SISTEMA DE CAPTURA DE GESTOS PARA LA MANIPULACIÓN DE ROBOTS QUIRÚRGICOS.

JUAN DIEGO HURTADO CHAVES
ALEX ALDEMAR NASTAR GUACALES

Director: Ing. Oscar Andrés Vivas Albán

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Febrero de 2014

SISTEMA DE CAPTURA DE GESTOS PARA LA MANIPULACIÓN DE ROBOTS QUIRÚRGICOS.

Documento Final de Trabajo de Grado para optar al título de Ingeniero en Automática Industrial

JUAN DIEGO HURTADO CHAVES
ALEX ALDEMAR NASTAR GUACALES

Director: Ing. Oscar Andrés Vivas Albán

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, Febrero de 2014

Nota de aceptación: _____

_____

_____

_____

_____

_____

_____

Firma del Presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Popayán, Febrero de 2014

AGRADECIMIENTOS

Primero que todo queremos agradecerle a Dios por sus bendiciones y su compañía en cada actividad que realizamos a diario, por darnos salud y fuerza para afrontar cada problema de nuestras vidas.

Agradecemos a nuestras familias quienes creyeron en nosotros, se preocuparon por nuestro bienestar y dieron lo mejor de ellos para darnos una educación, que sin la presencia de ellos en nuestras vidas nada de esto se hubiese podido realizar, nos brindaron su amor y nos formaron como personas para poder cumplir nuestros sueños y aprender que después de cada caída hay alguien que te ayudara a levantar.

Queremos agradecer a todas esas personas importantes que nos animaron a seguir adelante, a luchar por un sueño, a no decaer y de verdad muchas gracias por aguantar tantas situaciones incomodas en esos momentos difíciles.

Agradecemos al ingeniero Oscar Andrés Vivas por sus conocimientos, creer en nuestro potencial y guiarnos en la elaboración de este trabajo.

A los diferentes docentes que apoyaron nuestra iniciativa y aportaron ideas para el cumplimiento de todos los objetivos.

Y por último agradecemos a la Universidad del Cauca por formarnos como profesionales.

RESUMEN

Las necesidades y problemas que a diario surgen en las actividades de las personas, dejan millones de preguntas sin resolver. Con la ingeniería nace la posibilidad de abordar muchas de estas problemáticas al tener la capacidad de crear e ingeniar. El área de la medicina hace parte de los diferentes temas donde la ingeniería puede llegar, se hace muy importante debido a que involucra la salud de los seres humanos. En la cirugía existe una interacción entre el médico – paciente-instrumentos donde se corre el riesgo de fracasar en cada actividad debido a su complejidad. La experiencia de los cirujanos es una de las dos claves para disminuir el porcentaje de fracaso en las cirugías ya que el conocimiento y la práctica en esta rama de la medicina otorgan seguridad al cirujano. Las herramientas que se utilizan para la realización de cada operación es la otra clave para mejorar y disminuir los errores, por tanto buenas herramientas tecnológicas facilitarían cualquier proceso. El presente trabajo está enfocado en la realización de un sistema que consiga la interacción natural entre el médico y su instrumentación, más precisamente entre un cirujano y un robot virtual laparoscópico. El sistema está compuesto por un mando de interfaz natural de usuario, una herramienta software para cirugía con robots y un procesador con el software necesario para la compilación.

La herramienta software RoboSurgery es la base utilizada para el sistema de captura de gestos. Esta cuenta con tres robots para cirugía laparoscópica, en su principio son comandados por joystick y una interface virtual que muestra al robot porta-endoscopio Hibou y al robot quirúrgico Lapbot en una sala de cirugía. Cada robot fue modelado previamente en el entorno de desarrollo integrado “Microsoft Visual C++ 2008” por ingenieros de la Universidad del Cauca. Sobre este software se realiza un acople con el dispositivo Kinect de Microsoft que actúa como interface natural de usuario para su manejo. Con este dispositivo se crea la necesidad de migrar al entorno integrado “Microsoft Visual C++ 2010” donde las librerías de funcionamiento se pueden incorporar con facilidad.

Las librerías SDK de Microsoft son kits de desarrollo software para el Kinect, se usan para la configuración, programación y manipulación del dispositivo en el entorno C++ y conforman una parte importante a la hora de darle movimiento a los robots quirúrgicos por medio de las manos del usuario. Se logra así entonces un sistema que recolecta información del cuerpo humano para la manipulación de dos robots laparoscópicos virtuales del software RoboSurgery.

ABSTRACT

The needs and problems that arise in the daily activities of people, leave millions of unanswered questions. With engineering comes the possibility to address many of these problems by having the ability to create. Medicine is part of the various issues where engineering can get and it becomes very important because it involves the health of human beings. In surgery exists an interaction between the doctor- patients –instruments where there is a risk of failure in every activity because of its complexity. Surgeons experience is one of the two keys to decrease the failure rate of surgery since knowledge and practice in this branch of medicine surgeon provides security. The tools used to perform each operation is another key to improve and reduce errors therefore good technological tools facilitate any process. This work focuses on the implementation of a system that gets the natural interaction between the doctor and instrumentation, more precisely between a surgeon and a laparoscopic virtual robot, the system comprises a natural control user interface, a software tool for surgery with robot and a processor for compiling the necessary software.

The RoboSurgery software tool is the base used for the gesture capture system, this has three robots for laparoscopic surgery, in principle controlled by joystick, and a virtual interface that shows the robot endoscope holder Hibou and the surgical robot Lapbot in an operating room. Each robot was previously modeled in the integrated development environment "Microsoft Visual C++ 2008" by engineers at the University of Cauca. This software is realized by a coupling device Kinect of Microsoft that acts as a natural user interface for handling the Lapbot robots. It was necessary to migrate to the integrated environment "Microsoft Visual C++ 2010", operation where Kinect libraries can be incorporated easily.

The Microsoft SDK libraries are developed for the Kinect applications, used for configuration, programming and manipulation of the device in the C++ environment and to make an important time to give movement to surgical robots through the user's hands part. It achieves then a system that collects information of the human body for handling two virtual laparoscopic RoboSurgery software robots.

TABLA DE CONTENIDO

Pagina

1. Introducción. 01

2. Estado del arte. 03

2.1 Robótica Quirúrgica 03

2.2 Simuladores quirúrgicos 06

2.3 Dispositivos de control de interfaces naturales 11

3. Dispositivo Kinect con RoboSurgery 19

3.1 Módulos Kinect 20

3.1.1 Reconocimiento de imágenes 20

3.1.2 Reconocimiento de voz 22

3.1.3 El motor 23

3.1.4 Especificaciones del sensor Kinect 24

3.2 Kit de Desarrollo Software (SDK) 27

3.2.1 Libfreenect (OpenKinect) 27

3.2.2 OpenNi/NITE (Prime Sence) 28

3.2.3 SDK (Microsoft) 28

3.3 Microsoft Visual Studio 29

3.4 Librerías Microsoft SDKs 1.6 30

3.5 RoboSurgery-Kinect 31

4. Resultados. 34

4.1 Simulaciones de recolección de datos 35

4.2 Componentes de la herramienta desarrollada 43

4.3 Movilidad de los dos robots Lapbots 46

4.4 Cambio de instrumento quirúrgico 48

4.5 Simulación de una cirugía en RoboSurgery 49

4.6 Movilidad del robot físico Hibou 54

5. Conclusiones y Trabajos futuros 56

5.1 Conclusiones 56

5.2 Trabajos futuros 57

6. Bibliografía. 59

Anexo A: Instalación RoboSurgery y Kinect 65

Anexo B: Manual de Usuario 80

Anexo C: Ejemplo Kinect en Visual Studio 2010 86

LISTA DE FIGURAS

Figura 1: Aesop en cirugía 4
Figura 2: Brazos Robot Zeus 4
Figura 3: Sistema Da Vinci 5
Figura 4: Black Falcón 5
Figura 5: Robot de dos brazos CISOBOT 6
Figura 6: EndoBot 6
Figura 7: Simuladores físicos 7
Figura 8: LAP Mentor 8
Figura 9: Vist-C 8
Figura 10: Vist-Lab 9
Figura 11: ProMIS 9
Figura 12: Lapsim 10
Figura 13: RoboSurgery 11
Figura 14: Wiimote 12
Figura 15: PlayStation MOVE 12
Figura 16: Leap Motion 13
Figura 17: Emotiv EPOC 13
Figura 18: WAVI Xtion 14
Figura 19: Consola y dispositivo CT 510 14
Figura 20: Carmine 15
Figura 21: Capri1.25 15
Figura 22: Creative Senz3D 15
Figura 23: Kinect 16
Figura 24: Composición del sensor Kinect 20
Figura 25: Funcionamiento del sensor Kinect 21
Figura 26: Ubicación de puntos 22
Figura 27: Micrófonos multi array 22
Figura 28: Movilidad en grados del Kinect 23
Figura 29: Rango de trabajo del sensor. 24
Figura 30: Grafica error vs distancia 25
Figura 31: Valor de profundidad del Kinect vs distancia 26
Figura 32: Resolución vs Distancia 27
Figura 33: Arquitectura OpenNI 28
Figura 34: Articulaciones del esqueleto disponibles para seguimiento 31

Figura 35: Hibou y Lapbots de RoboSurgery 32
Figura 36: Componentes del sistema de captura de gestos 35
Figura 37: Referencia de línea para la toma de datos 36
Figura 38: Línea vertical en XY 36
Figura 39: Línea horizontal en YX 37
Figura 40: Línea en profundidad YZ 37
Figura 41: Referencias circulares para la toma de datos 38
Figura 42: Trayectoria circular obtenida y deseada en el plano XY 39
Figura 43: Error cartesiano circulo XY 39
Figura 44: Trayectoria circular obtenida y deseada en el plano ZY 40
Figura 45: Error cartesiano circulo ZY 40
Figura 46: Trayectoria circular obtenida y deseada en el plano XZ 41
Figura 47: Error cartesiano circulo ZX 41
Figura 48: Diagrama en bloques de simulink MGD 42
Figura 49: Trayectoria deseada robot Lapbot 43
Figura 50: Trayectoria resultante robot Lapbot 43
Figura 51: Gesto para inicialización del sistema 44
Figura 52: Mensaje sobre instrumento cargado en el robot Lapbot derecho 44
Figura 53: Interface para selección de paciente 45
Figura 54: Asignación de botones en el control del Xbox 45
Figura 55: Cámara interna del Hibou con robots quirúrgicos en posición normal 46
Figura 56: Manos del usuario en posición normal 47
Figura 57: Cámara interna del Hibou con robots quirúrgicos en Posición de cruz 47
Figura 58: Manos del usuario en posición de cruz. 47
Figura 59: Mensaje Cambio de Instrumentos 48
Figura 60: Gesto para cambio de instrumentos 49
Figura 61: Coger vesícula con Kinect 50
Figura 62: Posición para coger de vesícula 51
Figura 63: Colocación grapa 1 con Kinect 51
Figura 64: Posición para colocar grapa 1 52
Figura 65: Colocación grapa 2 con Kinect 52
Figura 66: Posición para colocar grapa 2 53
Figura 67: Extracción de vesícula con Kinect 53
Figura 68: Posición para extracción de vesícula 54
Figura 69: Efector final del Hibou en la parte inferior. 55
Figura 70: Efector final del Hibou en la parte superior. 55
Figura 71: Efector final del Hibou en la parte derecha. 56
Figura 72: Efector final del Hibou en la parte izquierda. 56

LISTA DE TABLAS

Tabla 1: Especificaciones técnicas del Kinect 24

1. INTRODUCCIÓN

La robótica nace por la necesidad de realizar tareas que el ser humano no quiere hacer, ya que pueden ser peligrosas, aburridas, difíciles o en algunos casos imposibles si se tienen limitaciones físicas, por esta razón se necesita de alguien o de algo que realice este trabajo. Esto ha llevado al ser humano a crear instrumentos capaces de realizar diferentes tipos de labores donde existe un usuario y un “esclavo”. Los robots son una muestra clara de la tecnología creada por el ser humano, máquinas capaces de realizar tareas que exigen precisión, fuerza o que en su momento se creyeron imposibles. Un robot hace el trabajo sin aburrirse, agotarse o quejarse, además es un instrumento donde la interacción humano-máquina se hace indispensable.

La cirugía es una de las tareas de la medicina donde la robótica ha llegado; ya que realiza procedimientos quirúrgicos donde la vida humana está en riesgo y donde la perfección se hace indispensable. Con los avances tecnológicos en la instrumentación se han creado sistemas capaces de realizar gran cantidad de tareas quirúrgicas, donde la experiencia del usuario y la robustez de la herramienta se hacen importantes en cada cirugía. Entre los dispositivos desarrollados para estas actividades se encuentran: el robot para el movimiento de endoscopio Aesop, el sistema quirúrgico Da Vinci para cirugías mini-invasivas, el robot Zeus versión mejorada del Aesop, el Black Falcón robot quirúrgico teleoperado, entre otros. Cabe resaltar que la comunicación entre el usuario y estos instrumentos se da de una forma háptica o por voz.

Como se había dicho la experiencia de los cirujanos es necesaria para la realización de cualquier cirugía, es por esto que hoy en día se han desarrollado simuladores quirúrgicos virtuales y reales para la práctica de estas tareas. Son sistemas que proporcionan un ambiente quirúrgico muy parecido al real dando así experiencia al cirujano antes de intervenir a un paciente real. Se encuentran sistemas como: TraumaMan, LapTrainer, EVA y MED3X, que utilizan ambientes físicos para la práctica, por otro lado se tiene los simuladores virtuales como el LAP Mentor, Vist-C, ProMIS, lapsim y Robosurgery. La mayoría de estas herramientas se centran en la cirugía laparoscópica siendo esta una de las técnicas más utilizadas en la cirugía mini-invasiva.

Los robots y simuladores quirúrgicos no poseen interfaces naturales ya que para su manejo se utilizan sistemas de mando o dispositivos como el ratón, teclado alfanumérico, lápiz óptico, touchpad o joystick, que exigen un aprendizaje o una práctica previa del usuario para habituarse a ellos. Gracias a los avances tecnológicos en el mercado encontramos gran variedad de dispositivos que ayudan a la manipulación de sistemas de una forma natural donde se perciben los deseos del usuario y se actúa en consecuencia a ellos. Este tipo de dispositivos se utilizaron inicialmente como controladores de videojuegos y gracias a su buen funcionamiento se han implementado en sistemas informáticos de medicina. Entre los dispositivos capaces de captar y procesar la información del movimiento del cuerpo humano encontramos:

El Wiimote, PlayStation MOVE, Kinect, Leap motion, Creative Senz3D, Carmine, Capri, entre otros. Son dispositivos que se encuentran en el mercado a muy bajo costo, algunos creados en su principio para video consolas y otros para la creación de aplicaciones. En este contexto se puede ver la idea de este proyecto; la integración de un sistema robótico quirúrgico con un sistema de captura de gestos.

Este documento está dividido en seis partes en las cuales se introduce y se explica lo relacionado con la captura de gestos para la manipulación de robots quirúrgicos.

La primera parte es la introducción, en la cual se intenta contextualizar el proyecto para su mejor comprensión, en este caso una explicación de lo general a lo específico, se desea dar entonces una idea clara sobre el contenido y fin del proyecto.

En la segunda parte, se presenta el estado del arte donde se tratan los temas de robótica quirúrgica, simuladores quirúrgicos y dispositivos de control de interfaces naturales, en este último se nombra el Kinect y algunos proyectos realizados a nivel nacional e internacional con este dispositivo.

La tercera parte muestra la interacción entre el dispositivo Kinect con el software RoboSurgery. Se comienza con la historia del Kinect, la explicación de sus funciones y módulos, especificaciones técnicas y la forma de cómo se puede utilizar, es decir las diferentes librerías que existen para la programación y creación de aplicaciones. Seguido de esto, se nombran algunas herramientas utilizadas para el funcionamiento del Kinect con RoboSurgery, Microsoft Visual

Studio como plataforma de programación y las librerías de Microsoft SDKs 1.6 como kit de desarrollo. Por último se menciona la herramienta RoboSurgey y algunas funciones que se usan para la manipulación de este sistema con el Kinect.

Los resultados obtenidos al realizar el proyecto se describen en la cuarta parte, se muestran las simulaciones, pruebas, trayectorias y datos de error del sistema. En esta parte se miran los nuevos componentes de la interface y el movimiento de los robots del software RoboSurgery, estos movimientos se producen por la captura de datos de las manos del usuario, utilizando el Kinect.

En las dos últimas partes se presentan las conclusiones, trabajos futuros y bibliografía.

2. ESTADO DEL ARTE

2.1 Robótica quirúrgica

La cirugía mini-invasiva (laparoscopia), reporta uno de los primeros casos de inspección abdominal en un perro a inicios del siglo XX. Desde entonces, se han desarrollado técnicas para mejorar estos procedimientos y aplicarlos en otros campos de la medicina. Teniendo en cuenta las ventajas en una intervención laparoscópica, durante el procedimiento quirúrgico y después de éste, se han desarrollado diferentes sistemas robotizados, de los cuales se presenta un pequeño resumen**.**

Aesop es un robot diseñado para el movimiento del endoscopio, este fue el primer robot en ser aprobado por la Administración de Alimentos y Medicamentos (FDA) de los Estados Unidos, para uso clínico en abdomen, permitiendo al cirujano liberarse de controlar un brazo adicional del robot, ya que se controla por la voz (figura 1) [1].

Figura 1. Aesop en cirugía [1].

Tres brazos robóticos conforman el robot Zeus, el primero de ellos es un robot Aesop que controla el endoscopio por medio de comandos de voz y los dos siguientes son brazos que simulan los movimientos de las extremidades superiores del cirujano (figura 2) [2].

Figura 2. Brazos Robot Zeus [3].

A finales de los noventa, incursiona el sistema quirúrgico Da Vinci, uno de los desarrollos más importantes en la robótica quirúrgica mini-invasiva. Este sistema consta de una camilla con 4 brazos robóticos interactivos, manipulados desde una consola ergonómica y un sistema de visión 3D (figura 3) [4].

Figura 3. Sistema Da Vinci [4].

El Black Falcón es un instrumento quirúrgico teleoperado para cirugía mínimamente invasiva, surgió experimentalmente en Estados Unidos por Akhil J. Madhani en el Massachusetts Institute of Technology, con 8 grados de libertad y con algunas mejoras respecto a instrumentos similares. Esta herramienta fue un gran avance para la cirugía laparoscópica al contar con un sistema de realimentación de fuerzas, una de las grandes deficiencias en este tipo de robots (figura 4) [5].

Figura 4. Black Falcón [5].

CISOBOT es un asistente robótico para cirugía mínimamente invasiva desarrollado en la Universidad de Málaga por el Instituto Andaluz de Automática Avanzada y Robótica (IAR). Este sistema es capaz de comprender la voz y los gestos del cirujano para realizar diferentes maniobras, ya que cuenta con un reconocedor de voz, una cámara y dos brazos robóticos, obteniéndose un sistema robótico autónomo (figura 5) [6].

Figura 5. Robot de dos brazos CISOBOT [6].

El EndoBot inicialmente se creó con propósitos investigativos en el Rensselaer Polytechnic Institute, de Nueva York. Este es un robot ligero con 4 grados de libertad capaz de realizar tareas quirúrgicas como costura, agarrare, disección, corte y perforación (figura 6) [7].

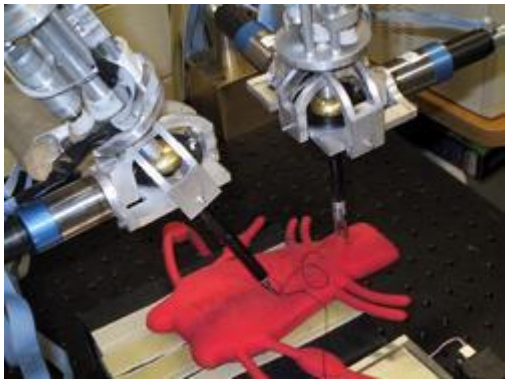


Figura 6. EndoBot [8].

2.2 Simuladores quirúrgicos

Los grandes avances tecnológicos aplicados a la robótica médica han impulsado la cirugía laparoscópica como una de las técnicas más utilizadas en la cirugía mínimamente invasiva [4]; sin embargo se evidencia la necesidad de los cirujanos en obtener un entrenamiento y una experiencia hacia este tipo de procedimientos [9] y [10].

En este sentido se pueden encontrar ambientes físicos o virtuales para desarrollar habilidades y destrezas. En el ambiente físico aparecen diversos tipos de sistemas, los cuales simulan el torso del cuerpo. Las cavidades torácicas se pueden encontrar en forma de cajas abiertas y/o cerradas, en éstas se ven implementados diversos tipos de sistemas de adquisición de imagen, los cuales además de dar una vista clara del interior de la cavidad torácica, facilitan el seguimiento de la rutina de entrenamiento. Simulab Corporation bajo la filosofía de reducción de errores médicos y el aumento de la seguridad en el paciente, ha desarrollado diversos simuladores como es el caso de: TraumaMan [11], y LapTrainer [12]. Pro Delphus ha desarrollado el modulo laparoscópico de simulación EVA [13]. Medical Simulator ha contribuido notablemente en este tipo de desarrollo, entre sus aplicaciones podemos encontrar: MATT (*minimal acces therapy trainer*) [14], entrenador laparoscópico MED3X [15] y MED4X [16] (figura 7).

Figura 7. Simuladores físicos [11] - [16].

Los ambientes virtuales constituyen el segundo grupo de simuladores quirúrgicos. Estos tienen una gran importancia en el entrenamiento, ya que permiten la adquisición de habilidades hápticas y visuales en aplicaciones quirúrgicas.

Hoy en día se pueden encontrar diversos desarrollos comerciales como también de carácter académico. A partir de la gran variedad de desarrollos resaltamos los siguientes:

LAP Mentor ™ de Simbionix, que se destaca entre los simuladores quirúrgicos mini invasivos por proporcionar una solución completa en distintas disciplinas: ginecología, urología, cirugía bariátrica y cirugía de colon y recto. Para ello implementa 7 procedimientos laparoscópicos en cerca de 60 escenarios [17] y [18], con estas características es posible perfeccionar las habilidades en este tipo de procedimientos, además de ser un sistema con diseño avanzado y ergonómico (figura 8).

Figura 8. LAP Mentor [17].

La compañía Mentice hace un aporte importante en el campo médico con sus productos Vist-C (figura 9) [19] y Vist-Lab (figura 10) [20]; estos dos sistemas están equipados con un motor de simulación y un maniquí de cuerpo completo; los cuales permiten que el médico adquiera habilidades y formación personal y grupal.

Figura 9. Vist-C [19].

Figura 10. Vist-Lab [20].

El simulador quirúrgico CAE ProMIS permite interactuar con modelos físicos o virtuales y es usado para el entrenamiento de cirujanos. Este sistema basa su tecnología en métodos para videojuegos de realidad virtual, seguimiento inteligente y análisis de imagen (figura 11) [21].

Figura 11. ProMIS [21].

Un sistema de realidad virtual llamado Lapsim utiliza tecnología avanzada en 3D para recrear con exactitud un entorno de trabajo, es idóneo para practicar cirugías simples o avanzadas y actúa como un programa de capacitación donde el aprendiz elige el nivel de complejidad de la cirugía (figura 12) [22].

Figura 12. Lapsim [22].

Pasando al campo investigativo nacional, encontramos proyectos en fases avanzadas, como es el caso de la EAFIT de Medellín, donde se desarrolló un simulador laparoscópico basado en realidad virtual, en el cual el cirujano pude hacer movimientos en los instrumentos y en el laparoscopio, además de manipulación y agarre de órganos humanos que se encuentran en tres dimensiones [23].

También se puede encontrar el prototipo de la Universidad del Cauca, desarrollado por el grupo de Investigación en Automática Industrial, el cual está integrado por el robot quirúrgico LapBot en un ambiente virtual, el cual es manipulado mediante interfaces hápticas y ayudado del motor gráfico Ogre3D [24].

Otro trabajo importante realizado en la Universidad del Cauca es la herramienta RoboSurgery, este sistema virtual usa la técnica para cirugía mínimamente invasiva llamada laparoscopia haciendo uso de tres robots, el primero de ellos el robot virtual porta endoscopio Hibou y los dos siguientes los robots quirúrgicos LapBot, codificado en lenguaje C++ en la plataforma “Microsoft Visual Studio 2008” y con ayuda de otras herramientas informáticas llega a ser un excelente simulador virtual quirúrgico (figura 13) [25].

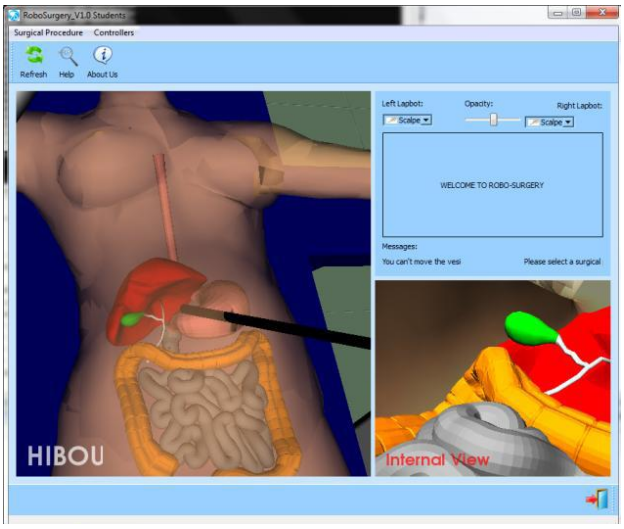


Figura 13. RoboSurgery [25].

2.3 Dispositivos de control de interfaces naturales

La interface natural de usuario se refiere a la forma de comunicación entre humanos y máquinas, donde se percibe de forma natural los deseos del usuario y se actúa en consecuencia a ellos. Esto se realiza sin tener un contacto directo con algún dispositivo de control artificial, dándose la comunicación por medio de gesticulaciones o movimientos específicos, o por voz. Dispositivos con interfaz natural de usuario son utilizados como controladores de videojuegos, aplicaciones educativas, modelado 3D, entre otros usos innovadores. Entre los diversos tipos de dispositivos que existen para la comunicación natural con el usuario encontramos:

Wiimote: Es un dispositivo de control inalámbrico para la consola Nintendo Wii que se comunica por bluetooth, consta de doce botones de los cuales cuatro están designados al direccionamiento y el resto distribuidos en el mando para otras acciones. Lo que hace a este dispositivo interesante es su capacidad de captura de movimiento por medio de un acelerómetro el cual detecta el movimiento en cualquier dirección, mientras que sus cámaras infrarrojas con ayuda de sensores obtienen información sobre posiciones y distancias según el mando. Es posible realizar aplicaciones mediante el uso de librerías no oficiales, codificadas en un ambiente de programación, gracias a que existen drivers para la conexión del dispositivo a un computador (figura 14) [26].

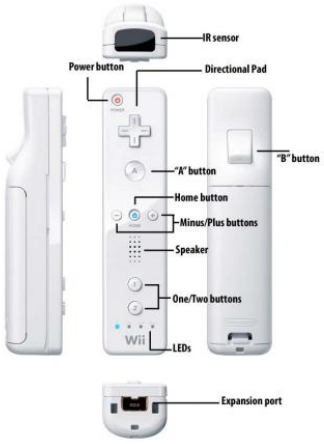


Figura 14. Wiimote [26].

PlayStation MOVE: Dispositivo para la consola de video juegos PlayStation 3 de Sony, se comunica vía bluetooth con la consola, y mediante sensores de inercia, acelerómetros y un magnetómetro se logra realizar el seguimiento del mando en todo momento. Su funcionamiento es muy similar al del Wiimote pero hasta el momento no se conocen librerías no oficiales para la realización de aplicaciones libres (figura 15) [27].

Figura 15. PlayStation MOVE [27].

Leap Motion: El control de movimiento para la comunicación humano-máquina de interface natural Leap, es un pequeño dispositivo USB que se ubica encima de una superficie plana o frente al computador con el objetivo de captar el movimiento de dedos o algún objeto con dimensiones parecidas. Los elementos que atraviesen el área de trabajo serán detectados por el dispositivo y su acción se verá reflejada en pantalla. Su espacio de trabajo tridimensional es de 242,84 cm cúbicos con un área aproximadamente semiesférica y se logra gracias a dos cámaras de infrarrojos y 3 leds. El costo de este dispositivo es muy bajo, su

sistema universal y sencillo va a permitir a millones de desarrolladores crear aplicaciones fácilmente (figura 16) [28].

Figura 16. Leap Motion [28].

Emotiv EPOC: Es un dispositivo con interface natural cerebro-máquina, se considera el primer controlador basado en reconocimiento neuronal, su objetivo es interactuar con video-juegos por medio del pensamiento consciente. La tecnología de este sistema procesa señales eléctricas que son reconocidas en diferentes puntos del cerebro por electrodos (electro-encefalografía no invasiva). Esta señal se envía al computador donde se interpretará como una expresión, acción o gesto del personaje del video juego. Cuenta con 14 electrodos y 1 giroscopio. Emotiv System ofrece kits de desarrollo de software con sus respectivas librerías para el desarrollo de aplicaciones (figura 17) [29].

Figura 17. Emotiv EPOC [29].

Wavi-Xtion: ASUS WAVI Xtion es una herramienta para ordenador que se puede manipular por gestos, está conformado por un sensor de movimiento y un dispositivo que se encarga de llevar la información del ordenador a una pantalla (televisor) sin necesidad de cables. La tecnología de WAVI es WHDI (*Wireless Home Digital Interface*) y la de Xtion está basada en infrarrojos (figura 18) [30] .

Figura 18. WAVI Xtion [30] .

Eedoo CT510: Es la primera consola de juegos creada en China por Lenovo, no es necesario el uso de alguna palanca de mando o controles para su manipulación ya que posee un sensor de movimientos 3D muy similar al Kinect de Microsoft. Eedoo DepthSense es la cámara desarrollada por SoftKinetic y Namuga, creada bajo la tecnología del sensor Kinect la cual proporciona al sistema una interfaz natural de usuario (figura 19) [31].

Figura 19. Consola y dispositivo CT 510 [31] .

Carmine y Capri: Sensores creados por la empresa PrimeSense expertos en el área de la información sensorial por detección 3D. Entre los dispositivos encontramos el Carmine 1.08 y el 1.09 que poseen un Campo de visión de 57,5 × 45 y una conexión USB (figura 20). Son sistemas para la interacción natural humano máquina o máquina entorno y se diferencia uno de otro por su rango de trabajo; el primero con un alcance de 0.8 metros a 3.5 metros y el segundo de 0.35 metros a 1.4 metros. El sensor Capri 1.25 es un dispositivo como el Kinect pero 10 veces más pequeño, considerado como un chip de detección 3D, ideal para usarlo con tabletas, televisores, PC, teléfonos móviles, ordenadores portátiles y en la robótica (figura 21) [32].

Figura 20. Carmine [33].

Figura 21. Capri 1.25 [34].

Creative Interactive Gesture Camera: Conocida como Senz3D es una cámara diseñada para procesadores de segunda generación o posteriores, donde el usuario podrá interactuar con el equipo de una manera natural, usa tecnología avanzada combinando una cámara, un micrófono de doble matriz y sensores de profundidad. La tecnología de detección 3D es capaz de reconocer manos, dedos y rostro para controlar el sistema o diversos tipos de juegos (figura 22) [35].

Figura 22. Creative Senz3D [35].

Sensor Kinect: Sensor creado por Microsoft para la consola Xbox 360, capaz de reconocer gestos, comandos de voz, objetos e imágenes, permite al usuario controlar la videoconsola sin la necesidad de un contacto físico, esta comunicación

se hace mediante una interface natural de usuario (NUI). En el año 2011 se lanzó una versión para PC a través de Windows 7 y Windows 8, con una interfaz de programación básica para el desarrollo de aplicaciones utilizando el dispositivo. El sensor consiste en una barra horizontal conectada a una base rotatoria y diseñado para estar en una posición horizontal, posee una cámara RGB, un sensor de profundidad o infrarrojo y múltiples micrófonos (figura 23) [36].

Como se comentó anteriormente, Microsoft dejó de forma gratuita y a disposición de todo el mundo los drivers y librerías especiales SDK para el control del dispositivo mediante un ordenador, permitiendo el desarrollo de todo tipo de aplicaciones. Por tal razón se mostrarán a continuación algunos proyectos en el mundo que han hecho uso del dispositivo Kinect.

Figura 23. Kinect [36].

En la Communication University de China, se llevó a cabo el proyecto de investigación “reconocimiento de gestos de la mano usando Kinect” en el año 2012. El proyecto consta de tres partes: detección de mano, identificación de dedos y reconocimiento de gestos. La adquisición de la información del sensor Kinect es realizada con ayuda de las librerías de OpenNI para ser procesadas. Para probar la robustez del sistema se tomaron 100 muestras, 50 con la mano izquierda y 50 con la mano derecha, formando 9 gestos [37].

En la Freie Univesität Berlin, de Alemania, en el proyecto “*Sing Language Recognition with Kinect*” se construyó un software para el reconocimiento de gestos utilizando el sensor Kinect. El sistema de reconocimiento de gestos identifica gestos de lenguaje mediante los modelos ocultos de Markov *(hidden Markov models*) los cuales permiten el reconocimiento y entrenamiento del

software. El sistema muestra una eficiencia mayor al 97% cuando está presente más de una persona formando las 8 o 9 señas establecidas [38].

La facultad de sistemas de la Universidad Technische Wien en Austria, hace uso del algoritmo KinectFusion para la reconstrucción en 3D dentro del *framework reshade* con el fin de lograr un modelo virtual para uso en diversas aplicaciones [39].

La Universidad de Noruega de Ciencia y Tecnología presenta en un trabajo de maestría un enfoque innovador en el reconocimiento de gestos. En este caso se hace uso de un sensor Kinect y librerías OpenNI para crear un reconocedor de movimientos que observa y analiza las acciones humanas con el fin de ser repetidas por robots [40].

En varias universidades de España se ha trabajado con el sensor Kinect. Específicamente en diciembre de 2012 la Universidad Politécnica de Valencia utilizó este sistema de captura de imagen para la realización de una aplicación que le permite al usuario resolver un juego de puzzle mediante una interface de usuario natural. Utilizan por lo tanto el sensor Kinect y da la opción de manejar el juego por medio del teclado y el mouse. Todo el desarrollo de esta interface se realizó desde cero, lo cual necesitó del uso de otras varias herramientas y librerías para culminar toda la aplicación. Entre los temas que se encuentran está el modelado de objetos en 3D, controladores o drivers para el funcionamiento óptimo del Kinect y librerías de gráficas y de gestión de elementos en tercera dimensión [41].

La Universidad Cardenal Herrera de España, ha hecho uso de un programa combinado de fisioterapia convencional y Kinect que facilita la terapia a los pacientes con parálisis cerebral, debido a que este tipo de rehabilitación es un poco tediosa. Se hace uso de la realidad virtual, método muy eficaz en actividades médicas fisioterapéuticas [42].

En la ciudad de Barcelona en el año 2011 dos ingenieros de la Universidad Politécnica de Cataluña hicieron uso del Kinect para el guiado gestual de un robot humanoide. Este proyecto consistía en el control de los brazos de un robot con extremidades muy similares a las de un humano. Se realizaron una serie de tareas entre las cuales cabe resaltar el estudio de la resolución y precisión de los datos ofrecidos por Kinect, la incorporación de librerías específicas para conocer el

esqueleto cómo funciona y qué datos proporciona, a la vez la unión del sistema de captura de imagen (Kinect) con el robot humanoide [43].

En América existen varios proyectos donde hacen uso de la captura de imagen para diferentes finalidades. Se encuentra entonces en Argentina un estudio sobre el funcionamiento del sensor Kinect para aplicaciones de bioingeniería. El sensor permite adquirir las imágenes en el espectro visible y con los datos proporcionados por sensores infrarrojos armar un plano tridimensional de los objetos [44]. En un artículo publicado por la Universidad de Missouri, Estados Unidos, se explica las funcionalidades del Kinect así como sus campos de trabajo, multifuncionalidades y el impacto que va a generar más allá de los video juegos [45]. En la Universidad de California se hizo un estudio de cómo el sistema de captura de gestos es una herramienta robusta para la rehabilitación de pacientes con lesiones en medula espinal, haciendo de este un método factible para la fisioterapia [46].

La Universidad de Tennessee ha desarrollado un traductor de lenguaje de señas automático usando el Kinect. Un usuario sordo se ubica frente a la cámara del dispositivo y realiza gestos del lenguaje de señas, esta información es leída y procesada por la herramienta para posteriormente ser traducidas a palabras [47]. En la Universidad de Notre Dame, Indiana, realizaron una herramienta para la rehabilitación de pacientes con accidentes cerebro vasculares. La herramienta ayuda en la recuperación física del usuario al permitir realizar ejercicios de movimientos, coordinación y equilibrio usando el Kinect ya que este rastrea posiciones con gran facilidad [48].

En la universidad del Cauca existen dos trabajos basados en captura de movimiento usando Kinect. El primero de ellos realizado por los ingenieros Manuel León y Julián Paz consiste en generar trayectorias para un robot puma utilizando la mano como referencia para la obtención de las coordenadas cartesianas [49]. El segundo es un proyecto que está en ejecución, desarrollado por el ingeniero Diego Alberto Bravo en su tesis de doctorado. Utiliza el Kinect para capturar el movimiento humano con el objetivo de generar trayectorias para el movimiento de un robot bípedo.

3. DISPOSITIVO KINECT CON ROBOSURGERY

El sensor Kinect creado por Alex Kipman y desarrollado por Microsoft para la consola de Xbox 360, es un dispositivo periférico de entrada que actúa como sensor de movimiento. Inicialmente conocido como "*proyect natal*", según Kipman, natal hace referencia a "el nacimiento de la próxima generación de entretenimiento en el hogar". *Proyect natal* toma el nombre de Kinect en el Electronic Entertainment Expo (E3) 2010 de Los Ángeles y deja gran expectativa entre los asistentes, puesto que se eliminan completamente los controladores tradicionales o mando, para abordar la era de la "interfaz de usuario natural, NUI", donde se manipula la consola por medio de gestos o comandos por voz sin necesidad de un contacto físico [50].

El dispositivo Kinect utiliza la tecnología basada en software creado por RARE, una compañía perteneciente a Microsoft Game Studios, posee una cámara capaz de medir distancias, leer cuerpos humanos y reconocer sus respectivos gestos, además de poseer el chip PS1080 capaz de procesar datos de profundidad a 30 imágenes por segundo. El dispositivo en general fue creado por PrimeSense, compañía israelí con gran experiencia en creación de interfaces naturales de usuario [51].

El sensor Kinect está diseñado ergonómicamente para ajustarse a espacios reducidos, en la parte superior o inferior del televisor, su longitud horizontal es de aproximadamente 23 centímetros y el vástago donde se encuentran los sensores se conecta a una base rectangular con un eje de rotación tipo rotula. Kinect se compone principalmente de:

- Una cámara tradicional (Resolución 640x480 RGB 30fps VGA).
- Un emisor de infrarrojos.
- Una cámara de infrarrojos.
- 4 Micrófonos multi-Array (16bit sampling rate: 16Hz).
- Un motor.

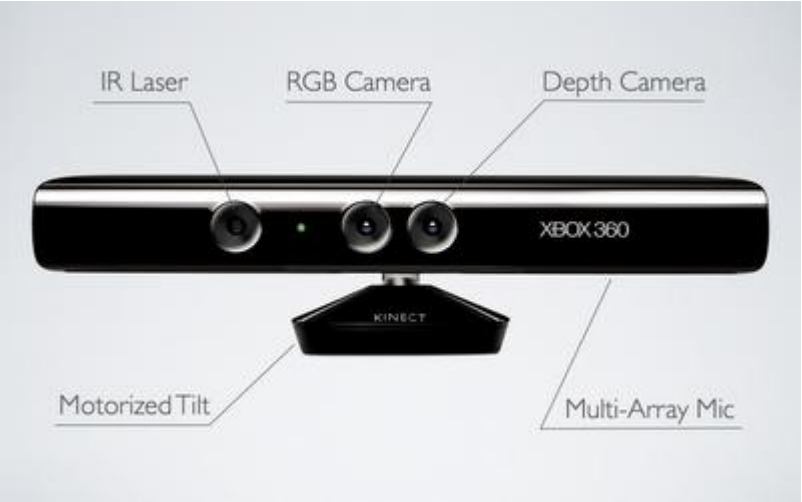


Figura 24. Composición del sensor Kinect [52].

3.1 Módulos Kinect

La tecnología usada en Kinect permite dividir este dispositivo en tres partes fundamentales, reconocimiento de imágenes, reconocimiento de voz y el motor; estas en conjunto garantizan el correcto funcionamiento y características únicas.

3.1.1 Reconocimiento de imágenes

Una de las funciones más importantes del sensor Kinect es la captura de movimientos, ésta se realiza gracias a dos componentes, un proyector láser de infrarrojos combinado con un sensor CMOS monocromo, que captura los datos de vídeo en 3D en cualquier condición de luz ambiental. Mediante el rebote del haz de infrarrojo en todo el campo de visión se puede identificar el movimiento, como levantarse de una silla, denominado “campo de profundidad”, el sensor infrarrojo, recibe los rebotes y dependiendo de la cercanía en la que se encuentren los cuerpos aparecen en diferentes colores. Las imágenes tomadas por el sensor pasan por una serie de filtros para la distinción de personas y objetos, además el Kinect está cargado con 200 poses comunes del ser humano que facilita el llenado de espacios en blanco en el sistema de directrices [53].

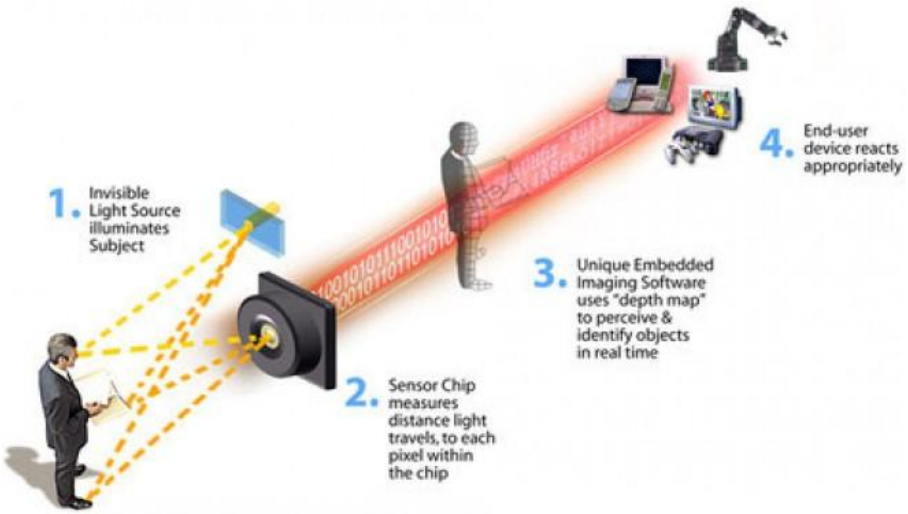


Figura 25. Funcionamiento del sensor Kinect [52].

Los sensores de salida de video del Kinect fueron determinados a una velocidad de ~ 9 Hz a 30 Hz, dependiendo de la resolución del video. El método predeterminado en sistema RGB utiliza una resolución VGA de 8 Bits empleando un filtro Bayer, sin embargo, el Kinect puede transmitir el punto de vista de su cámara de infrarrojos directamente como 640x480 vídeo, o 1280x1024 a una velocidad inferior.

La tecnología empleada en el Kinect para el mapeo.es capaz de rastrear hasta seis personas simultáneamente. Con la información adquirida y ordenada, el sensor convierte las partes del cuerpo en un esqueleto en movimiento, mediante la unión de cerca de 48 puntos del esqueleto humano ubicados en: cabeza, cuello, hombros, codos, muñecas, manos, abdomen, cadera, rodilla y pies (figura 26). El único inconveniente es que el sensor Kinect no segmenta dedos [53].

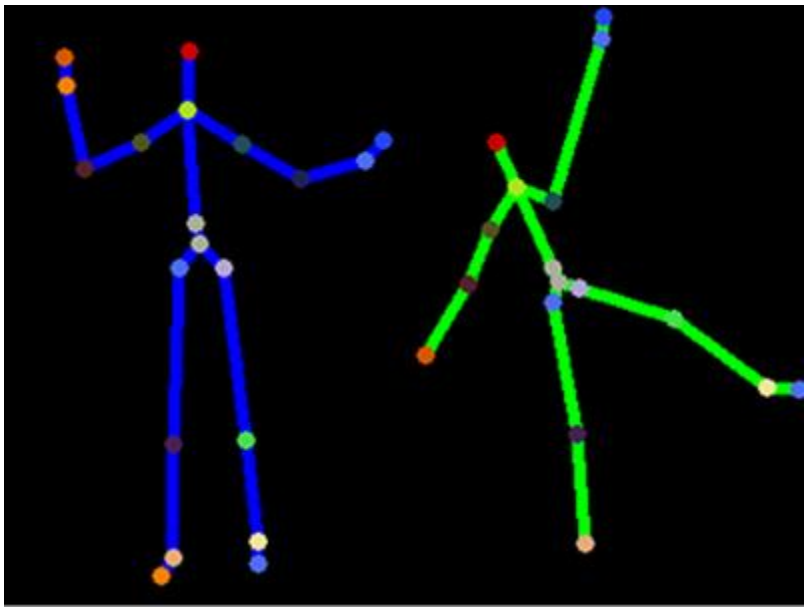


Figura 26. Ubicación de puntos [54].

3.1.2 Reconocimiento de voz

El reconocimiento de voz se hace mediante un array de 4 micrófonos ubicados en la parte inferior (figura 27), cada uno funciona en un canal de audio de 16 bits a una velocidad de muestreo de 16 kHz, resultado de experimentos realizados por el laboratorio de Microsoft Research en 250 residencias con 26 micrófonos en distintos ambientes, determinando la mejor configuración para superar perturbaciones. En primer lugar se necesita gran sensibilidad para captar e interpretar sonidos provenientes del usuario (aproximadamente 3 metros de distancia), al mismo tiempo que debe ignorar ruidos ambientales y otros sonidos que no sea la voz del usuario.

Figura 27. Micrófonos multi array [55].

El sistema de micrófonos multi-array permite que el sonido sea capturado de forma independiente, por tanto cada micrófono registra el sonido con desfase, lo

que permite calcular el lugar de origen del sonido (*beamForming*), para luego ser procesadas, eliminando el ruido del ambiente (niños, televisión, sistemas de sonido) y todo lo que se encuentra por fuera de la frecuencia de la voz humana (80 y 1100 Hz).

El Sensor Kinect está equipado con un “modelo acústico” para distintos países y dialectos regionales, construidos a partir de combinaciones hechas por actores de diferentes partes del mundo, el sistema de reconocimiento de voz es similar a la óptica, se ejecuta continuamente en modo micrófono abierto (todo el tiempo) [53].

3.1.3 El motor

Una de las partes fundamentales del Kinect es el motor, debido a su complejidad Microsoft invirtió mucho tiempo en la investigación y análisis de las diferentes configuraciones y posibilidades de cada uno de los espacios en que puede encontrarse. La conclusión a la que llegó Microsoft es que la cámara debe moverse verticalmente, para adaptarse a un ambiente determinado y a cualquier campo de visión, sin tener que adaptar el espacio a la cámara.

El motor de inclinación está ubicado en la base de la cámara (motivo por el cual es bastante pesada) y le da a ésta una movilidad de 27° hacia arriba y abajo (figura 28)**,** facilitando la calibración del espacio de detección con mayor precisión, además se recomienda que el dispositivo esté a una altura superior a un (1) metro [53].

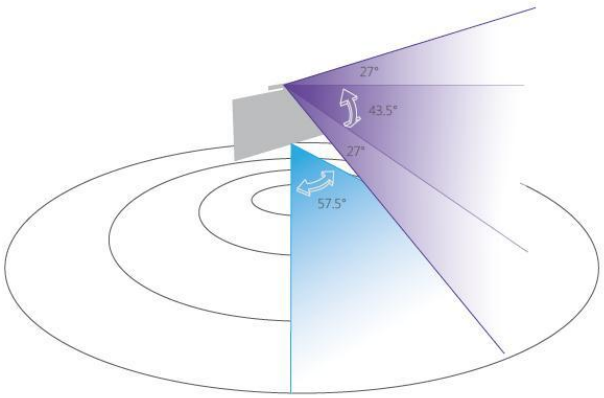


Figura 28. Movilidad en grados del Kinect [56].

3.1.4 Especificaciones del sensor Kinect

Las características mencionadas en la sección anterior hacen del Kinect un dispositivo periférico, con muchas posibilidades de desarrollo software, por esto se hace necesario conocer detalladamente sus especificaciones técnicas además del error en la medición.

Elemento	Rango
Angulo de visión.	- 43° vertical - 57°horizontal
Rango de movimiento de cabezal (vertical).	±27°
Velocidad de imagen (flujo de datos de color y profundidad).	30 marcos por segundo (FPS).
Resolución de la cámara de profundidad.	VGA (640X480).
Resolución de la cámara de color RGB.	VGA (640 X 480).
Formato de audio.	16 kHz *mono pulse code modulation* (PCM).
Características de entrada de audio.	Un *array* de cuatro micrófonos con un conversor analógico a digital de 24 bits.

Tabla 1. Especificaciones técnicas del Kinect [57].

Teniendo en cuenta las variables anteriores, principalmente el rango de visión y la inclinación del cabezal, es necesario conocer los rangos de distancia (metros) en la cual el sensor Kinect puede operar satisfactoriamente:

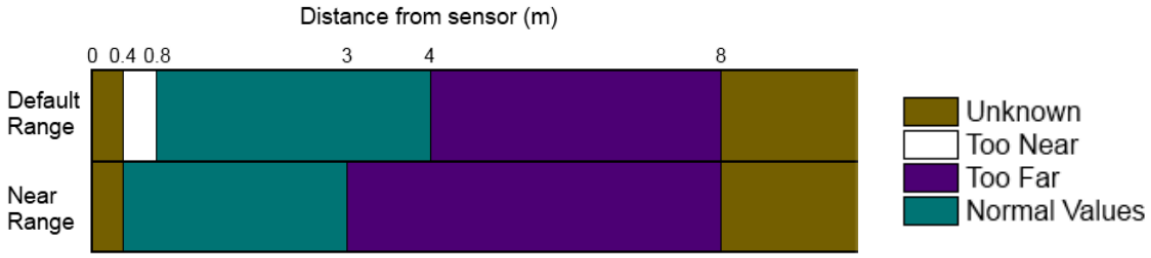


Figura 29. Rango de trabajo del sensor [57].

Pese a las especificaciones de uso del Kinect, entre ellas la distancia de trabajo pues lo ideal es a 2,5 metros, a distancias superiores a 3,5 metros el dispositivo empieza a tener capturas pobres en información, por lo tanto la precisión del

Kinect disminuye en función a la distancia. La figura 30 muestra el error de un pixel en centímetros según su distancia.

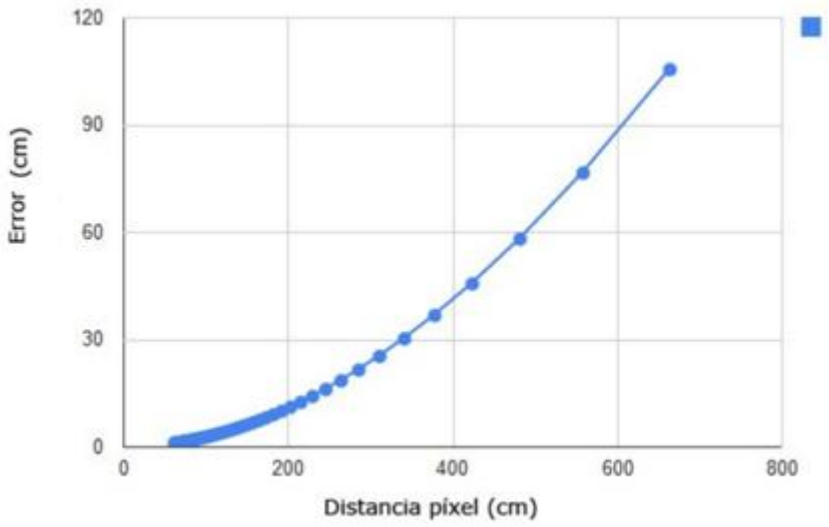


Figura 30. Grafica error vs distancia [43].

Para resolver algunas incógnitas sobre el Kinect y su error, en la Universidad de Aarhus, Dinamarca, se hicieron una serie de experimentos con el fin de describir algunas de las propiedades como resolución, exactitud y precisión de la profundidad.

Inicialmente se trabaja sobre la linealidad del Kinect y se señalan puntos a distancias diferentes perpendiculares a la superficie plana. En cada punto estimado por el sensor de profundidad se promedia en un área pequeña y se compara con la distancia real, siendo estas muy cercanas a la linealidad dentro del rango de medición del Kinect. La figura 31 muestra el valor de la profundidad versus distancia en un rango de medición de 0,5 m a 9,7 m. Cabe resaltar que si se exceden los rangos de medición la estimación de profundidad se hace 0.

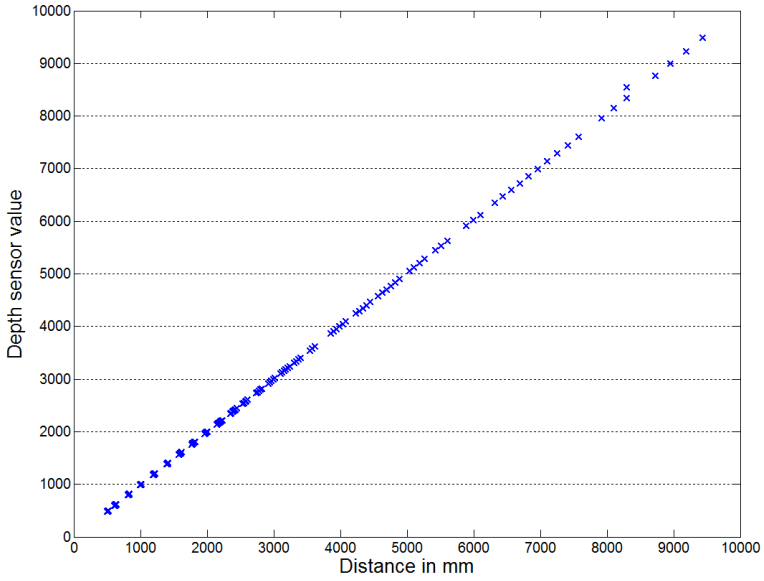


Figura 31. Valor de profundidad del Kinect vs distancia [58].

En la segunda configuración se ubica el Kinect perpendicularmente a una superficie plana y se registra un marco de profundidad. Esto se repite a diferentes distancias, guardando cada valor de profundidad para extraer sus datos y clasificarlos. Desde una visión real, los datos son continuos, idealmente todos los datos desde la profundidad mínima hasta la máxima deben estar presentes. Sin embargo, se registran datos discretos y se asume que cada valor está presente. En la figura 32, se muestran los resultados, lo que demuestra que la resolución es cada vez mayor a medida que aumenta la distancia.

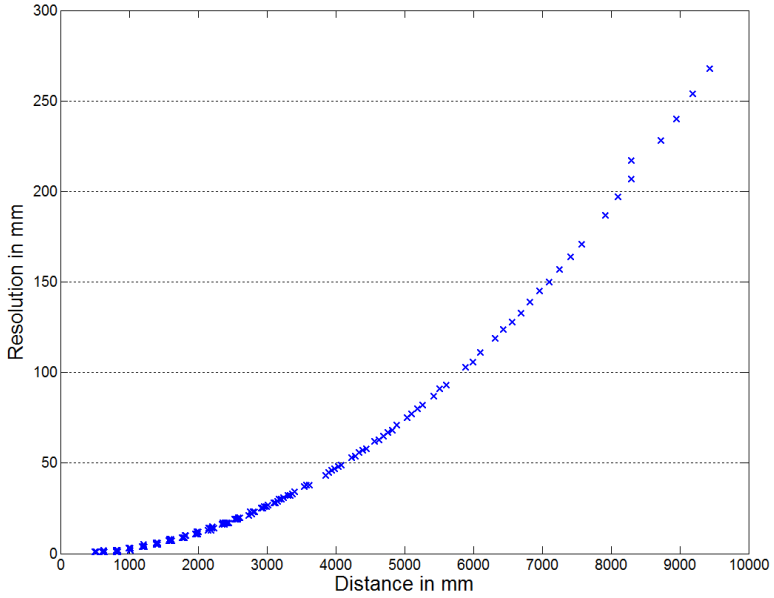


Figura 32. Resolución vs Distancia [58].

3.2 Kit de Desarrollo Software (SDK)

Es una herramienta software que permite desarrollar aplicaciones de algún dispositivo o sistema determinado, estos kits facilitan la tarea del programador al poder acceder a un paquete de librerías y funciones de algún sistema en concreto. Por lo general las SDK cuentan con ejemplos o material para el uso correcto de sus herramientas. Para este caso se mostraran los kits para la manipulación del dispositivo Kinect y sus lenguajes de programación.

3.2.1 Libfreenect (OpenKinect)

Este software realizado por el grupo OpenKinect es una interfaz que proporciona bibliotecas para hacer uso del dispositivo Kinect por medio de un ordenador o algún otro dispositivo. Estas bibliotecas de código abierto se diseñaron para el funcionamiento en Windows, MAC OSX y Linux. Héctor Martin publico los drivers para Linux con lo cual ganó el premio de Adafruit y fue uno de los primeros hacker en publicar un código abierto para el Kinect.

Este kit se logra enlazar con lenguajes/plataformas como C, C + +, NET (C # / VB.NET), Java y Python. Cuenta con bibliotecas gratuitas y código abierto que hacen de esta herramienta una plataforma ideal para trabajar con el Kinect fuera de la consola Xbox 360 [59].

3.2.2 OpenNi/NITE (PrimeSence)

Este driver de código abierto realizado por la empresa PrimeSence es unas de las SDK más utilizadas para la creación de aplicaciones con el hardware Kinect. OpenNI (*Open Natural Interaction*) está diseñado como un *framework* (marco de trabajo) permite el manejo de sensores 3D por medio de librerías, utiliza las mismas bibliotecas que el NITE (framework de código cerrado) de los desarrolladores de videojuegos para la XBOX 360.

Esta SDK permite un acceso a todos los módulos del Kinect: sensores, video y audio, que a su vez proporciona funciones como la captura de movimientos en tiempo real, reconocimiento de gestos con las manos, uso de comando de voz y analizador de escenas para la detección del entorno. Se diseñó este software para los sistemas operativos Windows 7 y Ubuntu 10.10 [60].

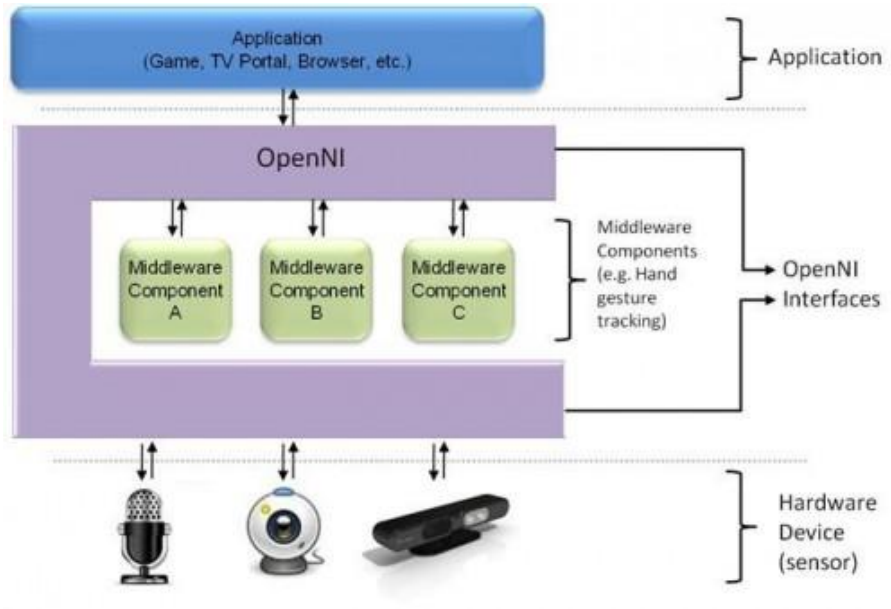


Figura 33. Arquitectura OpenNI [61].

3.2.3 SDK (Microsoft)

Es un kit de herramientas que facilita el desarrollo de aplicaciones y la interacción con el dispositivo Kinect, su funcionamiento principal es la toma de datos de las diferentes partes del cuerpo y articulaciones. Al poder reconocer todo el esqueleto humano el sistema permite recolectar información detallada de posiciones específicas en un plano x, y, z de alguna parte del cuerpo que esté definida en las librerías. Con la información que se logra recolectar es fácil crear software o

aplicaciones para diferentes ámbitos con la certeza que funcionarán con la interacción del cuerpo humano.

Cuando Microsoft lanzo el sensor 3D para su video consola encontró una gran cantidad de clientes que no solo querían entretenimiento lúdico, por esta razón y por los diferentes hackers que adaptaron sus dispositivos a ordenadores, Microsoft publicó su primera versión de librerías de código abierto Beta SDK para programadores y decidió que su uso se haría mediante Microsoft Visual Studio 2010 gracias a su .NET Framework 4. Hoy en día existen diferentes versiones que mejoran el rendimiento y las funciones de esta herramienta. Versiones: Beta1, Beta2, SDK 1.0, SDK1.5, SDK 1.6, SDK 1.7, SDK 1.8. Estas versiones permiten utilizar C++, C# o Visual Basic para la programación y creación de aplicaciones, las primeras dos versiones fueron diseñadas para el dispositivo Kinect de la consola Xbox 360 y no funcionan con el dispositivo Kinect para Windows, cada actualización deja mejoras y nuevas funcionalidades que los programadores podrán utilizar para crear aplicaciones más llamativas y funcionales.

La última versión del SDK de Microsoft es la 1.8.0.595 y fue lanzada el 13 de septiembre de 2013, soporta HTML y Java script, permite editar las imágenes de fondo del usuario bien sea eliminándolo o remplazándolo por uno artificial, permite la construcción de modelos 3D de una escena y logra bloquear escenas en la medida que se mueve la cámara logrando así un mejor escaneo para los modelos 3D [62].

3.3 Microsoft Visual Studio

El entorno de desarrollo integrado (IDE) para el sistema operativo de Windows es un conjunto de herramientas de desarrollo llamado Microsoft Visual Studio, este entorno permite la programación en varios lenguajes como Visual Basic, Visual C#, Visual J# y Visual C++. Gracias a la variedad de lenguajes es posible crear aplicaciones y soluciones mediante la combinación de estos.

Las versiones de este entorno inician en el año 1998 con el Visual Studio 6.0 y gracias a la buena acogida de los usuarios la empresa comienza la optimización y actualización de su plataforma. Entre las versiones encontramos: Visual Studio .NET (2002), Visual Studio .NET 2003, Visual Studio 2005, Visual Studio 2008, Visual Studio 2010 y Visual Studio 2012 [63].

La versión utilizada en el proyecto fue Microsoft Visual Studio 2010 Ultimate ya que utiliza .NET Framework 4, compatible con Kinect.

3.4 Librerías Microsoft SDKs 1.6

La versión 1.6 proporciona los elementos necesarios para acceder a la mayoría de funciones de todos los sensores del dispositivo y realizar innumerables aplicaciones. Entre las novedades más importantes de esta versión publicada en octubre de 2012 se encuentran:

- Soporte para Windows 8.
- Soporte para Visual Studio.
- Captura de datos del acelerómetro.
- Captura de datos a mayor profundidad.
- Regulación de las características de la cámara.
- Nuevo paquete de voz en idioma alemán.
- Soporte para varios dispositivos.

Los encabezados para acceder a las funciones y a los datos de los diferentes sensores en el lenguaje C++ son:

NuiApi.h: Al agregar esta librería se tiene acceso a funciones básicas como de inicialización, enumeración de dispositivos y acceso a múltiples dispositivos. Es el encabezado principal de un proyecto e incluye NuiImageCamera.h y NuiSkeleton.h.

NuiImageCamera.h: Esta interfaz permite la configuración de la cámara, datos de lectura y el acceso a servicios de imágenes.

NuiSkeleton.h: Se emplea para habilitar las funciones de seguimiento de esqueletos y extracción de datos. Da la posibilidad de rastrear la imagen de todo el esqueleto o de alguna articulación, de una o dos personas que se encuentren en el campo de visión del dispositivo.

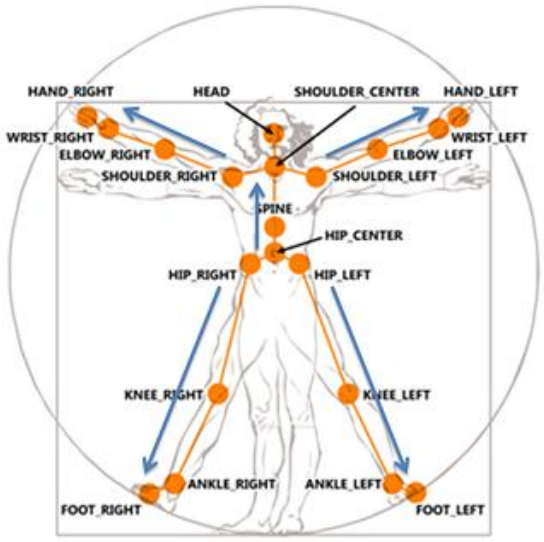


Figura 34.Articulaciones del esqueleto disponibles para seguimiento [57].

NuiSensor.h: Habilita funciones de los dispositivos de audio (micrófonos). Permite la supresión acústica de ruido y cancelación de eco [63].

3.5 RoboSurgery-Kinect

RoboSurgery

El software RoboSurgery es una herramienta académica para la práctica y experimentación con robots quirúrgicos virtuales, fue realizada en la plataforma Microsoft Visual Studio 2008 en el lenguaje C++ y cuenta con tres robots para la realización de una cirugía laparoscópica, los tres son manipulados por un control de mando o joystick. Este simulador quirúrgico se realizó como trabajo de grado en la Universidad del Cauca por el Ingeniero Diego Guzmán y cuenta con una metodología de desarrollo llamado ciclo de vida evolutivo o modular que facilita a nuevos programadores el entendimiento del sistema y su edición.

La mayoría de herramientas utilizadas para la implementación del proyecto deben estar instaladas en el ordenador del programador si se desea su edición, más no si se quiere hacer uso de la aplicación como tal. Entre estas herramientas software utilizadas encontramos: QT para el diseño de la interfaz (qt-win-opensource-4.7.4-vs2008.), Cmake para exportar archivos y codificarlos en Visual Studio (cmake-2.8.5), Visual Studio 2008 como entorno de desarrollo y VTK (*Visualzation Toolkit*),

bibliotecas para la visualización de geometrías 3D (vtk-5.8.0.). Blender (blender-2.6.1) y Make Human (makehuman-nightly-win32) son herramientas para la elaboración de los objetos en 3D de la herramienta.

Los robots virtuales Hibou y Lapbot realizan la cirugía laparoscópica, el primero es un porta endoscopio que permite la manipulación de la cámara dentro del paciente y el segundo es el encargado de realizar la cirugía como tal. El movimiento de los dos Lapbots y del Hibou que se muestra en el simulador se logra gracias al modelo geométrico inverso (MGI) obtenido de los parámetros geométricos. Las coordenadas x, y, z del joystick son utilizadas por el MGI y este retorna los ángulos y posiciones de cada articulación [25].

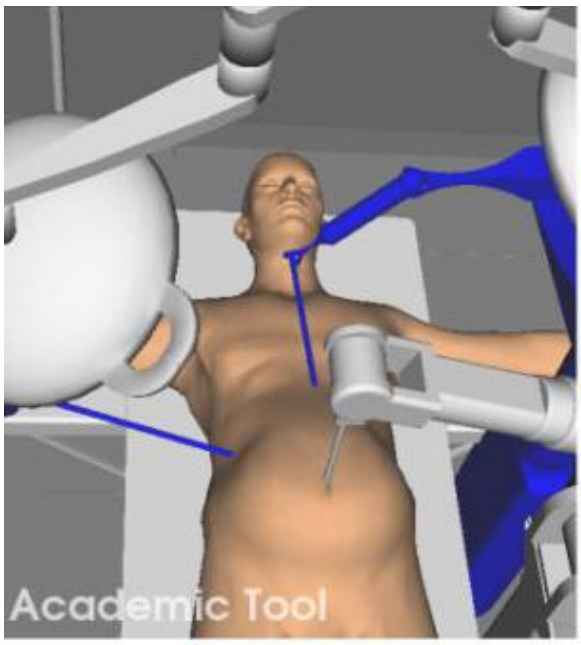


Figura 35.Hibou y Lapbots de RoboSurgery [25].

Para el presente proyecto se realizó la migración del entorno de desarrollo Visual Studio 2008 al Visual Studio 2010, logrando adecuar la herramienta software RoboSurgery para ser editada con herramientas y líneas de código del dispositivo Kinect que utiliza el SDK de Microsoft. Como se dijo anteriormente este kit fue desarrollado para su uso en la versión 2010 ya que contiene soporte .NET 4.0.Las herramientas software también deben ser actualizadas para que todo el sistema opere en el nuevo entorno (ver anexo A: Instalación de bibliotecas).

Código para el Kinect

Para el presente trabajo se utilizó una serie de funciones que permiten el encendido del dispositivo y la extracción de variables importantes en el lenguaje C++ con la librería SDK. Cada línea de código es solo un pequeño fragmento del kit de desarrollo que al utilizarlo adecuadamente proporciona información suficiente para hacer aplicaciones interesantes.

- NuiInitialize (NUI_INITIALIZE_FLAG_USES_SKELETON)

La primera parte inicializa el Kinect y es importante saber que solo se puede llamar una vez en toda la aplicación ya que otra llamada de NuiInitialize fallará. La segunda parte especifica los subsistemas Kinect para inicializar, en este caso una bandera NUI_INITIALIZE_FLAG_USES_SKELETON que le indica al software que debe inicializa el sensor para proporcionar datos del esqueleto. El encabezado para utilizar estas y las siguientes funciones es NuiApi.h.

- NUI_SKELETON_FRAME ___

Contiene los datos extraídos desde una fuente de información tipo esqueleto, es decir se crea un espacio donde los datos de un esqueleto se almacenaran.

- NuiSkeletonGetNextFrame (milisg, ___)

Esta función obtiene los datos de la siguiente captura o marco del esqueleto, en el primer argumento se especifica el tiempo en milisegundos que la función debe esperar antes de leer otro dato del esqueleto y en el segundo argumento el lugar donde se están almacenando los datos.

- (NUI_SKELETON_DATA)
  SkeletonData[]

Se utiliza para especificar el usuario al que se le extraerán los datos en caso de que se encuentre más de una persona frente al dispositivo.

- (NUI_SKELETON_POSITION_INDEX)
  SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT]

Información de la posición de un esqueleto, en este caso la posición de la mano derecha, permite especificar qué coordenada se desea extraer (x, y, z).

- NuiShutdown

Cerrar todo y desconectar el dispositivo.

4 RESULTADOS

Las investigaciones realizadas, las bases teóricas consultadas y el conocimiento de las funcionalidades de todos los programas mencionados anteriormente dieron como resultado la integración de dos sistemas; el primero un dispositivo de control de interfaz natural llamado Kinect y el segundo el simulador quirúrgico RoboSurgery realizado en la Universidad del Cauca, consiguiendo con esto el cumplimiento de los objetivos pautados para este trabajo.

En la figura 36 podemos observar el esquema de manipulación de los robots, la interfaz de entrenamiento RoboSurgery cuenta con las librerías necesarias para la adquisición de datos desde el joystick y Kinect proporcionadas por SDL y SDK 1.6 respectivamente. Las señales del joystick son procesadas por el ordenador y finalmente la interfaz registra eventos de sus botones y valores analógicos del joystick para el movimiento del robot porta endoscopio Hibou. Las señales obtenidas del dispositivo Kinect hacen posible el movimiento de los robots quirúrgicos LapBot mediante la extracción de las coordenadas x, y, z de la mano derecha e izquierda para cada robot; el cambio de instrumento se realiza por el movimiento vertical hacia arriba de la rodilla derecha.

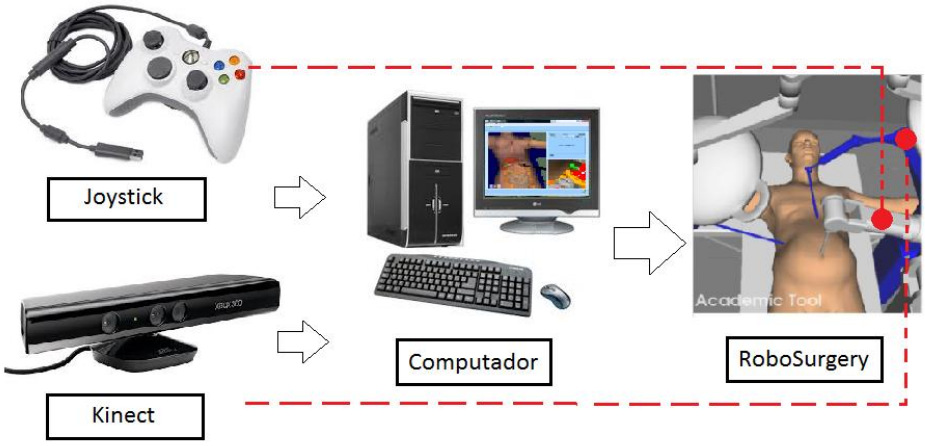


Figura 36. Componentes del sistema de captura de gestos.

4.1 Simulaciones de recolección de datos

Para conocer el funcionamiento del Kinect dentro del lenguaje de programación C++ y Visual Studio se realizaron una serie de pruebas extrayendo las coordenadas x, y, z de la mano derecha realizando diferentes movimientos. Con los datos guardados en un archivo de texto se procedió a graficar estos movimientos en Matlab; inicialmente se realizan líneas rectas de aproximadamente 0.5 metros de largo, seguido de la realización de círculos con 0.2 metros de radio, con éstos encontramos el error cartesiano y finalmente se hace la respectiva confirmación de los modelos MGI y MGD del robot LapBot, mostrando con esto la manipulación y seguimiento de la trayectoria. Para el muestreo de datos no se establece un número fijo de muestras, por el contrario, son almacenados los requeridos para elaborar la figura geométrica establecida puesto que la manipulación es realizada en tiempo real.

A continuación se muestra el desarrollo de los primeros muestreos correspondientes a una línea recta en los planos XY, YX y XZ, con el objetivo de aproximar la dispersión respecto a una línea. En primer lugar se toma el plano XY para la realización de la trayectoria con el Kinect (figura 37) y se procede a graficar estos datos en Matlab, obteniendo una gráfica con dos líneas, la de color azul representa los datos del dispositivo y la negra puntada indica la referencia (figura 38). Se puede observar que el punto flotante tomado por el Kinect oscila entre +0.01 y -0.01 m alrededor de la recta en el eje Y.

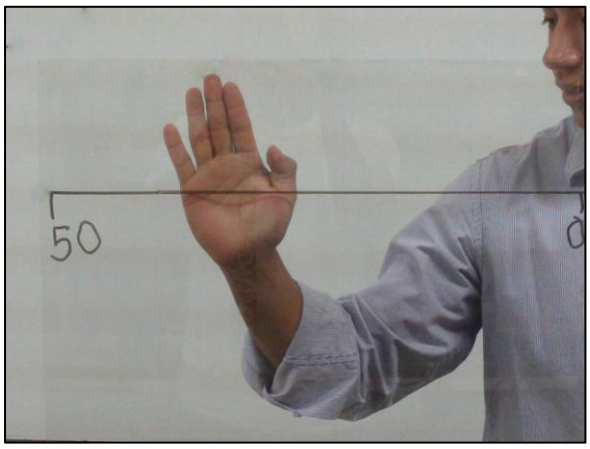


Figura 37.Referencia de línea para la toma de datos.

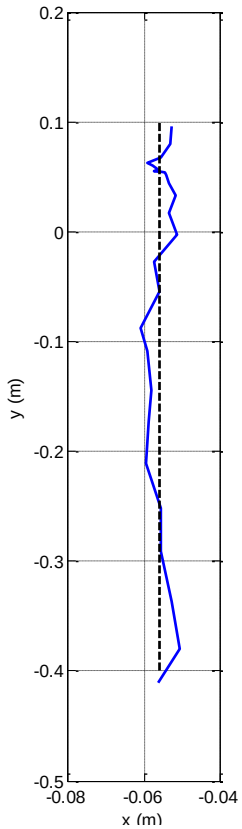


Figura 38.Línea vertical en XY.

En segunda instancia se hace un muestreo con el respectivo almacenamiento de los datos correspondientes al plano YX, para determinar la oscilación en el eje Y, la cual está aproximadamente entre +1cm y -1cm respecto a la recta trazada (figura 39).

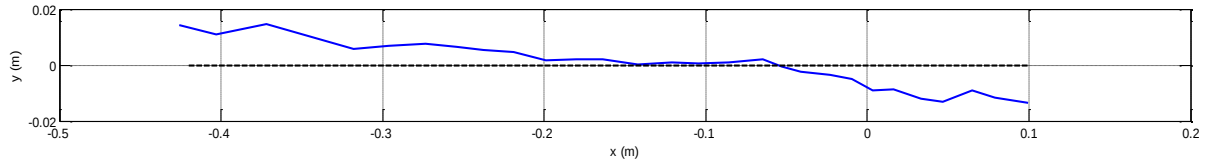


Figura 39. Línea horizontal en YX.

Se recolectan los datos para el plano YZ, este último muestreo cambia el orden debido a que es necesario comprobar la medición del dispositivo respecto a la profundidad. En la figura 40 se puede observar que el desplazamiento en el eje Z es de aproximadamente 0.5 m.

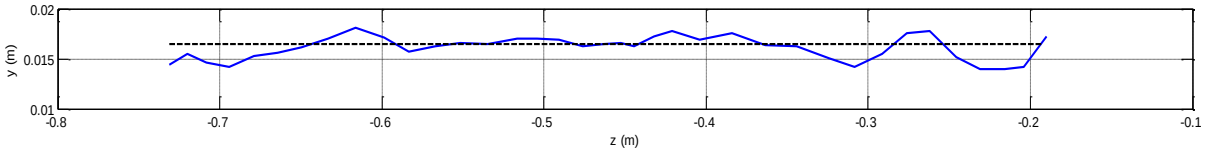
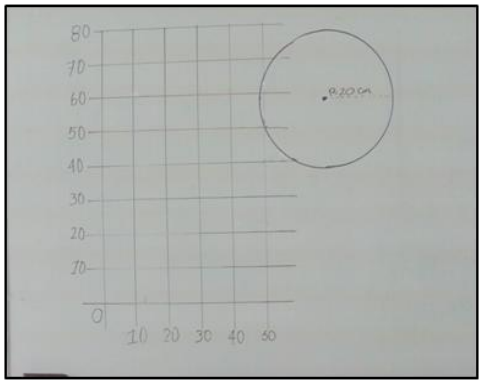
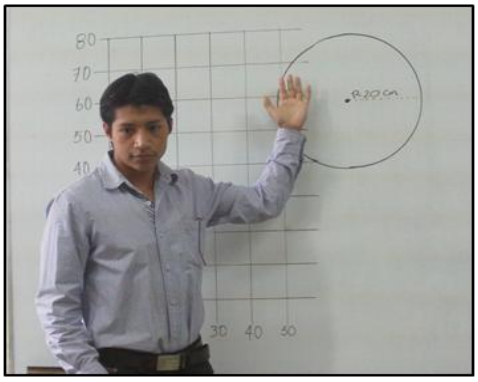


Figura 40. Línea en profundidad YZ.

En la segunda parte del proceso se realizan círculos de 0.2 m de radio. En color negro punteado se muestra el deseado y en azul los datos tomado por el dispositivo Kinect en los planos XY, YZ y XZ, como se muestran en la figura 42, 44 y 46, tomando como base un circulo hecho en tablero (figura a, b), vidrio (figura c, d) y en el aire (figura e, f).



a



b

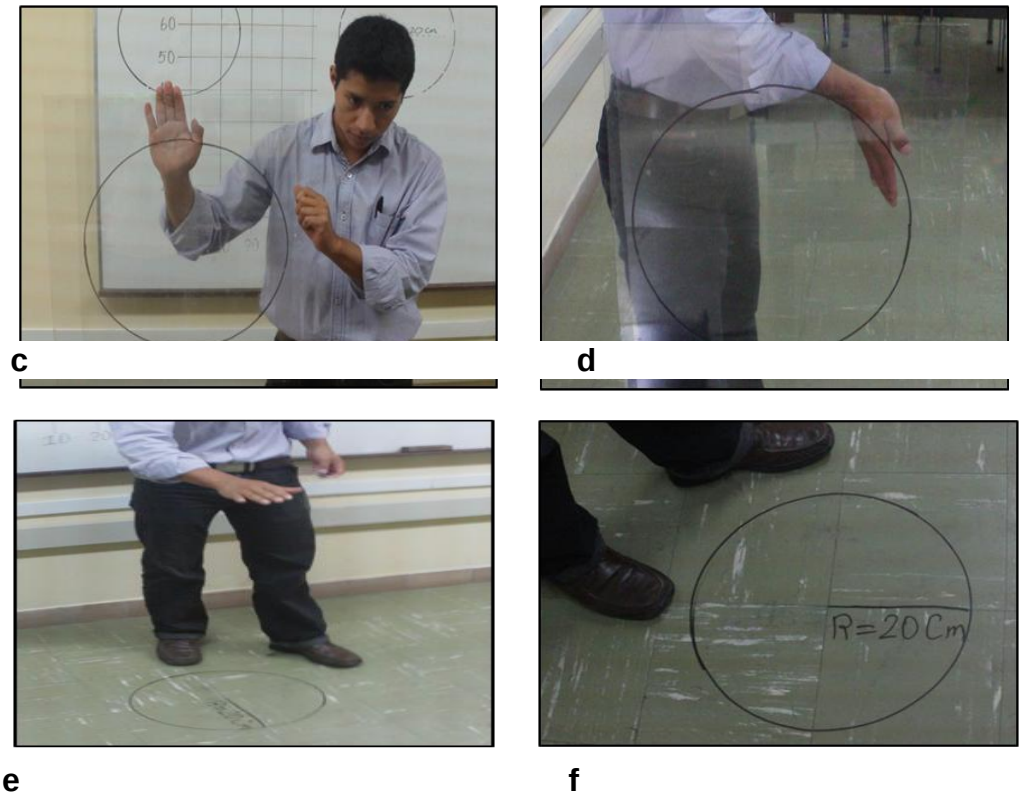


Figura 41. Referencias circulares para la toma de datos.

Con los datos obtenidos correspondientes a los círculos respecto a un plano específico se realizan las gráficas, a cada una de ellas se adiciona un círculo (deseado) generado por Matlab con el mismo radio y aproximadamente el mismo centro. Los círculos deseados se generan con el triple de datos que el obtenido, esto con el objetivo de tener mayor puntos de referencia con los cuales comparar los datos obtenidos. Los círculos obtenidos, debido al periodo de ejecución de la aplicación no poseen igual distancia entre cada punto en el plano cartesiano, mientras que el generado en Matlab posee simetría.

Los primero datos corresponden a un circulo hecho en el plano XY (figura 42), tiene como centro los puntos X = -0.26 e Y = -0.02 y un total de 39 datos, el error es calculado punto a punto y su valor máximo es de 0.025 m (figura 43).

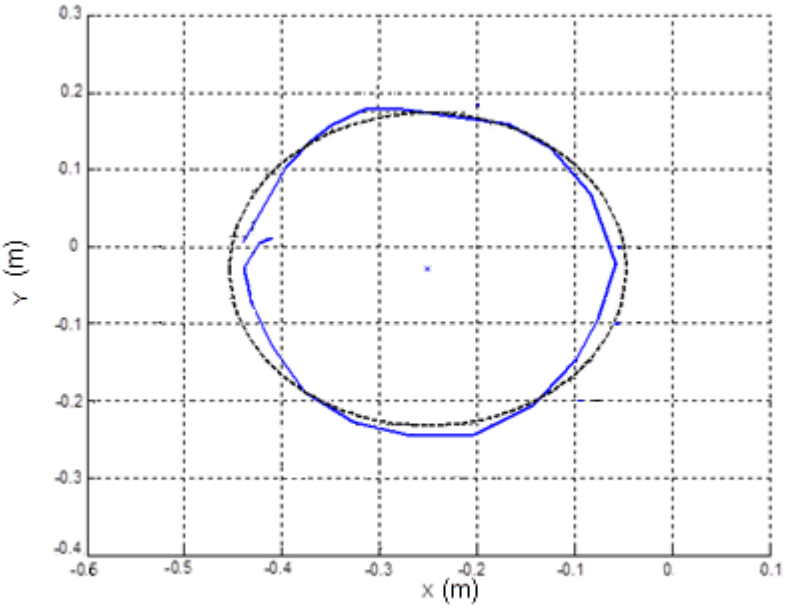


Figura 42.Trayectoria circular obtenida y deseada en el plano XY.

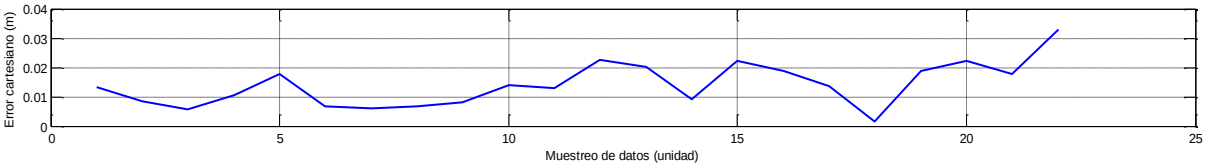


Figura 43.Error cartesiano circulo XY.

El siguiente circulo corresponde al plano YZ, ubicado en z = -0.064, y = -0.1 con 37 puntos (figura 44), el error es calculado de igual forma y su valor máximo es de 0.023 m (figura 45).

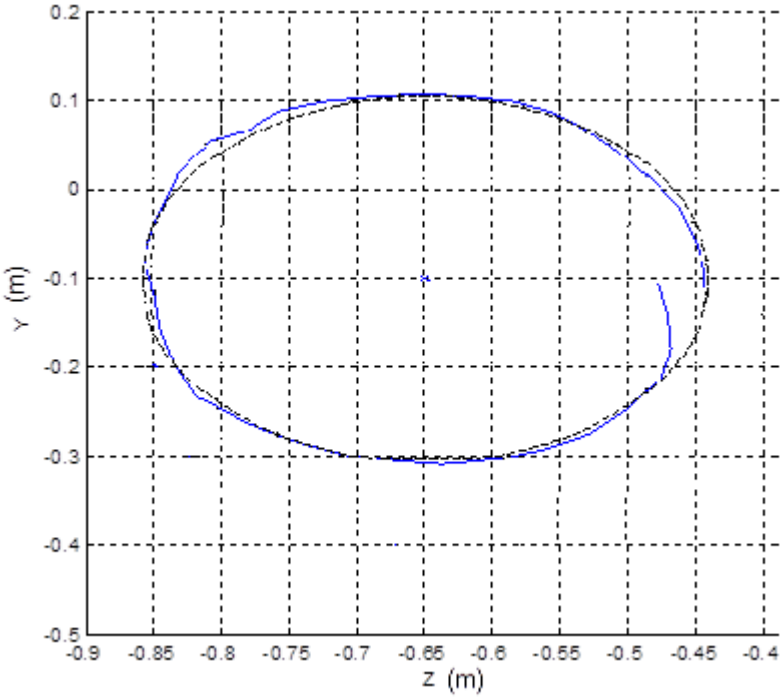


Figura 44. Trayectoria circular obtenida y deseada en el plano ZY.

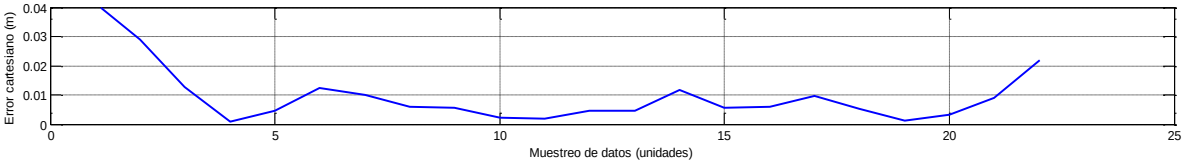


Figura 45. Error cartesiano circulo ZY.

Finalmente se toma un circulo en el plano XZ ubicado en x = -0.025, z = -0.3 y 51 datos (figura 46), el error máximo correspondiente al círculo es de 0.018 m (figura 47).

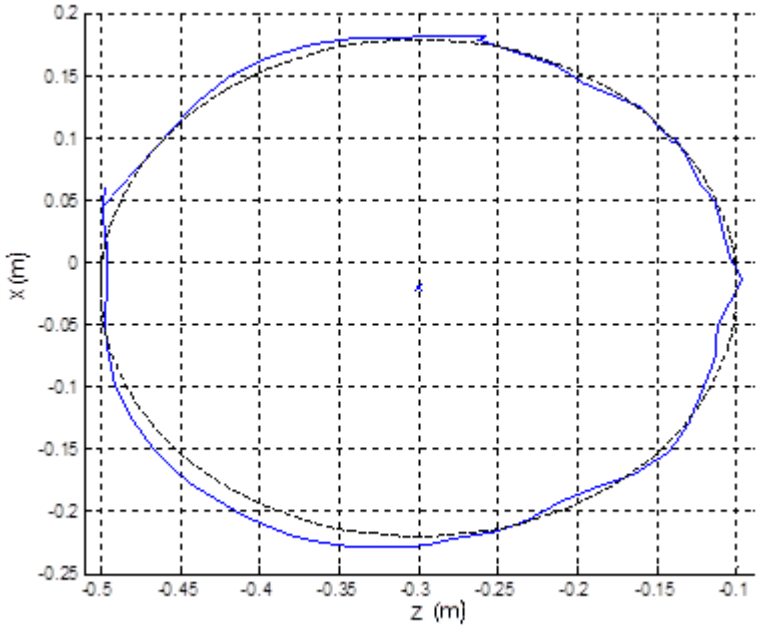


Figura 46. Trayectoria circular obtenida y deseada en el plano XZ.

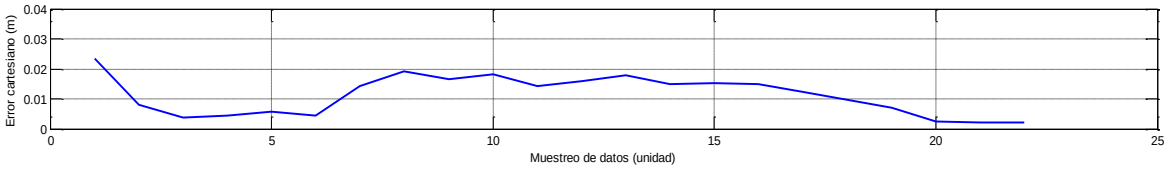


Figura 47. Error cartesiano circulo XZ.

En las figuras 43, 45 y 47 se observa que el error cartesiano es aproximadamente de 0.02 m respecto a un círculo, tomando los datos fuera de línea. Una vez iniciada la aplicación los datos muestreados mantienen su irregularidad y su valor analógico registrado se escala, por lo tanto el error cartesiano es de aproximadamente 0.001 m.

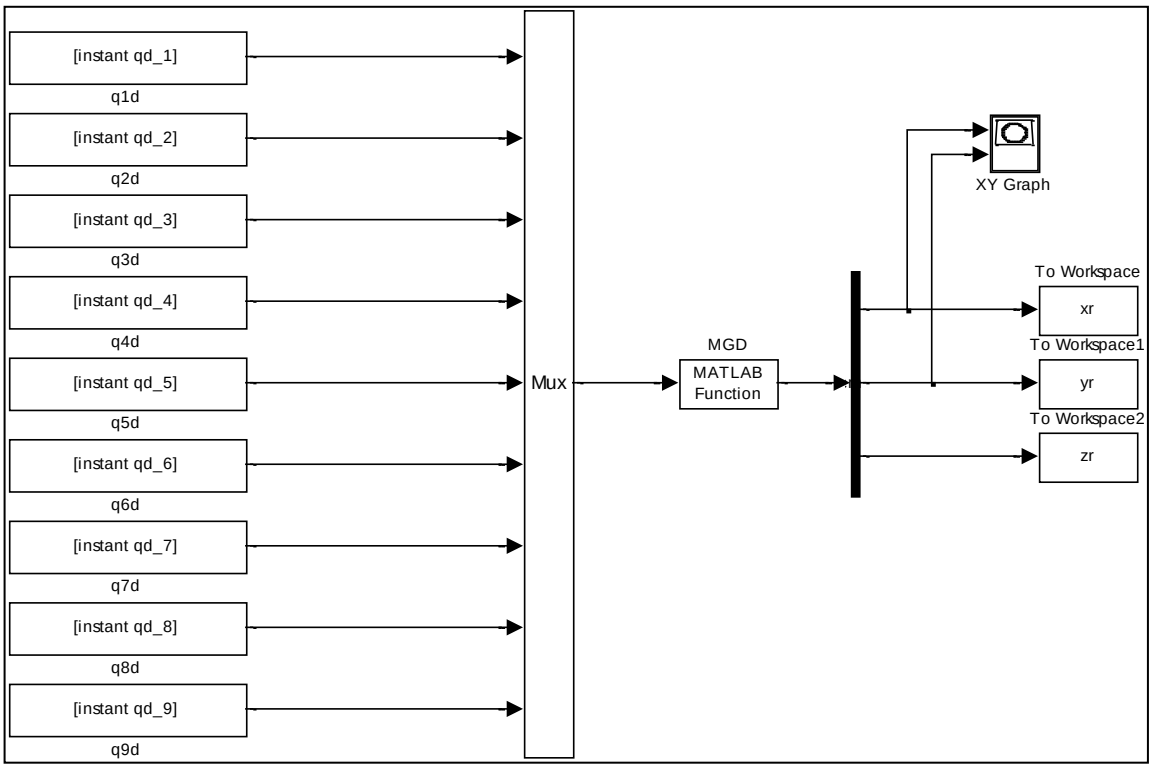


Figura 48.Diagrama en bloques de Simulink MGD.

En la figura 48 se muestra un diagrama de bloques en Simulink de Matlab, en el cual se ingresan los valores guardados previamente desde la aplicación, estos valores son tomados desde el Modelo Geométrico Inverso (MGI) en ángulos para articulaciones tipo rotacional y distancias para articulaciones tipo prismáticas, Los valores son tomados por el Modelo Geométrico Directo (MGD) para convertirlos en coordenadas cartesianas. Con lo anterior podemos observar cuál es la trayectoria seguida por el efector final. En la figura 49 encontramos la trayectoria deseada y en la figura 50 la obtenida. Las trayectorias son circulares, sin embargo, sus puntos de inicio y origen son diferentes, pues se realiza un intercambio de ejes para obtener movimientos naturales. En consecuencia se modifica la orientación del círculo obtenido respecto a la trayectoria deseada, dificultando la comparación entre las dos respuestas.

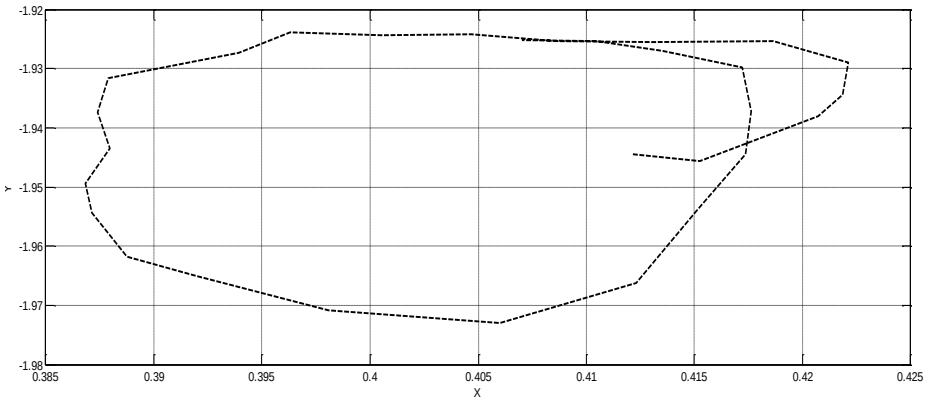


Figura 49.Trayectoria deseada robot Lapbot.

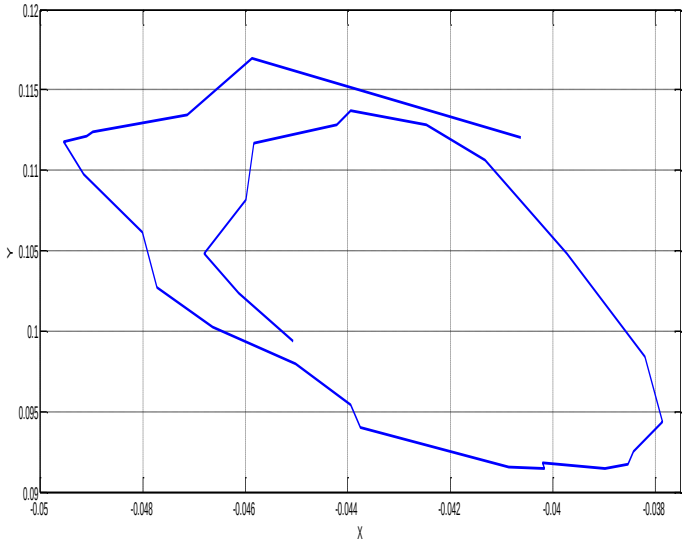


Figura 50. Trayectoria resultante robot Lapbot.

4.2 Componentes de la herramienta desarrollada.

Para el mejor funcionamiento del software se realizaron una serie de modificaciones en el código, la eliminación de algunas variables agilizan el procesamiento y la adición de pequeños procesos que dan robustez a la aplicación. Entre los cambios más representativos se encuentran:

Inicialización del programa:

Figura 51.Gesto para inicialización del sistema.

Antes de entrar al entorno virtual el sistema por medio del dispositivo Kinect reconoce que hay una persona frente a él y lee la posición de la mano izquierda. Cuando la mano se encuentre a 0.6 metros hacia arriba con respecto a la posición del Kinect el sistema accederá al entorno, esta acción se representa levantando la mano izquierda por encima de la cabeza y permite garantizar que hay una lectura de datos antes de manipular los robots.

Mensaje sobre los instrumentos de la mano derecha:

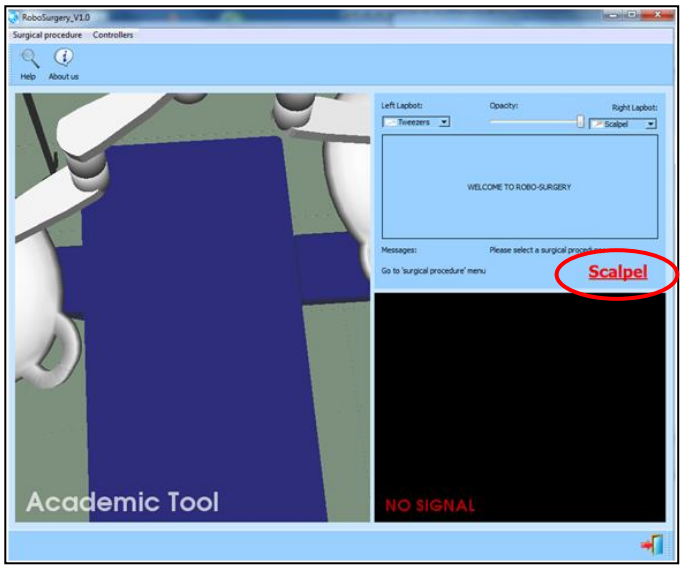


Figura 52. Mensaje sobre instrumento cargado en el robot Lapbot derecho.

El cambio de instrumentos del robot Lapbot derecho se efectúa por medio de la rodilla derecha, en la interface este cambio se observa en un mensaje de color rojo ubicado encima de la imagen de la cámara interna y ayudará al usuario a visualizar el instrumento que en el momento tiene configurado el robot para realizar la cirugía.

Eliminación del paciente niño:

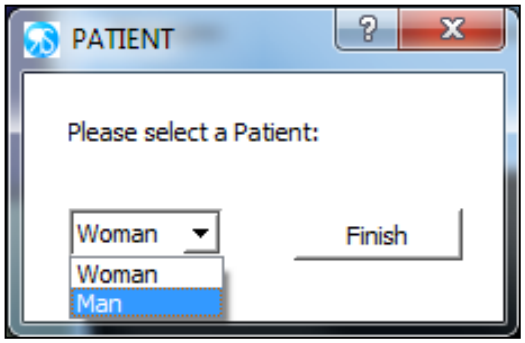


Figura 53. Interface para selección de paciente.

El software permitía la elección del paciente al cual se le realizará la cirugía o la inspección. Entre las opciones se encuentra hombre, niño o mujer y en este caso se elimina la opción niño para agilizar el código al no tener la necesidad de configurar las variables del Kinect para un nuevo tamaño del paciente.

Botones del joystick:

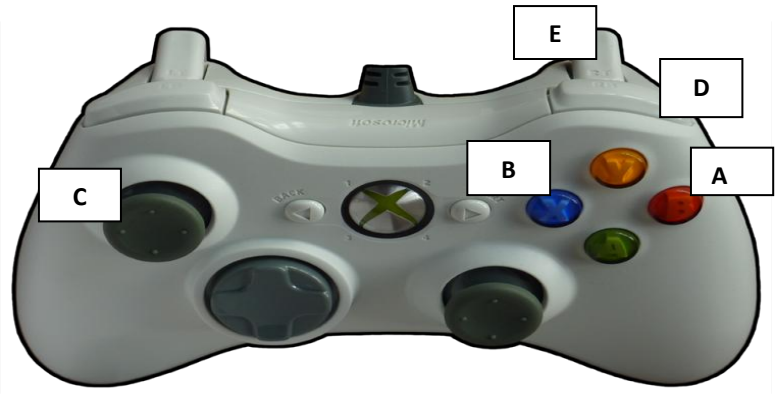


Figura 54. Asignación de botones en el control del Xbox.

El dispositivo de mando para la realización de pequeñas tareas y la manipulación del robot Hibou es un control de Xbox 360 con conexión USB. La elección de este dispositivo se debe a que el *stick* analógico de este tipo de controles es más fácil de manipular. Además realiza 500000 actuaciones verticales en 5 Hz mejorando la sensibilidad.

La asignación de los botones es la siguiente:

A. Botón (B), activa el robot Hibou.

B. Botón (X), activa los robots Lapbots.

C. *Stick* analógico izquierdo, manipulación en X e Y del robot Hibou.

D. *Bumper* derecho, confirma el funcionamiento del robot Hibou.

E. Disparador derecho, manipulación en Z del robot Hibou.

4.3 Movilidad de los dos robots Lapbots:

Una vez iniciada la aplicación es importante aclarar el funcionamiento de la interfaz, así como también la manipulación de los LapBots. En la imagen se puede observar los efectores finales de los LapBots izquierdo y derecho (figura 55,) los cuales son manipulados por la mano izquierda y derecha respectivamente (figura 56). Los efectores finales se mueven de acuerdo a como se mueve la mano, es decir si inicia un movimiento con la mano desde la izquierda hacia la derecha el robot actúa igual, de igual forma de arriba abajo y atrás adelante.

Figura 55. Cámara interna del Hibou con robots quirúrgicos en posición normal.

Figura 56. Manos del usuario en posición normal.

A continuación se muestran los efectores finales formando una cruz (figura 57), con el LapBot izquierdo por encima del derecho de acuerdo a la ubicación de las manos (figura 58). Además se puede observar que no hay choques ni monturas entre los efectores finales.

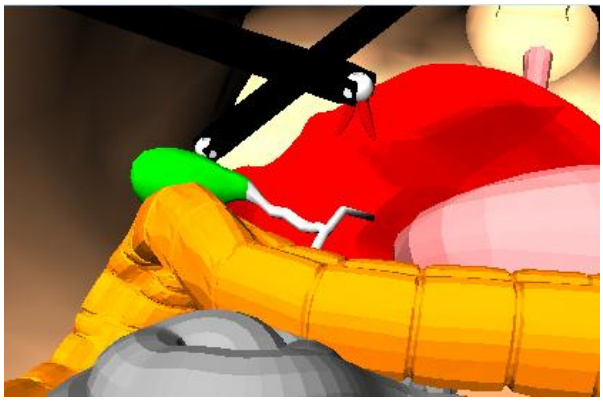


Figura 57. Cámara interna del Hibou con robots quirúrgicos en posición de cruz.

Figura 58. Manos del usuario en posición de cruz.

Para la captura de los datos utilizados en la movilidad de los robots se hace uso de las herramientas del SDK, en este caso las funciones:

- SkeletonPositions[NUI_SKELETON_POSITION__HAND_RIGHT]
- [NUI_SKELETON_POSITION__HAND_LEFT]

Estas funciones proporcionan las coordenadas X, Y, Z de la mano derecha e izquierda, solo se debe agregar la coordenadas que se desee obtener al final de la función. Esta información dada en metros se cambia de escala y luego es enviada al MGI de cada robot para que efectué el movimiento que está recibiendo. Cuando se realiza el envío de las 6 coordenadas es necesario hacer una delimitación de los valores, debido a que el espacio de trabajo del usuario es más grande que el de la simulación. Cada coordenada posee un valor máximo y mínimo que garantiza la estadía de los robots dentro del ambiente de trabajo evitando que se salgan.

4.4 Cambio de instrumento quirúrgico:

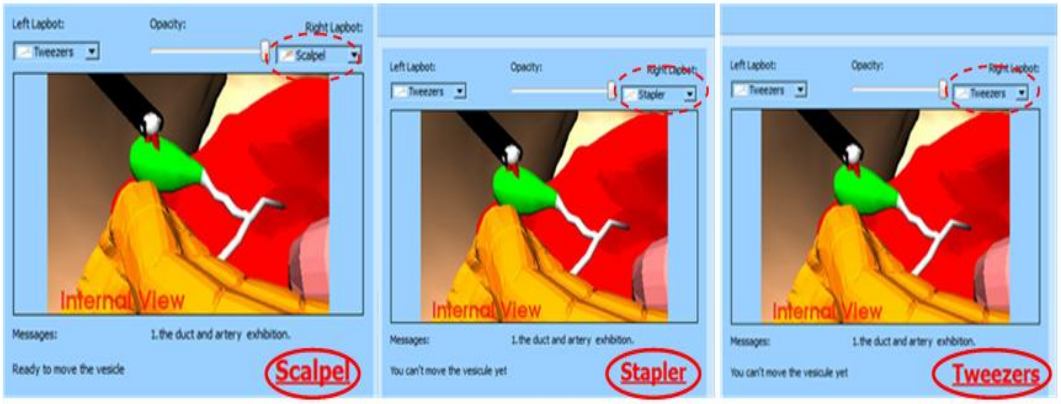


Figura 59.Mensaje cambio de instrumentos.

Figura 60. Gesto para cambio de instrumentos.

Lo correspondiente a la manipulación de los robots Lapbot se hace por medio del dispositivo Kinect, el cambio de instrumentos en el robot derecho se efectúa gracias a la lectura de la rodilla derecha que al ser levantada más de 0.06 m realiza el cambio de bisturí a grapadora, de grapadora a pinzas y de pinzas a grapadora cada vez que se levante, formando así un ciclo. Se podrá elegir cualquiera de los tres instrumentos siempre y cuando el usuario esté en la opción de colecistectomía y el cambio se verá representado en un mensaje de color rojo ubicado encima de la imagen de la cámara interna y en la parte superior derecha de la pantalla.

Para la captura de la posición de la rodilla se utiliza la función del SDK SkeletonPositions[NUI_SKELETON_POSITION__KNEE_RIGHT], estos valores de distancias están dados en metros y son utilizados en el método de equipamiento de instrumentos de los robots. Con contadores y afirmaciones se forma un ciclo que permite recorrer los 3 instrumentos cada vez que la rodilla esté arriba.

4.5 Simulación de una cirugía en RoboSurgery.

La cirugía a realizar es un procedimiento laparoscópico llamado colecistectomía. En la aplicación se encuentran dos opciones a elegir, la primera de ellas es una laparoscopia diagnóstica realizada por el robot Hibou, donde se permite observar el interior del paciente para dar una valoración, en la segunda se encuentra la colecistectomía en la cual se controlan los tres robots (2 Lapbot, 1Hibou) con el fin de extraer del paciente la vesícula enferma.

Etapas para la cirugía en un paciente hombre.

1. Coger vesícula:

Se debe posicionar la cámara en el lugar donde se pueda observar la vesícula biliar y sus arterias. Para esta tarea hay que usar el robot Hibou, que se activa oprimiendo el botón B seguido del *bumper* derecho del joystick, con estas dos órdenes el robot estará listo para manipularse y ser llevado al punto deseado con ayuda del *stick* analógico izquierdo.

Con la cámara posicionada en el sitio correcto se procede a activar los robots Lapbots. Oprimiendo el botón X los robots recibirán la información leída por el Kinect y podrán ser controlados por las manos del usuario. El objetivo es llevar el Lapbot izquierdo cargado con el instrumento pinzas (instrumento por defecto) hasta la posición de la vesícula para agarrarla (figura 61, figura 62).

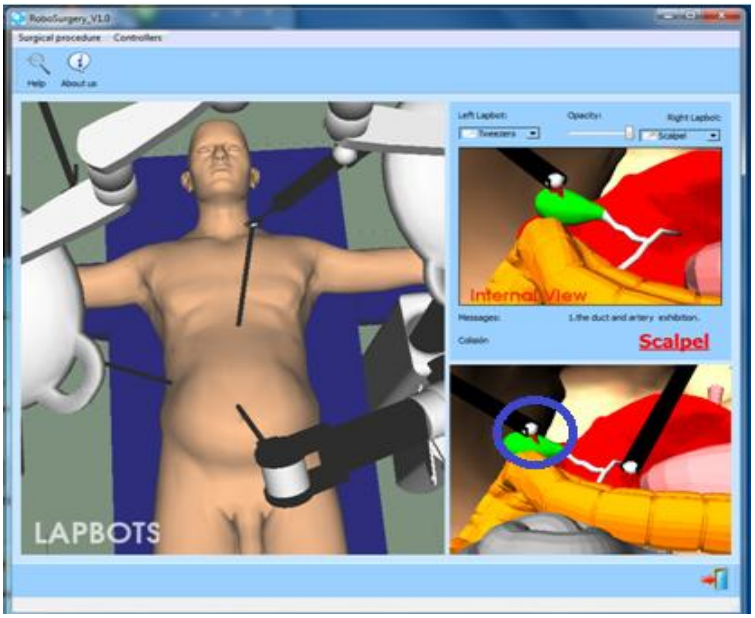


Figura 61.Coger vesícula con Kinect.

Figura 62.Posición para coger de vesícula.

2. Colocar grapas en las arterias:

Para colocar las grapas el robot derecho debe tener el instrumento grapadora (*stapler*). Esta opción se cambia al levantar la rodilla derecha 0.06 m, después de escoger la herramienta se debe dirigir el robot derecho a la parte inferior de la arteria como muestra la figura 63 hasta ver la grapa de color negro puesta en la arteria, después ir a la parte superior y ver la segunda grapa (figura 65).

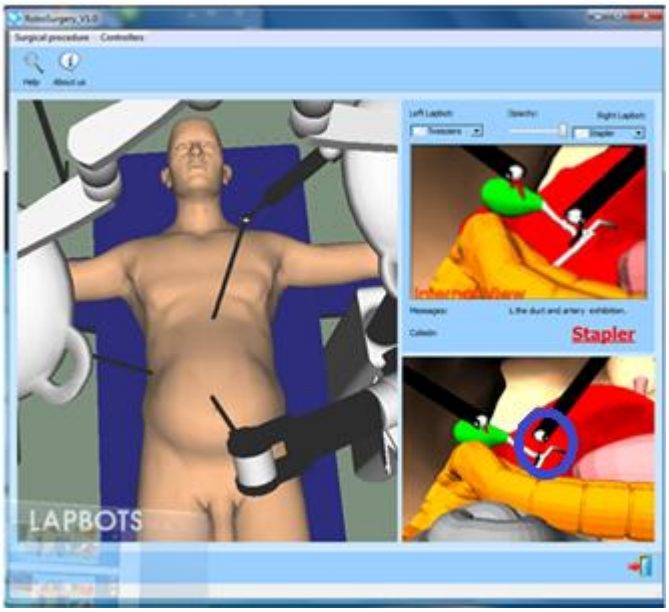


Figura 63.Colocación grapa 1 con Kinect.

Figura 64. Posición para colocar grapa 1.

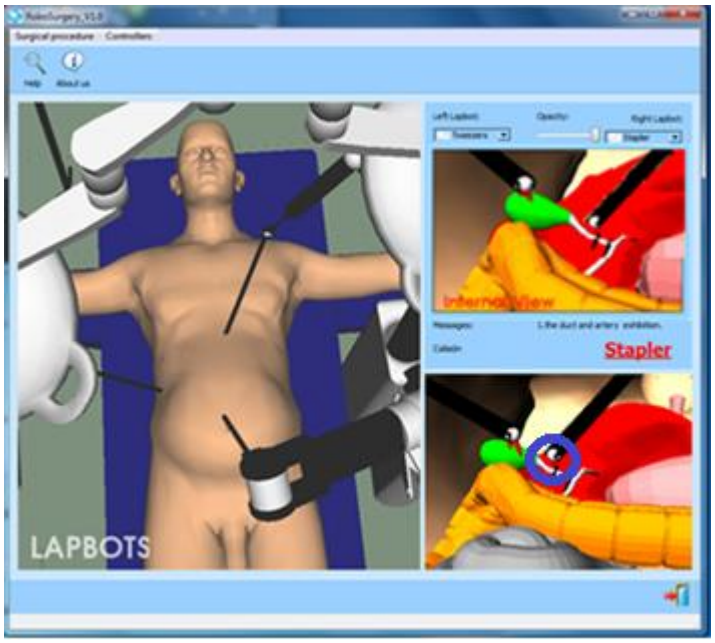


Figura 65.Colocación grapa 2 con Kinect.

Figura 66. Posición para colocar grapa 2.

3. Extracción de vesícula:

Con las grapas puestas se logra cortar el flujo de sangre y se procede al corte y extracción de la vesícula biliar, se cambia el instrumento del Lapbot derecho a bisturí y cuando la arteria este roja esta lista para extraer.

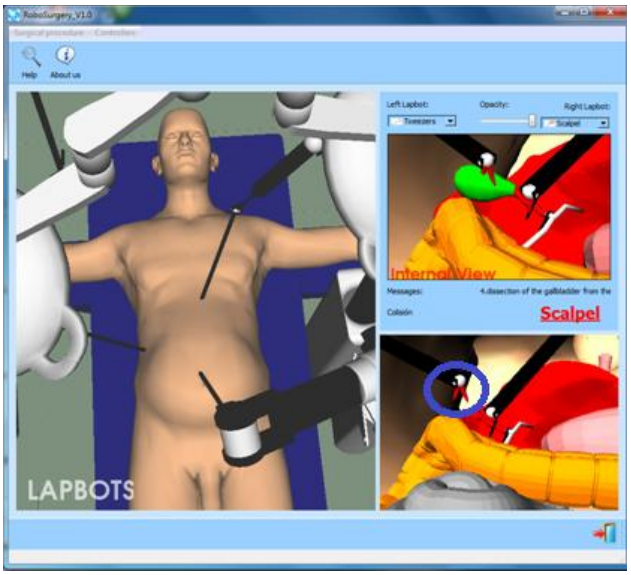


Figura 67. Extracción de vesícula con Kinect.

Figura 68.Posición para extracción de vesícula.

4.6 Movilidad del robot físico Hibou.

El robot porta endoscopio Hibou fue construido físicamente en la Universidad del Cauca en un trabajo de pregrado en el año 2013 [64], y al igual que el simulador RoboSurgery utiliza un joystick para manipular su efector final. Se hace uso de este trabajo para acoplar el robot real y el dispositivo Kinect, logrando así dirigirlo mediante un sistema de interface natural de usuario. El software del Hibou incluye la plataforma Visual Studio 2008 y un lenguaje de programación C++. Al agregar el dispositivo Kinect al sistema se migra a Visual Studio 2010 y se hace uso de las líneas de código correspondientes al dispositivo como se hizo en RoboSurgery. Igualmente se modifica la escala de los datos que van a ser enviados a la tarjeta de adquisición para que los movimientos sean un poco más suaves, y así no dañar los motores de las articulaciones del robot real.

El Hibou logra hacer varios movimientos como se aprecia en la figura 69 y 70, donde el efector final se desplaza verticalmente aproximadamente 25 cm al mover la mano derecha. Con un punto de color azul se puede apreciar el efector final y con un círculo grueso de color rojo el área de la mano.

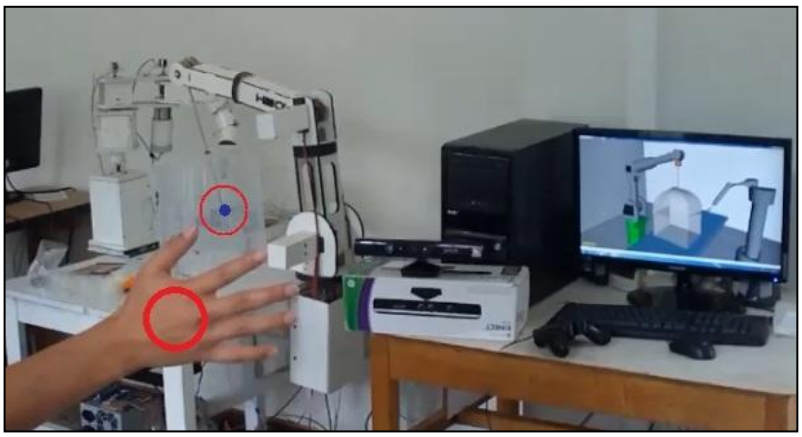


Figura 69. Efector final del Hibou en la parte inferior.

Figura 70. Efector final del Hibou en la parte superior.

En la figura 71 y 72 el efector final realiza movimientos horizontales al mover la mano de derecha a izquierda. El punto azul que representa el efector se ha trasladado aproximadamente 10 cm, se aprecia como el cabezal del robot se mueve en sentido contrario al movimiento de las manos.

Figura 71. Efector final del Hibou en la parte derecha.

Figura 72. Efector final del Hibou en la parte izquierda.

El robot actualmente no posee un control adecuado para su movimiento, lo que ocasiona que al intentar seguir la mano este lo haga con movimientos un poco bruscos.

5 CONCLUSIONES Y TRABAJOS FUTUROS

5.1 Conclusiones.

El software RoboSurgery fue un proyecto de grado de la Facultad de Ingeniería Electrónica y Telecomunicaciones presentado a mediados del año 2013, es una herramienta para la práctica y experimentación de la robótica quirúrgica, cuenta

con dos robots que se manipulan por joystick y realizan una cirugía laparoscópica en un ambiente virtual. Con base en él, este proyecto de captura de gestos para la manipulación de robots quirúrgicos es una primera aproximación hacia la cirugía con interfaces naturales de usuario y es la primera versión de un prototipo así en la Universidad del Cauca.

Los gestos o movimientos de las manos de un cirujano en los sistemas de cirugía laparoscópica en la mayoría de ocasiones son leídas por palancas, estos elementos exigen que haya un contacto directo con el cirujano. Con el dispositivo Kinect dentro del software RoboSurgery se elimina parcialmente la interacción física entre el usuario y el sistema, se logran trasmitir los movimientos del cirujano por medio del dispositivo para la manipulación natural del sistema, es decir que no se necesita un aprendizaje previo para que los robots hagan lo que nuestras manos les pidan.

La tecnología Kinect es una herramienta que facilitó la programación, la toma de datos y el cumplimiento de varias tareas en el sistema, con este dispositivo la culminación de los objetivos se hicieron de una forma más rápida. Los datos que proporcionó el Kinect de las posiciones de las manos fueron buenos, lo que permitió alcanzar resultados admisibles a la hora de realizar la cirugía laparoscópica virtual. Se logró ver que el error aproximado de una trayectoria es de más o menos 0.02 metros y que al cambiar de escala dentro del programa este valor disminuye aún más. El acople entre los dos sistemas utilizados en el proyecto demandó gran trabajo ya que los robots tenían una estructura y una programación establecida, y se debían hacer las respectivas modificaciones para que todo funcionara con una misma lógica.

5.2 Trabajos futuros.

Culminado el proyecto se ve la necesidad de versiones futuras capaces de procesar la información de una forma más rápida, permitiendo así aumentar el número de funcionalidades con el dispositivo Kinect, Se pueden implementar de igual forma más robots e instrumentos que solucionen y faciliten las diferentes problemáticas de las cirugías, se podría pensar en utilizar la programación por hilos para la herramienta RoboSurgery agilizando su procesamiento.

Se plantea la opción de introducir nuevos dispositivos de interfaz natural de usuario ya que cada día en el mercado mundial se lanzan dispositivos con mejores

características. La idea es experimentar con alguno de ellos para mejorar el software en su robustez y precisión.

Existen Robots académicos reales realizados en la Universidad del Cauca con características muy similares a los virtuales, se podría implementar un software donde exista un acople entre los dos sistemas logrando manipular los robots reales por medio del Kinect.

6 BIBLIOGRAFÍA

[1] A. Córdoba y G. Ballantyne, «Sistemas quirúrgicos robóticos y telerobóticos para cirugía abdominal,» *REVISTA GASTROENTEROL,* vol. 23, pp. 58-66, 2003.

[2] J. Marescaux and F. Rubino, "The ZEUS robotic system: experimental and clinical applications," *REVISTA SURGICAL CLINICS OF NORTH AMERICA,* vol. 83, pp. 1305-1315, 2003.

[3] «Cirugía Robótica,» [En línea]. Available: http://cirugiaroboticaull.blogspot.com/. [Último acceso: 11 Septiembre 2013].

[4] «Intuitive Surgical,» [En línea]. Available: http://www.intuitivesurgical.com/products/davinci_surgical_system/. [Último acceso: 9 Septiembre 2013].

[5] A. Madhani, G. Niemeyer and K. Salisbury, "The Black Falcon: A Teleoperated Surgical Instrument," in *Conference on Intelligent Robots and Systems*, Victoria, Canada, 1998.

[6] B. Estebanez, P. Orozco, I. García y V. Muñoz, «Interfaz multimodal para un asistente robótico quirúrgico: uso de reconocimiento,» *REVISTA RIAI,* vol. 8, nº 2, pp. 24-32, 2011.

[7] H. Kang and J. Wen, "EndoBot: a robotic assistant in minimally invasive surgeries," in *International Conference on Robotics and Automation*, Seul,Corea, 2001.

[8] «All About Robotic Surgery,» [En línea]. Available: http://www.allaboutroboticsurgery.com/surgicalrobots/surgicalrobotspage2.html. [Último acceso: 13 Septiembre 2013].

[9] E. Lanzarini, V. Schonstedt, M. Abedrapo, J. Yarmuch, A. Csendes y A. Rodríguez, «Simulación: Una herramienta útil en la formación quirúrgica moderna,» *Revista Chilena de Cirugía,* vol. 60, nº 2, pp. 167-169, 2008.

[10] «Simulador para el entrenamiento en cirugías avanzada,» [En línea]. Available: http://users.dsic.upv.es/~cmonserr/Articulos/AA028.pdf. [Último acceso: 14 Septiembre 2013].

[11] «SIMULAB CORPORATION,» [En línea]. Available: http://www.simulab.com/home-traumaman. [Último acceso: 15 Septiembre 2013].

[12] «SIMULAB CORPORATION,» [En línea]. Available: http://www.simulab.com/product/surgery/laparoscopic/laptrainer-simuvision. [Último

acceso: 15 Septiembre 2013].

[13] «ProDelphus,» [En línea]. Available: http://www.prodelphus.com.br/websiteSpanish/products/show.asp?prdCode=D5C29AAD-EFEE-4138-85CC-D95773C21F8F. [Último acceso: 11 Septiembre 2013].

[14] «Medical Simulator,» [En línea]. Available: http://www.medical-simulator.com/base.asp?idProducto=184&idFamilia=126&idFamiliaPadre=90. [Último acceso: 1 Septiembre 2013].

[15] «Medical Simulator,» [En línea]. Available: http://www.medical-simulator.com/base.asp?idProducto=2465&idFamilia=126&idFamiliaPadre=90. [Último acceso: 2 Septiembre 2013].

[16] «Medical Simulator,» [En línea]. Available: http://www.medical-simulator.com/base.asp?idProducto=2466&idFamilia=126&idFamiliaPadre=90.. [Último acceso: 2 Septiembre 2013].

[17] «Simbionix,» [En línea]. Available: http://simbionix.com/simulators/lap-mentor/. [Último acceso: 18 Septiembre 2013].

[18] A. Zhang, M. Hünerbein, Y. Dai, P. Schlag and S. Beller, "Construct validity testing of a laparoscopic surgery simulator (Lap Mentor): evaluation of surgical skill with a virtual laparoscopic training simulator," *Surgical Endoscopy,* vol. 22, no. 6, pp. 1440-1444, 2008.

[19] «MENTICE,» [En línea]. Available: http://www.mentice.com/our-simulators/vist-c/. [Último acceso: 10 Septiembre 2013].

[20] «MENTICE,» [En línea]. Available: http://www.mentice.com/our-simulators/vist-lab/. [Último acceso: 10 Septiembre 2013].

[21] «SIMMED,» [En línea]. Available: http://www.simmedsa.com/promis.html. [Último acceso: 20 Septiembre 2013].

[22] «Surgical Science,» [En línea]. Available: http://www.surgical-science.com/lapsim-the-proven-training-system/. [Último acceso: 25 septiembre 2013].

[23] «Universidad EAFIT,» 2009. [En línea]. Available: http://arcadia.eafit.edu.co/html/simulador.html. [Último acceso: 25 Septiembre 2013].

[24] S. Salinas, «Modelado y Simulación en 3D y Control de un Robot para Cirugía Laparoscópica,»

Tesis de Maestría, Universidad del Cauca, Popayán, Colombia, 2009.

[25] D. Guzmán, «Herramienta Software para la Práctica y Experimentación de la Robótica Quirúrgica,» Trabajo de Grado, Universidad del Cauca, Popayán, Colombia, 2013.

[26] «Studying The Wiimote Exploracion To Enhance The Feeling Of Interactivity In Applications,» [En línea]. Available: http://www.cs.vu.nl/~eliens/mma2/@archive/tutorial/wiimote.pdf. [Último acceso: 9 Agosto 2013].

[27] «AOL Inc,» [En línea]. Available: http://es.engadget.com/2010/06/03/playstation-move-probado-con-detenimiento. [Último acceso: 9 Agosto 2013].

[28] «LEAP MOTION,» [En línea]. Available: https://www.leapmotion.com/product. [Último acceso: 15 Septiembre 2013].

[29] «EMOTIV,» [En línea]. Available: https://www.emotiv.com/epoc/. [Último acceso: 20 Septiembre 2013].

[30] «ASUSTeK Computer Inc,» [En línea]. Available: http://event.asus.com/wavi/product/WAVI_Xtion.aspx. [Último acceso: 15 Septiembre 2013].

[31] «eedoo,» [En línea]. Available: http://www.eedoo.cn/product/zj/index.shtml. [Último acceso: 1 octubre 2013].

[32] «PrimeSense,» [En línea]. Available: http://www.primesense.com/wp-content/uploads/2013/02/PrimeSense_3DsensorsWeb.pdf. [Último acceso: 30 Septiembre 2013].

[33] «PrimeSense,» [En línea]. Available: http://www.primesense.com/developers/get-your-sensor/. [Último acceso: 30 Septiembre 2013].

[34] «EPFL,» [En línea]. Available: http://wiki.epfl.ch/la3/ranging.lasers. [Último acceso: 30 Septiembre 2013].

[35] «Creative Technology Ltd,» [En línea]. Available: http://us.creative.com/p/web-cameras/creative-senz3d. [Último acceso: 2 Octubre 2013].

[36] «3djuegos,» Kinect, [En línea]. Available: http://www.3djuegos.com/foros/tema/860680/0/todo-sobre-kinect-el-proyecto-natal/.

[37] Y. Li, «Hand Gesture Recognition Using Kinect,» Tesis de maestria, University of China, China,

2010.

[38] S. Lang, «Sign language Recognition whit Kinect,» Tesis de licenciatura, Freie Univesität Berlin, Berlin, Alemania, 2011.

[39] K. Jahrmann, «3D Reconstruction with the Kinect-Camera,» Tesis de licenciatura ,Technischen Universität Wien, Viena, Austria , 2013.

[40] S. Mørkved Albrektsen, «Using the Kinect Sensor for Social,» Tesis de maestria, University Norwegian , Trondheim, Noruega, 2011.

[41] E. Trilles, «Desarrollo de interfaces de usuario naturales con Kinect,» Proyecto final de carrera, Ingeniería Informática, Universidad Politécnica de Valencia, Valencia, España, 2012.

[42] A. Morcillo, «Efectos de un programa de realidad virtual sobre la marcha y el control postural en personas adultas con parálisis cerebral,» Trabajo final de Maestría, Master Universitario en Atención Fisioterápica en la Actividad Física y el Deporte, Valencia, España, 2013.

[43] S. Pfeiffer, «Guiado gestual de un robot humanoide mediante un sensor Kinect,» Proyecto final de carrera, Ingeniería Técnica en Informática de Sistemas, Universidad Politécnica de Cataluña, Barcelona, España, 2011.

[44] L. Mathe, D. Samban y G. Gómez, «Estudio del funcionamiento del sensor Kinect y aplicaciones para bioingeniería,» [En línea]. Available: http://argencon.org.ar/sites/default/files/123.pdf. [Último acceso: 5 julio 2013].

[45] W. Zeng, «Microsoft Kinect Sensor and Its Effect,» *Multimedia at Work,* vol. 19, pp. 4-10, 2012.

[46] Y. Chang, B. Lange, M. Zhang, S. Koenig, P. Requejo, N. Somboon, A. Sawchuk and A. Rizzo, "Towards Pervasive Physical Rehabilitation Using Microsoft Kinect," Pervasive Computing Technologies for Healthcare (PervasiveHealth), San Diego, California, 2012.

[47] D. Martínez Capilla, «Sign Language Translator using Microsoft,» Tesis de maestria ,University of Tennessee, Knoxville - USA, 2012.

[48] K. LaBelle, «Evaluation of Kinect Joint Tracking for Clinical and In-Home Stroke Rehabilitation Tools,» Tesis de pregrado, University of Notre Dame, Indiana- USA, 2011.

[49] M. León y J. Paz, «Generación de Trayectorias para un Robot Puma Simulado a Partir de Captura de Movimiento de la Mano,» Tesis de pregrado, Universidad del Cauca, Popayan,

Cauca, 2013.

[50] D. Rowan, «Kinect for Xbox 360: The inside story of Microsoft's secret 'Project Natal',» *Wired,* 2010.

[51] «WebAcademia 2013,» [En línea]. Available: http://centrodeartigos.com/articulos-enciclopedicos/article_96478.html. [Último acceso: 15 Septiembre 2013].

[52] M. Á. de Frutos, «AEROBOT,» 14 Junio 2011. [En línea]. Available: http://aerobotclubderobticadeaeronuticos.blogspot.com/2011/06/kinect-esta-de-moda.html. [Último acceso: 15 Septiembre 2013].

[53] R. Vílchez, «Funcionalidades de Audio de Kinect,» Proyecto Fin de Carrera, Ingeniería Informática Superior, Universidad Carlos III de madrid, Madrid, España, 2012.

[54] «Business Insider,» [En línea]. Available: http://www.businessinsider.com/microsoft-releases-kinect-sdk-2011-6. [Último acceso: 5 julio 2013].

[55] A. Davison, Kinect Open Source Programming Secrets, McGraw-Hill Professional, 2012.

[56] «Maleny,» [En línea]. Available: http://malenyabrego.wordpress.com/2012/10/22/sabias-que-kinect-para-windows-puede-escuchar-y-puede-verte/. [Último acceso: 28 Agosto 2013].

[57] E. Fernández, «Control de Software Educativo Mediante Kinect de Microsoft,» Trabajo Fin de Grado, Universidad Carlos III de Madrid, Madrid, España, 2012.

[58] M. R. Andersen, T. Jensen, P. Lisouski, A. K. Mortensen, M. K. Hansen, T. Gregersen and P. Ahrendt, "Kinect Depth Sensor Evaluation for Computer Vision Applications," *Revista Aarhus University,* 2012.

[59] Openkinect, [En línea]. Available: http://openkinect.org. [Último acceso: 1 Septiembre 2012].

[60] OpenNi, [En línea]. Available: http://openni.org. [Último acceso: 14 Septiembre 2012].

[61] «Yannick Loriot,» [En línea]. Available: http://yannickloriot.com/wp-content/uploads/2011/03/OpenNi-Architecture.png. [Último acceso: 14 Septiembre 2013].

[62] Microsoft Corporation, «Microsoft,» [En línea]. Available: http://www.microsoft.com/en-us/kinectforwindows/. [Último acceso: 2 Septiembre 2013].

[63] «msdn.microsoft,» [En línea]. Available: http://msdn.microsoft.com/es-

es/library/fx6bk1f4(v=vs.90).aspx. [Último acceso: 7 Agosto 2012].

[64] J. Samboni y C. Fuentes, «Base para Sistema de Entrenamiento Quirurgico: Robot Hibou,» Tesis de pregrado, Universidad del Cauca, Popayan, Cauca, 2013.

ANEXO A
INSTALACIÓN DE BIBLIOTECAS

1. Instalación de QT

1.1. Requisitos

Para la correcta instalación de la biblioteca son necesarias las instalaciones previas en el ordenador del siguiente software:

- Sistema operativo Windows Seven.
- Visual Studio 2010 ultimate.
- Herramienta de compresión de archivos.
- InstaladoresQT.rar.

1.2. Procedimiento

El primer paso es descomprimir los archivos contenidos en la carpeta InstaladoresQT.rar, carpeta en la cual se encuentran las versiones compatibles para Visual Studio 2010, posteriormente se instala qt-win-opensource-4.8.4-vs2010.exe, como se muestra en las siguientes figuras:

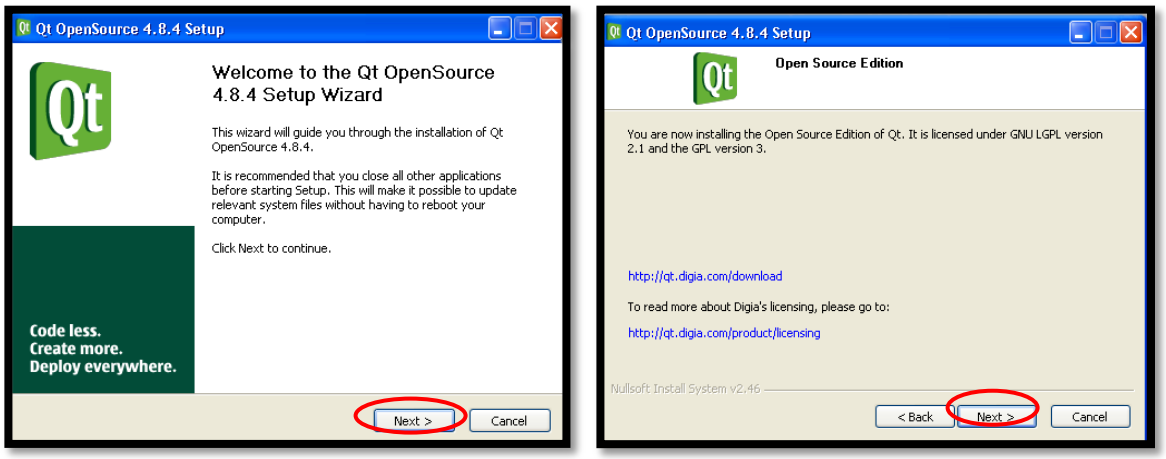


Figura 1. Instalación de QT.

Ahora se acepta las condiciones y términos de la licencia marcando la opción "*I accept the terms of the License Agreement*" y luego siguiente "*Next*" hasta terminar.

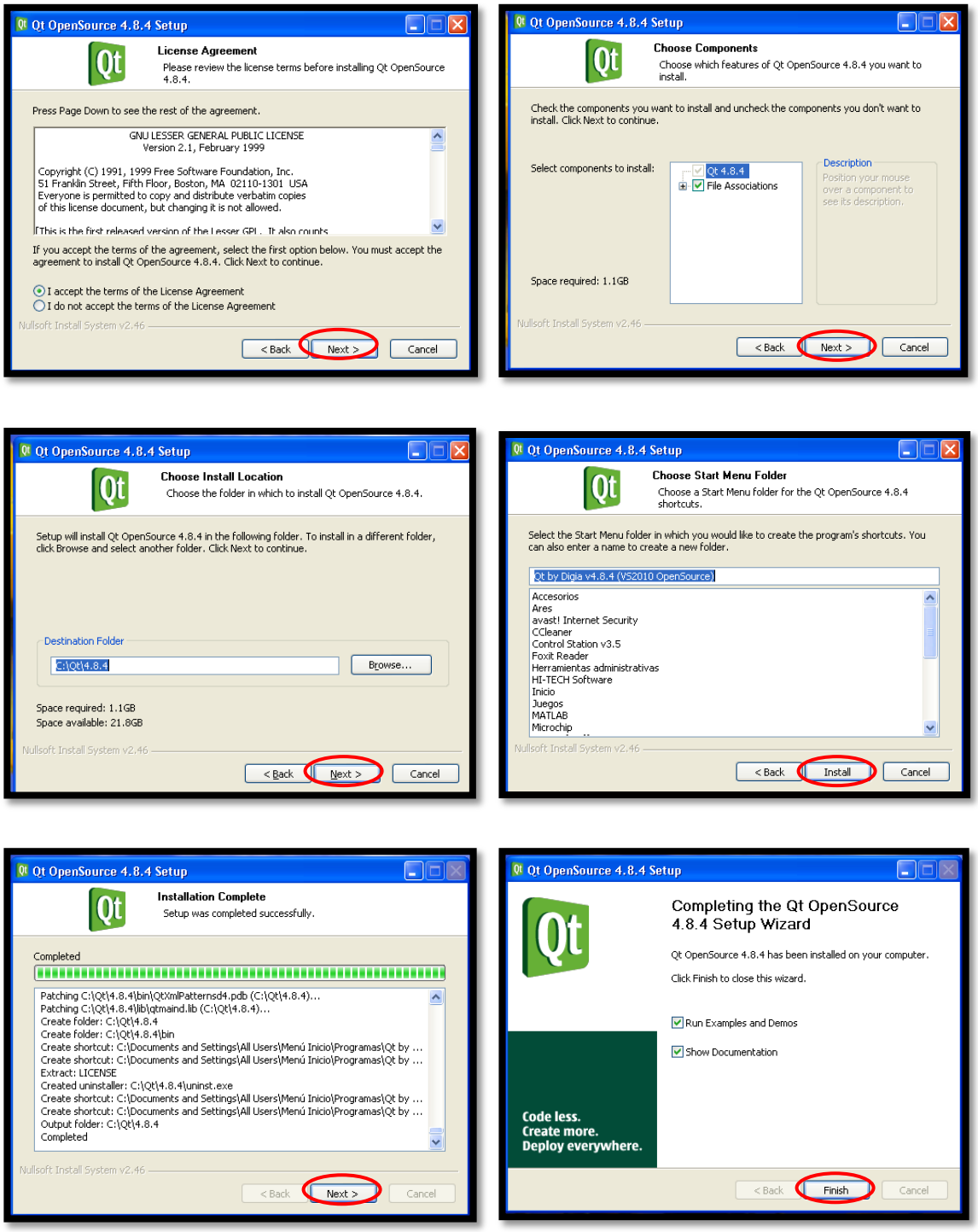


Figura 2. Acuerdo de licencia y finalización.

Una vez instalado, tenemos listo las librerías y archivos de compilación necesarios para generar la ventana secundaria donde se visualizaran los robots, sin embargo,

es necesario integrar estas librerías con las de Visual Studio 2010, el cual es nuestro entorno de desarrollo; para esto se ejecuta el archivo “qt-vs-addin-1.1.11-opensource.exe”:

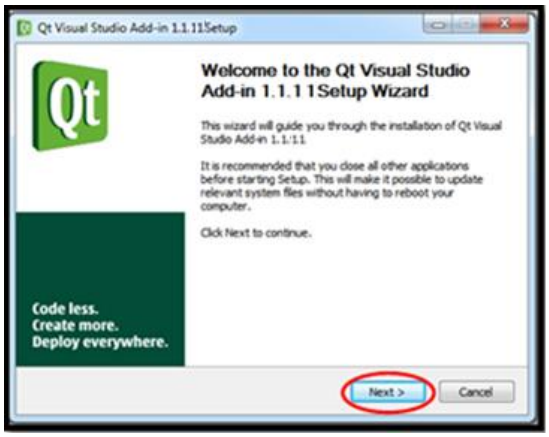
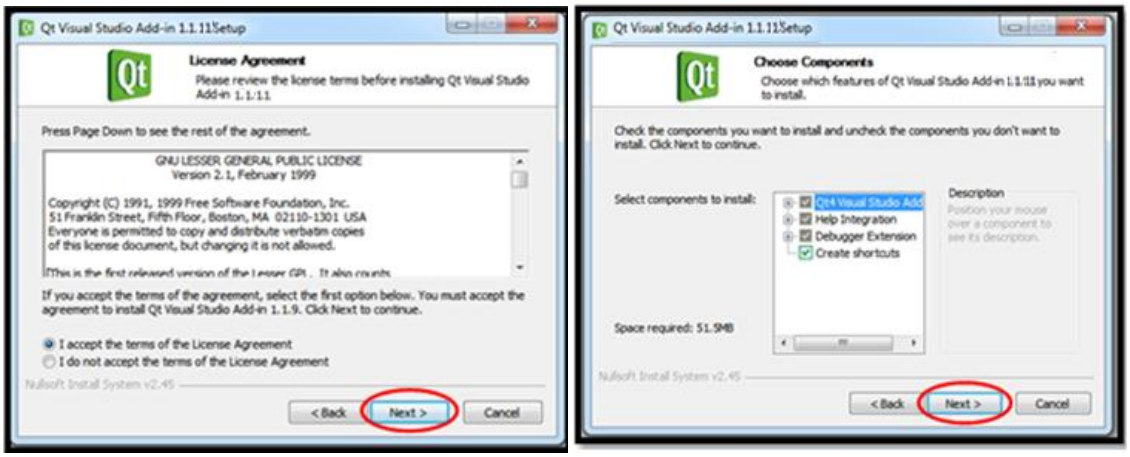


Figura 3. Inicialización del vinculador de QT con Visual Studio.

Al igual que el anterior instalador, es necesario aceptar las condiciones y términos de la licencia, marcando la opción “*I accept the terms od the License Agreement*” y siguiente.



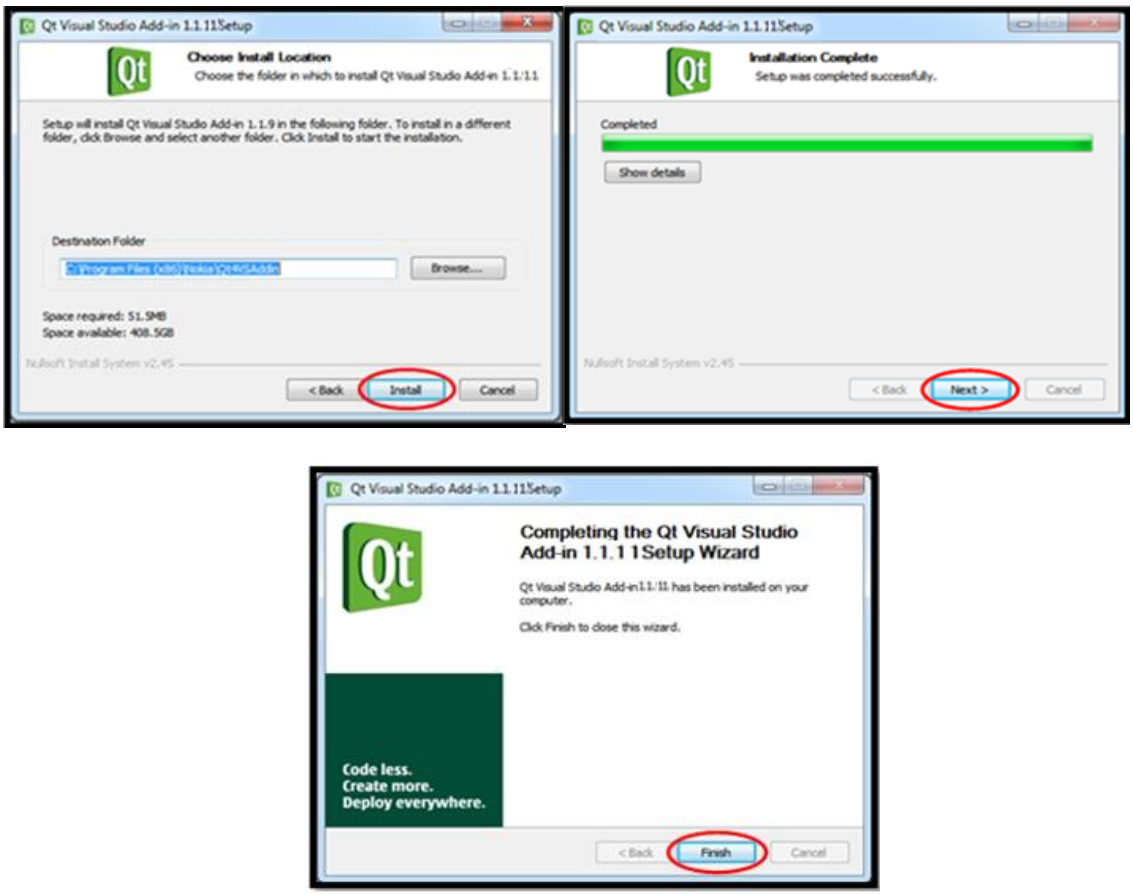


Figura 4. Acuerdo de licencias y finalización.

Finalmente hay que definir las variables de entorno del sistema, para esto accedemos desde las siguientes formas:

- Inicio > Panel de control > Sistema y Seguridad > Seguridad >Configuración avanzada del sistema

- Inicio > variables de entorno

En la figura 5. A podemos mirar a la izquierda la ventana desplegada tras hace clic en opciones avanzadas del sistema y posteriormente “variables de entorno”, por otro lado a la derecha después de escribir variables de entorno seleccionamos la segunda opción.

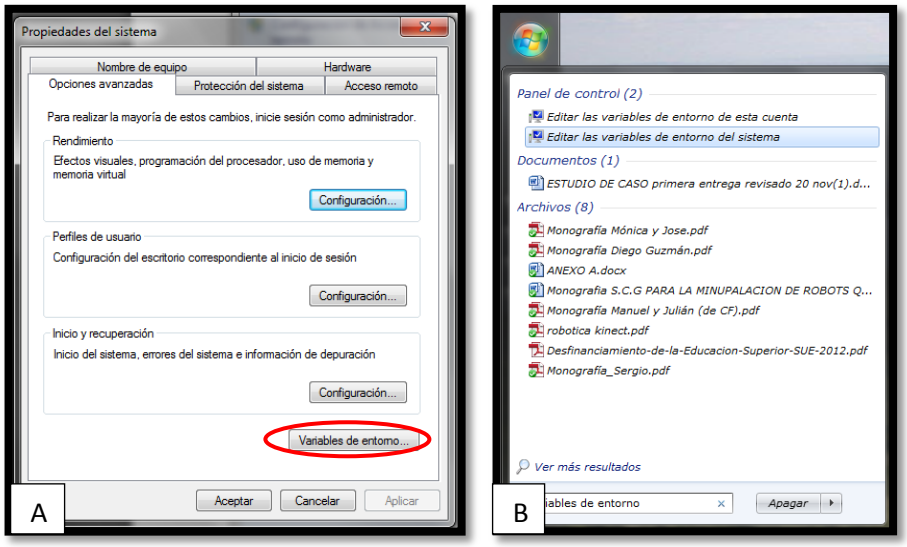


Figura 5. A propiedades de sistema, B variables de entorno desde inicio.

Cuando nos encontremos en la ventana que se muestra en la figura 6, damos clic en nueva.

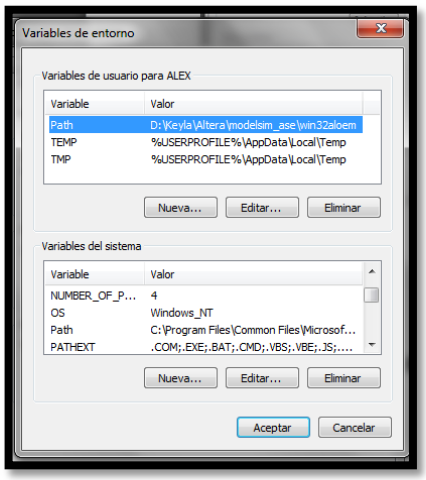


Figura 6. Variables de entorno.

En la nueva ventana se deben ingresar los datos para registrar la nueva variable, estos datos corresponden al nombre y la ubicación o ruta de acceso a él como se muestra en la figura 7, estos datos son:

- Nombre de la variable: QTDIR

- Valor de la variable: C:\Qt\4.8.4

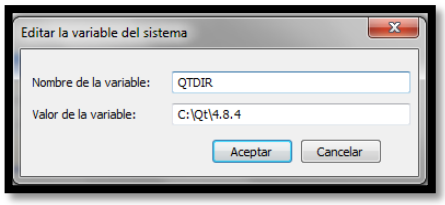


Figura 7. Edición de variables de entorno para QT.

Posteriormente modificamos la variable del sistema *Path*, para esto seleccionamos "*Path*" y clic en editar, como se muestra en la figura 8.

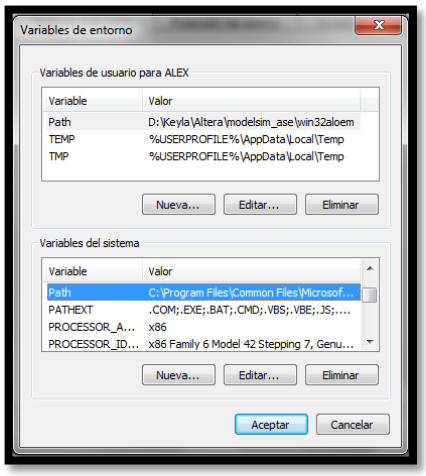


Figura 8. Edición del *path.*

Al dar clic en editar aparecerá una ventana como la observada en la figura 9, donde debemos adicionar al final del valor de la variable lo siguiente:

- C:\Qt\4.8.4\bin

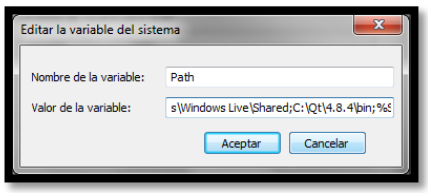


Figura 9. Edición del *path.*

Aceptar y cerrar la ventana de variables del sistema.

2. Instalación de Cmake

2.1. Requisitos

Ninguno

2.2. Procedimiento

La versión utilizada es la 2.8.5, que se puede descargar de la página de cmake, sin embargo, este ejecutable también se encuentra en el archivo InstaladoresQT.rar, de allí ejecutamos "cmake-2.8.5-win32-x86.exe", y en la tercera ventana marcamos la opción "*Add CMake to the system PATH for all users*", y continuamos hasta finalizar.

3. Instalación de VTK

3.1. Requisitos

- Visual studio 2010 ultimate
- Cmake 2.8.5
- Herramienta de compresión de archivos
- QT 4.8.4

3.2. Procedimiento

La librerías de VTK usadas son de uso libre y se pueden descargar desde la página Web oficial, http://www.vtk.org/VTK/resources/software.html. Se descargará un archivo .zip versión 5.8.0, el cual hay que descomprimirlo y almacenarlo en la carpeta VTK5, ubicada en el disco C y crear otra con el nombre VTK_Build.

Mediante el software Cmake compilamos las librerías VTK, seleccionamos las carpetas anteriormente creadas como se muestra en la figura 10, y marcamos la opción "*Advanced*"

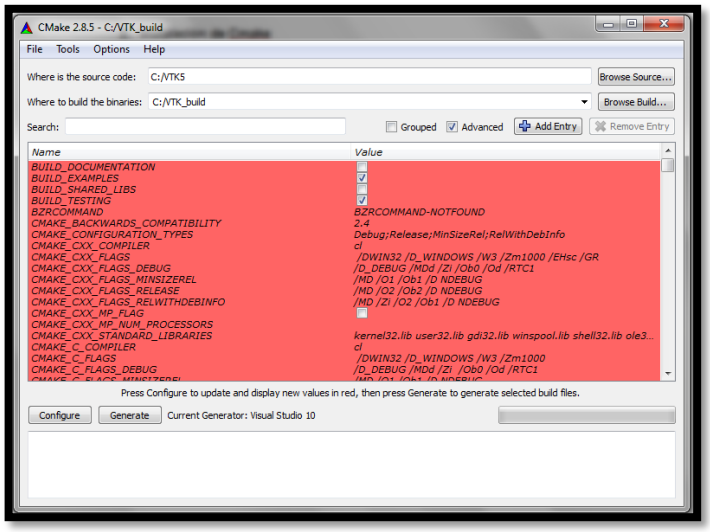


Figura 10. Inicialización de CMake.

Con el primer paso listo, damos clic en “*configure*”, pronto aparecerá una ventana que pide se indique el compilador a utilizar, y seleccionamos Visual Studio 2010. Al terminar la compilación aparecerán algunas líneas sin definir, para esto debemos marcar las opciones:

- BUILD EXAMPLES
- VTK_USE_QT

Con estas dos opciones seleccionadas, damos clic nuevamente en “*configure*”, tras finalizar la compilación marcamos la casilla VTK_USE_QVTK_QTOPENGL, y nuevamente configuramos.

Al finalizar es posible encontrar nuevas casillas en rojo, estas no son tenidas en cuenta y damos clic en configurar, al finalizar no debe aparecer ninguna casilla en rojo y damos clic en “*generate*”, al finalizar correctamente podemos cerrar Cmake.

En la carpeta VTK_Build, podemos encontrar los archivos a ser compilados mediante visual studio. Es de suma importancia ejecutar Visual Studio como administrador y luego buscar en la carpeta anteriormente nombrada el archivo VTK.sln y abrirlo, como se muestra en la figura 11.

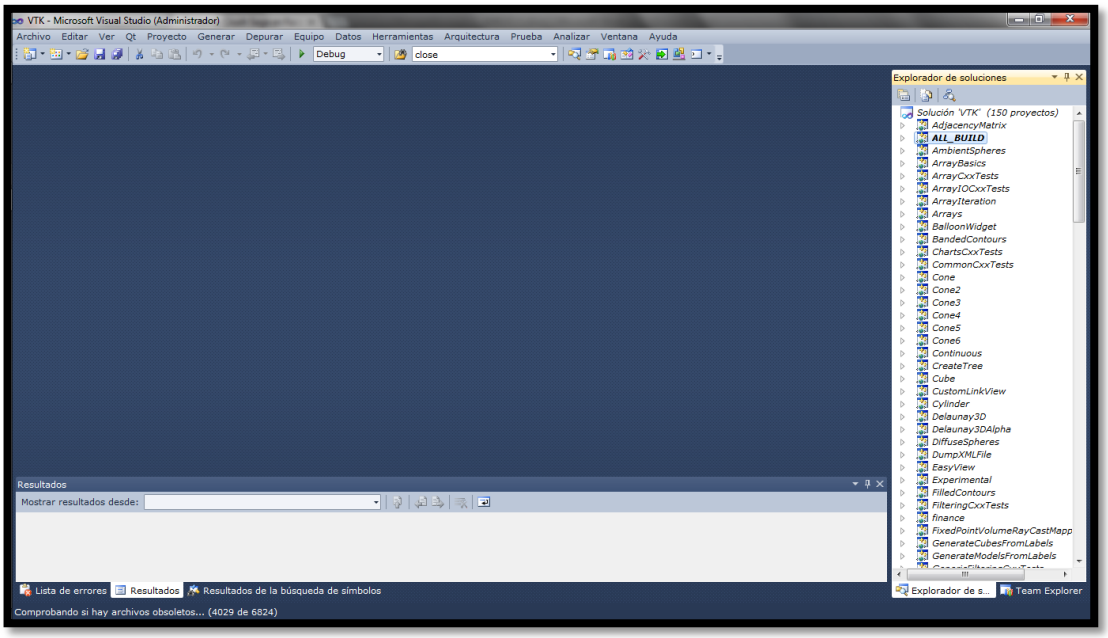


Figura 11. Ventana principal de Visual Studio con VTK.

Para la compilación de la biblioteca, marcamos la opción ALL_BUILD en el explorador de soluciones, y luego generar ALL_BUILD, dentro del menú generar como se muestra en la figura 12, al finalizar en parte inferior de la ventana debe aparecer 0 errores.

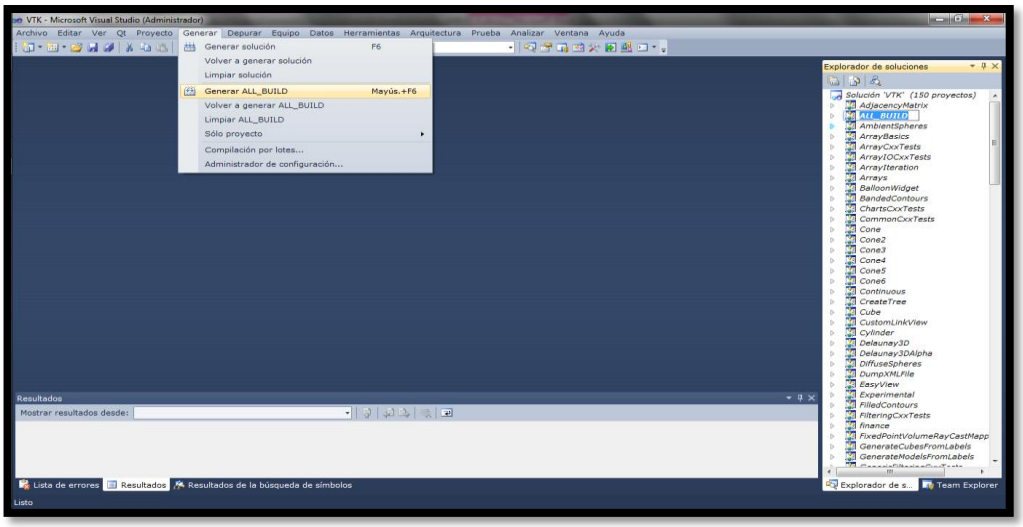


Figura 12. Compilación de VTK.

Para finalizar la instalación de VTK, marcamos la opción INSTALL dentro del explorador de soluciones y de igual forma que el anterior damos generar INSTALL, figura 13.

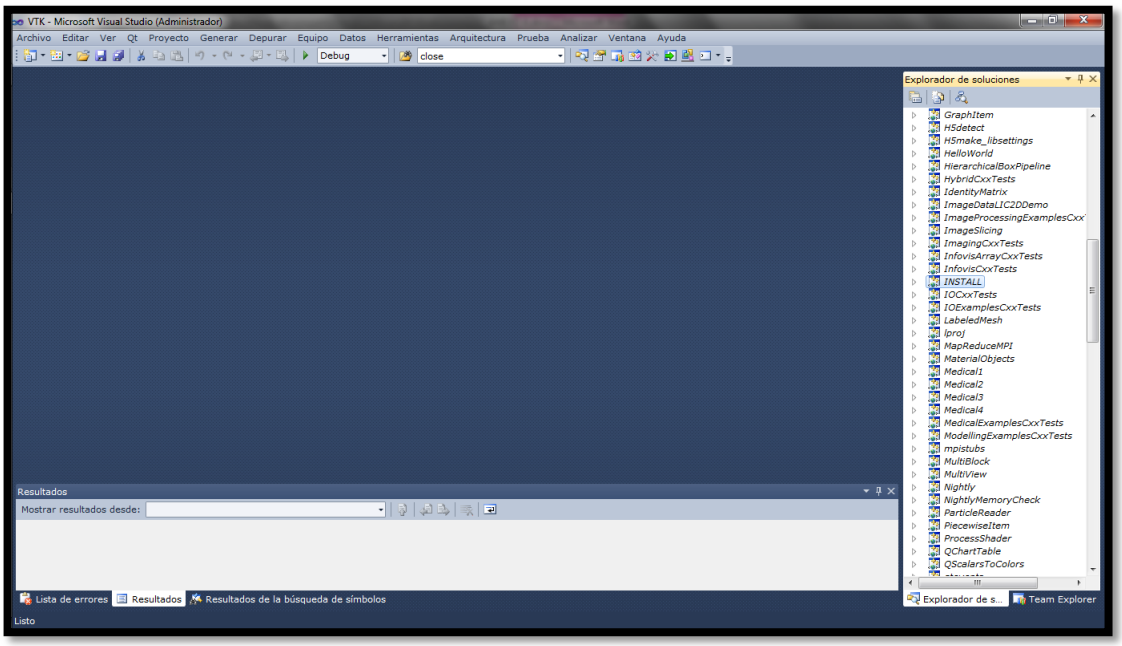


Figura 13. Generar el "*install*" de VTK.

4. Instalación de SDL y VCOLLIDE

4.1. Requisitos

- Herramienta de compresión de archivos

4.2. Procedimiento

Los archivos del SDL se encuentran en la carpeta Instaladores_QT.rar, y son necesarios para el movimiento del joystick; estos deben ser descomprimidos en la carpeta SDL, ubicada en el disco C, posteriormente se definen sus variables de entorno, figura 14, para esto creamos una nueva con los siguientes datos.

- Nombre de la variable: SDLDIR
- Valor de la variable: C:\SDL

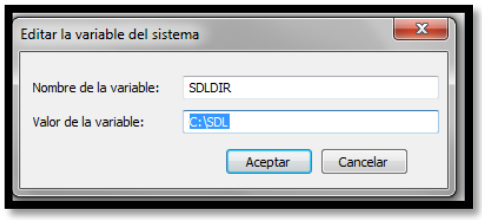


Figura 14. Edición de variable de entorno para SDL.

Del Instaladores_QT.rar extraemos la carpeta VCollide201 y la ponemos en el disco C, esta librería es la que soporta colisiones y deformaciones, para esto es necesario definir sus variables de entorno, figura 15, con los siguientes datos:

- Nombre de la variable: VCOLLIDEDIR
- Valor de la variable: C:\VCollide201

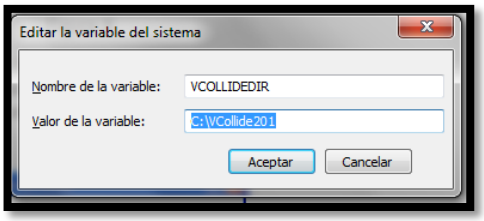


Figura 15. Edición de variable de entorno para Vcollide.

5. Instalación de Kinect

5.1. Requisitos

- Ninguno

5.2. Procedimiento

Para el uso del dispositivo Kinect de Windows, es necesario instalar los SDKs de este, para ellos se puede descargarlos de la página de Windows:

http://www.microsoft.com/en-us/kinectforwindowsdev/start.aspx, donde se encuentra la versión 1.8, sin embargo, la versión usada para el proyecto es la 1.6 contenida en la carpeta Instaladores_qt.rar, para instalarlos ejecutamos el archivo "KinectSDK-v1.6-Setup.exe" y aceptamos las condiciones y usos de la licencia

como se muestra en la figura 16, y finalmente al abrir el administrador de dispositivos encontraremos el dispositivo como se muestra en la figura 17.

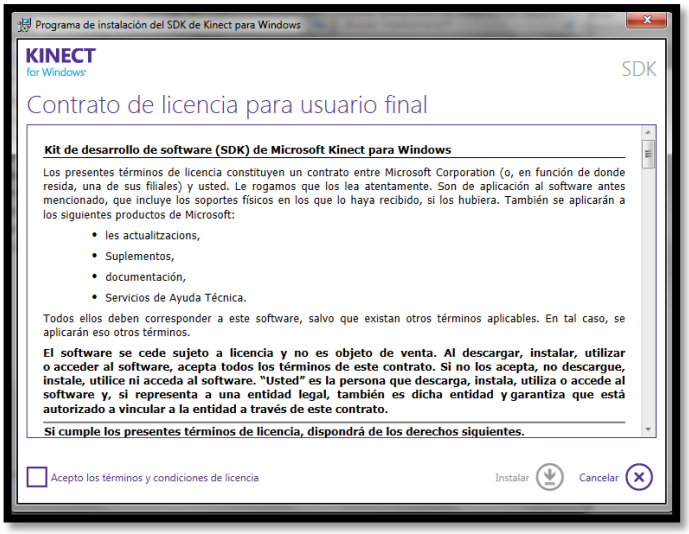


Figura 16. Instalación de SDK para Kinect.

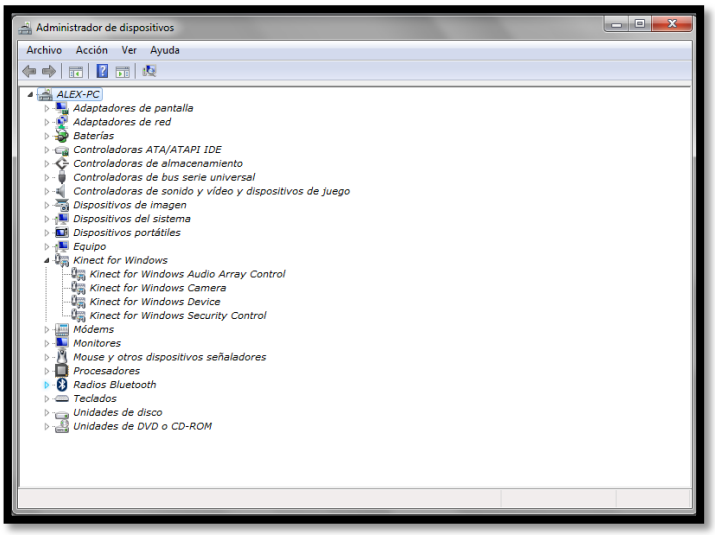


Figura 17. Drives del dispositivo Kinect en administrador de dispositivos.

6. Integración de Kinect a Visual studio

6.1. Requisitos

- Visual Studio 2010

6.2. Procedimiento

Una vez instalado el dispositivo Kinect, necesitamos modificar las propiedades de la aplicación desarrollada, para ello hacemos 3 pasos:

I. Configuración de directorios de C y C++

En el explorador de soluciones, seleccionamos aplicación y presionamos las teclas Alt + *Enter*, para entrar a las propiedades y sobre el menú C/C++ seleccionamos “General” y editamos directorios de inclusión adicionales, figura 18.

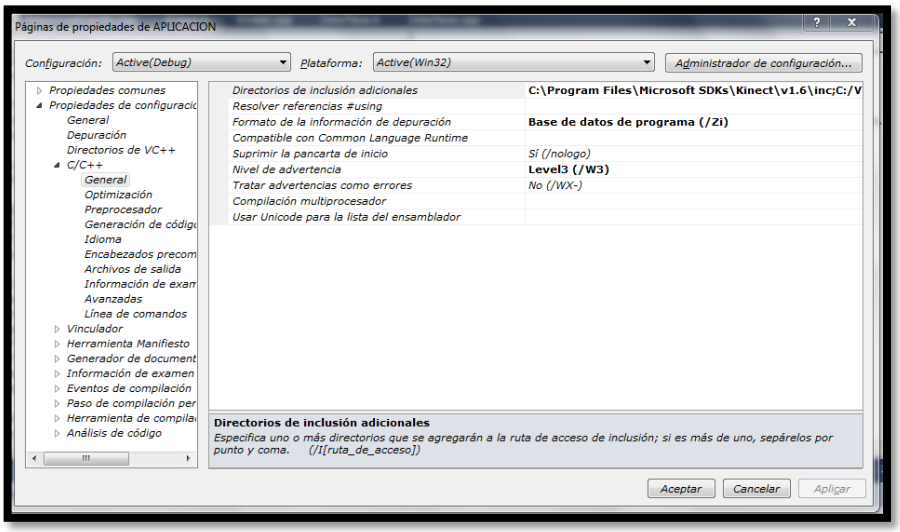


Figura 18. Propiedades del proyecto.

En directorios de inclusión adicionales, damos clic en la parte derecha y en la flecha desplegable seleccionamos editar para crear un nuevo directorio como se muestra en la figura 19 y buscamos el directorio:

- C:\Program Files\Microsoft SDKs\Kinect\v1.6\inc

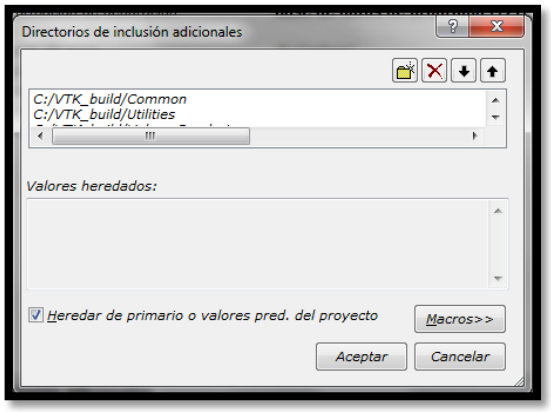


Figura 19. Inclusión de librería del Kinect.

II. Modificación de bibliotecas adicionales

En la parte izquierda sobre el menú “Vinculador” seleccionamos general, luego en la parte derecha deprendemos la flecha desplegable para adicionar la siguiente dirección:

- C:\Program Files\Microsoft SDKs\Kinect\v1.6\lib\x86

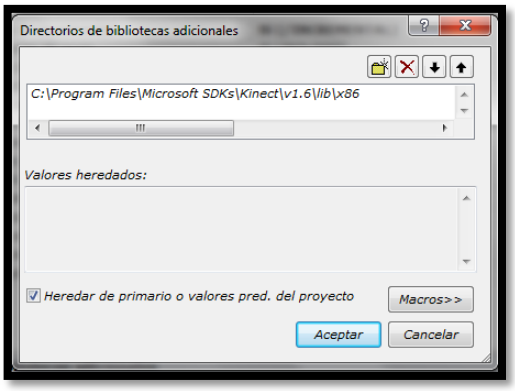


Figura 20. Inclusión de librerías adicionales.

III. Edición de dependencias adicionales

En el menú “Vinculador” seleccionamos “entrada” y editamos las dependencias adicionales, agregando la siguiente línea:

- C:\Program Files\Microsoft SDKs\Kinect\v1.6\lib\x86\Kinect10.lib

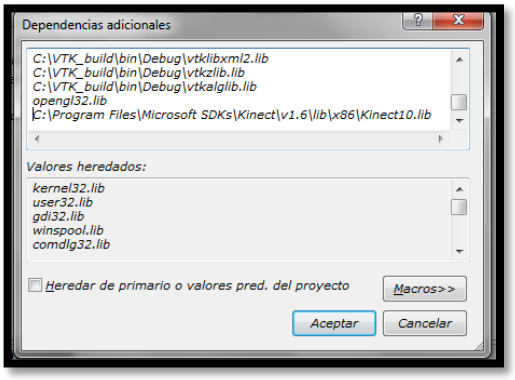


Figura 21. Dependencias adicionales.

ANEXO B

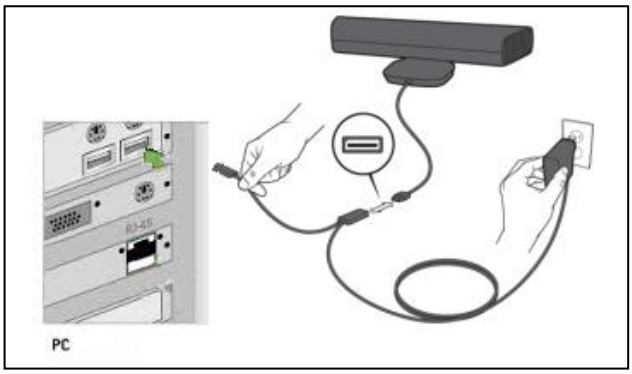
RoboSurgery con Kinect

Manual de usuario

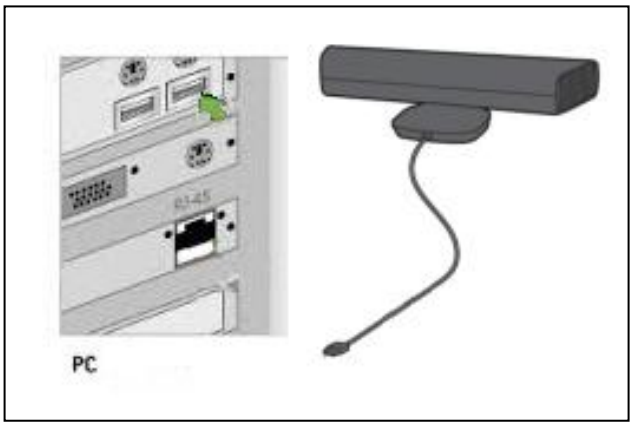
Introducción: Este manual le permitirá aprender a utilizar el RoboSurgery con Kinect, se debe realizar las siguientes tareas para el perfecto funcionamiento de la aplicación. **Nota**: Se debe haber instalado con anterioridad todo lo correspondiente al programa ver Anexo A. Para manipular la interface debe haber como mínimo dos personas usuario y asistente.

1. Conectar el dispositivo Kinect al ordenador y esperar a que sea reconocido y esté listo para funcionar.

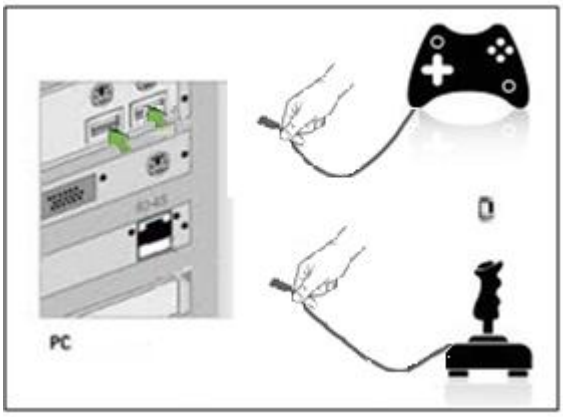
Kinect Xbox 360:



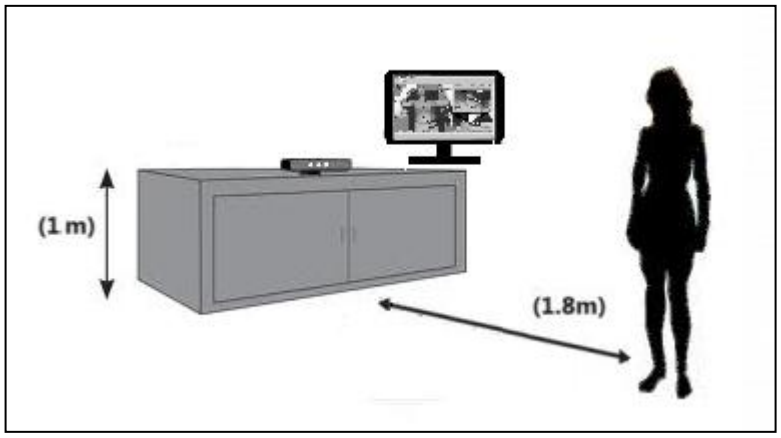
Kinect para Windows:



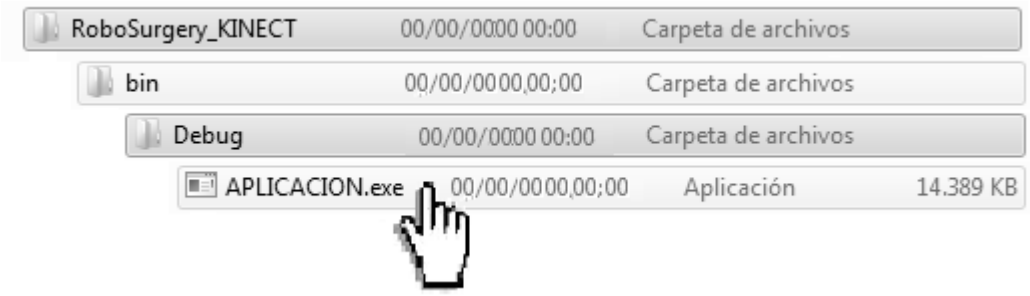
2. Conectar el joystick o mando deseado al ordenador.



3. Usuario. La persona que vaya a manipular la herramienta debe estar ubicada a 1.8m del dispositivo Kinect y tener la pantalla del ordenador lo más cerca posible sin bloquear la vista del Kinect. El Kinect debe estar ubicado a una altura de 1m.



4. Asistente. Acceder a la carpeta con el nombre “RoboSurgery_KINECT” donde se cargó el programa, ingresar en la carpeta “bin” y buscar en “Debug” el ejecutable con la extensión “.exe” y dar clic. Nota: Esto lo debe realizar un asistente.

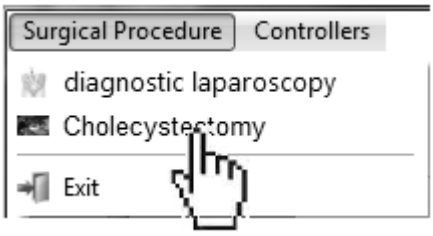


5. Usuario. En pantalla se observara la toma de datos de la mano izquierda con el Kinect, cuando comiencen aparecer números diferentes de cero se levanta la mano para acceder al sistema.

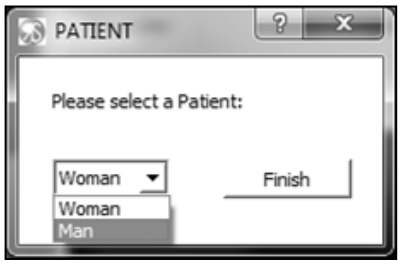
6. Usuario. Inmediatamente terminado el paso 5 posicionar las manos a la altura del pecho separadas aproximadamente 0.5m.

7. Asistente. Para la realización de la cirugía laparoscópica:

Ingresar a la pestaña “Surgical Procedure” ubicada en la parte superior de la pantalla y dar clic sobre “Cholecystectomy”.



Elegir el paciente de la cirugía y dar clic en “Finish”, esperar al que se carguen los elementos elegidos en la pantalla.



Ir a la pestaña “controllers” y dar clic en “Joystick”. Con estos pasos realizados se podrá efectuar la cirugía laparoscópica.

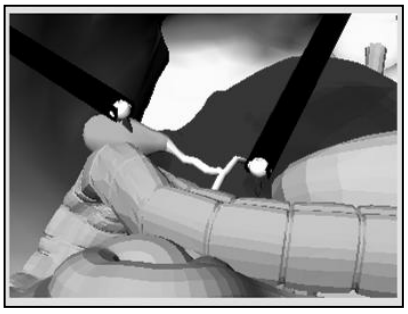
Etapas para la cirugía

Nota: Los botones nombrados en esta sección son para el control del Xbox.

Asistente. Coger vesícula: Se debe posicionar la cámara con ayuda del robot Hibou donde se pueda observar la vesícula biliar y sus arterias, oprimiendo el botón B seguidamente del *bumper* derecho se activará el robot y se podrá dirigir

con el *stick* analógico izquierdo. Se oprime el botón X y comenzará la tarea del usuario.

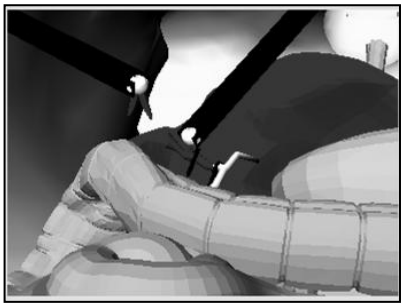
Usuario. Se lleva el robot izquierdo con el instrumento pinzas por defecto hasta tocar la vesícula para agarrarla (se logra con el movimiento natural de las manos).



Usuario. Para colocar las grapas el robot derecho debe tener el instrumento grapadora (Stapler), esta opción se cambia al levantar la rodilla derecha 0.06 m.

Usuario. Después de escoger la herramienta se debe dirigir el robot derecho a la parte inferior de la arteria hasta ver la grapa de color negro puesta, después ir a la parte superior y ver la segunda grapa.

Usuario. Con las grapas puestas se logra cortar el flujo de sangre y se procede al corte y extracción de la vesícula biliar, se cambia el instrumento del LapBot derecho a bisturí y cuando la arteria esté roja está lista para extraer.



Asistente. Ir a la pestaña inferior derecha y dar clic en exit.

IMPORTATE: El área de trabajo debe tener una buena iluminación, evitar la luz directa a los lentes del dispositivo.

ANEXO C

Ejemplo Kinect en Visual Studio 2010

Herramientas:

- Visual Studio 2010.
- Kinect for Windows SDK 1.6.
- Dispositivo Kinect.

Paso 1.

Abra Visual Studio y vaya a:

Archivo > Nuevo proyecto.

Elija: otros leguajes de programación > C++ > “Proyecto de consola Win32”, y en la parte inferior, se modifica los valores de nombre y ubicación.

Desactive la opción "encabezado pre compilado" y dar clic en finalizar. Cuando aparezcan las primeras líneas de código suministradas por Visual Studio 2010 compilamos.

Paso 2.

Enlazar con Kinect.

Se debe realizar las instrucciones explicadas en el anexo A en el punto número 6 (Integración de Kinect a Visual Studio).

Paso3.

Agregar las librerías:

```
#include "stdafx.h"
#include <iostream>
#include "Kinect.h"
```

Paso 4.

Agregar código a la clase principal para uso del dispositivo Kinect

a. Creamos el objeto de la clase Kinect y sus respectivas variables para obtención de datos

```
Kinect dispkin;                    //Objeto de la clase Kinect
double x = 0;
double y = 0;
double z = 0;
```

b. Reemplazamos

"int _tmain(int argc, _TCHAR argv[]) " por*

"using namespace std

Int main()"

c. Toma de datos

Las variables x, y, z son las encargadas de guardar los valores tomados por el Kinect y se realiza de la siguiente forma:

```
x = dispkin.obtener_ZHR();          //La variable x hace el llamado a la función
                                    //obtener_ZHR() para obtener los datos de
                                    la //mano derecha en movimientos de
                                    profundidad.

cout << "\n Mano derecha Z: \t" << x;   //Imprimir datos en consola de la variable
x.

y = dispkin.obtener_XHH();          //Obtenemos datos de movimientos
                                    horizontales //de la cabeza.

cout << "\n Cabeza      X: \t" << y;    //Imprimir datos en consola de la variable y

z = dispkin.obtener_RRK();          //Obtenemos datos de movimientos
                                    verticales de //la rodilla derecha.

cout << "\n Rodilla     Y: \t" << z;    //Imprimir datos en consola de la variable z
```

Paso 5.
Crear los archivos Kinect.h y Kinect.cpp

a. Crear Kinect.h

// Definir el nombre de la librería

```
#ifndef _KINECT_H_
#define _KINECT_H_
```

// Adicionar librerías par funcionamiento del Kinect.

```
#include <stdio.h>
#include <iostream>
#include <Windows.h>
#include <NuiApi.h>
```

//Nota: Las librerias Windows y NuiApi son las que utiliza el Kinect.
// Definir la clase Kinect

```
class Kinect
{
public:
NUI_SKELETON_FRAME ourframe;
```

//Definición de un objeto interno para guardar los datos obtenidos en un espacio de almacenamiento o de un cuerpo.

```
double YRK;                 //variables de la clase Kinect
double ZRK;
double RRK;
Kinect();                   //Constructor de la clase
~Kinect();                  //Destructor de la clase
double obtener_ZHR();       //Función para obtener los datos correspondientes a la
                                  //Mano derecha
double obtener_XHH();       //Función para obtener datos de la cabeza
double obtener_RRK();       //Función para obtener datos de la rodilla derecha
};
#endif                      //fin de definición de la clase con sus atributos
```

b. Crear Kinect.cpp

// Agregamos la librería Kinect.h

```
#include "Kinect.h"

// Definición del constructor.

Kinect::Kinect()
{
NuiInitialize(NUI_INITIALIZE_FLAG_USES_SKELETON);          // Inicialización de Kinect con
                                                           //banderas del esqueleto.
NUI_SKELETON_FRAME ourframe;
NuiSkeletonGetNextFrame(100, &ourframe);                   //Adquisición de datos del cuerpo
                                                           //cada 100 milisegundos y
                                                           //almacenados en ourframe.
}

Kinect::~Kinect()                                          // Definición del destructor.
{
NuiShutdown();                                             //Apagar dispositivo.
}

double Kinect::obtener_ZHR()                               //Función para obtener Z de la
                                                           //mano derecha.
{
NuiSkeletonGetNextFrame(100, &ourframe);
YRK = ourframe.SkeletonData[0].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].z;

                                                           // Obtenemos Z de la cabeza

return (YRK);
}

double Kinect::obtener_XHH()                               //Función para obtener x de la cabeza
{
NuiSkeletonGetNextFrame(100, &ourframe);
```

YRK=ourframe.SkeletonData[0].SkeletonPositions[NUI_SKELETON_POSITION_HEAD].x;

//Obtenemos X de la cabeza

return (YRK);
}
double Kinect::obtener_RRK() //Función para obtener Y de la rodilla derecha
{
NuiSkeletonGetNextFrame(100, &ourframe);
RRK = ourframe.SkeletonData[0].SkeletonPositions[NUI_SKELETON_POSITION_KNEE_RIGHT].y;

//Obtenemos Y de la rodilla derecha.

return (RRK);
}

A continuación se muestra una imagen con la aplicación realizada en funcionamiento