Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO DE UN SISTEMA PARA LA MONITORIZACIÓN DE OPERACIONES MANUALES MEDIANTE SENSOR KINECT

Autor:

Ojosnegros Ramos, Eduardo

Tutor:

Gómez García-Bermejo, Jaime
Departamento de Ingeniería de Sistemas y Automática

Co-Tutor:

Zalama Casanova, Eduardo
Departamento de Ingeniería de Sistemas y Automática

Valladolid, Septiembre 2014.

Resumen

El presente Trabajo Fin de Grado consiste en el desarrollo de un Sistema para la Monitorización de Operaciones Manuales mediante el uso del sensor Kinect. Con este fin se realiza el seguimiento en tiempo real de la herramienta utilizada mediante el rastreo de marcadores de colores adheridos a la misma.

El sistema permite la identificación de cualquier tarea ejecutada por el operario guiándole en el correcto cumplimiento de la misma. De esta forma es posible la corrección de fallos en el momento evitando arrastrar el error en tareas posteriores. Esto se traduce en una enorme reducción de los gastos de producción del producto.

Debido a los requisitos de desarrollo de aplicaciones con el controlador de Kinect de Microsoft el sistema se desarrolla en Windows mediante aplicaciones Windows Presentation Foundation (WPF). El modelo tridimensional de la escena es implementado con el motor de videojuegos Unity.

Palabras Clave

Visión 3D, Kinect, guiado de operaciones, seguimiento de herramientas, control de calidad.

AGRADECIMIENTO

Deseo agradecer a mis padres y mi familia por todo su apoyo y por haberme dejado ocupar su garaje durante casi dos meses.

Gracias a ti Sandra por estar siempre a mi lado y por conseguir sacarme una sonrisa en los momentos complicados.

A mis compañeros y amigos por estos cuatro años llenos de risas y momentos inolvidables.

Índice General

Índice de Figuras

Índice de Tablas

1. Introducción

En este capítulo se describe una breve introducción al problema que pretende solventa este Trabajo de Fin de Grado y los objetivos que lo han motivado. Tras esta descripción se expone la estructura del documento.

1.1. Marco del Proyecto

El número de procedimientos manuales realizados en la industria todavía es elevado respecto a los realizados por robots. Amortizar la gran inversión que supone su instalación no está al alcance de pequeñas y medianas empresas (PYMES), ya sea por el poco volumen de producción o por la gran variedad de los productos que exige una continua programación de las trayectorias a seguir por el robot.

Existe una gran problemática en las operaciones manuales que trata de abordarse en este trabajo. Un primer aspecto es que muchas de estas operaciones deben ser realizadas por personal nuevo, eventual, y recién contratado que exigen las fluctuaciones de la producción. Este personal necesita una formación adecuada, para su rápida incorporación al puesto de trabajo, sin menoscabo de la calidad del proceso realizado. Por otra parte al realizarse las operaciones de forma manual no es posible realizar la trazabilidad de las operaciones, por lo que no es posible determinar quién, cuándo, cómo y en qué tiempo se ha realizado una determinada operación. Por otra parte, la tasa de elementos que no superan los controles de calidad son superiores a otros procesos automáticos de fabricación, por la propia condición humana por lo que se hace necesario incorporar sistemas de control de calidad.

El control común de calidad de operarios se basa en la supervisión de un encargado de las acciones realizadas por ellos. Esto crea tensiones entre los trabajadores debido a que las decisiones sobre quien tiene la culpa de una mala realización de una operación se basan en informes del todo subjetivos, ya que es imposible que el encargado esté en todo momento al tanto de todas las acciones realizadas por los operarios a su cargo. Las solución actual se basa en colocar cámaras por las instalaciones de la zona de trabajo. Esto atenta contra la privacidad de los trabajadores y además es necesario el almacenamiento de un gran volumen de datos.

La solución que este trabajo plantea no es la de controlar visualmente a los operarios si no realizar un control de calidad en línea durante el propio proceso de mecanizado que permita subsanar errores in situ, y al mismo tiempo facilite el trabajo al operario sabiendo que dispone de un sistema que le guía y alerta cuando una operación no la realiza de forma correcta.

Como método para el registro de las actividades llevadas a cabo por el operario se plantea el uso de visión tridimensional, también conocida como visión 3d. Se entiende como visión 3d la parte de la Visión Artificial que trata de comprender escenas del mundo real de forma automática, utilizando para ello, sus propiedades tridimensionales inferidas de imágenes digitales.

Las técnicas de adquisición de información tridimensional clásicas se basaban en la relación entre las coordenadas del mundo físico y el de la imagen digital. El proceso de formación de una imagen bidimensional es un proceso proyectivo del mundo real tridimensional al subespacio bidimensional de la imagen, en el que desaparece una dimensión, la de la profundidad. Para recuperar la tercera dimensión, es necesario conocer la relación entre las coordenadas de la imagen digital y las de la escena 3D proyectada en ella. Este era un proceso costoso que necesita el uso de al menos dos cámaras las cuales hay que calibrar para obtener correctamente la información tridimensional de la escena.

El 1 de junio de 2009 fue presentado el sensor Kinect. Originalmente este dispositivo fue diseñado para su uso exclusivo en la consola Xbox 360, pero no tardó en hacerse popular en su uso en el entorno industrial debido a la facilidad de su uso frente a las técnicas de visión 3d descritas. El sensor Kinect puede ser utilizado como un sensor de visión 3d que permite crear un modelo tridimensional de la escena capturada por sus sensores con un precio mucho menor que los sistemas tradicionales.

No se encuentran proyectos parecidos que incluyan el sensor Kinect para detectar operaciones manuales de montaje. Sin embargo se encuentran muchos proyectos diferentes que utilizan este sensor e información en la web sobre la programación de este sensor. El sistema a desarrollar se plantea como alternativa viable a la instalación de robots, diseñado de cara a su adquisición por PYMES o por autónomos, por lo que el bajo precio se ajusta hacia ese sector de la industria. Por todo esto el dispositivo Kinect será el punto de partida del sistema a desarrollar.

1.2. Objetivos

El objetivo principal de este trabajo es desarrollar un sistema que ayude al operario en la correcta ejecución de las operaciones manuales realizadas en la fabricación de un producto y al mismo tiempo sirva como control de calidad de los procesos llevados a cabo para conseguirlo.

Para lograr este objetivo principal se plantean una serie de subobjetivos:

- Realizar un estudio sobre las características del dispositivo Kinect.
- Elegir entre las diferentes alternativas de control del dispositivo Kinect.
- Analizar los ejemplos de uso más actuales de Kinect.

- Realizar un estudio sobre el lenguaje de programación en el que realizar las aplicaciones.
- Analizar como registrar las actividades realizadas por un operario en la realización de operaciones manuales.
- Desarrollar un programa que permita localizar las coordenadas tridimensionales en las que se encuentra un objeto.
- Estudiar las técnicas actuales de simulación de entornos tridimensionales.
- Construir una interfaz gráfica que permita al usuario interactuar con el ordenador para que le oriente en la correcta ejecución de las operaciones.
- Realizar pruebas de calibración y optimización para lograr una aplicación robusta.

1.3. Estructura del documento

Tras este capítulo introductorio se describirán las características y controladores actuales del dispositivo Kinect, junto a los programas de ejemplo y aplicaciones estudiadas para conocer su funcionamiento.

El capítulo 3 resume los conceptos de información cromática y geometría espacial utilizados para el correcto desarrollo del sistema. De esta forma el lector de este trabajo tiene la información suficiente para entender los capítulos posteriores.

El desarrollo del proyecto ha seguido el modelo de prototipo, en el que el sistema continuo de testeo de la aplicación hace que la tarea de diseño se haga en paralelo con la de implementación pudiendo requerir volver a comprobar tareas supuestamente cerradas. Es por esto por lo que en el capítulo 5, el cual resume la implementación del sistema, haga continuamente referencia a los apartados del capítulo 4, donde se explica la metodología de diseño del sistema.

Los resultados de calibración y los ejemplos de uso de la aplicación serán analizados en el capítulo 6.

Tras el cálculo del presupuesto del proyecto, capítulo 7, se sacan las conclusiones del proyecto analizando tanto el presupuesto como los objetivos y características del sistema desarrollado.

2. El dispositivo Kinect

En este capítulo se habla sobre el estudio realizado del dispositivo Kinect. El primer paso fue estudiar sus características principales y los sensores de color y profundidad. Más tarde se analizaron los ejemplos proporcionados por el SDK de Kinect, y distintos middleware y aplicaciones, a destacar NimbleSDK con la cual se implementó una aplicación.

2.1. Características del sensor Kinect

Kinect, originalmente conocido por el nombre en clave "Proyecto Natal", es un dispositivo de sensor de movimiento creado por la empresa Microsoft en colaboración con la empresa Rare y PrimeSense, originalmente desarrollado para la videoconsola Xbox 360. A diferencia de otros dispositivos Kinect permite controlar la consola sin necesitar contacto físico con ningún controlador de videojuegos tradicional, siendo el jugador quien interactúa con una interfaz natural de usuario que reconoce gestos, movimientos del cuerpo y comandos de voz para controlar la consola.

Componentes de Kinect:

Kinect es un dispositivo que integra un sensor de profundidad, una cámara a color y un set de micrófonos, un acelerómetro de tres ejes, un motor y un procesador PrimeSense que capta el entorno en 3 dimensiones en una caja de forma horizontal.

- **Cámara RGB:** sensor de imagen CMOS que almacena los datos de los tres canales que envían los sensores en una resolución máxima de 1280 x 960, lo cual hace posible capturar una imagen en color y enviar datos a una frecuencia máxima de 30fps. Permite capturar la resolución espacial, es decir captar las coordenadas de los ejes X e Y.
- **Set de Micrófonos:** conjunto de cuatro micrófonos que permite la localización de la fuente acústica y la eliminación del ruido ambiente, permitiendo además la posibilidad de grabar audio.
- **Motor:** permite el movimiento del sensor verticalmente.
- **Sensor 3D de Profundidad:** Es la combinación de un emisor de infrarrojos (IR) Láser y un sensor de imagen CMOS. El emisor emite haces de luz IR y el sensor lee los haces IR reflectados que vuelven al sensor. Se calcula la distancia de profundidad a partir de la comparación de la imagen recogida con una imagen patrón.
- **Acelerómetro de 3 ejes:** configurado para un rango 2G, donde G es la aceleración de la gravedad, determina la orientación actual de la Kinect. Esto permite estabilizar las imágenes cuando se mueve el sensor.

- **CHIP PRIMESENSE PS1080:** sirve para reconstruir una captura de movimiento 3D de la escena que esté ubicada al frente del Kinect, ya que este chip captura su entorno en tres dimensiones y transforma esas capturas a imágenes sincronizadas en 3D. Todos los elementos del sensor actúan en conjunto con el chip interno PrimeSense.
- **Memoria RAM de 512 Mb.**

x

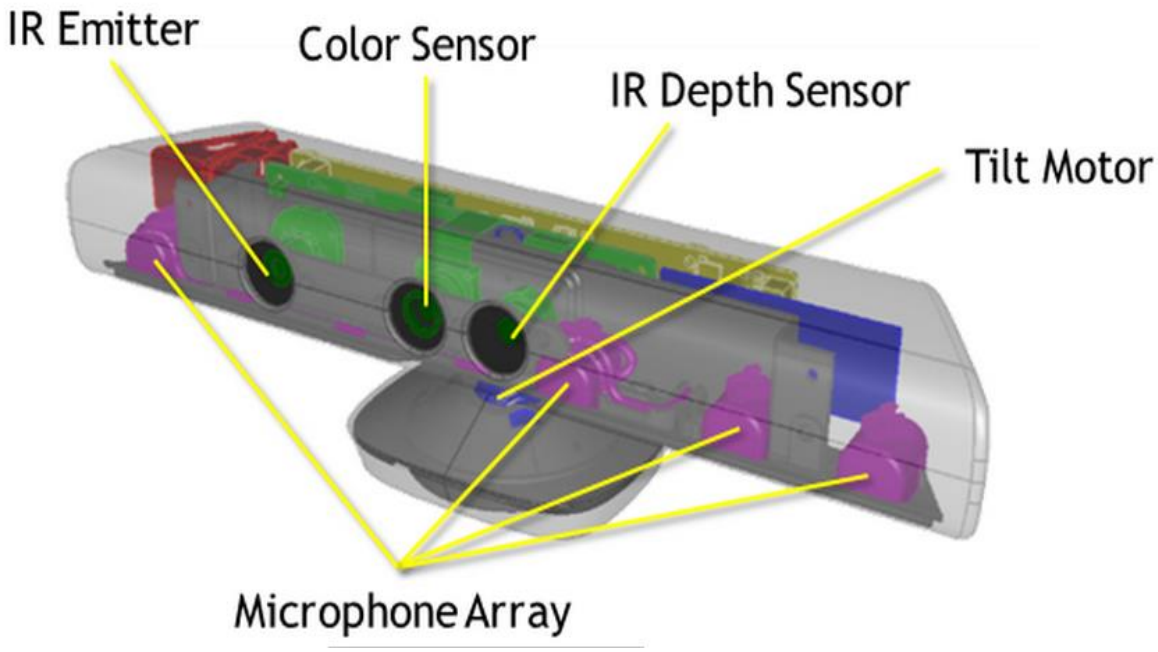


Figura 2.1 Componentes del sensor Kinect

Figura 2.2 El sensor Kinect por dentro

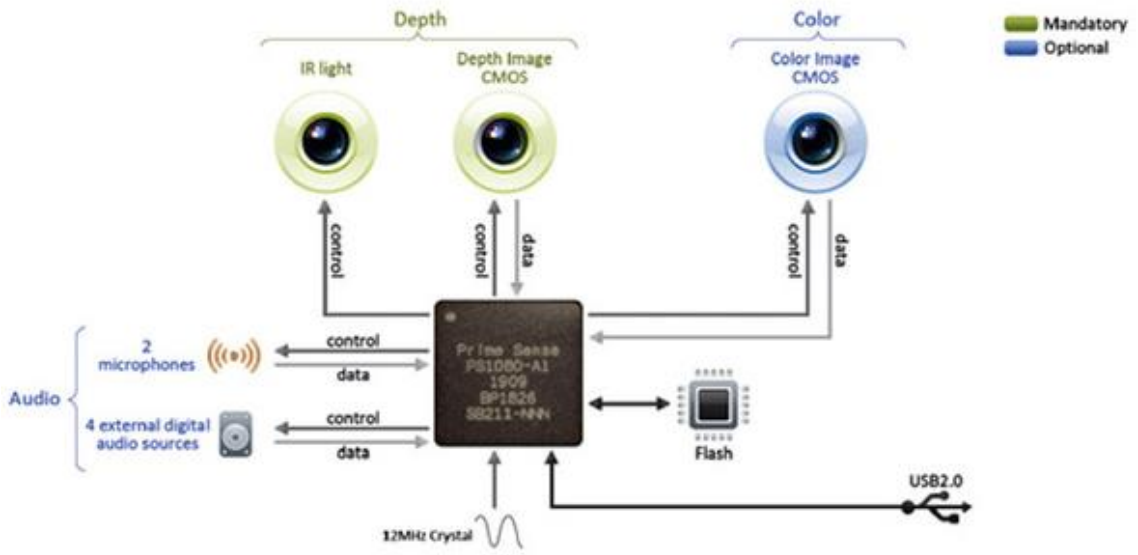


Figura 2.3 Arquitectura Kinect

Especificaciones de Kinect:

En la siguiente tabla se describirán las características de la Kinect utilizado en este proyecto, la Kinect para Windows:

Característica	Especificaciones
Campo de Visión	Vertical: 43° Horizontal: 57°
Rango de inclinación física	±27 grados
Características de entrada de Audio	Matriz de 4 micrófonos con un convertidor analógico-digital de 24-bit y un procesamiento de la señal que elimina el eco y el ruido.
Rango de Profundidad	Permite dos Modos: Default Mode y Near Mode. **Ver la figura 2.4.**
Flujo de Datos de Profundidad	80 x 60, 320 × 240, 640 x 480 @ 30fps
Flujo de Datos de Color	640 × 480 32-bit de color @30fps 1280 x 960 RGB @ 12fps Raw YUV 640 x 480 @ 15fps
Flujo de Datos de Audio	Audio de 32 y 64 bits, en función del SO.
Cantidad de sensores a usar en un equipo	Permite utilizar hasta 4 sensores a la vez en el mismo equipo bajo las siguientes premisas: - Cada sensor debe estar en un puerto USB diferente. - El PC debe tener potencia suficiente para poder soportar los 4 sensores.

Tabla 2.1 Especificaciones del sensor Kinect para Windows

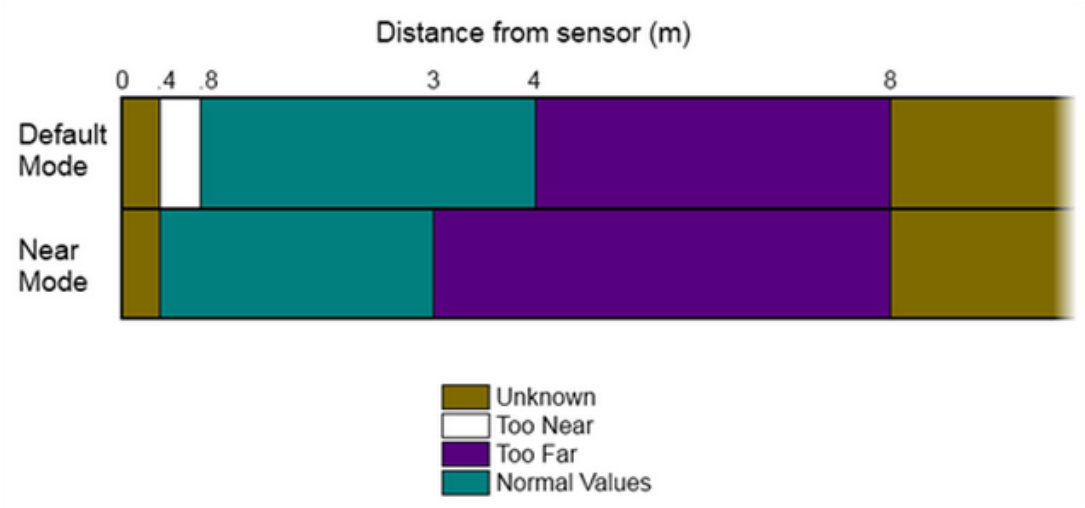


Figura 2.4 Modos de funcionamiento del sensor de profundidad

Historia de los Drivers de Kinect:

El sensor Kinect es una herramienta muy poderosa, pero necesita de un controlador que gestione las señales recibidas por el puerto USB.

Originalmente Kinect no estaba destinado a su uso fuera de la consola Xbox 360. En noviembre de 2010 a partir de ingeniería inversa y motivado por un concurso organizado por Adafruit Industries, Hector Martín fue el primero en lograr extraer la información que se obtiene de las cámaras de profundidad y RGB. Esto permitía controlar la Kinect desde un Pc. Héctor Martín se unió a la comunidad OpenKinect [1] creada por Josh Blake con la intención de reunir a todos los programadores interesados en la interfaz natural de usuario (natural user interfaces, NUIs). OpenKinect desarrolla y mantiene la biblioteca principal libfreenect para acceder a la cámara USB Kinect y crear aplicaciones para Windows, Linux y Mac.

En diciembre de 2010 PrimeSense creo la organización sin fines de lucro OpenNI, Iteración Natural Abierta. Esta organización fue creada para promover y certificar la compatibilidad e interoperabilidad de dispositivos de interacción natural (NI), aplicaciones y middleware. Con el fin de cumplir su objetivo, OpenNI lanzado un framework de código abierto llamado OpenNI Framework. Proporciona una API y middleware de alto nivel llamado NITE para implementar el seguimiento de la mano y esqueleto y el reconocimiento de gestos.

En junio de 2011 Microsoft lanzó el primer kit de desarrollo de software (SDK) de Kinect de uso no comercial para Windows, limitando las aplicaciones a su uso personal y dejando a fuera a las comunidades de programadores de Linux y Mac OSX.

Gracias a la gran cantidad de documentación, ejemplos, una habitual actualización de versión y varios libros de iniciación [2], [3], el SDK proporcionado por Microsoft empezó a ser utilizado por multitud de empresas y grupos de investigación y los colectivos empezaron a perder fuerza. Con la compra de Apple de la empresa PrimeSense en diciembre del 2013, la organización OpenNI que integraba gran cantidad de middleware y aplicaciones compatibles desaparece.

Actualmente los controladores de código abierto siguen siendo adquiribles en la web y la última versión actualizada del SDK de Windows es la v1.8.

2.2. Sensores de color y profundidad

Kinect es la base central de este proyecto. Es necesario disponer de un sensor de profundidad para poder ejecutar los programas que en este programa se desarrollan, y para ello hemos utilizado el sensor Kinect para Windows, una versión mejorada del primer sensor Kinect lanzado para Xbox 360, desarrollado especialmente para su uso en aplicaciones de Windows 7 y Windows 8.

De los sensores definidos en el apartado 2.1 en este proyecto se van a utilizar concretamente los sensores de color y profundidad. En este apartado se explicará cómo funcionan estos dos sensores.

La cámara a color

Esta cámara CMOS es la responsable de capturar y transmitir los datos de video. Su función es detectar los colores rojo, azul y verde de la escena. La cámara transmite estos datos como una sucesión de fotogramas. Los datos de color se pueden transmitir a una velocidad máxima de 30 fotogramas por segundo (FPS) en una resolución de 640*480 pixeles, y a 12 FPS a una resolución máxima de 1280*960 pixeles. Los valores de los fotogramas por segundo pueden variar dependiendo de la resolución usada por la transmisión de imagen. El rango de visión de la cámara Kinect es de 43 grados en vertical por 57 grados en horizontal.

La cámara a color es utilizada para capturar la resolución espacial, es decir captar las coordenadas de los ejes X e Y.

El sensor de Profundidad

El sistema Kinect permite construir un “mapa de profundidad” de un área frente a la cámara. Este mapa es producido completamente dentro del sensor y es trasmitido a un host por USB de la misma forma que la cámara a color, pero en este caso en vez de información de color por pixel en una imagen, el sensor transmite valores de distancia.

Se podría pensar que el sensor de profundidad usa algún tipo de radar de ultrasonidos para transmitir las medidas, pero esto no es así. Esto sería muy difícil

de hacer a cortas distancias. En su lugar, el sensor usa una técnica inteligente que consiste en un proyector de infrarrojos y una cámara que puede ver los pequeños puntos que este proyector produce.

En las siguientes imágenes sacadas del libro [4] se muestra como una cámara ve el sofá (Figura 2.5) y en como el sensor de profundidad ve la misma escena (Figura 2.6).

Figura 2.5 Sofá visto con la cámara RGB [4]

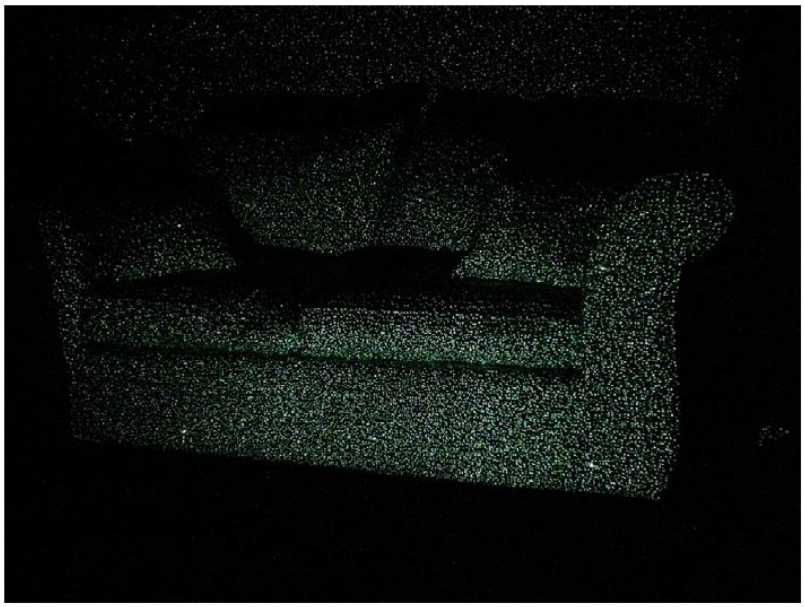


Figura 2.6 Sofá visto como el sensor de Profundidad [4]

El sensor de infrarrojos ve el sofá como unos cuantos pequeños puntos. El emisor emitirá constantemente estos puntos en el área frente a él.

La imagen anterior (Figura 2.6) fue tomada completamente a oscuras, con el sofá siendo iluminado solamente por la Kinect. El sensor de infrarrojos de la Kinect filtra la luz ordinaria, pudiendo ver los puntos incluso en una habitación bien iluminada. Los puntos están organizados en un patrón pseudo-aleatorio configurado dentro del sensor. En la siguiente imagen se pueden ver algunos de estos patrones.

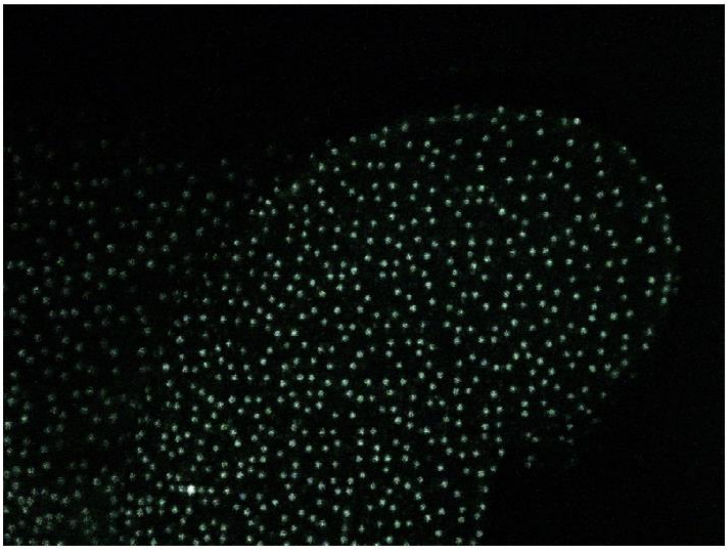


Figura 2.7 El patrón de puntos en el brazo del sofá [4]

Una secuencia pseudo-aleatoria puede parecer una secuencia aleatoria, pero en realidad es generada mecánicamente y fácil de repetir. Lo importante a recordar aquí es que el sensor de Kinect "sabe" lo que parece el patrón y cómo se dibuja. Se puede entonces comparar la imagen de la cámara con el patrón que sabe que está en pantalla, y puede utilizar la diferencia entre los dos para calcular la distancia de cada punto al sensor.

Esto se entiende mejor con un ejemplo. En la Figura 2.8 se muestra a dos personas, Pedro y José, uno frente a otro, a una distancia de metro y medio estando José desplazado ligeramente a la derecha. Pedro mantiene un puntero láser apuntando un papel que José sujeta, proyectando de este modo un pequeño punto en este papel. Si Pedro mantiene la dirección del haz de luz, a medida que José avanza hacia él el punto se verá desplazado ligeramente a la izquierda puesto que ahora golpea el papel antes de que haya viajado más lejos a la derecha.

Si sabes el lugar a donde está apuntando el punto puedes calcular a qué distancia está la posición del punto en el papel. Esto mismo hace la Kinect, pero calculando miles de puntos muchas veces por segundo. La cámara infrarroja en el Kinect le permite "ver" donde el punto aparece en la imagen. Debido a que el software sabe el patrón que el transmisor de infrarrojos dibujó, el hardware dentro de la Kinect hace todos los cálculos que se requieren para producir la "imagen de profundidad" de la escena que se envía al host receptor de la información.

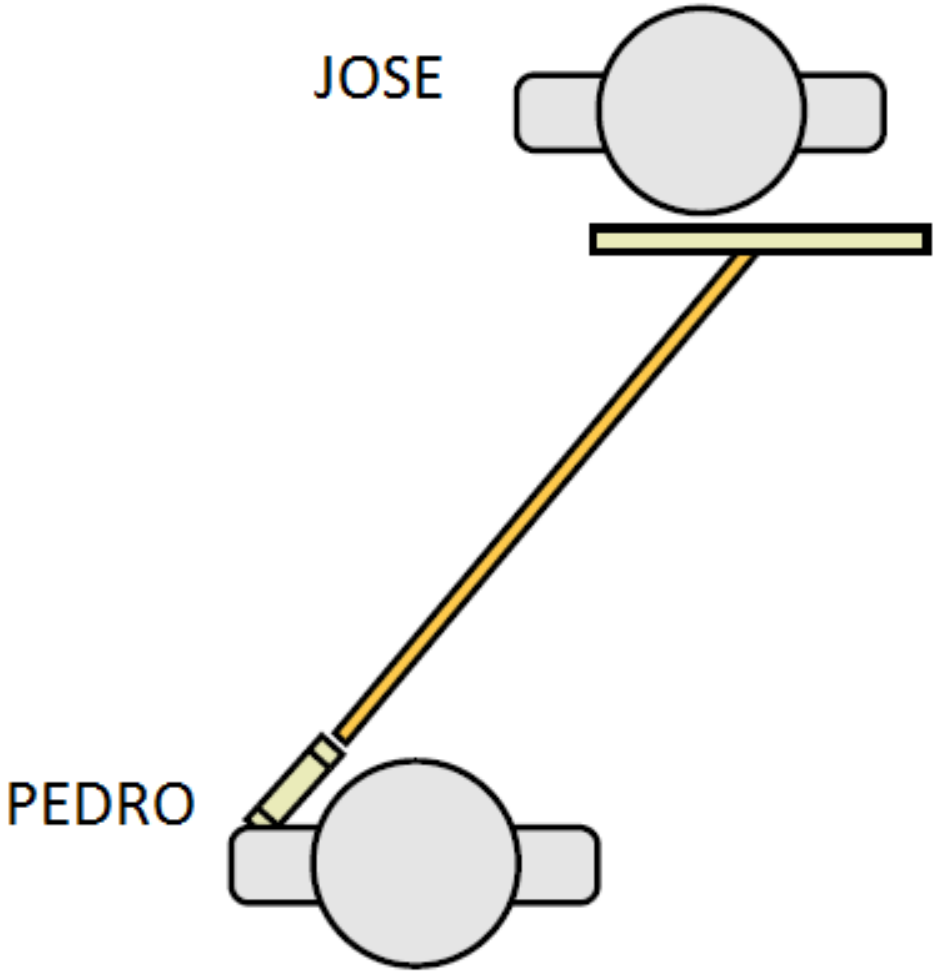


Figura 2.8 Demostración de cómo trabaja el sensor Kinect

2.3. Controlador del dispositivo

Con el fin de acceder a los datos trasmitidos por Kinect, es necesario instalar manejador de dispositivo en el ordenador. En este apartado se resumen los tres controladores de la Kinect mencionados en el apartado 2.1. Más tarde se justifica la utilización del SDK de Kinect.

OpenKinect: Drivers Libfreenect. Actualmente soporta el acceso a los datos RGB y profundidad, el motor de Kinect, el acelerómetro y el LED. No tiene acceso a los micrófonos.

PrimeSense: OpenNI/NITE 2

Proporciona una API y middleware de alto nivel llamado NITE para implementar el seguimiento de la mano/esqueleto y el reconocimiento de gestos. OpenNI no proporciona acceso ni al motor ni al acelerómetro.

Debido a OpenNI rompe la dependencia entre el sensor y el middleware (Figura 2.9), la API permite a los desarrolladores de middleware desarrollar algoritmos por encima de los formatos de datos en bruto, independientes del dispositivo que está produciendo los datos. De la misma manera, los fabricantes de sensores pueden construir sensores que funcionarán con cualquier aplicación compatible con OpenNI.

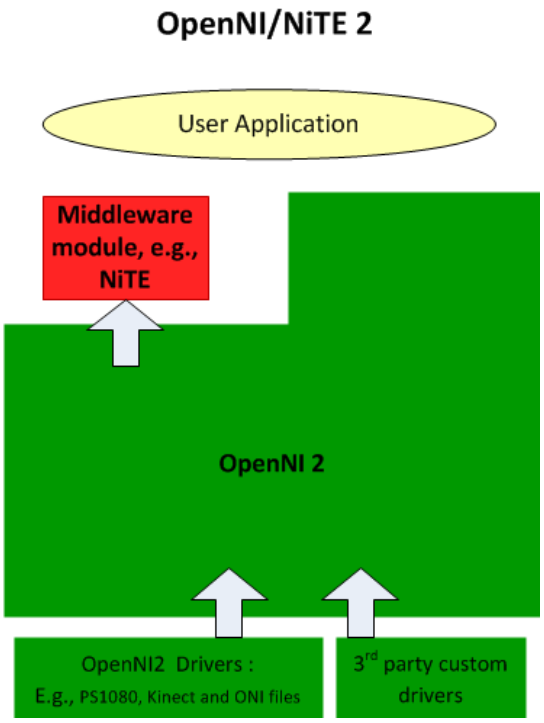


Figura 2.9 Vista en Capas de la arquitectura OpenNI

Microsoft Kinect Para Windows

Permite crear aplicaciones para Windows en distintos lenguajes de programación, como C++, C# o Visual Basic utilizando Microsoft Visual Studio 2010. Este kit es gratuito, pero es necesario tener una versión activada de Windows 7 o Windows 8 para poder utilizarlo. Su primera versión incluía las siguientes características:

- Acceso a las corrientes del sensor de profundidad y del sensor de la cámara a color.
- Seguimiento del Esqueleto.
- Capacidades avanzadas de audio.
- Códigos de Ejemplo y documentación.

El Kinect y la biblioteca de software permiten interactuar con la aplicación, como se muestra en la siguiente figura:

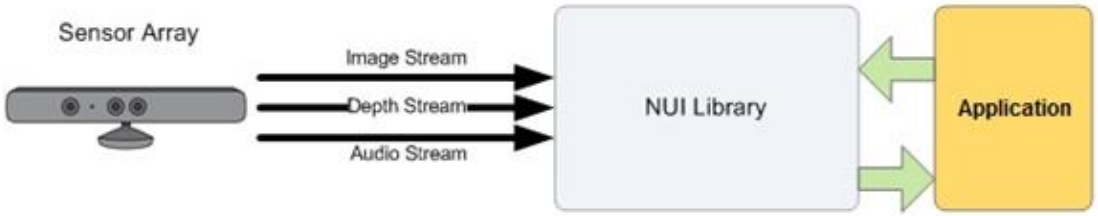


Figura 2.10 Hardware y Software iterando con una aplicación

La arquitectura del SDK de Microsoft incluye los siguientes componentes:

1) Hardware de Kinect: los componentes de hardware, incluyendo el sensor de Kinect y el conector USB a través del cual el sensor Kinect está conectado a la computadora.
2) Drivers de Kinect: los controladores de Windows para el Kinect, que se instalan como parte del proceso de instalación del SDK. Los drivers de Kinect son compatibles con:
   - El conjunto de micrófonos Kinect como un dispositivo de audio en modo de núcleo que se puede acceder a través de las API de audio estándar de Windows.
   - Audio y controles de transmisión de video para la transmisión de audio y vídeo (color, profundidad, y el esqueleto).
   - Funciones de enumeración de dispositivos que permitan a una aplicación utilizar más de un Kinect.
3) Componentes de Audio y Video:
   - Interfaz natural de usuario para el seguimiento de esqueleto, de audio, y el color y la profundidad de imagen.
4) DirectX Media Object (DMO) para la formación de haces conjunto de micrófonos y localización de la fuente de audio.
5) De Windows 7 API estándar: El audio, voz y APIs multimedia de Windows 7, como se describe en el SDK de Windows 7 y el Microsoft Speech SDK. Estas API también están disponibles para aplicaciones de escritorio en Windows 8.

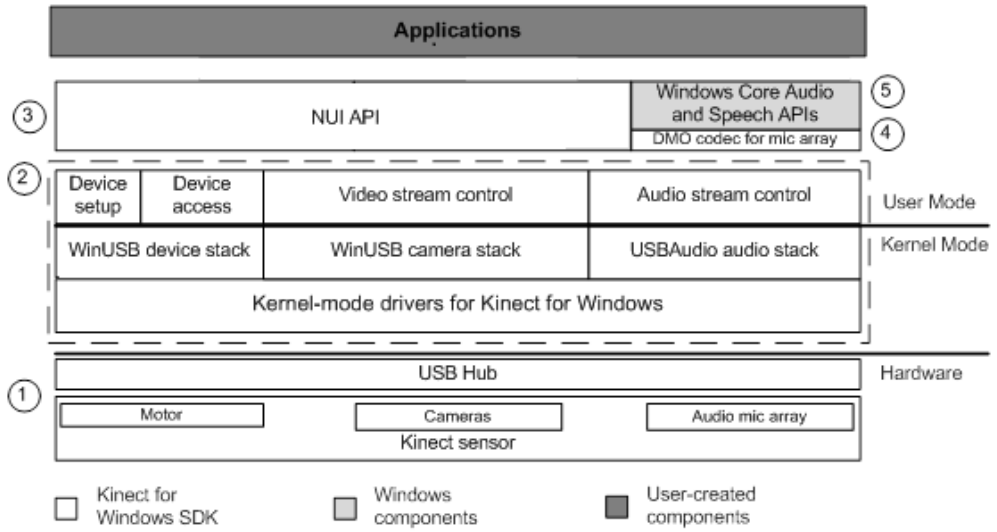


Figura 2.11 Componentes de la arquitectura SDK de Microsoft

Justificación de elección del SDK oficial de Microsoft Kinect para Windows

La primera diferencia obvia es que el SDK oficial es mantenido por el equipo de Investigación de Microsoft, mientras que las opciones de código abierto OpenKinect y OpenNI son mantenidas por la comunidad de código abierto. El SDK oficial es desarrollado por Microsoft, que también desarrolla el hardware y, por tanto, debe conocer la información interna sobre el dispositivo mientras que la sociedad de código abierto debe realizar ingeniería inversa. Obviamente, esto es una ventaja para Microsoft.

El uso del SDK desarrollado por Microsoft está limitado por los lenguajes de programación, c#, c++ o VisualBasic con Microsoft Visual Studio. OpenNI y OpenKinect pueden ser programados en Python, C, C++, C#, Java, Lisp y no requieren Visual Studio.

El SDK oficial sólo se instala en Windows 7, Windows 8 y Windows 8.1, mientras que OpenNI y OpenKinect se ejecuta en Linux, OS X y Windows. El ordenador en el que se iba a implementar el proyecto ejecuta Windows 8.1 por lo que es posible instalar el SDK oficial.

En cuanto a la documentación, el SDK tiene una clara ventaja ya que proporciona un foro de soporte [5] y una página de referencia en la que se describe las API de código administrado [6].

Otra característica a tener muy en cuenta es la calibración. Diferentes dispositivos Kinect pueden variar ligeramente dependiendo del lote en que se hayan producido. Por lo tanto la calibración del dispositivo es necesaria, pero en el momento de la producción los parámetros de calibración se escriben en el dispositivo Kinect, así que no es necesaria la calibración Oficial SDK de Microsoft, pero si es necesaria en OpenKinect.

Los contras que tiene el SDK oficial como son la limitación en cuanto al sistema operativo a utilizar no son un problema, ya que el ordenador a utilizar como se ha descrito en el apartado 4.1.2 de este documento trabaja con Windows 8.1.

Los términos de licencia del SDK de Microsoft Kienct para Windows [7] deben ser aceptados para poder utilizar y comercializar las aplicaciones desarrolladas con este material. El punto 4 del apartado 2.a.ii Requisitos de Distribución de este documento dice textualmente: "se *deberá distribuir el Código Distribuible que se incluya en un programa de instalación exclusivamente como parte de dicho programa sin modificación alguna".* Por lo tanto, se puede distribuir las aplicaciones desarrolladas en este proyecto, siempre que se distribuyan como código cerrado.

Por las ventajas numeradas que ofrece y las desventajas no relevantes para el objetivo del proyecto se ha elegido el SDK de Microsoft Kinect para Windows como el kit de desarrollo utilizado para desarrollar las aplicaciones de este proyecto.

C# como lenguaje de Programación

Los lenguajes de programación que ofrece el SDK de Microsoft Kinect para Windows son: VisualBasic, C# y C++. Los tres son orientados a objetos.

Antes de empezar el proyecto, se tenía conocimientos en C. Por lo tanto, VisualBasic queda descartado como lenguaje a utilizar porque su sintaxis es heredada del BASIC, del cual no se tenía ningún conocimiento.

C++ es un lenguaje de bajo nivel, lo que le hace más eficiente por ser más cercano al lenguaje máquina, compilando y ejecutándose más rápido que lenguajes de alto nivel como C#.

C# es un lenguaje avanzado que se está convirtiendo en uno de los lenguajes más populares, y permite crear aplicaciones que interactúan con los servidores o la web [8]. Con vistas a mi futuro como ingeniero y por la existencia de más ejemplos del "Buscador de Herramientas para Desarrolladores de Kinect" que de los otros lenguajes se eligió el lenguaje C#.

Windows Presentation Foundation

Puesto que es necesario la utilización de Visual Studio para desarrollar las aplicaciones del SDK de Microsoft, se opta por Windows Presentation Foundation, WPF, como tecnología en la que desarrollar las aplicaciones.

Se trata de un marco de interfaz de usuario de nueva generación para crear aplicaciones enriquecidas e interactivas. Es un subconjunto de .NET Framework del subsistema Windows orientado a unificar los mecanismos de creación y gestión de interfaces de usuario.

WPF usa un lenguaje XML llamado XAML para implementar la interfaz de una aplicación pudiendo crear ventanas, cuadros de diálogo y controles de usuario. La lógica de la aplicación se codifica en la plataforma .NET sobre el lenguaje C#.

Para aprender sobre las aplicaciones WPF se ha seguido el libro [9] y visualizado algunos videos tutoriales [10].

2.4. Herramientas para desarrolladores de Kinect para Windows

Herramientas para desarrolladores de Kinect para Windows es un buscador de aplicaciones utilizado en este proyecto para empezar a familiarizarse con el desarrollo de aplicaciones Kinect.

Entre los muchos ejemplos contenidos en la herramienta se van a describir los que han influenciado en el desarrollo de este proyecto. Además se describe la herramienta Kinect Studio.

Ejemplo C# Color Basic -WPF [11]

Cuando se ejecuta este ejemplo, se ve lo siguiente:

- El flujo de vídeo de color capturado y visualizado en la pantalla. Dado que se captura cada fotograma, se trata de una imagen en vivo de lo que el Kinect está viendo.
- Un botón para capturar la pantalla y guardarla en un archivo.

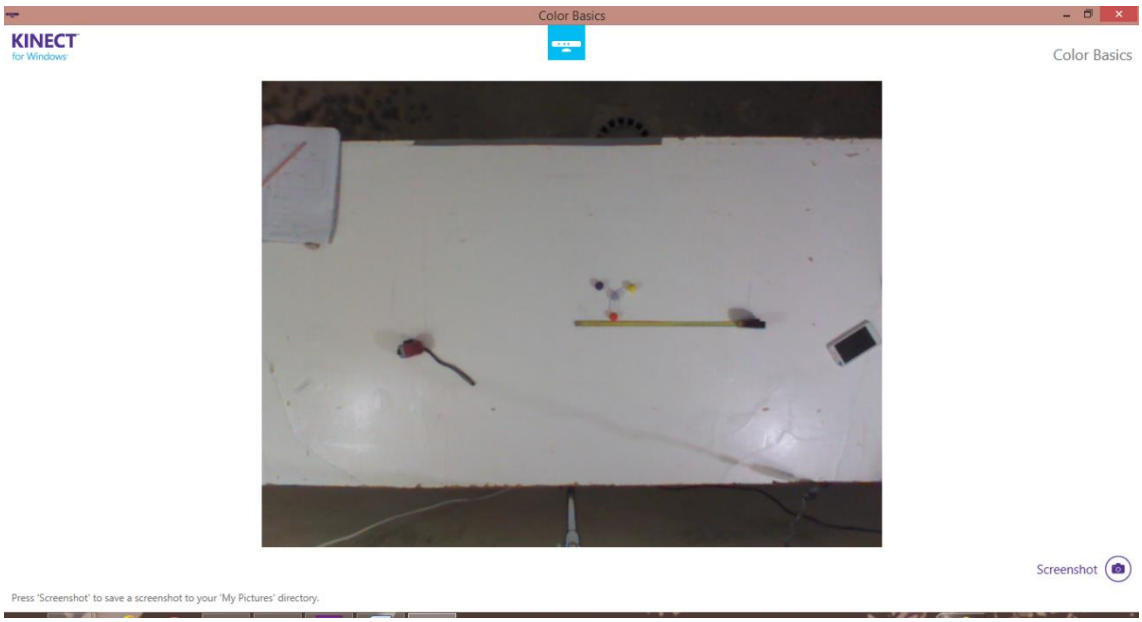


Figura 2.12 Utilizando Color Basic

A la hora de desarrollar la aplicación Seguimiento de Marcadores (apartado 6.1. de este documento) se analizó este código para entender cómo recibir los datos del sensor, establecer el formato y velocidad de fotogramas del flujo de vídeo capturado, especificar las propiedades del fotograma recibido, crear un controlador de eventos para los datos de color y escribir los datos de pixel recibidos dentro de un Bitmap.

Ejemplo C# Depth Basics-WPF[12]

- Los datos de profundidad son capturados y mostrados. Estos son un continuo flujo de una imagen en escala de grises que representa la distancia entre el sensor cualquier punto que está dentro de la mira del sensor. La cámara puede ver píxeles en el interior del espacio de interacción, coloreados en una escala de grises. Los píxeles que están fuera del espacio de interacción son de color negro.
- Una casilla de verificación para configurar el sensor al modo cercano.
- Un botón para capturar la pantalla y guardarla en un archivo.

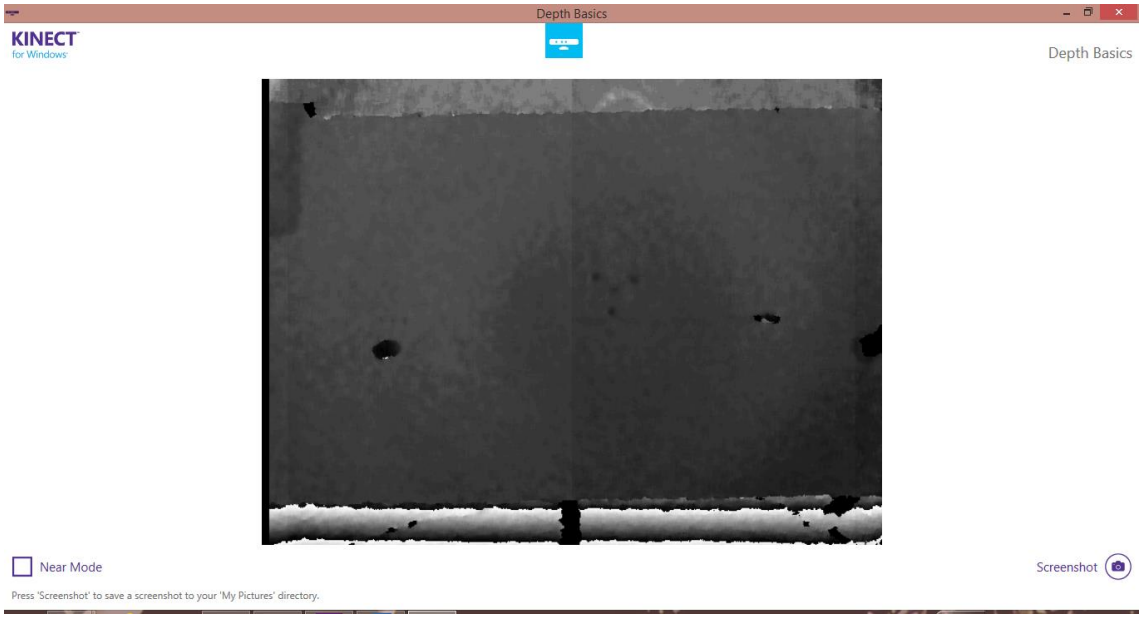


Figura 2.13 Utilizando Depth Basic

Este ejemplo también se analizó para la implementación del Seguimiento de Marcadores. Se analizó los distintos formatos y la velocidad de fotogramas, los permisos, métodos de encendido y propiedades de los datos de la imagen de profundidad y como crear un controlador de eventos para los datos de profundidad.

Ejemplo C# Infrared Basics-WPF[13]

Demuestra cómo habilitar la cámara de profundidad para transmitir luz infrarroja invisible y muestra una imagen en tiempo real de la resultante corriente de infrarrojos.

A diferencia de los dos apartados anteriores, este ejemplo no ha sido utilizado directamente en ningún programa desarrollado en este proyecto, pero ayudó a comprender como trabaja el sensor de profundidad de Kinect.

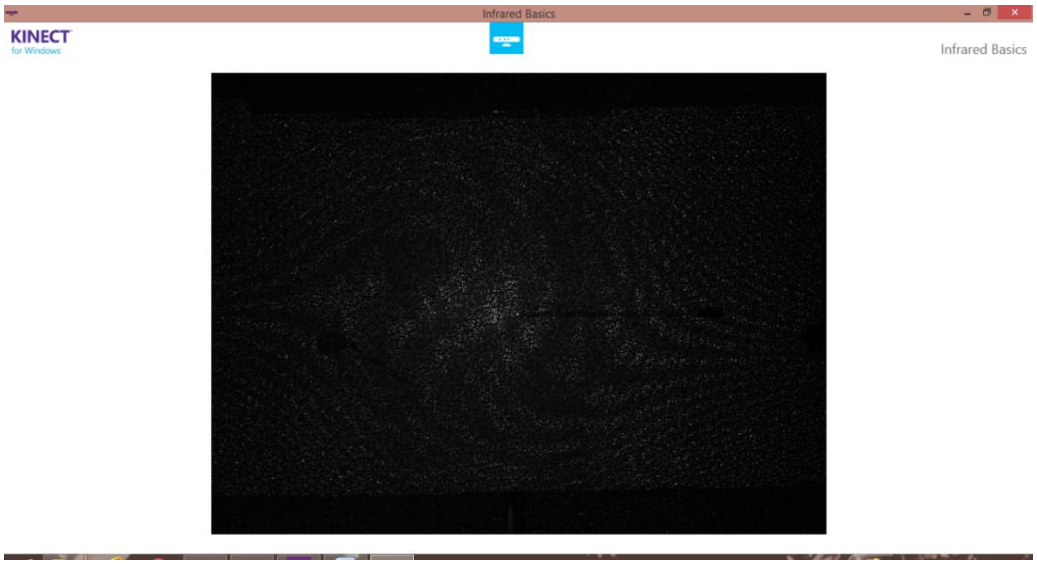


Figura 2.14 Utilizando Infrared Basic

Kinect Studio

Kinect Studio es una herramienta que ayuda a grabar y reproducir secuencias de profundidad y color de un Kinect. Será una herramienta muy utilizada durante el proyecto, tanto en fase de depuración de los programas como en la adquisición de datos para el análisis de resultados.

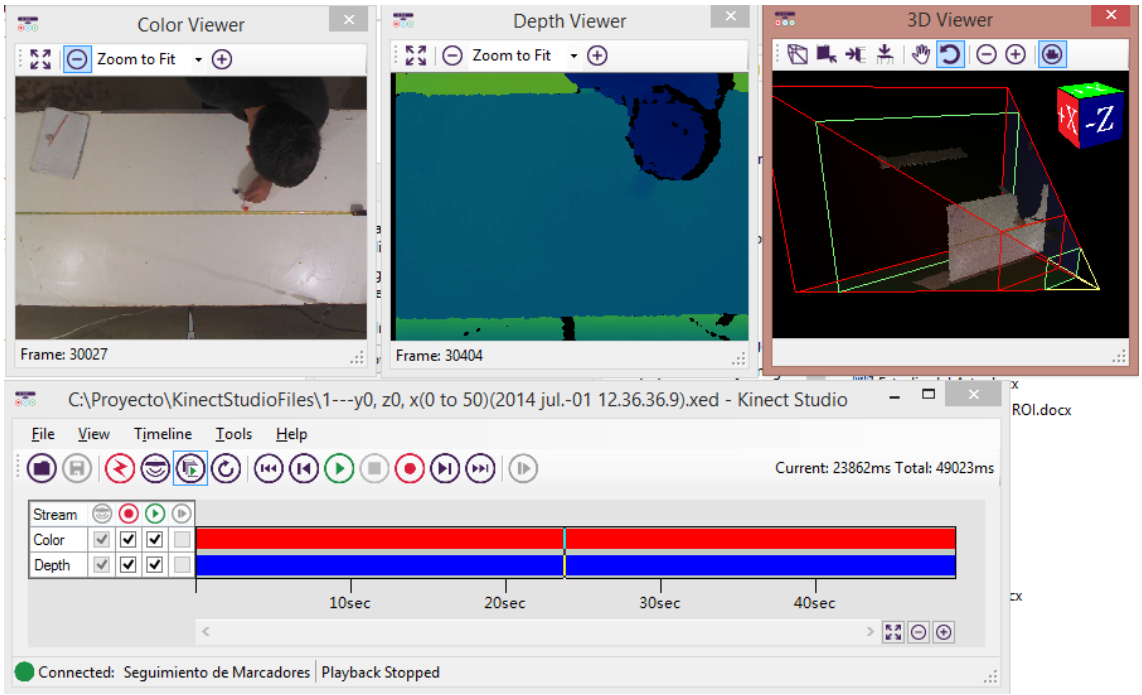


Figura 2.15 Utilizando Kinect Studio en las pruebas de precisión de la herramienta

2.5. Seguimiento de mano: NimbleSDK

El sistema de seguimiento de mano Nimble [14] desarrollado por 3GearSystems está diseñado para seguir la posición y la orientación de las manos de un usuario, así como la pose de los diez dedos. Su versión más actual se centra en la representación del seguimiento robusto de la mano y el movimiento de "pellizco" en particular [15].

Esta herramienta está diseñada para que un usuario interactúe con un entorno 3D con los gestos de la mano. Reconoce varios gestos que permiten crear eventos que encadenen acciones.

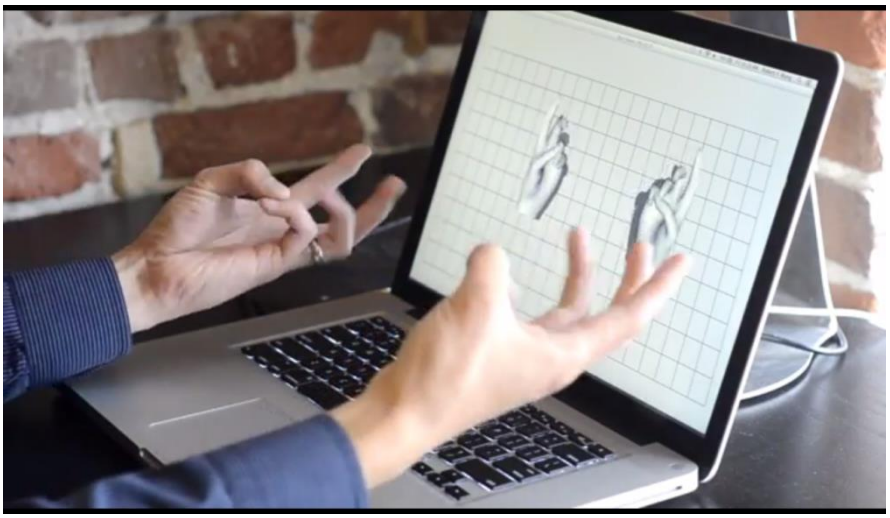


Figura 2.16 Sistema de Seguimiento de Mano 3GearSystem en modo arbitrario de seguimiento de 10 dedos [14]

Tiene algunas limitaciones como que actualmente rastrea algunas poses mejor que otras porque se basa en una base de datos de poses para rastrear las manos del usuario. No obstante permite un "modo arbitrario de seguimiento de los 10 dedos".

Funcionamiento del SDK [16]

El servidor de seguimiento de mano utiliza un protocolo de red simple para comunicarse con sus aplicaciones a través de un socket. Esto significa se puede escribir aplicaciones en cualquier lenguaje que se elija. Proporciona ejemplos de bibliotecas para comunicarse con el servidor de seguimiento de mano y una API para Java, C ++, y C #.

Una vez conectado al servidor, la biblioteca HandTrackingClient comienza su propio hilo, a la espera de nuevos mensajes (HandTrackingMessage). Los mensajes recibidos (por ejemplo, Prensado, Arrastrado, Lanzamiento, etc.) se pasan a las devoluciones de llamada registradas que implementan la interfaz HandTrackingListener.

Cada mensaje contiene la posición y orientación de las dos manos. El espacio de coordenadas de la posición de la mano es especificado por la etapa de configuración de la cámara. Para una configuración de la cámara estándar, eje x positivo hacia la derecha; el eje y apunta hacia arriba; y los el eje Z hacia fuera de la pantalla. Las unidades están en milímetros y el origen está en el centro del tablero de ajedrez que utilizó durante la calibración.

Descripción Aplicación Posición de las Manos:

Con el objetivo de entender el funcionamiento de esta SDK y la posibilidad de utilizar esta herramienta en el *Sistema de Seguimiento de Operaciones* desarrollado en este proyecto, se implementó una aplicación capaz de representar las coordenadas y orientación de la mano y de almacenar estas en el momento de captar un pinzamiento.

Esta aplicación se integró como un proyecto llamado "EscribirCoord" dentro de la solución "GesturalApps" con ejemplos escritos en C# para Visual Studio facilitada con la instalación de NimbleSDK. En la ruta *"D:\información adicional\Posición de las Manos\"* del **CD del Proyecto** se encuentra este proyecto.

El código que forma el proyecto EscribirCoord está contenido en el ANEXO II apartado 1. Está dividido en tres archivos:

- StartApp.cs: este código contiene las clases:
  - PrintMessage: se encarga de conectarse con el servidor (nimble_server.bat), de recibir el mensaje "PinchMessage" que contiene las coordenadas, orientación y modo de pinzado de las manos y de dividir este mensaje en un vector de strings llamado MiVariable utilizado por el archivo MainWindow.c.
  - StartApp: encargada de iniciar un cliente "HandTrackingClient" y arrancar la aplicación.
  - MiClase: contiene la variable pública MiVariable.
  - Global: contiene un array de 3 dimensiones llamado puntosxyz donde se almacenan las acciones realizadas.
- MainWindow.xaml: contiene el código que define la interfaz. En la Figura 2.17 podemos ver los controles que la forman:
  - Zona 1: en ella se escribe la mano cuyas coordenadas están siendo representadas.
  - Zona 2: representación de las coordenadas x, y, z y los cuaternos w, x, y, z de cada mano.
  - Zona 3: Botones encargados de manejar la ejecución de la aplicación: iniciar permite guardar las coordenadas de las operaciones "pinza" o "doble pinza" en modo **Referencia**; ejecución permite guardar las coordenadas de las operaciones en modo

Ejecución; parar hace que se deje de registrar operaciones; comprobar activa el modo **Comprobación** que compara las operaciones guardadas en modo de referencia y ejecución una a una.

- Zona 4: indica si las operaciones realizadas en ambos modos han sido similares.

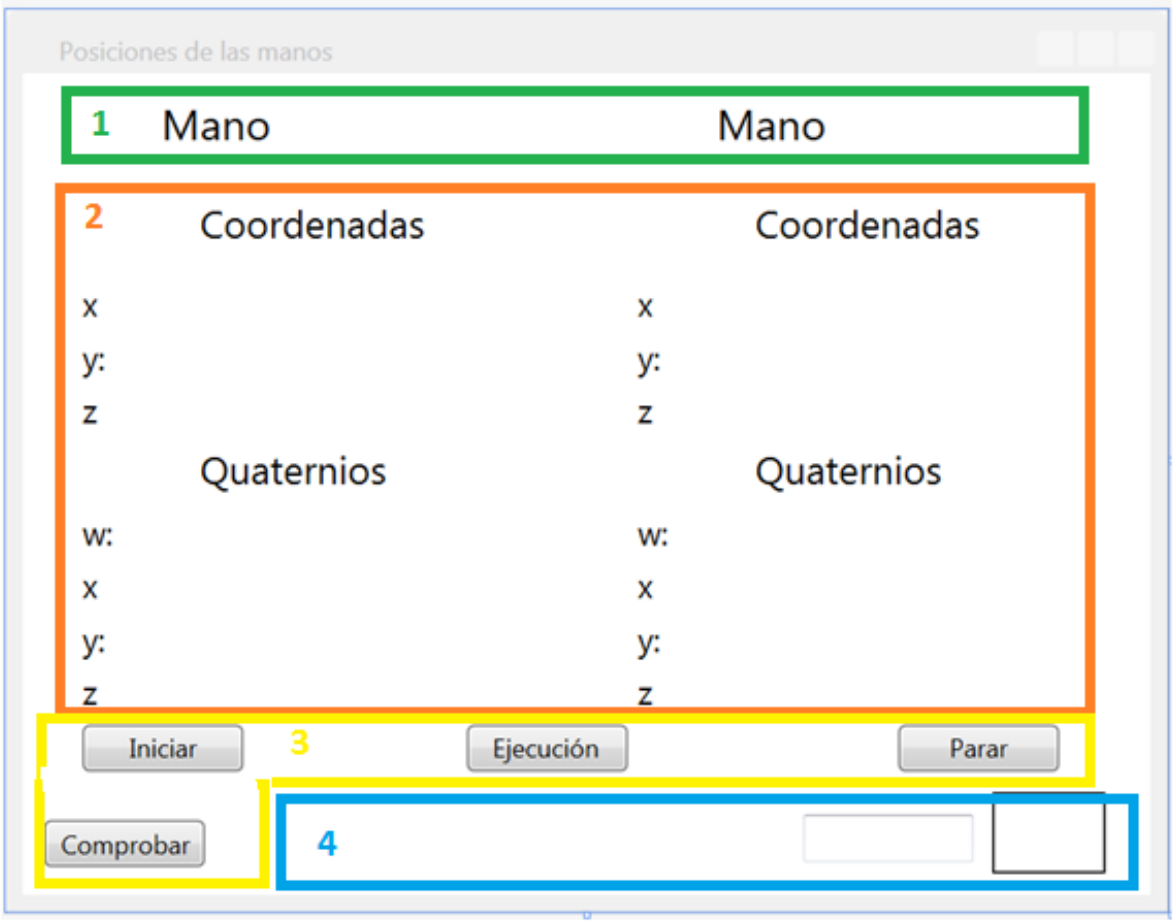


Figura 2.17 Interfaz desarrollada para el ejemplo

- MainWindows.cs: contiene el código que permite el funcionamiento de la aplicación. Está dividido en diferentes regiones:
  - Eventos de los Botones: controla la ejecución de los botones de la Zona 3.
  - Subprocesos Cíclicos: ciclo que representa los valores recogidos en la variable *MiVariable* en la Zona 2 y guarda las coordenadas de la mano cuando se ejecuta un pinzamiento.
  - Métodos: funciones utilizadas por los eventos de los botones y el subproceso de lectura como guardar las operaciones, imprimirlas coordenadas por pantalla, comprobar los campos, y limpiar las distintas acciones.

Pruebas y Resultados de la aplicación Posiciones de las Manos:

Para ejecutar esta aplicación es necesario que se esté ejecutándose el servidor "nimble_server.bat" facilitado en la instalación de NimbleSDK. Este proporciona los mensajes utilizados por nuestra aplicación funcionando de cliente.

Tanto en el modo **Referencia**, como en el modo **Ejecución** se representan las coordenadas de cada mano (Figura 2.18) y se guardan estas cuando el usuario hace un movimiento de pinzamiento.

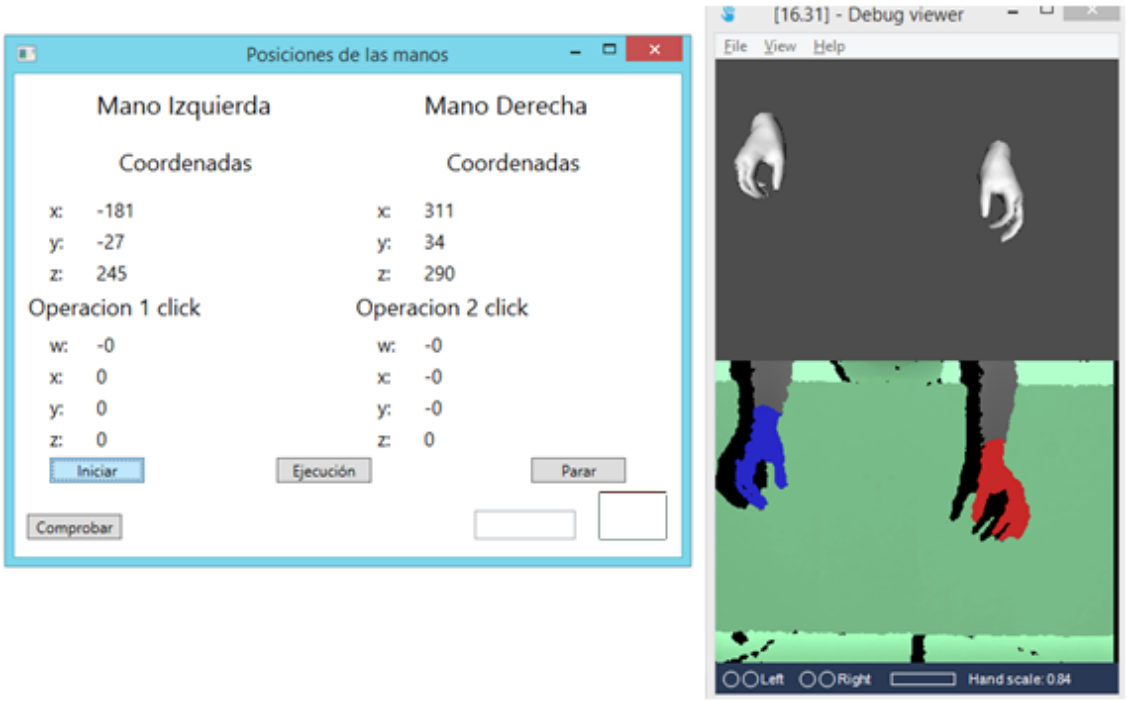


Figura 2.18 Ejecutando Aplicación Posiciones de las manos

En el modo **Comprobación** se comparan las operaciones registradas.

- Operación 1: se están comparando dos operaciones realizadas con la mano izquierda, y la aplicación determina que ambas son similares ya que las coordenadas (en mm) en las que han sido ejecutadas son cercanas, y en ambas se realiza la operación un "Click" o "pinzamiento".
- Operación 2: se están comparando dos operaciones realizadas con la mano izquierda, y la aplicación determina que ambas difieren en la posición en la que han sido ejecutadas.
- Operación 3: se están comparando dos operaciones realizadas con la mano izquierda, y la aplicación determina que ambas difieren en la operación realizada.

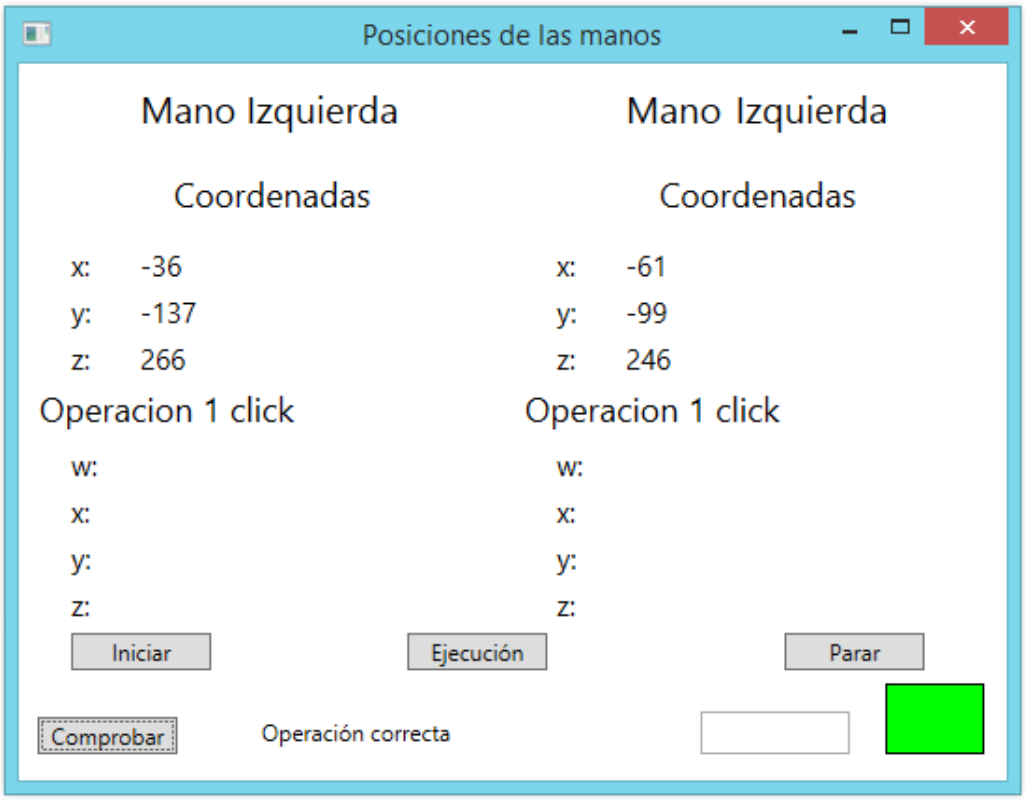


Figura 2.19 Comprobando Operación 1

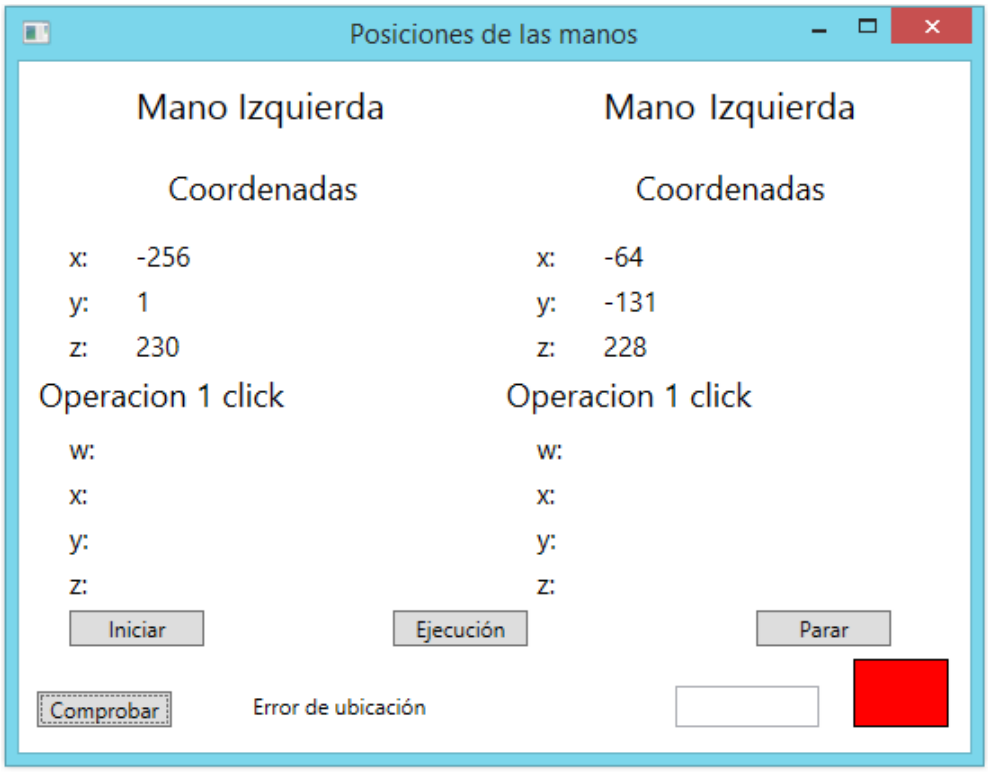


Figura 2.20 Comprobando Operación 2

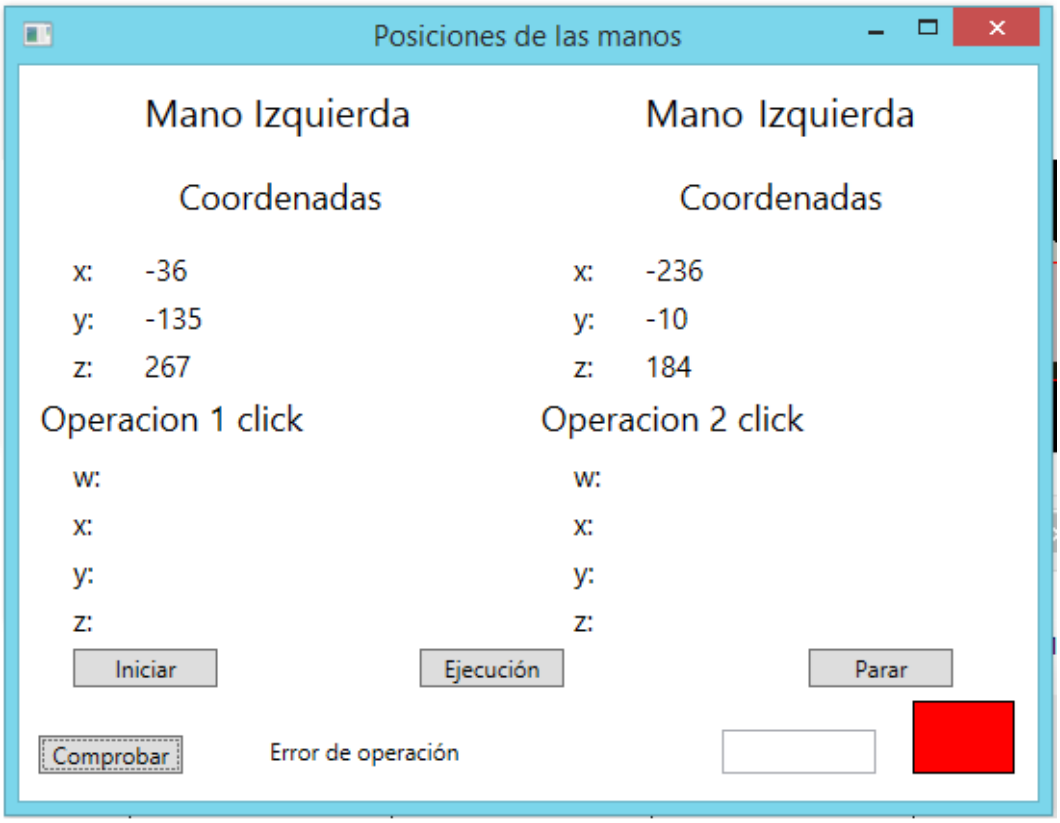


Figura 2.21 Comprobando Operación 3

Conclusiones:

La herramienta NimbleSDK es tremendamente precisa y con licencia gratuita para Estudiantes y empresas con menos de 100.000 $ de volumen de negocios. Además con una fácil calibración con un tablero de ajedrez se puede empezar a utilizar.

Pero su gran limitación y la razón por la que no es posible utilizarla en el Sistema de Seguimiento de Operaciones de este proyecto es que necesita ejecutar el servidor de seguimiento de mano (nimble_server.bat) para ejecutar cualquier aplicación que depende del seguimiento de la mano. Esto hace imposible su utilización en nuestro proyecto, ya que Kinect sólo permite que una aplicación recoja a la vez los datos que ofrece. 3GearSystem permite el acceso a los datos de profundidad pero no a los de la cámara RGB necesarios para ejecutar la aplicación desarrollada.

Otro punto en contra de esta aplicación es que cuando estas manipulando algún objeto con las manos el sistema detecta dicho objeto como parte de la mano. Esto daría lugar a falsos valores de la posición de las manos cuando el operario está manipulando una herramienta.

No obstante, su funcionamiento ha servido de inspiración en el diseño de este proyecto.

3. Conceptos sobre Información Cromática y Geometría Espacial

En este apartado se hablará de los conceptos teóricos utilizados durante el proyecto. Es importante entender estos conceptos para entender los apartados posteriores.

3.1. Espacios de color

Una de las tareas que necesita abordar la aplicación implementada en este proyecto es el tratamiento de las imágenes: simplificar la imagen a color para poder encontrar de manera efectiva los marcadores. Por ello es necesario realizar un estudio de los espacios de color [17] utilizados.

Espacio RGB

El espacio RGB es el espacio de color más extendido y el que utiliza la cámara a color de Kinect. Se representa como un cubo (ver figura 3.1) donde un color viene definido por la mezcla de valores de intensidad de tres colores primarios, rojo, verde y azul. Un color viene descrito por 3 coordenadas en el cubo. El color negro se representa por r=0, g=0, b=0 y el color blanco por r=255, g=255, b=255. La gama acromática de escala de grises está representada por la diagonal del cubo.

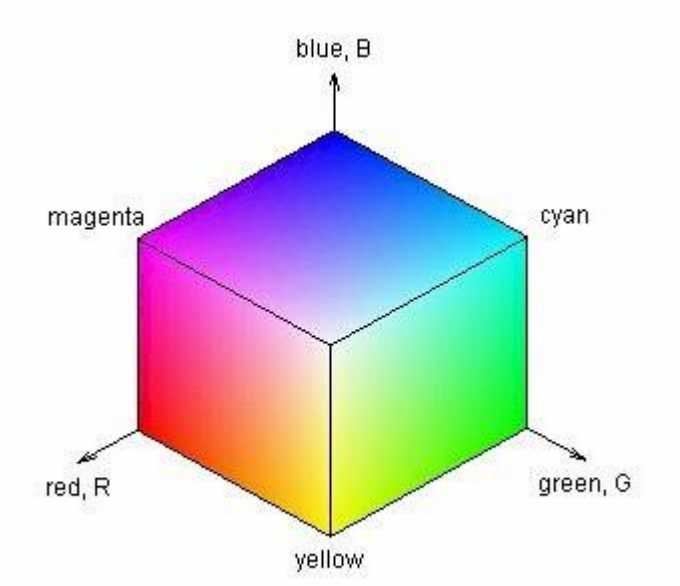


Figura 3.1 Cubo RGB

Espacio HSV

El espacio HSV hace más intuitiva la representación es espacio de coordenadas. Su interpretación geométrica viene determinada por un cono de base quasi-hexagonal (ver Figura 3.2).

Con esta representación cada color trabaja con 3 componentes básicas: matriz, saturación y brillo. El matiz, H_{HSV}, hace referencia al valor de cromaticidad o clase de color. La Saturación, S_{HSV}, se refiere a las longitudes de onda que se suman a la frecuencia del color, y determina la cantidad de blanco que contiene un color. Un color muy saturado tiene más cantidad de blanco que otro menos saturado, por tanto la saturación representa la cantidad de blanco que contiene un color. Así, la falta de saturación viene dada por la generatriz en la representación del cono HSV. Esta falta de saturación representa la gama de grises desde el blanco hasta el negro. El brillo o luminancia se corresponde con la apreciación subjetiva de claridad y oscuridad.

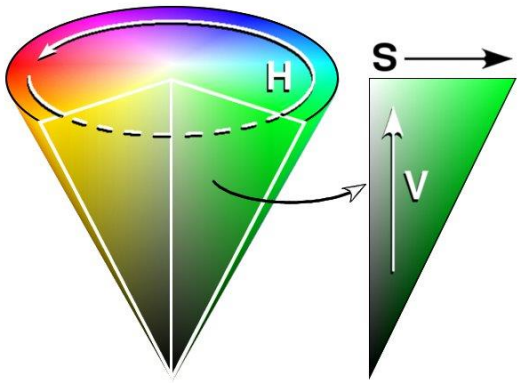


Figura 3.2 Espacio HSV

Cuando las imágenes son tomadas con una cámara RGB es necesaria la conversión a HSV. De esta manera el sistema HSV viene definido por:

$$h_{HSV} = \begin{cases} \dfrac{g-b}{\max(r,g,b)-\min(r,g,b)} & \text{si } r = \max(r,g,b) \quad (3.1) \\ \dfrac{b-r}{\max(r,g,b)-\min(r,g,b)} + 2 & \text{si } g = \max(r,g,b) \quad (3.2) \\ \dfrac{r-g}{\max(r,g,b)-\min(r,g,b)} + 4 & \text{si b} = \max(r,g,b) \quad (3.3) \end{cases}$$

$$s_{HSV} = \frac{\max(r,g,b)-\min(r,g,b)}{\max(r,g,b)} \quad (3.4)$$

$$v_{HSV} = \max(r,g,b) \quad (3.5)$$

Existen otras variantes del espacio color como el HSI y HLS en función de cómo se modifique su representatividad. El espacio de color HSV representa mejor que el HLS la saturación, aunque tiene peor representación de la luminancia.

3.2. Iluminación

En este apartado se hablará del concepto de reflexión y de la lámpara fluorescente compacta, fuente de iluminación utilizada en este proyecto.

Reflexión

La reflexión se produce cuando la luz incide en la interface entre dos medios. Depende del material y el acabado superficial de este. La luz reflejada en cada dirección del espacio es la suma de la componente difusa (zonas mates) y la componente especular (brillos).

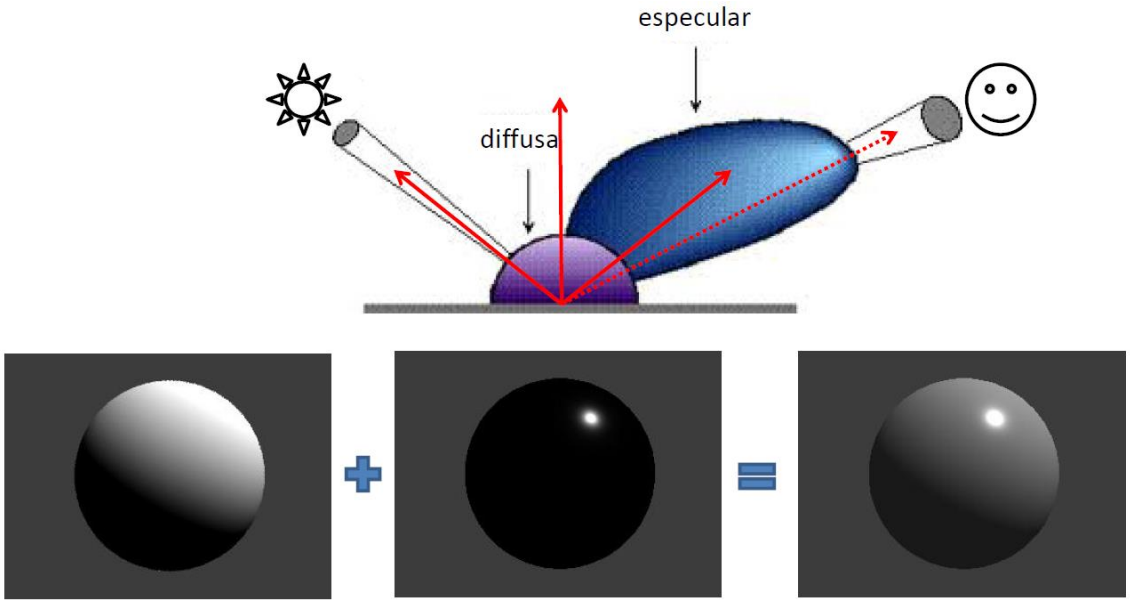


Figura 3.3 Reflexión Difusa + Especular.

La componente especular se concentra en la dirección simétrica de la incidencia. Son los brillos, más o menos intensos según el material y más o menos concentrados según la rugosidad.

La componente difusa es uniforme en el semiespacio. No depende de la dirección de observación y es proporcional a la cantidad de luz que incide en el objeto.

Lámpara fluorescente compacta

La lámpara fluorescente compacta es un tipo de lámpara que aprovecha la tecnología de los tradicionales tubos fluorescentes para hacer lámparas de menor tamaño que puedan sustituir a las lámparas incandescentes con pocos cambios en

la armadura de instalación y con menor consumo. La luminosidad emitida por un fluorescente depende de la superficie emisora, por lo que este tipo de lámparas aumentan su superficie doblando o enrollando el tubo de diferentes maneras.

Figura 3.4 Lámpara fluorescente compacta

En comparación con las lámparas incandescentes, las LFC tienen una vida útil más larga y consumen menos energía eléctrica para producir la misma cantidad de luz. Genera luz difusa.

El funcionamiento de una lámpara fluorescente compacta es el siguiente: la corriente eléctrica alterna pasa por el balasto electrónico, donde un rectificador diodo de onda completa la convierte en corriente continua. A continuación un circuito oscilador, compuesto fundamentalmente por un circuito transistorizado que funciona como amplificador de corriente, una bobina, condensador de flujo o transformador (reactancia inductiva) y un condensador (reactancia capacitiva), se encarga de originar una corriente alterna con una frecuencia de entre 20 y 60 kHz. El objetivo de esa alta frecuencia es disminuir el parpadeo que provoca el arco eléctrico que se crea dentro de las lámparas fluorescentes cuando se encuentran encendidas. De esa forma se anula el efecto estroboscópico que normalmente se crea en las antiguas lámparas fluorescentes de tubo recto cuando funcionan con balastos electromagnéticos (no electrónicos).

3.3. Tratamiento de imágenes

Las imágenes recogidas por la cámara RGB de Kinect deben ser tratadas. Para ello se va a utilizar la librería EmguCV [18], una plataforma de envoltorio cruzado .Net de la biblioteca de procesamiento de imágenes OpenCV [19]. De esta manera se pueden invocar las funciones de OpenCV desde la aplicación WPF desarrollada en lenguaje C#.

A continuación se describirán distintos conceptos utilizados para sacar la información deseada de la imagen a color.

Región de Interés. ROI

Una región de interés es un subconjunto de la imagen completa. La idea es que las funciones que normalmente operan en el total de la imagen, solamente actúen en el subconjunto indicado por el ROI. De esta forma se consigue aumentar la velocidad de procesamiento del ordenador cuando se realizan las operaciones de visión, ya que se reduce la cantidad de información.

Conversión de espacios de color

Como ya vimos en el apartado 3.1 existen varios espacios de color para una imagen. Por lo tanto Emgucv facilita métodos que permiten realizar conversiones entre el espacio RGB al HSV. Además permite descomponer la imagen HSV en sus tres planos H, S, V separados en imágenes en escala de grises. La escala de grises representa cada pixel de una imagen en valores desde 0 (negro) hasta 255 (blanco).

Binarizar estableciendo el rango

Se quiere comprobar si los píxeles de una imagen se encuentran dentro de un rango específico en particular. Cada píxel de la imagen se compara con el valor correspondiente en las imágenes inferiores y superiores. Si el valor en el original es mayor o igual al valor de la imagen inferior y también menor que el valor en la imagen superior, el valor correspondiente en la imagen resultante se establecerá en 1; de lo contrario, el valor de este se establece en 0. De esta manera se consigue dividir una imagen en dos regiones: el objeto del que extraeremos características y el fondo que no proporciona información. Esto se conoce como una imagen Binaria de sólo dos valores: 1 y 0 (Verdadero o Falso, blanco y negro). Es útil porque se reduce mucho la cantidad de información.

Intersección (A ∩ B)

Consiste en realizar una operación AND bit a bit de la matriz imagen. De esta manera se consigue que en la imagen Resultante solamente serán verdaderos los que lo son en ambas imágenes.

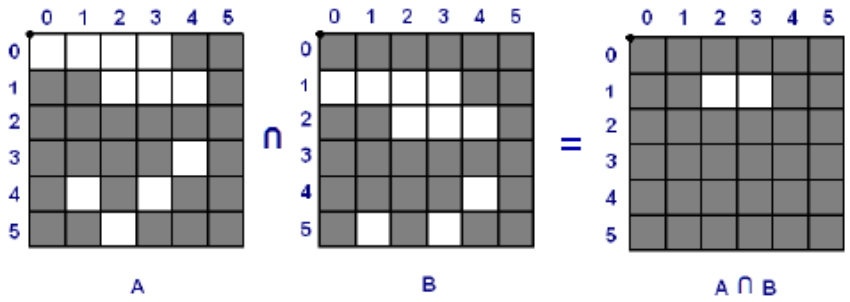


Figura 3.5 Ejemplo de Intersección (operador AND entre imágenes)

Dilatación

La dilatación es una convolución de una imagen que llamaremos A, con un elemento estructural, que llamaremos B. Consiste en tomar cada pixel objeto (valor "1") de la imagen A y cambiar todos los valores pertenecientes al fondo (valor "0") que tienen conectividad con el elemento estructural B. Es decir, poner a "1" los pixeles del fondo vecinos a los pixeles objeto.

Dependiendo del elemento estructural elegido se obtendrán un resultado u otro.

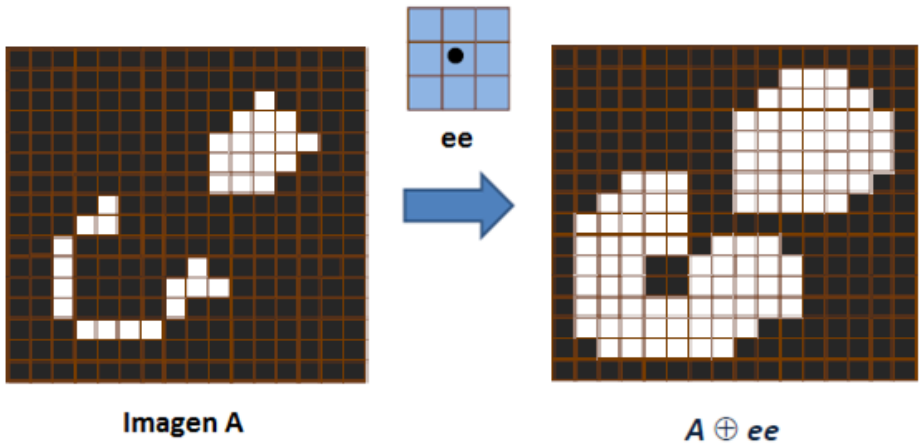


Figura 3.6 Dilatación de una imagen A con un elemento estructural rectangular 3x3

Erosión

La erosión es la operación inversa a la dilatación. Consiste en tomar cada pixel del objeto que tiene una conectividad con los píxeles del fondo y cambiarlos al valor "0". Es decir, poner a "0" los píxeles del objeto vecinos a los pixeles del fondo.

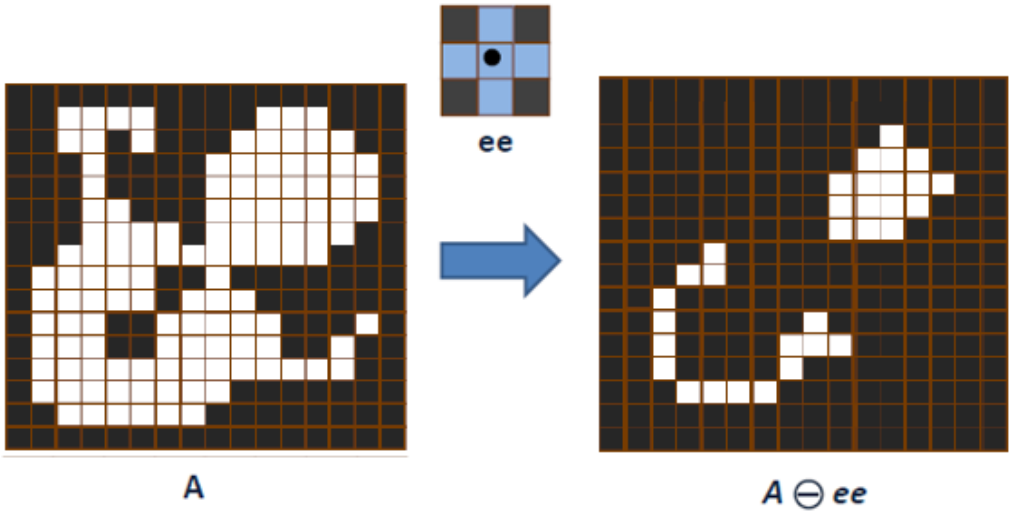


Figura 3.7 Erosión de una imagen A con un elemento estructural rectangular 3x3

Apertura

Una apertura es la combinación de una erosión seguida de una dilatación. Normalmente es utilizada para eliminar ruido y separar objetos.

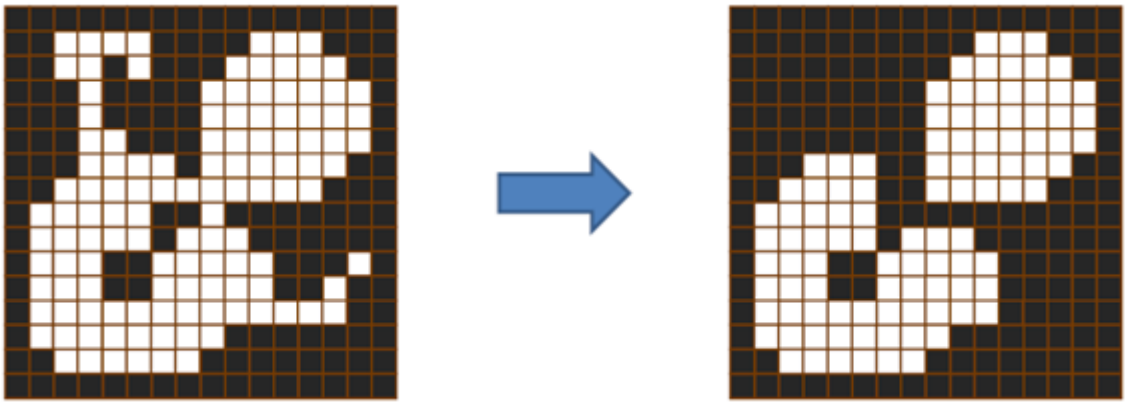


Figura 3.8 Dilatación de una imagen. Izquierda original, derecha resultado

3.4. Entornos de desarrollo tridimensionales

Uno de los objetivos de este proyecto es representar las operaciones realizadas para que sea más sencilla la iteración con el operario, sabiendo este en todo momento de una forma intuitiva si está realizando el trabajo de una manera correcta y eficiente.

Dentro del desarrollo en 3D en la plataforma Microsoft Windows existen dos grandes grupos: la herramienta XNA desarrollada por Microsoft y la API Direct3D de la colección DirectX. Para poder utilizar estas herramientas en aplicaciones escritas

en lenguaje C# se necesitan envoltorios como ANX.Framework [21] para XNA y SharpDX [22] y SlimDX [23] para DirectX.

Estas herramientas nos facilitan métodos para construir entornos 3D. Como el tiempo utilizado para desarrollar este entorno estaría fuera de los límites del proyecto, se opta por utilizar un entorno ya construido que facilite las tareas de modelado.

Entre los diferentes entornos se ha elegido Unity[24] para implementar este proyecto por su manejo intuitivo, multitud de guías en internet [25] de primeros pasos y por utilizar .Net para escribir los scripts asociados a los objetos. Unity es un ecosistema para el desarrollo de videojuegos, pero como ya veremos con detalle en el apartado 5.2. de este documento, lo utilizaremos para simular un lugar de trabajo en el que las herramientas sean movidas al igual que lo hacen las herramientas reales captadas por nuestro programa de seguimiento de marcadores.

A continuación se describirán varios conceptos de Geometría en tres dimensiones aplicados al entorno Unity.

Coordenadas Cartesianas

Los ejes de un sistema de coordenadas cartesiano son perpendiculares entre sí y se cortan en un punto, llamado origen de coordenadas. Cada eje tiene una letra asignada, siendo en Unity “x” el eje horizontal, “y” el vertical y “z” el de profundidad. Es importante señalar que Unity utiliza el Sistema de Coordenadas de la mano izquierda, mientras Kinect utiliza el de la mano derecha.

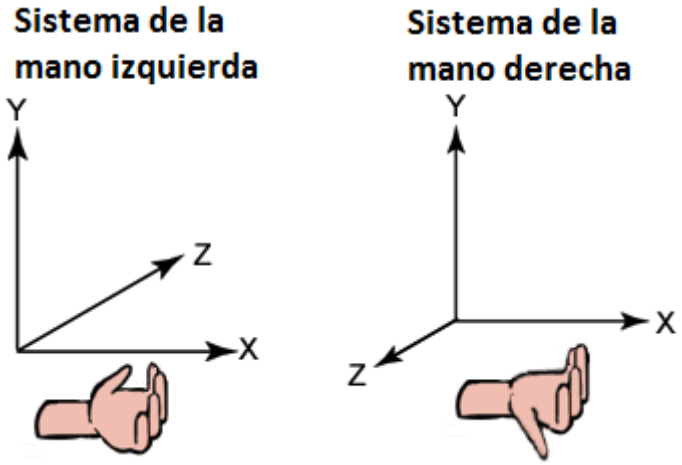


Figura 3.9 Coordenadas Cartesianas. Mano Izquierda vs Mano Derecha

Espacios de Coordenadas

En un motor de videojuegos como Unity podemos distinguir diferentes espacios de coordenadas:

- Espacio Global: el centro de coordenadas es el (0, 0, 0) de nuestro mundo virtual.
- Espacio Local: en este caso el centro de coordenadas es una posición dentro del espacio global, referenciado al objeto con el que estés trabajando. Esto implica tanto la posición como la rotación.

Operaciones con vectores

Existe el objeto Vector3 ya definido por Unity formado por las variables de tipo float (x, y, z). Este objeto contiene gran variedad de métodos que facilitan todas las operaciones matemáticas que explicaré ahora.

- Suma: se utiliza en entornos 3D para calcular un desplazamiento desde una posición conocida. se suman los componentes por parejas. El resultado es otro vector.

$$V1 + V2 = (V1x + V2x, V1y + V2y, V1z + V2z) \quad (3.6)$$

- Resta: usada en entornos 3D para obtener la dirección entre dos puntos. Se restan los componentes por parejas. El resultado es otro vector.

$$V1 - V2 = (V1x - V2x, V1y - V2y, V1z - V2z) \quad (3.7)$$

- Escalar un vector: útil para eliminar un componente del vector. Se multiplican los componentes de los dos vectores por parejas. El resultado es un vector.

$$V1 \cdot V2 = (V1x \cdot V2x, V1y \cdot V2y, V1z \cdot V2z) \quad (3.8)$$

- Normalizar un vector: La normalización de un vector consiste en asociarle otro vector unitario, de la misma dirección y sentido que el vector dado, dividiendo cada componente del vector por su módulo.

$$\vec{u}_{\vec{v}} = \frac{\vec{v}}{|\vec{v}|} \quad (3.9)$$

Ángulos de Euler

Los ángulos de Euler constituyen un conjunto de tres coordenadas angulares que sirven para especificar la orientación de un sistema de referencia de ejes ortogonales, normalmente móvil, respecto a otro sistema de referencia de ejes ortogonales normalmente fijos.

Dados dos sistemas de coordenadas xyz y XYZ con origen común, es posible especificar la posición de un sistema en términos del otro usando tres ángulos α, β y γ. La definición matemática es estática y se basa en escoger dos planos, uno en el sistema de referencia y otro en el triedro rotado. En el esquema de la imagen serían los planos xy y XY.

La intersección de los planos coordenados xy y XY escogidos se llama línea de nodos, y se usa para definir los tres ángulos:

- α es el ángulo entre el eje x y la línea de nodos.
- β es el ángulo entre el eje z y el eje Z.
- γ es el ángulo entre la línea de nodos y el eje X.

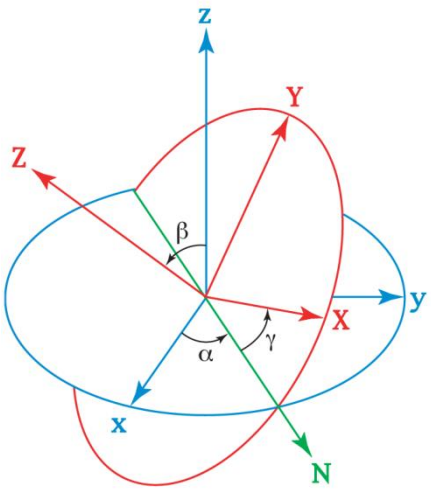


Figura 3.10 Dos sistemas de coordenadas ortogonales en el que se muestran los ángulos de Euler

4. Diseño del Sistema de Monitorización

El Sistema de Monitorización de Operaciones Manuales es un conjunto de aplicaciones que sirven como elemento de apoyo a los operarios en la realización de operaciones manuales realizadas con diversas herramientas. El sistema reconoce la realización de las tareas mediante el seguimiento de la posición y orientación de la herramienta.

El diseño del Sistema de Monitorización de Operaciones Manuales desarrollado en este proyecto está dividido en diferentes apartados. El primer paso es el Seguimiento de Marcadores, unos objetos colocados en la herramienta que nos sirven para conocer la posición y orientación de esta en el espacio tridimensional. Un seguimiento robusto de estos marcadores hará que el sistema funcione correctamente. En el primer y segundo apartado de este capítulo hablaremos de cómo se realiza el seguimiento de los mismos y de cómo se construyen estos marcadores.

Una vez realizado el seguimiento de los marcadores se debe definir la posición y orientación de la herramienta a partir de la posición de los mismos. Esto será analizado en el tercer apartado del capítulo.

El objetivo final del seguimiento de la herramienta es el poder monitorizar operaciones en tiempo real. En el cuarto apartado de este capítulo veremos como a partir del conocimiento de la orientación y posición del efector final de la herramienta se puede determinar si el operario ha realizado o no una acción.

El Sistema de Monitorización desarrollado podrá ser utilizado en un entorno de trabajo concreto. Este será desarrollado en el último apartado de este capítulo.

4.1. Seguimiento de Marcadores

Trabajos Base

El Seguimiento de Marcadores de colores de este proyecto está basado en los trabajos [26] y [27].

Los métodos que defienden estos trabajos se basan en utilizar el sensor Kinect para realizar un seguimiento en tres dimensiones de objetos diferenciados por su color. El objeto es segmentado usando un filtro HSV (Hue, Saturation, Value) sobre la información facilitada por el sensor RGB de la cámara.

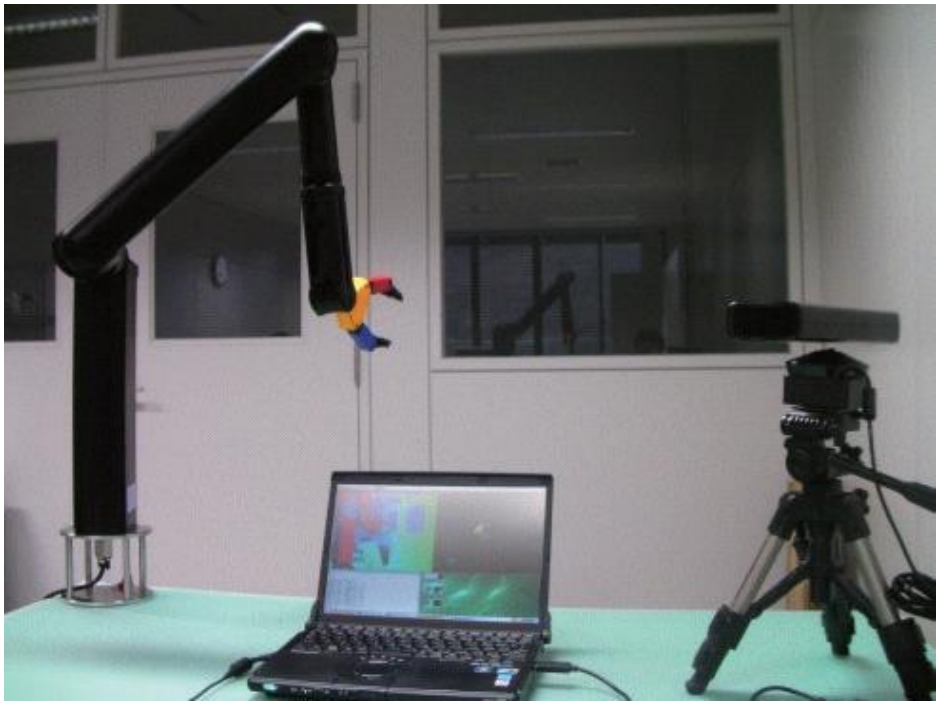


Figura 4.1 Vista del Entorno de Experimentación en [27]

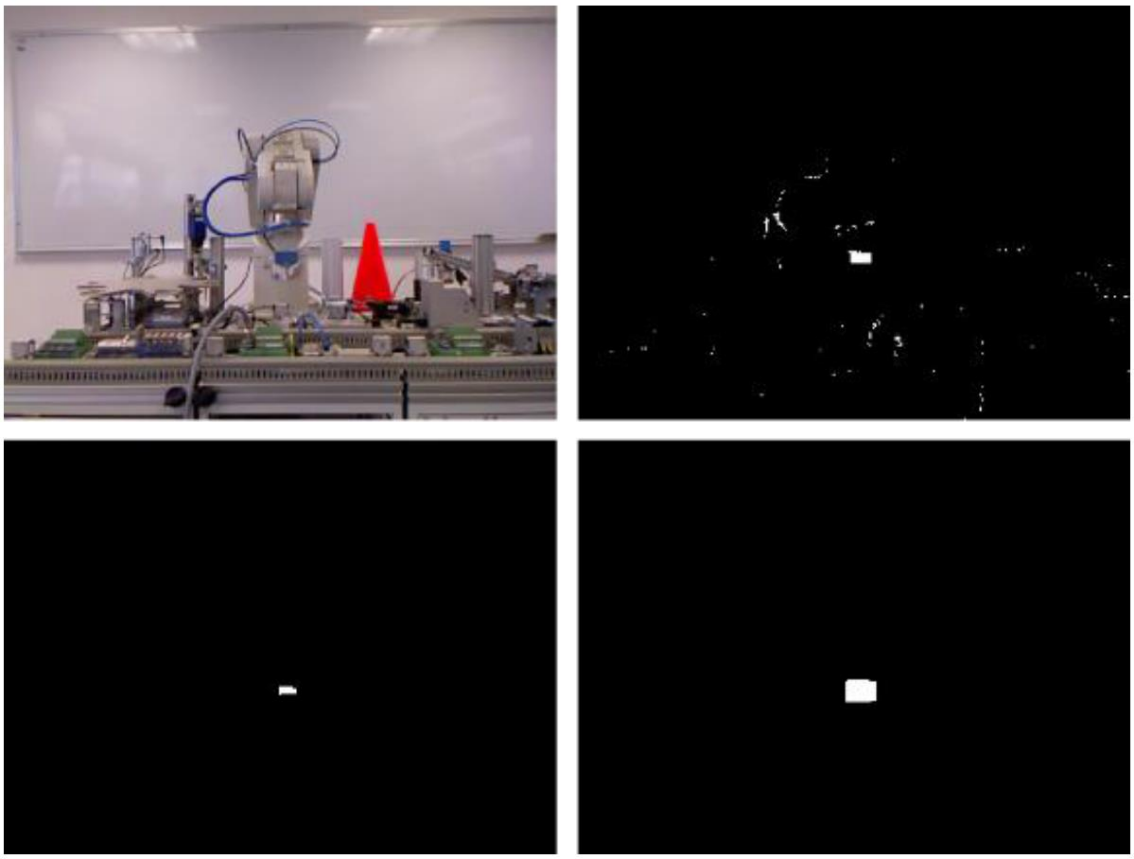


Figura 4.2 De Izquierda a derecha y de arriba abajo: RGB imagen, tras filtro HSV, Erosión, Erosión+Dilatación en [26]

Los pasos que siguen para conseguirlo en [26] son los siguientes:

1) Recogen los datos de color y profundidad del sensor Kinect.
2) Redimensionar la imagen RGB de 640x480 a 320x240 pixeles para aumentar la velocidad de procesamiento.
3) Convertir la imagen del modelo de color RGB a HSV.
4) Segmentar el objeto a seguir filtrando por color.
5) Eliminar el ruido con una apertura.
6) Obtener el centroide del objeto.
7) Obtener el valor de profundidad del centroide.

Este documento [26] además ofrece los resultados experimentales en los que se compara los puntos recogidos por el Sistema de Seguimiento con el movimiento realizado por una pinza de color azul acoplada a un robot. Los resultados obtenidos en este experimento arrojan los valores en la Figura 4.3.

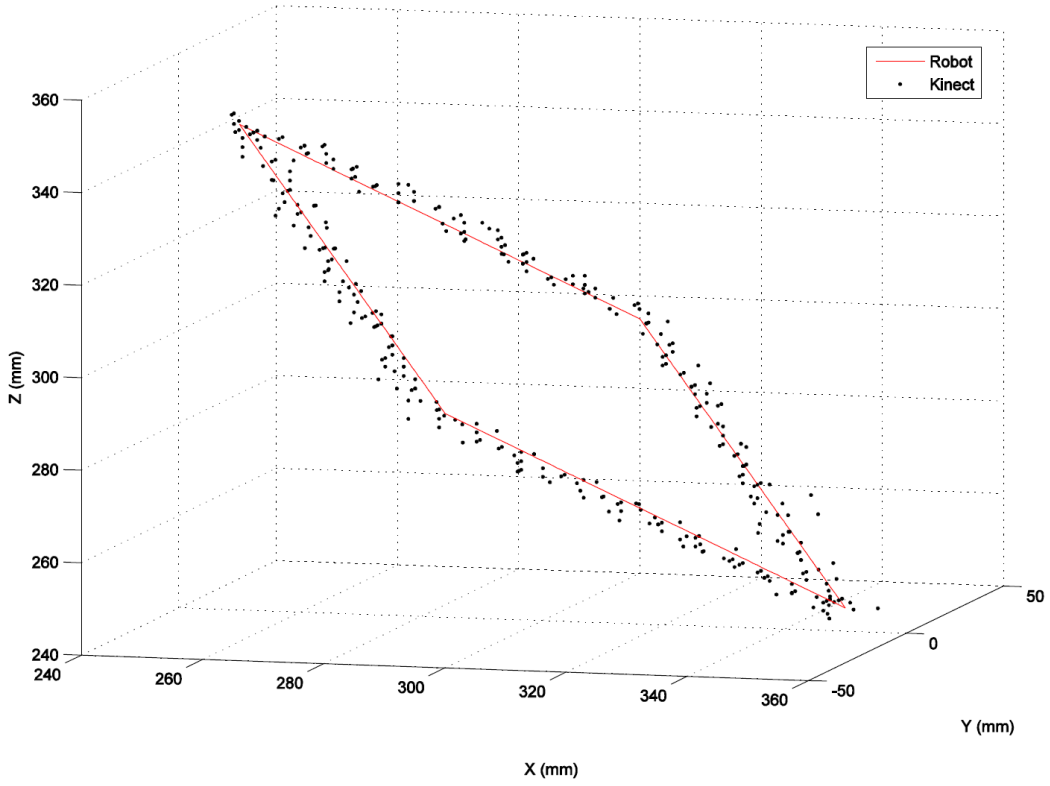


Figura 4.3 Trayectoria descrita por la pinza (línea) y las medidas de seguimiento ofrecidas por el sensor Kinect (puntos). [26]

Además ofrecen una gráfica con los errores de las mediciones del sensor, para cada eje, con respecto a la trayectoria descrita por la pinza (Figura 4.1). El procedimiento general para la segmentación, el post-procesamiento y la visualización de los resultados en la interfaz gráfica de usuario tiene un tiempo medio de 5 a 6 milisegundos a una velocidad de 30 Hz.

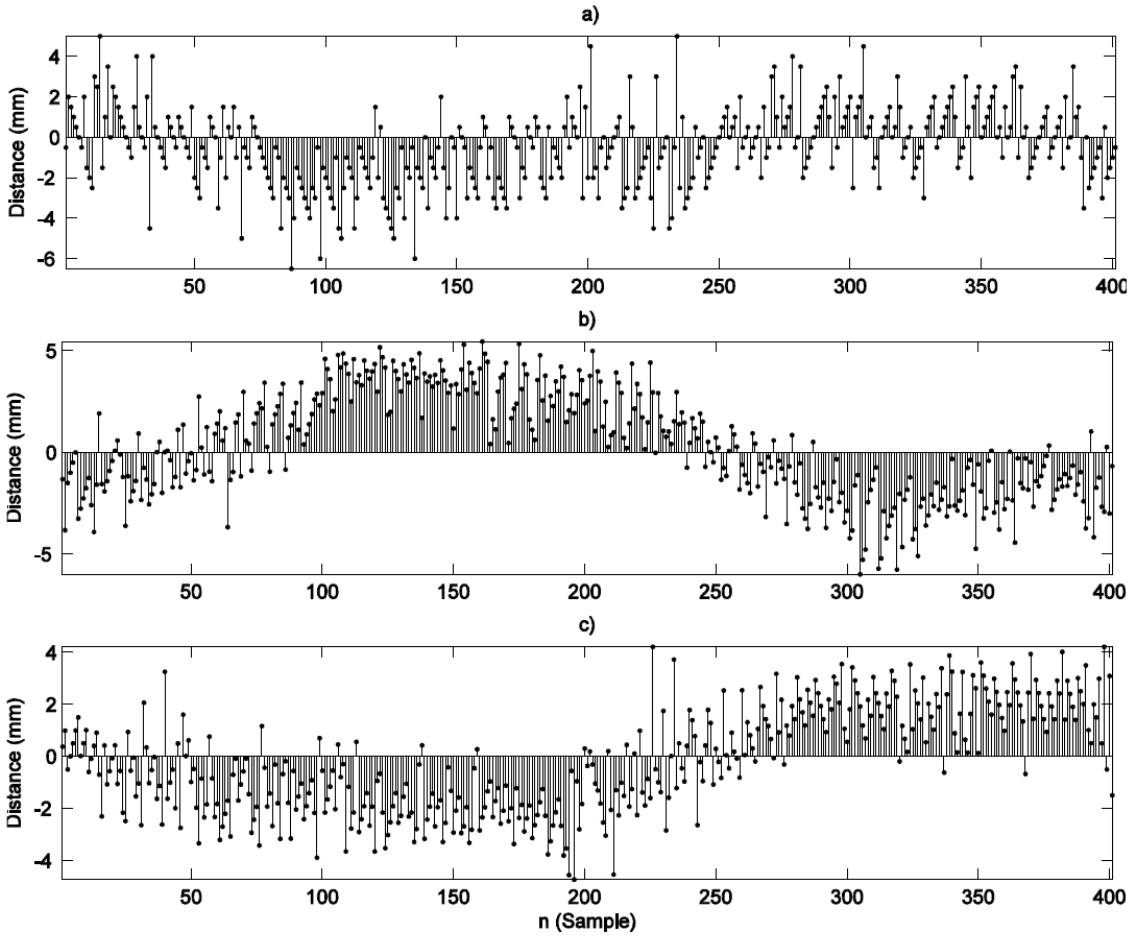


Figura 4.4 Error de seguimiento: a) Xc-Xtarget [mm vs n], b) Yc-Ytarget [mm vs n], c) depth(Xc, Yc) – Ztarget [mm vs n].[26]

Concluyen este trabajo diciendo que como se puede observar a partir de los resultados experimentales que el enfoque propuesto cumple con el seguimiento basado en el color 3-D logrando una buena precisión mediante un sistema de visión de bajo coste.

Diseño del Seguimiento de Marcadores

A la vista de los buenos resultados de este trabajo, se decidió realizar un seguimiento de marcadores de colores de la imagen a color facilitada por el sensor RGB de Kinect y posteriormente, gracias a los datos facilitados por el sensor de profundidad, sacar las coordenadas tridimensionales de estos.

Como vamos a necesitar el seguimiento de varios marcadores es conveniente el uso de hilos cíclicos que realicen el tratamiento sobre la imagen y obtengan el valor de las coordenadas tridimensionales simultáneamente cada vez que el sensor RGB actualiza la imagen.

El sensor Kinect permite una velocidad máxima de 30 fotogramas por segundo a una resolución de 640*480 pixeles. La cantidad de información que se ha de procesar a la vez satura el sistema, por lo que es necesario realizar el tratamiento de la imagen sobre Regiones de Interés (ROI). De esta manera se reduce enormemente la cantidad de memoria utilizada por la aplicación.

La Región de Interés será individual y dinámica para el seguimiento de cada marcador. La imagen RGB facilitada por el sensor será clonada en tantas imágenes como marcadores haya de seguirse. La posición del rectángulo de recorte del ROI de cada imagen irá variando en relación con la posición del centro del marcador seguido. Su tamaño será estable y se determinará idóneo por medio de experimentación.

Es necesario tener en cuenta la posibilidad de perder el marcador, es decir, que el marcador buscado no se encuentre dentro de la ROI recortada para ese marcador. Esto puede pasar por estar oculto tras algún objeto u otro marcador, o por haberse producido un movimiento brusco. Cuando esto ocurre ha de ejecutarse un método que logre encontrar este. Se plantean dos posibles soluciones:

1) Buscar el marcador en toda la imagen de forma progresiva:

   Partiendo de la última posición encontrada del centro del marcador, aumentar la región de interés de manera progresiva hasta volver a encontrarlo. Una vez encontrado, reducirlo hasta su tamaño mínimo.

2) Buscar el marcador en la posición de los otros marcadores:

   Bajo la premisa de que los marcadores han de encontrarse cercanos y que es poco probable que se oculten los tres marcadores a la vez, este método puede ser válido. Cuando un marcador se pierde, su ROI debe ser el de un marcador que actualmente este siguiéndose convenientemente. En el caso de que los tres marcadores se hayan perdido, se buscarán estos en el centro de la imagen.

Una vez obtenida una imagen de un tamaño adecuado se procederá al tratamiento de esta imagen para lograr una imagen binaria, donde el objeto sea el marcador y todo lo demás sea el fondo. Esto permitirá la búsqueda del centro del marcador.

Los pasos a realizar en el tratamiento de la imagen serán los siguientes:

1) Convertir la imagen del modelo de color RGB a HSV.
2) Segmentar el objeto a seguir filtrando por color.
3) Eliminar el ruido con una apertura.

La segmentación del objeto, será la parte más delicada, y se realizará de forma manual mediante una aplicación que se encargará de encontrar los valores mínimos y máximos de cada plano HSV. La funcionalidad de esta aplicación

denominada “Configuración de Seguimiento” será explicada en el siguiente capítulo.

A continuación se buscará el centro del marcador. Para ello se buscará los contornos de la imagen, lo que nos permite conocer las coordenadas x e y en pixeles de la imagen donde se encuentra el centro del marcador.

Una vez que es sabida la posición del marcador en la imagen a color es hora de obtener sus coordenadas en el espacio. Kinect para Windows ofrece un Espacio de Coordenadas tridimensional el cual llaman “Espacio de Esqueleto”.

El Espacio Esqueleto es un sistema de coordenadas de mano derecha expresado en metros. Los ejes X, Y, y Z son los ejes del sensor de profundidad siendo el origen el sensor y la dirección del eje Z positivo la dirección hacia donde señala el sensor. El eje Y positivo se extiende hacia arriba y el X positivo hacia la izquierda.

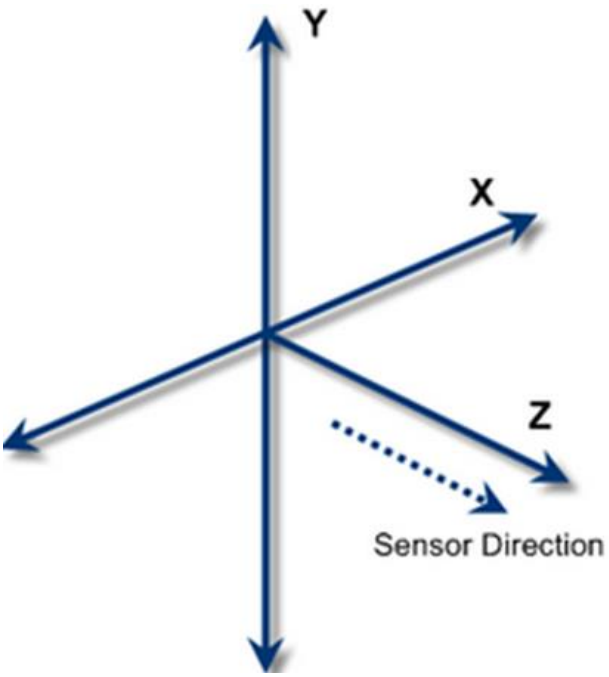


Figura 4.5 Ejes del Espacio Esqueleto

Dado un espacio bien definido, la solución es la siguiente. Se necesita asignar cada pixel de la imagen a color al Espacio de Esqueleto. Kinect para Windows ofrece métodos de asignación de coordenadas cuyo resultado será el de asignar al pixel de la imagen a color correspondiente al marco de profundidad asignado, y luego trasformado al Espacio de Esqueleto. Hay q tener varias consideraciones en cuenta [28]:

- Cualquier pixel de color sin un pixel de profundidad correspondiente dará lugar a un valor centinela, es decir, los puntos resultantes pueden ser comprobados.

- Debido a que los datos de imagen de profundidad y los datos de imagen de color provienen de sensores separados, los píxeles de las dos imágenes no siempre pueden alinear exactamente. Los dos sensores pueden tener diferentes campos de visión, o no podrían tener el objetivo, precisamente, en la misma dirección. Esto significa que un punto cerca del borde de la imagen de profundidad puede corresponder a un píxel justo más allá del borde de la imagen en color, o viceversa.

De esta manera ya tenemos los marcadores localizados en un espacio de coordenadas definido por sus ejes x, y, z en unidades expresadas en metros.

4.2. Construcción de Marcadores

Trabajo Base

El diseño de los marcadores utilizados en la aplicación Seguimiento de Marcadores está basado en los localizadores que utiliza el Sistema Robótico NeuroMate.

El Sistema Robótico NeuroMate [29] (Integrated Surgical Systems, Davis, CA) es un Sistema comercial de Robótica Asistida guiada por imagen utilizado para los procedimientos de estereotaxia en neurocirugía [30].

Figura 4.6 Sistema Robótico Neuromate [31]

Figura 4.7 Detalle de localizador de imagen situado en la herramienta en el Sistema Robótico NeuroMate[31]

Este Sistema utiliza unos localizadores de imagen (ver imagen 1) situados en la base del cráneo del paciente para construir una imagen preoperatoria, tales como CT, para identificar la región de la base del cráneo que puede ser perforado de forma segura (mediante la definición de un artefacto virtual). Una vez que esta imagen preoperatoria se ha registrado en los sistemas de coordenadas del robot, se puede asegurar que la herramienta de corte (ver imagen 2) montada en el robot permanece dentro de esta zona de seguridad.

Diseño de los Marcadores

La disposición de los marcadores que forman el localizador de imagen utilizado y su forma redonda inspiraron al marcador utilizado en el Sistema de Seguimiento de Operaciones implementado en este proyecto.

Para garantizar la máxima estabilidad del Sistema se necesita que la luz que incide en los marcadores sea lo más homogénea posible, garantizando de este modo que el sensor RGB de Kinect capte el mismo color sin depender en la posición en el espacio tridimensional en el que se encuentre.

Si el marcador tiene una forma esférica, se consigue que independientemente de la orientación respecto al Kinect, la superficie que se proyecta en la imagen siempre sea de forma redonda y de un radio que solo dependa de la profundidad a la que se encuentre y no de la posición x e y.

Para que el reflejo sobre los marcadores sea totalmente difuso, evitando reflejo especular, es necesario que:

a) El marcador disponga de luz propia. Esto se puede conseguir agrupando varios leds revestidos con silicona semitransparente, de la utilizada para la construcción de moldes.
b) Utilizando un material no metálico que no produzca brillos indeseados, como puede ser la arcilla de modelado.

Obviamente la construcción de varios marcadores con luz propia sería más tediosa. La arcilla de modelado permite moldear rápidamente y de forma manual cualquier forma deseada. Existen multitud de colores en el mercado por lo que con utilizar el color adecuado, en pocos minutos tendríamos construido el marcador.

Por otro lado, si se escogen leds RGB (o leds multicolor) se puede ajustar el color de estos para conseguir una binarización más correcta. Además se pueden apagar y encender cuando se requiera seguir una herramienta u otra. No obstante, esto añadiría cableado en la herramienta, la necesidad de programar un controlador y sincronizar este con la aplicación desarrollada.

A la vista de estas consideraciones se decide construir marcadores de varios colores con forma esférica con arcilla de secado rápido.

Figura 4.8 Arcilla de modelado

Figura 4.9 Leds RGB

4.3. Seguimiento de Herramientas

Una vez construido los marcadores, se necesita colocar estos en la herramienta de tal forma que se pueda relacionar sus coordenadas x, y, z con la posición y orientación en el espacio tridimensional de la herramienta.

Existen multitud de herramientas en el mercado para la realización de operaciones manuales. Para realizar el seguimiento de las operaciones realizadas con ellas lo importante es conocer la orientación y posición del efector final, es decir, la parte de la herramienta que manipula la pieza con la que se está trabajando.

Para conseguir esto es necesario el seguimiento de tres marcadores los cuales forman los puntos denominados M1, M2 y M3. Dadas las posiciones en el espacio de estos, se pueden definir dos vectores tridimensionales:

- VM1M2, Vector dirección entre los puntos M1 y M2. Se obtiene por la resta de M2 menos M1.

$$VM1M2 = M2 - M1$$

- VM1M3, Vector dirección entre los puntos M1 y M3. Se obtiene por la resta de M3 menos M1.

$$VM1M3 = M3 - M1$$

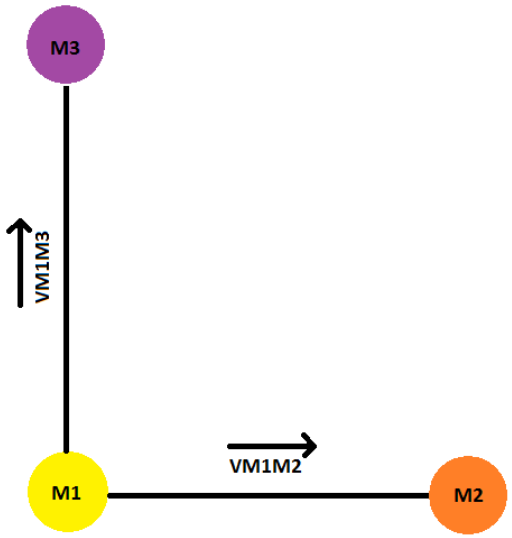


Figura 4.10 Puntos M1, M2, M3 y los vectores VM1M2, VM1M3

Orientación de la Herramienta

Se quiere definir la orientación de la herramienta en ángulos de Euler. Para ello se va a utilizar la dirección de los vectores VM1M2 y VM1M3. Los ángulos de Euler se obtendrán midiendo el ángulo que forma cada vector unitario del sistema de coordenadas con unos vectores que denominaremos vectores de comparación.

El procedimiento para hallar cada vector de comparación consiste en eliminar una de las componentes del vector, magnificar las restantes y normalizar ese vector. Para cada ángulo de Euler:

- Ángulo x: se escala el vector VM1M3 de tal manera que se elimine la componente X y se magnifiquen la componente Y, y Z. El ángulo X será el formado por este vector y el vector unitario z (0, 0, 1).
- Ángulo y: se escala el vector VM1M2 de tal manera que se elimine la componente Y, y se magnifiquen la componente X y Z. El ángulo Y será el formado por este vector y el vector unitario x (1, 0, 0).
- Ángulo z: se escala el vector VM1M2 de tal manera que se elimine la componente Z y se magnifiquen la componente X e Y. El ángulo Z será el formado por este vector y el vector unitario x (1, 0, 0).

Ángulo Calculado	Vector inicial	Transformación del vector *	Vector comparación	Valores unitario
Ángulo X	VM1M3	(0, 10, 10)	Comparación x	Vector z (0, 0, 1)
Ángulo Y	VM1M2	(10, 0, 10)	Comparación y	Vector x (1, 0, 0)
Ángulo Z	VM1M2	(10, 10, 0)	Comparación z	Vector x (1, 0, 0)

Tabla 4.1 Obtención de los ángulos de Euler

* Se magnifican las componentes que no se eliminan multiplicándolas por la constante 10 para que a la hora de normalizar el vector este sea de un tamaño mayor que la unidad.

Posición del efector final de la herramienta

En una operación realizada con una herramienta nos interesa saber la posición del efector final de la herramienta. Esta posición estará trasladada unos valores x, y, z de la posición de un marcador. Este marcador será la referencia de la posición en todo momento y a partir de ahora nos dirigiremos a él como M1.

Multiplicando el vector dirección local de la herramienta normalizado (*velan*) por una constante de desplazamiento (*cd*) obtenemos un vector el cual tiene la dirección local de la herramienta y el tamaño igual a este desplazamiento. A estos vectores les llamaremos traslación (*tras*) x, traslación y, y traslación z.

$$traslx = vdlhnx * cdx \quad (4.1)$$

$$trasly = vdlhny * cdy \quad (4.2)$$

$$traslz = vdlhnz * cdz \quad (4.3)$$

El desplazamiento (desp) en x será igual a la suma de la componente x de dirección x, más la suma de la componente x de dirección y, más la suma de la componente x de dirección z. Lo mismo ocurre para los otros dos desplazamientos.

$$despx = traslxx + traslyx + traslzx \quad (4.4)$$

$$despy = traslxy + traslyy + traslzy \quad (4.5)$$

$$despz = traslxz + traslyz + traslzz \quad (4.6)$$

La componente x de la posición del efector de la herramienta (*pefh*) será igual a la componente x de M1 más la distancia en la dirección x entre ambos, es decir, el desplazamiento en X calculado. Lo mismo ocurre con las otras dos componentes,

obteniéndose la posición del efector final de la herramienta formada por sus componentes x, y, z.

$$pefhx = M1x + despx \quad (4.7)$$

$$pefhy = M1y + despy \quad (4.8)$$

$$pefhz = M1z + despz \quad (4.9)$$

$$pefh(x, y, z) = (pefhx, pefhy, pefhz) \quad (4.10)$$

4.4. Monitorización de operaciones

Existen dos objetivos a realizar en la monitorización de operaciones. En primer lugar ofrecer al usuario la posibilidad de visualizar por pantalla en tiempo real la representación de la operación llevada a cabo. De esta forma puede comprobar si la actividad se está realizando de forma adecuada, corrigiendo errores en el momento. El otro objetivo a cumplir es el de almacenar las actividades realizadas para comparar posteriormente la actividad realizada con una actividad de referencia.

Las acciones que se pueden monitorizar con este sistema son:

- Lijado con Lija Eléctrica.
- Corte con Sierra de Calar.
- Corte con Radial o Amoladora
- Taladrado y Atornillado (con destornillador eléctrico).
- Fresado.

En definitiva sirve para monitorizar acciones realizadas con alguna herramienta en la que el movimiento y orientación de esta pueda determinar el trabajo realizada sobre una pieza fija.

Para facilitar la ordenación de las acciones realizadas por el usuario se usa una organización jerárquica. Una operación está dividida en diferentes actividades, y estas a su vez en superficies.

Una actividad simplifica el movimiento realizado con una herramienta en la realización de una acción en un conjunto de superficies que la definen. Estas superficies representan la forma y dimensiones de la superficie del efector final que está en contacto con la pieza. La orientación y posición de estas superficies será lo que determine si una actividad se ha realizado y si se ha hecho con una orientación similar a la actividad de referencia.

Figura 4.11 Organización Jerárquica de la Monitorización de Operaciones

Por lo tanto son necesarios dos tipos de actividades: actividades de **superficies de referencia** y actividades de **superficies de comparación**.

Las propiedades asociadas a una actividad de referencia son las siguientes:

- El nombre que se le asocia a la actividad, lijado de pieza 1.
- La herramienta utilizada está definida por su nombre y el desplazamiento desde el marcador referencia hasta el efector final. Una actividad se realiza con una única herramienta.

- Algunas herramientas no van a tener en cuenta la rotación sobre alguno de sus ejes. Por ejemplo en la realización de un taladro la orientación sobre el eje perpendicular al taladro no es relevante ya que no determinar si el agujero se ha realizado de forma adecuada. Esto se tiene en cuenta como propiedad de la actividad, ya que se puede descartar la restricción de la orientación sobre un eje a la hora de determinar si se ha completado cierta superficie de la actividad.

- Existirán actividades que requieran una mayor precisión que otras, por lo que se podrá seleccionar el rango válido de la posición y el de los ángulos de Euler.

- Con el fin de poder visualizar adecuadamente las superficies se permite elegir la posición de la cámara mientras se realiza la actividad.

- Es necesario almacenar la fecha de inicio y la duración de la actividad con el fin de comparar ciertas actividades.

- Como medio meramente informativo, se pueden asociar un comentario indicativo que defina aspectos que no tienen que ver con las propiedades descritas, como puede ser un cambio de accesorio de la herramienta, la trayectoria conveniente, etc.

- La forma de la superficie del efector final que está en contacto con la pieza a tratar difiere con cada herramienta o con cada accesorio utilizado. Por lo general, estas superficies se pueden simplificar en elipses y rectángulos.
- Una vez definida la forma es necesario definir las dimensiones de las superficies generadas para identificar la actividad.

Para el caso de actividades de superficies de comparación solo serán necesarias dos propiedades: la fecha de creación de la actividad y la duración de la misma. Las demás propiedades serán iguales a las de la actividad de referencia con la que se comparan, por lo que no es necesario definirlas.

Ambas actividades están formadas por las superficies que la definen. Se almacena la orientación y posición del efector de la herramienta en el momento de generarse una superficie. El conjunto de superficies más las propiedades forman la actividad.

Interfaz de Usuario

La interfaz de usuario de la aplicación utiliza distintos modos de funcionamiento: un primer modo para la configuración de las propiedades de la actividad y de los colores del Seguimiento de Marcadores, un segundo modo para el almacenado de superficies de referencia, un tercero para la modificación de dichas superficies, un cuarto para el guiado a partir de superficies de comparación y un quinto que compare ambas superficies.

Modo 1, Configuración del Sistema

Este sistema está pensado para que se ejecute de una forma eficiente con independencia de los marcadores utilizados y de las herramientas utilizadas. Para que esto sea posible, es necesaria la configuración del sistema.

Dependiendo de la zona de trabajo existirán ciertos colores que no podrán ser diferenciados del resto de colores de las piezas y de la herramienta. Como ya se ha

dicho anteriormente, es de gran importancia seleccionar adecuadamente estos colores para que la segmentación permita una binarización eficiente entre el marcador y la zona de trabajo. Para facilitar al usuario esta labor se ha desarrollado una aplicación que permita seleccionar los colores de una forma rápida.

Una vez seleccionado adecuadamente los colores que definen los marcadores de cada herramienta se debe configurar la actividad a realizar. Se facilita un programa que permite configurar las propiedades de la actividad.

Modo 2, Nueva Actividad

La creación de una actividad se basa en almacenar las superficies de referencia. Para que el usuario sea consciente de cómo están siendo almacenadas se representa por pantalla en tiempo real la posición del efector final de la herramienta y las superficies generadas. Además en este modo el usuario puede cambiar la posición y orientación de la cámara con el fin de facilitar el guiado.

Modo 3, Modificación de Actividades

Posteriormente el usuario puede rechazar superficies de referencia indeseadas almacenadas durante la creación de la actividad.

Modo 4, Guiado de Actividad

El cuarto modo tiene como objetivo guiar al operario en la realización de las actividades. Una vez facilitado por el usuario la actividad a realizar la aplicación carga la actividad con todas sus propiedades asociadas. En este momento se muestra por pantalla las superficies que debe completar, la herramienta a utilizar, el tiempo que debe tardar en completar la actividad y los comentarios informativos que hayan sido incluidos en la configuración.

Cuando el efector final, representado en todo momento en la pantalla, “choca” con una superficie de referencia se provoca un evento el cual puede desencadenar dos acontecimientos: si el efector no tiene la orientación correcta, la superficie de referencia cambiará a un color naranja, indicando así al usuario que la orientación no es la adecuada; en el caso de ser una orientación correcta, se elimina la superficie de referencia y se genera una superficie de comparación de color verde con su misma forma y dimensión pero con la orientación y posición del efector de la herramienta en ese momento.

El tiempo que se tarda en realizar la actividad está determinado por la diferencia entre la fecha en la que se completó la primera y la última superficie.

Cuando una actividad ha sido completada se almacena las superficies de comparación formando junto a sus propiedades la actividad de superficie de comparación correspondiente. El usuario entonces decide cambiar a otra actividad

o repetir de nuevo la actividad. Se permite el envío de varias actividades simultáneas de tal forma que cuando una actividad ha sido completada se carga la siguiente actividad, evitando que el usuario deba usar la interfaz de durante la realización de una operación completa. Una vez finalizada se podrá seleccionar la misma u otra operación.

Modo 5, Comprobación

Una vez almacenadas las dos superficies, la de referencia y la de comparación, es momento de pasar a un cuarto modo, denominado **Modo comparación**. En este modo se permite navegar por las distintas actividades. Se cargan tanto las superficies de referencia como las superficies de comparación mostrándose por pantalla las propiedades de la actividad y una lista de resultados donde aparece la fecha de creación, la duración de la actividad y el porcentaje de superficies completadas antes del tiempo límite.

4.5. Simulación de un entorno de trabajo

Una vez definido el diseño de todo el Sistema de Monitorización se puede concretar como es el entorno de trabajo idóneo para el correcto funcionamiento del Sistema.

El entorno de trabajo estará compuesto por un PC con un monitor de alta Resolución (1920*1080) más un teclado y ratón funcionando como interfaz de usuario, una mesa donde colocar la pieza a manipular y las herramientas y un soporte que permita colocar el sensor Kinect en la posición adecuada. Además es necesario añadir iluminación artificial para conseguir la estabilidad del sistema independientemente de la iluminación general de la sala o lugar de trabajo. En la Figura 4.12 se puede ver el entorno de trabajo utilizado durante los ensayos y experimentos realizados durante el proyecto. Ha continuación se van a comentar uno a uno cada elemento necesario en este entorno.

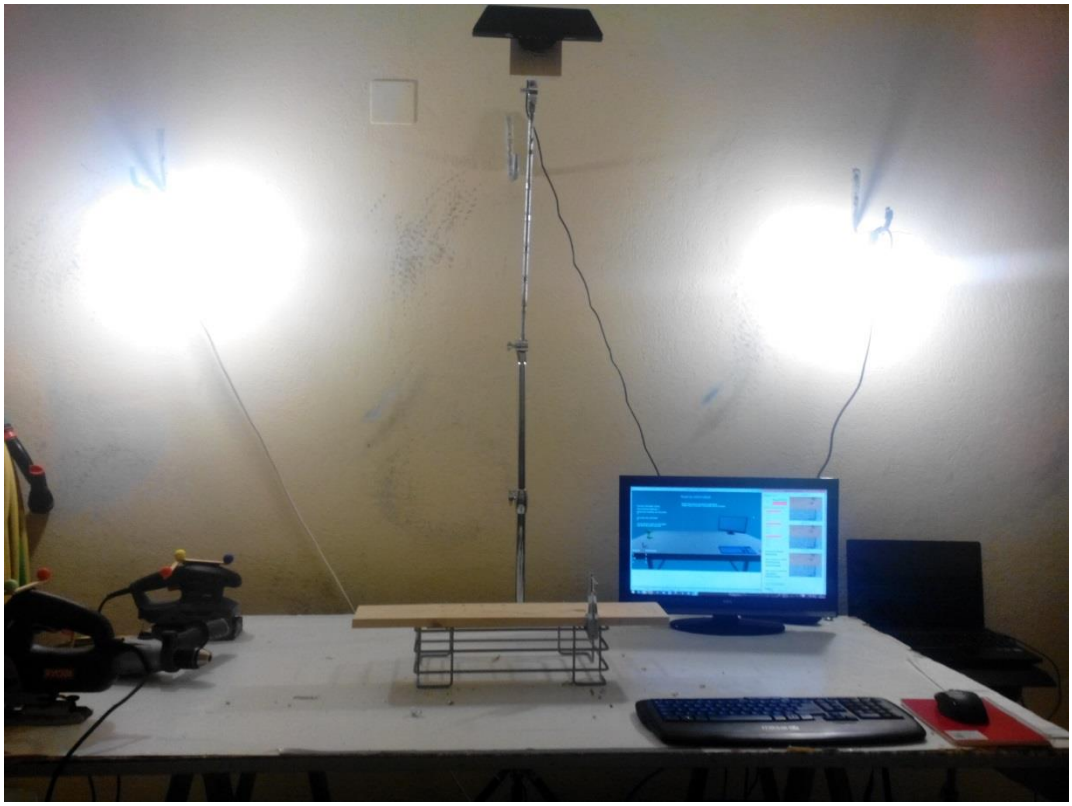


Figura 4.12 Vista frontal del entorno de trabajo

Figura 4.13 Vista lateral del entorno de trabajo

1) Iluminación artificial

Una parte muy importante del entorno de trabajo es la iluminación. Estamos tratando imágenes a color, donde la representación HSV de los colores utilizada para la segmentación de objetos cambiará dependiendo de la iluminación incidente en ellos. Los marcadores a seguir estarán construidos de arcilla de modelado. Utilizando bombillas que producen luz difusa, la reflexión de este material será totalmente difusa.

Para simular un ambiente en el que la luz reflejada en la bola sea lo más homogénea posible, como en la primera esfera de la Figura 4.14, en la que no existe ningún tipo de reflejo es necesario el uso de dos lámparas fluorescentes compactas a cada lado del sensor Kinect. De esta manera la sombra producida por la iluminación desde un lateral de la esfera es ocupada por la luz proveniente de la otra bombilla. Si estas luces están lo suficientemente alejadas de la mesa de trabajo se tendrá un reflejo homogéneo independientemente de la localización del marcador, simplificando enormemente la tarea de segmentación por color.

Figura 4.14 Diferentes ambientes de iluminación

2) Espacio de trabajo

Dentro de la mesa o superficie de trabajo debe existir una zona restringida, donde los únicos elementos que entren en ella sean la pieza sobre la que se vaya a actuar y la herramienta con la que se vaya a realizar la operación. De esta manera se conseguirá un seguimiento adecuado de la operación, evitando que la aplicación del Seguimiento de Marcadores se pierda con elementos extraños.

Figura 4.15 Soporte en el que situar las piezas

3) Posición del sensor Kinect

El sensor debe estar ubicado de tal forma que la focal de sus cámaras abarque la totalidad de la mesa de trabajo. Enfocar más superficie que la necesaria añadiría ruidos a la imagen y disminuiría la cantidad de pixeles disponibles para localizar los marcadores.

Con el fin de simplificar los cálculos en el seguimiento de la herramienta es necesario que la cámara este orientada frente al espacio de trabajo. Es adecuado la utilización de un soporte tipo jirafa que permita colocar la cámara en el centro de la zona de trabajo.

4) Marcadores

Para simplificar el seguimiento de la herramienta, es necesario que los marcadores estén dispuestos como se puede ver en la Figura 4.16. Deben existir tres marcadores dispuestos en el mismo plano. El marcador central, denominado en este proyecto como marcador principal, y otros dos marcadores, marcador 1 y marcador 2, deben estar dispuestos de tal forma que los vectores dirección compuesto por el marcador principal y el marcador 1 forme 90º con el vector dirección compuesto por el marcador principal con el marcador 2.

Figura 4.16 Colocación de los marcadores

Se construyó una estructura para sujetar estos marcadores debido a que no se deseaba pegarlos a la herramienta. No habría ningún problema en pegar los marcadores a la herramienta siempre que estén dispuestos como anteriormente se ha indicado.

5) Teclado y ratón

Es conveniente utilizar un teclado y un ratón inalámbrico para evitar que el cableado entre en la zona de trabajo.

6) Zona de Herramientas

Es conveniente reservar una zona no peligrosa donde dejar las herramientas que no están siendo utilizadas.

7) Monitor

Las aplicaciones han sido desarrolladas pensando en una resolución de 1920*1080. Con una resolución menor, es posible que algunos botones y funcionalidades de las aplicaciones queden ocultos.

5. Implementación de aplicaciones del Sistema de Monitorización

En este capítulo se va a describir la estructura del Sistema y el funcionamiento de las distintas aplicaciones ejecutadas por el sistema. El sistema arranca con la ejecución de un programa principal donde el usuario puede navegar por varios modos de funcionamiento para la realización de distintas tareas. Cada modo de funcionamiento lanza unos programas secundarios los cuales describiremos en los apartados segundo, tercero y cuarto de este capítulo. En el quinto capítulo se describe el flujo de control de los distintos modos de funcionamiento.

5.1. Características del Sistema

5.1.1. Estructura del Sistema

El Sistema de Monitorización de Operaciones Manuales es un conjunto de aplicaciones desarrolladas mediante las herramientas Unity y Visual Studio. Debido a las varias funcionalidades del sistema se han desarrollado varios modos de funcionamiento y diversas aplicaciones de apoyo.

La herramienta Unity nos permite trabajar en un entorno tridimensional con diversos objetos a los cuales se les puede asociar un comportamiento mediante la implementación de programas simples denominados “scripts”. El programa Monitorización de Operaciones será implementado con esta herramienta.

Unity permite la programación basada en la plataforma .NET 2.0 lo que hace imposible la programación del seguimiento de marcadores, el cual utiliza clases de plataformas .NET superiores, en esta herramienta. Por esta razón y por la comodidad que ofrece el desarrollo de aplicaciones WPF, el Sistema de Monitorización desarrollado está compuesto por el programa Monitorización de Operaciones trabajando como servidor junto a aplicaciones cliente WPF implementadas en Visual Studio que permite utilizar las plataformas .NET actuales (la plataforma .NET 4.5 es la plataforma más actual en el momento del desarrollo del sistema).

Durante la ejecución del sistema es posible navegar por los siguientes modos de funcionamiento:

1) Modo 1. Configuración del Sistema.
2) Modo 2. Nueva Actividad.
3) Modo 3. Modificación de Actividades.
4) Modo 4. Guiado de Actividades.
5) Modo 5. Comprobación de Actividades.

En la Figura 5.1 se representa la estructura del Sistema. Las flechas relacionan los modos con el programa cliente ejecutado.

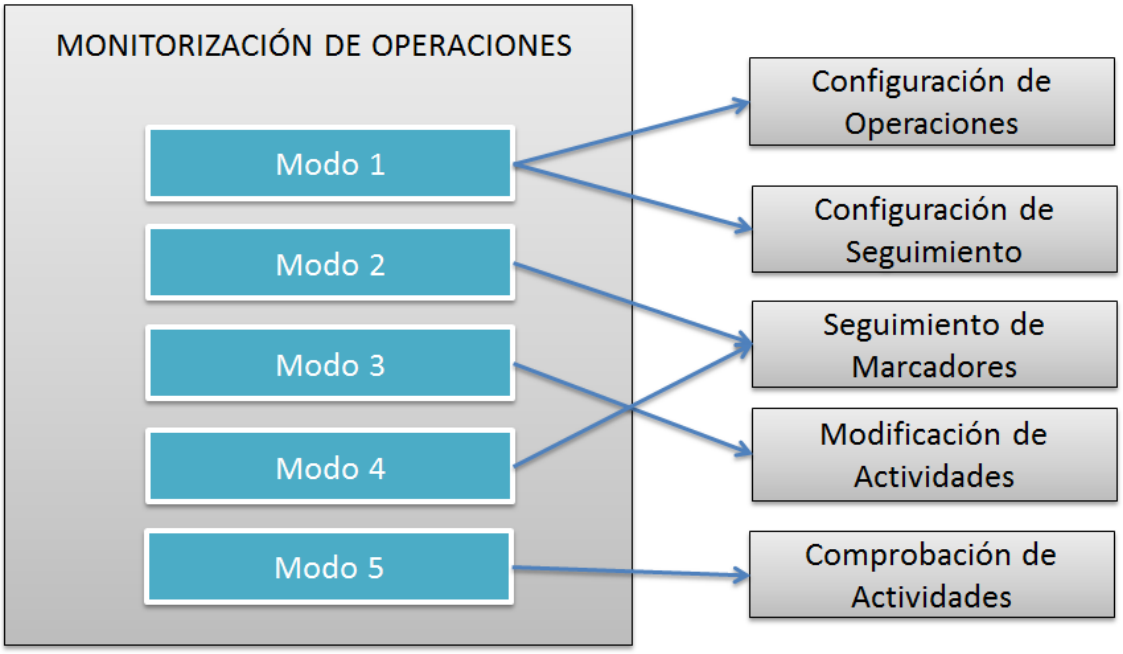


Figura 5.1 Relación entre los modos de funcionamiento y los programas ejecutados

Todas las carpetas con los archivos de configuración de las aplicaciones estarán contenidas dentro de la ruta "C:\MonitorOperaciones\Archivos\Configurar". Son archivos .txt que pueden ser modificados manualmente pero es conveniente modificarlos con las aplicaciones que en este proyecto se han diseñado para ello.

Las actividades se almacenarán en la ruta "C:\MonitorOperaciones\Archivos\Actividades".

5.1.2. Comunicación entre las aplicaciones

Algunos modos del sistema van a estar ejecutando varias aplicaciones de forma simultánea siendo necesario compartir datos entre ellas. Para la comunicación entre estas se va a utilizar una arquitectura cliente servidor. La interfaz Gráfica de Usuario implementado en Unity será el servidor y las aplicaciones secundarias ejecutadas en los distintos modos de funcionamiento serán los clientes.

La comunicación se basa en sockets. Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información.

Por definición del socket, para que dos programas puedan comunicarse entre sí es necesario que inicialicen dos recursos:

- Un par de direcciones del protocolo de red (dirección IP, porque en este caso se va a utilizar el protocolo TCP/IP), que identifican la computadora de origen y la remota.
- Un par de números de puerto, que identifican a un programa dentro de cada computadora.

En nuestro caso, al ejecutarse ambas aplicaciones en el mismo servidor, la dirección IP será la dirección local del ordenador("127.0.0.1"), y el puerto será el mismo tanto en el servidor como en el cliente.

Se van a utilizar dos conexiones, una desde el cliente al servidor y otra del servidor al cliente. La primera utiliza el puerto 13000 y la segunda el puerto 13001. Ambas comunicaciones deben ser iniciadas por el cliente. El servidor solicitara la conexión y espera a que el cliente inicie la comunicación.

En el programa que trabaja como cliente la lectura puede ser asíncrona, es decir, el hilo que ejecuta la lectura se lanza solamente cuando existan datos en el conector que los une. No ocurre lo mismo en el servidor, ya que trabaja con la plataforma .NET 2.0 que no permite hilos asíncronos. Por lo tanto el flujo de datos cliente servidor debe ser continuo para no crear un cuello de botella, retrasando el procesamiento de las otras actividades. Se enviarán datos continuos con las coordenadas en el caso del modo 2 y 4 y un dato que no contiene información valida en los otros modos.

Además debido a las características de los programas construidos con Unity, la aplicación de Monitorización de Operaciones solo puede recibir los datos si tiene el foco del programa. Entonces para enviar comandos desde el cliente al servidor primero deberá enviarse del servidor al cliente un comando diciendo que está listo para recibir datos y en ese momento el cliente enviará el comando.

5.1.3. Ordenador utilizado

El ordenador será el encargado de recibir los datos del sensor Kinect y de ejecutar los programas que se han desarrollado. Se utiliza el portátil Lenovo G510. Las características principales del ordenador utilizado son las siguientes:

- Procesador: Intel Core i7-4702MQ (4x2,20 GHz / 6 MB caché).
- Pantalla / Resolución: 15,6" brillante / 1366x768 píxeles.
- Memoria RAM / HDD: 4 GB RAM DDR3 / 500 GB (5400 RPM).
- Tarjeta gráfica: Intel HD Graphics 4600 (integrada).

- Dimensiones / Peso: 377x250x34 mm / 2,60 kg.
- Otras especificaciones: Windows 8.1, batería 6 celdas, 1xHDMI, 2xUSB 3.0, 1xUSB 2.0, Wifi b/g/n, Bluetooth 4.0, grabador DVD-DL, Webcam HD, lector tarjetas SD.

Figura 5.2 PC utilizado en el proyecto

5.2. Seguimiento de Marcadores

El programa Seguimiento de Marcadores es una aplicación WPF escrita en lenguaje c# bajo la plataforma .NET 4.5. Funciona como cliente encargado de localizar los marcadores en la zona de trabajo mediante los sensores de color y profundidad de Kinect, y de enviar las coordenadas de estos al programa de Monitorización de Operaciones.

Este programa necesita ejecutar ocho hilos simultáneamente:

- Recogida de datos del sensor, ColorFrameReady y DepthFrameReady: dos hilos asíncronos ejecutados cada vez que el sensor Kinect envía los datos de profundidad y de color.
- Búsqueda de marcadores, marcador1_tick, marcador2_tick y marcdor3_tick: tres hilos síncronos, uno para marcador, encargados de convertir la imagen a color en una imagen binaria y de adquirir las coordenadas que corresponden con cada marcador.
- Interfaz de Usuario: un hilo controlador de eventos de los botones utilizados como interfaz de usuario.
- Comunicación con el servidor, envio_Tick y lectura_Tick: el primero es un hilo síncrono encargado del envío de datos al servidor y el segundo es un hilo asíncrono lanzado cada vez que recibe un comando del servidor.

En los siguientes apartados se analizarán estos hilos.

5.2.1. Recogida de datos del sensor

Los datos de la imagen en color están disponible en diferentes resoluciones y formatos. El formato determina si el flujo de datos de imagen en color se codifica como RGB, YUV, o Bayer.

El sensor utiliza una conexión USB que proporciona una determinada cantidad de ancho de banda para pasar los datos. Las imágenes de alta resolución envían más datos por trama y se actualizan con menos frecuencia, mientras que las imágenes de menor resolución se actualizan con más frecuencia, con alguna pérdida en la calidad de imagen debido a la compresión.

Se va a utilizar el formato de color RGB con una resolución de 640*480. Estos datos serán actualizados a una velocidad de 30fps.

Cada fotograma de la corriente de datos de profundidad se compone de píxeles que contienen la distancia (en milímetros) desde el plano de la cámara al objeto más cercano. Estos están disponibles a una resolución máxima de 640*480 pixeles, actualizados a una velocidad de 30fps.

Cada vez que están listos los flujos de datos proporcionados por los sensores RGB y profundidad, se dispara un evento. Para utilizar este evento es necesario registrarlo al inicio de la ejecución de la aplicación mediante los siguientes métodos:

```
//Manejador del evento ColorFrame listo
this.sensor.ColorFrameReady += sensor_ColorFrameReady;
//Manejador del evento DepthFrame listo
this.sensor.DepthFrameReady += sensor_DepthFrameReady;
```

En el inicio de ejecución de la aplicación además ha de realizarse las siguientes acciones:

- Buscar un sensor conectado:

```
foreach (var potentialSensor in KinectSensor.KinectSensors)
{
	if (potentialSensor.Status == KinectStatus.Connected)
	{
		this.sensor = potentialSensor;
		break;
	}
}
```

- Permitir recibir el flujo de datos de color y establecimiento de la resolución:

```
//	Permitir	recibir	stream	de	color
this.sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
//	Permitir	recibir	stream	de	profundidad
this.sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
```

- Establecimiento del rango cercano de profundidad:

```
this.sensor.DepthStream.Range = DepthRange.Near;
```

- Reserva de espacio en memoria para los datos recibidos:

```
//Reservar espacio donde poner los pixeles de profundidad recibidos
this.depthPixels                                =                                new
DepthImagePixel[this.sensor.DepthStream.FramePixelDataLength];
//Reservar espacio donde poner los pixeles de color recibidos
this.colorPixels = new byte[this.sensor.ColorStream.FramePixelDataLength];
```

- Iniciar el sensor:

```
// Start the sensor!
try
{
        this.sensor.Start();
}
catch (InvalidOperationException)
{
        this.sensor = null;
}
```

Flujo de control de la trama de color

Cada vez que se dispara el evento generado cuando una nueva trama de color esta lista se va a realizar el siguiente flujo de control:

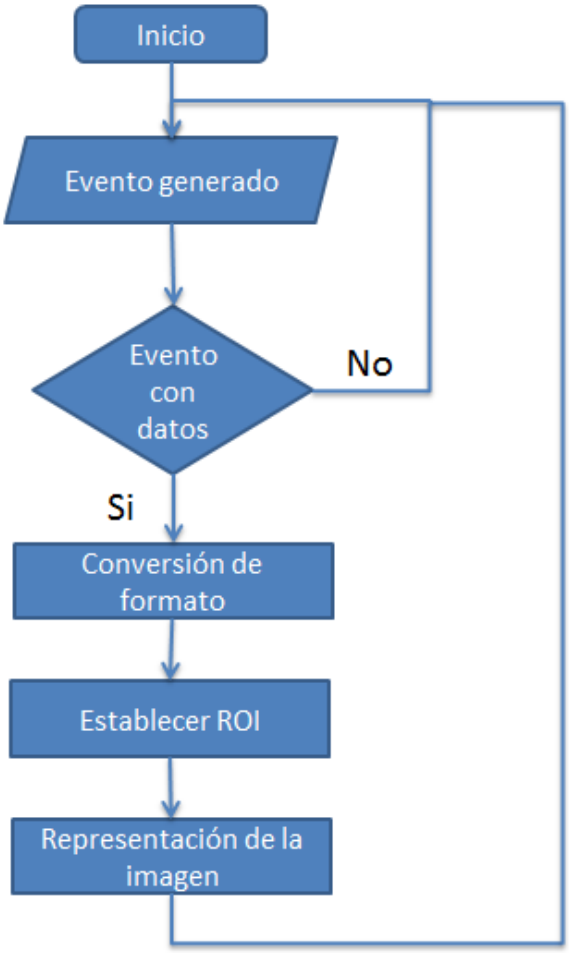


Figura 5.3 Diagrama de Flujo del evento trama de color lista

Es necesaria la comprobación de generación de datos nulos por parte del controlador de Kinect para evitarnos errores en la conversión de la imagen.
Los datos vienen dados como una imagen con el formato BitmapSource. La herramienta EmguCV con la que vamos a trabajar en el tratamiento de las imágenes utiliza el formato de imagen Bitmap, por lo que es necesaria la conversión previa:

```
Bitmap bmpFrame = ColorImageFrameToBitmap(colorFrame);
```

Como ya se explicó en el apartado 4.3. de este documento, con el fin de reducir la cantidad de información con la que trabajar, reduciéndose enormemente el tiempo de procesamiento, se debe acotar la imagen en tres regiones de interés individuales para la búsqueda de cada marcador:

```
ImageFrame1 = ImageFrame.Clone();
ImageFrame1.ROI = Rectangulo1;
```

Esta aplicación se estará ejecutando simultáneamente con la aplicación de Monitorización de Operaciones. Se representaran las tres regiones de interés con el fin de que el usuario sepa en todo momento donde se está buscando cada marcador:

```
ImagenMarcador1.Source = BitmapSourceConvert.ToBitmapSource(ImageFrame1);
```

Flujo de control de la trama de profundidad

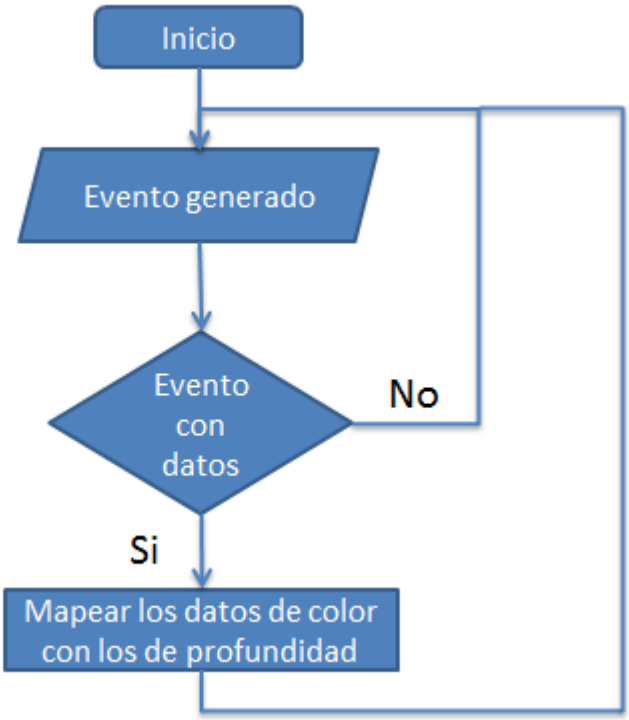


Figura 5.4 Diagrama de Flujo del control de la trama de profundidad

Tras haber comprobado que los datos de profundidad son correctos se pasa a mapear los datos de color con los de profundidad. Como ya se explicó en el diseño

de esta aplicación en el apartado 4.1. de este documento, para localizar el marcador en el espacio tridimensional se debe asignar el pixel de la imagen a color en el marco de profundidad correspondiente. Esto se realizará con el siguiente método:

```
this.sensor.CoordinateMapper.MapColorFrameToDepthFrame(this.sensor.ColorStream.Fo
rmat, this.sensor.DepthStream.Format, depthPixels, mappedColorLocations);
```

Siendo "mappedColorLocations" la variable en la que se almacenan estas localizaciones mapeadas.

5.2.2. Búsqueda de marcadores

Los ciclos de búsqueda de los tres marcadores, marcador1_tick, marcador2_tick y marcador3_tick, se lanzan cada 34 segundos equivalentes a la frecuencia de 30 fps a la que se actualizan los datos del sensor.

Estos hilos son registrados en el momento del inicio de la aplicación:

```
//Marcador 1
marcador1.Tick += new EventHandler(marcador1_Tick);
marcador1.Interval = 34; //34ms -> 30FPS
//Marcador 2
marcador2.Tick += new EventHandler(marcador2_Tick);
marcador2.Interval = 34; //34ms -> 30FPS
//Marcador 3
marcador3.Tick += new EventHandler(marcador3_Tick);
marcador3.Interval = 34; //34ms -> 30FPS
```

Y además deben ser iniciados:

```
marcador1.Enabled = true;
marcador1.Start();
marcador2.Enabled = true;
marcador2.Start();
marcador3.Enabled = true;
marcador3.Start();
```

Los ciclos de los marcadores utilizarán la imagen recortada de la región de interés individual para cada marcador, proporcionada por el evento ColorFrameReady descrito en el apartado anterior y la variable mappedColorLocations proporcionada por el evento DepthFrameReady.

El flujo de control de cada ciclo de búsqueda del marcador es el siguiente:

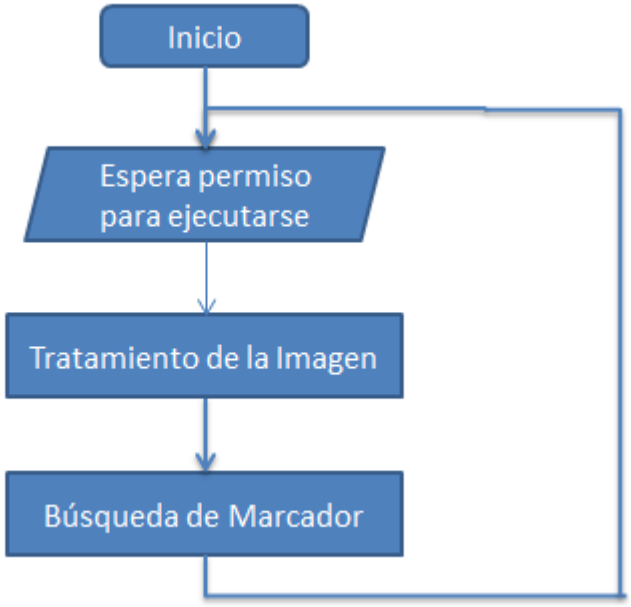


Figura 5.5 Diagrama de Flujo del hilo búsqueda de marcador

Tratamiento de la Imagen

```
Image<Gray, Byte> ImageFrameDetection = cvAndHsvImage(
                ImageFrame1, ValMar[herr_activa, 1, 0], ValMar[herr_activa, 1,
1], ValMar[herr_activa, 1, 2], ValMar[herr_activa, 1, 3]);
```

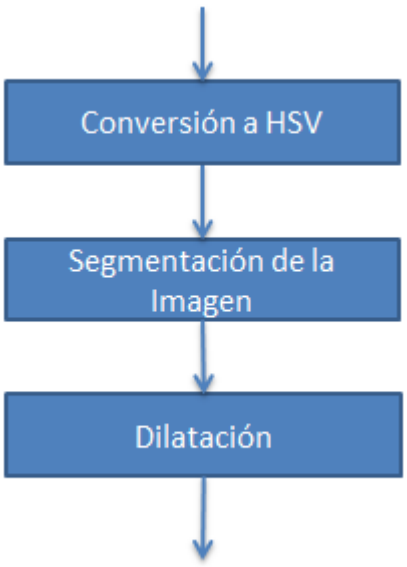


Figura 5.6 Diagrama de Flujo tratamiento de la imagen

El objetivo de este método es convertir la imagen a color en una imagen binarizada según los parámetros de color. Estos parámetros (ValMar) son individuales para cada marcador y son generados mediante la aplicación "Configuración de

Seguimiento" descrita en el apartado 5.4.2 de este documento. Se accede a ellos en el momento del inicio de la aplicación mediante la apertura de un archivo .txt almacenado en el disco:

```
cargarValores();
```

Estos valores establecen el rango máximo y mínimo de los valores matiz (H_{HSV}) y Saturación (S_{HSV}) de la imagen color generada por el evento ColorFrameReady descrito en el apartado anterior. El primer paso es convertir esta imagen del modelo de color RGB (imgFrame) al HSV (hsvImage):

```
Image<Hsv, Byte> hsvImage = imgFrame.Convert<Hsv, Byte>();
```

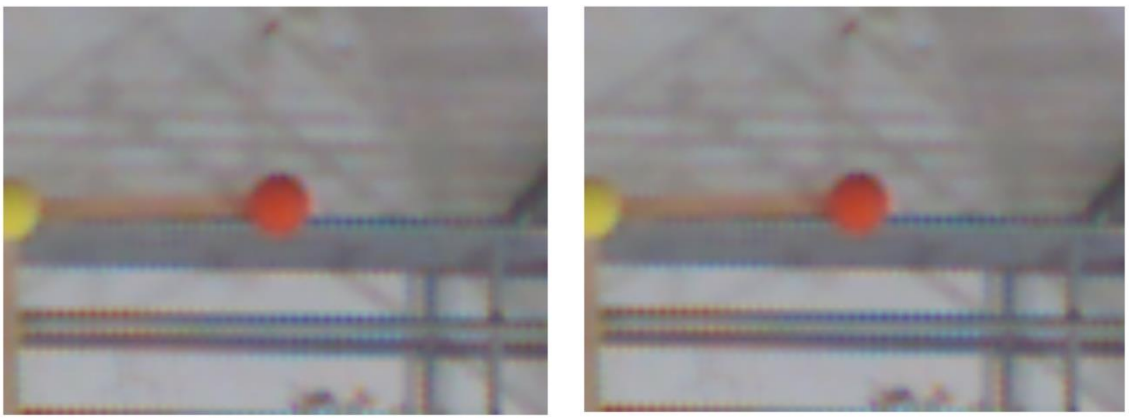


Figura 5.7 Conversión de formato RGB (izquierda) a HSV (derecha)

Los formatos RGB y HSV son representadas por la pantalla del ordenador aparentando ser la misma imagen, pero la información que contiene no es la misma, ya que al cambiar el formato podemos acceder a los valores matiz y saturación necesarios para la segmentación.

El siguiente paso es conseguir dos imágenes binarias, una generada con los rangos superior e inferior del matiz del color y otra con los de saturación. La función CvInRange sirve para comprobar si los píxeles de una imagen caen dentro de un rango especificado particular. Cada píxel de la imagen de entrada (hsvImage[con]) se compara con el valor correspondiente en las imágenes inferiores y superiores. Si el valor en esta es mayor o igual que el valor en la parte baja y también menor que el valor en la parte alta, entonces el valor correspondiente en la imagen destino (ResultImage) se ajustará al máximo valor; de lo contrario, el valor se establece en 0.

```
//Generación de Rango inferior
Image<Gray, Byte> IlowCh = new Image<Gray, Byte>(hsvImage.Width, hsvImage.Height,
new Gray(Lo));
//Generación de Rango superior
Image<Gray, Byte> IHiCh = new Image<Gray, Byte>(hsvImage.Width, hsvImage.Height,
new Gray(Hi));
//Imagen binaria (ResultImage) generada entre los rangos inferior y superior
CvInvoke.cvInRange(hsvImage[con], IlowCh, IHiCh, ResultImage);
```

Figura 5.8 Imagen binaria creada con los rangos superior e inferior de matiz (izquierda) y saturación (derecha)

En la Figura 5.8 vemos como es necesario hacer la restricción de la imagen de matiz y la de saturación. Utilizando solo la restricción de saturación, el marcador naranja que se desea segmentar no sería diferenciado con el marcador amarillo y utilizando solo los valores de matiz la imagen no quedaría totalmente clara.

Sumando estas dos imágenes binarias generadas se consigue obtener la segmentación del color entre los valores superior e inferior de matiz y saturación de la imagen color de entrada.

```
CvInvoke.cvAnd(ResultImageH, ResultImageS, ResultImage, System.IntPtr.Zero);
```

Figura 5.9 Suma de imagen binaria de matiz y saturación

Conseguimos de esta forma una diferencia clara entre el marcador en color blanco en la imagen y el fondo en color negro.

En la siguiente imagen podemos ver como hay veces que este método no consigue tal nitidez del marcador sobre el fondo, siendo necesaria una apertura, es decir, una erosión seguida de una dilatación:

```
//Erosionar la imagen
CvInvoke.cvErode(ResultImage, ResultImage2, (IntPtr)null, 1);
//Dilatar posteriormente
CvInvoke.cvDilate(ResultImage2, ResultImage3, (IntPtr)null, 1);
```

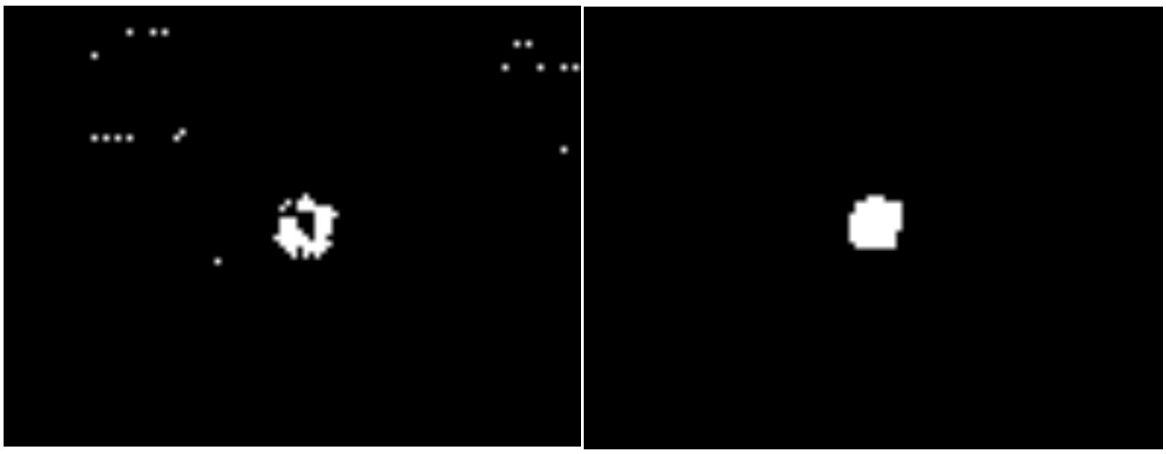


Figura 5.10 Segmentación del marcador morado antes (izquierda) y después (derecha) de la apertura

La apertura se hace con un elemento estructural cuadrado de 2x2 pixeles.

Adquisición de coordenadas de los marcadores

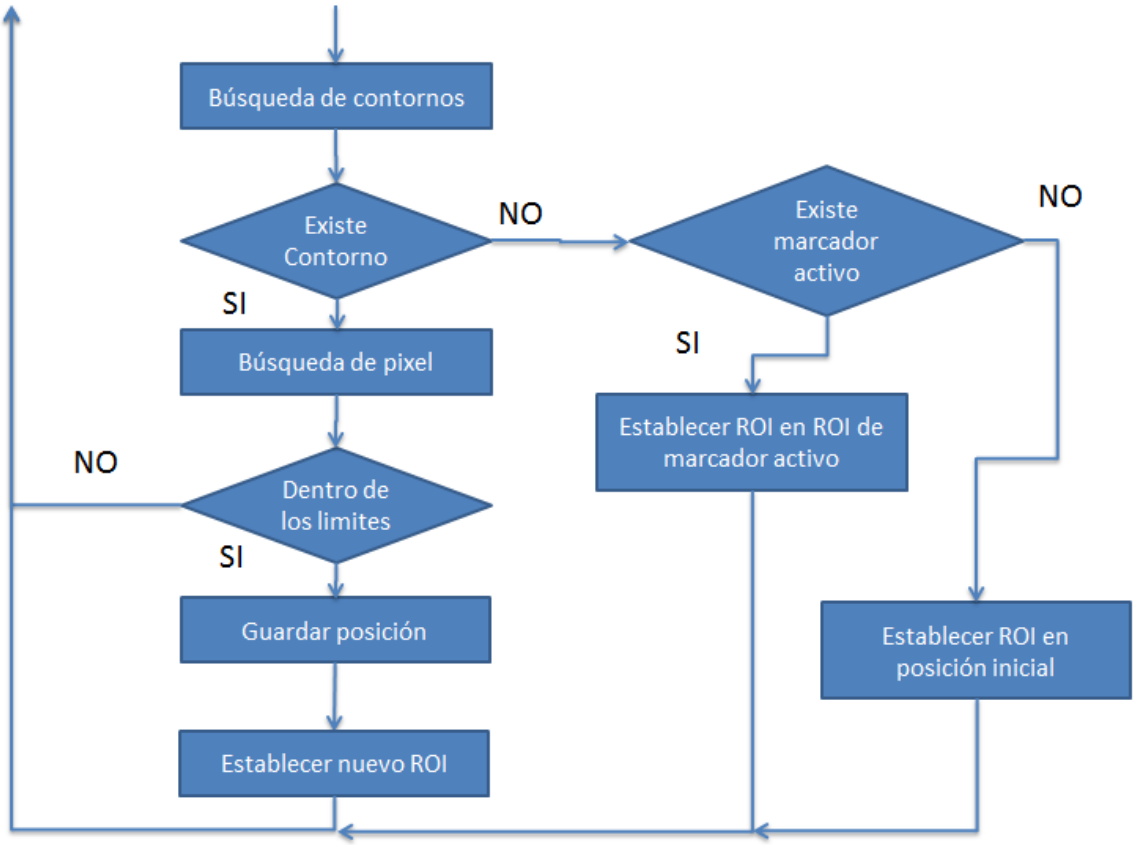


Figura 5.11 Diagrama de Flujo búsqueda de Marcador

Una vez conseguida la imagen binaria con una gran diferenciación entre el marcador y el fondo podemos localizar el pixel en el que se encuentra el centro de gravedad de este marcador. Esto lo realizamos con una búsqueda de contornos (cvFindContours)

```
//Encontrar Contorno
int numContornos = CvInvoke.cvFindContours(
       imgForContour,
       storage,
       ref contour,
       System.Runtime.InteropServices.Marshal.SizeOf(typeof(MCvContour)),
       Emgu.CV.CvEnum.RETR_TYPE.CV_RETR_EXTERNAL,
       Emgu.CV.CvEnum.CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_NONE,
       new System.Drawing.Point(Rectangulo1.X, Rectangulo1.Y));
```

Hay que tener en cuenta el último parámetro pasado a este método. La imagen que utiliza de entrada este método, imgForContour, es la imagen binaria que como vimos es la región de interés de la imagen original. Esto quiere decir que el valor devuelto de la posición del contorno está respecto la imagen recortada por lo que hay que tener en cuenta un desplazamiento correspondiente al valor de la esquina superior izquierda del rectángulo utilizado para crear la región de interés. Es decir:

Valor x obtenido en la imagen = valor x obtenido en ROI + desplazamiento;

Valor x obtenido en la imagen = 20 + 200 = 220;

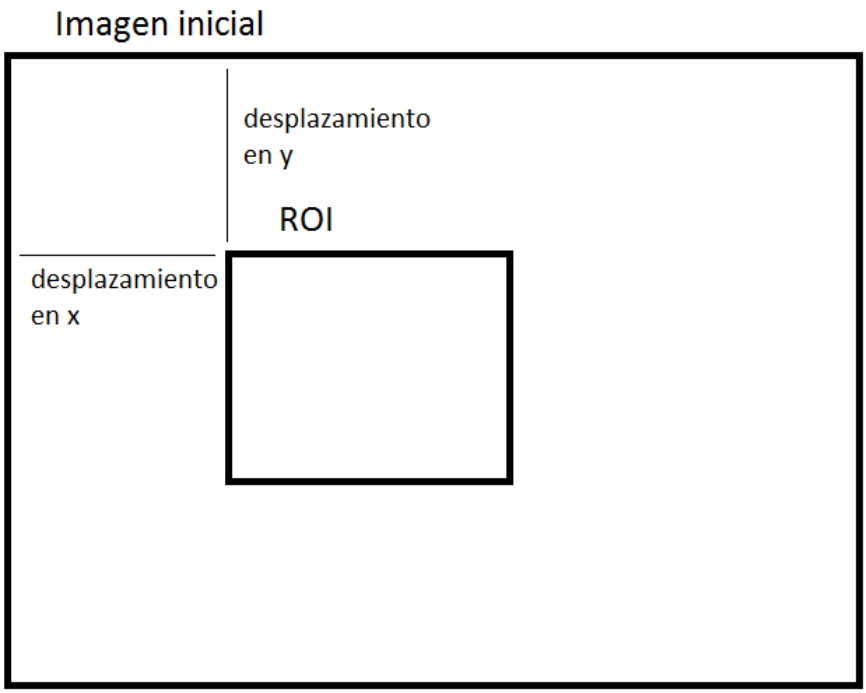


Figura 5.12 Desplazamiento entre el valor obtenido del contorno en la ROI y el de la imagen inicial.

Cuando existen contornos se pasará a calcular el centro de gravedad de este. Para ello se construye un rectángulo que rodea este contorno:

```
System.Drawing.Rectangle bndRec = CvInvoke.cvBoundingRect(contour, 1);
```

Y posteriormente se calcula el centro de ese contorno:

```
int result = ((bndRec.Y + despY + bndRec.Width / 2) * 640) + (bndRec.X + despX +
bndRec.Width / 2);
```

Es necesario sumar un desplazamiento a este valor debido a que los sensores de profundidad y color de la Kinect no están en la misma posición. Estos han sido calculados de forma experimental mediante pruebas con el programa "Configuración de Seguimiento". En el apartado 5.4.2. se hablará de cómo calcularlos.

Como se ha realizado un desplazamiento, puede que el valor calculado pueda estar fuera de los valores de la imagen. Se debe garantizar que no sea así para evitar errores posteriores:

```
if (result < (640 * 480))
{

       ...

       ...}
```

Con la garantía de que el pixel calculado corresponde con el centro de gravedad del marcador, podemos calcular los valores tridimensionales en metros donde se encuentra este marcador:

```
PuntoEnSkeleto                                                                    =
this.sensor.CoordinateMapper.MapDepthPointToSkeletonPoint(this.sensor.DepthStream
.Format, mappedColorLocations[result]);
```

Este método utiliza como entrada la variable "mappedColorLocations" calculada en el hilo DepthFrameReady.

Estos valores se guardaran en una variable global llamada "posición" y serán comunicadas al servidor por el hilo enviar.

Para conseguir que la región de interés recortada para la búsqueda del marcador cambie dinámicamente el centro de la ROI siempre coincidirá con el centro de gravedad del marcador. Es necesario comprobar que la posición del ROI calculada no se sale de los límites de la imagen:

```
//Establecer el nuevo xROI
if (((bndRec.X + bndRec.Width / 2) - Rectangulo1.Width / 2) >= 0) Rectangulo1.X =
(bndRec.X + bndRec.Width / 2) - Rectangulo1.Width / 2;
else Rectangulo1.X = 0;
//Establecer el nuevo yROI
```

```
if (((bndRec.Y + bndRec.Height / 2) - Rectangulo1.Height / 2) >= 0) Rectangulo1.Y
= (bndRec.Y + bndRec.Height / 2) - Rectangulo1.Height / 2;
else Rectangulo1.Y = 0;
//Si no está reducida la región al mínimo reducirla (focalizar la region)
if (Rectangulo1.Width > minwROI)
{
      Rectangulo1.Width = minwROI;
      Rectangulo1.Height = minhROI;
}
```

Además se indicará que este valor es correcto:

```
//el valor encontrado es correcto
else
{
     coorok1 = 1;
     correcto1.Fill = new System.Windows.Media.SolidColorBrush(ColorCorrecto);
}
```

Si un ciclo de búsqueda no encuentra el marcador buscará este en la región de interés de otro ciclo que si ha encontrado su marcador. Los marcadores van a estar cercanos entre sí por lo que realizando esta suposición reducimos enormemente el tiempo que tarda un ciclo de búsqueda en encontrar su marcador.

```
//LLevar ROI a la de un marcador activo cuando no se encuentra ningun contorno
(buscar el marcador)
if (seq != null && seq.Ptr.ToInt64() == 0)
{
       correcto1.Fill                                =                          new
       System.Windows.Media.SolidColorBrush(ColorIncorrecto);
       coorok1 = 0;
       //si el las coordenadas del marcador 2 son correctas llevar el marcador a
       su ROI pero ampliado
       if (coorok2 == 1)
       {
              Rectangulo1 = Rectangulo2;
              if ((Rectangulo1.X - 50) >= 0) Rectangulo1.X = Rectangulo1.X - 50;
              if ((Rectangulo1.Y - 50) >= 0) Rectangulo1.Y = Rectangulo1.Y - 50;
              Rectangulo1.Width = Rectangulo1.Width + 100;
              Rectangulo1.Height = Rectangulo1.Height + 100;
       }
       //si el las coordenadas del marcador 3 son correctas llevar el marcador a
       su ROI ampliado
       else if (coorok3 == 1)
       {
              Rectangulo1 = Rectangulo3;
              if ((Rectangulo1.X - 50) >= 0) Rectangulo1.X = Rectangulo1.X - 50;
              if ((Rectangulo1.Y - 50) >= 0) Rectangulo1.Y = Rectangulo1.Y - 50;
              Rectangulo1.Width = Rectangulo1.Width + 100;
              Rectangulo1.Height = Rectangulo1.Height + 100;
       }
```

En el caso de no existir ningún marcador localizado, se establece el ROI en la posición central de la zona de trabajo, a la espera de que el usuario pase la herramienta por esa zona.

```
       Else Rectangulo1 = ROIperdido;
```

5.2.3. Comunicación con Monitorización de Operaciones

La aplicación seguimiento de Marcadores necesita una comunicación bidireccional con el programa Monitorización de Operaciones. Como ya se ha explicado en el apartado 5.1. Funciona como cliente mientras que el programa de Monitorización trabaja como servidor.

Los datos entre cliente y servidor son de tres tipos:

1) Coordenadas de los marcadores. Continuamente se actualizan las coordenadas obtenidas de los marcadores y se envían a la aplicación servidor. El string enviado será de la forma:

```
PosMarcador1x;PosMarcador1y;PosMarcador1z;
PosMarcador2x;PosMarcador2y;PosMarcador2z;
PosMarcador3x;PosMarcador3y;PosMarcador3z;
```

2) Ruta. Se envía al servidor la ruta de la actividad con la que se esté trabajando. El formato será: Ruta;
3) Repeticiones. Número de veces que se ha de repetir una operación. El formato será: N;Número de Repeticiones;
4) Cadena de Actividades. Se envían varias actividades seguidas para que se ejecuten una tras otra:
   J; Número de Repeticiones;Ruta; Número de Repeticiones;Ruta;

Tipo	Significado	Comando	Ejemplo
1	Coordenadas de los marcadores	No tiene comando a interpretar	-0,01246707;0,08311375;1,187; 0,09111706;0,06626692;1,183; -0,02673887;-0,02468203;1,175;
2	Ruta Carpeta Actividad1	La “C“es interpretada como comando por el servidor	C:\MonitorOperaciones\Archivos\Actividades\Actividad1;
3	Realizar 3 Repeticiones	“N”	N;3;
4	Realizar 2 veces la Actividad 1 y 3 veces la Actividad 2	“J”	J;2; C:\MonitorOperaciones\Archivos\Actividades\Actividad1;3; C:\MonitorOperaciones\Archivos\Actividades\Actividad2;

Tabla 5.1 Ejemplos de cada tipo de mensaje en aplicación Seguimiento de Marcadores

Los datos de servidor a cliente son de tres tipos:

1) Cerrar aplicación, “S”.

   Es decisión del servidor cerrar la aplicación.

2) Listo Para recibir, “L”.

   Como el programa Monitorización de Operaciones solamente puede recibir datos si tiene el foco del programa, es necesario que envíe una señal de listo para recibir datos al cliente. En este momento el cliente envía el comando.

3) Herramienta Utilizada, “1”, “2” o ”3”.

   Es la herramienta que actualmente está en uso.

La lectura de estos comandos es continua gracias a trabajar con un hilo asíncrono, por lo que en cuanto llega el comando la aplicación de seguimiento actúa en consecuencia. Para los datos tipo 1 se cierra la aplicación, para el dos se envía el comando escrito y para 3 se cambia la herramienta actualmente activa.

5.2.4. Interfaz de Usuario

Esta aplicación será utilizada en los modos 2 y 4 correspondientes a la creación de una nueva actividad y al guiado de la actividad. En este apartado se van a explicar los controles disponibles y los gráficos facilitados por esta aplicación en la realización de una actividad.

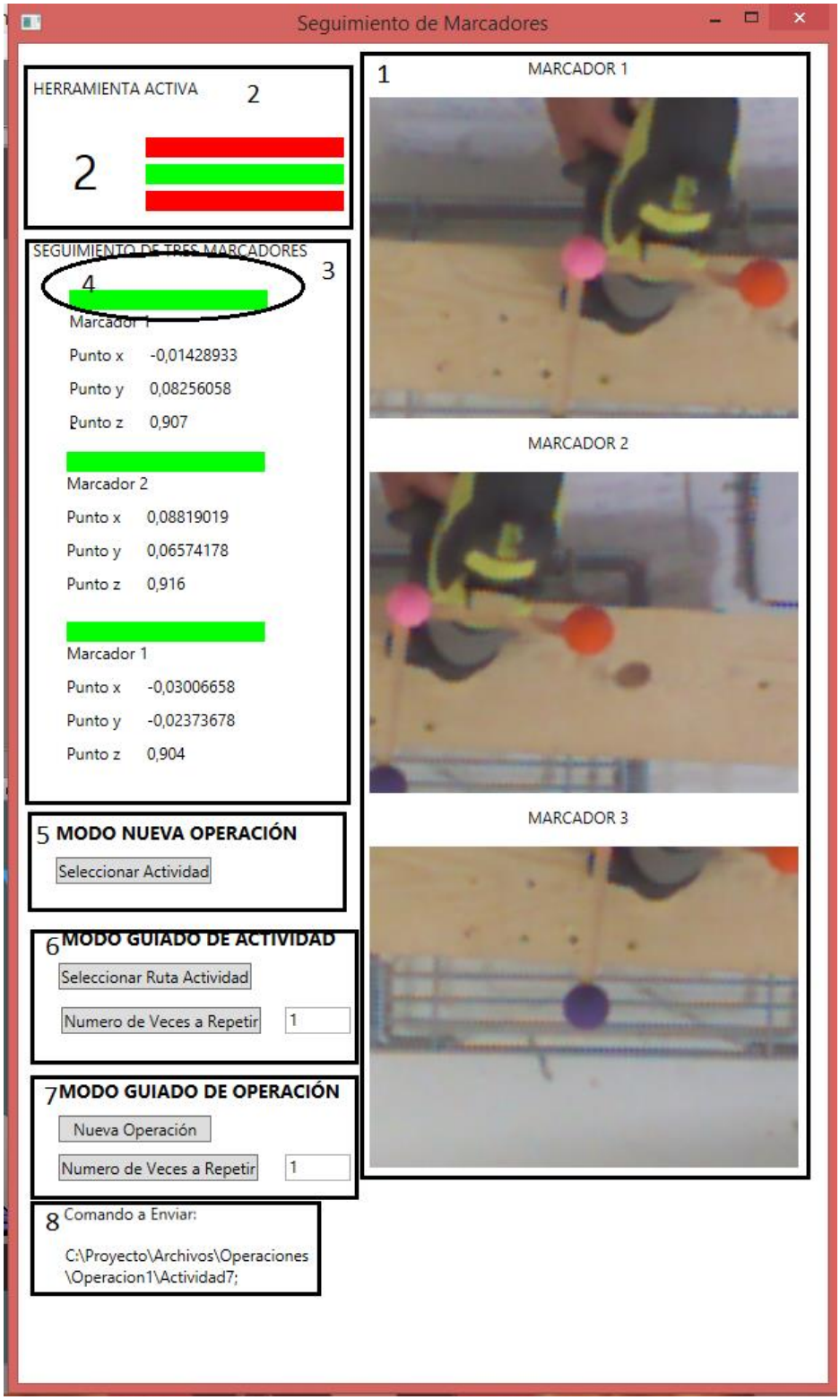


Figura 5.13 Aplicación Seguimiento de Marcadores durante una actividad de taladrado

En la Figura 5.13 podemos diferenciar 8 zonas:

1) Imagen de cada Marcador

Durante la realización de una actividad puede ocurrir que un marcador se haya tapado con algún otro objeto, o por el propio usuario. Para que el cálculo del efector final de la herramienta sea correcto es necesario que las tres coordenadas de los marcadores sean correctas. Si no es así el usuario vería como el seguimiento representado en el programa "Monitorización de Operaciones" no sería estable, y no coincidiría con los movimientos realizados por la herramienta. Se representan por pantalla los recortes de estos marcadores para facilitar al usuario la tarea de detectar que elemento está bloqueando cada marcador.

2) Herramienta activa

Cuando el programa "Monitorización de Operaciones" carga las propiedades de una actividad envía al usuario el identificador de la herramienta que se debe utilizar para la actividad a realizar. El identificador indica a la aplicación que marcadores debe buscar.

Este identificador se representa con un número y con un rectángulo que cambia de color a verde para la herramienta activa.

3) Seguimiento de tres Marcadores

El objetivo principal de esta aplicación es hallar las coordenadas correspondientes a cada marcador y enviárselas al programa "Monitorización de Operaciones". Estas coordenadas son representadas en esta zona. Están dadas en metros.

4) Identificador Marcador Encontrado

Cuando un marcador se ha perdido el rectángulo situado sobre las coordenadas de cada marcador cambia a color rojo. Si las coordenadas son correctas se mantiene de color verde.

5) Modo Nueva Actividad

Pulsando en el botón "Seleccionar Actividad" se abre un cuadro de diálogo donde se debe seleccionar el archivo de texto que contiene las propiedades de la nueva actividad a crear.

6) Modo Guiado de Actividad

En este modo el usuario tiene dos opciones pulsar el botón "Seleccionar Ruta de Actividad" enviando al servidor la ruta en la que se encuentra la actividad a almacenar, o pulsar el botón "Número de Veces a Repetir" para comunicar al servidor las veces que se debe ejecutar la actividad previamente enviada.

7) Modo Guiado de Operación

En este caso se quieren seleccionar varias actividades que deben ejecutarse una detrás de otra. Para generar una nueva serie de actividades ha de pulsarse el botón "Nueva Operación". Para agregar una actividad a la serie se debe escribir el número de veces que se va a repetir esta actividad y seguidamente pulsar el botón "Número de Veces a Repetir" para seleccionar su ruta.

8) Comando a Enviar

Como ya se ha dicho en el apartado anterior para realizar una comunicación correcta entre el servidor (el programa de Monitorización) y el cliente (las aplicaciones), es necesario que el servidor de permiso para recibir un nuevo comando. En Bloque de texto "Comando a Enviar:" se escribe el comando a modo de comprobación antes del envío.

5.3. Monitorización de Operaciones

Esta aplicación será la principal del sistema desarrollado. Permite la interacción directa con el usuario y representar las actividades a realizar. Se encargará de calcular el movimiento del efector final de la herramienta y de generar las superficies oportunas en cada modo. Además permite una navegación por su interfaz a modo de videojuego en primera persona.

Esta aplicación se ha desarrollado con la herramienta Unity. En el ámbito de programación facilitado por Unity se trabajan con objetos a los cuales se les asocia un "script", un pequeño programa que define su comportamiento. Estos scripts interactúan entre sí para lograr una ejecución fluida y dinámica de los elementos.

Se ha creado un entorno de trabajo virtual que simula el entorno de trabajo ideal descrito en el apartado 4.5 de este documento. Este tiene los mismos elementos principales:

1) Iluminación Artificial
2) Zona de trabajo
3) Sensor Kinect
4) Marcadores
5) Teclado y Ratón
6) Zona de Herramientas
7) Monitor
8) Ejes cartesianos

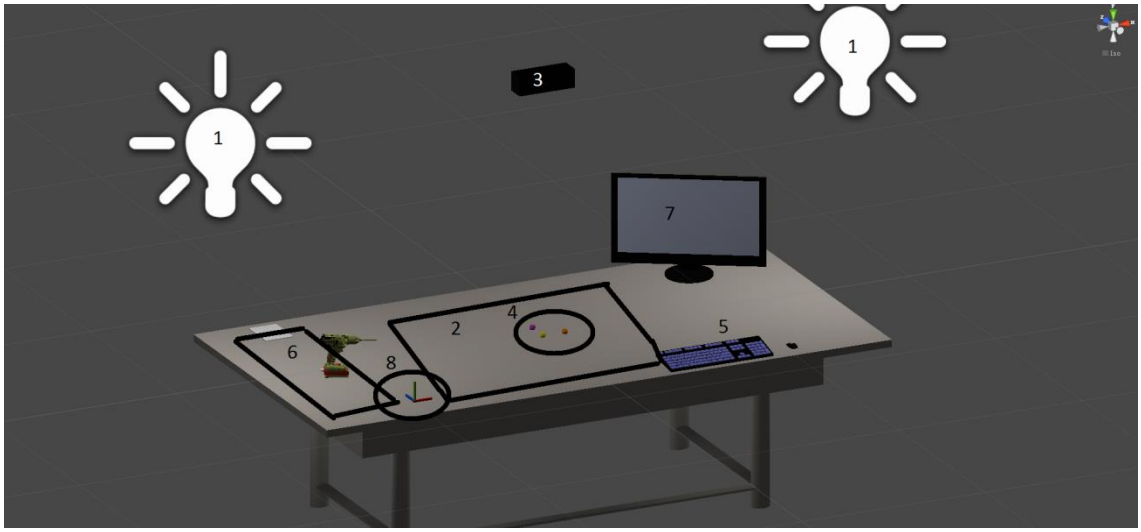


Figura 5.14 Entorno de Trabajo virtual

Se ha añadido un elemento, los ejes cartesianos. Es importante que el usuario tenga en cuenta estos ejes, ya que a la hora de configurar las actividades es necesario facilitar al programa la distancia entre el marcador principal y el efector final de la herramienta. El rectángulo verde corresponde al eje y, el rojo al x y el azul al eje z.

Varios de estos elementos son decorativos, pero otros tienen asociados un script que definen su comportamiento. En el siguiente diagrama se representan la relación entre los objetos con los scripts. La flecha indica la relación entre los objetos y los scripts. La memoria compartida es utilizada por el resto de scripts para compartir variables entre sí.

En los siguientes apartados se explicará el comportamiento de los scripts.

5.3.1. Interfaz Gráfica de Usuario

La cámara es un dispositivo a través del cual el jugador ve el entorno de trabajo virtual desarrollado. La interfaz gráfica de Usuario es un script asociado a esta cámara. Es el programa servidor encargado de recibir las ordenes de los programas clientes y de interactuar con el usuario. Está dividido en dos partes:

1) Controlador de eventos. Es la parte encargada de representar por pantalla los menús de interacción con el usuario, y de recoger las órdenes recibidas por teclado y por los programas clientes. Para llevar a cabo estas tareas se utiliza el método “OnGUI” de Unity el cual es llamado varias veces por frame para la prestación y gestión de eventos. En este caso los eventos corresponden a las órdenes por teclado y de los comandos de los programas cliente.

2) Control de Ejecución. Actúa en consecuencia de estos eventos. Es la parte que gestiona el funcionamiento del programa, actuando de diferente manera dependiendo del modo en el que se encuentre. Se utiliza el método "Update" llamado cada fotograma para implementar el comportamiento del programa.

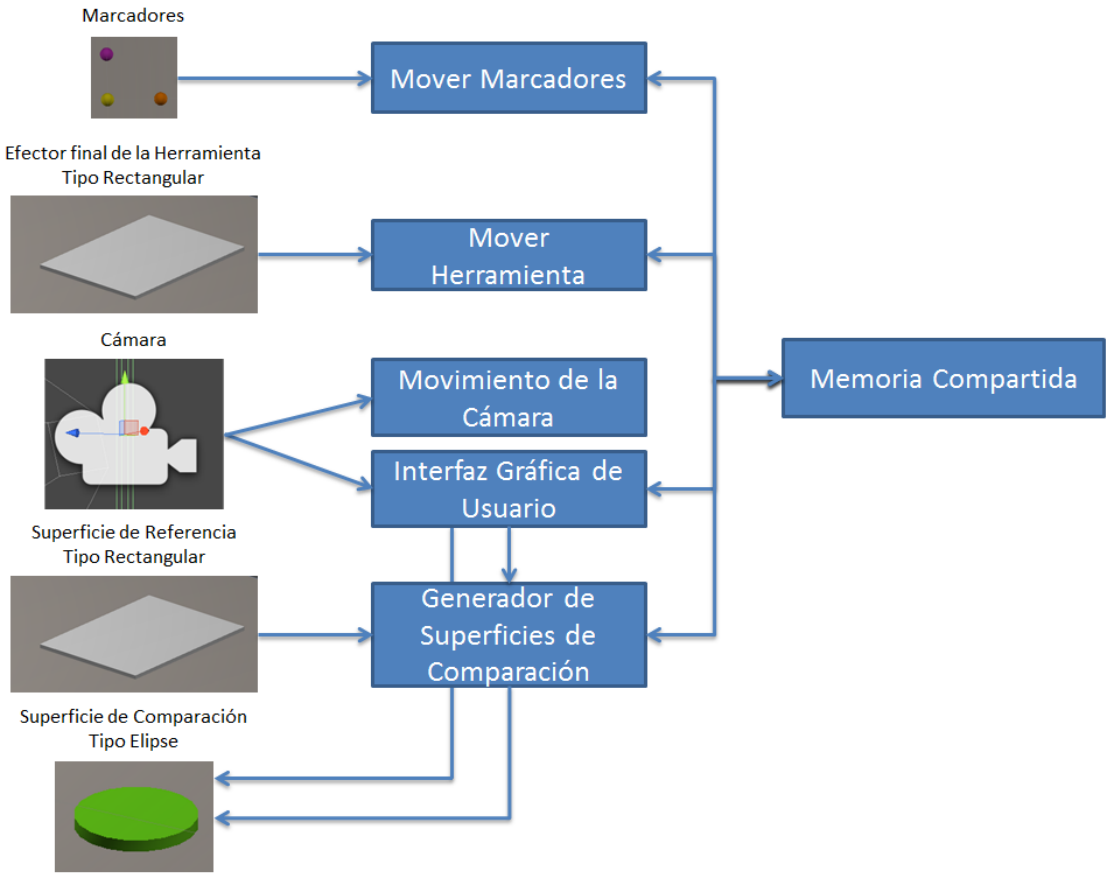


Figura 5.15 Diagrama que relaciona los scripts con los objetos

Con el fin de lograr todas las tareas necesarias para la buena iteración con el usuario de la aplicación, se van a definir varios modos de funcionamiento. El usuario puede alternar entre los modos de funcionamiento por medio de las teclas función del teclado. El modo 1 coincide con la función 1, el modo 2 con la función 2 y así con los 5 modos que completan la aplicación.

Figura 5.16 Menú inicial del programa Monitorización de Operaciones

Estos modos son:

1) Modo 1. Configuración del Sistema

El único cometido que tiene este modo es el de ejecutar las aplicaciones "Configuración de Operaciones" y "Configuración de Seguimiento".

2) Modo 2. Nueva Operación.

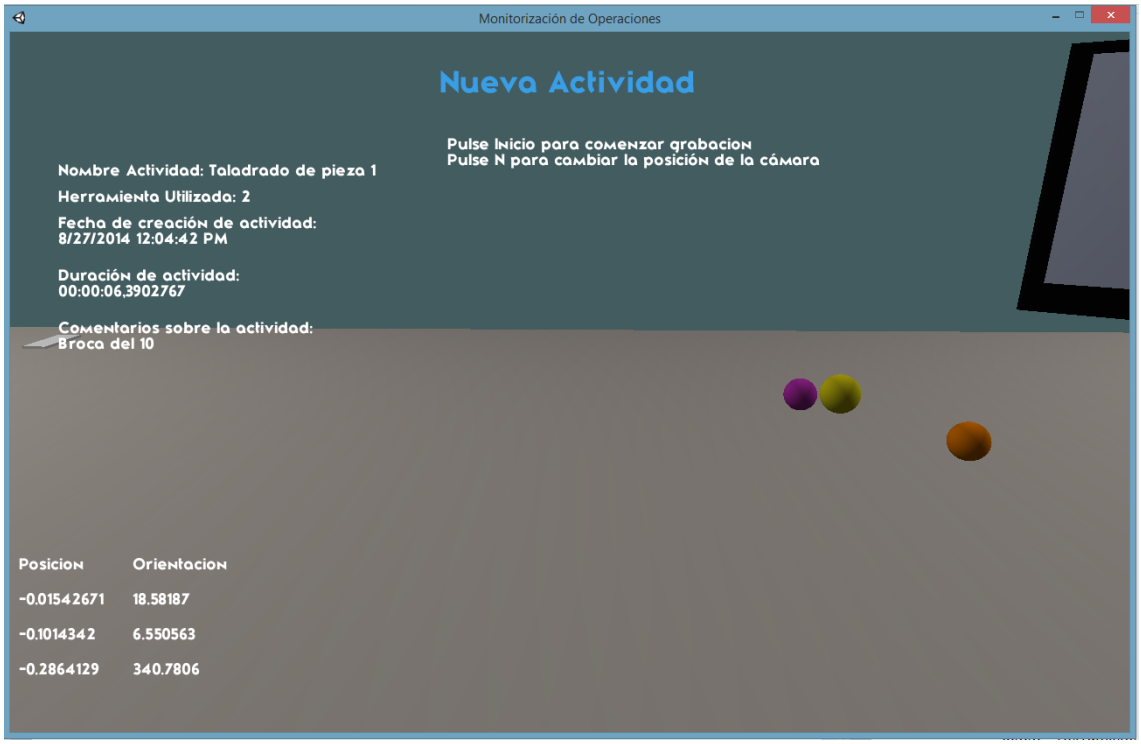


Figura 5.17 Interfaz Gráfica Modo Nueva Operación

Flujo de Control del Controlador de Eventos

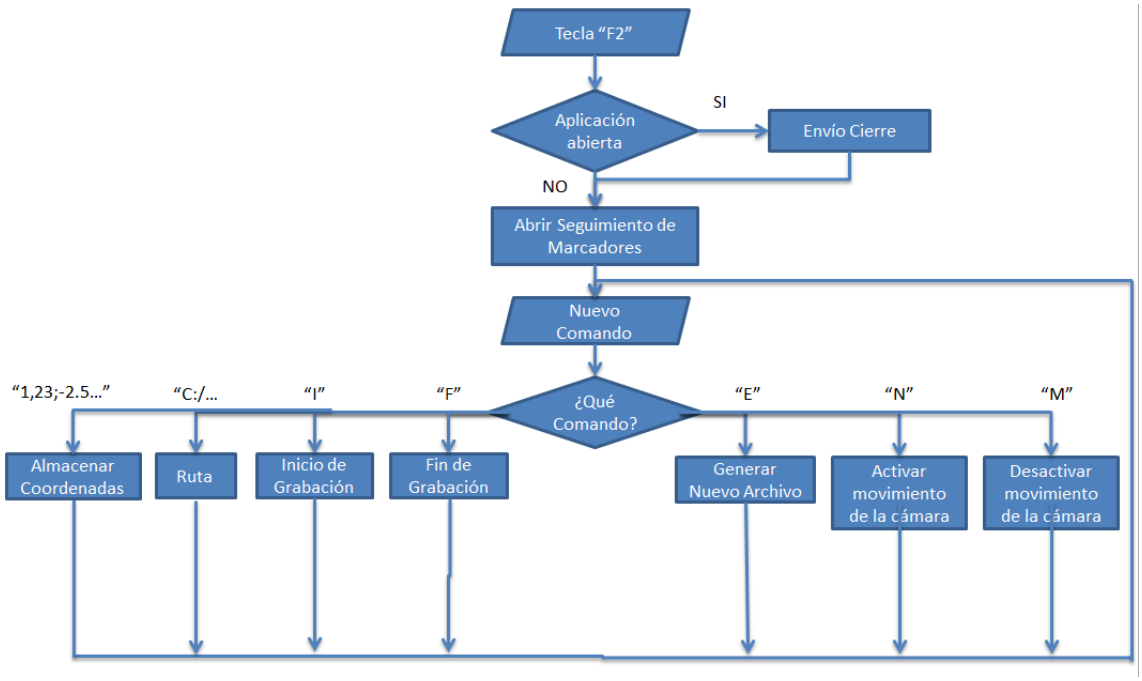


Figura 5.18 Diagrama de Flujo del Controlador de eventos en Modo 2

Cuando se pulsa la función “F2” se ejecuta la aplicación de “Seguimiento de Marcadores”. Previamente es necesario comprobar si existe alguna conexión activa para evitar problemas con los sockets. Si existe conexión quiere decir que ya hay una aplicación que utiliza conexión abierta, siendo necesario cerrarla. A partir de aquí está a la espera de un nuevo comando. Si es una coordenada la almacena en memoria compartida para ser posteriormente usada por los scripts encargados del movimiento de los marcadores. Del comportamiento del resto de comandos se encarga el control de ejecución.

Flujo de Control del Control de Ejecución

Para cada comando recibido se va desencaminar una acción:

a. Ruta, ‘C’

Descripción del comando: este comando es enviado por la aplicación Seguimiento de Marcadores. Contiene la ruta en la que se deben almacenar las superficies de la nueva actividad.

Acción desencadenada: se abre el archivo y se cargan las propiedades de la actividad.

b. Inicio de Grabación, ‘I’.

Descripción del comando: este comando es creado al pulsar la tecla “inicio” del teclado. Indica que se debe crear una superficie de referencia.

Acción desencadenada: Para crear una superficie de referencia se debe crear un nuevo objeto. Este tendrá a forma del efector proporcionada por las propiedades de la actividad. Su orientación posición y escala serán las que facilita el script “movimiento de la herramienta”. Su nombre será SR_ seguido del número de superficie. Además se almacenan los valores de posición y orientación en una cadena de caracteres para posteriormente crear la superficie de referencia.

c. Fin de Grabación, ‘F’.

Descripción del comando: este comando es creado al pulsar la tecla “fin” del teclado. Indica que se debe dejar de crear superficies de referencia.

Acción desencadenada: Cambia el comando a ‘F’ dejando de permitir la creación de superficies de referencia.

d. Activar movimiento de la cámara, ‘N’.

Descripción del comando: este comando es creado al pulsar la tecla “N” del teclado. Indica que se permite el movimiento de la cámara.

Acción desencadenada: se asocia el script “MouseLook” y “mover Cámara” a la cámara.

e. Desactivar movimiento de la cámara, ‘M’.

Descripción del comando: este comando es creado al pulsar la tecla “M” del teclado. Indica que se desactiva el movimiento de la cámara.

Acción desencadenada: se desvincula el script “MouseLook” y “mover Cámara” a la cámara.

f. Generar Nuevo Archivo, ‘E’.

Descripción del comando: este comando es creado al pulsar la tecla “Enter” (la de los números) del teclado. Con ello el usuario indica que la actividad ha sido creada.

Acción desencadenada: Se sobrescribe el archivo de la actividad de referencia. Se modifican las propiedades con la nueva posición de la cámara, la fecha y la duración de la actividad. Se almacenan de forma secuencial las coordenadas de la posición y orientación de todas las superficies de referencia.

3) Modo 3. Modificación de Actividades.

El controlador de eventos del modo 3 es similar al del modo 2. En este caso se abre la aplicación Cargar Actividades. Ahora no se reciben coordenadas por la aplicación cliente.

Flujo de Control del Control de Ejecución

Para cada comando recibido se va desencaminar una acción:

a. Ruta, ‘C’

Descripción del comando: este comando es enviado por la aplicación Cargar Actividades. Contiene la ruta en la que se encuentran la actividad a cargar.

Acción desencadenada: se abre el archivo y se cargan las propiedades de la actividad y las actividades de Referencia. Previamente se borran las superficies de referencia que estén representadas en el programa.

b. Cambiar a color rojo, ‘R’.

Figura 5.19 Interfaz Gráfica Modo Modificación Actividades

Flujo de Control del Controlador de Eventos

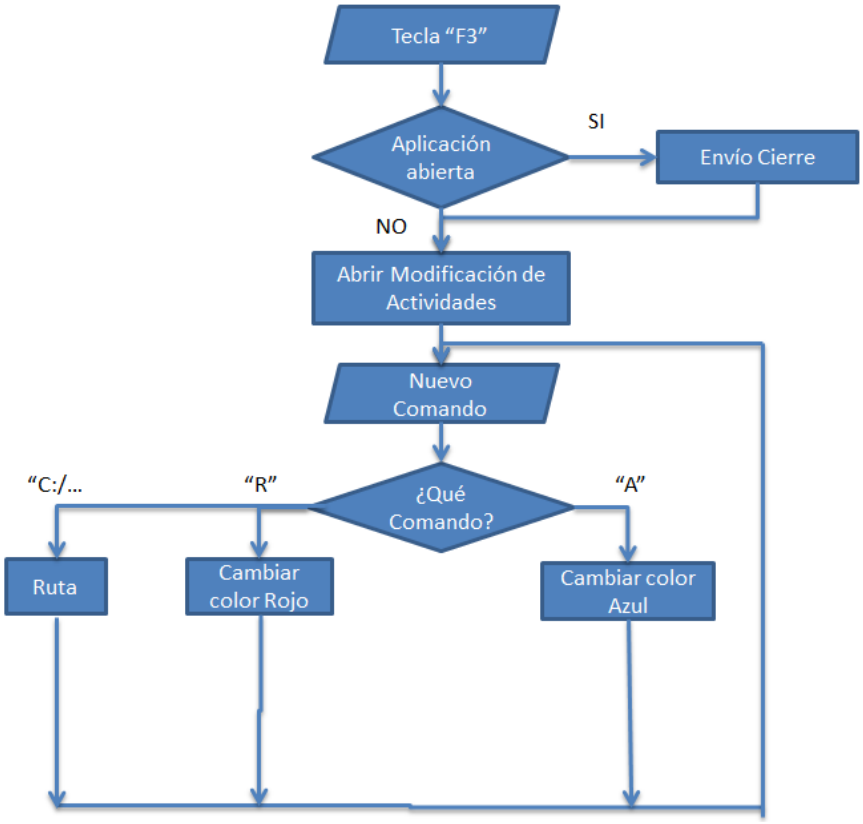


Figura 5.20 Diagrama de Flujo del Controlador de eventos en Modo3

Descripción del comando: este comando es enviado por la aplicación Cargar Actividades. Indica que las actividades marcadas van a ser eliminadas.

Acción desencadenada: El comando recibido tiene el siguiente formato: R;1;2;3;4;...; donde los números indican las superficies a ser coloreadas de rojo.

c. Cambiar a color azul, 'A'.

Descripción del comando: este comando es enviado por la aplicación Cargar Actividades. Indica una comprobación de las actividades.

Acción desencadenada: El comando recibido tiene el siguiente formato: A;1;2;3;4;...; donde los números indican las superficies a ser coloreadas de azul.

4) Modo 4. Guiado de Actividades.

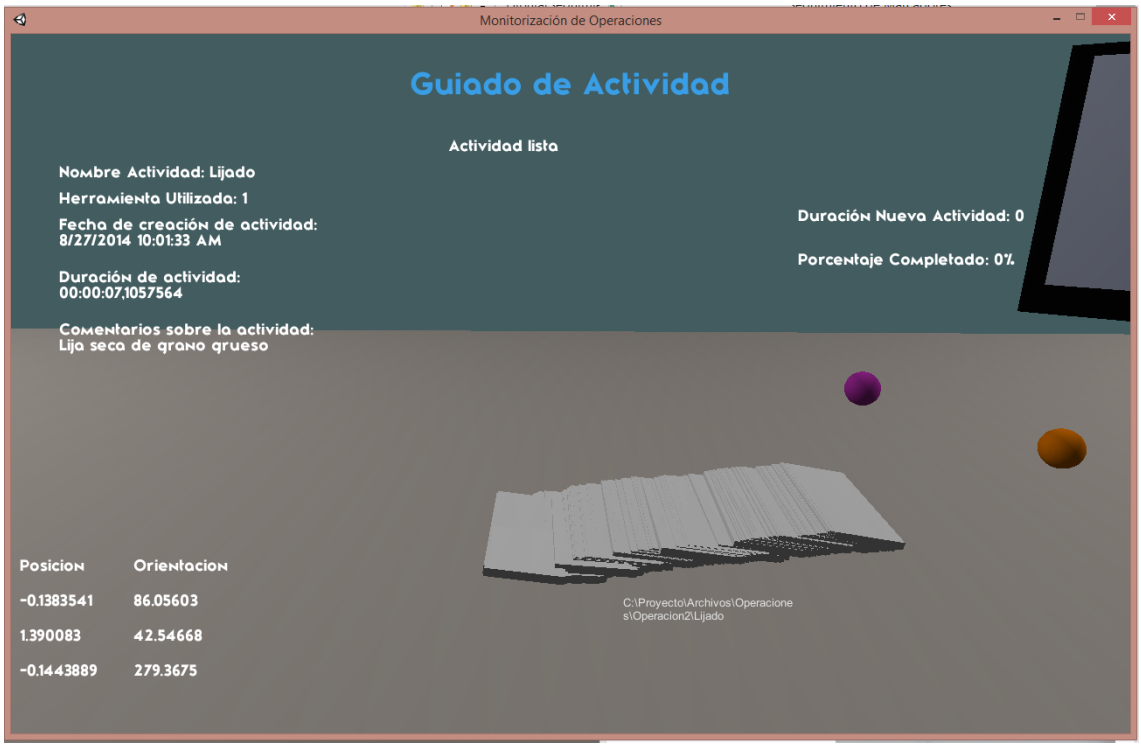


Figura 5.21 Interfaz Gráfica Modo Guiado de Actividades

Flujo de Control del Controlador de Eventos

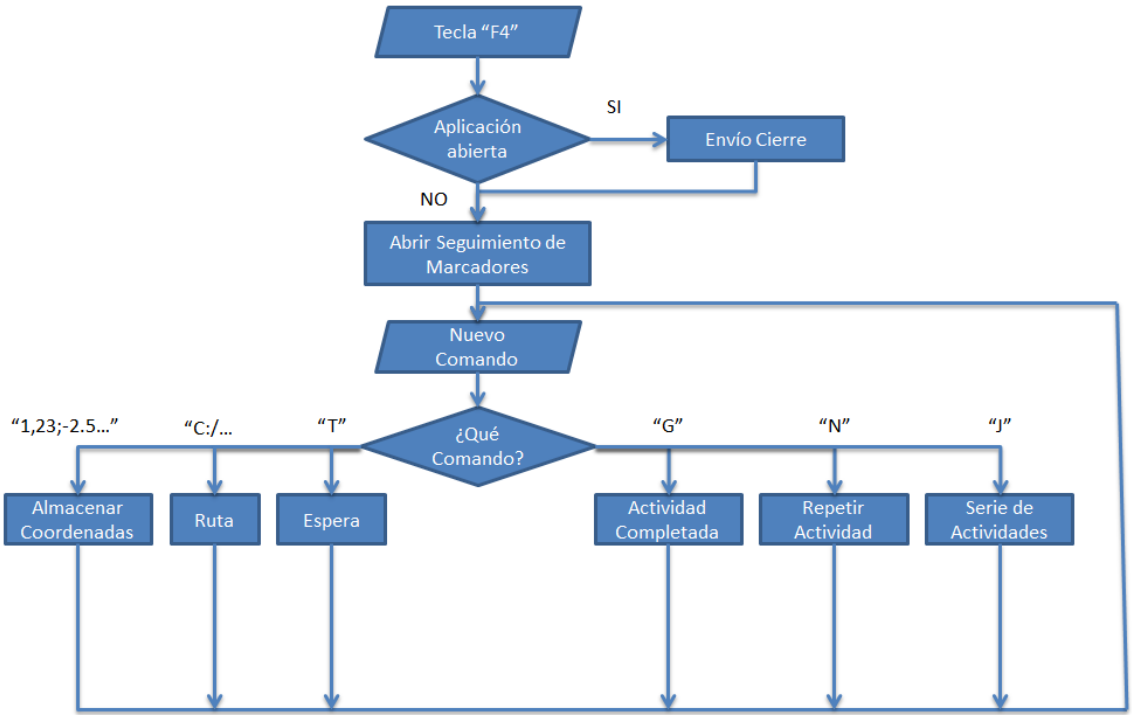


Figura 5.22 Diagrama de Flujo del Controlador de eventos en Modo 4

El flujo de control de eventos es similar al del modo 2. Además de cambiar los comandos recibidos se diferencia en que además es capaz de generar el comando 'G' Actividad completada. Esto ocurre cuando la actividad se ha completado al 100% o cuando se ha completado más del 75% de la actividad y se ha sobrepasado el tiempo en el que se debe realizar la actividad.

Flujo de Control del Control de Ejecución

a. Ruta, 'C'

Descripción del comando: este comando es enviado por la aplicación Seguimiento de Marcadores. Contiene la ruta de la actividad en la que se encuentran las superficies de referencia y donde se deben almacenar las nuevas superficies de comparación.

Acción desencadenada: se busca la ruta de la actividad de referencia y se cargan las propiedades de la actividad y las superficies. Se asocia el script "Generador de Superficies de Comparación" a cada superficie de referencia generada. Previamente se borran las superficies de referencia que estén representadas en el programa. Escribe en pantalla el mensaje "Actividad Lista" indicando al usuario que la actividad está lista para comenzar a almacenar las superficies de comparación.

b. Repetir Actividad, ‘N’

Descripción del comando: este comando es enviado por la aplicación Seguimiento de Marcadores. Indica las veces que se debe de repetir una actividad. El formato es de la forma: N;número_repeticiones;

Acción desencadenada: se reinicia el contador de actividades y escribe la ruta anterior como comando para ser interpretada como anteriormente se ha dicho.

c. Serie de Actividades, ‘J’

Descripción del comando: este comando es enviado por la aplicación Seguimiento de Marcadores. Indica que se deben de ejecutar una serie de actividades seguidas. El formato es de la forma: J;número_repeticiones_actividad1;ruta_actividad1;número_repeticiones _actividad2;ruta_actividad2;...;

Acción desencadenada: se almacenan las rutas y el número de repeticiones de las actividades. Reinicia la cuenta de actividades a realizar y genera la ruta de la primera actividad programada.

d. Actividad completada, ‘G’

Descripción del comando: este comando es provocado por el Controlador de Eventos. Indica que la actividad a realizar ha sido completada.

Acción desencadenada: se genera el archivo ActividadSC.txt que contiene la nueva actividad de referencia. Estarán formado por las propiedades más las superficies de comparación. Las propiedades de la actividad de comparación serán su fecha de creación, la duración de la actividad y porcentaje de superficies completado antes del tiempo estipulado por la actividad de referencia.
Si no se han completado el número de repeticiones de la actividad se genera el comando ‘T’. En el caso de haberse completado las repeticiones se comprueba si hay más actividades en la cola. En el caso de que no queden actividades se pide al usuario que reinicie la actividad o serie de actividades a ejecutar.

e. Espera, ‘T’

Descripción del comando: este comando es provocado por el comando ‘G’. Indica que se debe repetir la actividad.

Acción desencadenada: espera 5 segundos antes de volver a cargar la actividad.

5) Modo 5. Comprobación de Actividades.

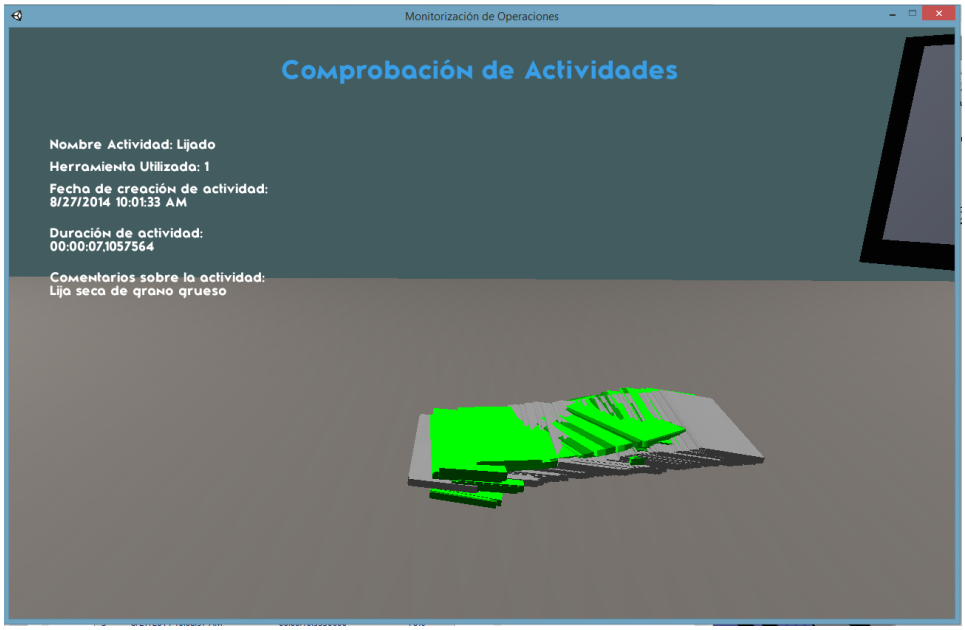


Figura 5.23 Interfaz Gráfica Modo Comprobación de Actividades

Flujo de Control del Controlador de Eventos

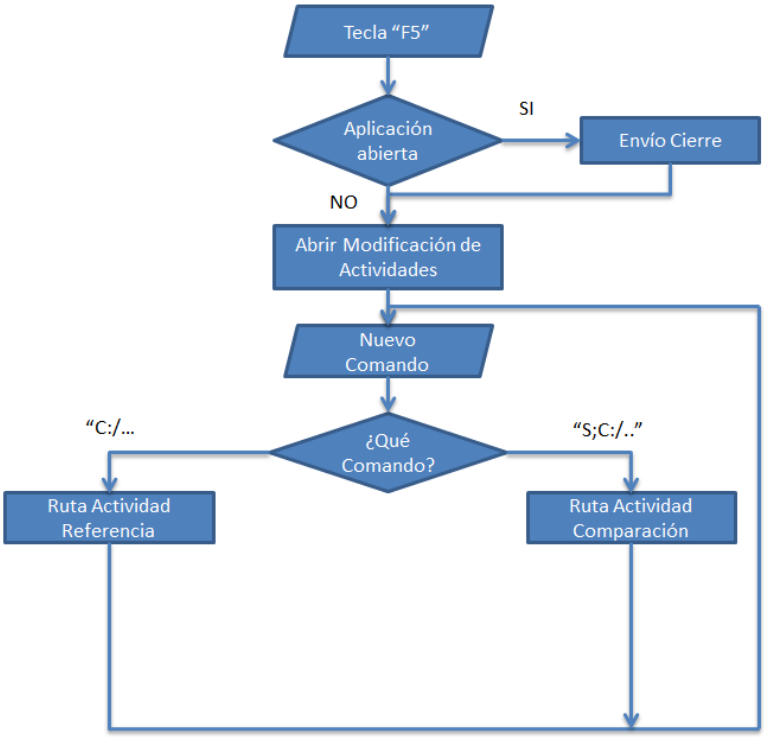


Figura 5.24 Diagrama de Flujo del Controlador de eventos en Modo 5

El controlador de eventos sigue el mismo patrón que los modos anteriores. No recibe coordenadas de posición de los marcadores y abre la aplicación Cargar Actividades.

Flujo de Control del Control de Ejecución

f. Ruta Actividad Referencia, 'C'

Descripción del comando: este comando es enviado por la aplicación Cargar Actividades. Contiene la ruta de la actividad de referencia

Acción desencadenada: se cargan las propiedades de la actividad y las superficies de Referencia. Previamente se borran todas las superficies existentes.

g. Ruta Actividad Comparación, 'S'

Descripción del comando: este comando es enviado por la aplicación Cargar Actividades. Contiene la ruta de una actividad de comparación.

Acción desencadenada: se cargan las superficies de comparación contenidas en el archivo ActividadSC.txt. Previamente se borran las superficies de comparación existentes. Estas superficies se representan en pantalla de color verde para diferenciarse de las superficies de referencia de color gris.

5.3.2. Movimiento de Marcadores

Los marcadores seguidos por la aplicación Seguimiento de Marcadores son simulados como objetos en el entorno de trabajo virtual desarrollado. Tienen una forma esférica y su tamaño está ligeramente ampliado para facilitar su visualización.

Cada marcador tiene asociado el script Movimiento de Marcadores. Utiliza los datos facilitados por la aplicación Seguimiento de Marcadores almacenados en memoria compartida por el script Interfaz Gráfica de Usuario. El funcionamiento es el mismo para cada marcador:

1) Cálculo la posición inicial como la posición del marcador.
2) Actualización de la posición final, la facilitada por la aplicación Seguimiento de Marcadores.
3) Traslación desde la posición inicial a la final a razón de 0.5 m/s.

Como se verá en el apartado 6.1 ensayos de calibración la aplicación Seguimiento de Marcadores no ofrece unos resultados estables. Para aportar estabilidad al sistema se debe limitar la velocidad a la que se desplazan los marcadores.

La velocidad de movimiento de un brazo humano durante la realización de una operación estándar es inferior a 0.1 m/s [32]. Con recudir la velocidad de la traslación de los marcadores a 0,5 m/s es suficiente para aumentar notablemente la estabilidad al sistema.

5.3.3. Movimiento y orientación de la Herramienta

La identificación de operaciones desarrollada en este proyecto se basa en el conocimiento de la posición y orientación del efector final de la herramienta. Este efector se simula en el sistema como un objeto al cual se le asocia el script Movimiento y orientación de la Herramienta.

Como ya se adelantó en el apartado 4.4 de este documento las superficies de referencia se pueden simplificar en una forma cilíndrica o cúbica. Por lo tanto existen dos tipos de herramientas: tipo uno con forma cúbica y tipo 2 con forma cilíndrica.

En la elaboración del script para ambas herramientas se han seguido los pasos descritos en el apartado 4.3 de este documento:

1) Copia de memoria compartida de la posición de los marcadores

```
Vector3 marcador1 = coordenadas.VM1;
Vector3 marcador2 = coordenadas.VM2;
Vector3 marcador3 = coordenadas.VM3;
```

Estas coordenadas son las facilitadas por los scripts de los movimientos de los marcadores

2) Cálculo de los vectores dirección VM1M2 y VM1M3

```
Vector3 VM1M2 = (marcador2 - marcador1);
Vector3 VM1M3 = (marcador3 - marcador1);
```

3) Cálculo de la orientación como ángulos de Euler

```
Vector3 compararx = new Vector3(VM1M3.x * 0, VM1M3.y * 10, VM1M3.z *
10).normalized;
Vector3 comparary = new Vector3(VM1M2.x * 10, VM1M2.y * 0, VM1M2.z *
10).normalized;
Vector3 compararz = new Vector3(VM1M2.x * 10, VM1M2.y * 10, VM1M2.z *
0).normalized;

GameObject Herramientax = new GameObject();
GameObject Herramientay = new GameObject();
GameObject Herramientaz = new GameObject();

Herramientax.transform.forward = compararx;
float angulox = Herramientax.transform.localEulerAngles.x;

Herramientay.transform.right = comparary;
float anguloy = Herramientay.transform.localEulerAngles.y;
```

```
Herramientaz.transform.right = compararz;
float anguloz = Herramientaz.transform.localEulerAngles.z;

Destroy(Herramientax);
Destroy(Herramientay);
Destroy(Herramientaz);

coordenadas.orientacion.x = angulox;
coordenadas.orientacion.y = anguloy;
coordenadas.orientacion.z = anguloz;
```

Es necesaria la creación temporal de tres objetos llamados “Herramientax” “Herramientay” y “Herramientaz”. Cada objeto será utilizado para calcular un ángulo.

4) Cálculo de la posición del efector final

```
/***********POSICION**************/
Vector3 traslx = (VM1M2 * 10F).normalized * coordenadas.desplazamiento.x;
Vector3 trasly = Vector3.Cross(10F * VM1M3, 10F * VM1M2).normalized *
coordenadas.desplazamiento.y;
Vector3 traslz = (VM1M3 * 10F).normalized * coordenadas.desplazamiento.z;

float despx = traslx.x + trasly.x + traslz.x;
float despy = traslx.y + trasly.y + traslz.y;
float despz = traslx.z + trasly.z + traslz.z;

coordenadas.pos.x = marcador1.x + despx;
coordenadas.pos.y = marcador1.y + despy;
coordenadas.pos.z = marcador1.z + despz;
```

El desplazamiento de la herramienta es una propiedad de la actividad, y la variable global desplazamiento es facilitada por el script Interfaz Gráfica de Usuario.

5) Transformar el objeto actual

```
this.transform.rotation = Quaternion.Euler(coordenadas.orientacion);
this.transform.position = coordenadas.pos;
this.transform.localScale = coordenadas.escala;
```

La escala viene dada como propiedad de la actividad, y es facilitada por la Interfaz Gráfica de Usuario como variable global.

Estos cálculos serán realizados únicamente cuando la propiedad “efector” de la actividad que se esté ejecutando sea del tipo que corresponde con la herramienta. En caso contrario la herramienta permanecerá en una posición fija.

5.3.4. Generación de Superficies de Comparación

Este script será asociado a las superficies de referencia en el momento de su carga cuando el Sistema se encuentre en el modo 4, Guiado de Actividades. El comportamiento de estos scripts se describe en el siguiente diagrama de flujo:

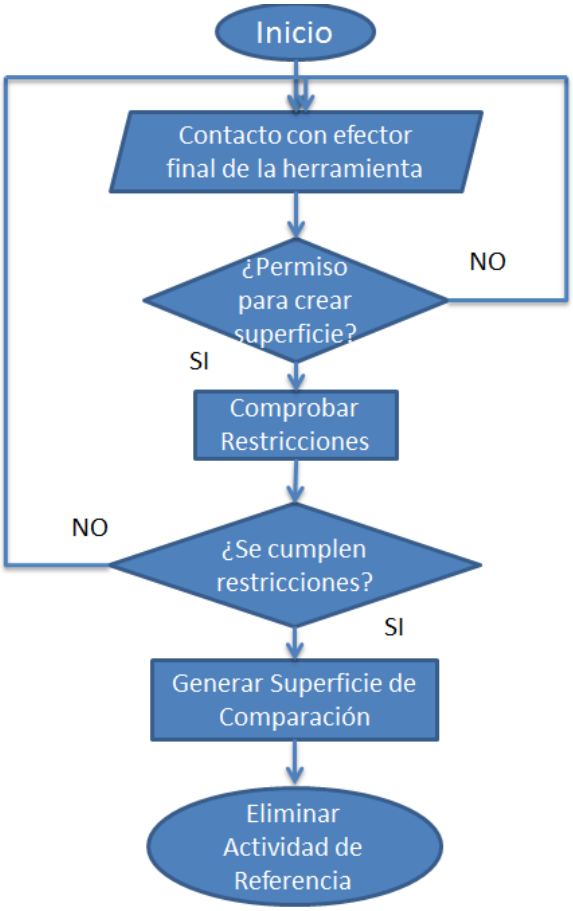


Figura 5.25 Diagrama de Flujo de Creación de una Superficie de Comparación.

En el modo 4 se desea comprobar si una operación realizada en un momento dado coincide con una de referencia almacenada previamente. Estas operaciones serán similares si el efector final pasa por las mismas coordenadas espaciales que este y con una orientación análoga.

La herramienta Unity facilita el método OnTriggerStay. Este método es un disparador que se lanza cuando un objeto permanece en contacto con otro objeto. Gracias a esto se reduce en gran medida el tiempo de procesamiento ya que las restricciones solo se calcularán para las superficies que estén en contacto con el efector de la herramienta.

Se deben de cumplir tres limitaciones:

1) Precisión de la posición: 10, 15, 20 y 40 mm para cada eje.
2) Precisión de la orientación: 5, 10,15 o 30 grados para cada ángulo de Euler.
3) Restricción de ángulos: rotación en el eje x, eje y o eje z no relevante.

Cumplidas estas restricciones se genera una nueva superficie de comparación con la misma escala que la superficie de referencia pero cambiando su posición y orientación, que será igual a la del efector de la herramienta en ese momento.

5.3.5. Movimiento de la cámara

El fin de este Sistema es que el usuario pueda identificar en tiempo real algún fallo en la realización de una operación manual. Es necesario por lo tanto que en la pantalla se representen las superficies a una escala visible por el operario que en muchos casos no podrá acercarse demasiado al monitor. Por esto se permite el movimiento de la cámara por el entorno de trabajo simulado. De esta forma el operario puede modificar el foco de la cámara hacia la zona donde están las superficies de Referencia.

La orientación y posición de la cámara se definirán en el momento de la creación de la superficie de referencia. La posición se ajusta con las siguientes teclas:

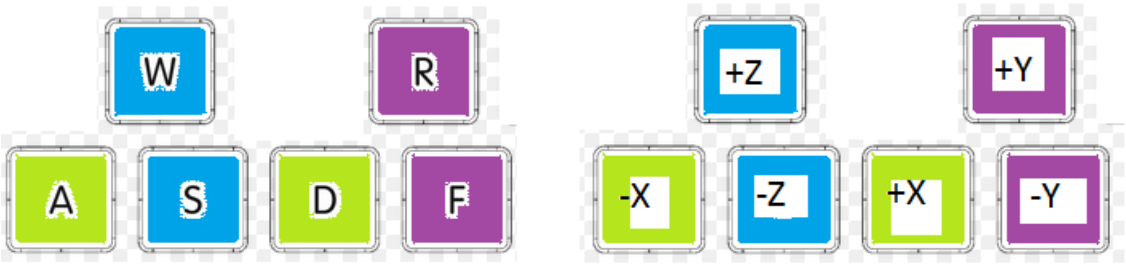


Figura 5.26 Teclas movimiento de la cámara

La orientación se modifica con el movimiento del ratón.

5.4. Otros Programas

5.4.1. Configuración de Actividades

Una actividad de referencia es almacenada en el sistema como un archivo de texto denominado ActividadSR.txt. La primera línea de este archivo son las propiedades de la actividad. Para facilitar al usuario la programación de estas propiedades se facilita el programa Configuración de Actividades. Esta aplicación no está conectada con la aplicación Monitorización de Operaciones, por lo que se puede ejecutar de manera independiente a este. Su interfaz es la siguiente:

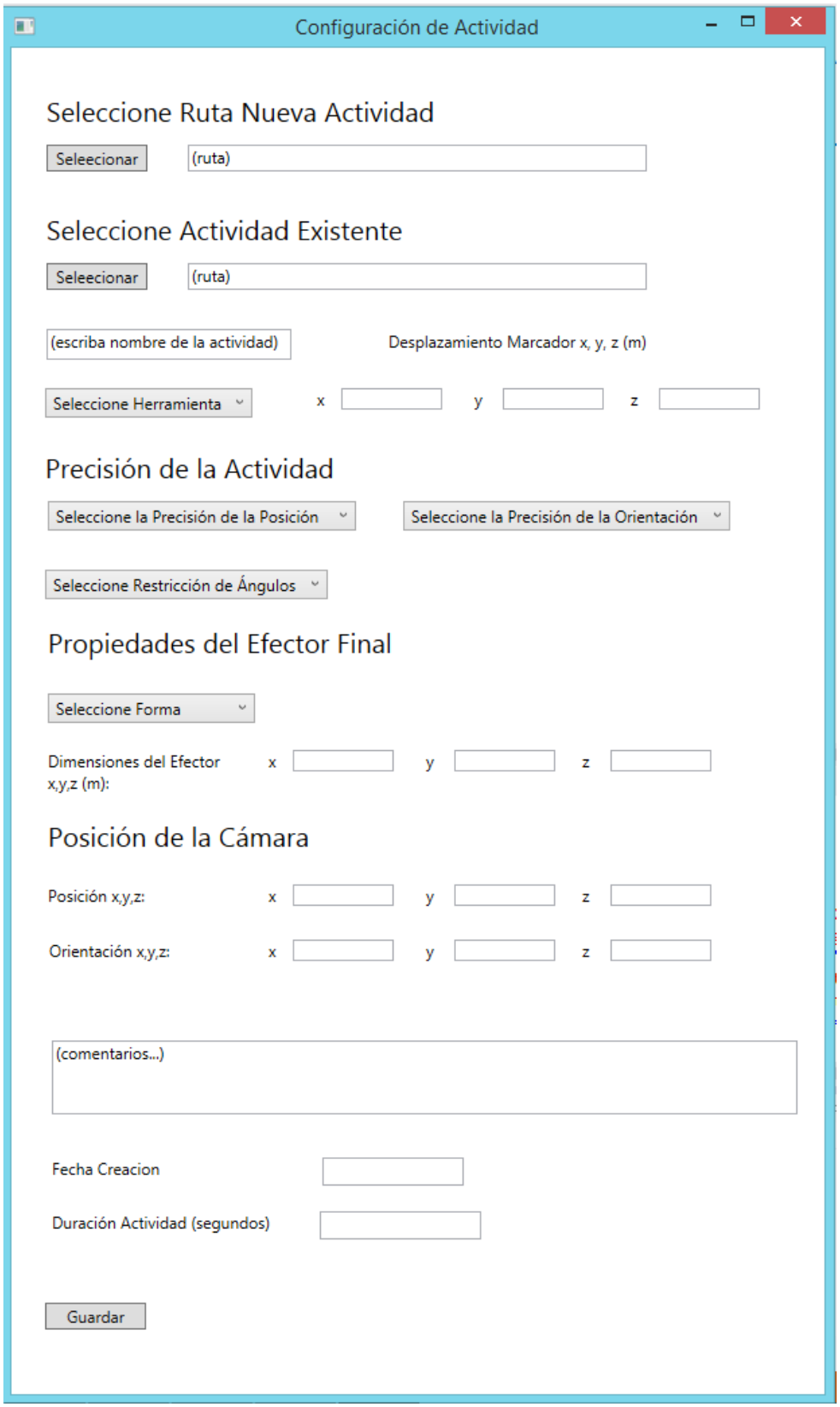


Figura 5.27 Interfaz Configuración de Actividades

Seleccione Ruta Nueva Actividad

La configuración de una nueva actividad se hace seleccionando una carpeta donde se van a guardar todos los datos de la actividad, es decir, las actividades de referencia y las de comparación. En la siguiente imagen se ha seleccionado la carpeta Lijado. Pulsando en "Aceptar" se crea el archivo de texto ActividadSR.txt donde se van a almacenar las propiedades. En el bloque de texto se escribe la ruta seleccionada.

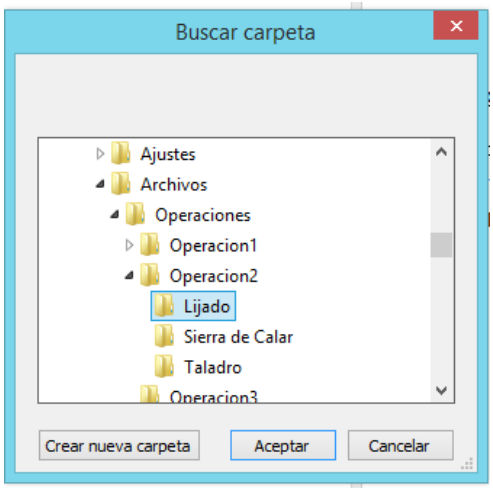


Figura 5.28 Seleccionando Ruta Lijado

Seleccione Actividad Existente

Pulsando en el botón seleccionar se abre un cuadro de texto donde se debe seleccionar el archivo ActividadSR.txt que se quiere modificar. Pulsando en "Abrir" se cargan en el programa las propiedades contenidas en este archivo.

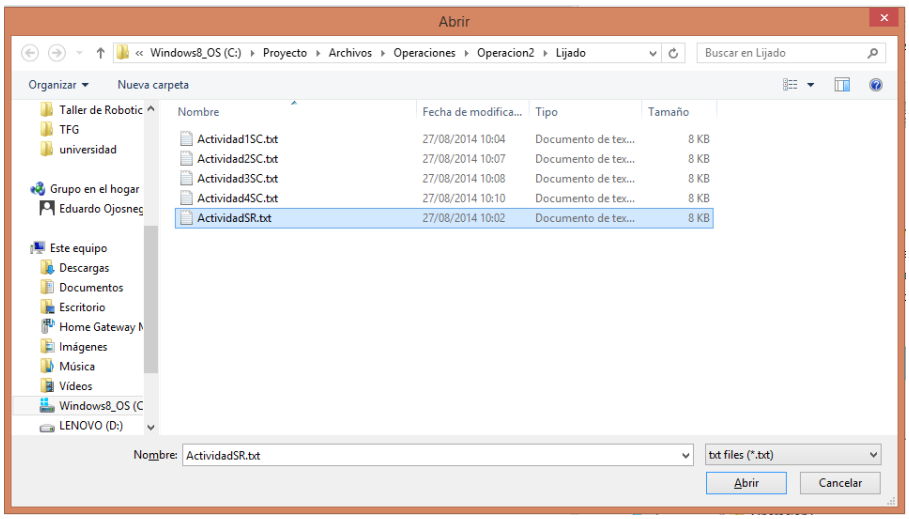


Figura 5.29 Seleccionando archivo ActividadSR.txt

(Escribe nombre de la actividad)

En este cuadro de texto se ha de escribir el nombre que se le quiere dar a la actividad. Este nombre será representado como información en la aplicación Monitorización de Operaciones en los modos 2 y 4.

Seleccionar Herramienta

Al pulsar en este botón se despliega una lista donde se puede elegir la Herramienta que se debe seguir durante esta actividad. La herramienta coincidirá con la establecida en el programa Configuración de Seguimiento.

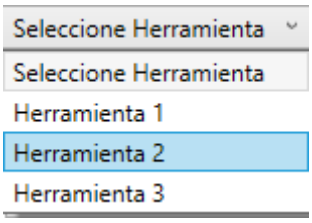


Figura 5.30 Selección de Herramienta 2

Desplazamiento Marcador x, y, z (m)

Este corresponde con el desplazamiento del marcador principal al efector final de la herramienta. Hay que tener en cuenta la dirección de los ejes. En el ejemplo se ha escrito un desplazamiento en el eje x de 5 cm, en el eje y de -9 cm y sin desplazamiento del eje z.

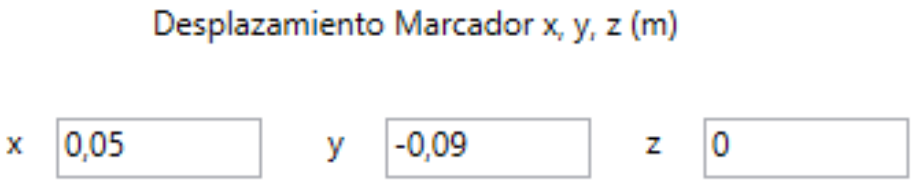


Figura 5.31 Desplazamiento del Marcador principal

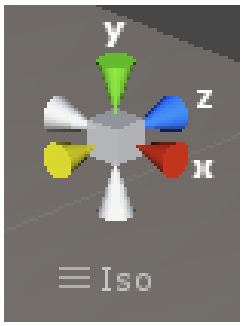


Figura 5.32 Coordenadas Cartesianas

Seleccione la Precisión de la Posición

En la lista desplegable se puede seleccionar la precisión requerida de la posición en mm.

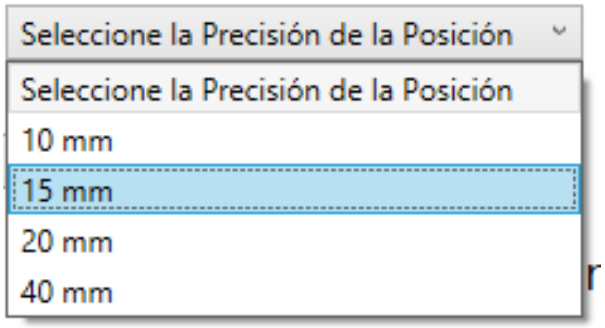


Figura 5.33 Precisión Posición = 15mm

Seleccione la Precisión de la Orientación

En la lista desplegable se puede seleccionar la precisión requerida de la orientación en grados de los ángulos de Euler.

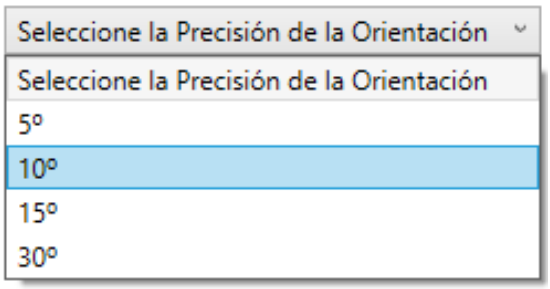


Figura 5.34 Precisión Orientación = 10º

Seleccione la restricción de Ángulos

En la lista desplegable se puede seleccionar un eje como no relevante en el cálculo de las restricciones de los ángulos.

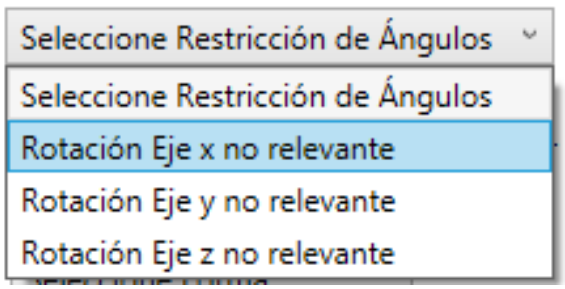


Figura 5.35 Precisión Orientación = 10º

Seleccione Forma:

Se permite seleccionar la forma del efector final.

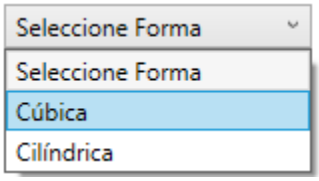


Figura 5.36 Forma del efector final = cúbica

Dimensiones del efector x, y, z (m)

Estas dimensiones estarán dadas en metros y siempre deben ser positivas. Se admite una dimensión mínima de 0,5 cm.

Figura 5.37 Dimensiones del Efector: x = 2cm; y = 5cm; z = 5mm

Posición y Orientación de la cámara

Esta propiedad se podrá modificar en el modo 2. Por defecto al seleccionar una nueva activad se cargan los valores que corresponden a la posición de la cámara cuando se inicia la aplicación Monitorización de Operaciones. Estos valores son los escritos en este ejemplo.

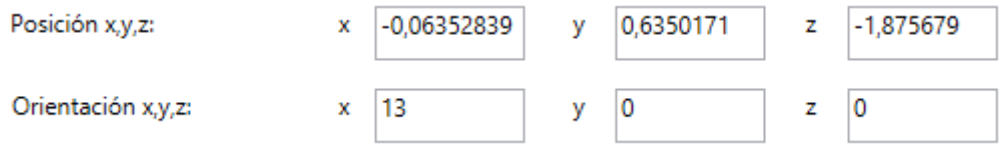


Figura 5.38 Posición y Orientación de la cámara

Comentarios

En este cuadro de texto se debe escribir un comentario meramente informativo al realizar la actividad.

Fecha de Creación y Duración de Actividad

Estas propiedades se podrán modificar en el modo 2. Se pueden modificar manualmente siempre que sigan el formato del ejemplo.

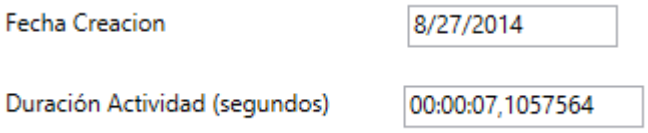


Figura 5.39 Fecha de Creación y Duración Actividad de Referencia

Guardar

Al pulsar en este botón se modifican las propiedades de la actividad seleccionada.

5.4.2. Configuración de Seguimiento

Esta puede ser sin duda la aplicación más importante del sistema. Para garantizar un seguimiento estable es necesario que la segmentación de cada color sea lo más precisa posible. Con el fin de facilitar esta labor al usuario se ha desarrollado la aplicación Configuración de Seguimiento. Su funcionamiento es similar a la aplicación Seguimiento de Marcadores. Utiliza los mismos procedimientos descritos en el apartado 4.3 de este documento para realizar el seguimiento de los marcadores, pero esta aplicación no está conectada con el programa Monitorización de Operaciones. Permite realizar varias configuraciones que se describen a continuación:

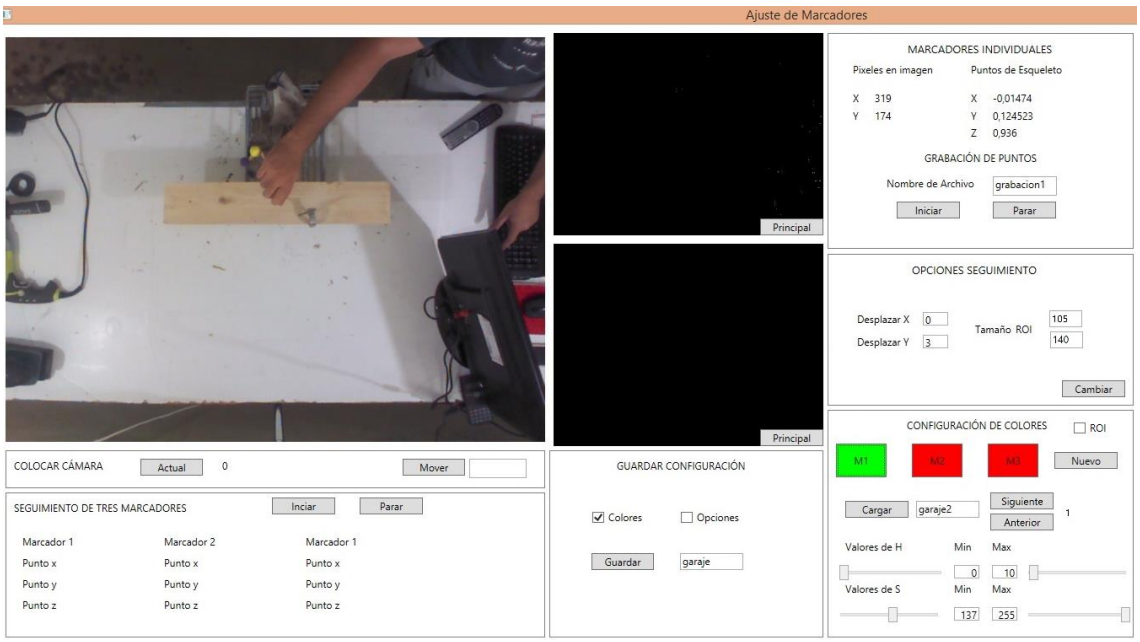


Figura 5.40 Interfaz Configuración de Seguimiento

1) Configuración de colores

La segmentación de los colores se realiza con la selección de los valores superiores e inferiores del matiz (H_{HSV}) y Saturación (S_{HSV}) del modelo de color HSV. Para una selección más cómoda se facilitan barras deslizantes.

Se permite configurar estos colores desde cero (botón “nuevo”) o modificar una configuración existente (“cargar”). Pulsando el botón “nuevo” se abre un cuadro de dialogo donde se ha de seleccionar la carpeta donde almacenar esta configuración. Pulsando en el botón “cargar” se debe seleccionar un archivo de texto que contenga la configuración a modificar.

Se han de configurar los colores de un máximo de tres herramientas. Se puede navegar por cada herramienta pulsando los botones “Siguiente” y “Anterior”. Pulsando en los botones “M1”, “M2” y “M3” se selecciona el marcador a buscar. Seleccionando la casilla ROI se bloquea la región de interés en el centro de la imagen. Esto puede ser útil a la hora de buscar los colores.

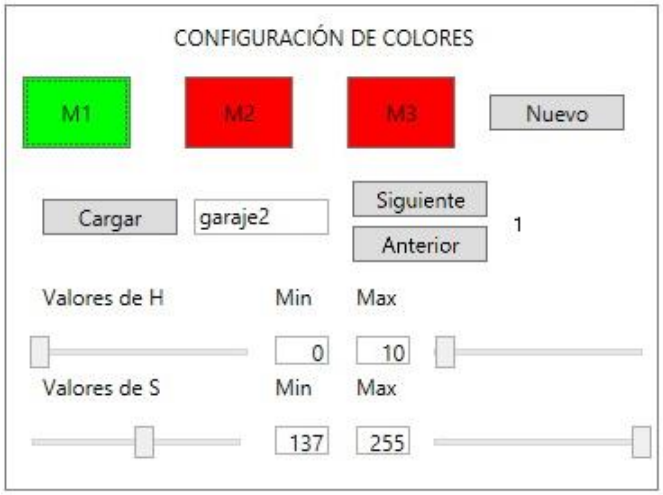


Figura 5.41 Configuración de Colores

2) Imágenes

Se muestran en pantalla la imagen de la región de interés a color, la imagen después de ser binarizada y tras realizar una apertura. Pulsando en los botones “principal”, la imagen representada en esa casilla se intercambia con la imagen de mayor tamaño.

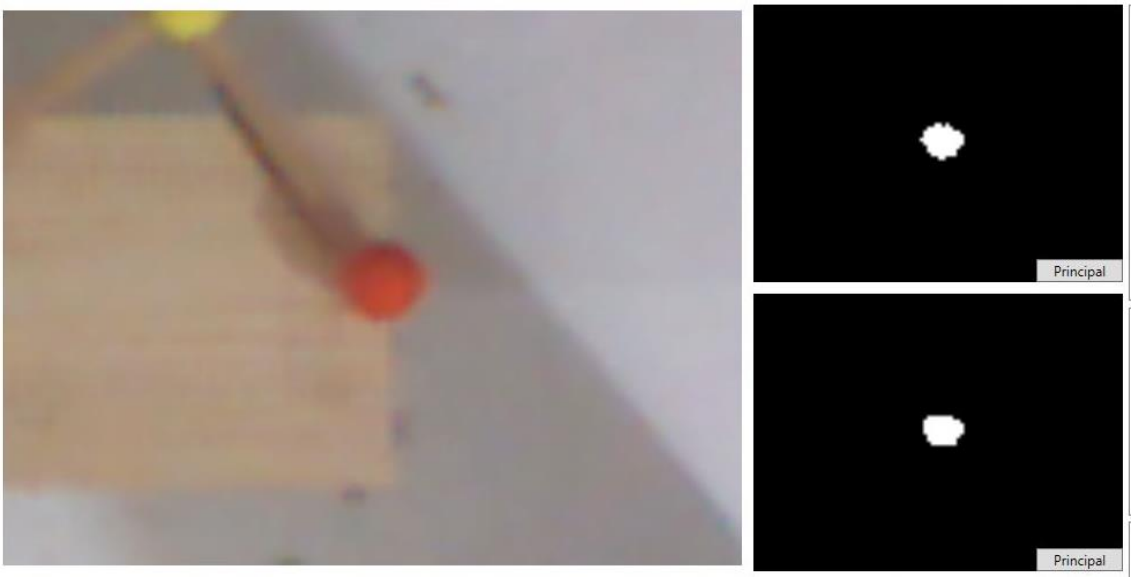


Figura 5.42 Imágenes seguimiento del Marcador

3) Opciones de Seguimiento

Como ya se dijo en el apartado 5.2.1 es necesario añadir un desplazamiento para que el pixel correspondiente al centro de gravedad del marcador corresponda las longitudes en tres dimensiones recogidas por el sensor de profundidad. Este se debe de calcular de forma experimental, como se verá en el apartado ensayos de calibración. El tamaño máximo del ROI puede hacer que el programa de Seguimiento de Marcadores se adapte mejor o peor a cambios bruscos de velocidad.

Figura 5.43 Opciones de Seguimiento

4) Grabación de Puntos

Este bloque sirve para comprobar si las medidas que se están calculando sobre el marcador activo son las correctas. Se indican las coordenadas en la que se encuentra el marcador activo en la imagen en pixeles y las longitudes tridimensionales en metros. Con los botones Iniciar y grabar se pueden almacenar

una secuencia de puntos para realizar ensayos de calibración como los realizados en el apartado 6.1 de este documento.

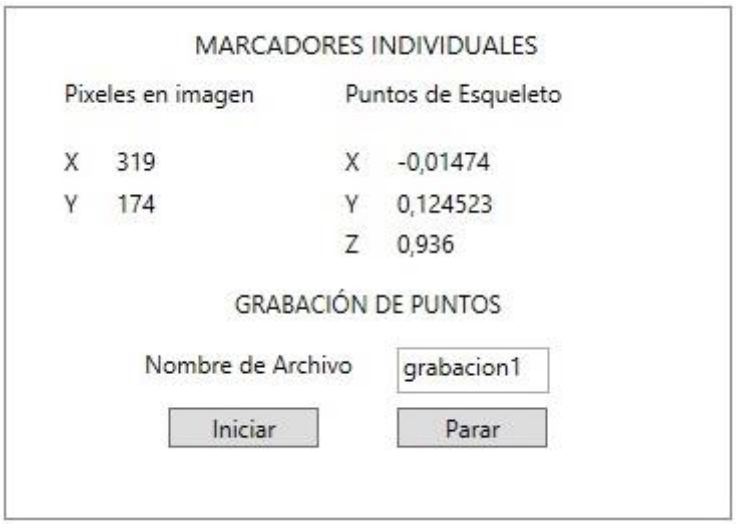


Figura 5.44 Grabación de Puntos

5) Colocar Cámara

Es necesario que la cámara este colocada de forma perpendicular a la imagen. Con el botón “actual" se calcula el ángulo actual formado entre el eje “y” cartesiano y el vector normal a los sensores. Con el botón “Mover” se permite mover la cámara para conseguir que esté perpendicular a la zona de trabajo, es decir a 90º.

Figura 5.45 Colocar Cámara

6) Seguimiento simultáneo de tres marcadores

Pulsando el botón “Iniciar” se permite el seguimiento simultáneo de los tres marcadores de la herramienta actualmente activa. Se indican sus coordenadas en tres dimensiones. Si la medida que se está dando de un marcador es la correcta el rectángulo debajo de las coordenadas se colorea de color verde; en caso contrario de color rojo. Pulsando el botón parar se pasa al modo de búsqueda de un solo marcador.

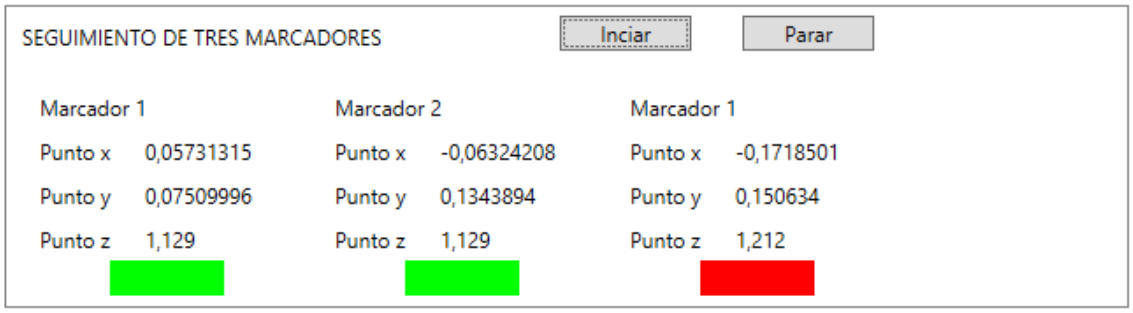


Figura 5.46 Seguimiento simultáneo de tres Marcadores

7) Guardar Configuración

Pulsando en el botón "Guardar" se almacena en el archivo de texto previamente seleccionado la configuración creada. Con la casilla "Opciones" seleccionada se modifican las opciones de seguimiento (descritas en 3)); con la casilla "Colores" seleccionada se modifica la configuración de colores (descrita en 1)).

Figura 5.47 Guardar Configuración

5.4.3. Cargar Actividades

El modo 3, Modificación de Actividades y el Modo 5, Comprobación de Actividades, permiten cargar actividades previamente construidas en los otros modos. Para ello no es necesario que el sistema esté siguiendo a los marcadores, por lo que se desarrolla la aplicación Carga de Actividades que trabaja como cliente de la aplicación Monitorización de Operaciones en estos modos. En definitiva la funcionalidad en estos dos modos es similar por la que se desarrolla una aplicación que sirva para ambos modos.

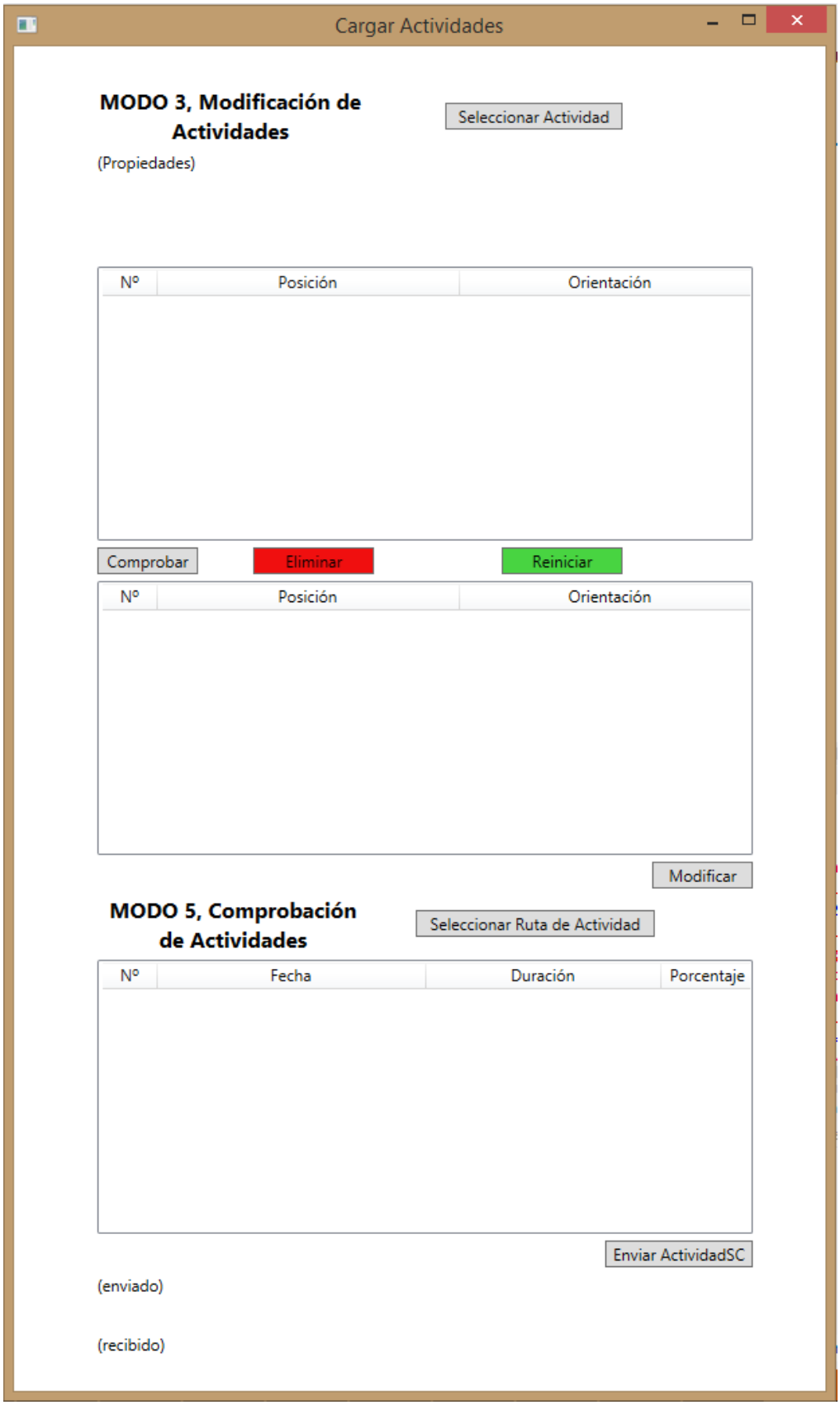


Figura 5.48 Interfaz Gráfica Cargar Actividades

1) Modo 3, Modificación de Actividades

Pulsando el botón “Seleccionar Actividad” se abre un cuadro diálogo donde se ha de seleccionar el archivo de texto ActividadSR.txt a modificar. En ese momento esta aplicación queda conectada al programa Monitorización de actividades. Se cargan en la lista superior las Superficies de Referencia contenidas en esta actividad en orden de creación con la posición y orientación que la definen.

Pulsando el botón “Comprobar” se prepara el envío de las actividades seleccionadas en la lista superior. Esto provoca que se pinten de color azul a modo de comprobación las superficies seleccionadas. La cadena de texto a enviar tendrá el formato:

A;identificador_superficie1,identificador_superficie2;...;

Figura 5.49 Comprobar superficies seleccionadas

El botón “Eliminar” baja a la lista central las superficies seleccionadas y prepara para el envío la cadena de texto con formato:

R; identificador_superficie1,identificador_superficie2;...;

Las superficies cambian a color rojo en el programa servidor.

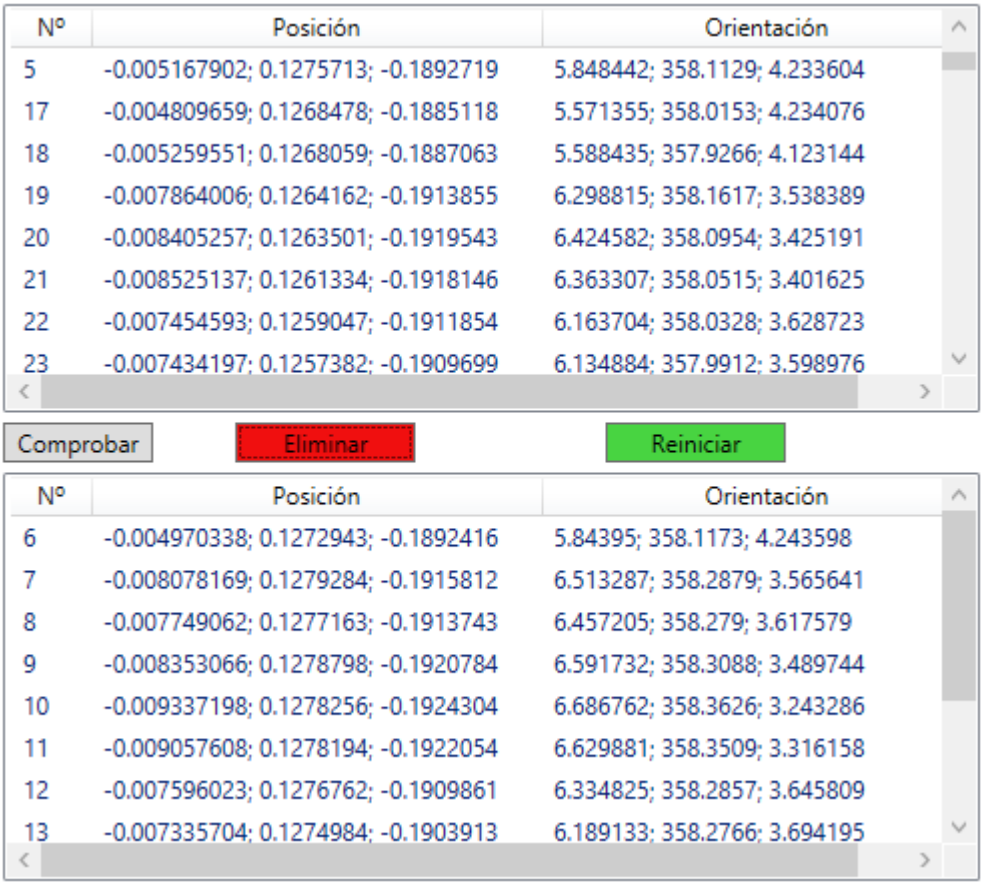


Figura 5.50 Eliminar superficies seleccionadas

Si las superficies propuestas a eliminar no han sido seleccionadas correctamente se puede pulsar el botón “Reiniciar”. De esta forma suben a la lista superior las superficies contenidas en la lista central y prepara para el envío la ruta de actividad, evitando al usuario tener que volver a seleccionar la ruta. Las superficies se vuelven a cargar en el servidor.

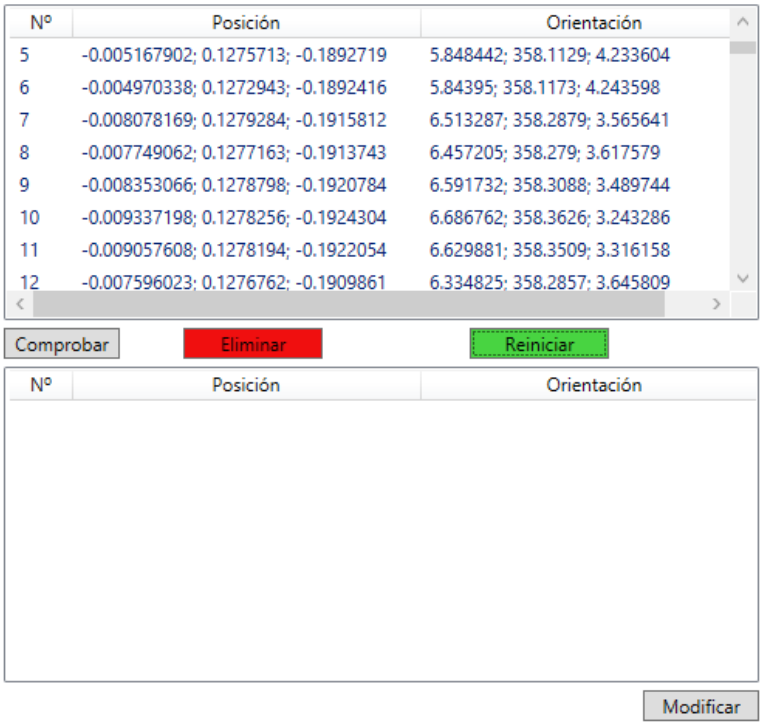


Figura 5.51 Reiniciar carga superficies

Una vez verificadas las superficies a eliminar se debe pulsar el botón "Modificar" que genera un archivo de texto ActividadSR.txt con las mismas propiedades que el archivo original y con las Superficies de Referencia modificadas.

2) Modo 5, Comprobación de Actividades

Figura 5.52 Selección ActividadSC

El botón "Seleccionar Ruta de Actividad" provoca la conexión con el servidor, el programa Monitorización de Operaciones. Se abre un cuadro de dialogo donde el

usuario puede seleccionar la ruta de la actividad a cargar. La ruta es enviada al servidor que cargará las superficies de referencia y propiedades de esta actividad.

En la lista inferior se cargan las ActividadesSC.txt en orden de creación. La lista superior indica la fecha de creación de la actividad, la duración de la actividad y el porcentaje completada respecto a la actividad de referencia.

Pulsando el botón “EnviarActividadSC” se prepara para el envío la ruta que contiene el archivo Actividad*SC.txt seleccionado en la lista inferior, donde el * es el Nº identificador que aparece en la lista. Esta actividad se cargará junto a la actividad de referencia en el programa Monitorización de Operaciones una vez recibida la cadena de texto con formato:

S;C:\...\Actividad*SC.txt;

6. Experimentación y Resultados

La configuración del seguimiento de marcadores puede parecer algo tedioso, por lo que en el primer apartado de este capítulo se describe el proceso paso a paso para lograr un sistema estable. En el segundo apartado se exponen los datos obtenidos en los ensayos de calibración del sistema. El siguiente y último apartado presenta los resultados obtenidos en operaciones manuales de taladrado, lijado y corte con sierra de calar.

6.1. Configuración del sistema

Una buena configuración óptima del seguimiento de los marcadores es crucial para el buen funcionamiento del total del sistema. A continuación se describen los pasos a seguir en la aplicación Configuración de Seguimiento para conseguir la máxima estabilidad del Sistema.

1. Colocar cámara

El primer paso es colocar la cámara a una altura entre 1 y 1,5 metros que permita la visualización de toda la mesa de trabajo. La orientación del vector normal a los sensores debe ser perpendicular al plano formado por la mesa. Es en el centro de los sensores donde mayor precisión se va a conseguir, por lo tanto se debe conseguir que el centro de la imagen coincida con zona de trabajo. Con este fin se propone el uso de soportes tipo jirafa como el de la Figura 6.1.

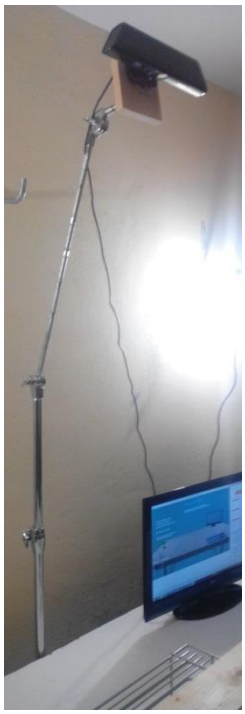


Figura 6.1 Soporte jirafa

Los botones de movimiento de la cámara y la imagen a color pueden ayudarnos en esta tarea. Se debe conseguir un ángulo de -90º

Figura 6.2 Colocar cámara

2. Búsqueda de colores

Los marcadores deben ser de colores llamativos diferentes a la herramienta a seguir. Una vez seleccionados los marcadores de la herramienta se han de buscar sus valores HSV pertenecientes a esos marcadores.

Ejemplo de Búsqueda del primer marcador: Marcador Naranja

a) Seleccionar “Nuevo” -> “M1” –> “casilla ROI”. De esta forma se bloquea la región de interés en la zona central de la pantalla. Colocamos en esa zona el marcador.

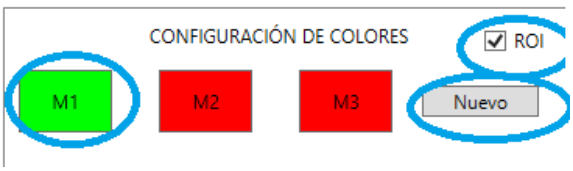


Figura 6.3 Nuevo Marcador

b) Desliza la barra inferior derecha hasta el máximo valor

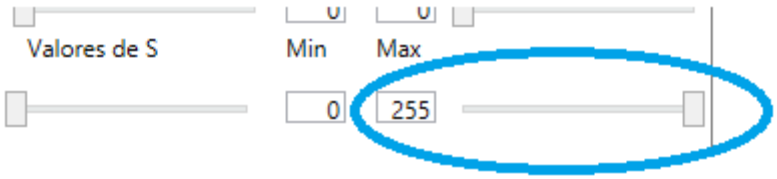


Figura 6.4 Deslizar la barra inferior

c) Desliza la barra superior derecha hasta ver el marcador totalmente en blanco.

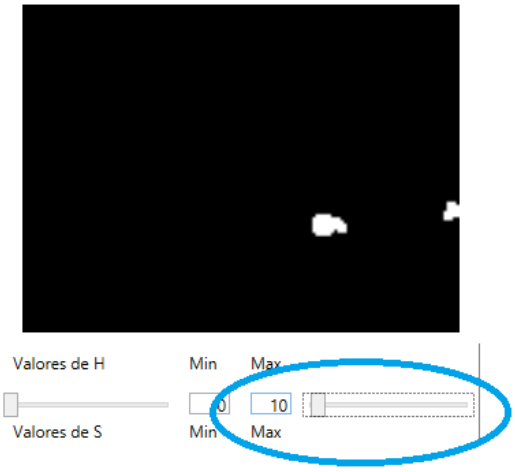


Figura 6.5 Deslizar Barra superior

d) Deslizar la barra inferior izquierda hasta eliminar todos los puntos blancos alrededor.

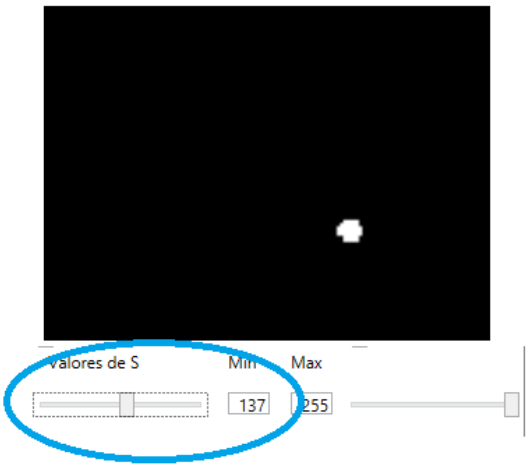


Figura 6.6 Deslizar Barra Inferior

e) Comprobar Seguimiento Estable. Deseleccionar la casilla ROI y juntar al marcador la herramienta a la cual pertenece. Si el seguimiento es estable se puede pasar a buscar otro marcador.

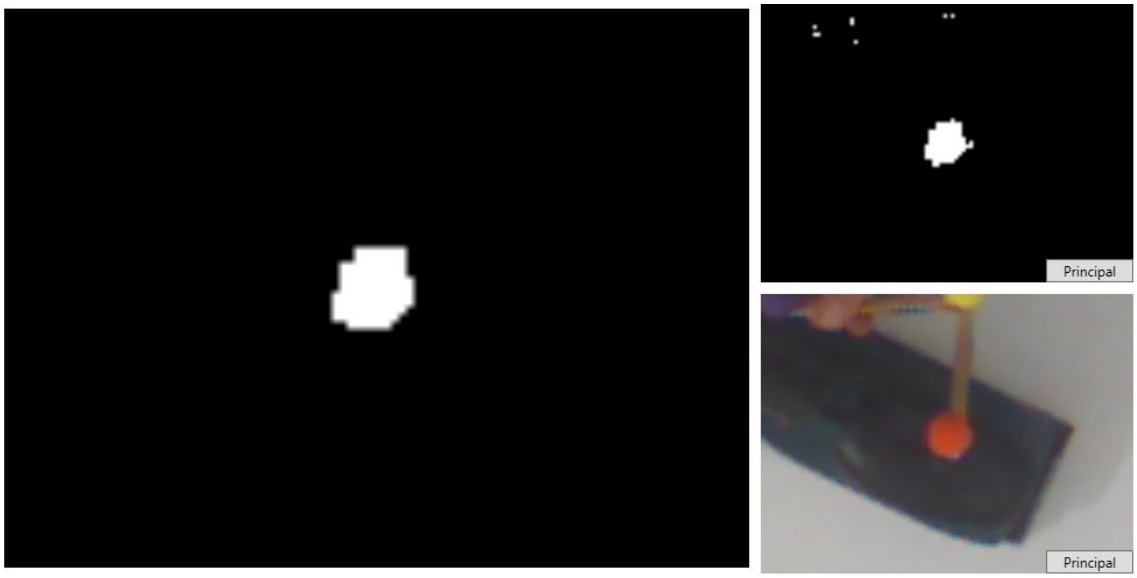


Figura 6.7 Comprobar seguimiento estable

f) Repetir los pasos anteriores con los demás marcadores.
g) Pulsar el botón siguiente para comenzar con la siguiente herramienta

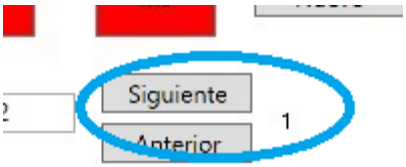


Figura 6.8 Pulsar Siguiente

3. Comprobar Distancias

En el cuadro “MARCADORES INDIVIDUALES” aparecen los puntos en el espacio en el que se encuentra el marcador. Con comprobar la distancia “Z”, distancia entre el sensor y el marcador, será suficiente.

Debido a la distancia entre el sensor de profundidad y de color de Kinect puede que estos valores no coincidan. Los valores preestablecidos de “X” e “Y” de las “OPCIONES DE SEGUIMIENTO” son 0 y 3 respectivamente. En el caso de no coincidir los valores encontrados con variar el desplazamiento en X un máximo de dos unidades (buscar valor entre 1 y 5) será suficiente.

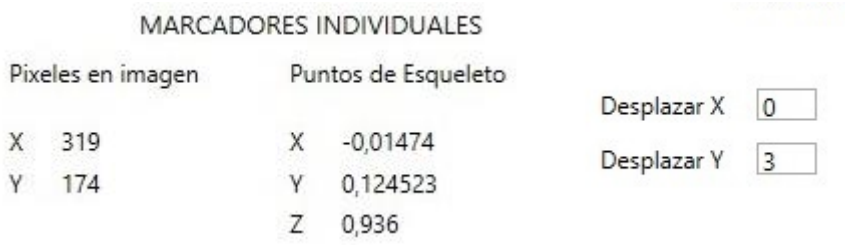


Figura 6.9 Comprobar coordenadas de marcador individual

4. Tamaño de la ROI

Pulsando el botón "Iniciar" comienza la búsqueda de los tres marcadores de la herramienta activa a la vez. Se puede modificar el tamaño de la región de interés para comprobar cómo se comporta el seguimiento ante movimientos bruscos. Los tamaños deben pertenecer a la resolución 4:3.

Figura 6.10 Determinar tamaño de la ROI

6.2. Ensayos de calibración

a) Precisión del sistema

Los documentos del Kinect de Windows para Kinect facilitan información sobre la precisión y exactitud del sensor de Profundidad del sistema NUI ST de cálculo de la posición del esqueleto del cuerpo humano. Se escriben en los siguientes dos párrafos:

"*Los errores de medición y el ruido son subproductos de casi cualquier sistema que mide una cantidad física a través de un sensor. Las características de este error se describen generalmente por la exactitud y la precisión del sistema, donde la exactitud se define como el grado de proximidad de la cantidad medida a su valor real, y la precisión se define como el grado en que las mediciones son repetidas*

cerca uno del otro. Un sistema preciso no tiene ningún error sistemático en las mediciones y por lo tanto no agrega un sesgo sistemático.

Al igual que cualquier sistema de medición, los datos de posiciones conjuntas devueltos por el sistema NUI ST tiene un poco de ruido. Hay muchos parámetros que afectan a las características y el nivel de ruido, que incluyen iluminación de la habitación; el tamaño del cuerpo de una persona; la distancia a la persona de la matriz de sensores; pose de la persona (por ejemplo, para los datos de mano, si la mano de la persona está abierta o puño); ubicación de la matriz de sensores; ruido de cuantificación; redondeo efectos introducidos por cálculos; etcétera. Tenga en cuenta que las posiciones conjuntas devueltos por ST son exactos, lo que significa que no existe un sesgo en los datos de posición conjuntas a las posiciones reales en el mundo real. Esto significa que si una persona está en reposo, entonces el promedio de los datos de posiciones conjuntas, con el tiempo, se encuentra cerca de las posiciones en el mundo real. Sin embargo, los datos de posiciones conjuntas no son necesariamente perfectamente precisos, lo que significa que se encuentran diseminados por las posiciones correctas en cada fotograma. En la práctica, las posiciones comunes son exactos dentro de un rango de los centímetros, no milímetros."

Se realiza el siguiente experimento para demostrar el ruido de nuestro sistema. La aplicación Configuración del Seguimiento permite almacenar en un archivo de texto las coordenadas pertenecientes a los marcadores.

Figura 6.11 Almacenar puntos para calibración

Se calculan los datos recogidos de las coordenadas en pixeles de un marcador ubicado en una posición fija. El gráfico de dispersión creado con los datos del ANEXO II: DATOS DE CALIBRACIÓN arroja los resultados de la Figura 6.12.

Vemos como la variación no es superior a un pixel. Esta variación traducida a longitudes creada con los datos del ANEXO II arroja los resultados mostrados en la Figura 6.13 para cada eje de coordenadas.

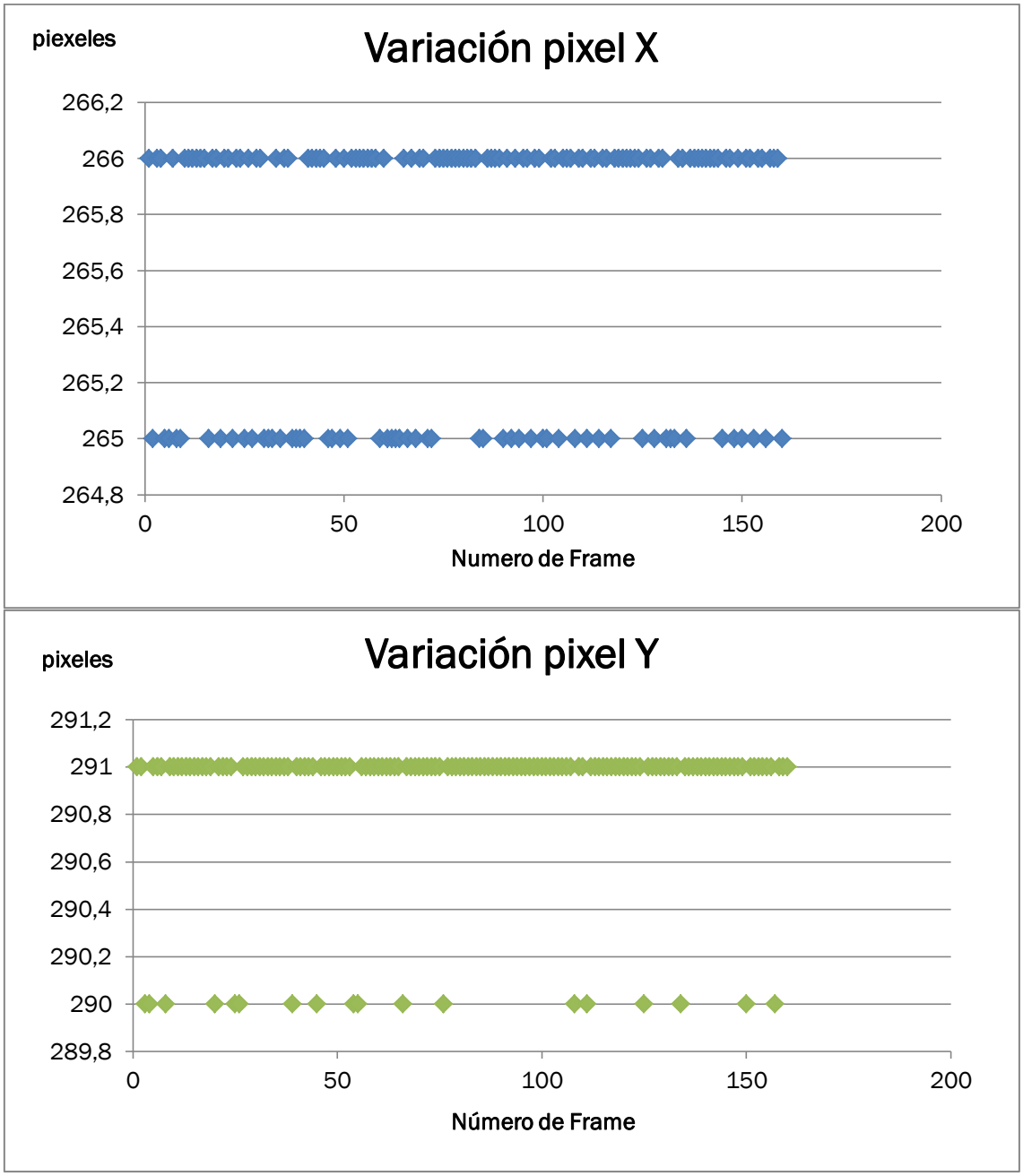


Figura 6.12 Diagrama de Dispersión de la variación de pixeles de un marcador en posición fija

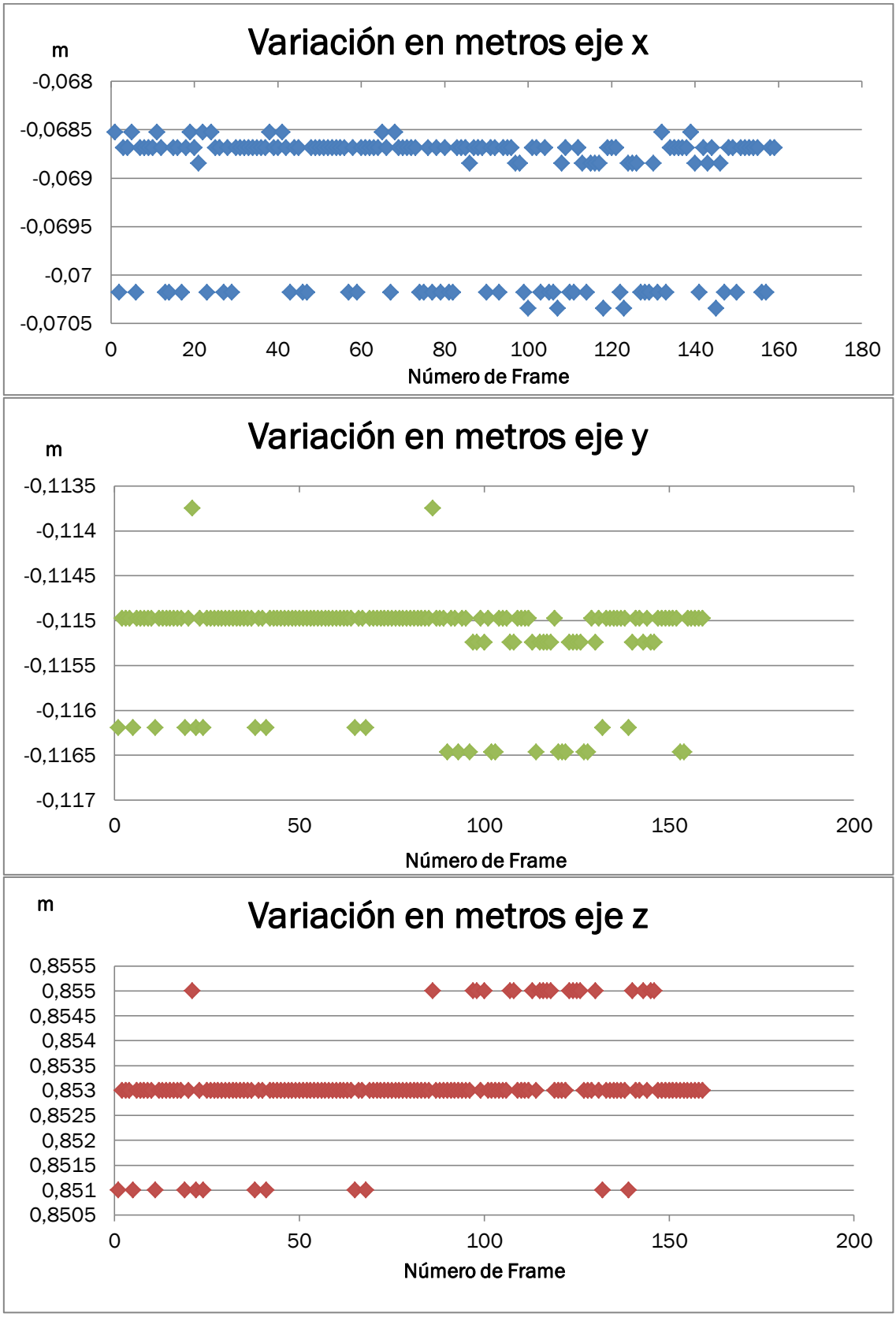


Figura 6.13 Variación en metros de las coordenadas sobre cada eje de un marcador fijo

Se demuestra lo que se expone en la documentación facilitada por Microsoft. El sistema es exacto pero no preciso. La variación máxima se produce en el eje x con un valor de 4 mm por lo tanto por los resultados de este experimento podríamos asegurar que el sistema desarrollado en este proyecto ofrece una precisión máxima de 4mm. No obstante Kinect solo asegura una precisión "*dentro de un rango de los centímetros*", por lo tanto **se establece la precisión del sistema como un centímetro.**

b) Exactitud del sistema

Los siguientes gráficos representan el desplazamiento sobre el eje x de 0 a 10 cm. El ensayo se ha realizado de forma manual. Los datos están contenidos en el Anexo III: DATOS DE CALIBRACIÓN.

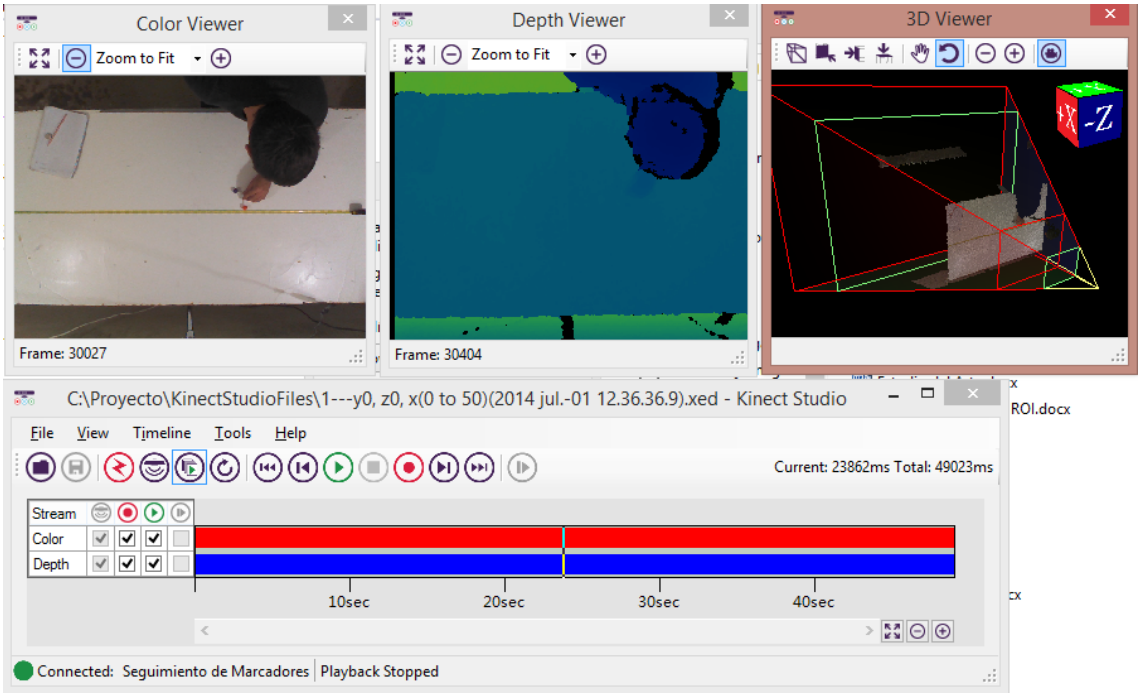


Figura 6.14 Realizando ensayo de exactitud

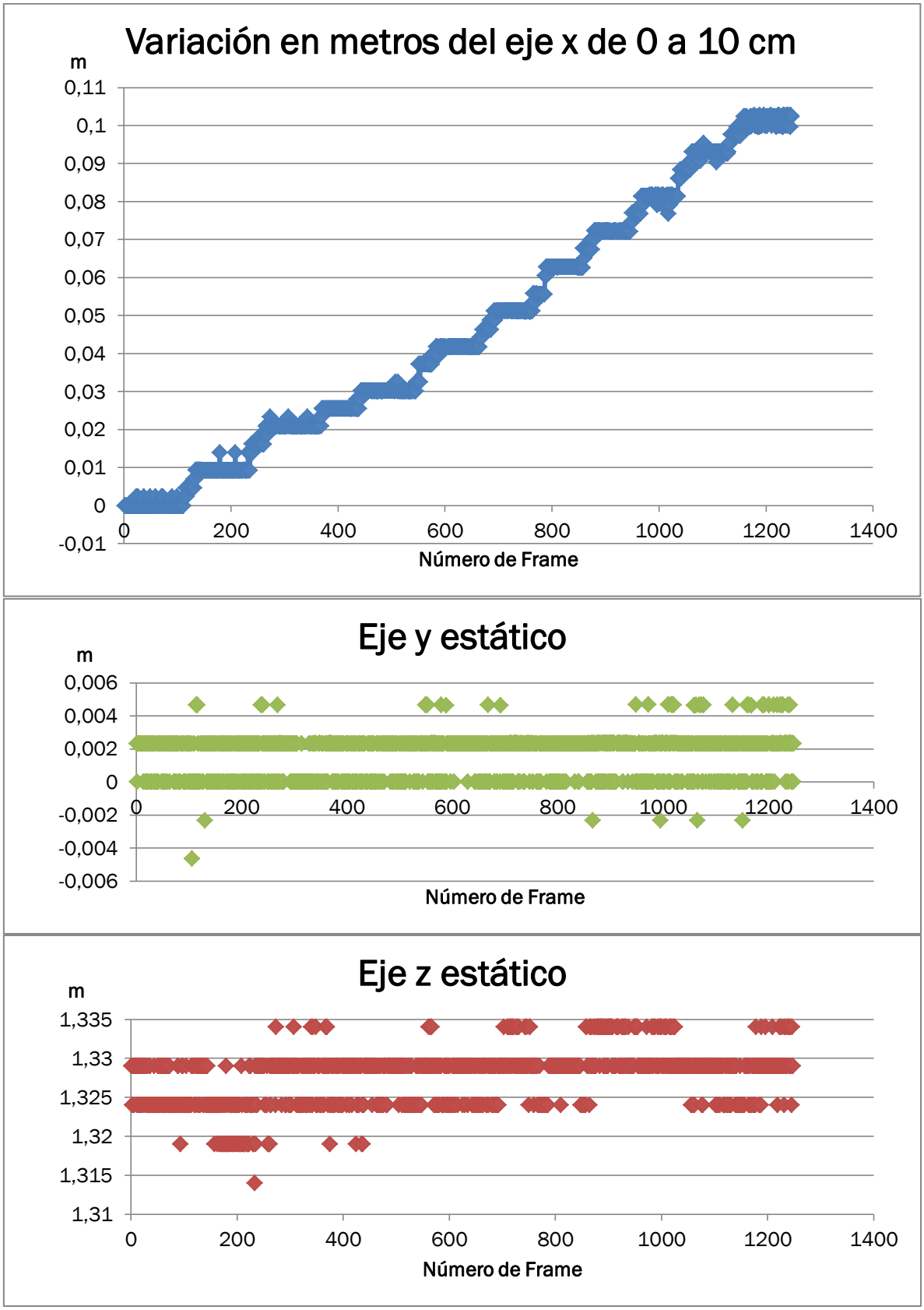


Figura 6.15 Variación en metros en los tres ejes del movimiento de 0 a 10cm

A la vista de estos resultados se demuestra que **el sistema es exacto** como garantizaba en la documentación de Microsoft.

c) Filtrado de los datos

La latencia en este sistema puede ser definida como el tiempo que tarda desde que el usuario realiza un movimiento de la herramienta hasta el momento en que el usuario ve la respuesta a su movimiento del cuerpo en la pantalla.

Un filtro típico en respuesta a un movimiento añade latencia ya que el filtrado de la coordenada es el desfase entre la salida y la entrada cuando hay movimiento en los datos de entrada. La cantidad de la latencia depende de la rapidez de la herramienta en movimiento.

Kinect ofrece el resultado de un filtro normal en respuesta al movimiento brusco de una articulación realizado por un usuario.

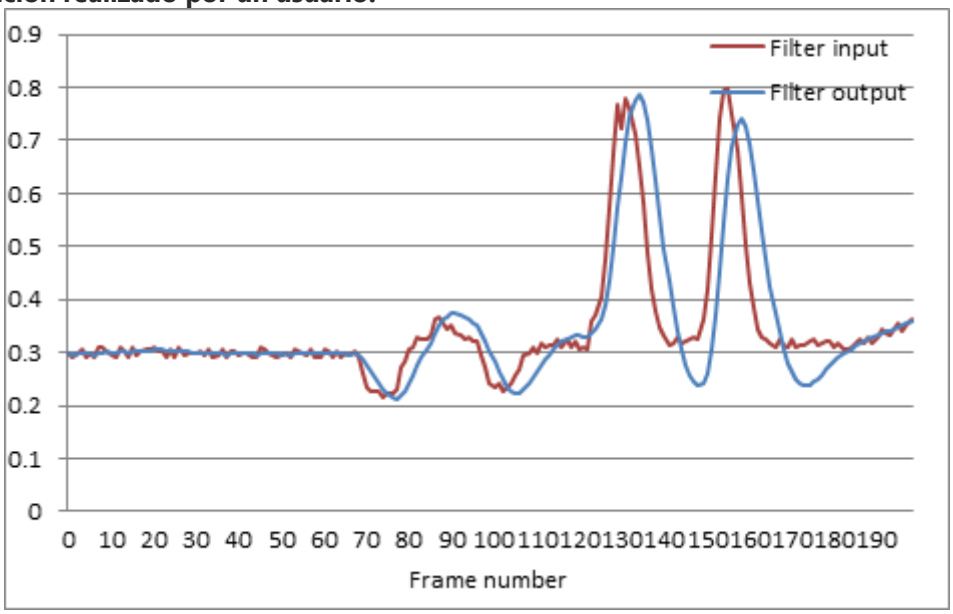


Figura 6.16 Filtro normal en respuesta al movimiento brusco de una articulación

Vemos que esta latencia no es muy elevada, si se tiene en cuenta que un operario nunca va a mover una herramienta a una velocidad superior de 0,5 m/s. Con un filtro sencillo se eliminan completamente el ruido que provoca la imprecisión de la operación. Este filtro es implementado en el script "Movimiento de Marcadores" descrito en el apartado 5.3.2 de este documento.

6.3. Resultados en operaciones manuales

El objetivo de este apartado es el de exponer un ejemplo del seguimiento real de operaciones manuales con el sistema desarrollado en este proyecto. Se describen a la vez los pasos a realizar en cada modo para operaciones de lijado, taladrado y serrado de una pieza de madera.

- Modo 1, Configuración una actividad de lijado

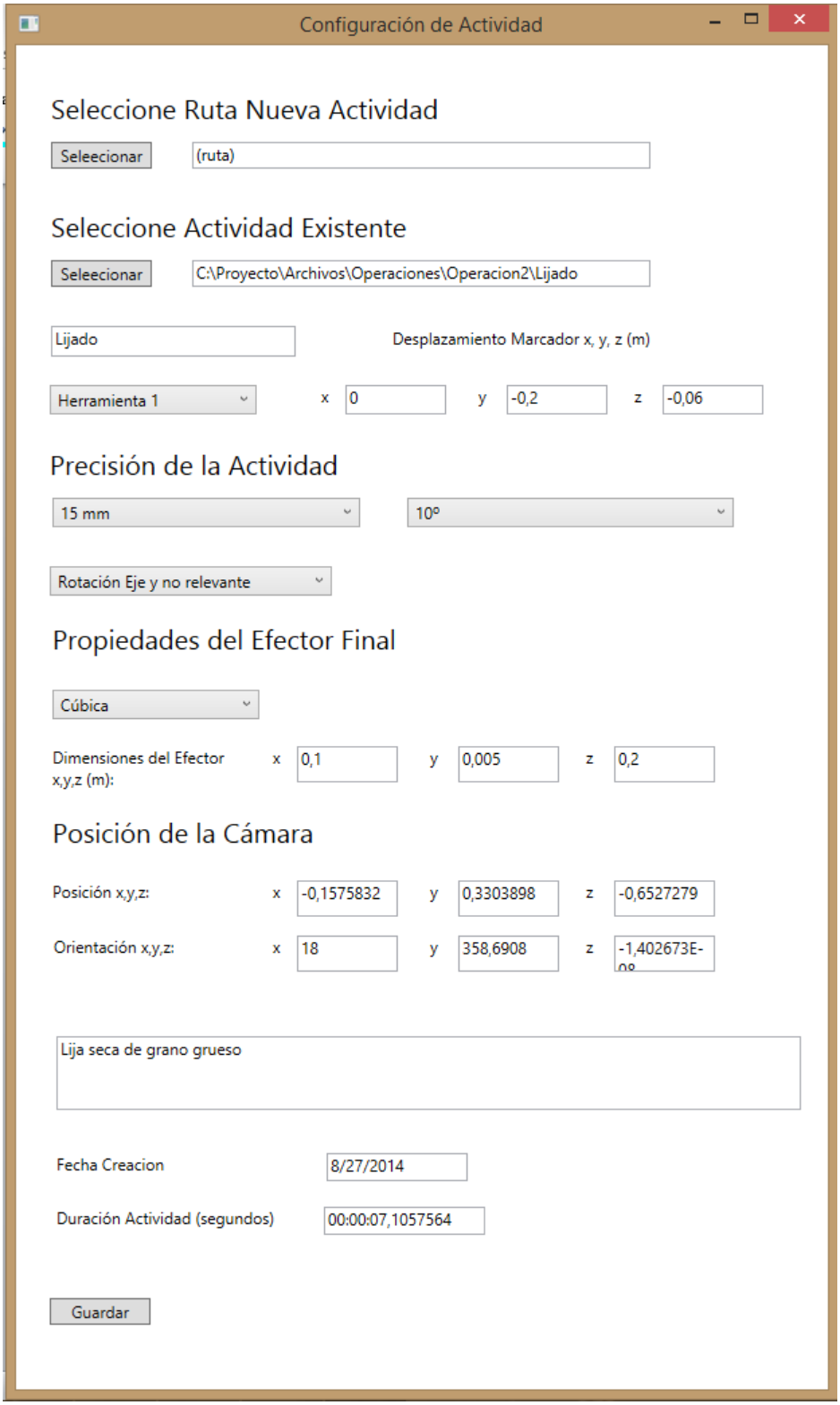


Figura 6.17 Configuración de actividad de lijado

El primer paso es seleccionar la ruta en la que almacenar la nueva actividad de lijado. La lija eléctrica utiliza los tres marcadores del tipo de herramienta 1 configurado en el apartado anterior: marcador 1 o marcador principal de color amarillo, marcador 2 de color naranja y marcador 3 de color morado.

En las siguientes imágenes se puede ver como se ha calculado el desplazamiento del marcador principal, el amarillo: sin desplazamiento en el eje x, -20 cm en el eje y, y -6cm del eje z.

Figura 6.18 Cálculo del desplazamiento

El efector final, la lija, tiene forma cúbica y unas dimensiones de 9cm de eje x, 18 cm del eje z y un espesor simbólico de 5mm.

Figura 6.19 Cálculo de la dimensión de la lija

Se elige una precisión máxima de 10 mm para la posición y de 15º para la orientación. La rotación del eje y no se tendrá en cuenta en las restricciones. Se incluye el mensaje “lija seca de grano grueso” como comentario indicativo al operario. Pulsando la tecla Guardar se crea el archivo ActividadSR en la ruta Las propiedades posición de la cámara, fecha inicial y duración de la actividad serán cambiadas en el modo 2.

- Modo 2, Almacenado de superficies lija de referencia

Al iniciar el modo 2 se debe seleccionar la ActividadSR previamente configurada en el programa Seguimiento de Marcadores. A partir de aquí se pueden realizar dos acciones. Pulsar la tecla “Inicio” para comenzar la grabación o Pulsar la tecla “N” para cambiar la posición de la cámara.

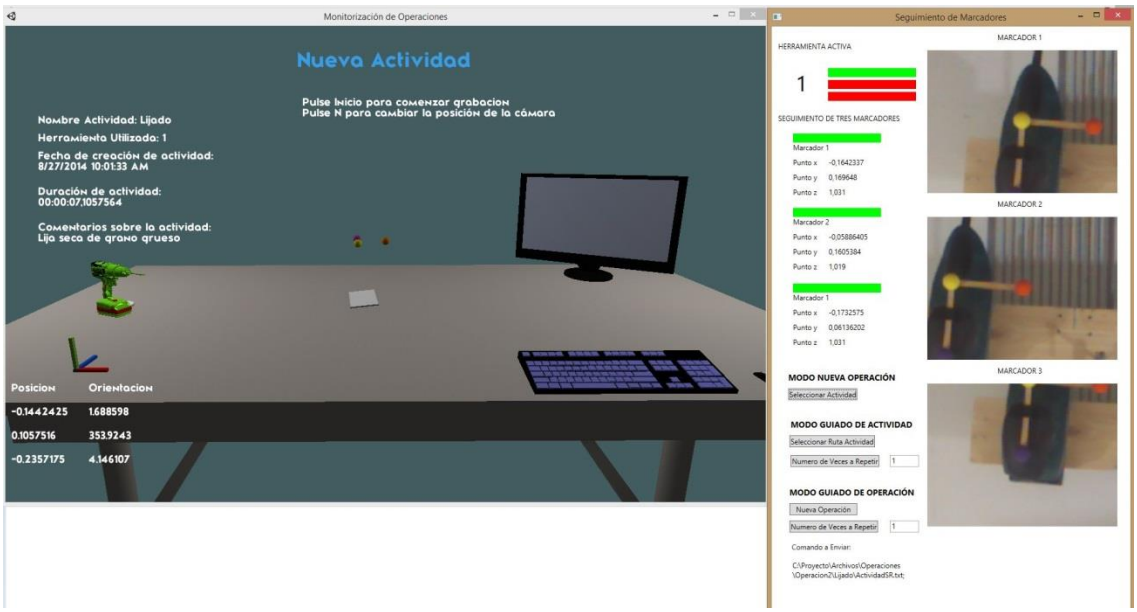


Figura 6.20 Representación inicial del modo 1

Se decide cambiar la posición de la cámara. Se debe pulsar la tecla “M” para fijar la posición de esta, quedando almacenada como propiedad de la actividad.

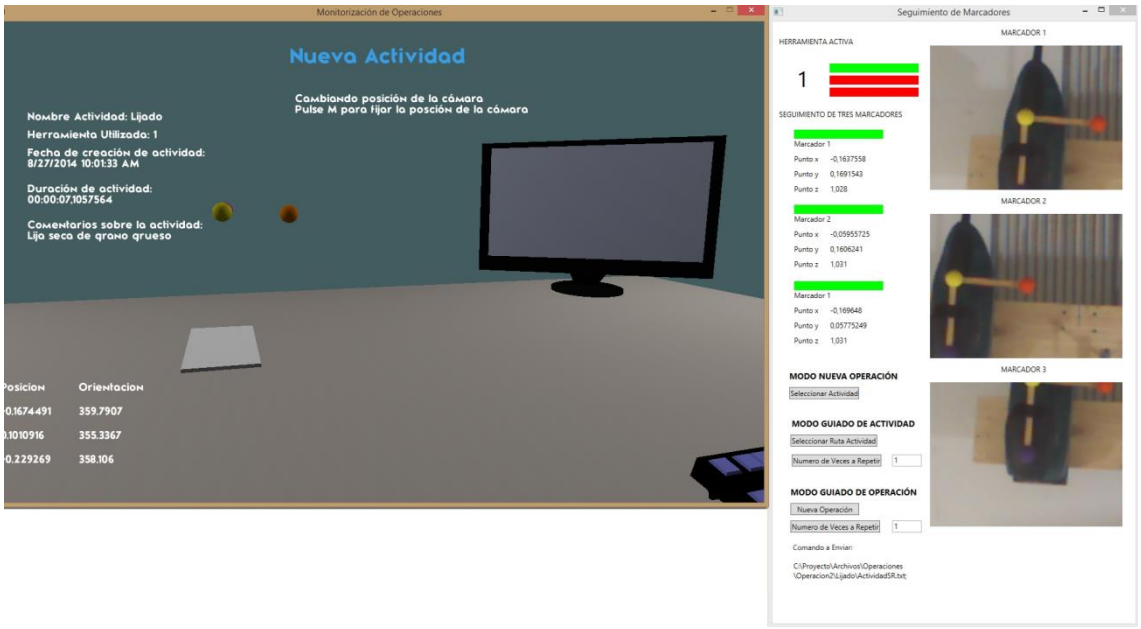


Figura 6.21 Cambiando la posición de la cámara

Se decide grabar las superficies de Referencia. En ese momento se guarda la fecha de creación de actividad como propiedad. Pulsando la tecla “Fin” se finaliza la grabación.

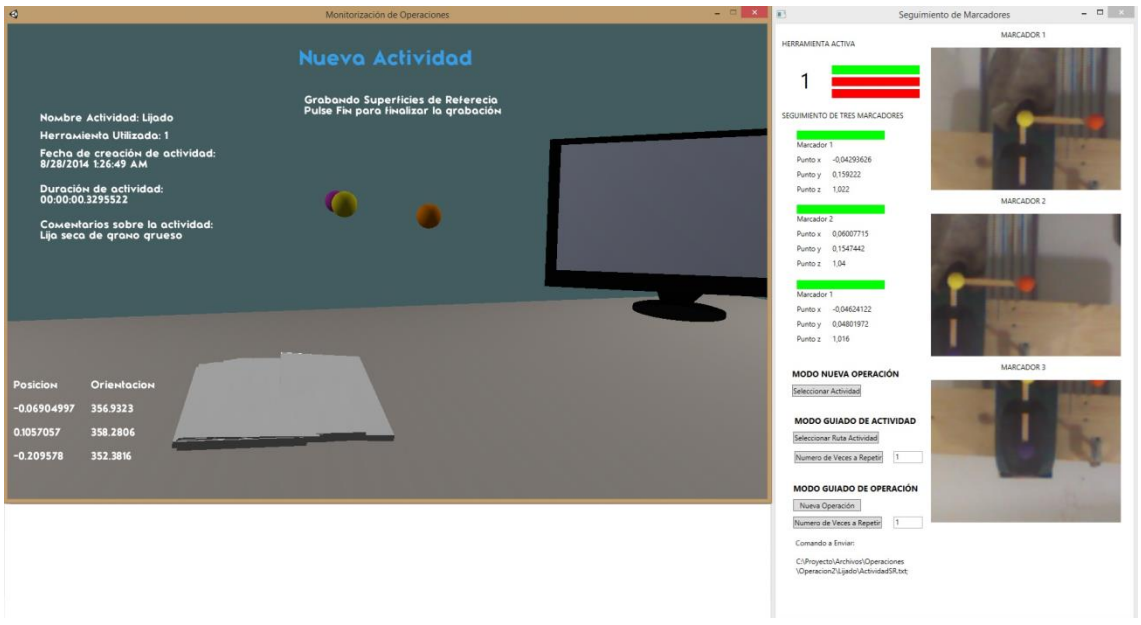


Figura 6.22 Guardando superficies de referencia

En este momento queda almacenada la duración de la actividad. Pulsando la tecla “Intro” se modifica el archivo ActividadSR.

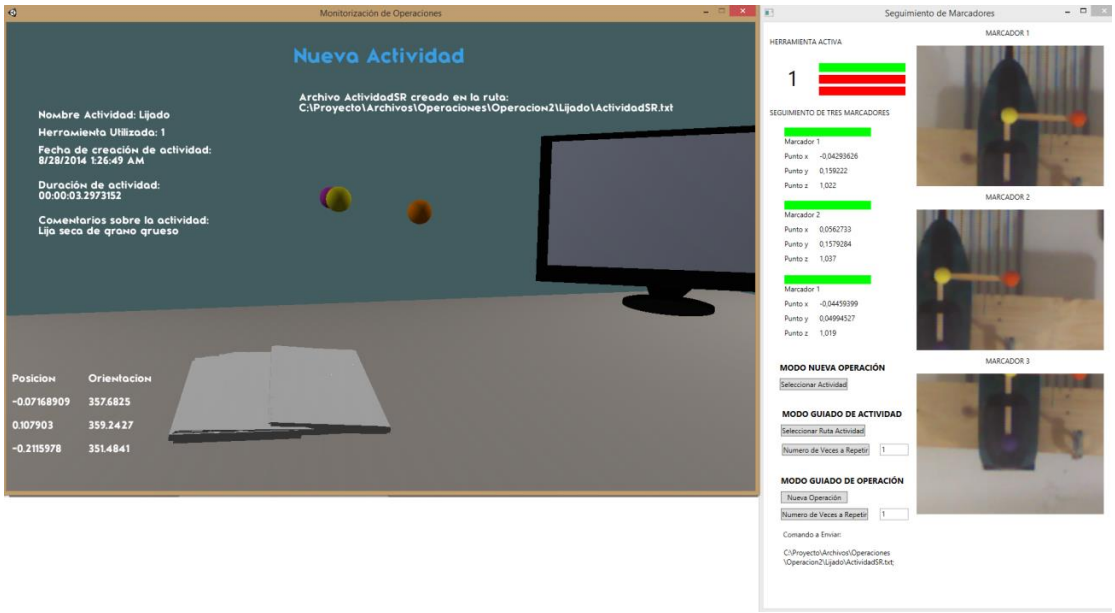


Figura 6.23 Modificación de Archivo ActividadSR.txt

- Modo 3, Modificación de actividad de taladrado

Para describir los pasos de este modo se utiliza una actividad previamente creada de taladrado.

El primer paso es seleccionar la ActividadSR en el programa Cargar Actividades. Automáticamente estas se cargan en la interfaz gráfica.

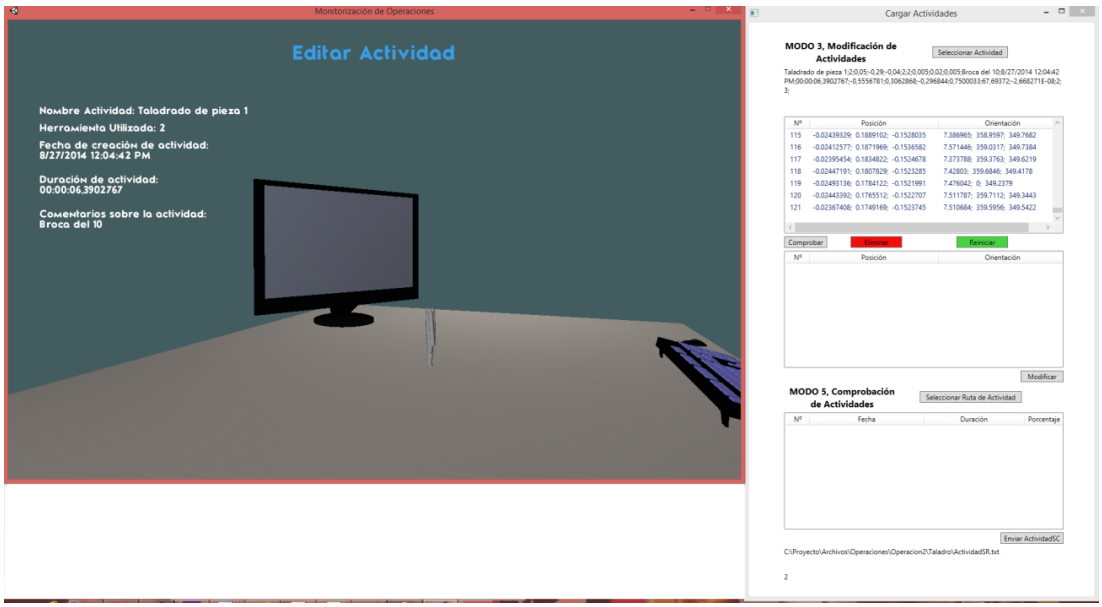


Figura 6.24 Carga de actividades

Seleccionando las superficies en la lista superior y pulsando en el botón "comprobar" cambian a color azul.

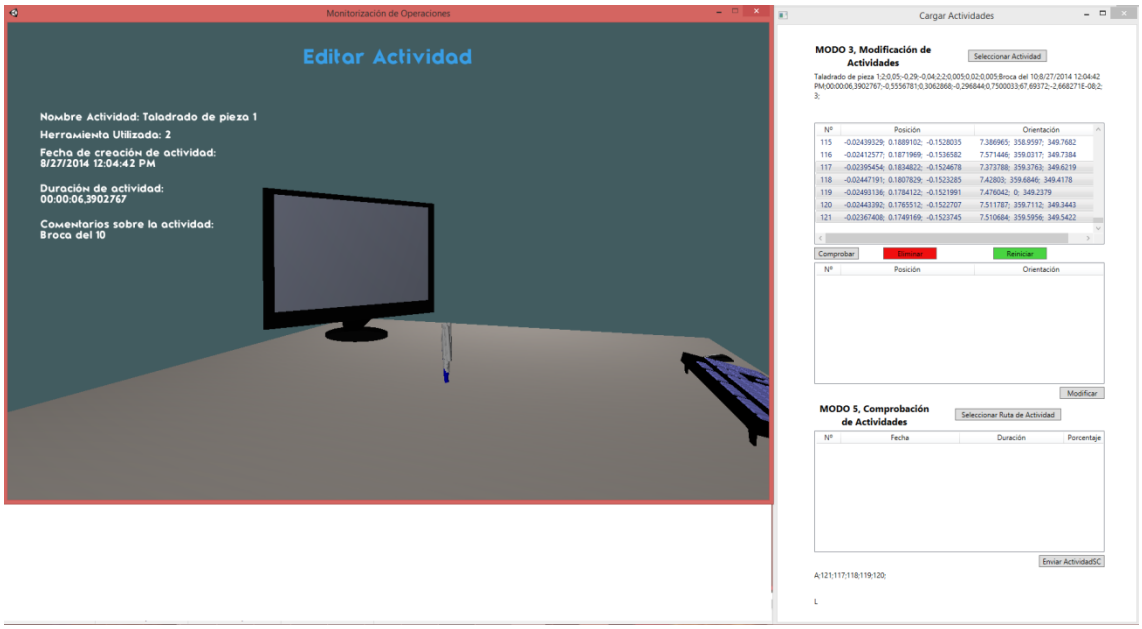


Figura 6.25 Comprobar superficies de referencia

Pulsando en el botón "Eliminar" pasan a la lista central y se colorean de color rojo.

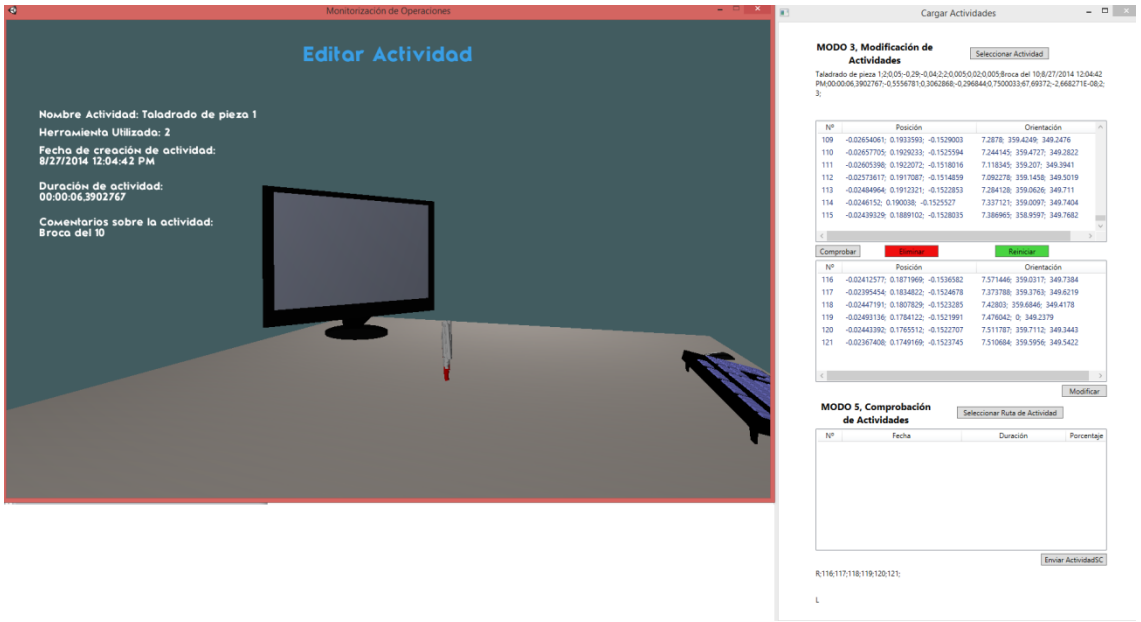


Figura 6.26 Superficies seleccionadas como superficies a eliminar

Pulsando en “Modificar” estas se eliminan del archivo ActividadSR.

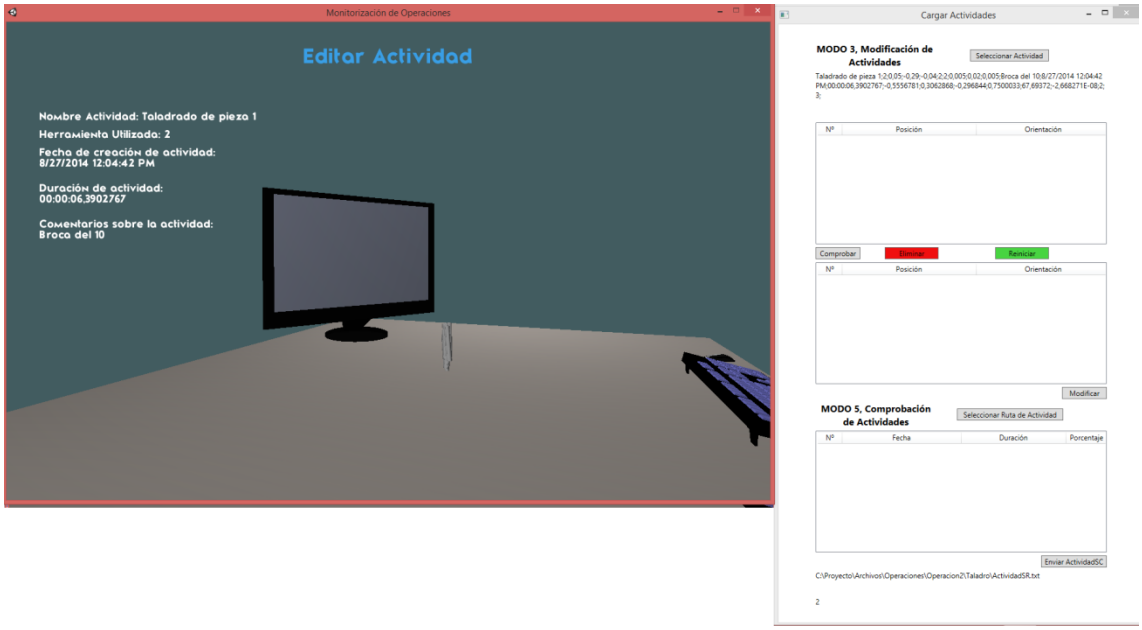


Figura 6.27 Actividad editada

- Modo 4. Guiado de Actividad de lijado

Toda la configuración de los modos previos tiene como objetivo garantizar la correcta monitorización de la actividad.

El primer paso es seleccionar la ruta de la actividad en el programa Seguimiento de Marcadores. Se cargan las superficies de referencia y sus propiedades. El sistema queda listo para el seguimiento de la actividad.

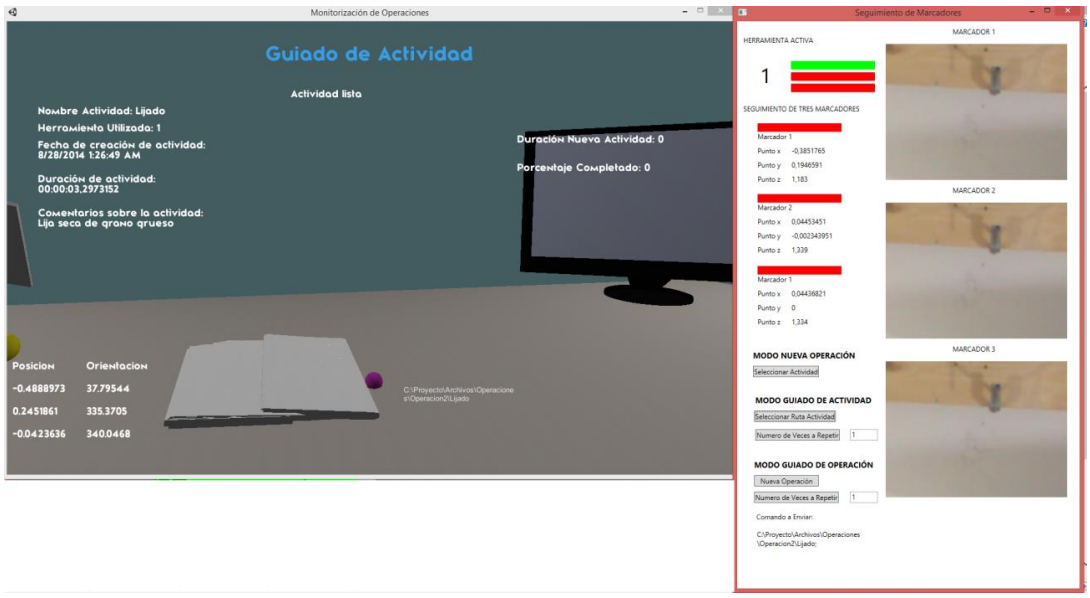


Figura 6.28 Carga de actividad en modo 4

El usuario lija la pieza mientras el sistema sigue el movimiento de del efector final de la herramienta, la lija. Se puede visualizar en todo momento el movimiento de la lija (color azul) con su posición y orientación en forma de coordenadas en metros y grados respectivamente (parte inferior izquierda de la pantalla), el porcentaje de actividad completado (superficie de color verde) respecto a lo que falta por completar (superficie de color gris), 35% en el instante de la captura de pantalla y el tiempo que se lleva realizando la actividad, 1 segundo.

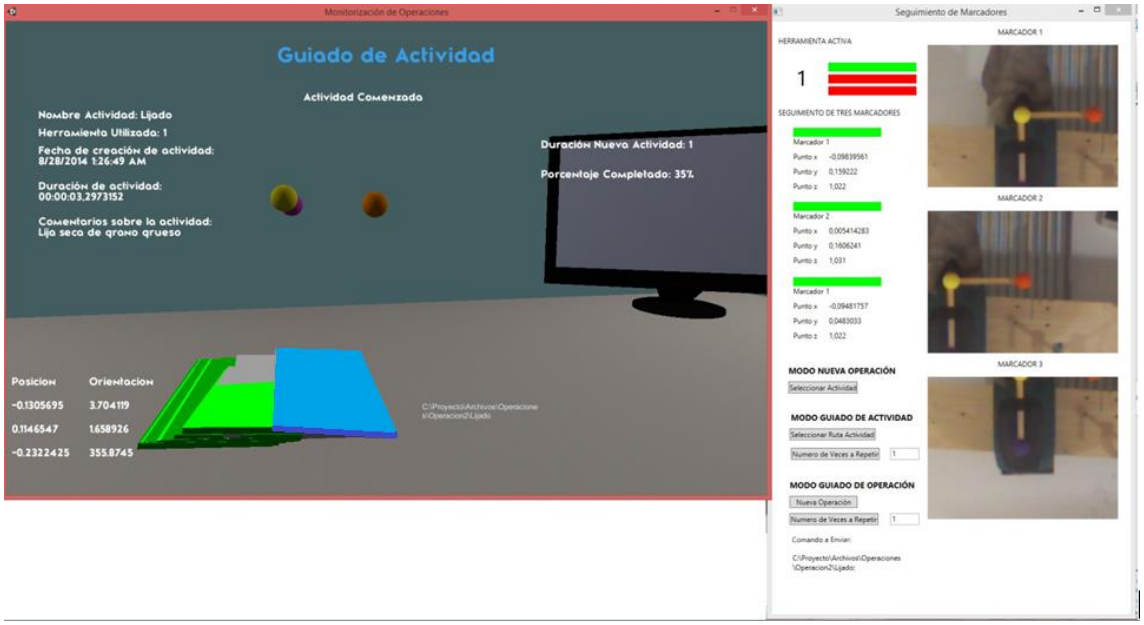


Figura 6.29 Guiando al usuario mientras realiza la actividad de lijado

Cuando se ha completado la operación se pide al usuario pulsar “Espacio” para reiniciar el guiado.

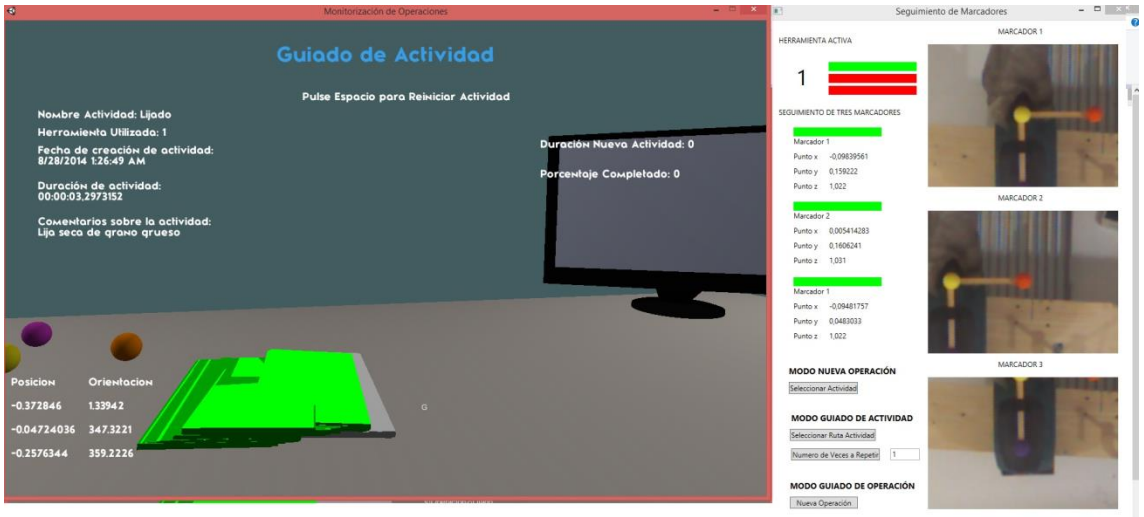


Figura 6.30 Actividad completada

- Modo 5, Comprobación de Actividades

Se utiliza para la explicación de este modo las actividades creadas en la realización del corte de la pieza con sierra de calar.

El primer paso es seleccionar la Ruta de la actividad a comprobar en el programa Cargar Actividades. Las actividades Actividad1SC y Actividad2SC se colocan en orden de creación en la lista inferior indicando además la duración de la actividad y el porcentaje cargado. Se cargan las propiedades y las superficies de referencia junto a las propiedades en el programa de Monitorización de Operaciones.

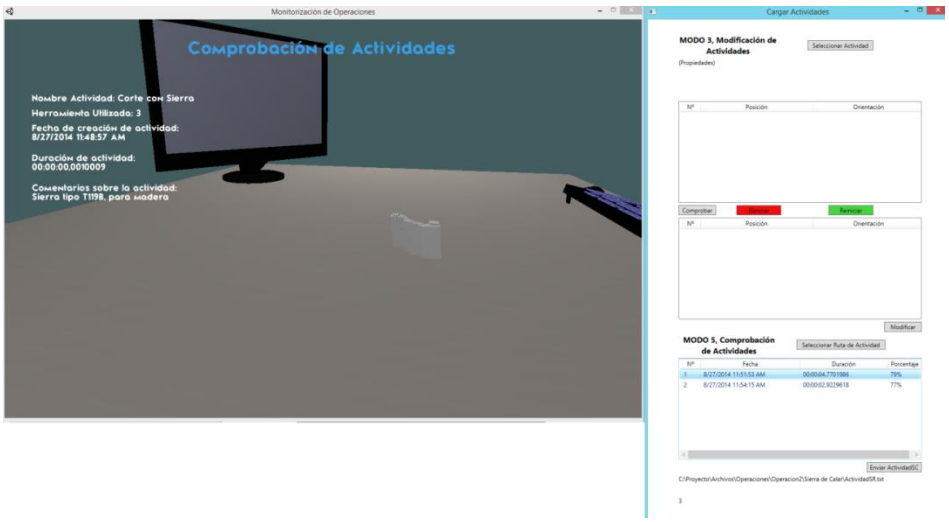


Figura 6.31 Carga de actividad en modo 5

La Actividad*SC seleccionada se dibuja en la pantalla.

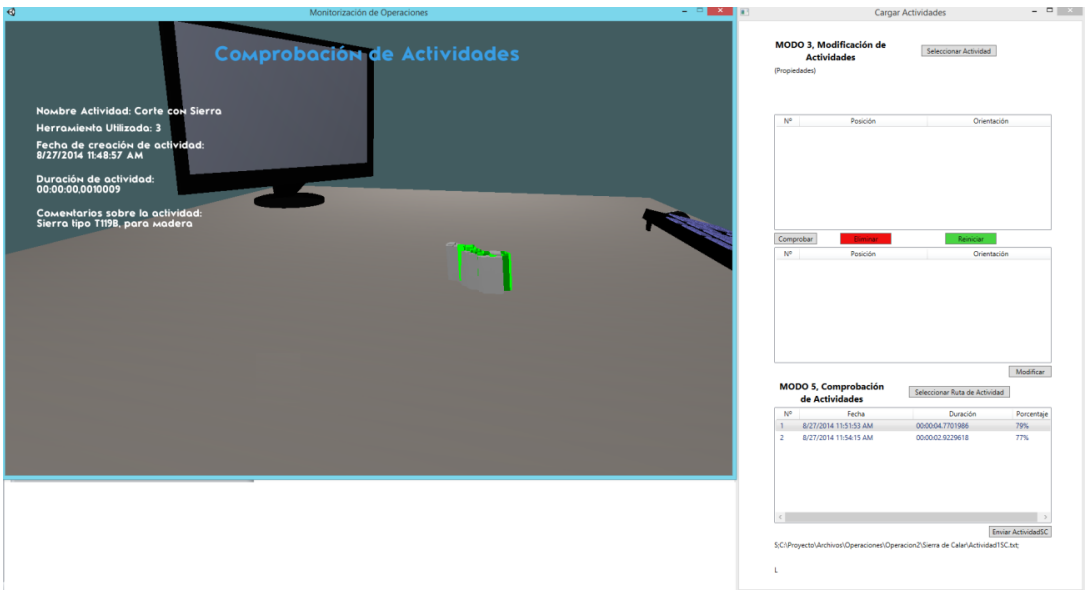


Figura 6.32 Comprobación Actividad

7. Presupuesto

En este capítulo se incluye una estimación del presupuesto de este proyecto. Se ha hecho una división por componentes necesarios y recursos humanos.

- Presupuesto de material

Existe material utilizado que es fungible, es decir, material que se agota tras el proyecto. Se utiliza por tanto la totalidad del producto en el desarrollo del proyecto.

En cuanto el material inventariable, cuyo uso será posible tras el desarrollo del proyecto, se calcula el uso del producto respecto a su vida útil, cuatro años. Como el tiempo tardado en desarrollar el proyecto ha sido de medio años, se determina el precio del producto como una octava parte de su precio en mercado.

Concepto	Cantidad	Precio/unidad	Uso del producto	Precio
Sensor Kinect	1	150€	1/8	18,75€
Portátil Lenovo G510	1	500€	1/8	62,5€
Monitor	1	150€	1/8	18,75€
Combo Teclado + Ratón inalámbrico	1	15€	1/8	1,87€
Soporte Jirafa	1	50€	1	50€
Marcadores	15	1€	1	15€
TOTAL				166,87€

Tabla 7.1 Presupuesto de Material

- Presupuesto de personal

En este segundo apartado se calcula el importe del personal. Se distinguen las horas de investigación y documentación que fueron necesarias para la preparación del proyecto. Estas horas se cobran a un valor estimado de 10 €/hora. También se incluyen las horas de diseño e implementación de las aplicaciones que se cobran a 15 €/hora. Por último, las horas de pruebas que se han tenido que realizar para comprobar el correcto funcionamiento del sistema a 10 € la hora.

Actividad	Nº horas	Precio/hora	Coste
Documentación e investigación	200	15	3000€
Diseño e implementación	350	20	7000€
Pruebas	50	10	500€
TOTAL			10500€

Tabla 7.2 Presupuesto personal

- **Gastos Indirectos**

Se incluye además los gastos indirectos:

Tipo de Coste	Precio
Gastos de secretaría	50€
Gastos de Instalación	50€
TOTAL	100€

Tabla 7.3 Gastos Indirectos

- **Presupuesto final**

Por tanto, el gasto final del proyecto es:

Tipo de coste	Precio
Coste de material	166,87€
Coste de personal	10500€
Gastos Indirectos	100€
TOTAL	10766,87€

Tabla 7.4 Presupuesto Final

8. Conclusiones y Trabajos Futuros

8.1. Conclusiones

El objetivo del proyecto era desarrollar un sistema que permitiera la monitorizar del seguimiento de operaciones manuales, implementado mediante aplicaciones de PC que ejecutasen las tareas de visión por computador requeridas para ello.

Para lograr este objetivo se han seguido una serie de fases. El primer paso ha sido la familiarización con el dispositivo Kinect. Un análisis a fondo de las características del Kinect, el estudio comparativo de los métodos de control y el manejo en programas de ejemplo han sido cruciales para el buen desarrollo posterior.

En vista a que la mayor cantidad programas analizados utilizaban c# como entorno de programación se eligió este para la implementación del sistema. Fue necesario un estudio sobre el uso de c#, debido al desconocimiento del mismo.

Con esto claro se comenzó a diseñar el sistema. Se requería localizar el movimiento de una herramienta para determinar las acciones realizadas por un operario. Como primera idea se propuso la identificación de estas herramientas por medio del uso de reconocimiento de objetos. Esto planteaba un importante problema: se debería fabricar una enorme biblioteca de herramientas para que el uso del sistema fuera posible con independencia de la actividad a monitorizar.

Con la utilización de un marcador acoplado a la herramienta esto no sería un problema. El sistema buscaría la posición espacial del marcador y a partir de ella se determinaría la posición de la herramienta. Esto simplificaba enormemente el problema por lo que se pasó a implementar una aplicación que lograse este objetivo.

El problema a estas alturas no estaba resuelto. La posición de la herramienta no era suficiente para identificar la acción cometida por ella: era necesario además determinar la orientación de esta. Como es comúnmente sabido, son necesarios tres puntos del espacio para construir un plano. Por lo tanto, con este plano bien definido es posible determinar además de la posición la orientación de la herramienta. El siguiente paso estaba claro: optimizar la aplicación para poder lograr el seguimiento de tres marcadores.

La determinación de si una operación de manual realizada con una herramienta se ha realizado correctamente tiene que ver con el conocimiento en todo momento del efector final de la herramienta. Realizar los cálculos de su posición en tres dimensiones era una tarea no inmediata. Se debían de realizar métodos específicos para este cometido que se salían del alcance de este proyecto. Se buscaron alternativas para realizar este cálculo y las soluciones estaban en la

industria del videojuego. En este momento se planteó una cuestión: ¿no sería más cómodo para el operario la visualización de estas operaciones?, ¿no se sentirá el operario menos controlado si visualiza en todo momento que es lo que realmente el sensor está obteniendo? La respuesta era obvia y mi motivación creció.

Se decide entonces implementar con la herramienta Unity un programa con el aspecto de un videojuego que represente en todo momento la posición en tiempo real del efector de la herramienta a la vez que calcula la correcta ejecución de la tarea.

El siguiente objetivo a cumplir fue la optimización de una aplicación que permita al usuario construir actividades de referencia que se utilizarán posteriormente para guiarle correctamente en la realización de las operaciones manuales. Todas las acciones realizadas con la herramienta quedarian registradas en el sistema.

Con el sistema construido el último paso ha sido la realización experimentos y ensayos para optimizar al máximo su rendimiento.

A la vista de los resultados expuestos se puede concluir que se han cumplido los objetivos, consiguiendo un sistema exacto, preciso (precisión máxima de un centímetro) y dinámico (velocidad de seguimiento máxima de 0,5 m/s).

El sistema cuenta además con una gran variedad de opciones de configuración del seguimiento, permitiendo el uso del producto en ambientes diversos. Siguiendo unos sencillos pasos el sistema queda configurado, siendo únicamente necesaria la modificación de la configuración si el ambiente de trabajo cambia.

Permite además una rápida configuración de actividades con un gran número de opciones que permiten el seguimiento de cualquier operación manual donde exista una pieza fija sobre la que se realizan trabajos de mecanizado por medio de una herramienta.

El guiado de operaciones facilita una cómoda interfaz de usuario que permite al operario el conocimiento de la correcta realización de la tarea en tiempo real. De esta forma es posible la corrección de fallos en el momento evitando arrastrar el error en tareas posteriores. Además las tareas realizadas quedan registradas de forma simplificada en archivos de texto en el sistema, pudiéndose realizar comprobaciones de las tareas. Esto permite localizar el paso que está fallando entre los múltiples pasos a realizar en la producción de una pieza, pudiéndose volver a visualizar la ejecución de la tarea y facilitando la fecha y duración de la misma. Esto se traduce en una enorme reducción de los gastos de producción del producto ya que permite un control de calidad exhaustivo en cada fase del proceso de producción.

El proyecto realizado puede servir como punto de partida para la programación por guiado de robots. El proyecto se basaría en utilizar los datos obtenidos de la posición y orientación del efector de la herramienta facilitados por este programa como los datos de entrada para la programación de las trayectorias a realizar por el efector final de la herramienta del robot.

8.2. Trabajos Futuros

Aunque el resultado del proyecto es satisfactorio, aún son posibles muchas mejoras y ajustes en general. De la versión actual se pueden realizar varias actualizaciones a nivel del software desarrollado, versiones 2.0 del producto y posteriores. La implementación se puede optimizar en varios puntos puesto que existe la versión 2.0 del sensor Kinect utilizado en este proyecto. Según la página web de Microsoft la Kinect 2.0. "Mejora la fidelidad de profundidad y reduce el nivel de ruido significativamente, dando una mejor visualización 3-D, mayor capacidad para ver objetos más pequeños y con mayor claridad". Adaptando el proyecto al nuevo sensor Kinect se conseguiría un aumento de la precisión del producto, actualmente establecida en 1 cm.

También es posible mejorar la interfaz gráfica de usuario, para hacerla más atractiva de cara al marketing del producto. Esto se sale del campo de la ingeniería, siendo una tarea de desarrolladores gráficos más que de desarrolladores de software.

.

BIBLIOGRAFÍA

[1] OpenKinect. Main Page. Visitado en marzo de 2014
http://openkinect.org/wiki/Main_Page

[2] Jarrett Webb , James Ashley, (2012). Beginning Kinect Programming with the Microsoft Kinect SDK. Apress.

[3] Abhijit Jana (2012). Kinect for Windows SDK Programming Guide. Packt

[4] Rob Miles. (2012) Start Here! Learn the Kinect API. O'Reilly Media, Inc.

[5] Foro de Soporte Kinect para Windows. Visitado en marzo de 2014
http://www.microsoft.com/en-us/kinectforwindows/

[6] Kinect for Windows Managed Reference. Visitado en marzo de 2014
http://msdn.microsoft.com/en-us/library/hh855367.aspx

[7] Kinect for Windows.Terminos de Licencia del SDK de Microsoft para Windows. Visitado en Junio de 2014. http://www.microsoft.com/en-us/kinectforwindows/develop/sdk-eula_sp-sp.aspx

[8] undemy/Blog. Comparación entre C, C++ y C#. Visitado a Marzo de 2014
https://www.udemy.com/blog/c-vs-c++-vs-c/

[9] Jeff Ferguson. (2003). La Biblia de C#. Anaya
[10] Youtube. Curso Visual C# 2012. Jesus Conde. Visitado en Junio de 2014
https://www.youtube.com/playlist?list=PLEtcGQaT56chI8fDX4d2spoaJYfPWEO-k

[11] Microsoft Developer Network. Color Basics-WPF C# Sample. Visitado en Julio de 2014. http://msdn.microsoft.com/en-us/library/hh855379.aspx

[12] Microsoft Developer Network. Depthr Basics-WPF C# Sample. Visitado en Julio de 2014. http://msdn.microsoft.com/en-us/library/hh855380

[13] Microsoft Developer Network. Infrared Basics-WPF C# Sample. Visitado en Julio de 2014. http://msdn.microsoft.com/en-us/library/jj663800.aspx

[14] 3GearSystem. Visitado a Julio de 2014. http://www.threegear.com/

[15] 3GearSystem. Nimble SDK User guide. Visitado en Julio de 2014.
http://www.threegear.com/latest/doc/

[16] 3GearSystem. Writing your own applications. Visitado a Julio de 2014.
http://www.threegear.com/latest/doc/api.html

[17] P.Gil, F. Torres (2004). "Detección de objetos por segmentación multinivel combinada de espacios de color". XXV Jornadas de Automática, Ciudad Real.

[18] EmguCV. Main Page. Visualizado en Julio de 2014. http://www.emgu.com/wiki/index.php/Main_Page

[19] OpenCV. Visualizado en Julio de 2014 http://opencv.org/

[20] Gary Bradski and Adrian Kaehler.(2008). Learning OpenCV. O'Reilly Media, Inc.

[21]. ANXframework. Main Page. Visualizado en Julio de 2014. http://anxframework.codeplex.com/

[22] Sharpdx. Home. Visualizado en Julio de 2014. http://sharpdx.org/

[23] SlimDX- What is SlimDX?. Visualizado en Julio 2014. http://slimdx.org/

[24] Unity. Main Page. Visualizado en Julio de 2014. http://unity3d.com/unity

[25] Trinit. Tutoriales Unity en Español. Visualizado en Julio de 2014 http://trinit.es/2010/10/22/tutorial-de-unity3d-en-espanol/

[26] Francisco Jurado. (2012). "3D Color-based Tracking System for Real-time using Kinect Sensor". SOMIXXVII Congreso de Instrumentación.

[27] Nakamura, T., "Real-time 3-D object tracking using Kinect sensor," Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on , vol., no., pp.784,788, 7-11 Dec. 2011.

[28] Microsoft Developer Network. CoordinateMapper.MapColorFrameToSkeletonFrame Method. Visualizado en Julio 2014. http://msdn.microsoft.com/en-us/library/microsoft.kinect.coordinatemapper.mapcolorframetoskeletonframe.aspx

[29[RENISHAW. Neuromate stereotatic robot. Visualizado a Julio de 2014. http://www.renishaw.com/en/neuromate-stereotactic-robot--10712

[30] The application accuracy of the NeuroMate robot--A quantitative comparison with frameless and frame-based surgical localization systems.Li QH, Zamorano L, Pandya A, Perez R, Gong J, Diaz F.Comput Aided Surg. 2002;7(2):90-8.

[31] Haidegger, T.; Tian Xia; Kazanzides, P., "Accuracy improvement of a neurosurgical robot system," Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS & EMBS International Conference on , vol., no., pp.836,841, 19-22 Oct. 2008.

[32] Antonio Barrientos.(2007). Fundamentos de Robótica, 2ª Edición. McGraw-Hill.

GLOSARIO

Radiación infrarroja, o radiación IR es un tipo de radiación electromagnética y térmica, de mayor longitud de onda que la luz visible, pero menor que la de las microondas. Consecuentemente, tiene menor frecuencia que la luz visible y mayor que las microondas.

RGB (en inglés Red, Green, Blue, en español rojo, verde y azul) es la composición del color en términos de la intensidad de los colores primarios de la luz

USB. El "Bus Universal en Serie" (BUS), en inglés: Universal Serial Bus más conocido por la sigla USB, es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.

Middleware es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos.

Framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Kit de desarrollo de software o **SDK** (siglas en inglés de software development kit) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, por ejemplo ciertos, paquetes de software, frameworks, plataformas, etc.

CMOS. Un Active Pixel Sensor (APS) es un sensor que detecta la luz basado en tecnología CMOS y por ello más conocido como Sensor CMOS. Gracias a la tecnología CMOS es posible integrar más funciones en un chip sensor, como por ejemplo control de luminosidad, corrector de contraste, o un conversor analógico-digital.

Frame, fotograma o cuadro es una imagen particular dentro de una sucesión de imágenes que componen una animación. La continua sucesión de estos fotogramas producen a la vista la sensación de movimiento, fenómeno dado por las pequeñas diferencias que hay entre cada uno de ellos.

Frecuencia. Es el número de fotogramas por segundo que se necesitan para crear movimiento. Se expresa en fotogramas o frames por segundo (fps) o en hercios (Hz).

Host ("anfitrión", en español) es usado en informática para referirse a las computadoras conectadas a una red, que proveen y utilizan servicios de ella. Los usuarios deben utilizar anfitriones para tener acceso a la red. En general, los anfitriones son computadores monousuario o multiusuario que ofrecen servicios de transferencia de archivos, conexión remota, servidores de base de datos, servidores web, etc.

Manejador de dispositivo o controlador de dispositivo (llamado en inglés driver o device driver), es un programa informático que permite al sistema operativo interaccionar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz -posiblemente estandarizada- para usarlo.

C: lenguaje de programación orientado a la implementación de Sistemas Operativos, popularmente usado para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Uno de los objetivos de diseño del lenguaje C es que sólo sean necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución.

C++: lenguaje de programación con la intención de extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje multiparadigma ya que admite programación estructurada y la programación orientada a objetos.

C#: lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Visual Basic: lenguaje de programación dirigido por eventos. Es un dialecto de BASIC, con importantes agregados. Su intención es simplificar la programación utilizando un ambiente de desarrollo que facilita en cierta medida la programación misma.

BASIC: siglas de Beginner's All-purpose Symbolic Instruction Code (Código simbólico de instrucciones de propósito general para principiantes), es una familia de

lenguajes de programación de alto nivel. BASIC originalmente fue desarrollado como una herramienta de enseñanza.

Programación orientada a objetos: paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. Su objetivo es hacer más sencilla la organización del código de las aplicaciones.

WPF: Windows Presentation Foundation es una tecnología de Microsoft que permite el desarrollo de interfaces de interacción en Windows tomando características de aplicaciones Windows y de aplicaciones web.

XML: es un lenguaje de marcas utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información

EmguCV. Plataforma de envoltorio cruzado .Net de la biblioteca de procesamiento de imágenes OpenCV.

OpenCV. Biblioteca libre de visión artificial originalmente desarrollada por Intel bajo licencia BSD que permite que sea usada libremente para propósitos comerciales y de investigación. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

Microsoft XNA. Conjunto de herramientas con un entorno de ejecución administrado proporcionado por Microsoft que facilita el desarrollo de juegos de ordenador y de gestión.

DirectX. Colección de API desarrolladas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

Direct3D: utilizado para el procesamiento y la programación de gráficos en tres dimensiones (una de las características más usadas de DirectX).

ANX.Framework: plataforma independiente para el desarrollo de juegos que permite desarrollar en aplicaciones .Net las herramientas Microsoft XNA.

Sharpdx: proyecto de código abierto libre y activo que desarrolla todas las funciones para manejar la API DirectX.

SlimDX: marco de código abierto que permite a los desarrolladores crear fácilmente aplicaciones DirectX utilizando .NET como C#.

Unity: motor de videojuego multiplataforma creado por Unity Technologies disponible como plataforma de desarrollo para Windows y OS X siendo su mayor virtud crear juegos para la mayoría de plataformas del mercado.

Scripts: archivo de órdenes, archivo de procesamiento por lotes o guion es un programa usualmente simple usado habitualmente para realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.

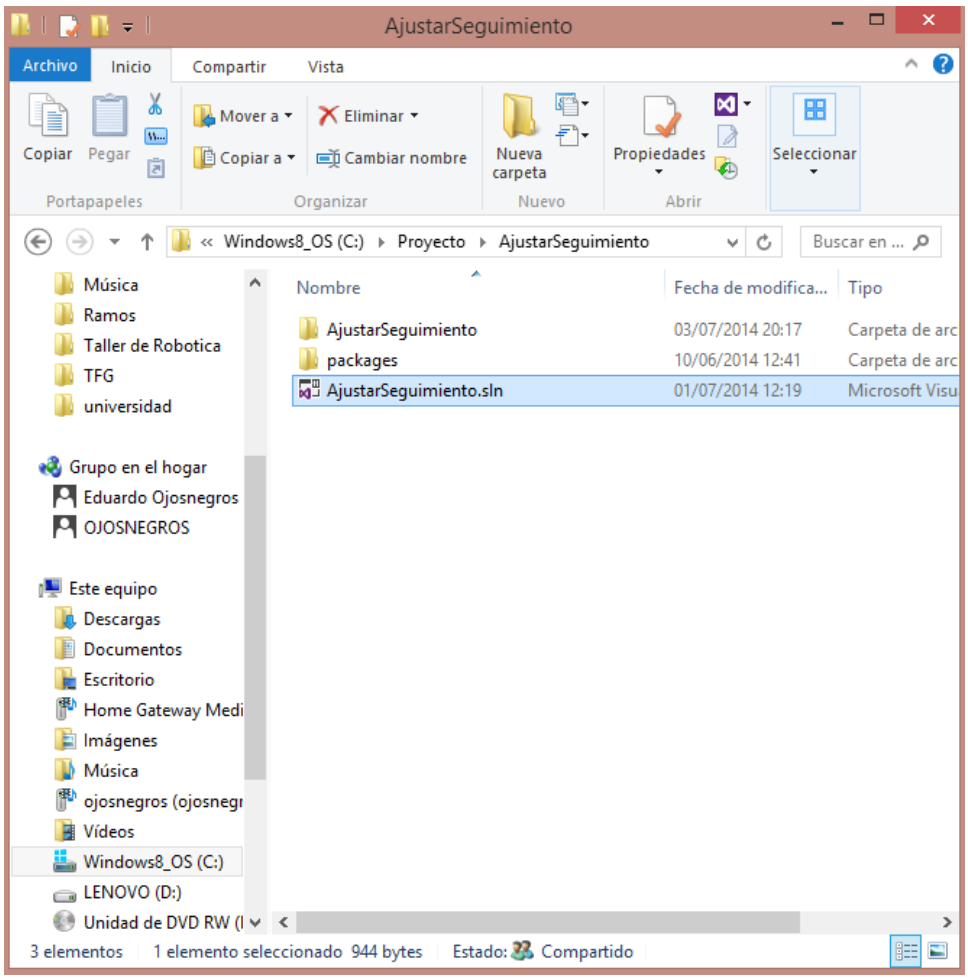
CONTENIDO DEL CD

La carpeta adicional contiene dos carpetas: el proyecto Posición de las Manos que es la aplicación desarrollada de ejemplo de NimbleSDK descrita en el apartado 2.5; la carpeta proyecto que engloba todo el contenido del sistema de monitorización de operaciones.

A su vez la carpeta proyecto contiene las soluciones en Visual Studio de las aplicaciones cliente:

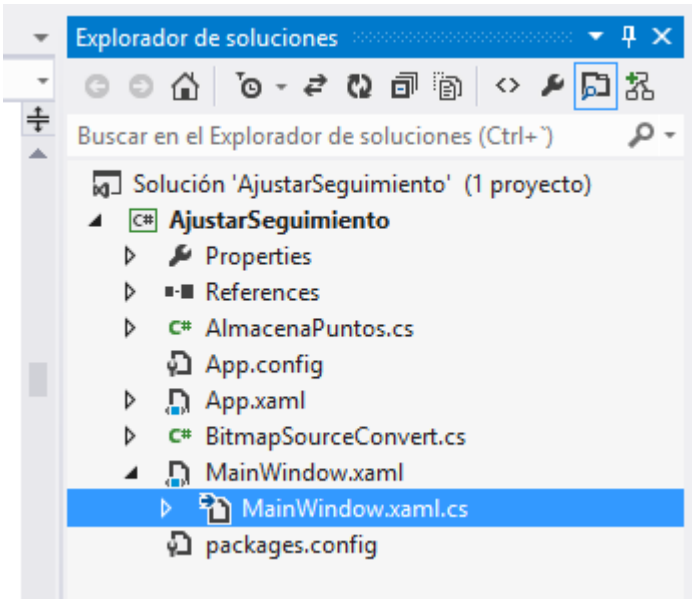
- Configurar Operaciones -> AjustarSeguimiento
- Configuración de Actividades -> ConfOperaciones
- Seguimiento de Marcadores -> SeguimientoMarcadores
- Cargar Actividades -> ModifSuperRef

Para cualquier duda sobre el código de estas aplicaciones se recomienda montar la solución de Visual Studio:



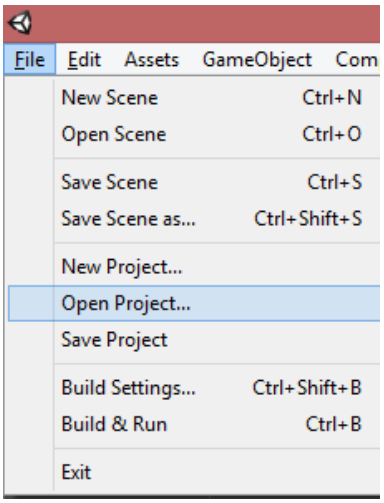
Montar solución haciendo doble click

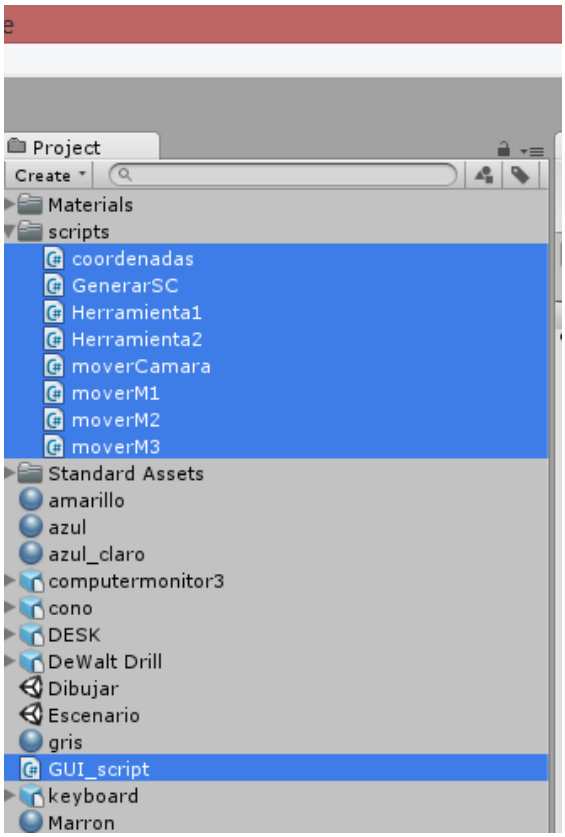
y consultar la sección que se desee:



Buscar el código a consultar en el Explorador de soluciones de visual Studio

La carpeta proyecto también contiene el proyecto de Unity DibujarSeguimiento correspondiente al programa Monitorización de Operaciones. Para consultar cualquier código se recomienda cargar el proyecto en Unity y consultar los scripts:





ANEXOS

ANEXO I: INSTALACIÓN DEL SISTEMA

Los pasos a seguir para la instalación del sistema son los siguientes:

1) Copiar la carpeta proyecto contenida en la siguiente ruta del CD.

   E: \información adicional\Proyecto

   Pegar carpeta en la unidad C:\ del ordenador

2) Crear un acceso directo en el escritorio del ejecutable:

   C: \Proyecto\DibujarSeguimiento\DibujarSeguimiento.exe

   en el escritorio. Haciendo doble click en el acceso directo creado se arranca la aplicación

3) Crear un acceso directo en el escritorio del ejecutable:

   C:\Proyecto\AjustarSeguimiento\AjustarSeguimiento\bin\Debug\ AjustarSeguimiento.exe

   En el escritorio. Haciendo doble click en él se abre el programa de Configuración de Seguimiento.

ANEXO II: DATOS DE CALIBRACIÓN

Coordenadas de los pixeles X (Pixel X) e Y (Pixel Y) y coordenadas en el eje x (M1X), eje y (M1Y) y eje z (M1Z) en metros del centro de gravedad del marcador naranja.

MX	MY	M1X	M1Y	M1Z
266	291	-0,06852508	-0,1161947	0,851
265	291	-0,07017931	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
265	291	-0,06852508	-0,1161947	0,851
265	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	290	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06852508	-0,1161947	0,851
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06852508	-0,1161947	0,851
266	290	-0,06868613	-0,1149746	0,853
266	291	-0,06884717	-0,1137475	0,855
265	291	-0,06852508	-0,1161947	0,851
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06852508	-0,1161947	0,851
265	290	-0,06868613	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
265	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853

265	291	-0,06852508	-0,1161947	0,851
265	290	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06852508	-0,1161947	0,851
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
265	291	-0,07017931	-0,1149746	0,853
265	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06852508	-0,1161947	0,851
265	290	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
265	291	-0,06852508	-0,1161947	0,851
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853

266	291	-0,07017931	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06884717	-0,1137475	0,855
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,07017931	-0,1164678	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1164678	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1164678	0,853
265	291	-0,06884717	-0,1152442	0,855
266	291	-0,06884717	-0,1152442	0,855
266	291	-0,07017931	-0,1149746	0,853
265	291	-0,07034386	-0,1152442	0,855
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1164678	0,853
266	291	-0,07017931	-0,1164678	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,07034386	-0,1152442	0,855
265	290	-0,06884717	-0,1152442	0,855
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,07017931	-0,1149746	0,853
265	290	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06884717	-0,1152442	0,855
265	291	-0,07017931	-0,1164678	0,853
266	291	-0,06884717	-0,1152442	0,855
266	291	-0,06884717	-0,1152442	0,855
265	291	-0,06884717	-0,1152442	0,855
266	291	-0,07034386	-0,1152442	0,855
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1164678	0,853
266	291	-0,06868613	-0,1164678	0,853
266	291	-0,07017931	-0,1164678	0,853
266	291	-0,07034386	-0,1152442	0,855

266	291	-0,06884717	-0,1152442	0,855
265	290	-0,06884717	-0,1152442	0,855
266	291	-0,06884717	-0,1152442	0,855
266	291	-0,07017931	-0,1164678	0,853
265	291	-0,07017931	-0,1164678	0,853
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06884717	-0,1152442	0,855
265	291	-0,07017931	-0,1149746	0,853
265	291	-0,06852508	-0,1161947	0,851
265	291	-0,07017931	-0,1149746	0,853
266	290	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06852508	-0,1161947	0,851
266	291	-0,06884717	-0,1152442	0,855
266	291	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06884717	-0,1152442	0,855
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,07034386	-0,1152442	0,855
266	291	-0,06884717	-0,1152442	0,855
266	291	-0,07017931	-0,1149746	0,853
265	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	290	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,06868613	-0,1164678	0,853
266	291	-0,06868613	-0,1164678	0,853
266	291	-0,06868613	-0,1149746	0,853
265	291	-0,07017931	-0,1149746	0,853
266	290	-0,07017931	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853
266	291	-0,06868613	-0,1149746	0,853

Desplazamiento de 0 a 10 cm sobre el eje x (M1X) permaneciendo estáticos el eje y (M1Y) y el eje z (M1Z).

M1X	M1Y	M1Z
0	0,002326412	1,329
0	0	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0	0,002326412	1,329
0	0,002326412	1,329
0	0,002326412	1,329
0	0,002326412	1,329
0	0,002326412	1,329
0	0,00231766	1,324
0	0,002326412	1,329
0	0	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0	0,002326412	1,329
0	0	1,324
0	0	1,324
0	0,002326412	1,329
0	0	1,324
0	0,002326412	1,329
0,002326379	0,002326412	1,329
0	0,002326412	1,329
0	0	1,324
0,002326379	0,002326412	1,329
0	0,002326412	1,329
0	0,002326412	1,329
0	0	1,324
0	0	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0	0,00231766	1,324
0	0	1,324
0	0,002326412	1,329
0	0,002326412	1,329
0,002317627	0	1,324
0	0	1,324
0	0	1,324

0	0,00231766	1,324
0	0	1,324
0	0	1,324
0	0,00231766	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0	0,00231766	1,324
0	0,00231766	1,324
0	0	1,324
0,002317627	0	1,324
0	0,00231766	1,324
0	0,002326412	1,329
0	0	1,324
0	0	1,324
0	0,002326412	1,329
0	0	1,324
0	0,002326412	1,329
0	0,002326412	1,329
0	0,00231766	1,324
0,002317627	0	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0	0,002326412	1,329
0	0	1,324
0	0,00231766	1,324
0	0,002326412	1,329
0	0,002326412	1,329
0	0,00231766	1,324
0	0,00231766	1,324
0	0,002326412	1,329
0,002326379	0,002326412	1,329
0	0	1,324
0	0,00231766	1,324
0,002326379	0,002326412	1,329
0	0	1,324
0	0,00231766	1,324
0	0	1,324
0	0,00231766	1,324
0	0,00231766	1,324
0	0,00231766	1,324
0	0,00231766	1,324
0	0	1,324
0	0	1,324

0	0	1,324
0	0	1,324
0	0,00231766	1,324
0	0,00231766	1,324
0	0,00231766	1,324
0	0,002326412	1,329
0	0,002326412	1,329
0,002317627	0	1,324
0	0,00231766	1,324
0	0	1,324
0	0	1,319
0	0	1,324
0	0,00231766	1,324
0	0,002326412	1,329
0	0,00231766	1,324
0,1023623	0,002326412	1,329
0,1000358	0,004652825	1,329
0,1023623	0,002326412	1,329
0,09965945	0,00231766	1,324
0,1023623	0,002326412	1,329
0,1023623	0,002326412	1,329
0,1000358	0,004652825	1,329
0,1027474	0,002335165	1,334
0,1027474	0,002335165	1,334
0,1023623	0,002326412	1,329
0,1000358	0,004652825	1,329
0,1004122	0,00467033	1,334
0,1000358	0,004652825	1,329
0,1023623	0,002326412	1,329
0,1023623	0,002326412	1,329
0,09965945	0,00231766	1,324
0,1027474	0,002335165	1,334
0,1023623	0	1,329
0,1027474	0,002335165	1,334
0,1023623	0,002326412	1,329
0,1023623	0,002326412	1,329
0,1000358	0,004652825	1,329
0,1027474	0	1,334
0,1027474	0,002335165	1,334
0,1027474	0,002335165	1,334
0,1000358	0,004652825	1,329
0,1023623	0,002326412	1,329