
        </Grid.RowDefinitions>-->
            <!--<Label x:Name="lblRegistro" Content="Registro"
HorizontalAlignment="Left" Margin="218,237,0,0" VerticalAlignment="Top" Height="35"
Width="79"/>-->

            <TextBlock HorizontalAlignment="Center" VerticalAlignment="Center"
Margin="623,492,809,265">
            <Hyperlink NavigateUri="RegistroUsuario.xaml" >
                Registro
             </Hyperlink>
            </TextBlock>
            <Label Content="&#x9;Elige tu Usuario o realiza un registro nuevo"
HorizontalAlignment="Center" Margin="492,362,554,377" VerticalAlignment="Center"
FontSize="18" Width="430"/>
            <ComboBox x:Name="cbUsuarios" HorizontalAlignment="Center"
Margin="623,423,609,318" VerticalAlignment="Center" Width="244" Height="32"/>
            <Button x:Name="btnIniciar" Content="Iniciar" HorizontalAlignment="Center"
Margin="700,486,702,265" VerticalAlignment="Center" Width="74"
Click="btnIniciar_Click"/>
            <Button x:Name="btnSalir" Style="{StaticResource MyButton}"
BorderBrush="{x:Null}" Content="" HorizontalAlignment="Right"
VerticalAlignment="Bottom" Width="75" Height="65" Click="btnSalir_Click"
Margin="0,0,30,30">
                <Button.Background>
                    <ImageBrush ImageSource="Images/LogOut.png"/>
                </Button.Background>
            </Button>
            <Label Content="Centro de Investigación y de Estudios Avanzados del
Instituto Politécnico Nacional" HorizontalAlignment="Center" VerticalAlignment="Center"
Margin="385,51,382,678"   FontSize="18" Height="44" Width="709" FontWeight="Bold"/>


            <Image Source="Images/cinves.png" x:Name="image"  Height="140"
Margin="668,193,668,440"  Width="140" RenderTransformOrigin="0.671,0.221" />
            <Label Content="Entrenador para cirugías de columna"
HorizontalAlignment="Center" VerticalAlignment="Center" Margin="550,95,550,634"
FontSize="18" Height="44" Width="342" FontWeight="Bold"/>
```

```
        </Grid>
    </Border>
</Page>
```

Apéndice D - Programa correspondiente al menú registro usuario.

Programa en C

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.SQLite;
namespace SpineTrainer
{
    /// <summary>
    /// Interaction logic for Registro.xaml
    /// </summary>
    public partial class Registro : Page
    {
        string dbConnectionString = @"Data Source=SurgeryTrainer.sqlite;Version=3;";

        public Registro()
        {
            InitializeComponent();
        }

        private void RadioButton_Checked_Si(object sender, RoutedEventArgs e)
        {
            Gveces.IsEnabled = true;
        }

        private void RadioButton_Checked_No(object sender, RoutedEventArgs e)
        {
            Gveces.IsEnabled = false;
        }

        private void btnRegistrar_Click(object sender, RoutedEventArgs e)
        {

            SQLiteConnection sqliteCon = new SQLiteConnection(dbConnectionString);
            sqliteCon.Open();
            //string Consulta = "select usuario from registro";
            //verifica si ya existe el usuario
```

```
            string Consulta = "SELECT usuario from registro where usuario='" +
txbUsuario.Text + "'";
            SQLiteCommand Comando1 = new SQLiteCommand(Consulta, sqliteCon);
            SQLiteDataReader dr = Comando1.ExecuteReader();
            dr.Read();
            if (dr.StepCount > 0)
            {
                MessageBox.Show("El usuario ya existe");
                return;
            }
            // string prueba=dr.GetString(0);



            string cirugia;
            if (rbNo.IsChecked == true)
            {
                cirugia = "NO";
            }
            else
            {
                cirugia = "SI";
            }


            Consulta = "INSERT INTO REGISTRO VALUES ('" + txbUsuario.Text + "','" +
txbNombre.Text + "','"
               + txbApaterno.Text + "','" + txbAmaterno.Text + "','" +
txbInstituto.Text + "','"
               + cirugia + "'," + txbVeces.Text + ")";
            Comando1.Dispose();
            SQLiteCommand Comando = new SQLiteCommand(Consulta, sqliteCon);

            Comando.ExecuteNonQuery();
            //SQLiteDataReader dr = Comando.ExecuteReader();
            //dr.Read();
            //    string prueba=dr.GetString(0);
            sqliteCon.Close();

            if (txbNombre.Text == String.Empty || txbApaterno.Text == String.Empty ||
txbAmaterno.Text == String.Empty || txbInstituto.Text == String.Empty ||
txbUsuario.Text == String.Empty)
            {
                MessageBox.Show("All fields are required!", "Warning");
                return;
            }
            else
                MessageBox.Show("Se ha registrado correctamente");

            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.GoBack();
            }


        }

        private void Page_Loaded_1(object sender, RoutedEventArgs e)
```

```
        {
            rbNo.IsChecked = true;
        }

        private void btnRegresar_Click(object sender, RoutedEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.GoBack();
            }
        }


        private void btnSalir_Click(object sender, RoutedEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.GoBack();
            }
        }
    }
}
```

Programa en XAML

```
<Page x:Class="SpineTrainer.Registro"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="613.078" d:DesignWidth="543.816"
       Title="Registro" Loaded="Page_Loaded_1">
    <Border Background="LightBlue"
          BorderBrush="Black"
          BorderThickness="2"
          CornerRadius="45"
          Padding="25">

        <Grid  Background="White" HorizontalAlignment="Stretch">
            <Grid HorizontalAlignment="Center" VerticalAlignment="Center">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                    <RowDefinition Height="Auto"></RowDefinition>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
```

```
                    <ColumnDefinition Width="Auto"></ColumnDefinition>
                    <ColumnDefinition Width="Auto"></ColumnDefinition>
                </Grid.ColumnDefinitions>

                <Label Content="Registro de Nuevo Usuario" HorizontalAlignment="Left"
VerticalAlignment="Top"/>
                <Label Grid.Row="1" Grid.Column="0" Content="Nombre:"
HorizontalAlignment="Center"  VerticalAlignment="Center"/>
                <TextBox Grid.Row="1" Grid.Column="1" x:Name="txbNombre"
HorizontalAlignment="Center" Height="23" TextWrapping="Wrap" VerticalAlignment="Center"
Width="231" TabIndex="0"/>

                <Label Grid.Row="2" Grid.Column="0" Content="Apellido Paterno:"
HorizontalAlignment="Center"  VerticalAlignment="Center" Width="106"/>
                <TextBox  Grid.Row="2" Grid.Column="1" x:Name="txbApaterno"
HorizontalAlignment="Center" Height="23"  TextWrapping="Wrap"
VerticalAlignment="Center" Width="231" TabIndex="1"/>

                <Label Grid.Row="3" Grid.Column="0" Content="Apellido Materno:"
HorizontalAlignment="Center"  VerticalAlignment="Center"/>
                <TextBox Grid.Row="3" Grid.Column="1" x:Name="txbAmaterno"
HorizontalAlignment="Center" Height="23" TextWrapping="Wrap" VerticalAlignment="Center"
Width="231" TabIndex="2"/>

                <Label Grid.Row="5" Grid.Column="0"  Content="Usuario:"
HorizontalAlignment="Center"  VerticalAlignment="Center"/>
                <TextBox Grid.Row="5" Grid.Column="1"  x:Name="txbUsuario"
HorizontalAlignment="Center" Height="23" TextWrapping="Wrap" VerticalAlignment="Center"
Width="231" TabIndex="4"/>

                <Label Grid.Row="4" Grid.Column="0" Content="Instituto:"
HorizontalAlignment="Center"  VerticalAlignment="Center"/>
                <TextBox  Grid.Row="4" Grid.Column="1" x:Name="txbInstituto"
HorizontalAlignment="Center" Height="23"  TextWrapping="Wrap"
VerticalAlignment="Center" Width="231" TabIndex="3"/>

                <Label  Grid.Row="6" Grid.Column="0" Content="¿Ha realizado cirugías de
columna?" HorizontalAlignment="Center"  VerticalAlignment="Center"/>
                <Grid Grid.Row="6" Grid.Column="1">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition />
                        <ColumnDefinition />
                    </Grid.ColumnDefinitions>
                    <RadioButton   Grid.Column="0" x:Name="rbSi" Content="Si"
HorizontalAlignment="Center" VerticalAlignment="Center" GroupName="SINO"
Checked="RadioButton_Checked_Si" TabIndex="6"/>
                    <RadioButton  Grid.Column="1" x:Name="rbNo" Content="No"
HorizontalAlignment="Center"  VerticalAlignment="Center" GroupName="SINO"
Checked="RadioButton_Checked_No" TabIndex="5" />

                </Grid>
                <Grid IsEnabled="False" x:Name="Gveces" Grid.Row="7"
Grid.ColumnSpan="2">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition />
                        <ColumnDefinition />
                    </Grid.ColumnDefinitions>
```

```
                    <Label Grid.Column="0" Content="¿Cuántas veces?"
HorizontalAlignment="Center" Width="100"/>
                    <TextBox  Grid.Column="1" x:Name="txbVeces" Height="23"
TextWrapping="Wrap"  Text="0"/>



                </Grid>

                <Button  Grid.Row="8" Grid.Column="1" x:Name="btnRegistrar"
Content="Registrar" HorizontalAlignment="Center"  VerticalAlignment="Center" Width="75"
Click="btnRegistrar_Click" TabIndex="7"/>
            </Grid>

            <Button x:Name="btnRegresar" Style="{StaticResource MyButton}"
BorderBrush="{x:Null}" Content="" HorizontalAlignment="Right"
VerticalAlignment="Bottom" Width="57" Height="65" Click="btnRegresar_Click"
Margin="0,0,37,15" ToolTip="Regresar">
                <Button.Background>
                    <ImageBrush ImageSource="Images/back.png"/>
                </Button.Background>
            </Button>
        </Grid>
    </Border>
</Page>
```

Apéndice E - Programa correspondiente al menú principal.

Programa en C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using HelixToolkit.Wpf;
using System.Windows.Media.Media3D;
using System.Threading;
using System.Diagnostics;
using System.Windows.Threading;
using WiimoteLib;
```

```
namespace SpineTrainer
{
    /// <summary>
    /// Lógica de interacción para ColExpuesta.xaml
    /// </summary>
    public partial class ColExpuesta : Page
    {
        string usuario = "";


        //helps in handling some of the  threading issues
        Mutex m = new Mutex();

        //Variables para manejar wiimote
        WiimoteCollection mWC;
        Wiimote wm1;
        Wiimote wm2;

        //Variables para las coordenadas
        int x1 = 0;
        int y1 = 0;
        int x2 = 0;
        int y2 = 0;

        int x3 = 0;
        int y3 = 0;
        int x4 = 0;
        int y4 = 0;


        //One point each to track each IR sensor
        System.Windows.Point point1 = new System.Windows.Point(0, 0);
        System.Windows.Point point2 = new System.Windows.Point(0, 0);
        System.Windows.Point point3 = new System.Windows.Point(0, 0);
        System.Windows.Point point4 = new System.Windows.Point(0, 0);
        System.Windows.Point point5 = new System.Windows.Point(0, 0);
        System.Windows.Point point6 = new System.Windows.Point(0, 0);


        System.Windows.Point pPoint = new System.Windows.Point(0, 0);
        System.Windows.Point PO = new System.Windows.Point(0, 0);
        System.Windows.Point PO2 = new System.Windows.Point(0, 0);
        System.Windows.Point pLat = new System.Windows.Point(0, 0);



        // Variables declarations
        float PilaWiimote = new float();
        float PilaWiimote2 = new float();
        bool Sensor1Found = new bool();
        bool Sensor2Found = new bool();
        bool Sensor3Found = new bool();
        bool Sensor4Found = new bool();
        int LED1_Size = new int();
        int LED2_Size = new int();
        int LED3_Size = new int();
        int LED4_Size = new int();
```

```
bool MostrarLEDs = new bool();
bool MostrarLEDs2 = new bool();

//Stopwatch declarations
DispatcherTimer dt = new DispatcherTimer();
Stopwatch stopWatch = new Stopwatch();
string currentTime = string.Empty;

//variable declarations
double z = 0;




//Hilo para actualizar las coordenadas del wiimote
private delegate void UpdateWiimoteStateDelegate(WiimoteChangedEventArgs args);


public ColExpuesta(string value)
{
    InitializeComponent();
    usuario = value;

    //Stopwatch Event Handlers
    dt.Tick += new EventHandler(dt_Tick);
    dt.Interval = new TimeSpan(0, 0, 0, 0, 1);

}



void dt_Tick(object sender, EventArgs e)
{
    if (stopWatch.IsRunning)
    {
        TimeSpan ts = stopWatch.Elapsed;
        currentTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
            ts.Hours, ts.Minutes, ts.Seconds, ts.Milliseconds / 10);
        timelapse.Text = currentTime;
    }
}



protected override void OnMouseLeftButtonDown(MouseButtonEventArgs e)
{
    base.OnMouseLeftButtonDown(e);

    // Begin dragging the window
    var parentWindow = this.Parent as Window;
    if (parentWindow != null)
    {
        parentWindow.DragMove();
    }


}
```

```
        private void Window_Loaded_1(object sender, RoutedEventArgs e)
        {

        }


        private void btnInicia_Click(object sender, RoutedEventArgs e)
        {


            //Stopwatch Starts
            stopWatch.Start();
            dt.Start();



            mWC = new WiimoteCollection();


            try
            {
                mWC.FindAllWiimotes();
            }
            catch (WiimoteNotFoundException ex)
            {
                MessageBox.Show(ex.Message, "Wiimote not found error");
            }
            catch (WiimoteException ex)
            {
                MessageBox.Show(ex.Message, "Wiimote error");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Unknown error");
            }

            // Connects & set leds of each wiimote
            wm1 = mWC[0];
            wm1.Connect();
            wm1.SetLEDs(1);
            MostrarLEDs = true;
            wm1.WiimoteChanged += wm_OnWiimoteChanged1;
            wm1.SetReportType(InputReport.IRExtensionAccel, IRSensitivity.Maximum,
true);


            wm2 = mWC[1];
            wm2.Connect();
            wm2.SetLEDs(2);
            MostrarLEDs2 = true;
            wm2.WiimoteChanged += wm_OnWiimoteChanged2;
            wm2.SetReportType(InputReport.IRExtensionAccel, IRSensitivity.Maximum,
true);

            //WPF annimation loop,
            //Rendering event, for visual updates
```

```
            //CompositionTarget.Rendering += new
EventHandler(CompositionTarget_Rendering);
            CompositionTarget.Rendering += new
EventHandler(CompositionTarget_Rendering);

        }




        void wm_OnWiimoteChanged1(object sender, WiimoteChangedEventArgs e)
        {
            //comentar mutex
            //m.WaitOne();
            try
            {
                Dispatcher.Invoke(new
UpdateWiimoteStateDelegate(UpdateWiimoteChanged1), e);
            }
            catch
            { }
            //m.ReleaseMutex();



        }

        void wm_OnWiimoteChanged2(object sender, WiimoteChangedEventArgs e)
        {
            //m.WaitOne();
            try
            {
                Dispatcher.Invoke(new
UpdateWiimoteStateDelegate(UpdateWiimoteChanged2), e);
            }
            catch
            {
            }
            //m.ReleaseMutex();
        }

        private void btnTermina_Click(object sender, RoutedEventArgs e)
        {


            wm1.Disconnect();
            wm2.Disconnect();
        }

        void CompositionTarget_Rendering(object sender, EventArgs e)
        {

            m.WaitOne();

            movePoints();
            displayWiiData();
```

```
    m.ReleaseMutex();
}

private void UpdateWiimoteChanged1(WiimoteChangedEventArgs args)
{



    try
    {
        GC.WaitForPendingFinalizers();
        WiimoteState ws = args.WiimoteState;


            x1 = ws.IRState.IRSensors[0].RawPosition.X;
            y1 = ws.IRState.IRSensors[0].RawPosition.Y;
            Sensor1Found = ws.IRState.IRSensors[0].Found;
            LED1_Size = ws.IRState.IRSensors[0].Size;

            x2 = ws.IRState.IRSensors[1].RawPosition.X;
            y2 = ws.IRState.IRSensors[1].RawPosition.Y;
            Sensor2Found = ws.IRState.IRSensors[1].Found;
            LED2_Size = ws.IRState.IRSensors[1].Size;


            PilaWiimote = ws.Battery;
    }

    catch
    {
    }



}



private void UpdateWiimoteChanged2(WiimoteChangedEventArgs args)
{
    try
    {

        GC.WaitForPendingFinalizers();
        WiimoteState ws2 = args.WiimoteState;

        PilaWiimote2 = ws2.Battery;
        x3 = ws2.IRState.IRSensors[0].RawPosition.X;
        y3 = ws2.IRState.IRSensors[0].RawPosition.Y;
        Sensor3Found = ws2.IRState.IRSensors[0].Found;
        LED3_Size = ws2.IRState.IRSensors[0].Size;

        x4 = ws2.IRState.IRSensors[1].RawPosition.X;
        y4 = ws2.IRState.IRSensors[1].RawPosition.Y;
        Sensor4Found = ws2.IRState.IRSensors[4].Found;
        LED4_Size = ws2.IRState.IRSensors[1].Size;
```

```
                xDist.Text = Convert.ToString(LED3_Size);
                yDist.Text = Convert.ToString(LED4_Size);


            }

            catch
            {

            }


        }

        public System.Windows.Point ConvertForScreen(System.Windows.Point targetPoint,
double targetWidth, double targetHeight)
        {
            double dRawX = Convert.ToDouble(targetPoint.X);
            double dRawY = Convert.ToDouble(targetPoint.Y);
            double finalX = ((dRawX * targetWidth) / 1023);
            double finalY = ((dRawY * targetHeight) / 767);
            System.Windows.Point finalPoint = new System.Windows.Point(finalX, finalY);

            return finalPoint;
        }




        private void movePoints()
        {

            m.WaitOne();



            if (MostrarLEDs )
            {

                if (Sensor1Found)
                {
                    VisiblePoint1.Visibility = Visibility.Visible;

                }
                else
                {
                    VisiblePoint1.Visibility = Visibility.Hidden;


                }

                if (Sensor2Found)
                {
                    VisiblePoint2.Visibility = Visibility.Visible;

                }
                else
                {
```

```
                VisiblePoint2.Visibility = Visibility.Hidden;

            }


        }

        else
        {

            VisiblePoint1.Visibility = Visibility.Hidden;
            VisiblePoint2.Visibility = Visibility.Hidden;

        }



        point1.X = Convert.ToDouble(x1);
        point1.Y = Convert.ToDouble(y1);

        point2.X = Convert.ToDouble(x2);
        point2.Y = Convert.ToDouble(y2);



        point3.X = Convert.ToDouble(x3);
        point3.Y = Convert.ToDouble(y3);

        point4.X = Convert.ToDouble(x4);
        point4.Y = Convert.ToDouble(y4);


        ////////////////////////////////////////////
        // Point 1
        Point1X_Text.Text = "" + (Convert.ToString(point1.X)) + ", ";
        Point1Y_Text.Text = Convert.ToString(point1.Y);

        System.Windows.Point point_convert = ConvertForScreen(point1,
LEDCanvas.ActualWidth, LEDCanvas.ActualHeight);

        VisiblePoint1.SetValue(Canvas.LeftProperty, 1023 - point_convert.X);
        VisiblePoint1.SetValue(Canvas.TopProperty, 767 - point_convert.Y);


        int circleSize1 = LED1_Size * 12;
        double circle1Radius = Convert.ToDouble(circleSize1 / 2);

        Point1Circle.Width = Convert.ToDouble(circleSize1);
        Point1Circle.Height = Convert.ToDouble(circleSize1);
        Point1Circle.CornerRadius = new CornerRadius(circle1Radius);


        ////////////////////////////////////////////
        // Point 2
        Point2X_Text.Text = "" + (Convert.ToString(point2.X)) + ", ";
        Point2Y_Text.Text = Convert.ToString(point2.Y);
```

```
            point_convert = ConvertForScreen(point2, LEDCanvas.ActualWidth,
LEDCanvas.ActualHeight);

            VisiblePoint2.SetValue(Canvas.LeftProperty, 1023 - point_convert.X);
            VisiblePoint2.SetValue(Canvas.TopProperty, 767 - point_convert.Y);


            int circleSize2 = LED2_Size * 12;
            double circle2Radius = Convert.ToDouble(circleSize2 / 2);

            Point2Circle.Width = Convert.ToDouble(circleSize2);
            Point2Circle.Height = Convert.ToDouble(circleSize2);
            Point2Circle.CornerRadius = new CornerRadius(circle2Radius);



            // DEFINICIONES PARA DIBUJAR LINEA AXIAL //
            // METODO 1 Triangulos similares y distancia focal
            //Calculate the distance between LEDs
            double d = 0;
            d = Math.Sqrt(Math.Pow(point2.X - point1.X, 2) + Math.Pow(point2.Y -
point1.Y, 2));
            //distleds.Text = d.ToString();
            //distance.Text = Convert.ToString(d);

            //Calculate coordinates x y z
            // double z=0;


            z = (1369 * 2) / d; // 3 = distancia Real en cm entre los LEDs; 1369
distancia focal de la camara
                                //zDist.Text = z.ToString();
            zDist.Text = Convert.ToString(Math.Round(z, 2));

            if (string.IsNullOrWhiteSpace(this.zDist.Text))
            {
                zDist.Text = "null";
            }


            double x = 0;
            x = (z * (point1.X + point2.X)) / (2 * (1369));
            //xDist.Text = x.ToString();
            xDist.Text = Convert.ToString(Math.Round(x, 2));

            double y = 0;
            y = (z * (point1.Y + point2.Y)) / (2 * (1369));
            //yDist.Text = y.ToString();
            yDist.Text = Convert.ToString(Math.Round(y, 2));

            //xSP.Text = point1.X.ToString();
            //ySP.Text = point1.Y.ToString();

            ////////////////////////////////////////////


            //METODO 2 FOV por pixel y teorema de pitagoras
```

```
            /*
            //Angular FOV per pixel
            double aFOV = (Math.PI * 0.040) / 180;
            //angle.Text = Convert.ToString(aFOV);

            //Calculate the distance between LEDs px
            double d = 0;
            d = Math.Sqrt(Math.Pow(point2.X - point1.X, 2) + Math.Pow(point2.Y -
point1.Y, 2));
            //distance.Text = Convert.ToString(d);

            //Calculate the depth "z"

            double z = 3 / (2 * Math.Tan((aFOV * d) / 2)); // 3 =  cm entre leds
            zDist.Text = Convert.ToString(z);
            */

            ////////////////////////////////////////////
            // Math operations to calculate the angle


            // distance from p2 to pref
            double cDist = Math.Abs(point1.Y - point2.Y);


            //Distance from p1 to pref
            double bDist = Math.Abs(point2.X - point1.X);


            // Calculate the angle using tan


            double alfa = Math.Atan(cDist / bDist);
            double ang1 = 0;

            if (point2.X > point1.X)
            {

                ang1 = (alfa * 180) / Math.PI;
                txbAngle.Text = Convert.ToString(Math.Round(ang1, 2));
                if (string.IsNullOrEmpty(txbAngle.Text))
                {
                    txbAngle.Text = " ";
                }

            }

            else
            {
                ang1 = 180 - ((alfa * 180) / Math.PI);
                txbAngle.Text = Convert.ToString(Math.Round(ang1));

            }

            if (string.IsNullOrEmpty(txbAngle.Text))
            {
                txbAngle.Text = " ";
            }
```

```
            //Calculate Needle Depth

            double instPix = (d * 15) / 3;  //15 longitud en cm del instrumental


            // Calculate instrumental  position


            double cA = Math.Cos(alfa) * instPix;
            double cO = Math.Sin(alfa) * instPix;



            if (point1.X < point2.X)
            {
                PO.X = point2.X - cA;
            }

            if (point1.X >= point2.X)
            {
                PO.X = point2.X + cA;
            }

            PO.Y = point2.Y - cO;

            // Hacer vibrar los controles en caso de que el instrumental perfore en la
medula espinal

            // Cuadro
            /*
            System.Windows.Point[] poly = new System.Windows.Point[4];
            poly[0] = new System.Windows.Point(571, 405);
            poly[1] = new System.Windows.Point(555, 454);
            poly[2] = new System.Windows.Point(448, 455);
            poly[3] = new System.Windows.Point(430, 416);
            */

             // Triangulo
            System.Windows.Point[] poly = new System.Windows.Point[3];
            poly[0] = new System.Windows.Point(571, 405);
            poly[1] = new System.Windows.Point(512, 470);
            poly[2] = new System.Windows.Point(430, 416);



             if (IsInPolygon(poly, PO))
             {
                 //wm1.SetRumble(true);
                 wm2.SetRumble(true);
             }

             else
             {
                 //wm1.SetRumble(false);
                 wm2.SetRumble(false);
```

```
             }



            double refx2 = 0;
            if (point1.X > point2.X)
            {
                refx2 = 511 - cA;
                // xSP.Text = refx2.ToString();
            } //se desplaza a los lados desde el centro
            else
            {
                refx2 = 511 + cA;
                //xSP.Text = refx2.ToString();
            }
            double refy2 = 767 - cO; //se desplaza hacia abajo desde la coordenada
superior
                                     //ySP.Text = refy2.ToString();




            if (refx2 >= 0 && refx2 <= 1023 && refy2 >= 0 && refy2 <= 767)
            {


                // LINEA AXIAL ///
                System.Windows.Point p1 = new System.Windows.Point(point1.X, point1.Y);
                System.Windows.Point p2 = new System.Windows.Point(PO.X, PO.Y);

               // xSP.Text = PO.X.ToString();
               // ySP.Text = PO.Y.ToString();

                point_convert = ConvertForScreen(p1, CanvasVistaAxial.ActualWidth,
CanvasVistaAxial.ActualHeight);
                point_convert = ConvertForScreen(p2, CanvasVistaAxial.ActualWidth,
CanvasVistaAxial.ActualHeight);


                Line l1 = new Line();
                l1.X1 = 1024 - p1.X;
                l1.Y1 = 768 - p1.Y;
                l1.X2 = 1024 - p2.X;
                l1.Y2 = 768 - p2.Y;
                l1.Stroke = System.Windows.Media.Brushes.Red;
                l1.StrokeThickness = 2;
                CanvasVistaAxial.Children.Clear();
                CanvasVistaAxial.Children.Add(l1);



            }

            // Hacer vibrar los controles en caso de que el instrumental perfore en la
medula espinal
```

```
            // Cuadro
            /*
            System.Windows.Point[] poly = new System.Windows.Point[4];
            poly[0] = new System.Windows.Point(571, 405);
            poly[1] = new System.Windows.Point(555, 454);
            poly[2] = new System.Windows.Point(448, 455);
            poly[3] = new System.Windows.Point(430, 416);
            */

            /* // Triangulo
            System.Windows.Point[] poly = new System.Windows.Point[3];
            poly[0] = new System.Windows.Point(571, 405);
            poly[1] = new System.Windows.Point(512, 470);
            poly[2] = new System.Windows.Point(430, 416);
            */

           /*
            if (IsInPolygon(poly, PO))
            {
                //wm1.SetRumble(true);
                wm2.SetRumble(true);
            }

            else
            {
                //wm1.SetRumble(false);
                wm2.SetRumble(false);
            }
            */

            // DEFINICIONES PARA DIBUJAR LINEA LATERAL //
            double d2 = 0;
                d2 = Math.Sqrt(Math.Pow(point4.X - point3.X, 2) + Math.Pow(point4.Y -
point3.Y, 2));

                //Calculate coordinates x y z
                double z2=0;
                z2 = (1369 * 2) / d2; // 3 = distancia Real en cm entre los LEDs; 1369
distancia focal de la camara
                                      //zDist.Text = z.ToString();

                // distance from p2 to pref
                double cDist2 = Math.Abs(point3.Y - point4.Y);

                //Distance from p1 to pref
                double bDist2 = Math.Abs(point4.X - point3.X);

                // Calculate the angle using tan

                double alfa2 = Math.Atan(cDist2 / bDist2);
                double ang2 = 0;

                if (point4.X > point3.X)
                {

                    ang2 = (alfa2 * 180) / Math.PI;
                    txbAngleLat.Text = Convert.ToString(Math.Round(ang2, 2));
```

```
        }

        else
        {
            ang2 = 180 - ((alfa2 * 180) / Math.PI);
            txbAngleLat.Text = Convert.ToString(Math.Round(ang2));

        }



        //Calculate Needle Depth

        double instPix2 = (d2 * 15) / 3;  //15 longitud en cm del instrumental


        // Calculate instrumental  position


        double cA2 = Math.Cos(alfa2) * instPix2;
        double cO2 = Math.Sin(alfa2) * instPix2;



        if (point3.X < point4.X)
        {
            PO2.X = point4.X - cA2;
        }

        if (point3.X >= point4.X)
        {
            PO2.X = point4.X + cA2;
        }

        PO2.Y = point4.Y - cO2;


        double refx4 = 0;
        if (point3.X > point4.X)
        {
            refx4 = 511 - cA2;

        }
        else
        {
            refx4 = 511 + cA2;

        }

        double refy4 = 767 - cO2;

    if (refx4 >= 0 && refx4 <= 1023 && refy4 >= 0 && refy4 <= 767)

    {
        // LINEA LATERAL ///
```

```
                System.Windows.Point p3 = new System.Windows.Point(point4.X, point4.Y);
                System.Windows.Point p4 = new System.Windows.Point(PO2.X, PO2.Y);

                point_convert = ConvertForScreen(p3, CanvasVistaLateral.ActualWidth,
CanvasVistaLateral.ActualHeight);
                point_convert = ConvertForScreen(p4, CanvasVistaLateral.ActualWidth,
CanvasVistaLateral.ActualHeight);


                Line l2 = new Line();
                l2.X1 =1024 - p3.X;
                l2.Y1 = 767 - p3.Y;
                l2.X2 = 1024 - p4.X;
                l2.Y2 = 767 - p4.Y;
                l2.Stroke = System.Windows.Media.Brushes.Red;
                l2.StrokeThickness = 2;


                CanvasVistaLateral.Children.Clear();
                CanvasVistaLateral.Children.Add(l2);



                System.Windows.Point p5 = new System.Windows.Point(point1.X, point1.Y);
                System.Windows.Point p6 = new System.Windows.Point(PO2.X, PO2.Y);

                point_convert = ConvertForScreen(p5, CanvasVistaPA.ActualWidth,
CanvasVistaPA.ActualHeight);
                point_convert = ConvertForScreen(p6, CanvasVistaPA.ActualWidth,
CanvasVistaPA.ActualHeight);


                Line l3 = new Line();
                l3.X1 = 1024 - p5.X;
                l3.Y1 = 767 - p5.Y;
                l3.X2 = 1024 - p6.X;
                l3.Y2 = 767 - p6.Y;
                l3.Stroke = System.Windows.Media.Brushes.Red;
                l3.StrokeThickness = 2;


                CanvasVistaPA.Children.Clear();
                CanvasVistaPA.Children.Add(l3);



            }

            m.ReleaseMutex();
        }



        // Method for displaying wiidata
        private void displayWiiData()
        {

            BatteryProgress.Value = PilaWiimote;
```

```
            float BatteryMeter_Percent = (PilaWiimote * 100) / 256;
            BatteryMeter_Text.Text = "" + Convert.ToString(BatteryMeter_Percent) + "%";


            BatteryProgress2.Value = PilaWiimote2;
            float BatteryMeter_Percent2 = (PilaWiimote2 * 100) / 256;
            BatteryMeter_Text2.Text = "" + Convert.ToString(BatteryMeter_Percent2) +
"%";

        }



        // Actions for when CheckSP its checked
        private void CheckSP(object sender, RoutedEventArgs e)

        {

            // wm.SetRumble(true);



            System.Windows.Point sPoint = new System.Windows.Point(0, 0);
            // sPoint.X = midPoint.X;
            //sPoint.Y = midPoint.Y;

            //////
            sPoint.X = PO.X;
            sPoint.Y = PO.Y;

           xSP.Text = sPoint.X.ToString();
           ySP.Text = sPoint.Y.ToString();

            /////

            // calculate deep of the needle //

            //Calculate the distance between LEDs
            double p2p;
            p2p = Math.Sqrt(Math.Pow(sPoint.X - PO.X, 2) + Math.Pow(sPoint.Y - PO.Y,
2));
            txbProfundidad.Text = p2p.ToString();
        }


        // Actions for UnCcheckSP
        private void UnCheckSP(object sender, RoutedEventArgs e)

        {

            // wm.SetRumble(false);
            //LEDCanvas.Children.Clear();
        }


        // Actions to make normal mode
        private void MakeNormalMode(object sender, RoutedEventArgs e)
        {
```

```
            ShowRawPointsCheck.IsEnabled = true;
            MostrarLEDs = ShowRawPointsCheck.IsChecked.Value;
        }


        // Actions to make visible leds
        private void MakeRawPointsVisible(object sender, RoutedEventArgs e)
        {
            MostrarLEDs = true;
        }

        // Actions to hide leds
        private void HideRawPoints(object sender, RoutedEventArgs e)
        {
            MostrarLEDs = false;
        }



        public static bool IsInPolygon(System.Windows.Point[] poly,
System.Windows.Point p)
        {
            System.Windows.Point p1, p2;
            bool inside = false;


            if (poly.Length < 3)
            {
                return inside;
            }

            var oldPoint = new System.Windows.Point(poly[poly.Length - 1].X,
poly[poly.Length - 1].Y);


            for (int i = 0; i < poly.Length; i++)
            {
                var newPoint = new System.Windows.Point(poly[i].X, poly[i].Y);


                if (newPoint.X > oldPoint.X)
                {
                    p1 = oldPoint;

                    p2 = newPoint;
                }

                else
                {
                    p1 = newPoint;

                    p2 = oldPoint;
                }

                if ((newPoint.X < p.X) == (p.X <= oldPoint.X) && (p.Y - (long)p1.Y) *
(p2.X - p1.X) < (p2.Y - (long)p1.Y) * (p.X - p1.X))
                {
```

```
                    inside = !inside;
                }

                oldPoint = newPoint;
            }
            return inside;
        }




        // Actions for btnSalir
        private void btnSalir_Click(object sender, RoutedEventArgs e)
        {
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.GoBack();
            }
        }
    }
}
```

Programa en XAML

```
<Page x:Class="SpineTrainer.ColExpuesta"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns:Microsoft_Windows_Themes="clr-
namespace:Microsoft.Windows.Themes;assembly=PresentationFramework.Aero"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="740.299" d:DesignWidth="641.791"
       Title="ColExpuesta">

    <Page.Resources>
        <LinearGradientBrush x:Key="ProgressBarBackground" EndPoint="1,0"
StartPoint="0,0">
            <GradientStop Color="#BABABA" Offset="0"/>
            <GradientStop Color="#C7C7C7" Offset="0.5"/>
            <GradientStop Color="#BABABA" Offset="1"/>
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarBorderBrush" EndPoint="0,1"
StartPoint="0,0">
            <GradientStop Color="#B2B2B2" Offset="0"/>
            <GradientStop Color="#8C8C8C" Offset="1"/>
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarGlassyHighlight" EndPoint="0,1"
StartPoint="0,0">
            <GradientStop Color="#50FFFFFF" Offset="0.5385"/>
            <GradientStop Color="#00FFFFFF" Offset="0.5385"/>
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarTopHighlight" EndPoint="0,1"
StartPoint="0,0">
```

```
            <GradientStop Color="#80FFFFFF" Offset="0.05"/>
            <GradientStop Color="#00FFFFFF" Offset="0.25"/>
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarIndicatorAnimatedFill" EndPoint="0,0"
StartPoint="-100,0" MappingMode="Absolute">
            <GradientStop Color="#00000000" Offset="0"/>
            <GradientStop Color="#FF000000" Offset="0.4"/>
            <GradientStop Color="#FF000000" Offset="0.6"/>
            <GradientStop Color="#00000000" Offset="1"/>
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarIndicatorDarkEdgeLeft" EndPoint="1,0"
StartPoint="0,0">
            <GradientStop Color="#0C000000" Offset="0"/>
            <GradientStop Color="#20000000" Offset="0.3"/>
            <GradientStop Color="#00000000" Offset="1"/>
        </LinearGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarIndicatorDarkEdgeRight" EndPoint="1,0"
StartPoint="0,0">
            <GradientStop Color="#00000000" Offset="0"/>
            <GradientStop Color="#20000000" Offset="0.7"/>
            <GradientStop Color="#0C000000" Offset="1"/>
        </LinearGradientBrush>
        <RadialGradientBrush x:Key="ProgressBarIndicatorLightingEffectLeft"
RelativeTransform="1,0,0,1,0.5,0.5" RadiusX="1" RadiusY="1">
            <GradientStop Color="#60FFFFC4" Offset="0"/>
            <GradientStop Color="#00FFFFC4" Offset="1"/>
        </RadialGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarIndicatorLightingEffect" EndPoint="0,0"
StartPoint="0,1">
            <GradientStop Color="#60FFFFC4" Offset="0"/>
            <GradientStop Color="#00FFFFC4" Offset="1"/>
        </LinearGradientBrush>
        <RadialGradientBrush x:Key="ProgressBarIndicatorLightingEffectRight"
RelativeTransform="1,0,0,1,-0.5,0.5" RadiusX="1" RadiusY="1">
            <GradientStop Color="#60FFFFC4" Offset="0"/>
            <GradientStop Color="#00FFFFC4" Offset="1"/>
        </RadialGradientBrush>
        <LinearGradientBrush x:Key="ProgressBarIndicatorGlassyHighlight" EndPoint="0,1"
StartPoint="0,0">
            <GradientStop Color="#90FFFFFF" Offset="0.5385"/>
            <GradientStop Color="#00FFFFFF" Offset="0.5385"/>
        </LinearGradientBrush>


        <LinearGradientBrush x:Key="IsActiveBrush" EndPoint="0.5,0.929"
StartPoint="0.5,0.071">
            <GradientStop Color="#FFD0E3FF" Offset="0"/>
            <GradientStop Color="#FF87B7FF" Offset="1"/>
        </LinearGradientBrush>
        <SolidColorBrush x:Key="IsActiveBorderBrush" Color="#FFF63333"/>


        <Style x:Key="Battery_Progress" TargetType="{x:Type ProgressBar}">
            <Setter Property="Foreground" Value="#01D328"/>
            <Setter Property="Background" Value="{StaticResource
ProgressBarBackground}"/>
            <Setter Property="BorderBrush" Value="{StaticResource
ProgressBarBorderBrush}"/>
```

```
            <Setter Property="BorderThickness" Value="1"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type ProgressBar}">
                        <Grid SnapsToDevicePixels="true" x:Name="Background">
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="0.95*"/>
                                <ColumnDefinition Width="0.05*"/>
                            </Grid.ColumnDefinitions>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="0.2*"/>
                                <RowDefinition Height="0.6*"/>
                                <RowDefinition Height="0.2*"/>
                            </Grid.RowDefinitions>
                            <Rectangle Fill="{TemplateBinding Background}" RadiusX="2"
RadiusY="2" Grid.ColumnSpan="1" Grid.RowSpan="3"/>
                            <Border Margin="1" Background="{StaticResource
ProgressBarGlassyHighlight}" CornerRadius="2" Grid.ColumnSpan="1" Grid.RowSpan="3"/>
                            <Border Margin="1" Background="{StaticResource
ProgressBarTopHighlight}" BorderBrush="#80FFFFFF" BorderThickness="1,0,1,1"
Grid.ColumnSpan="1" Grid.RowSpan="3"/>
                            <Rectangle Margin="1" x:Name="PART_Track"
Grid.ColumnSpan="1" Grid.RowSpan="3"/>
                            <Decorator HorizontalAlignment="Left" Margin="1,1,0,1"
x:Name="PART_Indicator" Grid.RowSpan="3">
                                <Grid x:Name="Foreground">
                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition MaxWidth="15"/>
                                        <ColumnDefinition Width="0.1*"/>
                                        <ColumnDefinition MaxWidth="15"/>
                                    </Grid.ColumnDefinitions>
                                    <Grid.RowDefinitions>
                                        <RowDefinition/>
                                        <RowDefinition/>
                                    </Grid.RowDefinitions>
                                    <Rectangle x:Name="Indicator"
Fill="{TemplateBinding Foreground}" Grid.ColumnSpan="3" Grid.RowSpan="2"/>
                                    <Rectangle x:Name="Animation"
Fill="{TemplateBinding Foreground}" Grid.ColumnSpan="3" Grid.RowSpan="2">
                                        <Rectangle.OpacityMask>
                                            <MultiBinding>
                                                <MultiBinding.Converter>

<Microsoft_Windows_Themes:ProgressBarHighlightConverter/>
                                                </MultiBinding.Converter>
                                                <Binding Source="{StaticResource
ProgressBarIndicatorAnimatedFill}"/>
                                                <Binding Path="ActualWidth"
ElementName="Background"/>
                                                <Binding Path="ActualHeight"
ElementName="Background"/>
                                            </MultiBinding>
                                        </Rectangle.OpacityMask>
                                    </Rectangle>
                                    <Rectangle Margin="1,1,0,1" x:Name="LeftDark"
Fill="{StaticResource ProgressBarIndicatorDarkEdgeLeft}" RadiusX="1" RadiusY="1"
Grid.RowSpan="2"/>
```

```
                                    <Rectangle Margin="0,1,1,1" x:Name="RightDark"
Fill="{StaticResource ProgressBarIndicatorDarkEdgeRight}" RadiusX="1" RadiusY="1"
Grid.Column="2" Grid.RowSpan="2"/>
                                    <Rectangle x:Name="LeftLight" Fill="{StaticResource
ProgressBarIndicatorLightingEffectLeft}" Grid.Column="0" Grid.Row="2"/>
                                    <Rectangle x:Name="CenterLight"
Fill="{StaticResource ProgressBarIndicatorLightingEffect}" Grid.Column="1"
Grid.Row="2"/>
                                    <Rectangle x:Name="RightLight"
Fill="{StaticResource ProgressBarIndicatorLightingEffectRight}" Grid.Column="2"
Grid.Row="2"/>
                                    <Border x:Name="Highlight1" Grid.ColumnSpan="3"
Grid.RowSpan="2" Background="{StaticResource ProgressBarIndicatorGlassyHighlight}"/>
                                    <Border x:Name="Highlight2" Grid.ColumnSpan="3"
Grid.RowSpan="2" Background="{StaticResource ProgressBarTopHighlight}"/>
                                </Grid>
                            </Decorator>
                            <Border BorderBrush="{TemplateBinding BorderBrush}"
BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="2"
Grid.ColumnSpan="1" Grid.RowSpan="3"/>
                            <Border HorizontalAlignment="Stretch" Margin="-1,0,0,0"
Width="Auto" Grid.Column="1" Grid.RowSpan="1" Grid.Row="1" BorderThickness="0,1,1,1"
CornerRadius="0,2,2,0">
                                <Border.Background>
                                    <LinearGradientBrush EndPoint="0,1"
StartPoint="0,0">
                                        <GradientStop Color="#FFA9A9A9" Offset="0"/>
                                        <GradientStop Color="#FF919191" Offset="1"/>
                                    </LinearGradientBrush>
                                </Border.Background>
                                <Border.BorderBrush>
                                    <LinearGradientBrush EndPoint="0,1"
StartPoint="0,0">
                                        <GradientStop Color="#FFAAAAAA" Offset="0"/>
                                        <GradientStop Color="#FF909090" Offset="1"/>
                                    </LinearGradientBrush>
                                </Border.BorderBrush>
                            </Border>
                        </Grid>
                        <ControlTemplate.Triggers>
                            <Trigger Property="Orientation" Value="Vertical">
                                <Setter Property="LayoutTransform"
TargetName="Background">
                                    <Setter.Value>
                                        <RotateTransform Angle="-90"/>
                                    </Setter.Value>
                                </Setter>
                                <Setter Property="LayoutTransform"
TargetName="PART_Track">
                                    <Setter.Value>
                                        <RotateTransform Angle="90"/>
                                    </Setter.Value>
                                </Setter>
                                <Setter Property="LayoutTransform"
TargetName="PART_Indicator">
                                    <Setter.Value>
                                        <RotateTransform Angle="90"/>
                                    </Setter.Value>
```

```
                                </Setter>
                                <Setter Property="LayoutTransform"
TargetName="Foreground">
                                    <Setter.Value>
                                        <RotateTransform Angle="-90"/>
                                    </Setter.Value>
                                </Setter>
                            </Trigger>
                            <Trigger Property="IsIndeterminate" Value="true">
                                <Setter Property="Visibility" TargetName="LeftDark"
Value="Collapsed"/>
                                <Setter Property="Visibility" TargetName="RightDark"
Value="Collapsed"/>
                                <Setter Property="Visibility" TargetName="LeftLight"
Value="Collapsed"/>
                                <Setter Property="Visibility" TargetName="CenterLight"
Value="Collapsed"/>
                                <Setter Property="Visibility" TargetName="RightLight"
Value="Collapsed"/>
                                <Setter Property="Visibility" TargetName="Indicator"
Value="Collapsed"/>
                            </Trigger>
                            <Trigger Property="IsIndeterminate" Value="false">
                                <Setter Property="Fill" TargetName="Animation"
Value="#80B5FFA9"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

    </Page.Resources>


    <Border Background="LightBlue"
          BorderBrush="Black"
          BorderThickness="2"
          CornerRadius="45"
          Padding="25">
        <Grid Background="White" ShowGridLines="False"
RenderTransformOrigin="0.463,0.459" Margin="-8,-9,8,9">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" MinWidth="221"/>
                <ColumnDefinition Width="380*"/>
                <ColumnDefinition Width="380*"/>
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left"
VerticalAlignment="Top" RenderTransformOrigin="0.253,0.581" Height="49" Width="182"
Margin="16,0,0,0" >
                <TextBlock FontSize="18" HorizontalAlignment="Center"
Margin="10,10,9,15"><Run Text="Registro de métricas"/></TextBlock>
            </StackPanel>

            <StackPanel Grid.Column="1" Grid.Row="0"  HorizontalAlignment="Stretch">
```

```
                <Label Content="Vista Axial" HorizontalAlignment="Center"
FontSize="16"/>
                <Viewbox>
                    <Canvas x:Name="CanvasVistaAxial"  VerticalAlignment="Top"
Height="768" Width="1024" Panel.ZIndex="1"    Margin="25,5"  >
                        <Canvas.Background>
                            <ImageBrush x:Name="ImgPA" ImageSource="Images/AXBW.jpg" />
                        </Canvas.Background>
                        <Canvas.LayoutTransform>
                            <ScaleTransform ScaleY="0.14" ScaleX="0.14" CenterX="2"
CenterY="2"/>
                        </Canvas.LayoutTransform>

                    </Canvas>
                </Viewbox>

                <Label Content="Vista Lateral" HorizontalAlignment="Center"
FontSize="16" Margin="130,0"/>
                <Viewbox >
                    <Canvas x:Name="CanvasVistaLateral"  VerticalAlignment="Bottom"
Width="1024" Height="768" Panel.ZIndex="1"  Margin="25,5"   >
                        <Canvas.Background>
                            <ImageBrush x:Name="ImgLateral"
ImageSource="Images/LatBW.jpg " />
                        </Canvas.Background>
                        <Canvas.LayoutTransform>
                            <ScaleTransform ScaleY="0.14" ScaleX="0.14" CenterX="2"
CenterY="2"/>
                        </Canvas.LayoutTransform>

                    </Canvas>
                </Viewbox>
            </StackPanel>

            <StackPanel Grid.Column="2" Grid.Row="0"  HorizontalAlignment="Stretch" >
                <Label Content="Vista Postero-anterior" HorizontalAlignment="Center"
FontSize="16"/>
                <Viewbox>
                    <Canvas x:Name="CanvasVistaPA"  VerticalAlignment="Top"
Width="1024" Height="768" Panel.ZIndex="1"  Margin="25,5"   >
                        <!-- Width="300" Height="200" -->
                        <Canvas.Background>
                            <ImageBrush x:Name="ImgAxial" ImageSource="Images/APBW.jpg
" />
                        </Canvas.Background>
                        <Canvas.LayoutTransform>
                            <ScaleTransform ScaleY="0.14" ScaleX="0.14" CenterX="2"
CenterY="2"/>
                        </Canvas.LayoutTransform>

                    </Canvas>
                </Viewbox>

                <Label Content="Instrumental Tracking" HorizontalAlignment="Center"
FontSize="16" Margin="130,0"/>

                <Viewbox>
```

```
                    <Canvas Background="#FFADADAD"  x:Name="LEDCanvas" Width="1024"
Height="768" Panel.ZIndex="1"  Margin="25,5"  VerticalAlignment="Bottom" >
                        <Canvas.LayoutTransform>
                            <ScaleTransform ScaleY="0.14" ScaleX="0.14" CenterX="2"
CenterY="2"/>
                        </Canvas.LayoutTransform>

                        <Grid Width="Auto" Height="Auto" x:Name="VisiblePoint2">

                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto" />
                                <RowDefinition Height="*"/>
                            </Grid.RowDefinitions>

                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="*"/>
                                <ColumnDefinition Width="0*"/>
                            </Grid.ColumnDefinitions>

                            <Border x:Name="Point2Circle" BorderThickness="1,1,1,1"
CornerRadius="20,20,20,20" BorderBrush="#FF000000" Background="#FFA30606"
Grid.RowSpan="1" Grid.ColumnSpan="2" Width="Auto"/>

                            <TextBlock Text="" TextWrapping="Wrap" Margin="0,0,0,0"
Grid.Row="1" x:Name="Point2X_Text" Grid.ColumnSpan="1" HorizontalAlignment="Right"/>

                            <TextBlock Text="" TextWrapping="Wrap" Margin="0,0,0,0"
x:Name="Point2Y_Text" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left"/>

                            <TextBlock Width="Auto" Height="Auto" Text="2"
TextWrapping="Wrap" HorizontalAlignment="Center" FontSize="18" FontWeight="Bold"
d:LayoutOverrides="Height" Margin="0,0,0,0" VerticalAlignment="Center"
Grid.ColumnSpan="2"/>

                        </Grid>

                        <Grid Width="Auto" Height="Auto" x:Name="VisiblePoint1">

                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto" />
                                <RowDefinition Height="*"/>
                            </Grid.RowDefinitions>

                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="*"/>
                                <ColumnDefinition Width="0*"/>
                            </Grid.ColumnDefinitions>

                            <!--    <Border x:Name="Point1Circle"
BorderThickness="1,1,1,1" CornerRadius="20,20,20,20" BorderBrush="#FF000000"
Background="#FF0E06A3" Grid.RowSpan="1" Grid.ColumnSpan="2"/> -->
                            <Border x:Name="Point1Circle" BorderThickness="1,1,1,1"
CornerRadius="20,20,20,20" BorderBrush="#FF000000" Background="#FFA30606"
Grid.RowSpan="1" Grid.ColumnSpan="2"/>

                            <TextBlock Text="" TextWrapping="Wrap" Margin="0,0,0,0"
Grid.Row="1" x:Name="Point1X_Text" Grid.ColumnSpan="1" HorizontalAlignment="Right"/>
```

```
                            <TextBlock Text="" TextWrapping="Wrap" Margin="0,0,0,0"
x:Name="Point1Y_Text" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left"/>

                            <TextBlock Width="Auto" Height="Auto" Text="1"
TextWrapping="Wrap" FontWeight="Bold" HorizontalAlignment="Center"
d:LayoutOverrides="Height" FontSize="18" VerticalAlignment="Center"
Grid.ColumnSpan="2"/>

                        </Grid>

                    </Canvas>

                </Viewbox>

            </StackPanel>

            <Label Content="Ángulo Axial" HorizontalAlignment="Left"
Margin="26,115,0,0" VerticalAlignment="Top" Height="33" Width="81"
RenderTransformOrigin="1.53,1.045"/>
            <TextBox x:Name="txbAngle" HorizontalAlignment="Left" Height="23"
Margin="26,148,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="81"
SelectionOpacity="-1"/>

            <Label Content="Ángulo Lateral" HorizontalAlignment="Left"
Margin="131,115,0,0" VerticalAlignment="Top" Height="33" Width="97"
RenderTransformOrigin="1.53,1.045"/>
            <TextBox x:Name="txbAngleLat" HorizontalAlignment="Left" Height="23"
Margin="131,148,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="81"
SelectionOpacity="-1"/>
            <Label Content="Coordenadas x, y, z en cm" HorizontalAlignment="Left"
Margin="26,192,0,0" VerticalAlignment="Top" Height="43" Width="159"/>
            <Label Content="Wiimote 1" HorizontalAlignment="Left" Margin="26,378,0,0"
VerticalAlignment="Top" Height="29" Width="64"/>
            <Label Content="Wiimote 2" HorizontalAlignment="Left" Margin="135,378,0,0"
VerticalAlignment="Top" Height="29" Width="71"/>

            <TextBox x:Name="xDist" HorizontalAlignment="Left" Height="23"
Margin="27,222,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="54"/>
            <TextBox x:Name="yDist" HorizontalAlignment="Left" Height="23"
Margin="93,222,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="54"/>
            <TextBox x:Name="zDist" HorizontalAlignment="Left" Height="23"
Margin="161,222,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="55"/>
            <Label Content="Profundidad de la aguja" HorizontalAlignment="Left"
Margin="26,257,0,0" VerticalAlignment="Top" Height="35" Width="165"/>
            <TextBox x:Name="txbProfundidad" HorizontalAlignment="Left" Height="22"
Margin="27,292,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
            <!--     <TextBox x:Name="distleds" HorizontalAlignment="Left" Height="22"
Margin="27,314,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>  -->
            <TextBox x:Name="timelapse" HorizontalAlignment="Left" Height="22"
Margin="27,581,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>



            <Border Margin="14,8,0,10" BorderBrush="#FF3D3D3D"
BorderThickness="1,1,1,1" CornerRadius="6,6,6,6">
                <StackPanel Height="Auto" Background="{x:Null}" Width="Auto"
Margin="4,4,4,4">
```

```
                    <Border Width="Auto" Height="Auto" Margin="4,4,4,4">
                    </Border>
                    <Border Width="Auto" Height="Auto" Margin="4,4,4,4">
                   </Border>
                    <Border Width="Auto" Height="Auto" Margin="4,4,4,4">

                        <Grid Width="Auto" Height="Auto" x:Name="ZAccelGrid"
Margin="0,5,0,5">
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width=".5*"/>
                                <ColumnDefinition Width="0.5*"/>
                            </Grid.ColumnDefinitions>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="20"/>
                            </Grid.RowDefinitions>

                        </Grid>
                    </Border>
                    <Grid Width="Auto" Height="Auto" Margin="0,5,0,5">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="111" />
                            <ColumnDefinition Width="102"/>
                            <ColumnDefinition Width="Auto" MinWidth="35" />
                        </Grid.ColumnDefinitions>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="333"/>
                            <RowDefinition Height="Auto"  MinHeight="67"  />
                            <RowDefinition Height="39.667"/>
                            <RowDefinition Height="120.333"/>
                        </Grid.RowDefinitions>

                        <ProgressBar Margin="8,8,8,8" Style="{DynamicResource
Battery_Progress}" HorizontalAlignment="Stretch" Grid.Row="1" MaxHeight="50"
x:Name="BatteryProgress" ToolTip="{Binding Value, ElementName=BatteryProgress,
Mode=Default}" Maximum="256"/>
                        <ProgressBar Margin="7,8,0,8" Style="{DynamicResource
Battery_Progress}" HorizontalAlignment="Stretch" Grid.Row="1" MaxHeight="50"
x:Name="BatteryProgress2" ToolTip="{Binding Value, ElementName=BatteryProgress,
Mode=Default}" Maximum="256" Grid.Column="1"/>

                        <ToggleButton Margin="11,0,20,38" Content="Starting Point"
VerticalAlignment="Bottom" Height="33" x:Name="SPBtn" Checked="CheckSP"
Unchecked="UnCheckSP"/>
                        <TextBlock Margin="10,0,27,0" VerticalAlignment="Top"
Height="40" Grid.Row="2" Text="" TextWrapping="Wrap" x:Name="BatteryMeter_Text"/>
                        <TextBlock Margin="18,0,10,0" VerticalAlignment="Top"
Height="40" Grid.Row="2" Text="" TextWrapping="Wrap" x:Name="BatteryMeter_Text2"
Grid.Column="1"/>
                        <TextBlock Margin="57,75,10,0" VerticalAlignment="Top"
Height="31" Grid.Column="1" Text="" TextWrapping="Wrap" x:Name="depth"/>
                        <TextBlock Margin="0,262,58,0" VerticalAlignment="Top"
Height="33" Grid.Column="1" Text="" TextWrapping="Wrap" x:Name="xSP"/>
                        <TextBlock Margin="59,262,0,0" VerticalAlignment="Top"
Height="33" Grid.Column="1" Text="" TextWrapping="Wrap" x:Name="ySP"
RenderTransformOrigin="-0.561,-2.462"/>
                        <Button x:Name="btnTermina" Style="{StaticResource MyButton}"
Content="" ToolTip="Termina" BorderBrush="{x:Null}" Click="btnTermina_Click"
```

```
Panel.ZIndex="1" Margin="85,-5,78,293" RenderTransformOrigin="0.938,0.46"
Grid.ColumnSpan="2">
                            <Button.Background>
                                <ImageBrush ImageSource="Images/stop.png"/>
                            </Button.Background>
                        </Button>
                        <Button x:Name="btnSalir" Style="{StaticResource MyButton}"
Content="" HorizontalAlignment="Left" VerticalAlignment="Top" Width="46" Height="45"
Click="btnSalir_Click" ToolTip="Salir" RenderTransformOrigin="1.109,0.7"
BorderBrush="{x:Null}" Margin="46,-5,0,0" Grid.Column="1">
                            <Button.Background>
                                <ImageBrush ImageSource="Images/Exit.png "/>
                            </Button.Background>
                        </Button>
                        <CheckBox HorizontalAlignment="Left" Width="107"
Content="Mostrar LEDs" IsChecked="True" VerticalAlignment="Center"
x:Name="ShowRawPointsCheck" Checked="MakeRawPointsVisible" Unchecked="HideRawPoints"
Height="16" Margin="10,23,0,81" Grid.Row="3" Grid.ColumnSpan="2"/>
                    </Grid>
                </StackPanel>
            </Border>
            <Grid Margin="10,8,6,8" Grid.Column="0" Grid.Row="0">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="94"/>
                    <ColumnDefinition Width="74"/>
                    <ColumnDefinition Width="54"/>
                    <ColumnDefinition Width="34"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="475*"/>
                    <RowDefinition Height="43*"/>
                    <RowDefinition Height="87*"/>
                    <RowDefinition Height="25*"/>
                    <RowDefinition Height="23*"/>
                    <RowDefinition Height="17"/>
                </Grid.RowDefinitions>

                <Button x:Name="btnInicia" Style="{StaticResource MyButton}" Content=""
Height="43" Width="48" ToolTip="Inicia"  BorderBrush="{x:Null}" Click="btnInicia_Click"
HorizontalAlignment="Left" VerticalAlignment="Top" VerticalContentAlignment="Stretch"
Margin="22,60,0,0">
                    <Button.Background>
                        <ImageBrush ImageSource="Images/play.png"/>
                    </Button.Background>
                </Button>

            </Grid>



        </Grid>
    </Border>
</Page>
```